Analiza algoritama

ispitni rad iz predmeta Složenost izračunavanja

Milan Banković 2021/06

07. 02. 2008.

Sažetak

Ovaj rad predstavlja kratak pregled nekih metoda analize algoritama, sa kojima se autor rada upoznao u okviru kursa „Složenost izračunavanja", na prvoj godini postdiplomskih studija na Matematičkom Fakultetu u Beogradu.

U okviru ovog rada ćemo najpre kroz primere podrobnije objasniti značaj analize prosečnog slučaja, kao i njen odnos sa analizom najgoreg slučaja. Nakon toga ćemo uspostaviti vezu analize prosečnog slučaja sa prebrojavanjem skupova kombinatornih objekata. U nastavku ćemo se upoznati sa rekurentnim relacijama, a zatim i sa funkcijama generatrisama kao osnovnim alatima koji se primenjuju u analizi algoritama i teoriji prebrojavanja. Daćemo primere upotrebe funkcija generatrisa u rešavanju rekurentnih relacija, ali i način da se rekurentne relacije u analizi algoritama primenom funkcija generatrisa potpuno zaobiđu. U tom svetlu, upoznaćemo simbolički metod. Na kraju ćemo se upoznati sa bivarijantnim funkcijama generatrisama i primenama ovih funkcija u analizi algoritama.

1 Uvod

Analiza algoritama je neophodna, kako u teorijskim razmatranjima, kada pokušavamo da odredimo složenost i težinu nekog problema, tako i u praktičnim primenama, kada želimo da što preciznije procenimo očekivano vreme izvršavanja nekog programa. Osim vremena, postoje i drugi resursi koji mogu biti od značaja (utrošena memorija, na primer). Osnovni cilj analize algoritama je utvrđivanje količine određenog resursa koja je potrebna za izvršavanje algoritma, na osnovu čega se može proceniti upotrebljivost algoritma za određenu svrhu. Takođe, analiza složenosti algoritma omogućava upoređivanje jednog algoritma sa drugim poznatim algoritmima iste namene. Najzad, analizom nam algoritmi postaju jasniji, a njihovi eventualni nedostaci uočljiviji, što dovodi do efikasnijih i elegantnijih rešenja.

Resursi potrebni za izvršavanje nekog algoritma obično zavise od veličine ulaza, pa se zato i složenost algoritma predstavlja kao funkcija te veličine. Međutim, vrlo često potrebni resursi zavise i od toga koja se konkretna istanca ulaza određene veličine razmatra. Neke instance mogu biti povoljnije za algoritam, dok neke mogu da zahtevaju više resursa[1]. U tom slučaju odgovor na pitanje: „Koliko je vremena potrebno za izvršenje algoritma, ako je ulaz veličine n?" nije jednoznačan, pa se moramo opredeliti za neki konkretan slučaj ulaza, ili nekako drugačije definisati veličinu koja nam je od interesa u analizi. Ovde postoji više različitih pristupa, u zavisnosti od ciljeva koje želimo da postignemo. Jedan pristup, često korišćen u teorijskim razmatranjima, je da se razmatra najgori mogući slučaj, kao i ponašanje algoritma za velike ulaze (tzv. asimptotska analiza). Ovaj pristup nam omogućava da procenimo upotrebljivost algoritma i da klasifikujemo algoritme po svojoj složenosti. Takođe, na ovaj način se može analizirati i složenost samog problema, kroz utvrđivanje donjih granica složenosti algoritama koji rešavaju dati problem. S druge strane, ovakva analiza najčešće nije dovoljna u praksi, zato što se njom prikrivaju detalji, pre svega zbog zanemarivanja konstantnih faktora, koji u asimptotskoj analizi nemaju značaj. Naime, u praksi nam je bitnije prosečno vreme izvršavanja, a ne izvršavanje u najgorem slučaju – zato nam je povoljnije da utvrdimo očekivano vreme, kao i standardno odstupanje. U sledećim odeljcima dajemo primere analiza najgoreg i prosečnog slučaja, kao i njihov odnos.

[1] U daljem tekstu ćemo bez ograničenja opštosti podrazumevati da je resurs koji se razmatra vreme izvršavanja.

1.1 Analiza najgoreg slučaja

Analiza najgoreg slučaja nam pruža garanciju da će se algoritam izvršiti u datom vremenu bez obzira na konkretnu istancu ulaza za koju se algoritam izvršava. Neka je $f(n)$ vreme dovoljno za izvršavanje algoritma u najgorem slučaju među svim dopustivim ulazima veličine n. Tada kažemo da algoritam ima *složenost* $f(n)$. Zbog toga se analiza najgoreg slučaja često naziva i *analiza složenosti* datog algoritma. Prilikom utvrđivanja složenosti algoritma ne insistiramo na detaljima, već nas pre svega zanima „red veličine" složenosti. Ovo olakšava upoređivanje algoritama i procenu upotrebljivosti algoritma.

Definicija 1.1. *Ako je data funkcija $f(n)$, tada:*

- $O(f(n))$ *označava skup svih funkcija $g(n)$ takvih da je $|g(n)/f(n)|$ ograničeno odozgo kada $n \to \infty$;*
- $\Omega(f(n))$ *označava skup svih funkcija $g(n)$ takvih da je $|g(n)/f(n)|$ ograničeno odozdo strogo pozitivnim brojem, kada $n \to \infty$;*
- $\Theta(f(n))$ *označava skup svih funkcija $g(n)$ takvih da je $|g(n)/f(n)|$ ograničeno i odozdo i odozgo kada $n \to \infty$.*

Oznaka $O(f(n))$ označava da je neka funkcija asimptotski ograničena *odozgo* funkcijom $f(n)$, uz zanemarivanje konstantnih faktora. Drugim rečima, funkcija je „reda veličine" *najviše* $f(n)$. Na primer, funkcija n^2+3n+1 pripada klasi $O(n^2)$, ali i funkcija $6n-1$ takođe pripada toj klasi. Ovakva notacija se obično koristi u analizi složenosti konkretnog algoritma, za utvrđivanje asimptotskog gornjeg ograničenja vremena izvršavanja algoritma za ulaze veličine n.

Oznaka $\Omega(f(n))$ označava asimptotsko ograničenje *odozdo* funkcijom $f(n)$, odnosno govori da je data funkcija „reda veličine" *najmanje* $f(n)$. Na primer, klasi $\Omega(n^2)$ pripada funkcija $n^2 - 5n$, ali i funkcija $n^3 + 2$. Ova notacija se najčešće koristi za opisivanje donjeg ograničenja složenosti svih algoritama koji rešavaju određeni problem, čime se suštinski odozdo ocenjuje složenost samog problema. Tada se kao osnovni cilj nameće pronalaženje konkretnog algoritma koji rešava dati problem, a čija se složenost poklapa sa prethodno utvrđenom donjom granicom. Tada kažemo da sam problem ima datu složenost.

Oznaka $\Theta(f(n))$ označava da je data funkcija „reda veličine" *tačno* $f(n)$. Na primer, funkcija $n(n+1)/2$ pripada klasi $\Theta(n^2)$, a funkcija $2n-1$ pripada klasi $\Theta(n)$. Ovakva notacija se obično koristi za opisivanje klase složenosti samog problema – kada se poklopi donje ograničenje složenosti svih potencijalnih algoritama koji rešavaju dati problem sa gornjim ograničenjem

složenosti nekog konkretnog algoritma koji taj problem rešava, posao analize tog problema sa teorijskog aspekta je završen.

1.1.1 Primer analize najgoreg slučaja – sortiranje niza

Primer problema na kome možemo demonstrirati značaj analize složenosti najgoreg slučaja je problem sortiranja niza.

Problem. *Za dati niz različitih celih brojeva dužine n, permutovati elemente, tako da novodobijeni niz bude u rastućem (opadajućem) poretku.*

Postoji više različitih pristupa ovom problemu, ali ćemo se mi ovde zadržati na metodama sortiranja koje su *zasnovane na upoređivanju elemenata*. Ideja je da se na osnovu definisane relacije poretka utvrđuje odnos između pojedinih elemenata niza, na osnovu čega se elementi po potrebi permutuju.

Jedan algoritam ovog tipa je tzv. *Sortiranje Objedinjavanjem* (eng. *Merge Sort*). Ovaj algoritam je tipičan primer tehnike *„zavadi pa vladaj"* (eng. *divide-and-conquer*) – rekurzivno se sortiraju leva i desna polovina niza, nakon čega se sortirani nizovi objedinjuju u jedan sortirani niz.

Teorema 1.1 (Sortiranje objedinjavanjem)**.** *Za sortiranje niza od n elemenata metodom Sortiranja objedinjavanjem, dovoljno je $n \lg n + O(n)$ upoređivanja.*

Gornji algoritam je dakle složenosti $O(n \log n)$. Primetimo da ne navodimo osnovu logaritma, zato što se logaritmi sa različitim osnovama razlikuju za konstantan faktor, što ne utiče na asimptotsku složenost. Takođe, podrazumeva se da se radi o najgorem slučaju, mada u slučaju sortiranja objedinjavanjem potrebno vreme zavisi samo od veličine ulaza, a ne i od konkretne istance ulaza date veličine. Ovo nije slučaj sa većinom drugih algoritama.

Iz teoreme 1.1 ne sledi da ne postoje algoritmi za sortiranje koji su zasnovani na upoređivanju i koji su manje složenosti od sortiranja objedinjavanjem. Dakle, mi za sada imamo samo gornju granicu za konkretan algoritam, ali ne i donju granicu za čitav skup algoritama sortiranja zasnovanih na upoređivanju. Saznanje o donjoj granici daje nam sledeća teorema.

Teorema 1.2 (Složenost sortiranja)**.** *Svako sortiranje zasnovano na upoređivanju elemenata mora koristiti najmanje $\lceil \lg n! \rceil > n \lg n - n/(\ln 2)$ upoređivanja za bar neku instancu ulaza veličine n.*

Na osnovu teoreme 1.2 zaključujemo da je sortiranje objedinjavanjem optimalan algoritam za sortiranje zasnovan na upoređivanju, u smislu da ne postoji algoritam za sortiranje koji je asimptotski brži od njega. Možemo da zaključimo da je sortiranje problem složenosti $\Theta(n \log n)$. Ovim je proučavanje složenosti sortiranja zasnovanog na upoređivanju sa teorijskog aspekta završeno – pronašli smo algoritam koji se po složenosti poklapa sa dokazanom donjom granicom.

Međutim, i dalje postoje pitanja na koja nismo dobili odgovor. Da li postoje i drugi algoritmi za sortiranje asimptotske složenosti $O(n \log n)$? Ako postoje, koji je među njima u praksi najbrži ? Da li postoji algoritam koji nije optimalan, ali se u praksi ipak bolje ponaša u prosečnom slučaju ? Za odgovore na sva ova pitanja potrebna nam je preciznija analiza.

1.2 Analiza prosečnog slučaja

Analiza prosečnog slučaja nam u praksi daje bolju predstavu o tome koliko će vremena biti potrebno konkretnom računaru za izvršavanje nekog algoritma. Analiza najgoreg slučaja je previše pesimistična i, mada daje garancije da će se algoritam završiti u određenom broju koraka bez obzira na ulaznu istancu date veličine, ipak prestrogo ocenjuje algoritam, i ponekad ga nepotrebno diskvalifikuje. Postoje algoritmi koji nisu optimalne složenosti, ali se u proseku izvršavaju znatno brže nego što bi se na osnovu njihove složenosti zaključilo, a ponekad brže i od pojedinih algoritama optimalnih za dati problem. O tome govori primer u nastavku.

1.2.1 Primer analize prosečnog slučaja – sortiranje niza

U ovom odeljku dajemo primer analize prosečnog slučaja algoritma *Brzog sortiranja* (eng. *QuickSort*). Ovaj algoritam se zasniva na tehnici *„zavadi pa vladaj"*, pri čemu se podela niza na dva dela vrši na osnovu jednog odabranog elementa, takozvanog *pivota*: svi elementi koji su manji od pivota premeštaju se levo od pivota, dok se elementi veći od pivota premeštaju desno od pivota. Nakon toga se pivot nalazi na pravom mestu, a delovi niza levo i desno od pivota se rekurzivno sortiraju.

Problem ovakvog pristupa je to što delovi niza nakon podele ne moraju biti jednake veličine. Kako će niz biti particionisan zavisi od same ulazne istance, tj. od permutacije elemenata niza.[2] Može se utvrditi da najgori slučaj nastupa kada je pivot najmanji ili najveći element niza – tada se jedan

[2]Po ovome se *QuickSort* razlikuje od *MergeSort*-a, kod koga potrebno vreme zavisi jedino od veličine ulaza.

deo niza degeneriše u prazan niz, dok je drugi deo samo za jedan manji od početnog niza. U tom slučaju imamo sledeću rekurentnu relaciju:

$$T_n = n + 1 + T_{n-1}$$

uz $T_0 = 0$, gde je T_i broj koraka dovoljan za izvršavanje algoritma u najgorem slučaju za ulaze veličine i. Rešavanjem ove jednačine dobijamo:

$$T_n = \sum_{1 \le k \le n} (k+1) = \frac{(n+1)(n+2)}{2} - 1 = O(n^2)$$

što znači da ovaj algoritam nije optimalan algoritam sortiranja. Ovakva analiza bi diskvalifikovala algoritam brzog sortiranja kao neefikasan.

Međutim, ispostavlja se da se ovaj algoritam u praksi ponaša bolje od svih drugih algoritama sortiranja zasnovanih na upoređivanju. To nas navodi na ideju da se možda ulazne istance koje dovode do kvadratnog vremena ne pojavljuju toliko često i da se zbog toga u proseku ovaj algoritam ipak ponaša kao da je optimalan. Analiza prosečnog slučaja nam zaista daje takav odgovor.

Teorema 1.3 (Brzo sortiranje)**.** *Za sortiranje niza dužine n čiji su elementi različiti i slučajno raspoređeni, algoritmu Brzog sortiranja potrebno je u proseku*

$$C_n = 2(n+1)(H_{n+1} - 1) \approx 2n \ln n - 0.846n$$

upoređivanja, gde je $H_n = \sum_{1 \le k \le n} \frac{1}{k}$.

Dakle, u proseku je za *QuickSort* potrebno $O(n \log n)$ upoređivanja. Napomenimo samo da se gornji rezultat odnosi na implementaciju algoritma u kojoj se za pivot uvek uzima krajnji desni element. Postoje i druge varijacije, koje mogu dovesti do nešto drugačijih rezultata[3]. Rezultat iz gornje teoreme sledi iz relacije:

$$C_n = n + 1 + \frac{1}{n} \sum_{1 \le j \le n} (C_{j-1} + C_{n-j}), \quad \text{za} \quad n > 0$$

uz $C_0 = 0$. Ova relacija važi pod pretpostavkom da su sve permutacije elemenata niza različite i jednako verovatne (otuda faktor $1/n$ ispred sume, to je verovatnoća da je krajnji desni element j-ti po veličini, za svako j). Sabirak $n + 1$ je broj upoređivanja u prvom stadijumu particionisanja (srazmeran je

[3]Na primer, izabrati tri elementa i srednji po veličini uzeti za pivot, čime se izbegava najgori slučaj.

dužini niza). Drugačiji model ulaza doveo bi do drugačije rekurentne relacije. Rešavanjem gornje relacije dobija se rezultat iz teoreme.

Ono što nam pokazuje teorema 1.3 je da algoritam brzog sortiranja u prosečnom slučaju daje rezultate koji su asimptotski jednaki optimalnim. Ispostavlja se u praksi da implementacije brzog sortiranja daju najbolje rezultate među svim algoritmima sortiranja koji su zasnovani na upoređivanju. Dakle, bila bi velika greška da smo na osnovu analize najgoreg slučaja odbacili ovaj algoritam kao neefikasan.

Sada nam je jasno koliko je analiza prosečnog slučaja značajna u praksi. Ono što je ostalo nejasno u gornjem primeru je kako doći do odgovarajuće rekurentne relacije i kako je rešiti. Da li postoji neki sistematski pristup analizi prosečnog slučaja? Kojim se matematičkim aparatom možemo služiti u analizi prosečnog slučaja kako bismo je učinili jednostavnijom? Odgovore na ova pitanja daćemo u sledećim odeljcima.

1.3 Analiza prosečnog slučaja – postavka zadatka

Analiza prosečnog slučaja je matematički gledano znatno složenija, pre svega zato što se proučavaju svi mogući ulazi date veličine, a ne samo ulazne istance za koje se dobijaju najgori rezultati. Zbog toga je potrebno analizirati učestalost određenih ulaznih istanci i na osnovu toga utvrđivati očekivano vreme izvršavanja algoritma.

Ulazne instance algoritma po pravilu možemo razmatrati kao kombinatorne objekte određenog tipa (permutacije, stringovi, stabla, grafovi itd.). Na primer, ulaz algoritma sortiranja je niz dužine n, koji se može razmatrati kao permutacija dužine n (pod pretpostavkom da su svi elementi različiti među sobom). Skup dopustivih instanci P ćemo, dakle, za potrebe analize algoritama razmatrati kao skup kombinatornih objekata.

Definicija 1.2. *Neka je dat skup kombinatornih objekata P i neka je $p \in P$. Tada uvodimo sledeće oznake:*

- $|p|$ *– veličina objekta p ($|p| \geq 0$);*
- $c(p)$ *– cena objekta p ($c(p) \geq 0$);*
- $p_{n,k}$ *– broj objekata $p \in P$, takvih da je $|p| = n$ i $c(p) = k$;*
- $p_n = \sum_{k \geq 0} p_{n,k}$ *– broj objekata $p \in P$, veličine n;*
- $q_k = \sum_{n \geq 0} p_{n,k}$ *– broj objekata $p \in P$, cene k;*
- $\sigma_n = \sum_{k \geq 0} k p_{n,k}$ *– kumulativna cena svih objekata veličine n;*

- $c_n = \frac{\sigma_n}{p_n} = \sum_{k \geq 0} k \frac{p_{n,k}}{p_n}$ – *prosečna cena objekata date veličine* n.

Intuitivno, u kontekstu analize algoritama, pod cenom objekta podrazumevamo potrebno vreme da se algoritam izvrši, ako se na ulazu nalazi upravo taj objekat, tj. ta ulazna instanca. Otuda je jasno da je prosečna cena c_n, tačnije niz $(c_n)_{n \geq 0}$ rezultat do koga želimo da dođemo prilikom analize prosečnog slučaja. Takođe je jasno da se određivanje ove vrednosti svodi na nalaženje veličina $p_{n,k}$ (ili σ_n i p_n), odakle sledi da se analiza prosečnog slučaja svodi na prebrojavanje elemenata skupova kombinatornih objekata. Definicija 1.2 nigde i ne spominje algoritam kao pojam; ovom definicijom se uvode osnovni pojmovi koji su nam potrebni u teoriji prebrojavanja kombinatornih objekata. Analiza prosečnog slučaja je zapravo samo jedna primena ove teorije u kojoj se elementi skupa P tumače kao ulazne instance nekog algoritma.

1.3.1 Analiza prosečnog slučaja sa stanovišta teorije verovatnoće

Iz definicije 1.2 jasno je da se prosečna cena c_n može tumačiti kao matematičko očekivanje slučajne veličine c, pod pretpostavkom da je $|p| = n$. Ulazne instance se u tom smislu razmatraju kao ishodi slučajnog eksperimenta. Pri tom je važno da *sve ulazne instance budu jednako verovatne*. Ako ovo nije slučaj, mora se razmotriti drugačiji verovatnosni model ulaza.

Objasnimo ovo podrobnije na primeru algoritma sortiranja niza. Ulaz ovog algoritma je niz koji se sortira. Pretpostavimo da su elementi niza različiti među sobom[4], kao i da su sve permutacije elemenata jednako verovatne. U ovom slučaju se ulazne instance veličine n mogu razmatrati kao permutacije dužine n, a skup P kao skup svih permutacija konačnih dužina. Međutim, može se pretpostaviti i drugačiji model: svaki element ulaznog niza je element nekog konačnog skupa elemenata E, pri čemu se svako $e \in E$ na svakoj od pozicija u nizu pojavljuje sa verovatnoćom $1/|E|$, gde je $|E|$ broj elemenata skupa E. U ovom slučaju se ulaz algoritma sortiranja može smatrati stringom nad azbukom E, pa je u tom slučaju $P = E^*$ – skup svih reči nad azbukom E. Naravno, uslov jednake verovatnosti svakog od ulaza date veličine n i u ovom slučaju važi – ako znamo da je string dužine n, tada se svaki od takvih stringova javlja sa verovatnoćom $1/|E|^n$.

[4] bez ograničenja opštosti možemo pretpostaviti da su elementi niza upravo brojevi $1, 2, 3, \ldots, n$.

1.3.2 Matematički aparat u analizi prosečnog slučaja i teoriji prebrojavanja

Bez obzira na to da li problem analize prosečnog slučaja svodimo na problem prebrojavanja ili koristimo neki drugi pristup, svakako dolazimo do veličine c_n, odnosno niza $(c_n)_{n\geq 0}$ koji treba odrediti. Uslovi koje niz mora da zadovoljava, tj. uslovi koji ga određuju, a koji su rezultat analize mogu se predstaviti na više načina. Jedan način je da niz predstavimo rekurentnom relacijom. Ovaj postupak nije uvek tako jednostavan. Rešiti dobijenu rekurentnu relaciju je obično još teži posao. O rekurentnim relacijama detaljnije govorimo u poglavlju 2. Drugi način je da se niz brojeva od interesa predstavi funkcijom generatrisom, a da se uslovi dobijeni analizom izraze u vidu funkcionalne jednačine po nepoznatoj funkciji generatrisi. Ova dva pristupa su suštinski ekvivalentna, ali se ispostavlja da je drugi način znatno sažetiji i da se njime dobar deo tehničkog posla izvođenja i transformacija rekurentnih relacija preskače. Postoji i kombinovani pristup – najpre postavimo rekurentnu relaciju, a zatim je odgovarajućim postupkom svedemo na funkcionalnu jednačinu po funkciji generatrisi. U poglavlju 3 govorimo detaljnije o funkcijama generatrisama.

2 Rekurentne relacije

Većina algoritama se mogu predstaviti kao iterativne ili rekurzivne procedure, pa se samim tim mogu posmatrati kao kompozicije procedura istog tipa koje rešavaju probleme manjih dimenzija. Zato se i potrebno vreme za izvršavanje algoritma za ulaz veće dimenzije najčešće može izraziti preko potrebnog vremena za ulaze manjih dimenzija. Ovakav pristup nas direktno vodi ka rekurentnim relacijama[5]. Rekurentne relacije se mogu primenjivati kako u analizi prosečnog, tako i u analizi najgoreg slučaja, kao i u mnogim drugim problemima, nevezanim za algoritme (kombinatorni problemi, teorija prebrojavanja). Zbog toga je od suštinskog značaja poznavanje rekurentnih relacija koje se često javljaju, kao i metoda za njihovo rešavanje.

Definicija 2.1. *Neka je dat niz* $a_n, n = 0, 1, 2, \ldots,$ *i neka za njegove elemente važi:*

$$a_n = f(a_{n-1}, a_{n-2}, \ldots, a_0, n)$$

[5]Rekurentne relacije se često nazivaju i *diferencne jednačine*, jer se mogu predstaviti u obliku tzv. konačnih razlika $\nabla f_n \equiv f_n - f_{n-1}$. One se mogu smatrati diskretnom verzijom diferencijalnih jednačina. Neke metode za rešavanje diferencijalnih jednačina se mogu primenjivati i na rekurentne relacije.

Tada kažemo da niz a_n zadovoljava rekurentnu relaciju datu gornjom jednačinom. Specijalno, ako se a_n izražava kao funkcija samo k prethodnih elemenata niza, tj. kao:

$$a_n = f(a_{n-1}, a_{n-2}, \ldots, a_{n-k}, n)$$

tada za rekurentnu relaciju kažemo da je reda k ($k \geq 1$).

Jasno je, naravno, da više različitih nizova mogu zadovoljavati istu rekurentnu relaciju. Međutim, lako se može pokazati da ako dva niza a_n i b_n zadovoljavaju istu rekurentnu relaciju reda k, i ako važi $a_i = b_i, 0 \leq i < k$, tada važi $a_n = b_n$ za svako $n \geq 0$. Drugim rečima, ako nam je poznato prvih k elemenata niza, tada je i ostatak niza jedinstveno određen. U tom slučaju, elemente $a_0, a_1, \ldots, a_{k-1}$ nazivamo *početnim uslovima* rekurentne relacije reda k.

Rešavanje rekurentne relacije može biti komplikovan zadatak. Za pretpostaviti je da se jednačine nižeg reda rešavaju jednostavnije, mada ni to ne mora biti slučaj. Težina rešavanja rekurentne relacije ipak najviše zavisi od same funkcije f kojom je definisana zavisnost od prethodnih članova niza u gornjoj definiciji. U nastavku ćemo upoznati neke često korišćene tipove rekurentnih relacija, kao i neke tipične metode za njihovo rešavanje.

2.1 Rekurentne relacije prvog reda

Kod rekurentnih relacija prvog reda, član niza a_n se definiše kao funkcija prethodnog člana niza a_{n-1}, tj:

$$a_n = f(a_{n-1}, n), \quad n > 0$$

Da bi rešenje ove relacije bilo jedinstveno, dovoljno je da znamo samo početni element niza, a_0. Ovakve relacije, bez obzira na jednostavnost njihove definicije, ponekad nije lako rešiti. Mi ćemo se ovde ipak zadržati na jednom specijalnom tipu rekurentnih relacija prvog reda za koje postoji jednoznačan postupak rešavanja. U pitanju su linearne rekurentne relacije prvog reda, odnosno relacije oblika:

$$a_n = x_n a_{n-1} + y_n, \quad \text{za} \quad n > 0, \quad \text{pri čemu je} \quad a_0 = 0 \tag{1}$$

Teorema 2.1 (Linearna rekurentna relacija prvog reda)**.** *Rekurentna relacija* (1) *ima rešenje dato izrazom:*

$$a_n = y_n + \sum_{1 \leq j < n} y_j x_{j+1} x_{j+2} \ldots x_n$$

Specijalni slučaj ovakve relacije dobija se za $x_n = 1$, za svako $n > 0$:

$$a_n - a_{n-1} = y_n, \quad \text{za} \quad n > 0, \quad \text{pri čemu je} \quad a_0 = 0,$$

a rešenje ovakve jednačine je na osnovu teoreme 2.1:

$$a_n = \sum_{1 \le j \le n} y_j$$

Dakle, rešavanje ovakvih jednačina svodi se na izračunavanje konačnih suma.

2.2 Rekurentne relacije višeg reda

U slučaju rekurentnih relacija višeg reda situacija se komplikuje zbog zavisnosti a_n od većeg broja prethodnih elemenata niza. Čak ni u slučaju linearnih rekurentnih relacija višeg reda, tj relacija oblika:

$$a_n = x_{n1}a_{n-1} + x_{n2}a_{n-2} + \ldots x_{nk}a_{n-k} + y_n, \quad \text{za} \quad n \ge k \tag{2}$$

ne postoji jednoznačan postupak kojim se određuje tačno rešenje proizvoljne jednačine ovog oblika. Ovakve jednačine se često mogu rešavati uz pomoć funkcija generatrisa ili nekom aproksimativnom metodom, ali postupak zavisi od izraza x_{nj} koji predstavljaju koeficiente linearne kombinacije na desnoj strani relacije i koji zavise od n. Problem postaje znatno jednostavniji ako pretpostavimo da su koeficienti x_{nj} konstantni, tj. da ne zavise od n, kao i da ne postoji koeficient y_n. U ovom slučaju, relacija (2) se svodi na:

$$a_n = x_1a_{n-1} + x_2a_{n-2} + \ldots x_ka_{n-k}, \quad \text{za} \quad n \ge k \tag{3}$$

i naziva se *homogena linearna rekurentna relacija reda k sa konstantnim koeficientima.* Rešenje ovakve rekurentne relacije dato je sledećom teoremom.

Teorema 2.2 (Linearna rekurentna relacija sa konstantnim koeficientima)**.** *Sva rešenja rekurentne relacije* (3) *mogu se predstaviti kao linearne kombinacije izraza oblika $n^j\beta^n$ gde je β koren karakterističnog polinoma:*

$$q(z) \equiv z^k - x_1z^{k-1} - x_2z^{k-2} - \ldots - x_k$$

i gde za j važi $0 \le j < s$, pri čemu je s višestrukost korena β. Tačno rešenje za zadate početne uslove $a_0, a_1, \ldots, a_{k-1}$ izračunava se rešavanjem sistema linearnih jednačina koji se dobija zamenom početnih uslova u dobijenoj linearnoj kombinaciji.

Početni uslovi jedinstveno određuju rešenje rekurentne relacije. Pritom, ovo rešenje može značajno da promeni oblik u zavisnosti od početnih uslova.

Primer 2.1. Neka je data jednačina:

$$a_n = 2a_{n-1} - a_{n-2}, \quad n \geq 2$$

Opšte rešenje ove jednačine na osnovu teoreme 2.2 dato je izrazom:

$$a_n = c_0 1^n + c_1 n 1^n$$

gde su c_0 i c_1 nepoznati koeficienti. Ako pretpostavimo da važe početni uslovi $a_0 = 1$ i $a_1 = 2$ dobijamo sledeći sistem lineranih jednačina:

$$\begin{array}{rcl} a_0 & = & 1 \quad = \quad c_0 \\ a_1 & = & 2 \quad = \quad c_0 + c_1 \end{array}$$

što daje rešenje $c_0 = c_1 = 1$ i samim tim važi $a_n = n + 1$. S druge strane, za početne uslove $a_0 = a_1 = 1$ rešenje bi bilo $a_n = 1$, za sve $n \geq 0$, što je konstanta, za razliku od linearnog rasta u prethodnom slučaju.

2.3 Relacije zasnovane na dekompoziciji

Algoritmi zasnovani na dekompoziciji (takozvana tehnika „zavadi pa vladaj") su veoma česti, a rekurentne relacije koje se prilikom njihove analize dobijaju prilično su slične, pa se primenjuju i slične tehnike za njihovo rešavanje. U ovom odeljku bavimo se pre svega *binarnom dekompozicijom*, tj. slučajem kada se problem deli na *dva* dela koji se zatim zasebno rešavaju, nakon čega se dobijena rešenja objedinjuju u jedinstveno rešenje celog problema.

Najčešća varijanta binarne dekompozicije je podela problema na dva potproblema *jednake* veličine. Poteškoća prilikom ovakvih dekompozicija sastoji se u tome što nije moguće za svako n podeliti problem na dva potpuno jednaka dela. Doslednu podelu na delove identične veličine moguće je u potpunosti ostvariti jedino ako je polazna veličina problema n oblika 2^k. Za drugačije vrednosti veličine ulaza mora se na izvesnom nivou odustati od idealne podele. Ovaj problem se reflektuje i na rekurentne relacije koje se zato dodatno komplikuju. Za slučaj $n = 2^k$ relacija se najčešće drastično pojednostavljuje i lako se dolazi do tačnog rešenja. U ostalim slučajevima moguće je utvrditi asimptotsko ponašanje koje se najčešće poklapa sa dobijenim rešenjem za $n = 2^k$. Međutim, nalaženje tačnog rešenja u ovim situacijama je znatno teže. Demonstrirajmo ovo na jednom primeru.

Primer 2.2. Rekurentna relacija kojom se definiše broj potrebnih upoređivanja prilikom sortiranja objedinjavanjem je

$$C_n = C_{\lfloor n/2 \rfloor} + C_{\lceil n/2 \rceil} + n, \quad \text{za} \quad n \geq 2 \tag{4}$$

pri čemu je $C_1 = 0$. Za $n = 2^k$, smenom promenljive $T_k = \frac{1}{2^k} C_{2^k}$, dobijamo relaciju:

$$T_k = T_{k-1} + 1, \quad \text{za} \quad k > 0$$

pri čemu je $T_0 = C_1 = 0$. Rešenje ove relacije je $T_k = k$. Vraćanjem na promenljivu n, dobijamo rešenje polazne relecije: $C_{2^k} = 2^k T_k = 2^k k$, odnosno $C_n = n \lg n$. Ovo je tačno rešenje, ali važi samo za $n = 2^k$. Međutim, da li možemo nešto reći o vrednosti C_n za proizvoljno n? Indukcijom se može pokazati da je niz C_n rastući. Otuda je $C_{2^{\lfloor \lg n \rfloor}} \leq C_n < C_{2^{\lceil \lg n \rceil}}$, odnosno $2^{\lfloor \lg n \rfloor} \lfloor \lg n \rfloor \leq C_n < 2^{\lceil \lg n \rceil} \lceil \lg n \rceil$, na osnovu čega možemo zaključiti da je $C_n = O(n \log n)$. Preciznijom analizom bismo mogli da dobijemo još bolji rezultat $C_n = n \lg n + O(n)$ (kako je i navedeno u teoremi 1.1, strana 4).

Za određivanje tačnog rešenja relacije (4) potrebno je primeniti složenije tehnike, zasnovane na uspostavljanju veze sa binarnim brojevima, koje ovde nećemo izlagati. Ispostavlja se da je tačno rešenje dato izrazom:

$$C_n = n \lfloor \lg n \rfloor + 2n - 2^{\lfloor \lg n \rfloor + 1}$$

Prilikom nalaženja tačnog rešenja, svi detalji koje jednačina sadrži moraju biti uzeti u obzir. Mala promena relacije će možda dati jednostavniji izraz, ali se dobijeno rešenje može znatno razlikovati od tačnog rešenja polazne jednačine. Razlog je to što se prilikom dekompozicije problem svodi na veliki broj podproblema malih dimenzija i svaka nepreciznost na tom nivou se mnogostruko manifestuje za velike vrednosti n. Na primer, da smo u relaciji (4), zamenili $C_{\lfloor n/2 \rfloor}$ sa $C_{\lceil n/2 \rceil}$ (što je naizgled mala promena), dobili bismo relaciju oblika $C_n = 2C_{\lceil n/2 \rceil} + n$, koja je jednostavnijeg oblika, ali čije tačno rešenje drastično odstupa od rešenja polazne relacije, iako se asimptotski ponaša isto.

Na kraju, navedimo još jednu teoremu, kojom se opisuje asimptotsko ponašanje rešenja rekurentne relacije koja je zasnovana na dekompoziciji, ovoga puta ne obavezno binarnoj. Teorema je data znatno opštije i može se odnositi na proizvoljnu funkciju $a(x) : \mathbf{R}^+ \to \mathbf{R}$ koja je definisana rekurentno.

Teorema 2.3 (Funkcije zasnovane na dekompoziciji)**.** *Ako funkcija $a(x)$ zadovoljava relaciju:*

$$a(x) = \alpha a(x/\beta) + x, \quad za \quad x > 1, \quad i \quad a(x) = 0 \quad za \quad x \leq 1$$

gde su $\alpha > 1$ *i* $\beta > 1$ *proizvoljni relani brojevi, tada važi* [6]*:*

$$a(x) \sim \begin{cases} \frac{\beta}{\beta-\alpha}x, & \alpha < \beta \\ x \log_\beta x, & \alpha = \beta \\ \frac{\alpha}{\alpha-\beta}\left(\frac{\beta}{\alpha}\right)^{\{\log_\beta \alpha\}} x^{\log_\beta \alpha}, & \alpha > \beta \end{cases}$$

Mnoge rekurentne relacije ipak nije tako jednostavno rešiti. Takođe, ponekad je teško i uočiti rekurentnu vezu koja opisuje veličinu od interesa koja se odnosi na dati algoritam. U nastavku upoznajemo pojam funkcija generatrisa kojima se oba ova problema mogu znatno efikasnije rešavati.

3 Funkcije generatrise

Funkcije generatrise su veoma koristan alat u rešavanju različitih problema u diskretnoj matematici, kombinatorici, pa samim tim i u analizi algoritama. Prva primena funkcija generatrisa koju ilustrujemo u ovom poglavlju je primena u rešavanju rekurentnih relacija, svođenjem istih na njima ekvivalentne funkcionalne jednačine. Druga primena koju takođe predstavljamo u ovom poglavlju je primena u teoriji prebrojavanja, a samim tim, indirektno, i u analizi algoritama. Pojam funkcije generatrise dat je sledećom definicijom.

Definicija 3.1 (Funkcija generatrisa)**.** *Ako je dat niz brojeva* $a_0, a_1, \ldots, a_n, \ldots$*, tada funkciju:*

$$A(z) = \sum_{n \geq 0} a_n z^n$$

nazivamo funkcijom generatrisom datog niza. Koristimo notaciju $[z^n]A(z)$ *za koeficient* a_n*, tj. za koeficient uz* z^n*.*

Primetimo da su funkcije generatrise u suštini stepeni redovi. Međutim, ovde ćemo ih pre svega razmatrati kao formalne sume koje koristimo za predstavljanje brojevnih nizova na jednostavan i kompaktan način, kao i za elegantnu i jednostavnu manipulaciju istim nizovima. Zbog toga nam neće biti bitno da li red konvergira, kao i za koje vrednosti z konvergira. Većina operacija nad funkcijama generatrisama (koje odgovaraju operacijama nad nizovima brojeva) su formalne operacije za koje nije potrebna konvergencija. I pored toga, većina ovako definisanih stepenih redova konvergira, makar za

[6] Izraz $\{\log_\beta \alpha\}$ koji je prisutan u tvrđenju teoreme predstavlja razlomljeni deo broja, $\{\log_\beta \alpha\} = \log_\beta \alpha - \lfloor \log_\beta \alpha \rfloor$

male vrednosti z, što nam omogućava da sumiramo dobijene stepene redove, i odgovarajuću funkciju generatrisu predstavimo analitičkim izrazom. Takođe, rešavanjem funkcionalnih jednačina po nepoznatoj funkciji generatrisi, rešenje se najčešće dobija u obliku analitičkog izraza, a za razvoj tog izraza u red neophodno je da funkcija bude beskonačno diferencijabilna u nekoj okolini nule.

Primer 3.1. Navedimo neke primere nizova i njima pridruženih funkcija generatrisa.

$1, 1, 1, \ldots, 1, \ldots$	$\frac{1}{1-z}$	$=$	$\sum_{n\geq 0} z^n$
$0, 1, 2, 3, \ldots, n, \ldots$	$\frac{z}{(1-z)^2}$	$=$	$\sum_{n\geq 1} nz^n$
$0, \ldots, 0, 1, m+1, \ldots, \binom{n}{m}, \ldots$	$\frac{z^m}{(1-z)^{m+1}}$	$=$	$\sum_{n\geq m} \binom{n}{m} z^n$
$1, c, c^2, c^3, \ldots, c^n, \ldots$	$\frac{1}{1-cz}$	$=$	$\sum_{n\geq 0} c^n z^n$
$1, 1, \frac{1}{2!}, \frac{1}{3!}, \frac{1}{4!}, \ldots, \frac{1}{n!}, \ldots$	e^z	$=$	$\sum_{n\geq 0} \frac{z^n}{n!}$
$0, 1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots, \frac{1}{n}, \ldots$	$\ln \frac{1}{1-z}$	$=$	$\sum_{n\geq 1} \frac{z^n}{n}$
$0, 1, 1+\frac{1}{2}, 1+\frac{1}{2}+\frac{1}{3}, \ldots, H_n, \ldots$	$\frac{1}{1-z} \ln \frac{1}{1-z}$	$=$	$\sum_{n\geq 1} H_n z^n$
$0, 0, 1, 3(\frac{1}{2}+\frac{1}{3}), 4(\frac{1}{2}+\frac{1}{3}+\frac{1}{4}), \ldots$	$\frac{z}{(1-z)^2} \ln \frac{1}{1-z}$	$=$	$\sum_{n\geq 0} n(H_n - 1) z^n$

Teorema 3.1 (Operacije sa funkcijama generatrisama)**.** *Neka su data dva niza brojeva $a_0, a_1, \ldots, a_n, \ldots$, i $b_0, b_1, \ldots, b_n, \ldots$, kao i njima pridružene generatrise $A(z) = \sum_{n\geq 0} a_n z^n$ i $B(z) = \sum_{n\geq 0} b_n z^n$. Tada sledećim operacijama nad ovim brojevnim nizovima odgovaraju date operacije nad njima pridruženim funkcijama generatrisama:*

- *Desnom pomeranju: $0, a_0, a_1, \ldots, a_{n-1}, \ldots$ odgovara operacija $zA(z)$.*
- *Levom pomeranju: $a_1, a_2, a_3, \ldots, a_{n+1}, \ldots$ odgovara operacija $\frac{A(z)-a_0}{z}$.*
- *Množenju indeksom: $a_1, 2a_2, 3a_3, \ldots, na_n, \ldots$ odgovara operacija $A'(z)$.*
- *Deljenju indeksom: $0, a_0, \frac{a_1}{2}, \frac{a_2}{3}, \ldots, \frac{a_{n-1}}{n}, \ldots$ odgovara operacija $\int_0^z A(t)dt$.*
- *Skaliranju: $a_0, \lambda a_1, \lambda^2 a_2, \ldots, \lambda^n a_n, \ldots$ odgovara operacija $A(\lambda z)$.*

- *Sabiranju:* $a_0 + b_0, a_1 + b_1, \ldots, a_n + b_n, \ldots$ *odgovara operacija* $A(z) + B(z)$.
- *Diferencnoj operaciji:* $a_0, a_1 - a_0, a_2 - a_1, \ldots, a_n - a_{n-1}, \ldots$ *odgovara operacija* $(1 - z)A(z)$.
- *Konvoluciji:* $a_0b_0, a_0b_1 + a_1b_0, a_0b_2 + a_1b_1 + a_2b_0, \ldots, \sum_{0 \leq k \leq n} a_k b_{n-k}, \ldots$ *odgovara operacija* $A(z)B(z)$.
- *Parcijalnoj sumi:* $a_0, a_0 + a_1, \ldots, \sum_{0 \leq k \leq n} a_k, \ldots$ *odgovara operacija* $\frac{A(z)}{1-z}$.

Koristeći ovu teoremu, možemo polazeći od nekih poznatih nizova, jednostavnim manipulacijama dobiti funkcije generatrise složenijih nizova. Obrnuto je takođe moguće, tj. ako funkciju generatrisu nekog nepoznatog niza možemo predstaviti kompozicijom jednostavnijih funkcija (pri čemu pod kompozicijom mislimo na gore navedene operacije), tada se kombinovanjem nizova koji su predstavljeni tim funkcijama generatrisama na odgovarajući način može dobiti niz koji odgovara datoj funkciji generatrisi. Ilustrujmo ovo primerima.

Primer 3.2. Neka je dat niz brojeva $0, H_1, H_2, \ldots, H_n, \ldots$, pri čemu je $H_n = \sum_{1 \leq k \leq n} \frac{1}{k}$ (tzv. *harmonijski brojevi*). Do funkcije generatrise ovog niza možemo doći na sledeći način: polazeći od niza $1, 1, \ldots, 1, \ldots$ (čija je generatrisa data izrazom $\frac{1}{1-z}$ na osnovu prethodnog primera), možemo najpre operacijom *deljenja indeksom* dobiti niz $0, 1, \frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{n}, \ldots$. Ovo odgovara operaciji integracije nad funkcijom generatrisom, što za rezultat daje funkciju $\ln \frac{1}{1-z}$. Nakon toga je još potrebno izračunati parcijalne sume, što odgovara operaciji množenja izrazom $\frac{1}{1-z}$. Rezultat je, dakle, funkcija $A(z) = \frac{1}{1-z} \ln \frac{1}{1-z}$.

Primer 3.3. Neka je data funkcija generatrisa $A(z) = \frac{e^{2z}}{1-2z}$. Do niza koji je definisan ovom funkcijom možemo doći na sledeći način. Funkcija e^z generiše niz $\sum_{n \geq 0} \frac{z^n}{n!}$. Zato funkcija $\frac{e^z}{1-z}$ generiše niz $\sum_{n \geq 0} \left(\sum_{0 \leq k \leq n} \frac{1}{k!} \right) z^n$, s obzirom na pravilo parcijalnih suma iz teoreme 3.1. Na kraju, na osnovu pravila skaliranja dolazimo do rezultata: $\frac{e^{2z}}{1-2z} = \sum_{n \geq 0} 2^n \left(\sum_{0 \leq k \leq n} \frac{1}{k!} \right) z^n$.

3.1 Rešavanje rekurentnih relacija uz pomoć funkcija generatrisa

Kao što je ranije rečeno, funkcije generatrise se mogu koristiti kao alat za lakše rešavanje rekurentnih relacija. Neka je dat nepoznati niz brojeva

$(a_n)_{n\geq 0}$ i neka je data rekurentna relacija kojom je ovaj niz definisan. Tada se može primeniti sledeći opšti postupak za svođenje date rekurentne relacije na funkcionalnu jednačinu po nepoznatoj funkciji generatrisi $A(z) = \sum_{n\geq 0} a_n z^n$:

- Pomnožimo celu relaciju sa z^n.
- Sumirajmo dobijenu jednakost po n.
- Izrazimo dobijene sume preko funkcije generatrise $A(z) = \sum_{n\geq 0} a_n z^n$.
- Rešimo dobijenu funkcionalnu jednačinu po $A(z)$.
- Razvijmo u red dobijenu funkciju generatrisu. Dobijeni koeficient $a_n = [z^n]A(z)$ je upravo opšti član traženog niza.

Funkcionalna jednačina koja se na ovaj način dobija može biti jednostavna, mada je često i znatno komplikovanija. Često je u pitanju diferencijalna jednačina. Međutim, koliko god da je komplikovana jednačina u pitanju, asortiman matematičkih alata i tehnika za njeno rešavanje je znatno raznovrsniji i bogatiji od skupa tehnika koje imamo na raspolaganju za direktno rešavanje rekurentnih relacija.

Primer 3.4. Neka je data rekurentna relacija:

$$a_n = 2a_{n-1} + 1$$

uz $a_0 = 1$. Pomnožimo relaciju sa z^n, a zatim sumirajmo za $n \geq 1$. Dobijamo:

$$\sum_{n\geq 1} a_n z^n = 2\sum_{n\geq 1} a_{n-1} z^n + \sum_{n\geq 1} z^n$$

odnosno:

$$A(z) = 2zA(z) + \frac{1}{1-z}$$

odakle je:

$$A(z) = \frac{1}{(1-2z)(1-z)}$$

Sada možemo razviti dobijenu funkciju u red:

$$A(z) = \frac{1}{(1-2z)(1-z)} = \frac{2}{1-2z} - \frac{1}{1-z} = 2\sum_{n\geq 0} 2^n z^n - \sum_{n\geq 0} z^n$$

odakle sledi da je $a_n = 2^{n+1} - 1$.

Sličnim pristupom kao u prethodnom primeru, možemo dokazati sledeću teoremu.

Teorema 3.2 (Linearne rekurentne relacije i funkcije generatrise)**.** *Ako a_n zadovoljava rekurentnu relaciju:*

$$a_n = x_1 a_{n-1} + x_2 a_{n-2} + \ldots x_k a_{n-k}, \quad za \quad n \geq k$$

tada je odgovarajuća funkcija generatrisa $A(z) = \sum_{n\geq 0} a_n z^n$ racionalna funkcija $A(z) = f(z)/g(z)$, gde je $g(z) = 1 - x_1 z - x_2 z^2 - \ldots - x_k z^k$, a polinom $f(z)$ je određen početnim uslovima $a_0, a_1, \ldots, a_{k-1}$ i može se izračunati iz jednakosti:

$$f(z) = g(z) \sum_{0\leq n<k} a_n z^n \quad (mod \quad z^k)$$

pri čemu je njegov stepen strogo manji od k,

Gornja teorema se, dakle, odnosi na linearne homogene relacije sa konstantnim koeficientima, za koje smo ranije već definisali jednoznačan postupak rešavanja korišćenjem karakterističnog polinoma:

$$q(z) \equiv z^k - x_1 z^{k-1} - x_2 z^{k-2} - \ldots - x_k$$

Možemo primetiti da važi: $g(z) = z^k q(1/z)$. Otuda, ako su $\beta_1, \beta_2, \ldots, \beta_n$ koreni polinoma $q(z)$, tada su $1/\beta_1, 1/\beta_2, \ldots, 1/\beta_n$ koreni polinoma $g(z)$ i on se može faktorisati kao:

$$g(z) = (1 - \beta_1 z) \cdot (1 - \beta_2 z) \cdot \ldots \cdot (1 - \beta_n z)$$

Ova činjenica ukazuje na suštinsku sličnost ova dva metoda za rešavanje istog tipa rekurentnih relacija. Štaviše, gornja faktorizacija pomaže u praksi prilikom razlaganja dobijene funkcije generatrise na više jednostavnijih funkcija, takozvanih *parcijalnih razlomaka*, za koje se jednostavno određuju njima pridruženi nizovi brojeva. Demonstrirajmo ovo na jednom primeru.

Primer 3.5. Neka je data relacija:

$$a_n = 2a_{n-1} + a_{n-2} - 2a_{n-3}$$

za $n > 2$, uz $a_0 = 0$ i $a_1 = a_2 = 1$. Najpre, na osnovu teoreme 3.2 računamo $g(z)$:

$$g(z) = 1 - 2z - z^2 + 2z^3 = (1 - z)(1 + z)(1 - 2z)$$

a zatim $f(z)$ računamo na osnovu početnih uslova:

$$f(z) = (z + z^2)(1 - 2z - z^2 + 2z^3) \quad (mod \quad z^3)$$

odakle sledi da je:

$$f(z) = z(1-z)$$

Sada je:

$$A(z) = \frac{f(z)}{g(z)} = \frac{z}{(1+z)(1-2z)} = \frac{1}{3}(\frac{1}{1-2z} - \frac{1}{1+z})$$

pa je otuda $a_n = \frac{1}{3}(2^n - (-1)^n)$.

Na kraju ovog poglavlja, navodimo još jedan primer rekurentne relacije, čijom se transformacijom dobija diferencijalna jednačina.

Primer 3.6. Neka je data rekurentna relacija[7]:

$$C_n = n + 1 + \frac{1}{n} \sum_{1 \le j \le n} (C_{j-1} + C_{n-j}), \quad \text{za} \quad n > 0$$

uz $C_0 = 0$. Elementarnim transformacijama dobijamo:

$$nC_n = n(n+1) + 2 \sum_{1 \le k \le n} C_{k-1}$$

Množenjem sa z^n, a zatim i sumiranjem po n, dobijamo:

$$\sum_{n \ge 1} nC_n z^n = \sum_{n \ge 1} n(n+1)z^n + 2 \sum_{n \ge 1} \left(\sum_{1 \le k \le n} C_{k-1} \right) z^n$$

što se dalje svodi na sledeću diferencijalnu jednačinu:

$$C'(z) = \frac{2}{(1-z)^3} + 2\frac{C(z)}{1-z}$$

Rešavanjem ove diferencijalne jednačine dobijamo:

$$C(z) = \frac{2}{(1-z)^2} \ln \frac{1}{1-z}$$

odakle se dobija:

$$C_n = [z^n]C(z) = 2(n+1)(H_{n+1} - 1)$$

[7] Setimo se da je ovo upravo rekurentna relacija koja definiše očekivani broj upoređivanja kod algoritma *QuickSort*.

3.2 Prebrojavanje pomoću funkcija generatrisa

U ovom poglavlju demonstriramo jedan zanimljiv pristup prebrojavanju skupova kombinatornih objekata.

Definicija 3.2. *Neka je dat skup kombinatornih objekata P i neka je sa p_n obeležen broj objekata $p \in P$ takvih da je $|p| = n$. Tada za funkciju generatrisu $P(z) = \sum_{n\geq 0} p_n z^n$ niza $(p_n)_{n\geq 0}$, kažemo da prebrojava skup P.*

Ideja ovog pristupa je da na neki način, direktno na osnovu svojstava samih kombinatornih objekata iz skupa P, odredimo funkcionalnu jednačinu po $P(z)$, bez prethodnog sastavljanja rekurentne relacije po nepoznatom nizu $(p_n)_{n\geq 0}$. Jedan način da ovo postignemo demonstriran je u sledećem primeru.

Primer 3.7. Neka je data azbuka Σ, i neka je $\sigma = |\Sigma|$. Neka je dalje skup svih reči nad azbukom Σ obeležen sa Σ^*. Tada je lako zaključiti da je broj reči dužine n nad azbukom Σ jednak σ^n. Ovo možemo pokazati i primenom funkcija generatrisa. Neka je p_n traženi broj i neka je $P(z) = \sum_{n\geq 0} p_n z^n$ funkcija generatrisa niza $(p_n)_{n\geq 0}$. Tada važi:

$$\begin{aligned} P(z) &= \sum_{n\geq 0} p_n z^n \\ &= \sum_{v\in\Sigma^*} z^{|v|} \\ &= 1 + \sum_{aw\in\Sigma^*} z^{|aw|} \\ &= 1 + \sum_{w\in\Sigma^*} \sum_{a\in\Sigma} z^{|w|+1} \\ &= 1 + \sigma z \sum_{w\in\Sigma^*} z^{|w|} \\ &= 1 + \sigma z P(z) \end{aligned}$$

Ključni korak u ovom izvođenju je prva jednakost, koja važi zato što se u drugoj sumi izraz $z^{|v|}$ pojavljuje onoliko puta koliko ima reči dužine $|v|$ u skupu Σ^*, pa je koeficient uz z^n upravo p_n. U daljem izvođenju smo koristili rekurzivnu definiciju reči nad azbukom Σ: reč je ili prazna reč ϵ ili slovo za kojim sledi reč. Praznoj reči odgovara sabirak 1, a ostatak sume se može predstaviti kao dvostruka suma (po prvom slovu i po ostatku reči). Odavde se lako dobija funkcionalna jednačina po $P(z)$. Rešavanjem ove funkcionalne jednačine, dobijamo:

$$P(z) = \frac{1}{1-\sigma z} = \sum_{n\geq 0} \sigma^n z^n$$

odakle sledi da je $p_n = \sigma^n$, kao što je i očekivano.

Sledeći primer demonstrira korišćenje funkcija generatrisa za prebrojavanje binarnih stabala.

Definicija 3.3 (Binarno stablo)**.** *Binarno stablo je kombinatorni objekat koji se sastoji iz unutrašnjih i spoljašnjih čvorova i definiše se rekurzivno na sledeći način:*

- *spoljašnji čvor e je binarno stablo.*
- *ako je dat unutrašnji čvor r, kao i binarna stabla t_l i t_r, tada je uređena trojka (t_l, r, t_r) takođe binarno stablo. Čvor r se tada naziva koren stabla, dok su stabla t_l i t_r redom levo i desno podstablo datog stabla.*

Dakle, binarno stablo je kombinatorni objekat sastavljen od čvorova. Pitanje koje se nameće je koliko različitih binarnih stabala postoje sa svojstvom da imaju tačno n unutrašnjih čvorova? Neka je tražena vrednost T_n. S obzirom da se može pokazati da je broj spoljašnjih čvorova uvek za jedan veći od broja unutrašnjih čvorova, tada se sa T_n takođe obeležava broj binarnih stabala sa tačno $n+1$ spoljašnjih čvorova.

Primer 3.8. Neka je $T(z)$ funkcija generatrisa koja je pridružena nizu brojeva $T_0, T_1, \ldots, T_n, \ldots$, pri čemu je sa T_n obeležen broj binarnih stabala sa n unutrašnjih čvorova. Obeležimo sa T skup svih binarnih stabala i sa $|t|$ broj unutrašnjih čvorova stabla $t \in T$. Tada važi:

$$\begin{aligned} T(z) &= \sum_{n\geq 0} T_n z^n \\ &= \sum_{t\in T} z^{|t|} \\ &= 1 + \sum_{(t_l,r,t_r)\in T} z^{|(t_l,r,t_r)|} \\ &= 1 + \sum_{t_l\in S}\sum_{t_r\in T} z^{|t_l|+|t_r|+1} \\ &= 1 + zT(z)^2 \end{aligned}$$

Prva jednakost važi zato što se u drugoj sumi svako $z^{|t|}$ pojavljuje onoliko puta koliko ima binarnih stabala sa $|t|$ čvorova, pa je koeficient uz z^n upravo T_n. Druga jednakost se može objasniti na sledeći način: najpre primetimo da je $T_0 = 1$, zato što postoji tačno jedno stablo sa nula unutrašnjih čvorova (stablo koje se sastoji samo iz jednog spoljašnjeg čvora). Izdvajanjem ovog sabirka, u sumi ostaju samo sabirci koji odgovaraju stablima sa bar jednim unutrašnjim čvorom. Po definiciji 3.3, ova stabla se sastoje iz jednog unutrašnjeg čvora r i dva podstabla t_l i t_r. Ako primetimo da važi i $|(t_l, r, t_r)| = |t_l| + |t_r| + 1$, tada je jednakost jasna. Na osnovu gornjeg izvođenja, dobijamo funkcionalnu jednačinu:

$$T(z) = 1 + zT(z)^2$$

odakle sledi da je:

$$T(z) = \frac{1 - \sqrt{1-4z}}{2z}$$

Najzad, razvijanjem dobijene funkcije generatrise dobijamo rezultat: [8]

$$T_n = \frac{1}{n+1}\binom{2n}{n}$$

Definicija 3.4 (Uopšteno stablo)**.** *Uopšteno stablo je uređeni par (v, s), gde je v čvor – koren stabla, a $s = (t_1, t_2, \ldots, t_k)$ niz disjunktnih stabala konačne dužine k, za proizvoljno $k \geq 0$.*[9] *Stabla iz s nazivamo podstablima datog stabla. Niz stabala s nazivamo i uređena šuma.*

Primer 3.9. Neka je T skup svih uopštenih stabala i neka je T_n broj svih takvih stabala sa n čvorova. Neka je, dalje, funkcija $T(z)$ generatrisa niza $(T_n)_{n\geq 0}$. Sada imamo sledeće izvođenje:

$$\begin{aligned}
T(z) &= \textstyle\sum_{n\geq 0} T_n z^n \\
&= \textstyle\sum_{t\in T} z^{|t|} \\
&= z + \textstyle\sum_{k\geq 1} \sum_{(v,(t_1,t_2,\ldots,t_k))\in T} z^{|(v,(t_1,t_2,\ldots,t_k)|} \\
&= z + \textstyle\sum_{k\geq 1} \sum_{t_1\in T} \sum_{t_2\in T} \cdots \sum_{t_k\in T} z^{|t_1|+|t_2|+\ldots+|t_k|+1} \\
&= z + \textstyle\sum_{k\geq 1} z \prod_{1\leq i\leq k} \sum_{t_i\in T} z^{|t_i|} \\
&= z + z\textstyle\sum_{k\geq 1} T(z)^k \\
&= z\textstyle\sum_{k\geq 0} T(z)^k \\
&= z\frac{1}{1-T(z)}
\end{aligned}$$

odakle se dobija funkcionalna jednačina:

$$T(z) = z \cdot \frac{1}{1 - T(z)}$$

Gornje izvođenje je analogno izvođenjima iz prethodnih primera, uz primedbu da smo sumu po svim stablima najpre razbili na sumu po k pri čemu je svaki sabirak ove sume suma po svim stablima sa tačno k podstabala. Sabirak z je izdvojen kao poseban slučaj stabla sa 0 podstabala. Odavde se dobija da je:

$$T(z) = \frac{1 - \sqrt{1 - 4z}}{2}$$

odakle se razvijanjem dobijene funkcije generatrise dobija da je broj svih stabala sa n čvorova jednak:

$$T_n = \frac{1}{n}\binom{2n-2}{n-1}$$

[8] Dobijeni brojevi nazivaju se *Katalanovi brojevi.*

[9] Sekvenca može biti dužine 0. U tom slučaju se stablo svodi na jedan čvor – koren v.

Primetimo da je broj svih uopštenih stabala sa n čvorova jednak broju svih binarnih stabala sa $n-1$ unutrašnjih čvorova. Ovo nije slučajno. Postoji jednoznačna korespodencija između binarnih stabala sa $n-1$ unutrašnjih čvorova i uopštenih stabala sa n čvorova. Opisivanje ove korespodencije prevazilazi okvire ovog rada. Za detalje čitalac može pogledati knjigu [1].

Prethodni primeri pokazuju kako možemo izbeći rekurentne relacije, ali ni ovakvo direktno izvođenje, videli smo, nije jednostavno. Na sreću, postoji jednostavniji i elegantniji način da se dođe do gore dobijenih funkcionalnih jednačina. O tome govorimo u sledećem poglavlju.

3.3 Simbolički metod

Svaki složeni kombinatorni objekat se po pravilu može predstaviti kao struktura sastavljena od manjih i jednostavnijih kombinatornih objekata koji su na određeni način povezani među sobom. Drugim rečima, složene kombinatorne objekte gradimo tako što na određeni način kombinujemo prostije objekte, primenjujući odgovarajuća pravila. O ovome govori sledeća definicija.

Definicija 3.5 (Konstrukcija kombinatornih objekata)**.** *Neka su dati skupovi kombinatornih objekata A i B. Tada se, polazeći od ovih skupova, mogu konstruisati sledeći skupovi:*

- *disjunktna unija skupova, u oznaci $A+B$, čiji su elementi svi objekti koji su elementi jednog od skupova A ili B i važi $A \cap B = \varnothing$. Pri tom je veličina svakog objekta skupa $A+B$ jednaka veličini koju je taj objekat imao u skupu iz koga potiče.*

- *Dekartov proizvod skupova, u oznaci $A \times B$, čiji su elementi uređeni parovi oblika (a,b), pri čemu je $a \in A$ i $b \in B$. Pri tom je $|(a,b)| = |a| + |b|$, tj. veličina svakog uređenog para jednaka je zbiru veličina objekata iz kojih je sastavljen.*

- *Dekartov stepen skupa, u oznaci $A^n, n \in \mathbf{N_0}$, čiji su elementi uređene n-torke oblika $(a_1, a_2, \ldots, a_n)$, pri čemu je $a_i \in A, i = 1, 2, \ldots, n$. Pri tom je $|(a_1, a_2, \ldots, a_n)| = |a_1| + |a_2| + \ldots + |a_n|$, tj. veličina svake n-torke jednaka je zbiru veličina objekata iz kojih je sastavljena. Specijalno, $A^0 = \{\epsilon\}$, gde je ϵ prazan niz i važi $|\epsilon| = 0$.*

Skupovi kombinatornih objekata od kojih polazimo obično sadrže najjednostavnije objekte. Od ovih skupova gore opisanim operacijama dobijamo skupove složenijih objekata.

Primetimo da smo u definiciji 1.2 (strana 7) veličinu objekta p (u oznaci $|p|$) definisali potpuno uopšteno, ne ulazeći u smisao ove funkcije. Sada vidimo da je veličina objekta jasno definisana na osnovu pravila za konstrukciju složenih objekata. Jedino što je potrebno je da definišemo veličinu objekata koji čine osnovni skup – veličina složenijih objekata je nakon toga jednoznačno određena.

Primer 3.10. Skup svih binarnih stabala se može konstruisati na sledeći način. Razmatrajmo osnovni skup $\{e, r\}$, pri čemu je e spoljašnji, a r unutrašnji čvor. Neka je $|e| = 0$ i $|r| = 1$. Sada se skup binarnih stabala može definisati rekurzivno[10] kao $T = \{e\} + T \times \{r\} \times T$ – svako stablo je ili prazno (jedan spoljašnji čvor) ili sadrži koren (unutrašnji čvor) i dva podstabla, levo i desno. S obzirom na način na koji smo definisali veličinu osnovnih objekata, jasno je da će veličina svakog stabla biti broj unutrašnjih čvorova. Ako želimo da definišemo veličinu stabla kao broj spoljašnjih čvorova, tada je dovoljno da definišemo $|e| = 1$ i $|r| = 0$.

Operacija Dekartovog stepena je u stvari višestruki Dekartov proizvod skupa samim sobom, tj. važi:

$$A^n = \underbrace{A \times A \times \ldots \times A}_{n}$$

Pravilo konstrukcije Dekartovim stepenom omogućava definisanje objekata koji predstavljaju nizove objekata nekog skupa određene konačne dužine.

Primer 3.11. Neka je dat osnovni skup $\{v\}$ (element v predstavlja čvor) i neka je $|v| = 1$. Sada se skup uopštenih stabala može definisati na sledeći način: $T = \{v\} \times (\sum_{n \geq 0} T^n)$ – stablo čine koren i niz njegovih podstabala.[11] Veličina stabla je u skladu sa ovom definicijom upravo broj čvorova u stablu.

Primer 3.12. Skup Σ^* svih reči nad azbukom Σ se može konstruisati na sledeći način. Neka je osnovni skup $\Sigma \cup \{\epsilon\}$, neka za $a \in \Sigma$ važi $|a| = 1$ i neka važi $|\epsilon| = 0$. Važi $\Sigma^* = \{\epsilon\} + \Sigma \times \Sigma^*$ – reč je ili prazna reč ϵ ili je u pitanju slovo za kojim sledi reč[12]. Na osnovu ove definicije proističe da je veličina objekta definisana kao broj slova u niski, što je prirodna definicija dužine.

Osnovno pitanje koje se ovde postavlja je kako prebrojati elemente složenog skupa kombinatornih objekata, ako znamo da prebrojimo skupove objekata iz kojih je ovaj skup konstruisan. Drugim rečima, kakva je veza

[10] S obzirom da su kombinatorni objekti obično rekurzivne strukture, uobičajeno je da se oni rekurzivno i definišu.

[11] Suma se ovde tumači kao disjunktna unija: $\sum_{n \geq 0} T^n = T^0 + T^1 + T^2 + \ldots + T^n + \ldots$.

[12] Drugi način je da primetimo da je $\Sigma^* = \sum_{n \geq 0} \Sigma^n$ – skup svih nizova konačne dužine objekata iz Σ.

funkcija generatrisa ovih skupova? Odgovor na ovo pitanje daje nam sledeća teorema.

Teorema 3.3. *Neka su data dva skupa kombinatornih objekata A i B i neka su funkcije $A(z)$ i $B(z)$ respektivno njima pridružene funkcije generatrise koje ih prebrojavaju. Tada važi sledeće:*

- *Disjunktnoj uniji $A + B$ datih skupova odgovara funkcija generatrisa $A(z) + B(z)$*
- *Dekartovom proizvodu $A \times B$ datih skupova odgovara funkcija generatrisa $A(z)B(z)$. Specijalno, Dekartovom stepenu A^n odgovara funkcija generatrisa $A(z)^n$.*
- *Skupu $\sum_{n\geq 0} A^n$ svih nizova objekata iz A konačnih dužina, odgovara funkcija generatrisa $\frac{1}{1-A(z)}$.*

Kao što vidimo iz ove teoreme, postoji veoma jednostavna i prirodna korespodencija između pravila za konstrukciju skupova kombinatornih objekata i pravila za izvođenje njima pridruženih funkcija generatrisa. Takođe, rekurzivna definicija skupa kombinatornih objekata prirodno definiše funkcionalnu jednačinu po nepoznatoj funkciji generatrisi. Demonstrirajmo ovo na sledećim primerima.

Primer 3.13. Analogno primeru 3.7, možemo razmatrati skup svih reči nad azbukom Σ. Primetimo da se na osnovu primera 3.12 skup Σ^* može rekurzivno predstaviti na sledeći način:

$$\Sigma^* = \{\epsilon\} + \Sigma \times \Sigma^*$$

Primetimo najpre da skup Σ prebrojava funkcija generatrisa σz zato što se ovaj skup sastoji iz σ objekata veličine 1. Odavde, na osnovu teoreme 3.3 izvodimo funkcionalnu jednačinu po funkciji generatrisi $P(z)$ skupa Σ^*:

$$P(z) = 1 + \sigma z P(z)$$

kao i u primeru 3.7. Alternativno, ako primetimo da je:

$$\Sigma^* = \sum_{n\geq 0} \Sigma^n$$

tada se, ponovo na osnovu teoreme 3.3, rešenje može dobiti i direktno:

$$P(z) = \frac{1}{1 - \sigma z}$$

Primer 3.14. Skup svih binarnih stabala T se može, na osnovu primera 3.10, rekurzivno opisati na sledeći način:

$$T = \{e\} + T \times \{r\} \times T$$

Primetimo da skup $\{r\}$ prebrojava funkcija generatrisa z, jer sadrži samo jedan objekat veličine 1. Odavde se dobija funkcionalna jednačina po nepoznatoj funkciji generatrisi $T(z)$ skupa T:

$$T(z) = 1 + T(z) \cdot z \cdot T(z) = 1 + zT(z)^2$$

kao i u primeru 3.8.

Primer 3.15. Skup svih uopštenih stabala T se može, na osnovu primera 3.11, rekurzivno opisati na sledeći način:

$$T = \{v\} \times (\sum_{n \geq 0} T^n)$$

Primetimo da skup $\{v\}$ prebrojava funkcija generatrisa z, jer sadrži samo jedan objekat veličine 1. Odavde se odmah dobija funkcionalna jednačina po nepoznatoj funkciji generatrisi $T(z)$ skupa T:

$$T(z) = z \cdot \frac{1}{1 - T(z)}$$

kao i u primeru 3.9.

Čitalac zainteresovan za više informacija o simboličkom metodu i funkcijama generatrisama može pogledati knjigu [1]. Više o primeni funkcija generatrisa u kombinatorici može se pročitati u knjizi [2].

3.4 Bivarijantne funkcije generatrise. Primena u analizi algoritama

Do sada smo upoznali osnovne tehnike prebrojavanja skupova kombinatornih objekata, pri čemu smo objekte razvrstavali po veličini. U slučaju analize algoritama, ovo nije dovoljno. Na osnovu definicije 1.2 znamo da nam je za izračunavanje prosečne cene objekata veličine n potrebno da znamo i koliko objekata veličine n ima cenu k, za svako $k \geq 0$. Ovo znači da nam je potrebno dodatno razvrstavanje objekata po ceni. Drugim rečima, dolazimo do dvodimenzionog niza $p_{n,k}$, kao u definiciji 1.2. Obične funkcije generatrise u ovom slučaju nisu dovoljne, zato što se one pridružuju jednodimenzionim nizovima. Zato uvodimo sledeću definiciju.

Definicija 3.6. *Neka je $(p_{n,k})_{n,k\geq 0}$ dvodimenzioni niz. Tada funkciju:*

$$P(u,z) = \sum_{n,k\geq 0} p_{n,k} u^k z^n$$

nazivamo bivarijantnom funkcijom generetrisom, koja je pridružena ovom dvodimenzionom nizu. Koristimo notaciju $[z^n][u^k]P(u,z)$ za koeficient $p_{n,k}$, tj. za koeficient uz $z^n u^k$.

Neka je dat skup P kombinatornih objekata koji predstavljaju moguće ulazne instance nekog algoritma. Neka je $p_{n,k}$ broj objekata skupa P veličine n i sa cenom k. Bivarijantna funkcija generatrisa koja je pridružena ovom dvodimenzionom nizu u sebi sadrži sve informacije o raspodeli po veličinama i cenama objekata skupa P. O ovome govori sledeća teorema.

Teorema 3.4. *Ako je $p_{n,k}$ broj kombinatornih objekata $p \in P$, veličine n i cene k i ako je $P(u,z) = \sum_{n,k\geq 0} p_{n,k} u^k z^n$ odgovarajuća bivarijantna funkcija generatrisa, tada važi:*

- $P(1,z) = \sum_{n\geq 0} p_n z^n$ *– funkcija generatrisa koja prebrojava skup P.*
- $\frac{\partial P}{\partial u}(u,z)\big|_{u=1} = \sum_{n\geq 0} \sigma_n z^n = C(z)$ *– kumulativna funkcija generatrisa.*
- $\frac{[z^n]\frac{\partial P}{\partial u}(u,z)\big|_{u=1}}{[z^n]P(1,z)} = c_n$ *– prosečna cena objekata veličine n.*
- $\frac{[z^n]\frac{\partial^2 P}{\partial u^2}(u,z)\big|_{u=1}}{[z^n]P(1,z)} + \frac{[z^n]\frac{\partial P}{\partial u}(u,z)\big|_{u=1}}{[z^n]P(1,z)} - \left(\frac{[z^n]\frac{\partial P}{\partial u}(u,z)\big|_{u=1}}{[z^n]P(1,z)}\right)^2$ *– disperzija slučajne veličine c, tj. kvadrat standardnog odstupanja od očekivane cene.*

U formulaciji teoreme uveden je pojam *kumulativne funkcije generatrise* $C(z) = \sum_{n\geq 0} \sigma_n z^n$. Ovo je u stvari obična funkcija generatrisa koja je pridružena nizu $(\sigma_n)_{n\geq 0}$ kumulativnih cena. Ako znamo sumu svih cena objekata veličine n, tada se deljenjem sa brojem objekata ove veličine dobija prosečna cena, što nam i jeste krajnji cilj. Dakle, put ka rešenju je sledeći:

- Odredimo odgovarajuću bivarijantnu funkciju generatrisu.
- Na osnovu ove funkcije i gornje teoreme izračunamo kumulativnu funkciju generatrisu, kao i funkciju generatrisu koja prebrojava skup kombinatronih objekata P (po veličinama).
- Količnik koeficienata uz n-ti stepen ovih stepenih redova je upravo tražena prosečna cena svih objekata veličine n.

Demonstrirajmo ovo na jednom primeru.

Primer 3.16. Neka je dat skup $B = \{0,1\}^*$ svih niski bitova i neka je cena $c(s)$ broj jedinica u niski s. Tada možemo izvesti sledeću bivarijantnu funkciju generatrisu:

$$\begin{aligned} B(u,z) &= \sum_{n,k\geq 0} p_{n,k} u^k z^n \\ &= \sum_{s\in B} u^{c(s)} z^{|s|} \\ &= 1 + \sum_{bs'\in B} u^{c(bs')} z^{|bs'|} \\ &= 1 + \sum_{s'\in B} u^{c(s')} z^{|s'|+1} + \sum_{s'\in B} u^{c(s')+1} z^{|s'|+1} \\ &= 1 + z\sum_{s'\in B} u^{c(s')} z^{|s'|} + zu\sum_{s'\in B} u^{c(s')} z^{|s'|} \\ &= 1 + zB(u,z) + zuB(u,z) \end{aligned}$$

Ovde smo, kao i u primeru 3.7, najpre iskoristili rekurzivnu definiciju reči nad azbukom, a zatim smo reči dužine bar jedan razbili na dve klase – one koje počinju jedinicom i one koje počinju nulom. U prvom slučaju cena reči je za jedan veća od cene ostatka reči (bez prvog slova) dok je u drugom slučaju cena jednaka ceni ostatka reči. Iz gornjeg izvođenja sledi da je:

$$B(u,z) = \frac{1}{1-z-uz}$$

Odavde dobijamo da je $\frac{\eth B(u,z)}{\eth u} = \frac{z}{(1-z-uz)^2}$, što za $u = 1$ daje kumulativnu funkciju generatrisu $C(z) = \frac{z}{(1-2z)^2} = \sum_{n\geq 1} n2^{n-1}z^n$. S druge strane, $B(1,z) = \frac{1}{1-2z} = \sum_{n\geq 0} 2^n z^n$. Odavde dobijamo da je prosečna cena $c_n = \frac{n2^{n-1}}{2^n} = \frac{n}{2}$, što je i bilo za pretpostaviti.

Sličan rezultat bi se dobio i korišćenjem simboličkog metoda:

$$B = \{\epsilon\} + \{0,1\} \times B$$

Imajući u vidu da skupu $\{0,1\}$ odgovara bivarijantna funkcija generatrisa $z+uz$ (jedan objekat veličine jedan i cene nula i jedan objekat veličine jedan i cene jedan), sledi da je[13]:

$$B(u,z) = 1 + (z+uz)B(u,z)$$

tj. dobijamo istu funkcionalnu jednačinu.

Prethodni primer ima i praktičnu primenu u analizi algoritama – ako razmatramo algoritam množenja neoznačenih binarnih brojeva, tada je jasno da će broj dodavanja množenika, a time i potrebno vreme izvršenja operacije

[13]Kada je u pitanju simbolički metod, za bivarijantne funkcije generatrise važi teorema analogna teoremi 3.3.

zavisiti direktno od broja jedinica u binarnom zapisu množioca. Rezultat ovog primera nam govori da se množenje n-tocifrenim binarnim brojem u proseku svodi na $n/2$ sabiranja binarnih brojeva.

Bivarijantnu funkciju generatrisu nije uvek tako lako izračunati. Međutim, ona nam često nije neophodna – ako uspemo da nekim lakšim putem dođemo do kumulativne funkcije generatrise, problem nalaženja prosečne cene biće rešen i bez poznavanja bivarijantne funkcije generatrise. Demonstrirajmo ovo na jednom primeru.

Primer 3.17. Uz oznake kao u prethodnom primeru, kumulativnu funkciju generatrisu $C(z) = \sum_{n\geq 0} \sigma_n z^n$ možemo izračunati i direktno, na sledeći način:

$$\begin{aligned} C(z) &= \textstyle\sum_{n\geq 0} \sigma_n z^n \\ &= \textstyle\sum_{s\in B} c(s) z^{|s|} \\ &= \textstyle\sum_{bs'\in B} c(bs') z^{|bs'|} \\ &= \textstyle\sum_{s'\in B} (c(s')+1) z^{|s'|+1} + \sum_{s'\in B} c(s') z^{|s'|+1} \\ &= 2z \textstyle\sum_{s'\in B} c(s') z^{|s'|} + \sum_{s'\in B} z^{|s'|+1} \\ &= 2zC(z) + \frac{z}{1-2z} \end{aligned}$$

Odavde je $C(z) = \frac{z}{(1-2z)^2}$, kao i u prethodnom primeru.

Sledeći primer prikazuje nešto složenije izvođenje.

Primer 3.18. Neka je kao i u prethodnom primeru dat skup $B = \{0,1\}^*$ svih niski bitova, ali neka je ovoga puta cena $c(s)$ niske s definisana kao broj „prelaza" sa nule na jedinicu ili obrnuto (npr. cena niske 0111101 je 3). Definišimo B_0 kao skup svih niski bitova koji počinju nulom, a B_1 kao skup svih niski bitova koje počinju jedinicom. Tada važi:

$$\begin{aligned} B &= B_0 + B_1 + \{\epsilon\} \\ B_0 &= \{0\} \times B_0 + \{0\} \times B_1 + \{0\} \\ B_1 &= \{1\} \times B_0 + \{1\} \times B_1 + \{1\} \end{aligned}$$

Dakle, svaka reč je ili prazna ili počinje nulom ili počinje jedinicom. Reč koje počinje nulom je ili reč 0 ili nakon početne nule sledi reč koja počinje nulom, ili nakon početne nule sledi reč koja počinje jedinicom. Analogno važi za reč koja počinje jedinicom. Na osnovu simboličkog metoda, dobijamo:

$$\begin{aligned} B(u,z) &= B_0(u,z) + B_1(u,z) + 1 \\ B_0(u,z) &= zB_0(u,z) + uzB_1(u,z) + z \\ B_1(u,z) &= uzB_0(u,z) + zB_1(u,z) + z \end{aligned}$$

Pri tom su funkcije $B_0(u,z)$ i $B_1(u,z)$ redom bivarijantne funkcije generatrise pridružene skupovima B_0 i B_1. Početno slovo može imati cenu 0 ili 1 u zavisnosti od toga koje slovo sledi nakon njega, tj. da li postoji „prelaz" ili

ne. Upravo zato početnom slovu u nekim situacijama odgovara bivarijantna funkcija generatrisa z, a u nekim uz. Rešavanjem sistema jednačina, po tri nepoznate funkcije, dobijamo da je:

$$B_0(u,z) = B_1(u,z) = \frac{z}{1-z-uz}$$

i da je:

$$B(u,z) = B_0(u,z) + B_1(u,z) + 1 = \frac{1+z-uz}{1-z-uz}$$

Diferenciranjem po u i za $u = 1$ dobijamo kumulativnu funkciju generatrisu:

$$\left.\frac{\eth B(u,z)}{\eth u}\right|_{u=1} = \left.\frac{2z^2}{1-z-uz)^2}\right|_{u=1} = \frac{2z^2}{(1-2z)^2}$$

S druge strane je:

$$B(1,z) = \frac{1}{1-2z}$$

kao što je i očekivano. Prosečna cena je:

$$c_n = \frac{[z^n]\left.\frac{\eth B}{\eth u}(u,z)\right|_{u=1}}{[z^n]B(1,z)} = \frac{(n-1)2^{n-1}}{2^n} = \frac{n-1}{2}$$

Dakle, u proizvoljnoj niski bitova dužine n ima u proseku $\frac{n-1}{2}$ prelaza.

Ovaj rezultat takođe ima veoma značajnu praktičnu primenu – u slučaju množenja označenih binarnih brojeva *Butovim algoritmom*, operacije sabiranja ili oduzimanja množenika se obavljaju samo ako postoji „prelaz" između dva susedna bita u množiocu. Samim tim se množenje dva označena broja svodi na u proseku $(n-1)/2$ sabiranja ili oduzimanja binarnih brojeva, u skladu sa rezultatom prethodnog primera.

4 Zaključak

U prethodnim poglavljima smo se upoznali sa različitim pristupima analizi algoritama i na primerima demonstrirali njihov odnos. Analizom najgoreg slučaja utvrđujemo gornju granicu vremena izvršavanja datog algoritma za ulaze određene veličine. Dobijena gornja granica predstavlja vreme koje je dovoljno za izvršavanje algoritma bez obzira na ulaznu instancu date veličine. S druge strane, postoje algoritmi koji u prosečnom slučaju daju znatno bolje rezultate nego u najgorem slučaju. Jasno je da ovakve algoritme ne treba odbaciti, jer je u praksi prosečno vreme izvršavanja najbolja mera upotrebljivosti algoritma. Upravo se u ovome ogleda značaj

analize prosečnog slučaja, jer se takvom analizom prepoznaju algoritmi koji daju dobre prosečne rezultate.

Analiza prosečnog slučaja se oslanja na teoriju verovatnoće, jer se prosečno vreme izvršavanja algoritma može razmatrati kao matematičko očekivanje odgovarajuće slučajne veličine. Ulazne instance se razmatraju kao ishodi slučajnog eksperimenta, na osnovu čega se određuje raspodela slučajne veličine koja se razmatra i izračunava njena očekivana vrednost. Određivanje raspodele se svodi na prebrojavanje skupova kombinatornih objekata.

Rekurentne relacije se u analizi algoritama intenzivno koriste, kako u analizi najgoreg slučaja, tako i u analizi prosečnog slučaja. Prilikom analize prosečnog slučaja po pravilu se dobijaju komplikovanije rekurentne relacije, a i njihovo izvođenje je znatno teže. Funkcije generatrise mogu rešiti oba ova problema. Sa jedne strane, funkcije generatrise se mogu koristiti za jednostavnije rešavanje rekurentnih relacija, njihovim svođenjem na funkcionalne jednačine po nepoznatoj funkciji generatrisi. Sa druge strane, ove funkcionalne jednačine se mogu izvesti i direktno, potpuno zaobilazeći rekurentne relacije, čime se zaobilaze i problemi koji nastaju prilikom izvođenja i trensformacija rekurentnih relacija.

Simbolički metod je jednostavan i elegantan pristup prebrojavanju skupova kombinatornih objekata. Ovim metodom se skupu kombinatornih objekata direktno na osnovu njegove rekurzivne definicije dodeljuje funkcionalna jednačina po nepoznatoj funkciji generatrisi, primenjujući veoma jednostavna i prirodna pravila. Rešavanjem dobijene funkcionalne jednačine dobijamo funkciju generatrisu koja prebrojava dati skup.

Prilikom analize prosečnog slučaja, kombinatorni objekti koji predstavljaju ulazne instance datog algoritma razvrstavaju se po dva različita parametra – po veličini i po ceni. Rezultat ovakvog razvrstavanja je dvodimenzioni niz koji se predstavlja bivarijantnom funkcijom generatrisom. Prilikom određivanja bivarijantne funkcije generatrise takođe se može koristiti simbolički metod. Bivarijantna funkcija generatrisa u sebi sadrži sve informacije o raspodeli, tako da je njeno poznavanje dovoljno da se izračunaju sve veličine od interesa (prosečna cena, standardno odstupanje, momenti višeg reda).

U ovom radu predstavljen je jedan zanimljiv pristup analizi algoritama. Uspostavljena je veza između analize algoritama i mnogih drugih matematičkih disciplina. Iznad svega, demonstriran je značaj analize algoritama, kako u teoriji, tako i u praksi. Imajući u vidu težinu matematičkih problema sa kojima se danas suočavamo i stalnu potrebu za efikasnijim rešenjima, značaj analize algoritama u narednim godinama biće sve veći, a tehnike analize poput tehnika koje su demonstrirane u ovom radu biće sve dragocenije.

Literatura

[1] Robert Sedgewick, Philippe Flajolet: *An introduction to the analysis of algorithms*, Addison-Wesley, 1996.

[2] James A. Anderson: *Diskretna matematika sa kombinatorikom*, CET, 2005.

[3] Miodrag Živković: *Algoritmi*, Matematički Fakultet, Beograd.