UNIVERZITET U TUZLI
PRIRODNO-MATEMATIČKI FAKULTET

Nermin Okičić Elvis Baraković

Matematičke osnove kompjuterske nauke

- Skripta -

Tuzla, 2022.

Sadržaj

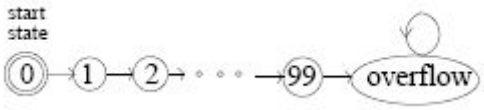
1

Mašine stanja

Mašine stanja su jednostavan matematički model za procese tipa korak-po-korak. Budući da većinu računarskih programa možemo shvatiti kao definisanje korak-po-korak izračunljivih procesa, nije iznenađenje da su se mašine stanja javile u računarskim naukama. Javljaju se i u drugim oblastima kao što su dizajniranje digitalnih kola i u modelovanju procesa vjerovatnoće.

Formalno, mašina stanja nije ništa drugo nego binarna relacija na skupu. Elementi skupa se nazivaju *stanja*, relacija se naziva *relacija tranzicije* dok se strelica u grafu relacije tranzicije naziva *tranzicija*. Relacija tranzicije se još naziva i *graf stanja* mašine a tranziciju iz stanja q u stanje r ćemo zapisati sa $q \to r$. Mašina stanja takođe mora imati i određeno početno stanje.

Primjer 1.1. *Jednostavan primjer mašine stanja je ograničeni brojač od 0 do 99 a koji se zatim vrti na sljedećem odbrojavanju. Dakle, stanja ove mašine su* $0, 1, 2, \ldots, 99, O$*, pri čemu smo sa* O *(overflow) označili stanje mašine stanja u kojem se brojač vrti. Početno stanje je 0, tranzicije su oblika* $n \to n+1$ *za* $0 \leqslant n < 99$*, zajedno sa tranzicijama* $99 \to O$ *i* $O \to O$*.*



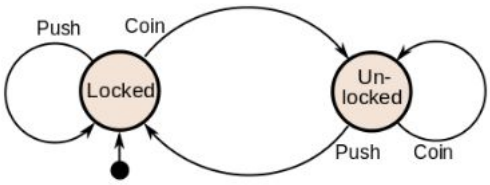
Ova mašina stanja i nije od neke koristi kada se jednom počne da vrti jer nikada neće promijeniti stanje. ∎

Naprimjer, mašine stanja za digitalna kola i algoritme podudaranja stringova obično imaju konačan broj stanja ali recimo mašine koje modeluju neprekidna izračunavanja obično imaju beskonačan broj stanja. U prethodnom primjeru smo imali konačan broj stanja. Međutim mogli smo u Primjeru (1.1) definisati neprekidni brojač, koji bi brojao neprekidno bez vrćenja na posljednjem stanju. Skup svih stanja bi u tom slučaju bio skup svih nenegativnih cijelih brojeva, što je beskonačan skup, te bi crtanje dijagrama bilo nemoguće.

Mašine stanja su obično definisane sa oznakama na stanjima i/ili tranzicijama da bi naznačili stvari kao što su vrijednosti ulaza i izlaza, cijene, kapacitete ili vjerovatnoće. Naše mašina stanja ne uključuju takve oznake jer nisu bitne za nas u ovom trenutku. Mi imenujemo stanja da bismo mogli o njima razgovarati, ali imena nisu dijelovi mašine stanja.

Primjer 1.2. *Primjer jednostavnog mehanizma koji se može modelovati mašinom stanja jeste rampa sa tri rotirajuće ruke u visini struka a koja se koristi za kontrolu pristupa autobuskim stanicama, parkovima i slično. U početku su ruke zaključane i*

blokiraju ulaz time sprečavajući prolaz. Ubacivanjem novčića u rampu otključavamo ruke što omogućava jednoj osobi da prođe. Nakon što jedna osoba prođe, ruke se zaključavaju dok neko ponovo ne ubaci novčić.



*Posmatrajući ovaj proces kao mašinu stanja, imamo dva moguća stanja (*otključano *i* zaključano*) i postoje dvije radnje koje mogu da utiču na stanja. To su ubacivanje novčića i guranje ruku na rampi. U zaključanom stanju, guranje ruku rampe ne dovodi do promjene stanja. Bez obira koliko puta gurnuli ruke rampe, stanje se neće promijeniti. Ubacivanje novčića u rampu dovodi do promjene stanja iz stanja zaključano u stanje otključano. U otključanom stanju, ubacivanje novčića ma koliko puta, ne dovodi do promjene stanja. Međutim, guranjem ruke rampe dolazi do promjene stanja u zaključano stanje.* ∎

1.1 Princip invarijantnosti

U ovoj sekciji predstavljamo *Princip invarijantnosti* što je verzija indukcije skrojena posebno za dokazivanje osobina mašina stanja. Jedna od najvažnijih primjena indukcije u računarskim naukama je dokazivanje jedne ili više željenih osobina koje nastavljaju da vrijede u svakom koraku u procesu. Osobina koja je sačuvana kroz cijeli niz operacija ili koraka se naziva *sačuvana invarijanta.*

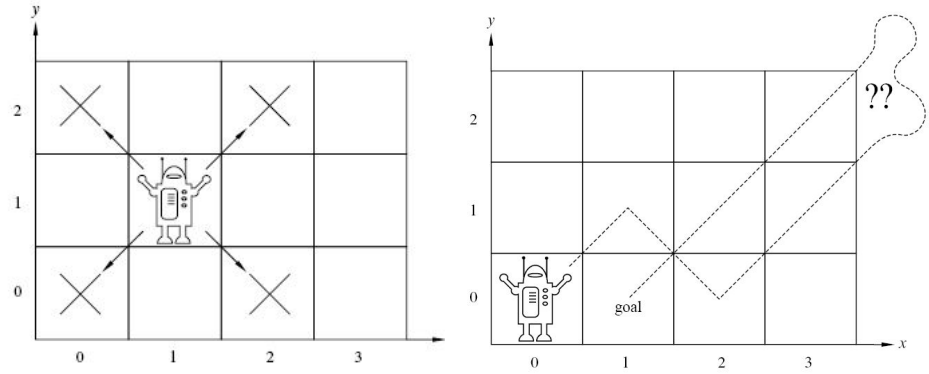
Princip invarijantnosti ćemo demonstrirati kroz primjer kretanja robota.

Pretpostavimo da imamo robota koji se kreće u ravni po dvodimenzionalnoj koordinatnoj mreži sa cjelobrojnim koordinatama. Pretposatvimo da robot počinje kretanje u koordinatnom početku i da se kreće samo dijagonalno od tačke do tačke. Stanje robota, u bilo kojem trenutku, može biti opisano sa uređenim parom cjelobrojnih koordinata (x, y) njegove trenutne pozicije. Početno stanje je stanje $(0, 0)$. Preciznije, tranzicije robota su opisane sa

$$\{(m, n) \to (m \pm 1, n \pm 1),\ m, n \in \mathbb{Z}\}.$$

Poslije prvog koraka robot može biti u četiri moguća stanja $(1, 1)$, $(-1, 1)$, $(1, -1)$ ili $(-1, -1)$. Poslije drugog koraka postoji devet novih mogućih stanja za robota (uključujući i stanje $(0, 0)$).

Postavlja se pitanje: može li robot ikada dostići poziciju $(1, 0)$?



Ukoliko analiziramo kretanje robota, možemo primjetiti da robot može dostići jedino pozicije (m, n) za koje je $m + n$ paran broj, što podrazumijeva da robot nikad ne može dostići poziciju $(1, 0)$. Ovo slijedi iz činjenice da je parnost sume koordinata pozicije robota osobina koja je sačuvana kroz kretanje. Da bi naglasili dio gdje koristimo indukciju, definišimo osobinu parnosti sume koordinata stanja sa

$$ParnaSuma((m, n)) := m + n \; je \; parno.$$

Lema 1.1.1. *Za bilo koju tranziciju $q \to r$ robota koji se dijagonalno kreće vrijedi da ako je $ParnaSuma(q)$ onda vrijedi $ParnaSuma(r)$.*

Ovo vrijedi na osnovu definicije tranzicija robota $(m, n) \to (m \pm 1, n \pm 1)$. Poslije tranzicije, suma koordinata se mijenja za $(\pm 1)+(\pm 1)$ što je 0 ili ± 2. Naravno, dodavajući 0 ili ± 2 parnom broju daje parni broj. Dakle, trivijalnom indukcijom na broj koraka sada možemo dokazati sljedeću teoremu.

Teorem 1.1.2. *Suma koordinata bilo kojeg stanja koje može dostići robot koji se dijagonalno kreće je parna.*

Dokaz. Induktivna hipoteza je data sa

$$P(n) := \text{ako je stanje } q \text{ moguće dostići u } n \text{ kretanja, tada je } ParnaSuma(q).$$

Dakle, $P(0)$ je tačno jer jedino stanje koje je moguće dostići u 0 koraka je stanje $(0, 0)$ a $0 + 0$ je paran broj.

Pretpostavimo da je $P(n)$ tačno i neka je r bilo koje stanje koje je moguće dostići u $n+1$ koraka. Trebamo pokazati da vrijedi $ParnaSuma(r)$. Budući da je stanje r moguće dostići u $n + 1$ koraka, tada mora postojati stanje q koje je moguće dostići u n koraka. Kako smo pretpostavili da je $P(n)$ tačno, tada vrijedi $ParnaSuma(q)$ ali tada na osnovu Leme 1.1.1 vrijedi $ParnaSuma(r)$. Ovim smo pokazali da iz $P(n)$ slijedi $P(n + 1)$.

Dakle, za svako $n \geqslant 0$ vrijedi da ako je stanje q moguće dostići tada vrijedi $ParnaSuma(q)$, što znači da za svako stanje koje robot može dostići ima osobinu parnosti sume koordinata stanja. □

Na osnovu ovog teorema zaključujemo da robot nikada ne može dostići stanje $(1, 0)$, jer $1 + 0$ nije paran broj.

Korištenje invarijate parne sume da bi razumjeli opisano kretanje robota je jednostavan primjer primjene metode koja se naziva *Princip invarijantnosti*. Princip ustvari pokazuje kako se indukcija na broj koraka koji su potrebni da se dostigne neko stanje primjenjuje na invarijante.

Izvršenje mašine stanja opisuje mogući niz koraka koje mašina može uzeti.

Definicija 1.1.1. *Izvršenje mašine stanja je niz stanja (moguće i beskonačno) sa osobinama da počinje sa početnim stanjem i da ako su q i r dva uzastopna stanja u nizu, tada je $q \to r$.*

Stanje se naziva *dostupnim* ako se pojavljuje u nekom izvršenju.

Definicija 1.1.2. *Sačuvana invarijanta mašine stanja je predikat P na stanju, takav da kad god je $P(q)$ tačan za stanje q i ako je $q \to r$ za neko stanje r, tada je i $P(r)$ tačan.*

Princip invarijantnosti možemo iskazati sa:

Ako je sačuvana invarijanta mašine stanja tačna za početno stanje, tada je tačna za svako stanje koje se može dostići.

Princip invarijantnosti nije ništa drugo nego princip indukcije prilagođen i skrojen u odgovarajućoj formi za mašine stanja. Dokaz da je predikat tačan u početnom stanju nije ništa drugo nego baza indukcije a dokaz da je predikat sačuvana invarijanta odgovara induktivnom koraku.

1.1.1 Jedan zanimljiv primjer za mašine stanja

U filmu *Umri muški III* možemo naći jedan zanimljiv primjer vezan za mašine stanja. Naime, u jednoj sceni u filmu, glavni likovi Bruce Willis i Samuel L. Jackson trebaju da deaktiviraju bombu koju je postavio Simon Gruber. U jednoj fontani se nalaze dva prazna kanistera, jedan zapremine 3 litra a drugi zapremine 5 litara. Pomoću njih je, u roku od 5 minuta, potrebno natočiti tačno 4 litra vode u veći kanistar te ga postaviti na vagu što će zaustaviti tajmer i detonaciju bombe. Srećom, Bruce i Samuel su uspjeli riješiti problem na sljedeći način:

Korak 1 natočili su pun kanister od 5 litara (veći) vodom iz fontane,

Korak 2 natočili su pun kanister od 3 litra (manji) vodom iz kanistera od 5 litara (u većem kanisteru ostalo tačno 2 litra vode),

Korak 3 ispraznili su kanister od 3 litra,

Korak 4 natočili su u kanister od 3 litara tačno 2 litra vode iz većeg kanistera,

Korak 5 ponovo su natočili pun kanister od 5 litara vodom iz fontane,

Korak 6 manji kanister od 3 litra su dopunili vodom (1 litar) iz velikog kanistera, čime je u velikom kanisteru ostalo tačno 4 litra vode.

Ovaj scenario punjenja i pražnjenja kanistera može se modelovati sa mašinom stanja. Označimo sa v količinu vode u većem kanisteru i sa m količinu vode u manjem kanisteru. Sa kanisterima od 3 i 5 litara stanja su uređeni parovi realnih brojeva (v, m) takvih da je $0 \leqslant v \leqslant 5$ i $0 \leqslant m \leqslant 3$. Sada nećemo pretpostavljati da su brojevi v i m ustvari cijeli nenegativni brojevi, mada je i tu osobinu, u ovom slučaju moguće dokazati. Početno stanje je $(0, 0)$ jer su u početku oba kanistera prazna.

Budući da količina vode u kanisterima mora biti tačno poznata, posmatraćemo samo poteze u kojima nalijemo pune kanistere ili ih u potpunosti ispraznimo. Zbog toga, postoji nekoliko mogućih tranzicija:

1. tranzicija $(v, m) \to (v, 3)$ za $m < 3$, što predstavlja punjenje manjeg kanistera,
2. tranzicija $(v, m) \to (5, m)$ za $v < 5$, što predstavlja punjenje većeg kanistera,
3. tranzicija $(v, m) \to (v, 0)$ za $m > 0$, što predstavlja pražnjenje manjeg kanistera,
4. tranzicija $(v, m) \to (5, m)$ za $v > 0$, što predstavlja pražnjenje većeg kanistera,
5. tranzicija
$$(v, m) \to \begin{cases} (v + m, 0) & \text{za } v + m \leqslant 5, \\ (5, m - (5 - v)) & \text{inače.} \end{cases}$$
za $m > 0$, što predstavlja presipanje vode iz manjeg u veći kanister,
6. tranzicija
$$(v, m) \to \begin{cases} (0, v + m) & \text{za } v + m \leqslant 3, \\ (v - (3 - m), 3) & \text{inače.} \end{cases}$$
za $v > 0$, što predstavlja presipanje vode iz većeg u manji kanister.

Analizirajući moguća stanja u ovoj mašini stanja vidimo da je stanje $(4, 3)$ dostupno. Kao što vidimo, u ovom primjeru postoji više od jedne moguće tranzicije i ovakva vrsta mašine stanja se naziva *neodređena.* Mašine stanja u kojima, iz stanja u stanje, postoji samo jedna moguća tranzicija naziva se *određena* mašina stanja, kao u Primjeru (1.1).

Ovaj primjer Simon je mogao malo modifikovati u smislu da je veći kanister od 5 litara zamijenio sa kanisterom od 9 litara te postavio iste uslove i isti zadatak. Mašina stanja u tom slučaju bi imala isti opis i stanja opisano ranije uz zamjenu broja 5 brojem 9, gdje god se pojavljuje. I u tom slučaju mogli bi postaviti pitanje: da li je stanje $(4, m)$ dostupno?

Dokazaćemo da nije koristeći princip invarijantnosti. Definišimo sačuvanu invarijantu kao predikat $P((v, m))$ gdje su v i m nenegativni cijeli brojevi djeljivi sa 3.

Pretpostavimo da je $P(q)$ tačno za neko stanje $q = (v, m)$ i neka je $q \to r$. Moramo pokazati da u tom slučaju je i $P(r)$ tačno. Dokaz za to se dijeli u nekoliko slučajeva u ovisnosti koju tranziciju smo koristili.

Recimo, u slučaju punjenja manjeg kanistera imamo da je $r = (v, 3)$. Kako je $P(q) = P(v, m)$ po pretpostavci tačno, slijedi da su v i m nenegativni cijeli brojevi djeljivi sa 3. Odavdje možemo zaključiti da su onda v i 3 nenegativni cijeli brojevi djeljivi sa 3, jer smo za vrijednost m uzeli baš broj 3. To implicira ja da je $P(r) = P(v, 3)$ tačno.

U slučaju presipanja vode iz većeg u manji kanister postoje dva podslučaja. Prvo, ako nema dovoljno mjesta u malom kanisteru za svu vodu, a to je slučaj kada je $v + m > 3$, imamo da je $r = (v - (3 - m), 3)$. Kako su v i m nenegativni cijeli brojevi djeljivi sa 3

tada su i brojevi $v-(3-m)$ i 3 djeljivi sa tri, što implicira da je $P(r)$ tačno. Na sličan način dokazujemo i ostale slučajeve.

Sada, na osnovu Principa invarijantnosti zakljucujemo da svako dostupno stanje zadovoljava uslov P. Budući da ne postoji stanje u obliku $(4, m)$ koje zadovoljava uslov P, dokazali smo da za ovakvu modifikaciju problema, nije moguće riješiti problem i detonirati bombu.

Primijetiomo da stanje $(1, 0)$ koje zadovoljava $\neg P$ ima tranziciju u stanje $(0, 0)$. Odavdje zaključujemo da negacija sačuvane invarijante možda neće biti sačuvana invarijanta.

1.2 Parcijalna ispravnost i prekid

U teorijskoj kompjuterskoj nauci kažemo da je algoritam ispravan u skladu sa specifikacijom ako se ponaša u skladu kako je specificiran. Najbolje je istražena funkcionalna ispravnost koja se odnosi na ulazn-izlaz ponšanje algoritma, tj. za svaki ulaz algoritam proizvodi izlaz zadovoljavajuci specifikaciju.

Parcijalna ispravnost zahtijeva da ako je vracen odgovor tada je ispravan i ona se razlikuje od potpune ispravnosti koja dodatno zahtijeva da se algoritam završava. Znaci, da bi dokazali potpunu ispravnost, potrebno je dokazati parcijalnu ispravnost i završavanje algoritma.

Drugim rijecima, za verifikaciju programa zahtijevaju se dvije osobine. Prva osobina se naziva *parcijalna ispravnost*. Kao što smo pjasnili, to je osobina koja kaže da finalni rezultati procesa (ako postoje) moraju zadovoljavati zahtjeve sistema. Riječ *parcijalna* dolazi iz posmatranja procesa koji se može ne završiti kao izračunavanje parcijalne relacije. Parcijalna ispravnost znači da kada postoji rezultat, on je tačan, ali proces ne mora uvijek da proizvede rezultat, jer možda recimo zapadne u petlju.Druga osobina ispravnosti, koju nazivamo prekid, znači da proces uvijek proizvede neku konačnu vrijednost.

Naprimjer, uzastopnim pretraživanjem cijelih brojeva $1, 2, 3, \ldots$ kako bismo provjerili možemo li pronaći primjer neke pojave. Recimo da tražimo neparan savršen broj prilično je jednostavno napisati djelomično ispravan program (pomoću dugog dijeljenja s dva za provjeru n kao savršen ili ne). Ali reći da je ovaj program potpuno tačan značilo bi utvrditi nešto što trenutno nije poznato u teoriji brojeva. Parcijalno ispravan program u C-u izgleda:

```
// vraca sumu djelilaca od n
static int divisorSum(int n) {
   int i, sum = 0;
   for (i=1; i<n; ++i)
      if (n % i == 0)
         sum += i;
   return sum;
}
//vraca najmanji savršen neparan broj
int leastPerfectNumber(void) {
   int n;
   for (n=1; ; n+=2)
      if (n == divisorSum(n))
         return n;
```

}

Parcijalna ispravnost obično se može dokazati koristeći princip invarijantnosti dok se prekid može obično dokazati koristeći dobro ureden princip. Ovo ćemo ilustrovati na sljedećem primjeru.

1.2.1 Brzo stepenovanje

Najjednostavniji način za izračunavanje n-tog stepena realnog broja a jeste da broj a pomnožimo samim sobom $n-1$ puta. Ovaj način izračunavanja zahtijeva veliki broj množenja te se zbog toga pronašla tehnika koja zahtijeva manji broj množenja od uobičajenog koja je nazvana *brzo stepenovanje*.

U ovom dijelu mi se nećemo baviti samim algoritmom brzog stepenovanja nego ćemo konstruisati mašinu stanja za ovaj problem.

PROGRAM BRZOG STEPENOVANJA

Za zadane $a \in \mathbb{R}$ i $n \in \mathbb{N}$ na početku odaberimo da su x, y, z jednaki $a, 1, n$ redom i ponavljamo sljedeći niz koraka dok se proces ne završi:

- ako je $z = 0$ vrati y i završi proces;
- r :=ostatak $(z, 2)$;
- z :=količnik $(z, 2)$;
- ako je $r = 1$ onda je $y := xy$;
- $x := x^2$.

Ovaj proces će se uvijek završiti i daće rezultat $y = a^n$, vidi [3]. Ovaj problem ćemo modelovati sa mašinom stanja na sljedeći način:

- stanja su uređene trojke brojeva $(x, y, z) \in \mathbb{R} \times \mathbb{R} \times \mathbb{N}$;
- početno stanje je uređena trojska $(a, 1, n)$;
- tranzicije su defisane sa

$$(x, y, z) \to \begin{cases} (x^2, y, \text{količnik}(z, 2)), & \text{ako je } z \text{ različit od nule i paran;} \\ (x^2, xy, \text{količnik}(z, 2)), & \text{ako je } z \text{ različit od nule i neparan;} \end{cases}$$

- sačuvana invarijanta $P((x, y, z))$ je $z \in \mathbb{N}$ i $yx^z = a^n$.

Da bismo dokazali da je P sačuvana invarijanta, pretpostavimo da vrijedi $P((x, y, z))$ i neka je $(x, y, z) \to (x_1, y_1, z_1)$ jedna od tranzicija. Trebamo dokazati da vrijedi i $P((x_1, y_1, z_1))$. Budući da je tranzicija iz (x, y, z), tada je $z \neq 0$ i budući da je $z \in \mathbb{N}$, posmatramo dva slučaja.

Ako je z paran broj, tada je $x_1 = x^2$, $y_1 = y$ i $z_1 = \frac{z}{2}$. Tada je $z_1 \in \mathbb{N}$ i

$$y_1 x_1^{z_1} = y(x^2)^{\frac{z}{2}} = yx^z = a^n.$$

Ako je z neparan broj, tada je $x_1 = x^2$, $y_1 = xy$ i $z_1 = \frac{z-1}{2}$. Tada je $z_1 \in \mathbb{N}$ i

$$y_1 x_1^{z_1} = xy(x^2)^{\frac{z-1}{2}} = yx^z = a^n.$$

Dakle, P je doista sačuvana invarijanta.

Sada je lako dokazati parcijalnu ispravnost: ako se program za brzo stepenovanje završava, završava se sa a^n pridruženo y. Ovo vrijedi jer je $1 \cdot a^n = a^n$, što znači da početno stanje $(a, 1, b)$ zadovoljava P. Prema principu invarijante, P vrijedi za sva moguća stanja koja se mogu dostići. Ali program staje samo kad je $z = 0$. Ako je završno stanje $(x, y, 0)$ stanje koje se može dostići, tada je $y = yx^0 = a^n$, kao što je i zahtijevano.

Budući da je program parcijalno ispravan, možemo postaviti pitanje brzine programa? Odgovor na to pitanje je da je broj množenja da bi se izračunalo a^n grubo rečeno dužina binarne reprezentacije broja n. Odnosno ovaj program, ugrubo rečeno, koristi $\log n$ množenja[1], poredeći sa pristupom množenja broja a samim sobom $n-1$ puta. Tačnije, ova tehnika zahtijeva najviše $2(\lceil \log n \rceil + 1)$ množenja za izračunavanje stepena a^n, $(n > 1)$. Razlog za to je što je u početku z jednako n i biva barem prepolovljen u svakoj tranziciji. Zbog toga ne može biti prepolovoljen više od $\lceil \log n \rceil + 1$ puta prije nego postane 0 i time uslovi završetak programa. Budući da svaka tranzicija uključuje najviše dva množenja, ukupan broj množenja prije nego z dosegne vrijednost 0 je $2(\lceil \log n \rceil + 1)$.

1.2.2 Izvedene varijable

Prethodni dokaz za prekid programa uključivao je pronalaženje nenegativne cjelobrojne mjere pridružene stanju. Ovu mjeru možemo nazvati "veličina" stanja. Poslije toga smo pokazali da se veličina stanja smanjuje sa svakom tranzicijom. Po principu dobre uređenosti, veličina se ne može povećavati do u beskonačno, pa prema tome kada je dosegnuto stanje sa minimalnom veličinom, tranzicije više nisu moguće, što znači da je proces okončan.

Tehnika pridruživanja vrijednosti stanjima (ne nužno nenegativnih cjelobrojnih vrijednosti i ne nužno opadajućih po tranzicijama) je veoma korisna u analizi algoritama. U kontekstu procesa izračunavanja, takve vrijednosti dodjeljene stanjima se nazivaju *izvedene varijable.*

Naprimjer, u Sekciji 1.1.1 predstavili smo izvedenu varijablu f : stanje $\to \mathbb{R}$ za količinu vode u oba kanistera stavljajući $f((a, b)) := a + b$. Slično, u problemu kretanja robota, pozicija robota duž x ose bi bila izvedena sa x koordinatom, pri čemu je $x((i, j)) := i$.

Postoji nekolio stanadardnih osobina izvedenih varijabli koje su zgodne u analizi mašina stanja.

Definicija 1.2.1. *Izvedena varijabla f : stanja $\to \mathbb{R}$ je strogo opadajuća akko vrijedi*

$$q \to q' \Rightarrow f(q') < f(q)$$

a slabo opadajuća ako vrijedi

$$q \to q' \Rightarrow f(q') \leqslant f(q)$$

Na sličan način se definišu strogo/slabo rastuće izvedene varijable.

[1] U računarskim naukama $\log n$ se obično odnosi na logaritam po bazi 2.

Prekid procedure za brzo stepenovanje smo potvrdili time što je izvedena varijabla z imala nenegativnu cjelobrojnu vrijednost i bila je striktno opadajuća. Dokazivanje prekida možemo dokazati koristeći sljedeći teorem.

Teorem 1.2.1. *Ako je f izvedena varijabla mašine stanja strogo opadajuća sa vrijednostima u skupu $\mathbb{N}$, tada je dužina bilo kojeg izvršenja koje počinje u stanju q najviše $f(q)$.*

Drugim riječima vrijedi, *ako počnete odbrojavati od neke nenegativne cjelobrojne vrijednosti $f(q)$, tada to možete uraditi najviše $f(q)$ puta.*

Teorem 1.2.2. *Ako postoji striktno opadajuća izvedena varijabla čiji je rang dobro uređen skup, tada se svako izvršenje završava.*

Primjer 1.3. *Posmatrajmo robota koji se kreće po kvadrantu $\mathbb{N}^2$, tojest kvadrantu sa nenegativnim cjelobrojnim vrijednostima. Ako je robot na nekoj poziciji $(x, y) \neq (0, 0)$, robot se mora pomaknuti na jedan od sljedećih načina*

- *jedinični pomak na Zapad, tojest $(x, y) \to (x - 1, y)$ za $x > 0$, ili*
- *jedinični pomak na Jug kombinovan sa proizvoljnim skokom na Istok, tojest $(x, y) \to (z, y - 1)$ za $z \geqslant x$*

čime je omogućeno da robot ne napušta kvadrant. ∎

Lema 1.2.3. *Robot će se uvijek zaglaviti u koordinatnom početku $(0, 0)$.*

Definišimo izvedenu varijablu f koja preslikava stanja kretanja robota u dobro uređen skup $\mathbb{N} + \mathbb{F}$ (skup $\mathbb{F}$ je skup svih razlomaka koji se mogu zapisati u obliku $\frac{n}{n+1}$, $(n = 0, 1, 2, \ldots)$. Ustvari, definišimo preslikavanje $f : \mathbb{N}^2 \to \mathbb{N} + \mathbb{F}$ sa

$$f(x, y) := y + \frac{x}{x + 1}.$$

Lako je provjeriti da ako je $(x, y) \to (x', y')$ pokret robota, tada je $f(x, y) < f(x', y')$. dakle, f je striogo opadajuća izvedena varijabla, pa na osnovu Teoreme 1.2.2 zaključujemo da će se robot uvijek zaglaviti u koordinatnom početku.

1.3 Problem stabilnog braka

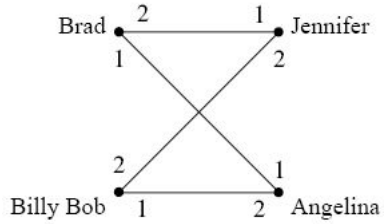
Još jedan interesantan problem ćemo modelovat i riješiti koristeći mašine stanja. Naime, radi se o tzv. *problemu stabilnog braka* koji predstavlja problem iskazan u ovom obliku radi lakšeg shvatanja problema.

Pretpostavimo da imamo skupinu od jednakog broja muškaraca i žena i gdje svaka osoba ima preferencijalnu listu osoba suprotnog spola sa kojima bi voljeli stupiti u brak. Dakle, svaki muškarac ima preferencijalnu listu svake žene za kojom bi volio sklopiti brak, i obrnuto. Preferenecije ne moraju biti simetrične, svaki muškarac mora oženiti tačno jednu djevojku, i obrnuto.

Naš cilj je naći "perfektno podudaranje" bračnih parova u brakovima koji su *stabilni* u smislu da ne postoji par muškarca i žene koji preferiraju jedno drugo više nego svoje

bračne drugove. Dakle, par (m, z) muškarca i žene nazivamo *nestabilnim* ako oni nisu u braku jedno sa drugim i koji više vole jedno drugo nego svoje partnere sa kojima su u braku. U traženju perefktnog podudaranja izbjegavat ćemo nestabilne parove jer prijete stabilnosti brakova. S druge strane, ako nema nestabilnih parova, tada za bilo kojeg muškarca i bilo koju ženu koji nisu vjenčani jedno za drugo, najmanje jedno voli svog bračnog druga više od nekog drugog.

Primjer 1.4 (Primjer nestabilnog para)**.** *Posmatrajmo dva bračna para, prvi čine Brad i Jennifer dok drugi par čine Billy Bob i Angelina. Neka je svako od njih, sa druge osobe suprotnog spola iskazao preferencije kao na slici*



Kao što vidimo, Angelina i Brad preferišu jedno drugo više nego svoje bračne partnere, što njihove brakove dovodi u opasnost. Par Angelina i Brad predstavljaju nestabilan par. ■

Definicija 1.3.1. *Stabilno podudaranje je podudaranje bez nestabilnih parova.*

Postavlja se pitanje, da li uvažavajući preference svih muškaraca i svih žena, možemo naći stabilno podudaranje? Ispostavlja se da uvijek postoji stabilno podudaranje, vidi [3].

Primjer 1.5 (Primjer stabilnog braka)**.** *Ukoliko bi u Primjeru* **??** *dopustili da se Brad i Angelina vjenčaju, tada bi se po pravilima vjenčali Billy Bob i Jennfier i to bi predstavljalo stabilno podudaranje.* ■

Procedura za nalaženje stabilnog podudaranja (između osoba suprotnog spola!) se može opisati kao lako pamtljiv ritual koji se odvija u nekoliko dana na sljedeći način:

Jutro Svaki muškarac (prosac) dolazi pod balkon žene koja je na vrhu njegove liste preferencija i prosi je.

Podne Svaka žena, koja ima jednog ili više prosaca, svom omiljenom proscu kaže "Da. Mogli bi se vjenčati. Budi negdje tu!" a svim ostalim kaže "Ne. Nikada se neću udati za tebe. Zbogom!"

Veče Muškarac koji je odbijen od neke žene križa ime te žene na svojoj listi.

U ovom slučaju, uslov završetka (prekida) se može iskazat sa: kada dođe dan kada svaka žena ima najviše jednog prosca, ritual se završava tako što se žena udaje za svog prosca, ako ga ima.

Dokazat ćemo da ovaj ritual ispunjava uslove:

- ritual eventualno postiže uslov prekida,
- ritual omogućava da su na kraju svi parovi u braku i
- svi konačni brakovi su stabilni.

U tom cilju ćemo opisani ritual tretirati kao mašinu stanja. Stanje na početku svakog dana je određeno odlukom svakog muškarca koju ženu će zaprositi tog dana. To je žena na vrhu njegove liste poslije križanja imena žena koje su ga odbile ranijih dana.

Lako je vidjeti da će u ovom ritualu doći dan kada će se svi vjenčati. Svakog dana u kojem se ritual nije završio, najmanje jedan muškarac križa ženu sa svoje liste. Ako je ritual počeo sa n žena i n muškaraca, tada svaka od listi muškaraca, kojih ima n, u početku ima n žena na njoj, što u konačnici daje n^2 ulaznih podataka. Kako se žene više ne dodaju na listu, broj žena na listi se svakim danom smanjuje kako se ritual nastavlja. Zbog toga se ritual završava za najviše n^2 dana. Dakle, proces se završava pa će doći dan kada će se svi vjenčati.

Dokažimo da ovaj ritual stavlja svakog u stabilan brak. U tu svrhu primijetimo važnu činjenicu: ako nekog jutra žena ima bilo kojeg prosca, tada će je njen omiljeni prosac dolaziti sljedećeg jutra jer se njegova lista preferencija ne mijenja. Zbog toga je sigurna da će omiljenog današnjeg prosca imati i sutra među proscima. To znači da će ona biti u mogućnosti izabrati omiljenog udvarača sutra koji je u najmanju ruku jednako poželjan kao današnji favorit. Zbog toga, iz dana u dan, njen udvarač može biti isti ili samo bolji i bolji, nikada gori, što nam može zvučati kao invarijanta.

Definišimo predikat P sa

"za svaku ženu z i muškarca m, ako je z prekrižena sa liste muškarca m, tada z ima prosca kojeg ona preferira u odnosu na m."

Lema 1.3.1. *Predikat P je sačuvana invarijanta u ovom ritualu.*

Dokaz. Žena z bude prekrižena sa liste muškarca m samo kada žena z ima prosca kojeg preferira više od muškarca m. Nakon toga, njen omiljeni prosac se ne mijenja dok joj ne naiđe bolji. Zbog toga, ako je njen prosac bolji do muškarca m, tada će svaki novi bolji prosac to takođe biti. □

Teorem 1.3.2. *Na kraju ovog rituala svi su vjenčani.*

Dokaz. Pretpostavimo suprotno, tj. da je u zadnjem danu jedan muškarac, nazovimo ga Nedim, ostao neoženjen. To znači da se Elvis ne može doći nikom udvarati, njegova lista je prazna. Znači da je svaka žena na njegovoj listi prekrižena, a kako je predikat P tačan, svaka žena ima udvarača kojeg preferira više od Nedima. U suštini, svaka žena ima nekog udvarača, a budući da je zadnji dan, one imaju samo jednog udvarača, a to je onaj za koga će se vjenčati. Ali, budući da je broj žena i broj muškaraca isti, i kako su sve žene udate, tada su i svi muskarci oženjeni. To je u suprotnosti sa pretpostavkom da Elvis nije oženjen. □

Teorem 1.3.3. *Ovaj ritual proizvodi stabilno podudaranje.*

Dokaz. Neka su Brad i Jen par koji nisu vjenčani zadnjeg dana rituala. Dokazat ćemo da Brad i Jen nisu nestašan par, tojest da su zadnjeg dana rituala svi brakovi stabilni. Imamo dva slučaja.

(a) Jen nije na Bradovoj listi do kraja. Tada, po invarijanti P, znamo da Jen ima udvarača (a time i muža) kojeg preferira više od Brada. Zbog toga nece pobjeći sa bradom, tojest Brad i Jen ne mogu biti nestabilan par.

(b) Jen je na Bradovoj listi. Budući da Brad bira ženu za udvaranje idući kroz njegovu listu, njegova žena mora biti na višem mjestu od Jen. Zbog toga neće pobjeći sa Jen i ponovo Brad i Jen ne mogu biti nestabilan par.

□

Postavlja se pitanje, ko je favorizovaniji u ovom ritualu? Muškarci ili žene. Čini se da žene imaju veću moć, svakog dana izaberu omiljenog udvarača, a ostale odbace i kako se ritual nastavlja, njen udvarač može biti samo bolji. S druge strane, muškarac nastavlja da se udvara ženi koju najviše preferira sve dok ne dođe momenat kada je mora prekrižiti sa liste. Sa njegove tačke gledišta, žena kojoj se udvara može biti samo gora i gora, u odnosu na njegovu listu preferencija. Čini se da su žene u povoljnijem položaju.

Ali nije tako!

Budući da ovaj ritual proizvodi jedno stabilno podudaranje, stabilna podudaranja ne moraju biti jedinstvena. Naprimjer, ako obrnemo uloge muškaraca i žena, često ćemo dobit drugačija stabilna podudaranja među njima. Tako muškarac može imati različite žene u različitim skupovima stabilnih brakova. U nekim slučajevima, muškarac može stabilno oženiti svaku ženu, ali u većini slučajeva, postoje žene koje ne mogu biti u braku sa muškarcima u nekom stabilnom podudaranju. U Primjeru 1.4, Jennifer ne može biti žena Bradu u bilo kojem stabilnom podadaranju jer ako nju oženi Brad, tada bi Brad i Angelina činili nestašan par. Za Jennifer nije izvodivo da bude u stabilnom braku sa Bradom.

Definicija 1.3.2. *Za dati skup prefernecija muškaraca i žena, jedna osoba je izvodljivi supružnik drugome kada postoji stabilno podudaranje u kojem su ove dvije osobe u braku.*

Definicija 1.3.3. *Neka je Q predikat: za svaku ženu z i muškarca m, ako je z prekrižena sa liste od m, tada z nije izvodljiv supružnik za m.*

Lema 1.3.4. *Predikat Q je sačuvana invarijanta za ritual podudaranja.*

Definicija 1.3.4. *Za dati skup preferenecija muškaraca i žena, optimalni supružnik nekoj osobi je njegov izvodljivi supružnik kojeg najviše preferira. Pesimalni supružnik nekoj osobi je njegov izvodljivi supružnik kojeg najmanje preferira.*

Svi imaju optimalnog i pesimalnog supružnika, budući da znamo da postoji najmanje jedano stabilno podudaranje.

Teorem 1.3.5. *Ritual podudaranja vjenčava svakog muškarca sa njegovom optimalnom ženom.*

Dokaz. Ako je Bob vjenčan sa Alice u zadnjem danu rituala, tada su svi iznad na listama od Boba i Alice prekriženi, pa prema osobini Q, sve prekrižene žene nisu bile izvodljive za Boba. Znači, Alice je Bobu najviše rangirani izvodljivi supružnik, tojest njegova optimalna žena. □

Lema 1.3.6. *Za dati skup preferenecija muškaraca i žena, svako je pesimalni supružnik svom optimalnom supružniku.*

Lema 1.3.7. *Ritual podudaranja vjenčava svaku ženu sa njenim pesimalnim mužem.*

2

Rekurzije

U matematici se često funkcije pozivaju na same sebe. Naprimjer, funkcija faktorijel $f(n) = n!$ za $n \in \mathbb{N}$ je definisana sa

$$f(n) = \begin{cases} 1 & \text{za} \quad n \leq 1, \\ n \cdot f(n-1) & \text{za} \quad n > 1. \end{cases}$$

Ovakva definicija nam govorida je vrijednost funkcije $f(n)$ jednaka 1 za n manje ili jednako od 1, ali kada je n veće od 1, funkcija $f(n)$ je definisana rekurzivno, tojest poziva samu sebe. Naprimjer, vrijedi $f(2) = 2 \cdot f(1) = 2 \cdot 1 = 2$ a onda $f(3) = 3 \cdot f(2) = 3 \cdot 2 = 6$, itd.

Drugi primjer rekurzivno zadane funkcije je Fibonačijev niz brojeva, koji je definisan sa

$$f(0) = 0;\ f(1) = 1;\ f(n) = f(n-1) + f(n-2) \text{ za } n > 2.$$

Slobodno možemo reći da se funkcija $f(n)$ definiše induktivno. Zato postoje bazni slučajevi koje treba posebno definisati i rekurzija se primjenjuje samo za n veće od baznih slučajeva.

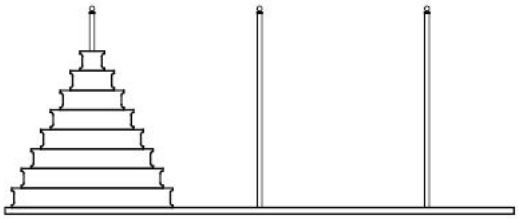
Za niz $(a_0, a_1, a_2, \ldots, a_n, \ldots)$, jednačina koja povezuje a_n za bilo koje n sa jednim ili više članova a_i, $(i < n)$ je rekurentna jednačina neke numeričke funkcije. Rekurentne jednačine se također nazivaju i *diferentne jednačine.*

Primjer 2.1. *Za niz* $(2^0, 2^1, 2^2, \ldots, 2^n, \ldots)$*, opšti član je dat sa* $a_n = 2^n$ *za* $n \geq 0$*. Ova numerička funkcija se može izraziti i na drugi način. Budući da je vrijednost od* a_n *dva puta veća od vrijednosti od* a_{n-1} *za svako n, pa kada znamo vrijednost od* a_{n-1}*, znamo i od* a_n*. Dalje, vrijednost od* a_{n-1} *dva puta veća od vrijednosti od* a_{n-2} *za svako n koja je opet dva puta veća od vrijednosti* a_{n-3}*, itd. Na kraju, dođemo do vrijednosti* a_0*, čija je vrijednost poznata i jednaka je* 1*. Na osnovu svega, možemo pisati* $a_n = 2 \cdot a_{n-1}$ *za* $n \geq 1$*.*

Jasno je da iz rekurzije možemo računajući korak po korak izračunati vrijednost a_n pomoću $a_{n-1}, a_{n-2}, \ldots$ i također izračunati vrijednost a_{n+1} pomoću $a_n, a_{n-1}, \ldots$, koristeći bazne slučajeve. Odavdje zaključujemo da numerička funkcija može biti opisana pomoću rekurzije zajedno sa baznim slučajevima. *Numerička funkcija se također smatra rješenjem rekurentne relacije (rekurentne jednačine).*

2.1 Hanojske kule

Prema legendi, u Hanoju postoji hram sa 3 štapa i 64 zlatna diska različitih veličina. Svaki disk ima rupu u sredini tako da se može staviti na disk. U davnoj prošlosti, svi diskovi su bili poredani na prvom štapu, najveći se nalazio na dnu a najmanji na vrhu štapa, kao što je prikazano na slici

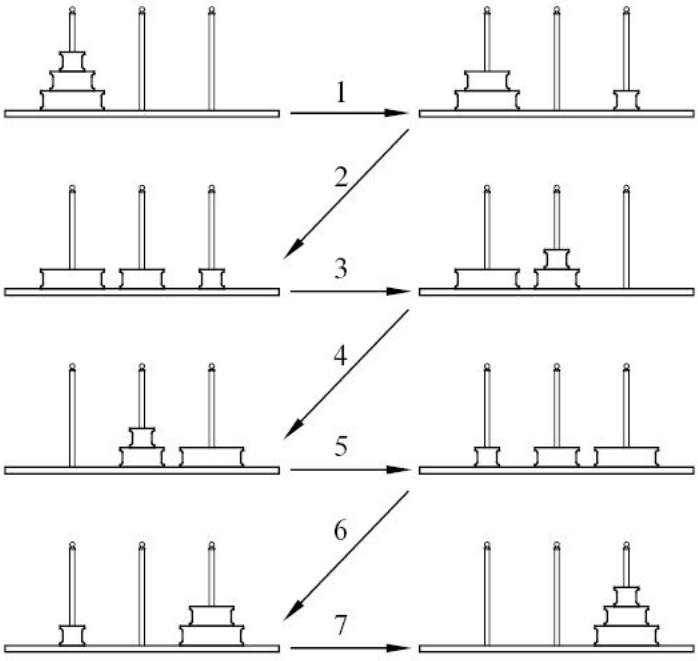


Monasi u hramu su bili zaduženi da diskove sa jednog štapa prebace na jedan od preostala dva štapa, ali uz određena pravila. Prvo pravilo je da se disk sa vrha jednog štapa može prebaciti na drugi štap i drugo pravilo je da veći disk nikad ne smijem biti iznad malog na bilo kojem štapu. Uzimanje svih poredanih diskova sa jednog štapa na drugi nije dozvoljeno.

Legenda također kaže da kada monasi u hramu završe posao da će to biti kraj svijeta u kojem postojimo. Postavljamo onda pitanja da li je problem moguće riješiti i ako je moguće, koliko vremena nam treba da ga riješiti? Za koliko će se onda desiti kraj svijeta u kojem postojimo?

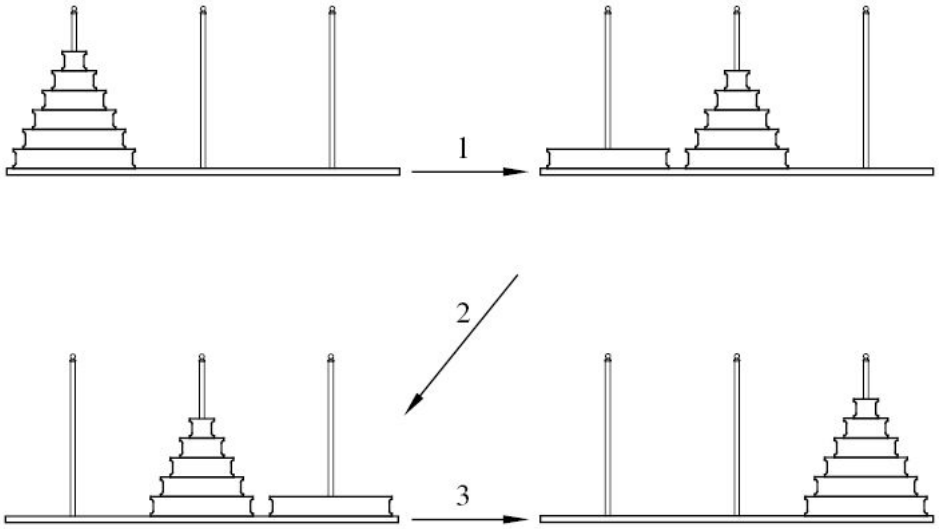
Ovaj problem je postavljen 1883. godine od strane francuskog matematičara Edouarda Lucasa.

Prvo, pojednostavimo problem posmatrajući tri zlatna diska, poredana po veličini na prvom štapu. Tada problem možemo riješiti u 7 poteza, kao što je prikazano na slici



Problem Hanojskih kula se može riješiti rekurzivno. Neka je T_n minimalni broj koraka u kojima možemo riješiti problem sa n diskova. Naprimjer, veoma lako možemo zaključiti da je $T_1 = 1$, $T_2 = 3$. Za tri diska smo eksperimentalno pokazali da je $T_3 \leq 7$. Sada

možemo generalizirati pristup za 3 diska na problem sa n disokova koristeći sljedeći algoritam. Rekurzivno rješenje se provodi u tri koraka, kao što je prikazano na slici



Dakle, koraci su:

Korak 1 prebaciti (sa vrha) $n-1$ disk sa prvog štapa na drugi štap, što možemo da uradimo u T_{n-1} koraka,

Korak 2 pomaknuti najveći disk sa prvog štapa na treći štap što možemo uraditi u jednom koraku i zatim

Korak 3 prebaciti $n-1$ disk sa drugog štapa na treći štap, što možemo da uradimo u T_{n-1} koraka.

Pomoću ovog algoritma, pokazuje se da minimalni broj koraka T_n je najviše $T_{n-1}+1+T_{n-1}=2T_{n-1}+1$. Zato je naprimjer $T_3 \leqslant 7$ i $T_4 \leqslant 15$. Nastavljajući ovaj proces možemo izračunati gornju granicu za T_{64}. Ovo nam daje odgovor na prvi dio pitanja da je problem moguće riješiti u konačnom broju koraka.

U ovom momentu ne možemo izračunati tačan broj koraka za pomijeranje svih 64 diskova ali možemo dati gornju granicu. U stvari nema boljeg algoritma a evo i zašto. U nekom koraku, monasi moraju da pomjere n-ti disk sa prvog na neki drugi štap. Da bi se ovo desilo, preostali $n-1$ diskova mora biti poredano na preostalom štapu. Preslaganje $n-1$ diskova na ovaj način zahtijevat će najmanje T_{n-1} koraka. Nakon što se pomakne najveći disk, najmanje još T_{n-1} koraka je potrebno da bi se $n-1$ diskova prebacilo na vrh štapa. Ovo nam pokazuje da je broj koraka najamanje $2T_{n-1}+1$. Budući da smo dali algoritam sa tačnim brojem koraka, možemo zaključiti da je rekurzija

$$T_1 = 1;$$
$$T_n = 2T_{n-1}+1, (n \geqslant 2),$$

rekurzija koja daje broj koraka da bi se završio problem Hanojskih kula. Odavdje dobijamo $T_2 = 3,\ T_3 = 7,\ T_4 = 15, \ldots$.

Izračunavanje T_{64} iz date rekurzije zahtijevao bi mnogo posla i zbog toga bi bilo dobro imati egzaktan izraz za T_n, čime bismo mogli brzo da izračunamo broj koraka za bilo koje n. Postoji nekoliko načina da se ova rekurzija riješi. Najlakši metod je pogoditi (ukoliko je moguće) rezultat te dokazati (verifikovati) njegovu tačnost, obično indukcijom. Ovaj metod se naziva *pogodi i verifikuj* ili *zamjena*.

Za početak pogađanja rezultata, posmatrajmo prvih nekoliko vrijednosti za T_n:

n	1	2	3	4	5	6
T_n	1	3	7	15	31	63

Odavdje naslućujemo da je $T_n = 2^n - 1$ i dokažimo indukcijom da je ovaj rezultat tačan.

Teorem 2.1.1. *Ako je*

$$T_1 = 1;$$
$$T_n = 2T_{n-1} + 1, (n \geqslant 2),$$

tada je $T_n = 2^n - 1$.

Dokaz. Dokaz izvodimo indukcijom po n. Neka je $P(n)$ tvrdnja da je $T_n = 2^n - 1$.

Tvrdnja $P(1)$ je tačna jer je $T_1 = 2^1 - 1 = 1$.

Pretpostavimo da je $T_n = 2^n - 1$ i dokažimo da je $T_{n+1} = 2^{n+1} - 1$ za $(n \geqslant 1)$. Imamo $T_{n+1} = 2T_n + 1 = 2(2^n - 1) + 1 = 2^{n+1} - 1$. dakle, tvrdnja vrijedi za svaki prirodan broj n. □

Dakle, vrijedi $T_{64} = 2^{64} - 1$ što je 18446744073709551615 koraka. Ukoliko je recimo monasima potrebna jedna sekunda za prebacivanje jednog diska, onda da bi riješili kompletan problem Hanojskih kula biće im potrebno $2^{64} - 1$ sekundi, što je otprilike $6 \cdot 10^{11}$ godina. Time smo dali odogovor na pitanje za koliko će se, prema legendi, završiti svijet kojeg poznajemo. Nije loše, zar ne?

U opštem slučaju, pogodi i verifikuj metod je dobar način da se problem riješi. Nedostatak je pogađanje rezultata, što često jeste slučaj. Postoji jedan alternativni metod za rješavanje pod nazivom *ubaci i pojednostavi* metod, koji ustvari koristi "sirovu snagu" za rješavanje problema. To, u slučaju problema Hanojskih kula, možemo prikazati sa sljedećih nekoliko koraka.

Prvi korak: ubaci i pojednostavi.

$$\begin{aligned} T_n &= 1 + 2T_{n-1} \\ &= 1 + 2(1 + 2T_{n-2}) \\ &= 1 + 2 + 4T_{n-2} \\ &= 1 + 2 + 4(1 + 2T_{n-3}) \\ &= 1 + 2 + 4 + 8T_{n-3} \\ &= 1 + 2 + 4 + 8(1 + 2T_{n-4}) \\ &= 1 + 2 + 4 + 8 + 16T_{n-4} \end{aligned}$$

U ovom koraku posebno treba voditi računa prilikom pojednostavljivanja, da se ne bi izgubio uzorak.

Drugi korak: prepoznavanje i verifikacija problema. U ovom koraku treba prepoznati uzorak za rekurentnu jednačinu nakon i koraka ubacivanja i pojednostavljivanja. Zatim treba verifikovati da je ovaj uzorak tačan nakon provođenja još jednog koraka ubacivanja i pojednostavljivanja.

U našem primjeru Hanojskih kula, uzrak je očigledan: T_n je uvijek suma uzastopnih stepena od 2 zajedno sa ranijim članom, tojest $T_n = 1 + 2 + 4 + \cdots + 2^{i-1} + 2^i T_{n-i}$. Uradimo još jednom korak ubacivanja i pojednostavljivanja pa ćemo dobiti

$$\begin{aligned} T_n &= 1 + 2 + 4 + \cdots + 2^{i-1} + 2^i(1 + 2T_{n-(i+1)}) = \\ &= 1 + 2 + 4 + \cdots + 2^{i-1} + 2^i + 2^{i+1} T_{n-(i+1)}. \end{aligned}$$

Treći korak: iskazati n-ti član pomoću prethodnih. Uvrstiti vrijednost za i u uzorak tako da T_n bude izražen kao funkcija samo baznog slučaja. Uvrstiti vrijednosti za ove izraze da bi dobili obični nerekurentni izraz za T_n. Za problem Hanojskih kula, zamjenom $i = n - 1$ u opštu formu koju smo odredili u drugom koraku, daje

$$\begin{aligned} T_n &= 1 + 2 + 4 + \cdots + 2^{n-2} + 2^{n-1} T_1 = \\ &= 1 + 2 + 4 + \cdots + 2^{n-2} + 2^{n-1} \end{aligned}$$

jer je $T_1 = 1$. Time smo dobili obični i nerekurentni izraz za T_n.

Četvrti korak: naći egzaktan izraz za n-ti član. Preostaje još naći egzaktan izraz za izraz T_n. U slaučaju Hanojskih kula, radi se o sumi geometrijskog reda, pa je

$$T_n = 1 + 2 + 4 + \cdots + 2^{n-2} + 2^{n-1} = \sum_{i=0}^{n-1} 2^i = 2^n - 1.$$

2.2 Sortiranje spajanjem (Merge sort)

Jedan od najpopularnijih algoritama za sortiranje liste on n elemenata je svakako sortiranje spajanjem. Posmatrajmo listu od $n \geqslant 1$ elemenata sa članovima $x_1, x_2, x_3, \ldots, x_n$. Ako je $n = 1$ algoritam vraća jedan element x_1. Ako je $n > 1$, tada se početna lista podijeli na dvije liste koje se rekurzivno sortiraju a zatim se spajaju u cijelu sortiranu listu sa n elemenata.

Naprimjer, posmatrajmo listu $10, 7, 23, 5, 2, 4, 3, 9$. Buduci da je lista duža od jedan, podijelimo je u dvije liste $10, 7, 23, 5$ i $2, 4, 3, 9$, koje rekurzivno sortiramo i dobijamo nove liste $5, 7, 10, 23$ i $2, 3, 4, 9$. Ove dvije "male" liste trebamo spojiti u jednu "veliku" listu. To radimo tako što počnemo sa jednom velikom praznom listom i dodajemo član po član. U svakom koraku poredimo prve članove u malim listama i manjeg od ta dva stavljamo na kraj velike liste. Taj proces nastavljamo dok jedna od malih lista ne postane prazne. U tom slučaju, preostala mala lista se dodaje velikoj listi čime se proces završava. Ilustrujmo to sljedećom tabelom.

mala lista ♯1	mala lista ♯2	velika lista
5,7,10,23	2,3,4,9	
5,7,10,23	3,4,9	2
5,7,10,23	4,9	2,3
5,7,10,23	9	2,3,4
7,10,23	9	2,3,4,5
10,23	9	2,3,4,5,7
10, 23		2,3,4,5,7,9
		2,3,4,5,7,9,10,23

U analizi ovog algoritma sortiranja, glavno pitanje jeste koji je maksimalni broj upoređivanja u sortiranju liste od n elemenata?

Broj upoređivanja se uzima kao procjena vremena izvođenja. U slučaju sortiranja spajanjem, možemo pronaći rekurziju za ovu veličinu. Rješavanje rekurzije će nam omogućiti proučavanje asimptotskog ponašanja ovog algoritama.

Radi lakše analize, pretpostavimo za sada da je broj članova (u listi koju sortiramo) stepen broja 2. Ovo osigurava da možemo podijeliti (izvornu) listu elemenata tačno na pola na svakom koraku rekurzije.

Neka je $T(n)$ maksimalan broj upoređivanja u implementaciji algoritma sortiranje spajanjem primijenjenog na listu od n elemenata. Ako je u listi samo jedan element, nema upoređivanja, pa je $T(1) = 0$. Ako je $n > 1$, tada je $T(n)$ suma:

- broja poređenja korištenih u sortiranju obje polovine liste, što je najviše $2T(n/2)$;
- broja poređenja korištenih u spajanju dvije "male" liste u "veliku" listu dužine n. Ovaj broj je najviše $n-1$ budući da je jedan član dodan velikoj listu poslije svakog poređenja i najmanje jedan dodatni član je dodan velikoj listi u završnom koraku kada jedna od manjih listi postane prazna. Budući da velika lista eventualno sadrži n članova, tada može najviše da bude $n-1$ poređenja. Mi ovdje analiziramo najgori slučaj jer u praksi može da bude manji broj poređenja.

Prema tome, broj poređenja potrebnih za sortiranje liste od n elemenata je najviše

$$T(n) = 2T(n/2) + n - 1.$$

Odredimo eksplicitnu formulu za broj poređenja u implementaciji algoritma na listu od n elemenata. To zahtijeva rješavanje rekurzije

$$\begin{aligned} T(1) &= 0 \\ T(n) &= 2T(n/2) + n - 1 \quad (\text{za } n > 1) \end{aligned}$$

Ako izračunamo prvih nekoliko članova, dobijamo

n	1	2	4	8	16
$T(n)$	0	1	5	17	49

odakle ne može uočiti nikakav uzorak. Pokušajmo drugom metodom, uvrštavanjem i

pojednostavljivanje. Tako dobijamo

$$\begin{aligned}
T(n) &= n - 1 + 2T(n/2) \\
&= (n-1) + 2(n/2 - 1 + 2T(n/4)) \\
&= (n-1) + (n-2) + 4T(n/4) \\
&= (n-1) + (n-2) + 4(n/4 - 1 + 2T(n/8)) \\
&= (n-1) + (n-2) + (n-4) + 8T(n/8) \\
&= (n-1) + (n-2) + (n-4) + 8(n/8 - 1 + 2T(n/16) \\
&= (n-1) + (n-2) + (n-4) + (n-8) + 16T(n/16)
\end{aligned}$$

Sada možemo uočiti obrazac i uraditi još jedan korak uvrštavanji u pojednostavljivanja, da bi dokazali da smo u pravu:

$$\begin{aligned}
T(n) &= (n-1) + (n-2) + \cdots + (n - 2^{i-1}) + 2^i T(n/2^i)) \\
&= (n-1) + (n-2) + \cdots + (n - 2^{i-1}) + 2^i (n/2^i - 1 + 2T(n/2^{i+1})) \\
&= (n-1) + (n-2) + \cdots + (n - 2^{i-1}) + (n - 2^i) + 2^{i+1} T(n/2^{i+1}))
\end{aligned}$$

Sada ćemo u dobijeni obrazac uvrsiti vrijednost za i tako da $T(n)$ ovisi samo od baznih slučajeva. Prirodno je izabrati $i = \log_2 n$, jer je tada $T(n/2^i) = T(1)$. Ova smjena nam daje da $T(n)$ ovisi samo do $T(1)$ čija je vrijednost jednaka 0. Dakle,

$$\begin{aligned}
T(n) &= (n-1) + (n-2) + \cdots + (n - 2^{\log_2 n - 1}) + 2^{\log_2 n} T(n/2^{\log_2 n})) \\
&= (n-1) + (n-2) + \cdots (n - n/2) + nT(1) \\
&= (n-1) + (n-2) + \cdots (n - n/2)
\end{aligned}$$

Dalje je

$$\begin{aligned}
T(n) &= (n-1) + (n-2) + \cdots (n - n/2) \\
&= n \log_2 n - (1 + 2 + 4 + \cdots + n/2) \\
&= n \log_2 n - n + 1 \sim n \log_2 n
\end{aligned}$$

Ako to uporedimo sa ranijim računom, dobijamo da je formula za $T(n)$ tačna:

n	$n \log_2 n - n + 1$
1	$1 \log_2 1 - 1 + 1 = 0$
2	$2 \log_2 2 - 2 + 1 = 1$
4	$4 \log_2 4 - 4 + 1 = 5$
8	$8 \log_2 8 - 8 + 1 = 17$
16	$16 \log_2 16 - 16 + 1 = 49.$

2.3 Još o rekurzijama

Uporedimo sada rekurzije Hanojskih kula i sortiranja spajanjem.

Hanojske kule	$T(n) = 2T(n-1) + 1$	$\Rightarrow T(n) \sim 2^n$
Sortiranje spajanjem	$T(n) = 2T(n/2) + n - 1$	$\Rightarrow T(n) \sim n \log_2 n$

Iako su rekurzije prilično slične, rješenja su jako različita.

Na prvi pogled, svaka rekurzija ima jednu prednost i jednu slabost. U stvari, kod Hanojskih kula, dijelimo problem veličine n na dva problema veličine $n-1$ (što je veliko) i potreban nam je još jedan korak (što je malo). U sortiranju spajanjem, dijelimo problem veličine n na dva problema veličine $n/2$ (što je malo) ali trebamo dodatnih $n-1$ koraka (što je veliko). Ali, bez obzira na to, sortiranje spajanjem je mnogo brži algoritam od Hanojskih kula. Dakle, generiranje manjih problema je važnije za brzinu algoritma nego reduciranje dodatnih koraka po rekurzivnom pozivu.

2.3.1 Brzi algoritam

Isprobajmo još jedan algoritam. Pretpostavimo da imamo brzi algoritam sa najboljim osobinama prethodnih algoritama, tj. da je u svakoj fazi problem podijeljen na pola i da imamo samo jedan dodatni korak. Vrijeme izvršenja je opisano sljedećom rekurzijom

$$\begin{aligned} S(1) &= 0 \\ S(n) &= 2S(n/2) + 1, \quad \text{za } n \geqslant 2 \end{aligned}$$

Pokušajmo prvo pogoditi i verifikovati rezultat. Kao i obično, predstavimo tabelarno nekoliko vrijednosti za $S(n)$ i pretpostavimo da je n stepen broja 2:

n	$S(n)$
1	0
2	$2S(1)+1=1$
4	$2S(2)+1=3$
8	$2S(4)+1=7$
16	$2S(8)+1=15.$

Očigledno je $S(n) = n-1$. Dokažimo sljedecu lemu.

Lema 2.3.1. *Pretpostavimo da je*

$$\begin{aligned} S(1) &= 0 \\ S(n) &= 2S(n/2) + 1, \quad \text{za } n \geqslant 2. \end{aligned}$$

Ako je n stepen broja 2, tada je $S(n) = n-1$.

Dokaz. Dokaz izvodimo jakom indukcijom. Neka je $P(n)$ tvrdnja da ako je n stepen broja 2, tada je$S(n) = n-1$.

Bazni slučaj, tojest $P(1)$ je tačan jer je $S(1) = 1-0 = 0$.

Pretpostavimo da je su $P(1), P(2), \ldots, P(n-1)$ tačni i dokažimo da je $P(n)$ tačan za $n \geqslant 2$. Dalje, imamo

$$S(n) = 2S(n/2) + 1 = 2(n/2-1) + 1 = n-1.$$

□

Zbog svega, vrijeme izvršenja brzog algoritma je $S(n) \sim n$ i to je bolje od $T(n) \sim n\log_2 n$ što je vrijeme izvršenja algoritma sortiranje spajanjem, ali samo malo bolje. Ovo je konzistentno sa idejom da je smanjenje broja dodatnih koraka po rekurzivnom pozivu manje bitno nego smanjenje veličine podproblema.

Ponekad provjera rješenja rekurzije koristeći indukciju može biti nezgodna. Naprimjer, pretpostavimo da smo uzeli rekurziju brzog algoritma i da pokušavamo dokazati da je $S(n) \leqslant n$. To jeste istina, ali je dokaz pogrešan.

Lema 2.3.2. *Pretpostavimo da je*

$$\begin{aligned} S(1) &= 0 \\ S(n) &= 2S(n/2) + 1, \quad \text{za } n \geqslant 2. \end{aligned}$$

Ako je n stepen broja 2, tada je $S(n) \leqslant n$.

Dokaz. Dokaz izvodimo jakom indukcijom. Neka je $P(n)$ tvrdnja da ako je n stepen broja 2, tada je $S(n) \leqslant n$.

Iskaz $P(1)$ je tačan jer je $S(1) = 1 - 0 = 0$.

Pretpostavimo da su za $n \geqslant 2$ izrazi $P(1), P(2), \ldots, P(n-1)$ tačni i dokažimo da je $P(n)$ tačan za $n \geqslant 2$. Dalje, imamo

$$S(n) = 2S(n/2) + 1 \leqslant 2(n/2) + 1 = n + 1 \not\leqslant n.$$

□

Znamo da je rezultat tačan, ali dokaz nije uspio. Prirodno se nameće da pojednostavimo i dokažemo nešto slabiji rezultat, recimo $S(n) \leqslant 2n$, a to je i ovdje pogrešan način jer

$$S(n) = 2S(n/2) + 1 \leqslant 2n + 1 \not\leqslant 2n.$$

Dakle, kao i kod drugih dokaza indukcijom, ključ je koristiti jaču induktivnu pretpostavku kao $S(n) = n - 1$ ili recimo $S(n) \leqslant n - 1$.

Šta će se desiti ako pokušamo sa jačom induktivnom pretpostavkom? Da li bi dokaz trebao biti jednostavniji?

Naprimjer, pretpostavimo da je $S(n) \leqslant n - 2$. Ova pretpostavka nije tačna jer smo dokazali da je $S(n) \leqslant n - 1$, ali pogledajmo gdje je greška u dokazu. Imamo,

$$S(n) = 2S(n/2) + 1 \leqslant 2(n/2 - 2) + 1 = n - 3 \leqslant n - 2.$$

Upravo smo dokazali pogrešnu tvrdnju. Gdje je greška? Greška leži u tome da nismo ispisali cijeli dokaz, tj. ignorisali smo bazni slučaj. Budući da je $S(1) = 0 \not\leqslant -1$, induktivna pretpostavka je pogrešna u baznom slučaju. Zbog toga ne možemo konstruisati validan dokaz sa "prejakom" indukcionom pretpostavkom.

Neke varijante algoritma sortranje spajanjem imaju doista čudna rješenja. Glavna razlika u ovim varijnantama jeste da zamijenimo konstantu 2 (koju smo dobili jer smo kreirali 2 podproblema) sa parametrom a. Tako dobijamo

$$\begin{aligned} T(1) &= 0 \\ T(n) &= aT(n/2) + n, \quad \text{za } n \geqslant 2. \end{aligned}$$

Intuitivno, parametar a je broj podproblema veličine $n/2$ generisanih u svakom koraku rekurzije ali ipak ne zahtijeva se da parametar a bude cio broj. Rekurzija se može riješiti

metodom uvrštavanja i pojednostavljivanja i ovdje ne dajemo te detalje. U zavisnosti od parametra a, imamo

$$T(n) \sim \begin{cases} \dfrac{2n}{2-a} & \text{za} \quad 0 \leq a < 2, \\ n \log_2 n & \text{za} \quad a = 2, \\ \dfrac{a n^{\log_2 a}}{a-2} & \text{za} \quad a > 2. \end{cases}$$

Ovo nam pokazuje da je sortiranje umetanjem jako osjetljivo na vrijednost parametra a, posebno kad je a blizu 2, jer naprimjer

$a = 1.99$	$T(n) = \Theta(n)$
$a = 2$	$T(n) = \Theta(n \log_2 n)$
$a = 2.01$	$T(n) = \Theta(n^{1.007\ldots})$

2.4 Akra-Bazzi metod

Sortiranje umetanjem i sve varijacije koje smo posmatrali se nazivaju *podijeli i osvoji* rekurzijama jer se svo vrijeme pojavljuju u *podijeli i osvoji* algoritmima. U opštem slučaju podijeli i osvoji rekurzije imaju oblik

$$T(x) = \begin{cases} \text{je definsana} & \text{za} \quad 0 \leqslant x \leqslant x_0, \\ \displaystyle\sum_{i=1}^{k} a_i T(b_i x) + g(x) & \text{za} \quad x > x_0. \end{cases}$$

Ovdje je x nenegativan realan broj koji ne mora biti stepen broja 2 ili čak cio broj. Brojevi $a_1, a_2, \ldots, a_k$ su pozitivne konstante, brojevi $b_1, b_2, \ldots, b_k$ su konstatne između 0 i 1 i broj x_0 je "dovoljno veliki" da osigura da je $T(n)$ dobro definisano. Ova opšta forma opisuje sve rekurzije u ovom dijelu, osim rekurzije za Hanojske kule. Naprimjer rekurzija $T(x) = 2T(x/2) + 8/9T(3x/2) + x^2$ pripada klasi podijeli i osvoji rekurzija.

Prije nekoliko godina, dvojac iz Bejruta pod imenima Akra i Bazzi su otkrili elegantan način da se riješe sve podijeli i osvoji rekurzije.

Teorem 2.4.1. *Pretpostavimo da je*

$$T(x) = \begin{cases} \text{je definsana} & \text{za} \quad 0 \leqslant x \leqslant x_0, \\ \displaystyle\sum_{i=1}^{k} a_i T(b_i x) + g(x) & \text{za} \quad x > x_0. \end{cases}$$

pri čemu su $a_1, a_2, \ldots, a_k$ su pozitivne konstante, brojevi $b_1, b_2, \ldots, b_k$ su konstatne između 0 i 1, broj x_0 je "dovoljno veliki" (u tehničkom smislu kojeg u ovom dijelu prihvatamo neodređeno) te da je $|g'(x)| = O(x^c)$ za neko $c \in \mathbb{N}$. Tada je

$$T(x) = \Theta\left(x^p \left(1 + \int_1^x \frac{g(u)}{u^{p+1}} du\right)\right)$$

pri čemu p zadovoljava jednačinu $\sum_{i=1}^{k} a_i b_i^p = 1$.

Ovu teoremu nećemo dokazivati već samo primijeniti na rekurziju $T(x) = 2T(x/2) + 8/9T(3x/2) + x^2$.

Prvi korak je odrediti p koje je definisano jednačinom

$$2\left(\frac{1}{2}\right)^p + \frac{8}{9}\left(\frac{3}{4}\right)^p = 1.$$

Ovakve jednacine je u opštem slučaju jako teško riješiti, ali ovdje vidimo da je rješenje jednostavno i da je 2. Dalje, vidimo da $g(x)$ ne raste prebrzo jer $|g'(x)| = |2x| = O(x)$. Konačno, možemo izračunati rješenje rekurzije sa

$$T(x) = \Theta\left(x^2\left(1 + \int_1^x \frac{g(u)}{u^{p+1}}du\right)\right) = \Theta(x^2(1 + \ln x)) = \Theta(x^2 \log_e x).$$

Posmatrajmo još jednu rekurziju. Pretpostavimo da za dovoljno veliko x vrijedi rekurzije

$$T(x) = T(x/3) + T(x/4) + x.$$

Primijetimo da je $|g'(x)| = 1 = O(1)$ te možemo primijeniti Akra-Bazzi teorem. Dalje, trebamo odrediti parametar p iz jednakosti

$$\left(\frac{1}{3}\right)^p + \left(\frac{1}{4}\right)^p = 1$$

što je problem. Ali barem možemo procijeniti da je $p < 1$, pa primjenom Akra-Bazzi teoreme, dobijamo

$$\begin{aligned}
T(x) &= \Theta\left(x^p\left(1 + \int_1^x \frac{g(u)}{u^{p+1}}du\right)\right) \\
&= \Theta\left(x^p\left(1 + \int_1^x u^{-p}du\right)\right) \\
&= \Theta\left(x^p\left(1 + \frac{u^{1-p}}{1-p}\Big|_{u=1}^{x}\right)\right) \\
&= \Theta\left(x^p\left(1 + \frac{x^{1-p}-1}{1-p}\right)\right) \\
&= \Theta\left(x^p + \frac{x}{1-p} - \frac{x^p}{1-p}\right) \\
&= \Theta(x)
\end{aligned}$$

U zadnjem koraku smo koristili da je $x^p = O(x)$ jer je $p < 1$, odnosno da izraz x dominira nad izrazima koji sadrže x^p, pa ovo drugo možemo zanemariti. U svakom slučaju, ovaj račun nam pokazuje da ne moramo znati tačnu vrijednost parametra p jer se svakako poništava.

Vidjeli smo da je asimptotska notacija korisna, posebno u vezi sa rekurzijama, gdje je indukcija jako omiljen i koristan način dokazivanja. Ali miješanje ovo dvoje je jako rizično jer se često potkrade greška. Ilustrujmo to na primjeru.

Lema 2.4.2. *Ako je*

$$T(1) = 1$$
$$T(n) = 2T(n/2) + n, \quad \text{za } n \geqslant 2.$$

tada je $T(n) = O(n)$.

Ova tvrdnja je pogrešna jer Akra-Bazzi teorem kaže da je tačno rješenje $T(n) = \Theta(n \log n)$. Gdje je greška sljedećem u dokazu?

Dokaz. Dokaz izvodimo jakom indukcijom. Neka je $P(n)$ tvrdnja da je $T(n) = O(n)$.
Bazni slučaj, tvrdnja $P(1)$ je tačna jer je $T(1) = 1 = O(1)$.
Pretpostavimo da je za $n \geqslant 2$ tvrdnje $P(1), P(2), \ldots, P(n-1)$ tačne i dokažimo da je $P(n)$ tačno. Imamo

$$T(n) = 2T(n/2) + n = 2O(n/2) + n = O(n).$$

Prva jednakost je rekurzija, druga je iskorištena pretpostavka a treća je pojednostavljenje. □

Primijetimo da na problem nailazimo već u induktivnoj hipotezi. Tvrdnja "$T(n) = O(n)$" je tačna ili netačna i njena vrijednost ne ovisi od određene vrijednosti n. Prema tome, sama ideja da se pokuša dokazati da je tvrdnja tačna za $n = 0, 1, 2, \ldots$ je pogrešna.
Pokušajmo sada dokazati gornju tvrdnju koristeći definiciju asimptotske notacije. Sjetimo se da $f(n) = O(n)$ znači da postoje konstante n_0 i $c > 0$ takve da je $f(n) \leqslant cn$ za svako $n \geqslant n_0$. Ako sve bude u redu, pokušaj dokazivanja bi trebao propasti na neki očigledan način, umjesto na suptilan, teško uočen način poput prethodnog argumenta. Budući da je naš cilj da dokažemo da tvrdnja neće da vrijedi za bilo koje konstante n_0 i c, zasad ćemo ih zanemariti i pretpostaviti da su izabrane tako da vrijedi bazni slučaj, tj. $T(n_0) \leqslant cn$. Dokaz ponovo izvodimo potpunom indukcijom. Pretpostavimo da je $P(n)$ tvrdnja da je $T(n) \leqslant cn$.
Očigledno vrijedi bazni slučaj $P(n_0)$ jer je $T(n_0) \leqslant cn$.
Za $n > n_0$ pretpstavimo da su $P(n_0), \ldots, P(n-1)$ tačni i dokažimo da je $P(n)$ tačno. Imamo

$$T(n) = 2T(n/2) + n = 2c(n/2) + n = cn + n = (c+1)n \not\leqslant cn.$$

Prva jednakost je rekurzija, zatim koristimo indukciju i pojednostavljivanje dok nam tvrdnja ne propadne.

2.5 Linearne rekurzije

U prethodnim lekcijama smo vidjeli kako se rješavaju rekurzije tipa podijeli i osvoji. Sada ćemo posmatarti drugu familiju rekurzija pod nazivom *linearne rekurzije* koje se također jako često koriste u kompjuterskim naukama i drugim disciplinama. Prvo ćemo vidjeti kako se rješavaju određeni tipovi linearnih rekurzija a onda generalizirati metodu za rješavanje svih linearnih rekurzija.

Posmatrajmo problem kojeg ćemo nazvati *Izgledi za posao diplomiranih studenata.* Recimo da u naučnoj oblasti kompjuterskih nauka ima samo određeni broj fakultetskih pozicija širom svijeta. U početku, nema dovoljno kvalifikovanih kandidata pa su neke

pozicije nepopunjene. Ali vremenom, novi diplomirani studenti popunjavaju pozicije, čineći tako izglede budućim diplomiranim studetima lošijim. Još više, povećanje broja profesora koji mogu podučavati povećan broj studenata čineći da se pozicije brže popune. To može dovesti do toga da su univerziteti zasićeni ovim kadrom te više uopšte neće biti mogućnosti za akademskim napredovanjem u ovoj oblasti, jer novi diplomci neće imati šansu za akademskom karijerom.

Naš zadatak jeste da odredimo kada će univerziteti prestati angažovati nove diplomce. Ovo su detalji problema:

- Postoji ukupno N slobodnih pozicija širom svijeta. Broj pozicija se nikad ne mijenja zbog budžetskih ograničenja.
- Postoji pravilo da nema dobrovoljnog penzionisanja, tojest jednom kada je pozicija na fakultetu popunjena, nikada se više ne otvara.
- Svake godine svaki profesor obuči tačno jednog studenta koji će nastaviti dalje i postati također profesor sljedeće godine. Jedini izuzetak je da prve godine profesori ne obučavaju studente.
- U nultoj godini nema profesora kompjuterskih nauka, prve godine je jedan angažovan.

U idealnom slučaju, možemo odrediti formulu za broj profesora na svijetu u određenoj godini. Zatim možemo odrediti godinu u kojoj će svih N pozicija biti popunjeno. Neka je $f(n)$ broj profesora u godini n. Da bismo razvili intuiciju oko ovog problema, možemo izračunati prvih nekoliko vrijednosti za malo n.

$f(0) = 0$	nema profesora kompjuterskih nauka
$f(1) = 1$	1 novi profesor ali ne podučava studenta
$f(2) = 1$	1 stariji profesor koji podučava studenta
$f(3) = 2$	1 novi i 1 stari profesor, novi ne podučava a stariji podučava studenta
$f(4) = 3$	1 novi i 2 stara profesora, novi ne podučava a dva starija oba podučavaju
$f(5) = 5$	2 nova i 3 starija profesora
$f(6) = 8$	3 nova i 5 starijih prfesora

U opštem slučaju,broj profesora u godini n je broj profesora u prošloj godini plus broj novih angažmana. Broj profesora u prošloj godini je $f(n-1)$ dok je broj novih angažmana jednak broju profesora od prije dvije godine, tj. $f(n-2)$ jer je svaki od ovih profesora podučavao studenta prošle godine. Ovo nam daje sljedeću rekurziju za broj profesora

$$\begin{aligned} f(0) &= 0 \\ f(1) &= 1 \\ f(n) &= f(n-1) + f(n-2), \ (n \geqslant 2) \end{aligned}$$

Ovo je poznata Fibonačijeva rekurzija i ako se vratimo unazad malo, vidimo i da su vrijednosti od $f(n)$ koje smo izračunali, ustvari prvih nekoliko članova Fibonačijevog niza.

Ovdje postavljamo pitanje koliko je $f(n)$? Naravno, uvijek možemo izračunati koliko želimo članova Fibonačijevog niza ali bi bilo dobro naći eksplicitnu formulu za $f(n)$,

što ćemo i uraditi. Rješavanje Fibonačijeve rekurzije je prilično lagan posao jer je ona linearna. Linearna rekurzija je rekurzija oblika

$$f(n) = a_1 f(n-1) + a_2 f(n-2) + \cdots + a_d f(n-d) = \sum_{i=1}^{d} a_i f(n-i)$$

pri čemu su $a_1, a_2, \ldots, a_d$ konstante. Red ove rekurzije je d. Naprimjer, Fibonačijeva rekurzija je reda 2 i ima koeficijente $a_1 = a_2 = 1$.

Sada ćemo pokušati riješiti Fibonačijevu rekurziju. Tako ćemo možda uočiti način kako riješiti linearnu rekurziju u opštem slučaju. Naše pravilo uvrštavanja i dokazivanja je prvi metod kojeg ćemo primijeniti na nepoznate rekurzije. ispostavlja se da je za linearne rekurzije eksponencijalno rješenje dobra pretpostavka. Zbog toga, pretpostavimo da je $f(n) = cx^n$, gdje su c i x parametri uvedeni da poboljšaju izglede da ćemo tačno pogoditi rješenje dok ćemu u procesu dokazivanja odabrati vrijednosti da bismo ga dokazali. Za sada možemo i zanemartiti početne uslove $f(0) = 0$ i $f(1) = 1$.

Uvrštavanjem $f(n) = cx^n$ u Fibonačijevu rekurziju, dobijamo

$$\begin{aligned} f(n) &= f(n-1) + f(n-2) \\ cx^n &= cx^{n-1} + cx^{n-2} \\ x^2 &= x + 1 \\ x^2 - x - 1 &= 0 \\ x &= \frac{1 \pm \sqrt{5}}{2} \end{aligned}$$

Odavdje zaključujemo da konstanta c može biti prizvoljna, ali vrijednosti za x moraju biti $\frac{1 \pm \sqrt{5}}{2}$. Dakle, imamo dva rješenja Fibonačijeve rekurzije:

$$f(n) = c\left(\frac{1+\sqrt{5}}{2}\right)^n, \quad f(n) = c\left(\frac{1-\sqrt{5}}{2}\right)^n.$$

Ustvari, bilo koja kombinacija ova dva rješenja je također rješenje. Sljedeći teorem kaže da je ovo tačno u uopštem slučaju za linearne rekurzije.

Teorem 2.5.1. *Ako su $f(n)$ i $g(n)$ rješenja linearne rekurzije (bez početnih uslova), tada je i $cf(n) + dg(n)$ također rješenje te rekurzije.*

Dokaz. Budući da su $f(n)$ i $g(n)$ rješenja linearne rekurzije, tada je

$$f(n) = \sum_{i=1}^{d} a_i f(n-i)$$

$$g(n) = \sum_{i=1}^{d} a_i g(n-i)$$

Množenjem prve jednakosti sa c i druge sa d, zatim sabiranjem, dobijamo:

$$\begin{aligned} cf(n) + dg(n) &= c\left(\sum_{i=1}^{d} a_i f(n-i)\right) + d\left(\sum_{i=1}^{d} a_i g(n-i)\right) \\ &= \sum_{i=1}^{d} a_i \left(cf(n-i) + dg(n-1)\right) \end{aligned}$$

odakle zaključujemo da je i $cf(n) + dg(n)$ rješenje rekurzije. □

Ovu osobinu, da je linearna kombinacija rješenja također rješenje, nalazimo i u oblasti diferencijalnih jednačina kao i mnogih fizičkih sistema.

Prethodni teorem implicira da je

$$f(n) = c_1\left(\frac{1+\sqrt{5}}{2}\right)^n + c_2\left(\frac{1-\sqrt{5}}{2}\right)^n$$

rješenje Fibonačijeve rekurzije bez početnih uslova, za sve konstante c_1 i c_2. Preostaje još odabrati konstante c_1 i c_2 tako da budu zadovoljeni početni uslovi. Iz prvog uslova znamo

$$f(0) = c_1\left(\frac{1+\sqrt{5}}{2}\right)^0 + c_2\left(\frac{1-\sqrt{5}}{2}\right)^0 = c_1 + c_2 = 0$$

dok iz drugog

$$f(1) = c_1\left(\frac{1+\sqrt{5}}{2}\right)^1 + c_2\left(\frac{1-\sqrt{5}}{2}\right)^1 = c_1\frac{1+\sqrt{5}}{2} + c_2\frac{1-\sqrt{5}}{2} = 1.$$

Ovo su dvije linearne jednačine sa dvije napoznate čije je rješenje $c_1 = \dfrac{1}{\sqrt{5}}$ i $c_2 = -\dfrac{1}{\sqrt{5}}$. Dakle, opšte rješenje Fibonačijeve rekurzije sa početnim uslovima je

$$f(n) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n.$$

Primijetimo da iako ovaj izraz sadrži $\sqrt{5}$, Fibonačijevi brojevi su cijeli brojevi.

Vratimo se sada početnom pitanju: Koliko vrmena će proći dok se ne popune svih N pozicija na fakultetima? Da bi odgovorili na ovo pitanje, trebamo pronaći najmanju vrijednost od n za koju je $f(n) \geqslant N$, tojest trebamo odrediti godinu n u kojoj je profesora onoliko koliko je univerzitetskih pozicija. Budući da izraz za $f(n)$ ima veoma složen oblik, teško je izračunati tačnu vrijednost za n, ali možemo naći odličnu aproksimaciju. Primijetimo da u opštoj formi, drugi izraz brzo opada ka 0, tojest

$$f(n) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n + o(1).$$

Odavdje možemo da procijenimo godinu koja se traži sa

$$f(n) \approx \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n \geqslant N$$

odakle je

$$n = \frac{\log(\sqrt{5}N + o(1))}{\log\left(\frac{1+\sqrt{5}}{2}\right)} = \Theta(\log N)$$

što ima i smisla budući da se broj profesora povećava eksponecijalno. Ako odaberemo da je naprimjer $N = 10000$ pozicija, bilo bi potrebno oko $n = 20.8$ godina.

Broj $\gamma = \frac{1+\sqrt{5}}{2}$ se naziva "zlatni presjek", te formulu za opšti član Fibonačijevog niza možemo pisati u obliku

$$f(n) = \frac{\gamma^n}{\sqrt{5}} + o(1)$$

Metod koji smo koristili da riješimo Fibonačijevu rekurziju se može iskoristiti da se riješi bilo koja opšta linearna rekurzija. Ako u linearnu rekurziju

$$f(n) = a_1 f(n-1) + a_2 f(n-2) + \cdots + a_d f(n-d)$$

uvrstimo da je $f(n) = x^n$, dobijamo

$$x^n = a_1 x^{n-1} + a_2 x^{n-2} + \cdots + a_d x^{n-d}$$

i dijeljenjem ove jednačine sa x^{n-d}, dobijamo

$$x^d = a_1 x^{d-1} + a_2 x^{d-2} + \cdots + a_{d-1} x + a_d$$

i nazivamo je *karakteristična jednačina* rekurzije.

Rješenja linearne rekurzije su definsana sa rješenjima karakteristične jednačine. Zanemarujući zasad početne uslove, vrijedi:

- ako je r rješenje karakteristične jednačine koje se ne ponavlja (višestukosti 1), tada je r^n rješenje rekurzije;
- ako je r rješenje karakteristične jednačine koje se ponavlja k puta (višestrukosti k), tada su
$$r^n, nr^n, n^2 r^n, \ldots, n^{k-1} r^n$$
rješenja rekurzije.

Također, linearna kombinacija ovih rješenja je također rješenje rekurzije.

Primjer 2.2. *Pretpostavimo da postoji biljka koja živi vječno i koja se reproducira samo u prvoj godini života. Postavimo pitanje: Koliko brzo će vrsta da raste? Primijetimo da je ovo obrnuti problem od problema kojeg smo obradili sa popunjavanjem akademskih pozicija, gdje se "fakultet reproducirao" u svakoj godini osim u prvoj.*

Neka je $f(n)$ broj biljaka u godini n i kao početne uslove postavimo $f(0) = 0$ i $f(1) = 1$. Sada je populacija biljke u godini n jednaka populaciji biljke u prethodnoj godini plus broj novih biljaka. Populacija u prethodnoj godini je $f(n-1)$ dok je broj novih biljaka u ovoj godini jednak broju novih biljaka prošle godine $f(n-1) + f(n-2)$. Posmatrajući sve ovo zajedno, dobijamo rekurziju za populaciju biljke u obliku

$$f(n) = f(n-1) + (f(n-1) - f(n-2)) = 2f(n-1) - f(n-2).$$

Karakteristična jednačina je $x^2-2x+1=0$, čije je rješenje $x=1$ višestrukosti 2. Dakle, opšte rješenje rekurzije je

$$f(n) = c_1 1^n + c_2 n 1^n = c_1 + c_2 n.$$

Ako sikoristimo početne uslove, dobijamo da je $f(n) = n$.
Dakle, u $n-$toj godini imamo ukupno n biljaka.

2.5.1 Nehomogene rekurzije

Ako homogenoj rekurziji

$$f(n) = a_1 f(n-1) + a_2 f(n-2) + \cdots + a_d f(n-d)$$

na desnoj strani dodamo član $g(n)$, dobijamo opšti oblik nehomogene rekurzije

$$f(n) = a_1 f(n-1) + a_2 f(n-2) + \cdots + a_d f(n-d) + g(n).$$

Naprimjer, ako Fibonačijevoj rekurziji dodamo član 1, dobijamo nehomogenu rekurziju u obliku

$$f(n) = f(n-1) + f(n-2) + 1.$$

Rješavanje nehomogenih rekurzija radimo u nekoliko koraka:

- Rješavamo pripadajuću homogenu rekurziju ($g(n) = 0$) na raniji način. U ovom dijelu ignorišemo početne uslove. Rješenje homogene rekurzije ćemo zvati *homogeno rješenje.*
- Dalje tražimo tzv. *partikularno rješenje* nehomogene rekurzije. Postoji opšti metod za nalaženje partikularnog rješenja i u ovom dijelu također ignorišemo početne uslove.
- Sabiranjem homogenom i partikularnog rješenja dobijamo *opšte rješenje rekurzije* i u ovom dijelu koristimo početn uslove da bismo odredili nepoznate konstante.

Primjer 2.3. *Riješiti rekurziju*

$$f(1) = 1$$
$$f(n) = 4f(n-1) + 3^n$$

Prvo, posmatrajmo odgovarajuću homogenu rekurziju $f(n) = 4f(n-1)$. Odgovarajuća arakteristična jednačina je $x-4=0$, čije je rješenje $x=4$. Dakle, homogeno rješenje je $f(n) = c4^n$.
Odredimo sada partikularno rješneje rekurzije $f(n) = 4f(n-1) + 3^n$. Pretpostavimoda je partikularno rješenje u obliku $d3^n$, pri čemu je d proizvoljna konstanta. zamjenom u rekurziju, dobijamo

$$d3^n = 4d3^n + 3^n$$
$$3d = 4d + 3$$
$$d = -3.$$

Dakle, partikularno rješenje je $f(n) = -3 \cdot 3^n = -3^{n+1}$. Zbir homogenog i partikularnog rješenja je $f(n) = c4^n - 3^{n+1}$. Ako iskoristimo početni uslov, tj. da je $f(1) = 1$, dobijamo da je $c = \frac{5}{2}$.
Opšte rješenje rekurzije je $f(n) = \frac{5}{2}4^n - 3^{n+1}$.

Najteži dio pri rješavanju rekurzija jeste odrediti partikularno rješenje jer podrazumijeva pogađanje. Ipak, postoje neka pravila koja nam olakšavaju ova pogađanja. U opštem slučaju, treba tražiti partikularno rješenje u istom obliku kao nehomogeni dio $g(n)$. Evo nekih od pravila.

- Ako je $g(n)$ konstanta, partikularno rješenje tražimo u obliku $f(n) = c$, a ako ne uspijemo onda u obliku $f(n) = bn + c$ ili $f(n) = an^2 + bn + c$, itd.
- Ako je $g(n)$ polinom, partikularno rješenje tražimo u obliku polinoma istog stepena, a ako ne uspijemo onda u obliku polinoma stepena većag za 1, itd. Naprimjer, ako je $g(n) = 6n + 5$, partikularno rješenje tražimo u obliku $f(n) = bn + c$, a ako ne uspijemo onda u obliku $f(n) = an^2 + bn + c$, itd.
- Ako je $g(n)$ eksponencijalna funkcija, kao naprimjer 3^n, tada partikularno rješenje tražimo u obliku $f(n) = c3^n$, a ako ne uspijemo onda u obliku $bn3^n + c3^n$ ili obliku $an^2 3^n + bn3^n + c3^n$, itd.

3

Generirajuće funkcije

Generirajuće funkcije predstavljaju veoma koristan alat u diskretnoj matematici. Ukratko, generirajuće funkcije predstavljaju alat za transformaciju problema o nizovima u algebarske probleme, što je dobro jer poznajemo veliki broj algebarskih pravila koje možemo primjenjivati. Generirajućim funkcijama probleme vezane za nizove svodimo na ispitivanje algebarskih izraza i time možemo riješiti veliki broj problema brojanja.

Postoji razne vrste generirajućih funkcija kao što su *obične*, *eksponecijalne* i *Dirichletove generirajuće funkcije*. U ovom dijelu ćemo se baviti samo sa običnim generirajućim funkcijama što je dovoljno za prezentaciju ideje korištenja ovog alata. Recimo, $\mathcal{Z}$-transformacije, koje su usko povezane sa običnim generirajućim funkcijama, su veoma bitne u teoriji kontrole i obradi signala. Generirajuće funkcije prvi je uveo *Abraham de Moivre* 1730. godine, kako bi se riješio opći problem linearne rekurzije.

Definicija 3.0.1. *Posmatrajmo niz realnih brojeva* $f_0, f_1, f_2, \ldots, f_n, \ldots$. *Generirajuća funkcija* $F(x)$ *je beskonačni red*

$$F(x) = f_0 + f_1 x + f_2 x^2 + \cdots = \sum_{n=0}^{\infty} f_n x^n. \tag{1}$$

Koeficijent uz stepen x^n generirajuće funkcije (1) ćemo označavati sa $[x^n]F(x)$. Dakle,

$$[x^n]F(x) := f_n.$$

Osobine bilo kojeg niza realnih brojeva $f_0, f_1, f_2, \ldots, f_n, \ldots$ možemo analizirati posmatrajući ih kao uzastopne koeficijente generirajuće funkcije. Ispostavlja se da neke osobine veoma "komplikovanih" nizova realnih brojeva koji su nastali prebrojavanjem nečega mnogo lakše objasniti koristeći generirajuće funkcije.

Primjer 3.1. *Posmatrajmo konstantni niz relanih brojeva* $1, 1, 1, \ldots$. *Generirajuća funkcija u ovom slučaju je*

$$G(x) = 1 + x + x^2 + \cdots = \sum_{n=0}^{\infty} x^n. \tag{2}$$

Koristeći algebarska pravila, lako možemo izvesti jednostavniju formulu za generirajuću funkciju $G(x)$. *Iz* (2) *dobijamo da je*

$$-xG(x) = -x - x^2 - \cdots. \tag{3}$$

Sabiranjem (2) *i* (3), *dobijamo da je* $G(x) - xG(x) = 1$, *odakle je* $G(x) = \frac{1}{1-x}$. *Dakle,*

$$G(x) = \sum_{n=0}^{\infty} x^n = \frac{1}{1-x} \tag{4}$$

odnosno

$$[x^n]\left(\frac{1}{1-x}\right) = 1.$$

■

Primjer 3.2. *Posmatrajmo niz relanih brojeva* $1, 2, 3, \ldots$. *Generirajuća funkcija u ovom slučaju je*

$$N(x) = 1 + 2x + 3x^2 + \cdots = \sum_{n=0}^{\infty}(n+1)x^n. \tag{5}$$

Iz (5) *dobijamo da je*

$$-xN(x) = -x - 2x^2 - 3x^3 - \cdots . \tag{6}$$

Sabiranjem (5) *i* (6), *dobijamo da je* $N(x) - xN(x) = 1 + x + x^2 + \cdots = G(x)$, *odakle je* $N(x) = \frac{G(x)}{1-x}$, *odnosno* $N(x) = \frac{1}{(1-x)^2}$ *Dakle,*

$$N(x) = \sum_{n=0}^{\infty}(n+1)x^n = \frac{1}{(1-x)^2} \tag{7}$$

odnosno

$$[x^n]\left(\frac{1}{(1-x)^2}\right) = n + 1. \tag{8}$$

■

Veoma bitno je napomenuti da generirajuće funkcije ne posmatramo kao formalne stepene redove i zbog toga pitanje konvergencije neće biti razmatrano u radu sa njima. Identitete (2) i (5) posmatramo kao simboličke identitete koje vrijede čisto na osnovu algebarskih identita. Naravno, numerički gledano identiteti (2) i (5) su tačni samo ako je $|x| < 1$.

3.1 Prebrojavanja pomoću generirajućih funkcija

Generirajuće funkcije su veoma korisne za predstavljanje i prebrojavanje načina izbora n stvari. Naprimjer, pretpostavimo da imamo dvije vrste loptica, zelene i crvene. Označimo sa l_n broj načina na koji možemo izabrati n zelenih ili crvenih loptica. Tada je $l_n = n+1$ jer postoji $n + 1$ način za izbor loptica: da izaberemo nula zelenih i n crvenih, jednu zelenu i $n - 1$ crvenih, dvije zelene i $n - 2$ crvene, i na kraju mogućnost da izaberemo n zelenih i 0 crvenih. Zbog toga, definišemo generirajuću funkciju $L(x)$ za prebrojavanje izbora loptica uzimajući da su l_n koeficijenti uz x^n. Na osnovu (5) dobijamo da je

$$L(x) = \frac{1}{(1-x)^2}. \tag{9}$$

Posmatrajmo sada malo složeniji slučaj. Pretpostavimo da imamo dvije vrste voća, jabuke i kruške i da imamo neka ograničenja u vezi njihovog izbora. Neka je j_n broj načina da izabermo n jabuka i neka je k_n broj načina da izabermo n krušaka. Generirajuća funkcija za prebrojavanje jabuka je

$$J(x) := \sum_{n=0}^{\infty} j_n x^n$$

a generirajuća funkcija za prebrojavanje krušaka je

$$K(x) := \sum_{n=0}^{\infty} k_n x^n$$

Dalje, pretpostavimo da jabuke dolaze u pakovanjima od po 6 jabuka. Prema tome ne postoji način da izaberemo od jedne do pet jabuka (nula nacina) a samo je jedan način da izaberemo 6 jabuka, ne postoji način da izabermo 7 jabuka, itd. Drugim riječima

$$j_n := \begin{cases} 1 & \text{ako je } n \text{ djeljivo sa 6,} \\ 0 & \text{inače.} \end{cases}$$

U ovom slučaju imamo

$$J(x) := \sum_{n=0}^{\infty} j_n x^n = 1+x^6+x^{12}+\cdots = (\text{smjena } x^6 = y) = 1+y+y^2+\cdots = \frac{1}{1-y} = \frac{1}{1-x^6}.$$

Pretpostavimo sada da imao dvije vrste krušaka, žute i zelene. Posmatrajući problem prebrojavanja krušaka kao ranije opisani problem prebrojavanja loptica, zaključujemo da je $k_n = n + 1$, pa osnovu (5) dobijamo da je

$$K(x) = \frac{1}{(1-x)^2}.$$

Sada možemo postaviti pitanje: na koliko načina možemo izabrati mješavinu jabuka i krušaka? U tom cilju, prvo moramo odlučiti koliko jabuka želimo da izaberemo. Ako taj broj označimo sa m tada vrijednosti m idu od 0 do n. Prema tome, možemo izabrati jabuke na j_m načina. Izborom m jabuka, preostaje nam $n-m$ krušaka pa kruške možemo izabrati na k_{n-m} načina. Ukupni broj načina za izbor m jabuka i $n - m$ krušaka, sa ranije datim ograničenjima je dakle

$$j_0k_n + j_1k_{n-1} + j_2k_{n-2} + \cdots + j_nk_0. \tag{10}$$

Veoma je zanimljiva i veza između prebrojavanja elemenata nekog skupa i generirajućih funkcija. Naime, izraz (10) je ustvari koeficijent uz x^n u proizvodu $J(x) \cdot K(x)$, tojest vriejdi sljedeći teorem.

Teorem 3.1.1 (Pravilo proizvoda)**.** *Za generirajuće funkcije $J(x)$ i $K(x)$ vrijedi*

$$[x^n](J(x) \cdot K(x)) = j_0k_n + j_1k_{n-1} + j_2k_{n-2} + \cdots + j_nk_0. \tag{11}$$

Da bi ovo pojasnili, posmatrajmo sljedeću tabelu u kojoj se nalaze proizvodi elemenata generirajućih funkcija:

	k_0x^0	k_1x^1	k_2x^2	k_3x^3	$\cdots$
j_0x^0	$j_0k_0x^0$	$j_0k_1x^1$	$j_0k_2x^2$	$j_0k_3x^3$	$\cdots$
	$\iddots$	$\iddots$	$\iddots$		
j_1x^1	$j_1k_0x^1$	$j_1k_1x^2$	$j_1k_2x^3$	$\cdots$	
	$\iddots$	$\iddots$			
j_2x^2	$j_2k_0x^2$	$j_2k_1x^3$	$\cdots$		
	$\iddots$				
j_3x^3	$j_3k_0x^3$	$\cdots$			
$\vdots$	$\cdots$				

U ovom zapisu, svi članovi koji sadrže isti stepen od x leže na jednoj dijagonali. Dakle, članovi koji sadrže x^1 se nalaze da prvoj dijagonali, članovi koji sadrže x^2 se nalaze na drugoj dijagonali, itd. Zbog toga je koeficijent uz x^n u proizvodu $J(x)\cdot K(x)$ ustvari zbir koeficijenata članova na $n-$toj dijagonali, odnosno to je zbir (10).

Definicija 3.1.1. *Konvolucija nizova $j_0, j_1, j_2, \ldots$ i $k_0, k_1, k_2, \ldots$ je niz $j_0k_0, j_1k_0 + j_0k_1, j_2k_0 + j_1k_1 + j_0k_2, \ldots$, tojest niz koeficijenata u proizvodu $J(x)\cdot K(x)$.*

Konvolucija nizova igra veoma važnu ulogu u teoriji kontrola i obradi signala.

Koristeći proizvod generirajućih funkcija, lako se pokazuje da je suma geometrijskog reda jednaka $\dfrac{1}{1-x}$, bez ispitivanja konveregencije, što je naš pristup ovdje. Naime, kako je

$$1 = 1 + 0x + 0x^2 + 0x^3 + \cdots$$

i $1 - x = 1 + (-1)x + 0x^2 + 0x^3 + \cdots$ za generirajuću funkciju (2), koristeći pravilo (10) za množenje generirajućih funkcija, dobijamo

$$(1-x)G(x) = 1 + 0x + 0x^2 + 0x^3 + \cdots = 1$$

odakle dobijamo formulu (4).

Specijalan slučaj je kada generirajuću funkciju množimo proizvoljnom konstantom, što iskazujemo sljedećim teroremom bez dokaza.

Teorem 3.1.2 (Pravilo konstante)**.** *Ako je c proizvoljna konstanta i $F(x)$ proizvoljna generirajuća funkcija, tada vrijedi*

$$[x^n](c\cdot F(x)) = c\cdot [x^n]F(x).$$

Svu raniju diskusiju možemo ustvari iskazati sljedećim teoremom.

Teorem 3.1.3 (Pravilo konvolucije)**.** *Neka je $A(x)$ generirajuća funkcija za izbor elemenata iz skupa $\mathcal{A}$ i neka je $B(x)$ generirajuća funkcija za izbor elemenata iz skupa $\mathcal{B}$, pri čemu je $\mathcal{A} \neq \mathcal{B}$. Generirajuća funkcija za izbor elemenata iz skupa $\mathcal{A}\cup\mathcal{B}$ je*

proizvod $A(x) \cdot B(x)$.

Ovdje moramo naglasiti da se uslovi i restrikcije koje smo imali pri izboru elemenata u skupovima $\mathcal{A}$ i $\mathcal{B}$ prenose na izbor elemenata u skupu $\mathcal{A} \cup \mathcal{B}$.

Vratimo se sada prebrojavanju crvenih i zelenih loptica. Kako je samo jedan način da izaberemo tačno n zelenih loptica, to znači da je svaki koeficijent pripadajuće generirajuće funkcije jednak 1, pa je generirajuća funkcija u tom slučaju $\frac{1}{1-x}$. Istim rezonovanjem zaključujemo da je genereririrajuća funkcija za izbor tačno n crvenih loptica također $\frac{1}{1-x}$. Koristeći pravilo konvolucije, zaključujemo da je generirajuća funkcija za broj načina izbora n crvenih i zelenih loptica jednaka $\frac{1}{1-x} \cdot \frac{1}{1-x} = \frac{1}{(1-x)^2}$, što je ustvari formula (9).

Primjena pravila konvolucije na izbor zelenih i crvenih loptica se može prenijeti i na izbor loptica sa k različitih boja. U tom slučaju pripadajuća generirajuća funkcija je $\frac{1}{(1-x)^k}$. Iz kombinatorike je poznato da ako želimo izabrati n loptica sa k boja, onda to možemo uraditi na $\binom{n+(k-1)}{k}$ načina. Dakle,

$$[x^n]\left(\frac{1}{(1-x)^k}\right) = \binom{n+(k-1)}{n}. \tag{12}$$

Formulu (12) možemo izvesti i pomoću Maclaurinove formule

$$f(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \cdots .$$

Odavdje vidimo da će koeficijent uz x^n u razvoju funkcije $f(x) = \frac{1}{(1-x)^k}$ biti jednak $\frac{f^{(n)}(0)}{n!}$. Računajući $n-$ti izvod funkcije $f(x) = \frac{1}{(1-x)^k}$ dobijamo

$$f^{(n)} = k(k+1)(k+2)\cdots(k+n-1)(1-x)^{-(k+n)}.$$

Dakle,

$$\begin{aligned}[x^n]\left(\frac{1}{(1-x)^k}\right) &= \frac{f^{(n)}(0)}{n!} = \frac{k(k+1)(k+2)\cdots(k+n-1)(1-0)^{-(k+n)}}{n!} \\ &= \binom{n+(k-1)}{k},\end{aligned}$$

što je formula (12).

Pravilo konvolucije možemo iskoristiti i za izvođenje binomne formule. Ako posmatramo skup $\{a_1\}$, onda immao jedan način da ne izaberemo nijedan član skupa i jedan način da izaberemo samo član a_1. Ukoliko želimo izabrati dva, tri ili više elemenata iz ovog skupa, onda to ne možemo uraditi, tojest broj načina je nula. Zbog toga je pripadajuća generirajuća funkcija $1 + x$. Slično, izbor n elemenata iz bilo kojeg jednočlanog skupa $\{a_i\}$ ima generirajuću funkciju $1 + x$.

Sada, koristeći pravilo konvolucije zaključujemo da izbor n elemenata iz skupa $\{a_1, a_2, \ldots, a_m\}$ ima generirajuću funkciju $(1 + x)^m$ koja je proizvod generirajućih funkcija za izbor u svakom jednočlanom podskupu ovog skupa. Iz kombinatorike znamo da je izbor n elemenata u skupu od m elemenata jednak $\binom{m}{n}$ pa je

$$[x^n](1+x)^m = \binom{m}{n}$$

što predstavlja binomni teorem.

Primjeri sa prebrojavanjem loptica, jabuka i krušaka na govore da generirajuće funkcije omogućavaju da se problemi prebrojavanja elemenata nekog skupa mogu biti riješeni algebarskim manipulacijama. Vrijedi i obrnuto, pomoću njih i algebarski identiteti mogu biti izvedeni tehnikama prebrojavanja elemenata.

3.2 Jedan zanimljiv problem prebrojavanja

Ovdje ćemo riješiti sljedeći problem prebrojavanja, koji na prvi pogled izgleda jako komplikovan a kojeg ćemo riješiti pomoću generirajućih funkcija na, ispostavit će se, mnogo lakši način.

Naime, postavlja se pitanje na koliko načina možemo napuniti vreću sa n voćki ako imamo sljedeća ograničenja:

- broj jabuka mora biti paran;
- broj banana mora biti djeljiv sa 5;
- može biti najviše 4 narandže;
- može biti najviše jedna kruška.

Ovakvi uslovi su veoma komplikovani i na prvi pogled dobiti odgovor u slučaju n voćki izgleda nemoguć. Naprimjer, ako želimo vreću napuniti sa 6 voćki, onda to možemo uraditi na 7 načina:

Jabuke	6	4	4	2	2	0	0
Banane	0	0	0	0	0	5	5
Narandže	0	2	1	4	3	1	0
Kruške	0	0	1	0	1	0	1

Konstruišimo sada generirajuću funkciju za izbor jabuka. Naime, 0 jabuka možemo izabrati na jedan način, jednu jabuku na nula načina, dvije na jedan način, tri na nula načina, i tako dalje. To je iz razloga što broj jabuka koje biramo mora biti paran. Dakle, pripadajuća generirajuća funkcija je

$$J(x) = 1 + x^2 + x^4 + \cdots = \frac{1}{1-x^2}.$$

Sličnim razmišljanjem generirajuća funkcija za izbor banana je

$$B(x) = 1 + x^5 + x^{10} + \cdots = \frac{1}{1-x^5}.$$

Nula, jednu, dvije, tri i četiri narandže možemo izabrati na jedan način. Pet, šest i više narandži možemo izabrati na nula načina. Dakle, generirajuća funkcija u ovom slučaju je

$$N(x) = 1 + x + x^2 + x^3 + x^4 = \frac{1-x^5}{1-x}.$$

Budući da možemo izabrati nula ili jednu krušku, generirajuća funkcija u ovom slučaju je

$$K(x) = 1 + x.$$

Koristeći pravilo konvolucije dobijamo da je generirajuća funkcija za izbor sve četiri vrste voćki jednaka

$$J(x)\cdot B(x)\cdot N(x)\cdot K(x)=\frac{1}{1-x^2}\cdot\frac{1}{1-x^5}\cdot\frac{1-x^5}{1-x}\cdot(1+x)=\frac{1}{(1-x)^2}.$$

Na osnovu formula (5), (7) i (8) zaključujemo da je n voćki, sa zadanim ograničenjima, moguće izabrati na $n+1$ način.

3.3 Fibonaccijev niz brojeva

Fibonaccijev niz brojeva je niz brojeva u kome je svaki član niza zbir prethodna dva člana tog niza. Prva dva člana niza se zadaju i obično su to broj 1. Ukoliko želimo da odredimo n-ti član ovog niza, onda to može biti komplikovano jer ako koristimo ovaj pristup, onda bismo prije toga morali odrediti sve članove niza do $n-$tog člana.

Ako Fibonacijev niz označimo sa $f_0, f_1, f_2, \ldots$, ovaj niz može biti zadan i rekurzivnom formulom na sljedeći način

$$\begin{aligned}f_0&=0\\f_1&=1\\f_n&=f_{n-1}+f_{n-2},\ (n\geqslant 2).\end{aligned}$$

Izvedimo sada formulu za opšti član niza koristeći generirajuće funkcije. Neka je

$$F(x)=f_0+f_1x+f_2x^2+\cdots \tag{13}$$

generirajuća funkcija za Fibonaccijev niz brojeva. Iz (13) imamo

$$-xF(x)=-xf_0-f_1x^2-f_2x^3+\cdots \tag{14}$$

i

$$-x^2F(x)=-x^2f_0-f_1x^3-f_2x^4+\cdots \tag{15}$$

Sabiranjem (13), (14) i (15) dobijamo

$$F(x)(1-x-x^2)=f_0+(f_1-f_0)x+(f_2-f_1-f_0)x^2+(f_3-f_2-f_1)x^3+\cdots,$$

te na osnovu osobina Fibonaccijevog niza, dobijamo

$$F(x)(1-x-x^2)=x$$

odakle je

$$F(x)=\frac{x}{1-x-x^2}.$$

Rastavljanjem racionalne funkcije $F(x)$ na proste racionalne funkcije, dobijamo

$$F(x)=\frac{x}{1-x-x^2}=\frac{1}{\sqrt{5}}\left(\frac{1}{1-\alpha x}-\frac{1}{1-\beta x}\right)$$

gdje su $\alpha = \frac{1+\sqrt{5}}{2}$ i $\beta = \frac{1-\sqrt{5}}{2}$. Koristeći formule (2) i (4), dobijamo

$$\begin{aligned} F(x) &= \frac{1}{\sqrt{5}}\left((1+\alpha x+\alpha^2x^2+\cdots)-(1+\beta x+\beta^2x^2+\cdots)\right) \\ &= \frac{1}{\sqrt{5}}\left((\alpha-\beta)x+(\alpha^2-\beta^2)x^2+\cdots\right). \end{aligned}$$

Dakle,

$$[x^n]F(x) = \frac{1}{\sqrt{5}}(\alpha^n-\beta^n) = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

pa je

$$f_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right). \tag{16}$$

Formula (16) se naziva *Binetova formula* i predstavlja eksplicitnu formulu za opsti član Fibonaccijevog niza. Iako na prvi pogled izgleda komplikovano, primijetiomo da je izraz na desnoj strani jednakosti (16) uvijek prirodan broj.

3.4 Hanojske kule

O problemu Hanojskih kula i odgovarajućoj rekurziji smo se upoznali u Sekciji 2.1. Ovdje je samo rješavamo koristeći generirajuće funkcije.

Neka je

$$H(x) = h_0 + h_1x + h_2x^2 + \cdots \tag{17}$$

generirajuća funkcija za problem Hanojskih kula. Iz (17) imamo

$$-2xH(x) = -2h_0x - 2h_1x^2 - 2h_2x^3 + \cdots \tag{18}$$

i iskoristimo da je

$$-\frac{1}{1-x} = -1 - x - x^2 - \cdots. \tag{19}$$

Iz (17), (18) i (19) dobijamo

$$H(x)(1-2x) - \frac{1}{1-x} = h_0 - 1 + 0x + 0x^2 + \cdots = -1.$$

Dakle,

$$H(x)(1-2x) = \frac{1}{1-x} - 1 = \frac{x}{1-x}$$

odnosno

$$H(x) = \frac{x}{(1-2x)(1-x)} = \frac{1}{1-2x} - \frac{1}{1-x}.$$

Odavdje je na osnovu formula (2) i (4)

$$[x^n]H(x) = [x^n]\left(\frac{1}{1-2x}\right) - [x^n]\left(\frac{1}{1-x}\right) = 2^n - 1.$$

Dakle,$h_n = 2^n - 1$.

4

Uvod u jezike i konačne automate

4.1 Osnovni koncepti

Prvo ćemo predstaviti osnovne koncepte i pojmove kao što su *alfabet*, *string* i *jezik* kao i definicije koje ćemo koristiti u ovom dijelu.

Alfabet, u oznaci Σ, je konačan i neprazan skup simbola.

Primjer 4.1. *Primjeri alfabeta su:*

- $\Sigma = \{a, b, \ldots, z\}$*, skup svih malih pisanih slova,*
- $\Sigma = \{0, 1\}$*, skup binarnih cifara,*
- $\Sigma = \{0, 1 \ldots, 9\}$*, skup dekadnih cifara,*
- $\Sigma = \{0, 1 \ldots, 9, a, b, \ldots, z\}$*, skup alfanumeričkih simbola.*

■

String je konačan poredak simbola izabranih iz alfabeta Σ. Naprimjer, *abc*, *acb*, *bca*, *abca* su neki od stringova iz alfabeta $\Sigma = \{a, b, c\}$. Primijetimo da su a, b i c također stringovi iz alfabeta $\Sigma = \{a, b, c\}$. Kada pišemo a a, b i c kao elemente alfabeta $\Sigma = \{a, b, c\}$, tada se oni odnose kao simboli, ne stringovi.

Stringove obično možemo klasificirati po njihovoj dužini a to je broj pojavljivanja simbola u stringu. Standardna notacija za dužinu stringa x je $|x|$. Naprimjer, $|abc| = 3$ i $|a| = 1$.

String u kojem nema pojavljivanja simbola nekog alfabeta se naziva *prazan string* ili *nula string*, što označavamo sa ϵ (epsilon) i $|\epsilon| = 0$. Prema tome ϵ je string izabran iz proizvoljnog alfabeta Σ.

Za bilo koji alfabet Σ njegov stepen, tojest Σ^k, $(k \in \mathbb{N}_0)$ predstavlja skup svih stringova dužine k formiranih nad Σ.

Primjer 4.2. *Neka je* $\Sigma = \{0, 1\}$*, tada je*

- $\Sigma^0 = \{\epsilon\}$,
- $\Sigma^1 = \{0, 1\}$,
- $\Sigma^2 = \{00, 01, 10, 11\}$,
- $\ldots$
- $\Sigma^k = \{00 \ldots 0, 0 \ldots 1, 1 \ldots 0, \ldots\}$*, gdje je svaki string dužine* k*.*

■

Skup svih mogućih stringova nad alfabetom Σ oznacavamo sa Σ^*, pri čemu se operator $*$ naziva *Kleenyevo zatvorenje* i ustvari vrijedi $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots$.

Primjer 4.3. *Neka je* $\Sigma = \{0, 1\}$*, tada je*

$$\begin{aligned}\Sigma^* &= \{\epsilon\} \cup \{0,1\} \cup \{00,01,10,11\} \cup \{000,001,010,011,\ldots\} \cup \cdots \\ &= \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots\}\end{aligned}$$

i ovaj skup sadrži beskonačno mnogo stringova. ■

Ako iz skupa Σ^* isključimo prazan string ϵ, tada imamo neprazan skup stringova nad alfabetom Σ. Taj skup označavamo sa Σ^+, pri čemu se $+$ naziva *pozitivni operator*. Dakle, vrijedi $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$.

Jezik je skup stringova izabranih iz Σ^* za proizvoljan alfabet Σ. Ako je L jezik nad alfabetom Σ, tada je $L \subseteq \Sigma^*$. Zbog toga jezik L može da sadrži beskonačno mnogo stringova. Budući da su jezici skupovi stringova, novi jezici mogu biti konstruisani korištenjem operacija unije, presjeka, razlike i komplemneta. Naprimjer, komplement jezika L nad alfabetom Σ je dat sa $L' = \Sigma^* - L$, pri čemu Σ^* sadrži sve moguće skupove stringova formiranih nad Σ, kojeg uzimamo za univerzalni skupa. Prema tome, skup L' sadrži sve stringove iz Σ^* koji nisu u skupu L.

Novi jezik takođe može biti konstruisan korištenjem operacije *konkatenacije*, tj. spajanjem ili nadovezivanjem stringova. Neka su x i y dva stringa. Tada spoj stringova x i y je novi string xy koji je ustvari 'string gdje nakon x slijedi y'. Naprimjer, ako je x string od n simbola $a_1a_2\ldots a_n$ i ako je y string od m simbola $b_1b_2\ldots b_m$, tada je xy string od $n + m$ simbola $a_1a_2\ldots a_nb_1b_2\ldots b_m$. Primijetimo da spajanje nije komutativno, tojest $xy \neq yx$ sve dok je $x \neq y$. S druge strane, spajanje je asocijativno, tojest za tri proizvoljna stringa x, y i z vrijedi $(xy)z = x(yz)$. To omogućava spajanje stringova bez restrikcije kojim redom će se različite operacije spajanja izvoditi.

Također, operaciju spajanja možemo primijeniti i nad jezicima. Naprimjer, ako posmatra dva jezika $L_1 \subseteq \Sigma^*$ i $L_2 \subseteq \Sigma^*$, tada je njihov spoj jezik L_1L_2, tojest

$$L_1L_2 = \{xy : x \in L_1 \wedge y \in L_2\}.$$

Primjer 4.4. *Posmatrajmo dva jezika* $L_1 = \{\epsilon\}$ *i* $L_2 = \{00, 01, 10, 11\}$*. Tada je*

$$L_1L_2 = \{\epsilon\}\{00,01,10,11\} = \{\epsilon00, \epsilon01, \epsilon10, \epsilon11\} = \{00,01,10,11\}$$

jer je $\epsilon x = x$*.* ■

Također, vrijedi i sljedeće:

(a) Spajanje dva nula stringa ili jezika koji sadrže samo nula stringove je nula string, tojest $\epsilon\epsilon = \epsilon$.

(b) Ako je jezik L_1 prazan, tojest $L_1 = \emptyset$, tada je spajanje L_1L_2 prazan za bilo koji jezik L_2. Naprimjer, ako je $L_2 = \{ab, bc, abc\}$, tada je $L_1L_2 = \emptyset\{ab, bc, abc\} = \emptyset$.

(c) Ako su $L_1 = \emptyset$ i $L_2 = \emptyset$, tada je $L_1L_2 = \emptyset$.

Stringovi su po definiciji konačni ali jezik sadrži beskonačan broj stringova. Zbog toga je veoma bitno ograničenje za jezike u smislu da se specificiraju da budu konačni. Naprimjer, neka je $\Sigma = \{a\}$, tada je $\Sigma^* = L = \{\epsilon, a, aa, aaa, \ldots\}$, gdje imamo beskonačno mnogo stringova. Beskonačni stringovi jezika L mogu ekvivalentno biti predstavljeni na konačan način sa $L = \{a\}^*$.

Posmatrajmo drugu ilustraciju, gdje je jezik dat sa $L = \{0, 1\}^* \cup \{a\}\{b\}^*$. Tada se jezik može konstruisati, bilo spajanjem proizvoljnog broja stringova, od kojih je svaki 0 ili 1, ili spajanjem stringa a sa proizvoljnim brojem kopija od b.

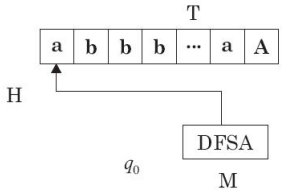
Slično, jezik $L = \{0x0 : x \in \{ab\}^*\}$ koji sadrži stringove x sa dodanom 0 na oba kraja, gdje je x proizvoljan string formiran od a i b.

Budući da je $L^0 = \{\epsilon\}$, $L^1 = L$, $L^2 = L^1L = LL^1$, $L^3 = L^2L$, ..., $L^k = L^{k-1}L$, tada je $L^* = L^0 \cup L^1 \cup \cdots \cup L_k \cup \cdots$. Dakle, $L^* = \bigcup_{k=0}^{\infty} L^k$, odakle je $L^+ = \bigcup_{k=1}^{\infty} L^k = LL^* = L^*L$.

4.2 Deterministički konačni automat (DFA)

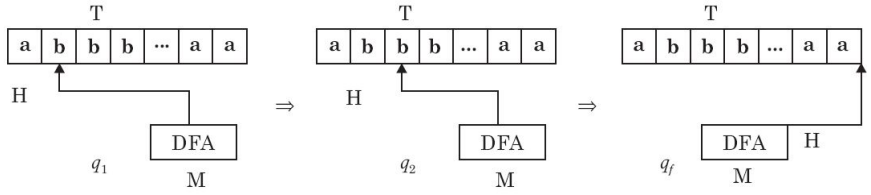
U teoriji izračunavanja, grani teorijske kompjuterske nauke, deterministički konačni automat (DFA[1]) - također poznat i kao deterministički konačni akceptor (DFA[2]), deterministička konačna mašina (DFSM[3]) ili deterministički automat konačnog stanja (DFSA[4]) - je mašina s konačnim stanjima koji prihvaća ili odbija zadani niz simbola, prolazeći kroz niz stanja jedinstveno određen nizom. Deterministički se odnosi na jedinstvenost proračunskog ciklusa.

DFA se odnosi na apstraktno posmatranje mašine čija je tranzicija jedna i samo jedna poslije čitanja niza ulaza sa svakog stanja. Apstraktno posmatarnje te mašine je prikazano sljedećom slikom.



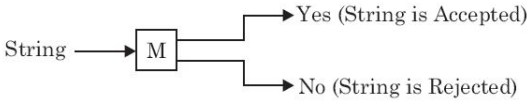
Sa M označavamo mašinu DFA koja ima traku T konačne dužine. Ćelije trake sadrže ulazne simbole stringa. Mašina ima glavu trake za čitanje (i samo čitanje) H i njeno kretanje je ograničenona striktno naprijed/desno. Jednom kada se pomakne naprijed/desno ne vraća se nazad/lijevo. Pretpostavimo da je automata M u početnom trenutku $t = 0$ (ili uopšte ne mjerimo vrijeme) u stanju q_0. Glava trake očitava trenutnio simbol a i automata ide na sljedeće stanje q_1. Sada je glava trake pokazuje simbol b, mašina očitava simbol b i ide na sljedeće stanje q_2.

[1]Deterministic finite automata.
[2]Deterministic finite acceptor.
[3]Deterministic finite state machine.
[4]Deterministic finite state automata.



Na ovaj način mašina skenira sve ulaze na ćelijama trake i dostiže zadju ćeliju. Pretpostavljano da, nakon nekog vremena, mašina M dostiže finalno stanje q_f, nakon skeniranja cijele trake i također pretpostavljanmo da mašina staje.

dakle, nakon skeniranja svih ulaza na traci kojaa sadrži string, mašina M ide u stanje koje je konačno stanje. To znači da je string prihvaćen ili je odbjen od mašine M (to znači da mašina M ne dostiže njeno krajnej stanje dok skenira cijeli string). Ovi mogući ishodi DFA su prikazani sa



Definicija 4.2.1. *Deterministički konačni automat (DFA) ima:*

(a) konačan skup stanja Q,

(b) konačan skup ulaznih simbola Σ,

(c) funkciju tranzicije δ koja definiše sljedeću tranziciju (pokret) nad ulaznim simbolom,

(d) početno stanje q_0, gdje je $q_0 \in Q$ (bilo koje stanje u skupu Q je pošetno stanje),

(e) skup prihvatljivih stanja (konačno stanje) F. Jedno ili više stanja mogu biti finalno stanje/stanja, za koji ej sigurno $F \subseteq Q$.

Dakle, deterministički konačni automat (DFA) je uređena petroka $M = (Q, \Sigma, \delta, q_0, F)$ gore definisanih objekata.

Primijetimo da je funkcija tranzicije ustvari preslikavanje $\delta : Q \times \Sigma \to Q$. Naprimjer, za stanje $q \in Q$ i simbol $a \in \Sigma$ vrijedi $\delta(q, a) \to p \in Q$. Mašina M je u stanju q i poslije očitavanja simbola a se pomiče na sljedeće stanje p koje može biti isto kao i stanje q, što znači da mašina ostaje u svom stanju uzimajući ulazni simbol, tojest $\delta(q, a) \to q$.

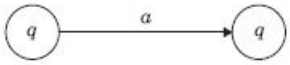
Očigledno je da DFA ima konačan broj stanja i tranzicije između stanja su definisane u skupu stanja nad skupom ulaznih simbola, što vraća skup stanja. DFA možemo predstaviti kroz dijagrame stanja i kroz tabele tranzicije.

4.2.1 Dijagrami stanja

U dijagramu stanja koristimo sljedeće oznake:

- Stanje je prikazano krugom. Pretpostavimo da je p stanje, tada ga predstavljamo sa

- Tranziciju među stanjima prikazujemo usmjerenim lukom između njih koji sadrži ulazni simbol. Naprimjer, $\delta(q, a) = p$ predstavljemo kao



  pri čemu su p i q stanja i pri čemu je luk, koji sadrži ulazni simbol a, prikazuje tranziciju iz stanja q u stanje p.

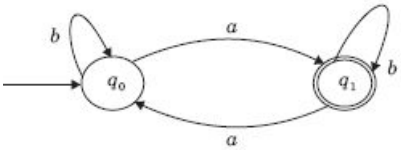
- Početno (inicijalno) stanje označavamo sa početnom strelicom. Naprimjer, ako je q početno stanje, tada ga predstavljamo kao

- Finalno (prihvatljivo) stanje označavamo sa duplim krugom, naprimjer ako je p finalno stanje, tada ga predstavljamo kao

Primjer 4.5. *Deterministički konačni automat (DFA)* $M = (Q, \Sigma, \delta, q_0, F)$ *ima skup stanja* $Q = \{q_0, q_1\}$*, skup ulaznih simbola* $\Sigma = \{a, b\}$*,* $F = \{q_1\}$ *i* q_0 *je početno stanje dok je funkcija tranzicije* δ *definisana kao*

$$\delta(q_0, a) = q_1;\ \delta(q_0, b) = q_0;\ \delta(q_1, a) = q_0;\ \delta(q_1, b) = q_1.$$

Dijagram ovog determinističkog konačnog automata M *predstavljamo sa*



■

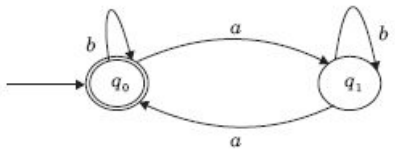
Odavdje vidimo da postoji tačno jedna i samo jedna tranzicija (izlazni luk) iz svakog stanja u svaki ulazni simbol, pa je prema tome automata 'određena'. U protivnom ako konačna automata ima više od jedne tranzicije7tranzicija (izlaznih lukova) iz stanja u jedan/isti simbol, onda za automatu kažemo da je 'neodređena'.

Primjer 4.6. *Deterministički konačni automat (DFA)* $M = (Q, \Sigma, \delta, q_0, F)$ *ima skup stanja* $Q = \{q_0, q_1\}$*, skup ulaznih simbola* $\Sigma = \{a, b\}$*, početno stanje* q_0 *dok skup finalnih*

stanja sadrži samo stanje q_0, tj. $F = \{q_0\}$ pri čemu je $F \subseteq Q$ dok je funkcija tranzicije δ definisana kao

$$\delta(q_0, a) = q_1;\ \delta(q_0, b) = q_0;\ \delta(q_1, a) = q_0;\ \delta(q_1, b) = q_1.$$

Dijagram ovog determinističkog konačnog automata M predstavljamo sa



Ovdje je početno stanje označeno sa strelicom također i finalno stanje označeno sa duplim krugom. Iz date funkcije tranzicije δ, možemo zaključiti sljedeće:

- *Početno stanje je i konačno stanje. Posmatrajmo specijalni string ϵ za koji vrijedi $\delta(q_0, \epsilon) = q_0$, tj. stanje ostaje nepromijenjeno.*
- *Bilo koji strng koji sadrži jedan ili više simbola b je također prihvatljiv, tj $\{b, bb, bbb, \ldots\}$.*
- *Ako string sadrži simbol a, tada mora sadržavati drugi simbol a i tako M vraća svoje finalno stanje q_0, tj. prihvatljivi stringovi su $\{aa, aaaa, aaaaaa, \ldots\}$.*
- *Ako je početni simbol nula, jedan ili više znakova b, tada su prihvatljivi stringovi $\{aa, aaaa, aaaaaa, \ldots\}$, $\{baa, baaaa, baaaaaa, \ldots\}$, $\{bbaa, bbaaaa, bbaaaaaa, \ldots\}$, $\ldots$, $\{bb \ldots baa, bb \ldots baaaa, bb \ldots baaaaaa, \ldots\}$.*
- *Ako bilo koji broj stringova b leži između a, tada su prihvatljivi stringovi*

$$\{aba, abba, \ldots, abababa, abbabbabba, \ldots, abababababa, abbabbabbabbabba, \ldots\}.$$

Dakle, bilo koji string koji sadrži paran broj simbola a je prihvatljiv za mašinu M. Prema tome, skup stringova prihvatljivih za mašinu M je

$$\{\epsilon, b, bb, bbb, \ldots, aa, aaaa, \ldots, aba, abba, \ldots, abababa, \ldots\}.$$

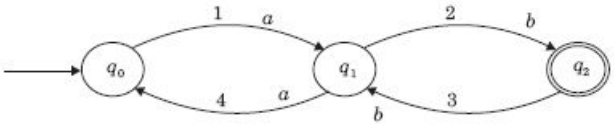
■

Primjer 4.7. *Konstruisati DFA koja prihvaća stringove sastavljene od neparnog broja simbola a i simbola b.* ■

Rješenje. Posmatrajmo mašinu $M = (Q, \Sigma, \delta, q_0, F)$ pri čemu je $\Sigma = \{a, b\}$. Pretpostavimo da je konačan skup stanja $Q = \{q_0, q_1, \ldots, q_i\}$, pri čemu je q_0 početno stanje i jedno (više) stanje je finalno stanje (stanja), tj. pripada skupu F .

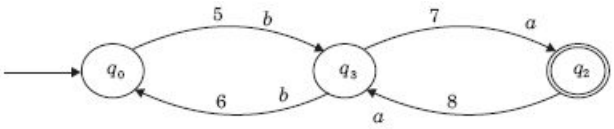
(a) Ako string počinje sa simbolom a (strelica 1), tada u sljedećem stanju q_1 mora biti neparan broj simbola b (1,3,5,...) i paran broj simbola a (2,4,6,...).

(b) Ako postoji string koji sadrži simbol a ispred kojeg je jedan b (strelica 2), tada se dostiže stanje q_2 što je prihvatljivo stanje. Također je tačno sa a-ove, $3a$-ove, $5a$-ove itd. ispred kojeg su b-ovi, $3b$-ovi, $5b$-ovi, itd.

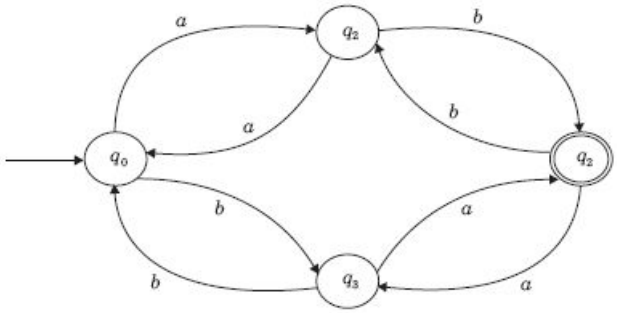


(c) Ako string počinje sa simbolom b (strelica 5), tada u sljedećem stanju q_3 mora biti neparan broj simbola a (1,3,5,...) i paran broj simbola b (2,4,6,...).

(d) Ako postoji string koji sadrži simbol b (strelica 5) iza kojeg slijedi a (strelica 7), tada se dostiže stanje q_2 što je prihvatljivo stanje, (koje je tačno za neparan broj a-ova i b-ova).



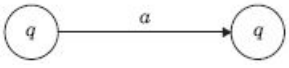
(e) Kombinujući prehodna dva dijagrama, konačni dijagram mašine $M = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta, q_0, \{q_2\})$ je dat kao



Slika 4.1: Dijagram mašine $M = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta, q_0, \{q_2\})$

4.2.2 Tabele tranzicije

Ponekad dijagrami tranzicije za mnoge DFA mogu biti jako komplikovani za predstavljanje pa postoji jednostavnije predstavljanje funkcija tranzicije sa tabelama. Vrste tabela sadrže stav stanja skupa Q a kolone tabele sadrže sve ulazne simbole skupa S. Naprimjer, posmatrajmo funkciju tranzicije $\delta(q, a) = p$. Njen dijagram tranzicije je



a tabela tranzicije je

Stanje	Ulazni simbol
	a
q	p

Ovo pokazuje da je mašina u stanju q i nakon očitanja simbola a njeno stanje se mijenja u p.

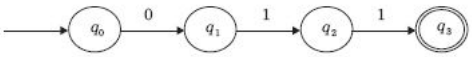
U tabeli tranzicije, početno stanje je označeno sa strelicom a finalno stanje je označeno ispunjenim kružićem. Naprimjer, mašina M na Slici 4.1 se prikazuje sljedećom tabelom tranzicije

Stanje	Ulazni simbol	
	a	b
$\longrightarrow q_0$	q_1	q_3
q_1	q_0	q_2
• q_2	q_3	q_1
q_3	q_2	q_1

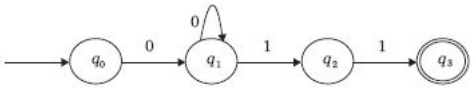
Primjer 4.8. *Konstruisati DFA koja prihvata string koji sadrži uzorak* 011. ∎

Rješenje. Konstruišimo DFA nad alfabetom $\Sigma = \{0, 1\}$ pretpostavljajući da je q_0 početno stanje.

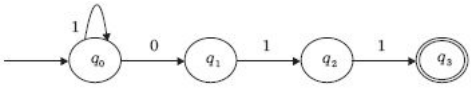
(a) Od stanja q_0 pa nadalje mora se utrošiti string 011 (koji je neophodan substring ili substring koji sadrži uzorak pronađen u svim prihvatljivim stanjima)



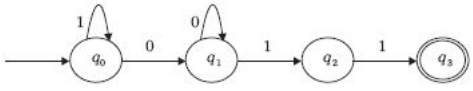
(b) Bilo koji string koji počinje sa simbolom 0 iza kojeg je bilo koji broj nula prije uzorka 011 je prihvatljiv $(011, 0011, 00011, \ldots)$. Prema tome, mora se desiti ponavljajuća tranzicija na stanju q_1 na simbolu 0.



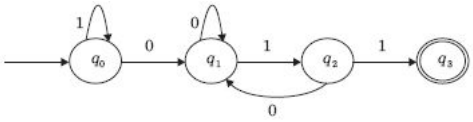
(c) Bilo koji string koji počinje sa simbolom 1 iza kojeg slijedi bilo koji broj jedinica prije uzorka 011 je prihvatljiv $(1011, 11011, 111011, \ldots)$. Prema tome, mora se desiti ponavljajuća tranzicija na stanju q_0 na simbolu 1.



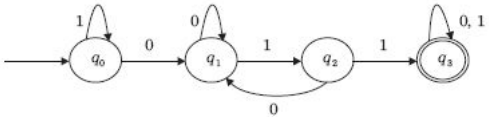
(d) Kombinujući (b) i (c) dobijamo traženi DFA



(e) Ako string koji sadrži 01 iza kojeg slijedi uzorak 011, tada iz stanja q_2 postoji luk koji vraća na stanje q_1 na ulazni simbol 0, tako da mašina dostiže prihvatljivo stanje q_3 nakon čitanja uzorka 011.



(f) Poslije uzorka 011 string može da sadrži bilo koji broj nula i/ili jedinica pa zbog toga postoji ponavljajući luk na stanju q_3 nad simbolima 0 ili 1.



Primijetimo da u ovoj mašini postoji jedan i samo jedan izlaz na svakom simbolu na svakom stanju, pa je mašina određena.

Konačno, DFA je $M = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, \{q_3\})$, gdje su funkcije δ prikazane u sljedećoj tabeli

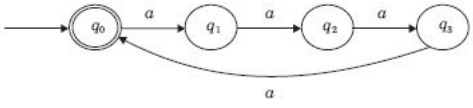
Stanje	Ulazni simbol	
	0	1
$\longrightarrow q_0$	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
• q_3	q_3	q_3

Primjer 4.9. *Konstruisati DFA nad alfabetom $\Sigma = \{a\}$ koja prihvaća stringove iz skupa $\{\epsilon, aaaa, aaaaaaaa, \ldots\}$, (nijedan ili višekratnik od 4 simbola a).* ■

Rješenje. Neka je M mašina i q_0 početno stanje.

(a) Ako je prvi simbol nula string ϵ, tada je početno stanje i krajnje stanje, pa će dijagram stanja biti

(b) Od stanja q_0 pa naprijed, postoji utrošak od 4 uzastopna simbola a kojeg slijedi jedan ili nijedan takav skup simbola a, tj.



Konačno, DFA je $M = (\{q_0, q_1, q_2, q_3\}, \{a\}, \delta, q_0, \{q_0\})$, gdje su funkcije δ prikazane u sljedećoj tabeli

Stanje	Ulazni simbol
	a
$\longrightarrow \bullet q_0$	q_1
q_1	q_2
q_2	q_0
q_3	q_0

4.2.3 δ-glava

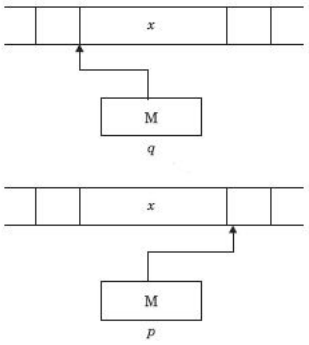
Umjesto definisanja ponašanja funkcije tranzicije nad jednim simbolom može se također zahtijevati i ponašanje te funkcije nad proizvoljnim stringom tako da mašina ne čita samo jedan simbol već niz simbola ili string.

Definicija 4.2.2. *Neka je string definisan nad alfabetom Σ, tj. skup svih mogućih stringova je u Σ^*. Tada je δ-glava definisana kao funkcija*

$$\hat{\delta} : Q \times \Sigma^* \to Q,$$

odnosno kažemo da je δ-glava funkcija tranzicije koja slika stanje sa stringom u stanje.

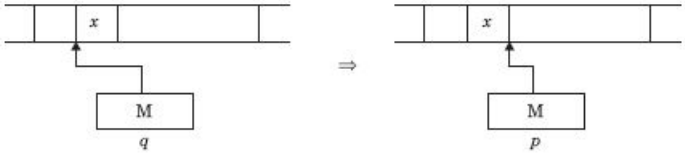
Neka je automata M u stanju q i poslije očitovanja stringa x (ćelije trake sadrže string x), automata dostiže stanje p, kao što je prikazano na slici



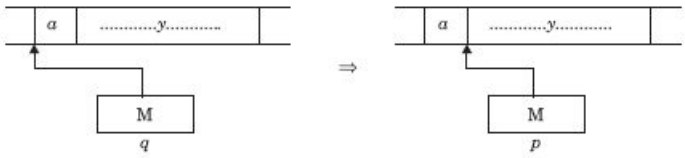
Dakle, ponašanje δ-glave nad stringom x je dato sa $\hat{\delta}(q, x) = p$, gdje string x može biti i jedan simbol.

Uočimo da δ-glava ima sljedeće osobine.

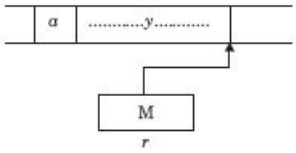
(a) Za proizvoljno stanje q, ako je ulazni string nula string ϵ, tada stanje automate ostaje nepromijenjeno, tj. vrijedi $\hat{\delta}(q, \epsilon) = q$. A ako je string jedan simbol, odnosno dužine jedan, tada su δ-glava i δ funkcija iste, tj. $\hat{\delta}(q, a) = \delta(q, a) = p$.



(b) Ako string x nije jedan simbol već pretpostavimo da je formiran od alfabeta a i substringa y, tj. $x = ay$, tada je $\hat{\delta}(q, ay) = \hat{\delta}(\delta(q, a), y)$. Naprimjer ako je $x = 1011$, tada je $a = 1$ i $y = 011$.



Ovdje je $\delta(q, a) = p$ i poslije očitovanja substringa y automata dostiže stanje r, tj. $\hat{\delta(q, x)} = r$



Na ovaj način automata završava tranziciju nad stringom x.

Primjer 4.10. *Mašina* $M = (\{q_0, q_1\}, \{a\}, \delta, q_0, \{q_1\})$ *ima sljedeće osobine tranzicije*

Ispitati ponašanje DFA nad stringom aaa. ■

Rješenje. Pretpostavimo da je automata u početnom stanju q_0 i ispitajmo njeno ponašanje nad stringom aaa, tj.

$$\hat{\delta}(q_0, aaa) = \hat{\delta}(\delta(q_0, a), aa) = \hat{\delta}(q_1, aa) = \hat{\delta}(\delta(q_1, a), a) = \hat{\delta}(q_0, a)$$
$$= \hat{\delta}(q_0, a\epsilon) = \hat{\delta}(\delta(q_0, a)) = \hat{\delta}(q_1, \epsilon) = \{q_1\}.$$

Budući da je stanje (q_1) prihvatljivo stanje, tada je i string aaa prihvatljiv za DFA.

Primjer 4.11. *Konstruisati DFA M koja prihvata skup stringova koji imaju neparan broj simbola a i neparan broj simbola b.* ■

Rješenje. Mašina M koja prihvaća stringove neparan broj simbola a i neparan broj simbola b je $M = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta, q_0, \{q_2\})$ i funkcije tranzicije su opisane tabelom

Stanje	Ulazni simbol	
	a	b
$\longrightarrow q_0$	q_1	q_3
q_1	q_0	q_2
• q_2	q_3	q_1
q_3	q_2	q_1

Sada definišemo pokrete DFA nad stringom *abaabb* koji mora biti prihvatljiv tj. nakon očitanja cijelog stringa mašina dostiže finalno stanje $\{q_2\}$. Dakle,

$$\begin{aligned}\hat{\delta}(q_0, abaabb) &= \hat{\delta}(\delta(q_0, a), baabb) = \hat{\delta}(q_1, baabb) = \hat{\delta}(\delta(q_1, b), aabb) = \hat{\delta}(q_2, aabb) \\ &= \hat{\delta}(\delta(q_2, a), abb) = \hat{\delta}(q_3, abb) = \hat{\delta}(\delta(q_3, a), bb) = \hat{\delta}(q_2, bb) = \hat{\delta}(\delta(q_2, b), b) \\ &= \hat{\delta}(q_1, b) = \hat{\delta}(q_1, b\epsilon) = \hat{\delta}(\delta(q_1, b), \epsilon) = \hat{\delta}(q_2, \epsilon) = \delta(q_2, \epsilon) = \{q_2\}\end{aligned}$$

gdje je $\{q_2\}$ prihvatljivo stanje za mašinu M.

4.2.4 Jezik deterministiˇckih konaˇcnih automata

Poznavajući ponašanje automata nad proizvoljnim stringom i nad skupom stringova, možemo definisati i jezik deterministiˇckih konaˇcnih automata. Neka je M DFA, tada je jezik prihvaćen od M, u oznaci L_M, je

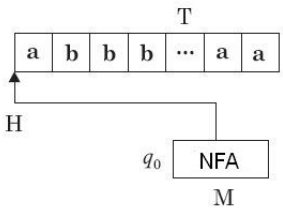
$$L_M = \{x : x \in \Sigma^* \text{ i } \hat{\delta}(q_0, x) \in F\}.$$

Ustvari, kažemo da proizvoljni string x iz skupa Σ^* će biti jezik mašine M ako iz početnog stanja poslije čitanja kompletnog stringa x, dostiže svoje finalno stanje. Sada vidimo da ako je Σ alfabet tada je skup svih mogućih stringova formiranih nad Σ ustavri Σ^*. U skupu Σ^* postoje dvije moguće klase jezika, tj. L_M i $\Sigma^* - L_M$, pri čemu L_M sadrži onaj skup stringova koji su prihvatljivi za automat M i $\Sigma^* - L_M$ preostale stringove koji su odbačeni od automata M. Zbog toga su L_M i $\Sigma^* - L_M$ dva disjunktna skupa.

4.3 Nedeterministički konačni automat (NFA)

U prethodnom dijelu smo razmatrali deterministiˇcke konaˇcne automate DFA čije su tranzicije stanja određene, što znači da postoji jedan i samo jedan izlazni luk iz svakog stanja na ulaznom simbolu. Zato možemo reći da DFA daje statički pogled na konaˇcne automate.

Sada ćemo razmatrati fleksibilni/dinamički pogled na konaˇcne automate čije tranzicije na stanjima nisu deterministiˇcke prirode (nedeterministički su), što znači da postoji mogućnost više tranzicija na nekom ulaznom simbolu iz jednog stanja. Automate takve kategorije su poznate kao neterministiˇcke konaˇcne automate (NDFSA/NDFSM/NDFA/NFA). Obilježje dinamičnosti predstavljeno kod konaˇcnih automata daje više snage konaˇcnim automatima. Abstraktni pogled na NFA, sličan je pogledu na DFA, je prikazan na sljedećoj slici



Definicija 4.3.1. *Nedeterministički konačni automat (NFA) ima:*

(a) konačan skup stanja Q,

(b) konačan skup ulaznih simbola Σ,

(c) početno stanje q_0, gdje je $q_0 \in Q$,

(d) skup prihvatljivih stanja (konačno stanje) $F \subseteq Q$,

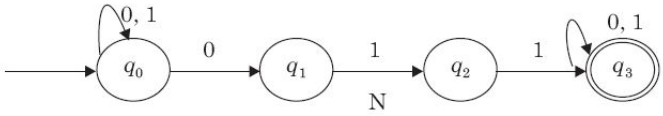
(e) funkciju tranzicije δ koja definiše sljedeću tranziciju iz trenutnog stanja nad ulaznim simbolom (koja ne doseže stanje ili doseže jedno ili više stanja). Da ne doseže stanje znači da nema izlaznog luka iz trenutnog stanja na tom simbolu, odnosno da nema tranzicije iz stanja definisanog nad tim simbolom. Tranzicija na jednom stanju znači da postoji jedinstvena tranzicija iz stanja nad tim simbolom, tj. postoji samo jedan izlazni luk iz tog stanja. Tranzicija na više stanja znači da postoji mogućnost dvije ili više tranzicija na tom stanju nad tsimbolom, tj. da postoje dva ili više izlazna luka iz tog stanja.

Dakle, nedeterministički konačni automat (NFA) je uređena petroka $M = (Q, \Sigma, \delta, q_0, F)$ gore definisanih objekata. Funkcija tranzicije je preslikavanje stanja Q sa ulaznim simbolom na partitivni skup skupa Q, tj.

$$\delta : Q \times S \to P(S).$$

Funkcija tranzicije vraća stanje/stanja koja su u partitivnom skupu skupa Q, pa je konačno stanje F je također podskup od skupa Q. Partitivni podskup skupa Q uključuje i Φ (element bez stanja), tj. Φ definiše tranziciju bez stanja na bilo kojem simbolu. Dinamičnost prirode NFA je razlog što funkcija δ vraća stanje koje je u skupu $P(S)$. Tranzicijski luk na jednom simbolu završava na jednom, dva ili više stanja ili možda na Φ. Ove mogućnosti skupa stanja su uključene u skupu $P(S)$.

Predstavljanje NFA je slično predstavljanju DFA, tj. NFA možemo predstaviti ili pomoću dijagram stanja ili kroz tabelu tranzicija. Na sljedećoj slici smo konstruisali NFA koja prihvata sve stringove koji sadrže podstring 011 ili uzorak simbola 011.



Ova NFa se može predstavti kao $N = (\{q_0, q_1, q_2, q_3\}, 0, 1, \delta, q_0, \{q_3\})$, dok su funkcije tranzciije definisane na sljedeći način:

(a) $\delta(q_0, 0) = \{q_0, q_1\}$,

(b) $\delta(q_0, 1) = \{q_0\}$,

(c) $\delta(q_1, 0) = \Phi$,

(d) $\delta(q_1, 1) = \{q_2\}$,

(e) $\delta(q_2, 0) = \Phi$,

(f) $\delta(q_2, 1) = \{q_3\}$,

(g) $\delta(q_3, 0) = \{q_3\}$,

(h) $\delta(q_3, 1) = \{q_3\}$.

4.3.1 δ-glava

Poput δ-glave DA, δ-glava NFA definiše ponašanje funkcije tranzicije nad proizvoljnim stringom. Pretpostavimo da je string definsian nad alfabetom Σ. Tada je skup svih mogućih stringova skup Σ^*. Tada, δ-glava je definisana kao

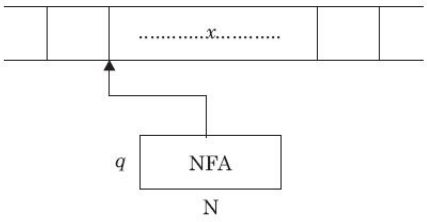
$$\hat{\delta} : Q \times \Sigma^* \to P(Q)$$

tj. δ-glava je tranzicijska funkcija koja preslikava stanje iz Q i string iz Σ^* u partitivnio skup skupa Q.

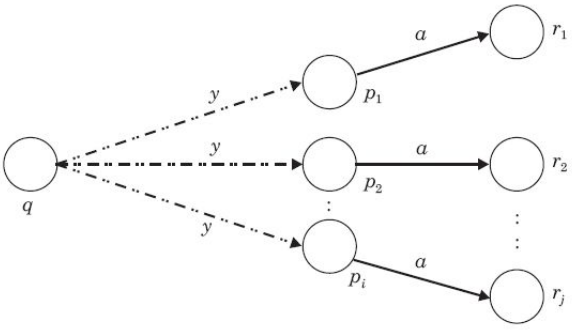
Neka je N NFA čije je trenutno stanje q i njena traka sadrži string x prikazan kao na slici. Tada je ponašanje δ-glave nad stringom x opisano ka0:

(a) ako je x nula string (ϵ), tada je $\hat{\delta}(q, \epsilon) = \{q\}$, tj. stanje ostaje nepromijenjeno;

(b) ako je x sastavljen od dva ili više simbola, tada dekompoziramo string x u podstring y i simbol a tako da je $x = ya$ i onda je $\hat{\delta}(q, x) = \hat{\delta}(q, ya) = \delta(\hat{\delta}(q, y), a)$.



Pretpostavimo da, nakon prihvatanja podstringa y, iz stanja q se mogu dostići stanja $p_1, p_2, \ldots, p_i$, tj. da je $\hat{\delta}(q, y) = \{p_1, p_2, \ldots, p_i\}$, kao što je prikazano na slici



Odavdje je

$$\delta(\hat{\delta}(q, y), a) = \delta(\{p_1, p_2, \ldots, p_i\}, a)$$

$$= \delta(p_1, a) \cup \delta(p_2, a) \cup \cdots \delta(p_i, a) = \bigcup_{k=1}^{i} \delta(p_k, a) = \{r_1, r_2, \ldots, r_j\}$$

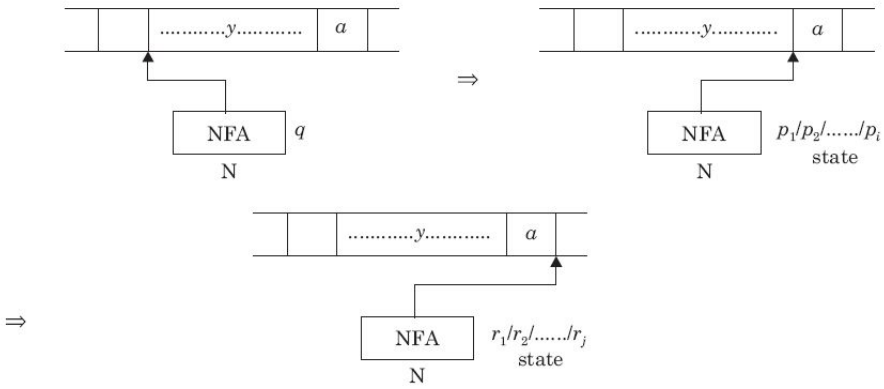
Zbog dinamičnosti priirode NFA, imamo sljedeće slučajeve:

(a) $j = i$, tada svako stanje ima jednu tranziciju na svakom simbolu, pa se vraća isti broj stanja,

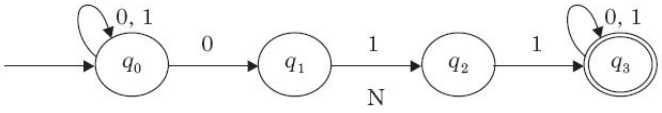
(b) $j = 0$, tada nema tranzicijskog luka nad simbolom a na stanju p_k,

(c) $j > i$ ili $j < i$, u zavisnosti od prirode tranzicije na stanju p_k nad simbolom a.

Abstraktni pogled na automatu za vrijeme skeniranja stringa $x = ya$ je prikazan sa sljedećom slikom



Primjer 4.12. *Posmatrajmo automatu N*



Provjeriti da li prihvata string 101101. ∎

Rješenje. Automata N se može definisati kao $M = (\{q_0, q_1, q_2, q_3\}, 0, 1, \delta, q_0, \{q_3\})$ gdje su funkcije tranzicija δ prikazane u sljedećoj tabeli tranzicija

Stanje	Ulazni simbol	
	0	1
$\longrightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	Φ	$\{q_2\}$
q_2	Φ	$\{q_3\}$
$\bullet q_3$	$\{q_3\}$	$\{q_3\}$

Provjerimo ponašanje automate N nad stringom 101101 sa početnim stanjem q_0.

$$\hat{\delta}(q_0, 101101) = \hat{\delta}(\delta(q_0, 1), 01101) = \hat{\delta}(q_0, 01101) = \hat{\delta}(\delta(q_0, 0), 1101) = \hat{\delta}(\{q_0, q_1\}, 1101)$$
$$= \hat{\delta}(q_0, 1101) \cup \hat{\delta}(q_1, 1101)$$

Dalje je

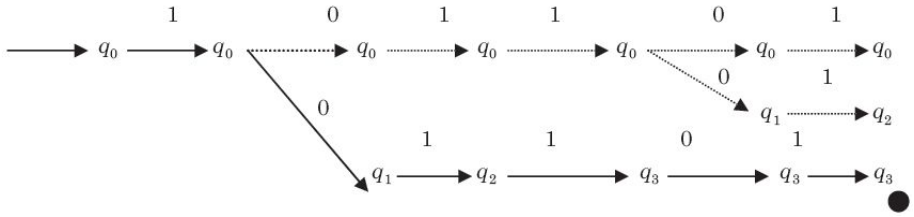
$$\begin{aligned}\hat{\delta}(q_0,1101) &= \hat{\delta}(\delta(q_0,1),101) = \hat{\delta}(q_0,101) = \hat{\delta}(\delta(q_0,1),01) = \hat{\delta}(q_0,01) = \hat{\delta}(\delta(q_0,0),1)\\ &= \hat{\delta}(\{q_0,q_1\},1) = \hat{\delta}(\{q_0,q_1\},1.\epsilon) = \hat{\delta}(q_0,1.\epsilon)\cup\hat{\delta}(q_1,1.\epsilon)\\ &= \hat{\delta}(\delta(q_0,1),\epsilon)\cup\hat{\delta}(\delta(q_1,1),\epsilon) = \hat{\delta}(q_0,\epsilon)\cup\hat{\delta}(q_2,\epsilon) = \{q_0\}\cup\{q_2\} = \{\mathbf{q_0},\mathbf{q_2}\}\end{aligned}$$

odakle vidimo da nijedno stanje nije prihvatljivo stanje.

Također,

$$\begin{aligned}\hat{\delta}(q_1,1101) &= \hat{\delta}(\delta(q_1,1),101) = \hat{\delta}(q_2,101) = \hat{\delta}(\delta(q_2,1),01) = \hat{\delta}(q_3,01)\\ &= \hat{\delta}(\delta(q_3,0),1) = \hat{\delta}(q_3,1) = \hat{\delta}(q_3,1.\epsilon) = \hat{\delta}(\delta(q_3,1),\epsilon) = \hat{\delta}(q_3\epsilon) = \{\mathbf{q_3}\}\end{aligned}$$

što je prihvatljivo stanje. Prema tome, automata N će se kretati nad stringom 101101 sljedećim putanjama



Jedina prihvatljiva putanja iz početnog stanja q_0 je označeno punom linijom. Duž ove putanje automata N dostiže prihvatljivo stanje q_3 na kraju stringa 101101.

4.3.2 Jezik NFA

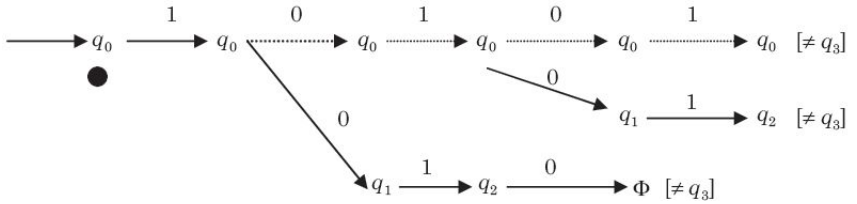
Poslije utvrđivanja ponašanja NFA nad proizvoljnim stringom, sada lako možemo definisati jezik NFA. Neka je $N = \{Q, \Sigma, \delta, q_0, F\}$ NFA, tada je L_N jezik prihvaćen od NFA gdje je

$$L_N = \{x : x \in \Sigma^* \text{ i } \hat{\delta}(q_0,x) \in P(Q) \text{ tako da } \hat{\delta}(q_0,x) \cap F \neq \Phi\}.$$

To znači da jezik L_N sadrži proizvoljni string x izabran iz skupa svih mogućih stringova tako da NFA N, poslije očitovanja stringa x, dostiže stanje koje je u partitivnom skupu skupa Q i takav da nije odvojen od skupa F, tj. $\hat{\delta}(q_0,x) \cap F \neq \Phi$. Odnosno, postoji bar jedno stanje koje je i u skupu F i u skupu $P(Q)$.

Primjer 4.13. *Za NFA iz Primjera 4.12, testirati string* 10101 *koji nije u jeziku od* N.
■

Rješenje. Neka je $x = 10101$. Ako je x jezik od N, tj. $x \in \Sigma^*$ gdje je $\Sigma = \{0,1\}$, tada nakon čitanja stringa x, NFA N dostiže svoje finalno stanje, tj. bilo koje stanje koje pripada skupu $P(Q)$ koji sadrži finalno stanje $\{q_3\}$. Sada možemo konstruisati kretanje NFA nad stringom x koje je prikazano na slici



Budući da je $\hat{\delta}(q_0, 10101) = \Phi$ ili $\{q_0\}$ ili $\{q_2\}$, tada nijedan skup ne sadrži $\{q_3\}$. Prema tome, $\hat{\delta}(q_0, 10101)_{\cap}\{q_3\} = \Phi$, pa string x nije prihvatljiv sa NFA N.

Bibliografija

[1] Y. N Shing: *Mathematical Foundation of Computer Science*, NEW AGE INTERNATIONAL (P) LIMITED, PUBLISHERS, 2005.

[2] T. Sundstrom: *Mathematical Reasoning - Writing and Proof*, Grand Valley State University Allendale, MI, 2013, 2019.

[3] E. Lehman, F.T. Leighton, A.R. Meyer: *Mathematics for Computer Science*, This work is available under the terms of the Creative Commons Attribution-ShareAlike 3.0 license, 2017.