Klasifikacija vjerojatnosnih algoritama

Peterfaj, Ana

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
MATEMATIČKI ODSJEK

Ana Peterfaj

KLASIFIKACIJA VJEROJATNOSNIH ALGORITAMA

Diplomski rad

Voditelj rada:
prof. dr. sc. Mladen Vuković

Zagreb, 2019. god.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

Uvod

Deterministički algoritam je algoritam koji će pri svakom izvršavanju u bilo kojim uvjetima za isti unos dati isti izlaz slijedeći svaki put isti niz naredbi. Za razliku od determinističkih, postoje i stohastički algoritmi koji barem u jednom dijelu izvršavanja donose odluku o daljnjem tijeku izvršavanja slučajnim odabirom. To bi značilo da za iste ulaze, pod istim uvjetima, isti stohastički algoritmi mogu dati različite izlaze. Dakle, algoritme s obzirom na način donošenja odluka možemo podijeliti na stohastičke i determinističke. Algoritmi koje proučavamo u ovom radu su vjerojatnosni algoritmi i oni su posebna vrsta stohastičkih algoritama.

Kvaliteta stohastičkih algoritama mjeri se efikasnošću (vremenom izvršavanja) i točnošću (vjerojatnošću dobivanja točnog rezultata za slučajno odabrani ulaz). Primijetimo da vrijeme izvršavanja kod općenitih stohastičih algoritama ovisi o ulazu. Vjerojatnosni algoritmi, rekli smo, su posebna vrsta stohastičkih algoritama, ali takvih da su za sve ulaze efikasni, tj. vrijeme izvršavanja je dovoljno kratko i daju točne rezultate s velikom vjerojatnošću.

Uvođenjem slučajnih događaja u algoritme, na neki način smo unijeli nesigurnost u rezultate programa. Postavlja se pitanje zašto bismo uopće koristili računalo ako rezultat možda neće biti točan. Naime, pokazalo se da su vjerojatnosni algoritmi često efikasniji i jednostavniji od uobičajenih determinističkih algoritama. Drugim riječima, kod nekih problema vrijedi „žrtvovati" točnost da bismo dobili na brzini. Pokazalo se da su neki vjerojatnosni algoritmi koji imaju visoku točnost pouzdaniji od njihovih determinističkih inačica. Naime, teorijski gledano, deterministički algoritmi su apsolutno točni, dok vjerojatnosni algoritmi mogu dati pogrešno rješenje. Ali u praksi ni deterministički algoritmi nisu 100% pouzdani jer tijekom njihovog izvršavanja može doći do kvara na hardveru računala što može rezultirati krivim rezultatom, a vjerojatnost pojave takvog kvara povečava se proporcionalno s vremenom izvršavanja algoritma. Stoga, efikasni vjerojatnosni algoritmi su pouzdaniji u tom smislu od svojih sporih determinističkih verzija. Isto tako za neke probleme ne postoje efikasni egzaktni algoritmi pa je potrebno pribjeći alternativnim načinima rješavanja. Primjerice za računanje volumena n-dimenzionalnog konveksnog tijela ne znamo bolji algoritam od onog vjerojatnosnog. Stoga ima smisla proučavati ovakvu vrstu algoritama, čemu je i posvećen ovaj rad.

Prvi vjerojatnosni algoritam osmislio je Michael O. Rabin za problem najbližih točaka[1]. Proučavanje vjerojatnosnih algoritama potaknuto je 1977. godine otkrićem vjerojatnosnog algoritma za testiranje prostosti brojeva (za dani broj algoritam vraća je li prost ili nije), budući da u to vrijeme nije bio poznat niti jedan efikasan deterministički algoritam za testiranje prostih brojeva.

Diplomski rad se sastoji od 3 poglavlja. U Poglavlju 1 ćemo uvesti osnovne pojmove iz teorije vjerojatnosti koji su nam neophodni za razumijevanje daljnjeg rada. U Poglavlju 2 opisat ćemo dva najčešća teorijska modela vjerojatnosih algoritama i analizirati ih na odgovarajućim primjerima. U Poglavlju 3 detaljno ćemo opisati i dati primjere za pojedine vrste vjerojatnosnih algoritama: Las Vegas algoritme, algoritme s jednostranom greškom, algoritme s ograničenom greškom te algoritme s neograničenom greškom.

[1] za danih n točaka u metričkom prostoru, potrebno je naći par točaka s najmanjom udaljenošću između njih.

Poglavlje 1

Osnove teorije vjerojatnosti

U ovom poglavlju ćemo uvesti osnovne pojmove iz teorije vjerojatnosti koji su nužni za daljnje razumijevanje rada.

Teorija vjerojatnosti je razvijena za modeliranje i analizu situacija i eksperimenata čiji rezultati mogu biti različiti. Poznati primjer je bacanje kocke sa šest strana. Rezultat može biti bilo koji od brojeva označenih od 1 do 6, a vjerojatnost dobivanja svakog broja je $\frac{1}{6}$.

Ako ne postoji mogućnost predviđanja rezultata eksperimenta, tada govorimo o slučajnim događajima. Tijekom modeliranja vjerojatnosnog eksperimenta, razmatramo sve mogućnosti rezultata, koje zovemo elementarnim događajima. U našem primjeru s bacanjem kocke imamo šest elementarna događaja: 1, 2, 3, 4, 5 i 6. Skup svih elementarnih događaja nekog eksperimenta zove se prostor elementarnih događaja.

Događaj je skup koji sadrži elementarne događaje. Primjerice, $\{1, 3, 5\}$ je događaj bacanja neparnog broja na kocki. Primijetimo da je svaki elementarni događaj ujedno i događaj koji se sastoji od samo jednog elementa u skupu.

Sada možemo navesti definiciju vjerojatnosti.

Definicija 1.1. Neka je S prostor elementarnih događaja nekog vjerojatnosnog eksperimenta. Vjerojatnosna distribucija na S je svaka funkcija

$$\text{Prob} : \mathcal{P}(S) \to [0, 1]$$

koja zadovoljava sljedeće uvijete:

- $\text{Prob}(\{x\}) \geq 0$ za svaki elementarni događaj x
- $\text{Prob}(S) = 1$
- $\text{Prob}(A \cup B) = \text{Prob}(A) + \text{Prob}(B)$ za sve događaje $A, B \subseteq S$, takve da $A \cap B = \emptyset$

Vrijednost $\text{Prob}(A)$ zovemo vjerojatnost događaja A. Par (S, Prob) zovemo vjerojatnosni prostor. $\triangleleft$

Bez dokaza navodimo neka svojstva koja vrijede za svaki vjerojatnosni prostor (S, Prob).

Propozicija 1.2. *Neka je* (S, Prob) *vjerojatnosni prostor. Tada:*

- $\text{Prob}(\emptyset) = 0$
- $\text{Prob}(S \setminus A) = 1 - \text{Prob}(A)$*, za svaki* $A \subseteq S$
- $\text{Prob}(A) \leq \text{Prob}(B)$*, za skupove* $A \subseteq B \subseteq S$
- $\text{Prob}(A \cup B) = \text{Prob}(A) + \text{Prob}(B) - \text{Prob}(A \cap B)$*, za sve* $A, B \subseteq S$
- $\text{Prob}(A) = \sum_{x \in A} \text{Prob}(x)$*, za svaki* $A \subseteq S$

Vratimo se na primjer s bacanjem kocke. Zamislimo da imamo 3 bacanja. Znamo da smo u ta tri bacanja dobili barem jednu šesticu i zanima nas kolika je vjerojatnost da smo dobili barem dvije šestice. Ovu vrstu vjerojatnosti nazivamo uvjetovana vjerojatnost.

Definicija 1.3. Neka je (S, Prob) vjerojatnosni prostor. Neka su A i B događaji takvi da $A, B \subseteq S$ i neka je $\text{Prob}(B) \neq 0$. Uvjetovana vjerojatnost događaja A, ako znamo da se događaj B dogodio, je

$$\text{Prob}(A|B) = \frac{\text{Prob}(A \cap B)}{\text{Prob}(B)}$$

$\triangleleft$

Primijetimo da je definicija intuitivna jer se u $A \cap B$ nalaze elementarni događaji koji su i u A i u B. $\text{Prob}(B)$ se nalazi u nazivniku jer znamo da se događaj B dogodio, stoga događaji iz $S \setminus B$ više nisu mogući.

Definicija 1.4. Neka je (S, Prob) vjerojatnosni prostor. Dva događaja A i B su nezavisna ako

$$\text{Prob}(A \cap B) = \text{Prob}(A) \cdot \text{Prob}(B)$$

$\triangleleft$

Lema 1.5. *Neka je* (S, Prob) *vjerojatnosni prostor. Neka su* $A, B \subseteq S$ *i* $\text{Prob}(B) \neq 0$*. Tada,* A *i* B *su nezavisni događaji ako i samo ako*

$$\text{Prob}(A|B) = \text{Prob}(A)$$

Dokaz. Neka su A i B nezavisni i $\text{Prob}(B) \neq 0$. Tada prema definiciji 1.4 imamo

$$\text{Prob}(A \cap B) = \text{Prob}(A) \cdot \text{Prob}(B)$$

Stoga,

$$\text{Prob}(A|B) = \frac{\text{Prob}(A \cap B)}{\text{Prob}(B)} = \frac{\text{Prob}(A) \cdot \text{Prob}(B)}{\text{Prob}(B)} = \text{Prob}(A)$$

Dokažimo drugu stranu bikondicionala. Neka je $\text{Prob}(A|B) = \text{Prob}(A)$ i $\text{Prob}(B) \neq 0$. Tada koristeći definiciju uvjetovane vjerojatnosti imamo

$$\text{Prob}(A) = \text{Prob}(A|B) = \frac{\text{Prob}(A \cap B)}{\text{Prob}(B)}$$

Množeći prethodnu jednakost s $\text{Prob}(B)$ dobivamo traženi rezultat

$$\text{Prob}(A \cap B) = \text{Prob}(A) \cdot \text{Prob}(B)$$

□

Sada, uvodimo termin koji je važan za analizu slučajnih eksperimenata, a kasnije i za analizu vjerojatnosnih algoritama.

Definicija 1.6. Neka je S konačan prostor elementarnih događaja vjerojatnosnog eksperimenta. Svaka funkcija X sa S na $\mathbb{R}$ zove se slučajna varijabla na S. ◁

Poanta je da možemo izabrati slučajnu varijablu X, i koristiti je s ciljem da izrazimo što nas u eksperimentu zanima. Primjerice, kasnije ćemo razmatrati vjerojatnosni prostor kao vjerojatnosnu distribuciju na svim izračunavanjima vjerojatnosnog algoritma s fiksnim ulazom. Kada želimo odrediti efikasnost vjerojatnosnog algoritma na njegovom ulazu, tada je prigodno izabrati slučajnu varijablu koja dodjeljuje duljinu svakom izračunavanju. Koristeći takvu slučajnu varijablu, možemo odrediti očekivanu složenost algoritma kao težinsku srednju vrijednost slučajnih varijabli, gdje su težine zapravo vjerojatnosti pojedinih izračunavanja.

Drugi prigodni odabir slučajne varijable može biti pridruživanje „1” ako algoritam vraća točan rezultat i „0” ako ne. Na taj način možemo računati vjerojatnost dobivanja točnog rezultate na danom ulazu.

Kada je S konačan skup, tada je i sljedeći skup konačan za svaku slučajnu varijablu X sa S na $\mathbb{R}$

$$\mathbb{R}_X = \{x \in \mathbb{R} \,|\, \exists s \in S, \text{ takav da } X(s) = x\}$$

Definicija 1.7. Neka je (S, Prob) vjerojatnosni prostor, a X slučajna varijabla na S. Za svaki $z \in \mathbb{R}$, definiramo događaj $X = z$ kao

$$\text{Event}(X = z) = \{s \in S \,|\, X(s) = z\}$$

Funkciju $f_X(z) = \text{Prob}(\text{Event}(X = z))$, za svaki $z \in \mathbb{R}$ zovemo funkcijom gustoće slučajne varijable X. Funkcija distribucije za X je funkcija $\text{Dis}_X : \mathbb{R} \to [0, 1]$ definirana sa

$$\text{Dis}_X(z) = \text{Prob}(X \leq z) = \sum_{y \leq z,\, y \in \mathbb{R}_X} \text{Prob}(\text{Event}(X = y))$$

◁

Koristit ćemo kraću notaciju: $X = z$ umjesto $\text{Event}(X = z)$ te $\text{Prob}(X = z)$ umjesto $\text{Prob}(\text{Event}(X = z))$.

Lema 1.8. *Neka je* (S, Prob) *vjerojatnosni prostor, a* X *slučajna varijabla na* S. *Tada za svaki* $z \in \mathbb{R}$ *vrijedi:*

1. $\text{Prob}(X = z) \geq 0$
2. $\sum_{y \in \mathbb{R}_X} \text{Prob}(X = y) = 1$
3. $f_X(z) = \text{Prob}(X = z) = \sum_{s \in S,\, X(s) = z} \text{Prob}(\{s\})$

Dokaz. Prvo svojstvo je očito jer je $X = z$ događaj iz S. Treće svojstvo je direktna posljedica definicije funkcije gustoće f_X i činjenice da je Prob vjerojatnosna distribucija na S. Drugo svojstvo je direktna posljedica trećeg svojstva jer

$$\sum_{y \in \mathbb{R}_\mathbb{X}} \text{Prob}(X = y) = \sum_{y \in \mathbb{R}_\mathbb{X}} \sum_{s \in S,\, X(s) = y} \text{Prob}(\{s\}) = \sum_{s \in S} \text{Prob}(\{s\}) = 1$$

□

Zanimljiva posljedica leme 1.8 je što odabirom slučajne varijable na S iz vjerojatnosnog prostora (S, Prob), stvaramo novi vjerojatnosni prostor $(\mathbb{R}_X, F_X)$, gdje je

$$F_X(B) = \sum_{y \in B} f_X(y), \quad \text{za svaki } B \subseteq \mathbb{R}_X$$

Sada ćemo definirati neovisnost dviju slučajnih varijabli kao prirodnu generalizaciju definicije dvaju nezavisnih događaja.

Definicija 1.9. Neka je (S, Prob) vjerojatnosni prostor, neka su X i Y dvije slučajne varijable na S. Uvodimo notaciju:

$$\text{Event}(X = x \text{ i } Y = y) = \text{Event}(X = x) \cap \text{Event}(Y = y)$$

Kažemo da su dvije slučajne varijbale X i Y nezavisne, ako za sve $x, y \in \mathbb{R}$ vrijedi:

$$\text{Prob}(X = y \text{ i } Y = y) = \text{Prob}(X = x) \cdot \text{Prob}(Y = y)$$

◁

Nama najkorisnija primjena distribucije slučajne varijable je njena težinska srednja vrijednost. Tu srednju vrijednost nazivamo očekivana vrijednost. Već smo spomenuli, glavna motivacija za uvođenje očekivane vrijednosti slučajne varijable je analiza efikasnosti i pouzdanosti vjerojatnosnih algoritama.

Definicija 1.10. Neka je (S, Prob) vjerojatnosni prostor, te neka je X slučajna varijabla na S. Očekivana vrijednost X-a je

$$\text{E}[X] = \sum_{x \in \mathbb{R}_X} x \cdot \text{Prob}(X = x)$$

ako je suma konačna ili ako apsolutno konvergira. $\triangleleft$

Sljedeća lema nam omogućava drugi način za računanje očekivane vrijednosti slučajne varijable X.

Lema 1.11. *Neka je (S, Prob) konačan vjerojatnosni prostor i neka je X slučajna varijabla na S. Tada*

$$E[X] = \sum_{s \in S} X(s) \cdot \text{Prob}(\{s\})$$

Dokaz. Za svaki slučajnu varijablu X na S po definiciji očekivane vrijednosti vrijedi:

$$\text{E}[X] = \sum_{x \in \mathbb{R}_X} x \cdot \text{Prob}(\{X = x\})$$

Budući da je $\text{Event}(X = x) = \{s \in S \mid X(s) = x\}$

$$\begin{aligned}
\text{E}[X] &= \sum_{x \in \mathbb{R}_X} x \cdot \sum_{s \in S,\, X(s) = x} \text{Prob}(\{s\}) \\
&= \sum_{x \in \mathbb{R}_X} \sum_{s \in S,\, X(s) = x} x \cdot \text{Prob}(\{s\}) \\
&= \sum_{x \in \mathbb{R}_X} \sum_{s \in S,\, X(s) = x} X(s) \cdot \text{Prob}(\{s\})
\end{aligned}$$

Budući da je X funkcija na S, za svaki $s \in S$ postoji točno jedan $x \in \mathbb{R}$ sa svojstvom $X(s) = x$, pa imamo

$$\text{E}[X] = \sum_{s \in S} X(s) \cdot \text{Prob}(\{s\})$$

$\square$

U analizi vjerojatnosnih algoritama često ćemo koristiti posebnu vrstu slučajnih varijabli koje ćemo zvati indikatorske varijable.

Definicija 1.12. Neka je (S, Prob) vjerojatnosni prostor, te neka je X slučajna varijabla na S. Slučajnu varijablu X zovemo indikatorskom varijablom ako poprima vrijednosti 0 ili 1, tj. ako je X funkcija sa S na $\{0, 1\}$. $\triangleleft$

Primijetimo da indikatorska varijabla razdvaja skup S na dva podskupa. Jedan sadrži sve elementarne događaje s iz S za koje vrijedi $X(s) = 1$, a drugi elementarne događaje u iz S za koje vrijedi $X(u) = 0$.

To često koristimo u praksi tako da ako je S skup svih izračunavanja vjerojatnosnog algoritma, tada skup S možemo podijeliti na skup svih izračunavanja koja daju točan rezultat te na skup svih izračunavanja koja daju netočan rezultat.

Općenito, za svaki događaj $A \subseteq S$ možemo definirati indikatorsku varijablu X_A tako da

$$A = \{s \in S \mid X_A(s) = 1\} = \text{Event}(X_A = 1)$$
$$S \setminus A = \{s \in S \mid X_A(s) = 0\} = \text{Event}(X_A = 0)$$

Lema 1.13. *Neka je (S, Prob) vjerojatnosni prostor. Za svaku indikatorsku varijablu X_A na S s događajem $A = \{s \in S \mid X_A(s) = 1\}$ vrijedi*

$$E[X_A] = \text{Prob}(A)$$

Dokaz. Prema lemi 1.11 imamo

$$\text{E}[X_A] = \sum_{s \in S} X_A(s) \cdot \text{Prob}(\{s\})$$

Particiramo skup S na A i $S \setminus A$

$$\text{E}[X_A] = \sum_{s \in S} X_A(s) \cdot \text{Prob}(\{s\}) + \sum_{s \in S \setminus A} X_A(s) \cdot \text{Prob}(\{s\})$$

Koristeći pretpostavku leme dobivamo

$$\begin{aligned}\text{E}[X_A] &= \sum_{s \in S} 1 \cdot \text{Prob}(\{s\}) + \sum_{s \in S \setminus A} 0 \cdot \text{Prob}(\{s\}) \\ &= \sum_{s \in A} \text{Prob}(\{s\}) \\ &= \text{Prob}(A)\end{aligned}$$

□

Prije nego dokažemo svojstvo linearnosti očekivane vrijednosti, primijetimo da za svaku slučajnu varijablu X na S i bilo koju funkciju $g : \mathbb{R} \to \mathbb{R}$, funkcija $Z = g(X)$ definirana sa $Z(s) = g(X(s))$, za svaki $s \in S$, je također slučajna varijabla na S. Ova tvrdnja izravno povlači svojstvo koje zovemo slaba linearnost očekivane vrijednosti:

$$\text{E}[a \cdot X + b] = a \cdot \text{E}[X] + b$$

gdje su $a, b \in \mathbb{R}$.

Sada slijedi dokaz svojstva linearnosti očekivane vrijednosti.

Lema 1.14. *Neka je* (S, Prob) *vjerojatnosni prostor. Neka su* X *i* Y *dvije slučajne varijable na* S*. Tada*

$$E[X + Y] = E[X] + E[Y]$$

Dokaz. Označimo sa Z slučajnu varijablu $X + Y$. Tada prema lemi 1.11:

$$\text{E}[X + Y] = \text{E}[Z] = \sum_{s \in S} Z(s) \cdot \text{Prob}(\{s\})$$

Koristeći definiciju slučajne varijable Z:

$$\begin{aligned}
\text{E}[X + Y] &= \sum_{s \in S} (X(s) + Y(s)) \cdot \text{Prob}(\{s\}) \\
&= \sum_{s \in S} X(s) \cdot \text{Prob}(\{s\}) + \sum_{s \in S} Y(s) \cdot \text{Prob}(\{s\}) \\
&= \text{E}[X] + \text{E}[Y]
\end{aligned}$$

□

Indukcijom se lako dokaže da vrijedi i generalizacija na više pribrojnika:

$$\text{E}[X_1 + \cdots + X_n] = \text{E}[X_1] + \cdots + \text{E}[X_n]$$

Poglavlje 2

Teorijski modeli vjerojatnosnih algoritama

Cilj ovog poglavlja je pokazati kako se vjerojatnosni algoritmi mogu modelirati pomoću vjerojatnosnih prostora. Već je rečeno da se kvaliteta vjerojatnosnih algoritama mjeri efikasnošću i točnošću pa ćemo u tu svrhu pokazati kako se koncept slučajnih varijabli može koristiti za analizu izračunavanja vjerojatnosnih algoritama. Prema [3] postoje dva teorijska modela za opisivanje algoritama. Objasnit ćemo i analizirati oba te navesti primjere algoritama koji su građeni po odgovarajućem modelu.

2.1 Prvi model

Počet ćemo s jednostavnijim od dva modela. Prvi model promatra vjerojatnosni algoritam A kao distribuciju vjerojatnosti na konačnom skupu deterministčkih algoritama A_1, A_2, ..., A_n. To bi značilo da za svaki ulaz w, vjerojatnosni algoritam A slučajnim odabirom bira deterministički algoritam A_i i od tada A_i radi s ulazom w. Na ovaj način imamo točno n izračunavanja algoritma A za ulaz w, od kojih je svako izračunavanje od A dano izračunavanjem jednim od algoritama iz skupa $\{A_1, A_2, \ldots, A_n\}$. Dakle, modeliramo rad algoritma A za ulaz w kao vjerojatnosni prostor

$$(S_{A,w}, \text{Prob})$$

gdje je

$$S_{A,w} = \{A_1, A_2, \ldots, A_n\}$$

a Prob je distribucija vjerojatnosti na $S_{A,w}$. Ponekad želimo promatrati

$$S_{A,w} = \{C_1, C_2, \ldots, C_n\}$$

skup svih izračunavanja od A na w, gdje C_i označava izračunavanje algoritma A_i na w. Primijetimo da je izračunavanje C_i jednoznačno određeno algoritmom A_i i ulazom w, pa je zapravo svejedno promatramo li skup $S_{A,w}$ kao skup algoritama ili skup izračunavanja.

Izračunavanja iz skupa $\{C_1, C_2, \ldots, C_n\}$ mogu biti različitih duljina. Ako želimo proučavati efikasnost algoritma A s ulazom w, trebamo promatrati slučajnu varijablu

$$\text{Time} : S_{A,w} \to \mathbb{N}$$

gdje $\text{Time}(C_i)$ (za $i = 1, 2, \ldots, 3$) označava duljinu izračunavanja.

Sada možemo efikasnost algoritma A mjeriti **očekivanom vremenskom složenošću na w**:

$$\text{Exp-Time}_A(w) = \text{E}[\text{Time}] = \sum_{i=1}^{n} \text{Prob}(\{C_i\}) \cdot \text{Time}(C_i)$$

Ovo nam nema poseban značaj jer je to složenost samo za taj specifični ulaz. Htjeli bismo gornju granicu za očekivanu vremensku složenost algoritma A za bilo koji ulaz duljine n. Stoga definiramo **očekivanu vremensku složenost algoritma A**: $\text{Exp-Time}_A : \mathbb{N} \to \mathbb{N}$ koja je definirana s

$$\text{Exp-Time}_A(n) = \max\{\text{Exp-Time}_A(w) \,|\, \text{duljina ulaza } w \text{ je } n\}, \quad \text{za svaki } n \in \mathbb{N}$$

Neformalno rečeno, kako bismo dobili očekivanu vremensku složenost algoritma A za ulaz duljine n trebamo pogledati očekivane vremenske složenosti za svaki ulaz w koji imaju duljinu n i uzeti najveću. Time smo dobili gornju ogradu za očekivanu složenost algoritma A na w. Dakle, ni za jedan ulaz w fiksne duljine n, očekivana složenost na w neće biti veća od $\text{Exp-Time}_A(n)$.

Ako želimo strogu gornju ogradu vremenske složenosti za svako izvođenje vjerojatnosnog algoritma, koristimo **vremensku složenost algoritma A**: $\text{Time}_A(n) : \mathbb{N} \to \mathbb{N}$ definiranu s

$$\text{Time}_A(n) = \max\{\text{Time}(C) \,|\, C \text{ je izračunavanje od } A \text{ na ulazu duljine } n\}$$

Na ovaj način ćemo mjeriti efikasnost nekog algoritma pa nam još preostaje vidjeti kako ćemo mjeriti točnost. Za to će nam biti potrebna indikatorska varijabla X. Ako pretpostavimo da promatramo algoritam A s ulazom w, s već uvedenim oznakama C_i za izračunavanje algoritma A_i. **Indikatorsku varijablu** $X : S_{A,w} \to \{0, 1\}$ definiramo

$$X(C_i) = \begin{cases} 1, \text{ ako } C_i \text{ daje točan rezultat na } w \\ 0, \text{ ako } C_i \text{ daje krivi rezultat na } w \end{cases}$$

za $i = 1, \ldots, n$.

Sada je jasno da ako želimo računati **vjerojatnost uspjeha algoritma A za ulaz w** trebamo računati zbroj svih umnožaka vjerojatnosti da je izabrano izračunavanje C_i ($\text{Prob}(\{C_i\})$) i vrijednosti njegove indikatorske varijable ($X(C_i)$), tj:

$$\mathrm{E}[X] = \sum_{i=1}^{n} X(C_i) \cdot \mathrm{Prob}(\{C_i\})$$

Sumu možemo rastaviti na dva dijela: u prvoj sumi sumiramo po i-evima za koje vrijedi $X(C_i) = 1$. To je suma vjerojatnosti izračunavanja koja daju točan rezultat. U drugoj sumiramo po i-evima za koje izračunavanja C_i daju pogrešan rezultat.

$$\begin{aligned}\mathrm{E}[X] &= \sum_{X(C_i)=1} 1 \cdot \mathrm{Prob}(\{C_i\}) + \sum_{X(C_i)=0} 0 \cdot \mathrm{Prob}(\{C_i\}) = \\ &= \sum_{X(C_i)=1} \mathrm{Prob}(\{C_i\})\end{aligned}$$

Što je jednako vjerojatnosti da je X jednako 1, tj.

$$\mathrm{E}[X] = \mathrm{Prob}(\mathrm{Event}(X = 1)) \tag{2.1}$$

Ovo posljednje je jednako vjerojatnosti da algoritam A računa točan rezultat. Dakle, $\mathrm{E}[X]$ je vjerojatnost uspjeha algoritma A s ulazom w. Po toj logici, vrijednost

$$\mathrm{Error}_A(w) = 1 - \mathrm{E}[X] \tag{2.2}$$

nazivamo **vjerojatnost greške algoritma *A* s ulazom *w*.**

Slično kao i kod složenosti, definirat ćemo gornju ogradu za vjerojatnost greške algoritma A koja ovisi o duljini ulaza (a ne o samom ulazu) $\mathrm{Error}_A(n) : \mathbb{N} \to \mathbb{N}$ sa

$$\mathrm{Error}_A(n) = \max\{\mathrm{Error}_A(w) \,|\, \text{duljina od } w \text{ je } n\}$$

Drugim riječima vrijednost $\mathrm{Error}_A(n)$ nam kaže kolika će najviše biti vjerojatnost greške za svaki ulaz duljine n.

2.1.1 Protokol R_k

Objasnit ćemo i analizirati protokol R_k jer je upravo on dobar primjer vjerojatnosnog algoritma građenog po upravo opisanom modelu.

Pretpostavimo da imamo dva udaljena računala R_I i R_{II}. Svaki od njih ima svoju bazu podataka, pri čemu su te dvije baze jednakih veličina. Naš zadatak je provjeriti sadrže li njihove baze iste podatke.

Neka je n veličina baze podataka u bitovima. Željeli bismo konstruirati komunikacijski protokol koji određuje jesu li podaci na oba računala isti. U tom slučaju složenost komunikacije je broj razmijenjenih bitova između računala i željeli bismo da bude što manja. Problem je očito vrlo lako teorijski riješiti deterministič kim algoritmom. Dovoljno je da prvo računalo pošalje drugom svoje podatke i potom da ih

drugo računalo usporedi. No, vrijednost n-a u današnje vrijeme je približno 10^{16} što svaki deterministički algoritam čini prilično neefikasnim. Zato razmatramo sljedeći vjerojatnosni algoritam - protokol R_k.

- *Početna situacija:* Računalo R_I ima spremljene bitove $x = x_1x_2 \dots x_n$, a R_{II} ima spremljene $y = y_1y_2 \dots y_n$
- *Prva faza:* R_I na slučajan način izabire k prostih brojeva iz skupa $\{2, 3, \dots, n^2\}$. Označimo ih s $p_1, p_2, \dots, p_k$
- *Druga faza:* Definirajmo funkciju $\text{Number}(x) = \sum_{i=1}^{n} 2^{i-1} \cdot x_i$. Drugim riječima, funkcija $\text{Number}(x)$ pretvara binarni broj x u dekadski. Nakon što izabere k prostih brojeva, računalo R_I računa

  $$s_i = \text{Number}(x) \bmod p_i$$

  za $i = 1, 2, \dots, k$ i šalje njihovu binarnu reprezentaciju kao i binarnu reprezentaciju izabranih prostih brojeva računalu R_{II}.
- *Treća faza:* Nakon što je računalo R_{II} primilo binarnu reprezentaciju od $p_1, p_2, \dots, p_k, s_1, s_2, \dots, s_k$, ono računa

  $$q_i = \text{Number}(y) \bmod p_i$$

  za $i = 1, 2, \dots, k$.
  - Ako postoji i iz skupa $\{1, 2, \dots, k\}$ tako da $q_i \neq s_i$ tada R_{II} vraća $x \neq y$.
  - Inače vraća $x = y$.

Ako s $C_{p_1,\dots,p_k}$ označimo izvršavanje protokola R_k zadano s prostim brojevima p_1, p_2, $\dots$, p_k za ulaz (x, y) s vjerojatnosnim prostorom

$$(S_{R_k,(x,y)}, \text{Prob}_k)$$

gdje je

$$S_{R_k,(x,y)} = \{C_{p_1,\dots,p_k} \,|\, p_1, \dots, p_k \in \text{PRIM}(n^2)^1\}$$

a gdje je Prob_k uniformna distribucija na $S_{R_k(x,y)}$ (neformalno, to bi značilo da svaka uređena k-torka prostih brojeva iz $\text{PRIME}(n^2) \times k$ ima jednaku vjerojatnost biti izabrana). Budući da postoji bijekcija između $S_{R_k,(x,y)}$ i $\text{PRIM}(n^2) \times k$ za modeliranje rada protokola R_k ćemo koristiti vjerojatnosni prostor

$$(\text{PRIM}(n^2) \times k, \text{Prob}_k)$$

[1] $\text{PRIM}(n^2) = \{p \text{ je prost broj} \,|\, p \leq n^2\}$

Sljedeći korak bi trebao biti definiranje indikatorske varijable $X(C_{p_1,\ldots,p_k}) \in \{0,1\}$ pa pogledajmo kada to algoritam vraća $x = y$, a kada $x \neq y$ ovisno o izabranim prostim brojevima.

Jasno je da će vjerojatnost greške kod ulaza (x, y) za koji vrijedi $x = y$ biti jednaka nuli, jer će tada Number(x) i Number(y) biti jednaki te će vrijediti $s_i = q_i$ za svaki $i \in \{1, \ldots, k\}$. Zato ćemo promatrati zanimljiviji slučaj kada je $x \neq y$, a algoritam vraća $x = y$.

Rekli smo da algoritam vraća $x = y$ kada za svaki $i \in \{1, \ldots, k\}$ vrijedi

$$s_i = q_i$$

$$\text{Number}(x) \bmod p_i = \text{Number}(y) \bmod p_i$$

Drugim riječima Number(x) i Number(y) pri djeljenju s brojem p_i daju isti ostatak. Zapišimo to na ovaj način

$$\text{Number}(x) = x' \cdot p_i + r$$

$$\text{Number}(y) = y' \cdot p_i + r$$

gdje je r upravo taj ostatak pri djeljenju brojeva Number(x) i Number(y) s p_i, a x' i y' su pozitivni cijeli brojevi. Pogledajmo razliku brojeva Number(x) i Number(y):

$$\begin{aligned}\text{Number}(x) - \text{Number}(y) &= x' \cdot p_i + r - y' \cdot p_i - r = \\ &= x' \cdot p_i - y' \cdot p_i = \\ &= (x' - y') \cdot p_i\end{aligned}$$

Iz ovoga se može zaključiti da ako $x \neq y$, i ako svaki od izabranih prostih brojeva $p_1, \ldots, p_k$ **dijeli razliku dekadske reprezentacije brojeva x i y**, algoritam će vratiti $x = y$, iako to nije istina. Primijetimo da je to jedini slučaj u kojem algoritam daje netočan rezultat. U tom smislu skup PRIME(n^2) možemo podijeliti na skup dobrih i loših prostih brojeva. Dobri su oni koji ne dijele već spomenutu razliku, a loši oni koji ju dijele. Sada indikacijsku varijablu možemo zadati na sljedeći način

$$X(C_{p_1,\ldots,p_k}) = \begin{cases} 1 \text{ , ako je barem jedan od brojeva } p_1, \ldots, p_k \text{ dobar za } (x,y) \\ 0 \text{ , ako su } p_1, \ldots, p_k \text{ loši za } (x,y) \end{cases}$$

Prije nego odredimo vjerojatnosti uspjeha i greške protokola R_k s ulazom (x, y), dokažimo sljedeću lemu.

Lema 2.1. *Za svaki ulaz (x, y), gdje su x i y prirodni brojevi u binarnom zapisu duljine n; broj loših prostih brojeva iz skupa PRIM(n^2) je najviše $n - 1$.*

Dokaz. Neka je

$$w = |\text{Number}(x) \text{ - Number}(y)| = \left| \sum_{i=1}^{n} 2^{i-1} \cdot x_i - \sum_{i=1}^{n} 2^{i-1} \cdot y_i \right| =$$
$$= \left| \sum_{i=1}^{n} 2^{i-1} \cdot (x_i - y_i) \right| \leq \sum_{i=1}^{n} 2^{i-1} |x_i - y_i| \leq \sum_{i=1}^{n} 2^{i-1} = 2^n - 1 < 2^n$$

Jednakost $\sum_{i=1}^{n} 2^{i-1} = 2^n - 1$ se lako dokaže indukcijom:

Bazni slučaj: $n = 1$

$$2^{1-1} = 1 = 2 - 1 = 2^1 - 1$$

Pretpostavka: Pretpostavimo da za neki $k \in N$ vrijedi: $\sum_{i=1}^{k} 2^{i-1} = 2^k - 1$

Korak: Dokažimo da za $n = k + 1$ vrijedi $\sum_{i=1}^{k+1} 2^{i-1} = 2^{k+1} - 1$:

$$\sum_{i=1}^{k+1} 2^{i-1} = \sum_{i=1}^{k} 2^{i-1} + 2^k \overset{pretp.}{=} 2^k - 1 + 2^k = 2 \cdot 2^k - 1 = 2^{k+1} - 1$$

Po principu matematičke indukcije možemo zaključiti da je $\sum_{i=1}^{n} 2^{i-1} = 2^n - 1$.

Također, w možemo faktorizirati

$$w = p_1^{i_1} \cdot p_2^{i_2} \cdot \ldots \cdot p_k^{i_k}$$

Gdje su $p_1 < p_2 < \ldots < p_k$ prosti brojevi, a $i_1, \ldots, i_k$ pozitivni cijeli brojevi.

Prisjetimo se, prost broj je loš za ulaz (x, y) ako dijeli razliku dekadske reprezentacije brojeva x i y (u našem slučaju, ako dijeli w). Dakle, loši brojevi za (x, y) su prosti faktori od w, tj. $p_1, p_2 \ldots, p_k$. Njih ima točno k. Kada bismo dokazali da je $k \leq n - 1$, imali bismo tvrdnju leme.

Pretpostavimo suprotno, tj. $k \geq n$. Tada

$$w = p_1^{i_1} \cdot p_2^{i_2} \cdot \ldots \cdot p_k^{i_k} =$$
$$\geq p_1 \cdot p_2 \cdot \ldots \cdot p_n$$
$$> 1 \cdot 2 \cdot \ldots \cdot n = n! > 2^n$$

Time smo dobili kontradikciju, jer smo na početku dokaza pokazali da $w < 2^n$. □

Vratimo se na vjerojatnost uspjeha protokola R_k s ulazom (x, y). Koristeći formulu (2.1) dobivamo

$$\begin{aligned} E[X] &= \mathrm{Prob}_k(\mathrm{Event}(X = 1)) = \\ &= 1 - \mathrm{Prob}_k(\mathrm{Event}(X = 0)) = \\ &= 1 - \mathrm{Prob}_k(\mathrm{Event}(p_1, \ldots, p_k \text{ su svi loši})) \end{aligned}$$

Budući da su prosti brojevi p_i, $i \in \{1, \ldots, k\}$ izabrani neovisno jedan o drugom, vrijedi:

$$E[X] = 1 - \mathrm{Prob}(p_1 \text{ je loš}) \cdot \mathrm{Prob}(p_2 \text{ je loš}) \cdot \cdots \cdot \mathrm{Prob}(p_k \text{ je loš})$$

Vjerojatnost da je izabran loš prost broj iz skupa $\mathrm{PRIM}(n^2)$ mora biti manja ili jednaka $\dfrac{n-1}{|\mathrm{PRIM}(n^2)|}$, jer smo dokazali da u skupu $\mathrm{PRIM}(n^2)$ ima najviše $n-1$ loših prostih brojeva. I to vrijedi za svaki naš p_i, $i \in \{1, 2, \ldots, k\}$:

$$E[X] \geq 1 - \left(\frac{n-1}{|\mathrm{PRIM}(n^2)|}\right)^k$$

Znamo da vrijedi $|\mathrm{PRIM}(n^2)| \geq \dfrac{n^2}{\ln n^2}$, pa dobivamo

$$E[X] \geq 1 - \left(\frac{\ln n^2}{n}\right)^k = 1 - \frac{2^k \cdot \ln^k n}{n^k}$$

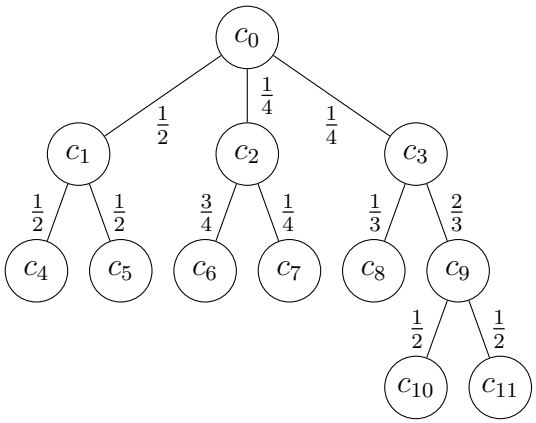
Sada lako možemo izračunati i vjerojatnost greške koristeći (2.2):

$$\mathrm{Error}_{R_k}((x, y)) = 1 - \mathrm{E}[X] \leq \frac{2^k \cdot \ln^k n}{n^k}$$

Da bismo dočarali neznatnost ove vjerojatnosti, uzmimo primjerice $n = 10^{16}$ i $k = 5$. Tada vjerojatnost greške protokola R_5 iznosi najviše $1.024 \cdot 10^{-52}$.

2.2 Drugi model

Budući da je ponekad prirodnije prikazivati vjerojatnosni algoritam kao nedeterminističke algoritam s razdiobom vjerojatnosti za svaki nedeterministički izbor, proučit ćemo i malo složeniji model. Takve algoritme možemo najbolje prikazati pomoću stabla izračunavanja gdje čvorovi stabla označuju konfiguracije algoritma A, a svaki put od korijena do lista odgovara jednom izračunavanju algoritma A s ulazom w. Svaki brid u stablu je označen vrijednošću iz $[0, 1]$, što određuje vjerojatnost izbora upravo



Slika 2.1: Primjer stabla izračunavanja

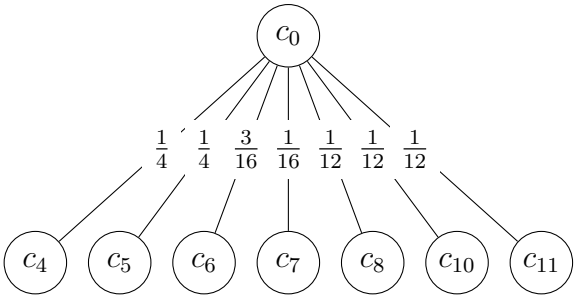
te konfiguracije. Jednostavni primjer takvog stabla izračunavanja možemo vidjeti na Slici 2.1.

Formule navedene u prethodnom potpoglavlju ostaju iste, samo ćemo u ovom slučaju imati drugačiji vjerojatnosni prostor, tj. vjerojatnost pojedinog izračunavanja: $\text{Prob}(\{C_i\})$.

Neka je prostor $S_{A,w}$ skup svih izračunavanja algoritma A s ulazom w, tada on tvori vjerojatnosni prostor $(S_{A,w}, \text{Prob})$. Gdje je $\text{Prob}(\{C\})$, za bilo koje izračunavanje $C \in S_{A,w}$, produkt svih vjerojatnosti na bridovima koji vode od korijena do čvora s oznakom C u odgovarajućem stablu izračunavanja. Primjerice, za stablo izračunavanja na Slici 2.1.: $\text{Prob}(c_{10}) = \frac{1}{4} \cdot \frac{2}{3} \cdot \frac{1}{2} = \frac{1}{12}$.

Indukcijom po dubini stabla izračunavanja se lako dokaže da je funkcija Prob: $S_{A,w} \to [0, 1]$ vjerojatnosna distribucija na $S_{A,w}$.

Jasno je da je drugi model generalizacija prvog. Drugim riječima, bilo kojem algoritmu koji odgovara prvom modelu možemo pridružiti stablo izračunavanja dubine 1. Drugi model također možemo svesti na prvi, ako transformiramo stablo tako



Slika 2.2: Transformacija drugog u prvi model

da korijen ima onoliko djece koliko stablo ima listova. Slika 2.2. prikazuje stablo izračunavanja sa Slike 2.1., ali transformirano tako da odgovara prvom modelu.

Zbog jednostavnosti prvog modela, nastojimo ga koristiti svaki put kada je to moguće. Ako pak imamo algoritam koji više puta donosi slučajne odluke, bolje je koristiti drugi model. U tom slučaju bi svođenje na prvi model bilo suviše kompliciranо.

2.2.1 Vjerojatnosni Quicksort algoritam

Sada ćemo navesti vjerojatnosnu verziju jednog poznatog algoritma, te ćemo na njemu ilustrirati do sada uvedene pojmove. Riječ je o Quicksort algoritmu.

Algoritam kao ulaz prima proizvoljni niz elementa, a vraća isti niz u sortiranom poretku. Prvi korak se sastoji od biranja tzv. pivot elementa. U običnom Quicksort algoritmu pivot se izabire na neki deterministički način, dok je kod vjerojatnosnog Quicksort algoritma pivot element iz niza dobiven slučajnim odabirom. Dani niz rastavljamo na dva dijela. Prvi se sastoji od elemenata koji su manji, a drugi od elemenata koji su veći od pivota. Sve dok je broj elemenata u nizu veći od 1, rekurzivno pozivamo Quicksort na dva dobivena niza.

Slijedi pseudokod vjerojatnosnog Quicksort algoritma (RQS):

- *Ulaz:* Niz elemenata, A
- *Korak 1:*
  - Ako $A = \{b\}$, tj. $|A| = 1$, vrati b
  - Ako $|A| > 1$, slučajnim odabirom izaberi $a \in A$
- *Korak 2:* Rastavi niz A na:
  - $A_< := \{b \in A \,|\, b < a\}$
  - $A_> := \{c \in A \,|\, c > a\}$
- *Korak 3:* Vrati $\mathrm{RQS}(A_<)$, a, $\mathrm{RQS}(A_>)$

Očito, algoritam na kraju vraća sortirani niz elemenata i ne postoji izračunavanje koje završava s pogrešnim rezultatom. Stoga je vjerojatnost greške, za bilo koji ulaz, jednaka 0, tj. $\mathrm{Error}_A(n) = 0$.

Primijetimo da se složenost, koju ćemo mjeriti brojem usporedbi dvaju elemenata iz niza A, jako razlikuje od izračunavanja do izračunavanja.

Propozicija 2.2. *Ako Quicksort algoritam s ulazom duljine n, za pivota uvijek bira najmanji ili najveći element, tada postoji $n-1$ rekurzivnih poziva i broj izvršenih usporedbi je $\dfrac{n \cdot (n-1)}{2}$, tj.*

$$\mathrm{Time}_{\mathrm{RQS}}(n) \in \mathcal{O}(n^2)$$

Dokaz. Pretpostavimo da imamo niz $A = \{1, 2, \ldots n-1, n\}$ i da algoritam uvijek odabire najveći element za pivota. U prvom koraku pivot je n. Quicksort algoritam izvršava $n-1$ usporedbi jer svaki element (osim pivota) mora biti uspoređen s pivotom. Vraća niz $A_<$ duljine $n-1$, n, te $A_>$ duljine 0 - s njim više ništa ne moramo raditi.

U sljedećem koraku algoritam izabire $n-1$ za pivota, provodi $n-2$ usporedbi, te vraća nizove duljina $n-2$ i 0. Postupak se nastavlja sve dok prvi niz ne bude duljine 1.

Dakle, ukupan broj usporedbi je:

$$(n-1) + (n-2) + \ldots + 2 + 1 = \frac{n \cdot (n-1)}{2}$$

A složenost algoritma je

$$\mathrm{Time}_{\mathrm{RQS}}(n) \in \mathcal{O}(n^2)$$

Dokaz je analogan i za algoritam koji uvijek bira najmanji element za pivota. □

Upravo smo prikazali izračunavanje u kojem Quicksort ima najgoru složenost. Za računanje najbolje složenosti trebat će nam sljedeći teorem, čiji dokaz možete naći u [1] - (Theorem 2.2.2.24, str. 37).

Teorem 2.3 (Glavni teorem)**.** *Neka su a, b i c pozitivni cijeli brojevi i neka je T proizvoljna funkcija s domenom $\mathbb{R}^+$ za koju vrijedi: $T(1) = 0$ i $T(n) = a \cdot T\left(\frac{n}{c}\right) + b \cdot n$. Za svaki prirodan broj n tada:*

$$T(n) \in \begin{cases} \mathcal{O}(n), & \text{ako } a < c \\ \mathcal{O}(n \log n), & \text{ako } a = c \\ \mathcal{O}\left(n^{\log_c a}\right), & \text{ako } c < a \end{cases}$$

Propozicija 2.4. *Ako Quicksort algoritam s ulazom duljine n za pivota uvijek bira srednji element po veličini, tada je* $\mathrm{Time}_{\mathrm{RQS}}(n) \in \mathcal{O}(n \cdot \log n)$.

Dokaz. Zadani problem razbijamo u nekoliko manjih istovrsnih problema, tako da rješenje polaznog problema možemo lako konstruirati iz rješenja manjih problema. Riječ je o klasičnoj analizi strategije podijeli pa vladaj.

Ako za pivota izaberemo srednji element po veličini, od jednog niza duljine n dobivamo dva niza duljine $\frac{n}{2}$. Pri podjeli na dva niza smo izvršili $n-1$ usporedbi, tj. usporedili smo pivota sa svakim preostalim elementom iz niza. Na taj način dobivamo sljedeću jednakost:

$$\mathrm{Time}_{\mathrm{RQS}}(n) = 2 \cdot \mathrm{Time}_{\mathrm{RQS}}\left(\frac{n}{2}\right) + n - 1$$

.

Koristeći Glavni teorem možemo zaključiti da je

$$\mathrm{Time}_{\mathrm{RQS}}(n) \in \mathcal{O}(n \cdot \log n)$$

□

Ovo su bile složenosti slučajnog Quicksort algoritma u najboljem i u najgorem slučaju, a u nastavku ćemo proučavati očekivanu složenost.

Vidjeli smo da stablo izračunavanja za RQS nije samo jako veliko, već je i nepravilno. Duljine putova od korijena do listova mogu biti između $\log_2 n$ i $n-1$, te sva izračunavanja imaju različite vjerojatnosti. Ako promotrimo vjerojatnosni prostor $(S_{\mathrm{RQS}(A)}, \mathrm{Prob})$ i definiramo slučajnu varijablu

$$X(C) = \text{broj usporedbi u izračunavanju } C, \quad \text{za svaki } C \in S_{\mathrm{RQS}}(A)$$

tada možemo koristiti X za računanje očekivane vremenske složenosti RQS-a.

Neka je $s_1, s_2, \ldots, s_n$ izlaz RQS(A). Drugim riječima, neka je s_i i-ti najmanji element u A. Definirajmo indikatorsku varijablu X_{ij} za sve $i, j \in \{1, 2, \ldots, n\}$ tako da $i < j$:

$$X_{ij}(C) = \begin{cases} 1, & \text{ako su } s_i \text{ i } s_j \text{ uspoređeni u izračunavanju } C, \\ 0, & \text{ako } s_i \text{ i } s_j \text{ nisu uspoređeni u izračunavanju } C \end{cases}$$

Tada slučajna varijabla T, definirana na sljedeći način, računa ukupni broj usporedbi:

$$T(C) = \sum_{i=1}^{n} \sum_{j>i} X_{ij}(C)$$

Budući da smo rekli da ćemo vremensku složenost mjeriti brojem usporedbi, očekivana vremenska složenost algoritma će biti $\mathrm{E}[T]$. Koristeći definiciju slučajne varijable T i linearnost očekivanja dobivamo:

$$\mathrm{E}[T] = \mathrm{E}\left[\sum_{i=1}^{n} \sum_{j>i} X_{ij}(C)\right] = \sum_{i=1}^{n} \sum_{j>i} \mathrm{E}\left[X_{ij}\right] \tag{2.3}$$

Sada još trebamo izračunati $\mathrm{E}\left[X_{ij}\right]$. U tu svrhu definirajmo p_{ij} kao vjerojatnost da su s_i i s_j uspoređeni tijekom izvršavanja RQS(A). Budući da je X_{ij} indikatorska varijabla, prema Lemi 1.13 vrijedi:

$$\mathrm{E}[X_{ij}] = p_{ij} \cdot 1 + (1 - p_{ij}) \cdot 0 = p_{ij} \tag{2.4}$$

Dakle, zanima nas koja je vjerojatnost da su s_i i s_j uspoređeni tijekom izvršavanja algoritma. Istaknimo sljedeća tri podskupa skupa A: $L = \{s_1, s_2, \ldots, s_{i-1}\}$, $M = \{s_{i+1}, \ldots, s_{j-1}\}$ i $R = \{s_{j+1}, \ldots, s_n\}$.

Brojevi s_i i s_j mogu biti uspoređeni jedino u slučaju da je jedan od njih izabran kao pivot element prije nego su podijeljeni u dva različita podskupa. Drugim riječima, ako je jedan od njih izabran za pivota prije bilo kojeg elementa iz M. Naime, ako je pivot iz M tada s_i ide u $A_<$, a s_j u $A_>$ i ni na koji način više ne mogu biti uspoređeni.

Svaki element iz A ima istu vjerojatnost da bude izabran za pivota. Ako element iz L ili R bude izabran za prvog pivota, to nema nikakvog utjecaja na to hoće li s_i i s_j biti uspoređeni u daljnjem izvršavanju jer tada s_i, s_j i svi elementi između njih odlaze u isti podskup te smo u istoj situaciji kao na početku. Stoga ćemo ovu situaciju modelirati kao vjerojatnosni eksperiment uniformnog biranja elementa na slučajan način iz skupa $M \cup \{s_i, s_j\}$. Dakle,

$$p_{ij} = \frac{|\{s_i, s_j\}|}{|M \cup \{s_i, s_j\}|} = \frac{2}{j - i - 1 + 2} = \frac{2}{j - i + 1} \tag{2.5}$$

Prije nego počnemo računati $\mathrm{E}[T]$ uvest ćemo pojam n-tog harmonijskog broja i dokazati korisno svojstvo harmonijskih brojeva:

Definicija 2.5. n-ti harmonijski broj je suma recipročnih vrijednosti prvih n prirodnih brojeva. Dakle,

$$\mathrm{Har}(n) = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = \sum_{k=1}^{n} \frac{1}{k}$$

◁

Lema 2.6. *Vrijedi:*

$$\mathrm{Har}(n) = \ln n + \Theta(1)$$

Dokaz. Neka je

$$f(n) = \mathrm{Har}(n) - \ln n$$

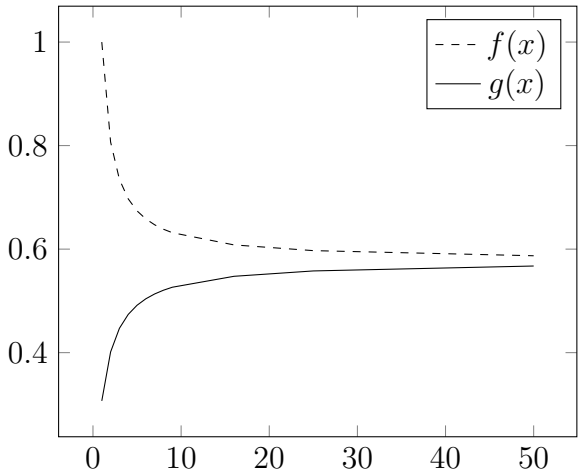
Primijetimo da je $f(1) = 1$ i da funkcija strogo pada. Nadalje, neka je

$$g(n) = \mathrm{Har}(n) - \ln(n + 1)$$

primijetimo da je $g(1) = 1 - \ln 2 \approx 0.307$, te da funkcija strogo raste. Na Slici 2.3 možemo vidjeti njihove grafove.

Kada $n \to \infty$, vrijedi

$$(f(n) - g(n)) \to 0$$



Slika 2.3: Grafovi funkcija f i g

Dakle, funkcije imaju isti limes, označimo ga s $\gamma \approx 0.5772157$[2].

Sada imamo,

$$\mathrm{Har}(n) > \gamma + \ln n \quad \text{i}$$

$$\mathrm{Har}(n) < \gamma + \ln(n+1)$$

što zajedno daje

$$\gamma + \ln n < \mathrm{Har}(n) < \gamma + \ln(n+1)$$

Drugim riječima, vrijedi:

$$\mathrm{Har}(n) = \ln n + \Theta(1)$$

□

Vratimo se računanju očekivane složenosti slučajnog Quicksort algoritma. Uvrštavanjem (2.5) u (2.4) dobivamo:

$$\mathrm{E}[X_{ij}] = \frac{2}{j-i+1}$$

A zatim uvrštavanjem gornje jednakosti u (2.3) dobivamo:

[2]Ova konstanta je poznata kao Euler-Mascheronijeva vrijednost i jednaka je $\gamma = \lim_{n\to\infty}\left(-\ln n + \sum_{k=1}^{n}\frac{1}{k}\right)$

$$\begin{aligned}
\mathrm{E}[T] &= \sum_{i=1}^{n} \sum_{j>i} \frac{2}{j-i+1} = \\
&\leq \sum_{i=1}^{n} \sum_{k=1}^{n-i+1} \frac{2}{k} \leq 2 \cdot \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{1}{k} = \\
&= 2 \cdot \sum_{i=1}^{n} \mathrm{Har}(n) = 2 \cdot n \cdot \mathrm{Har}(n) = \\
&= 2 \cdot n \cdot (\ln n + \Theta(1)) = \\
&= 2 \cdot n \cdot \ln n + \Theta(n)
\end{aligned}$$

Dakle, $\text{Exp-Time}_{\text{RQS}}(n) \in \mathcal{O}(n \log n)$ što je najbolja složenost za algoritme sortiranja u kojima na početku ništa ne znamo o elementima, osim da se mogu uspoređivati. Time smo pokazali da je slučajni Quicksort algoritam u praksi veoma koristan i efikasan algoritam.

Poglavlje 3

Podjela vjerojatnosnih algoritama

Cilj ovog poglavlja je opisati glavnu i općeprihvaćenu podjelu vjerojatnosnih algoritama na Las Vegas algoritme, algoritme s jednostranom greškom, algoritme s ograničenom greškom i algoritme s neograničenom greškom. Ova klasifikacija se temelji na vrsti i vjerojatnosti greške algoritama. Navest ćemo i analizirati neke algoritme iz svake skupine.

3.1 Las Vegas algoritmi

Las Vegas algoritmi su vrsta vjerojatnosnih algoritama koji uvijek vraćaju točan rezultat. Postoje dva modela ovih algoritama. Razlika je u tome što jedan model dopušta odgovor „?" što bi značilo da algoritam ne zna odrediti rješenje, dok drugi ne dopušta takav izlaz. Počnimo s modelom koji ne dopušta „?".

Definicija 3.1. Za vjerojatnosni algoritam A, kažemo da je **Las Vegas algoritam** koji računa funkciju F, ako za svaki ulaz x (x je argument od F) vrijedi

$$\text{Prob}(A(x)) = F(x)) = 1$$

$\triangleleft$

Slijedi definicija drugog modela za Las Vegas algoritme.

Definicija 3.2. Neka je A vjerojatnosni algoritam koji dopušta odgovor „?". Kažemo da je A **Las Vegas algoritam** koji računa funkciju F, ako za svaki ulaz x vrijedi

- $\text{Prob}(A(x)) = F(x)) \geq \frac{1}{2}$, i
- $\text{Prob}(A(x)) = \text{"?"}) = 1 - \text{Prob}(A(x) = F(x)) \leq \frac{1}{2}$

$\triangleleft$

Vidimo da definicije isključuju pojavu pogrešnog rješenja. U definiciji 3.2. izlaz je točno rješenje ili „?". U prvom uvjetu zahtijevamo da algoritam računa točan rezultat s vjerojatnošću većom od $\frac{1}{2}$, ali $\frac{1}{2}$ možemo zamijeniti bilo kojom konstantom iz $\langle 0, 1\rangle$. Razlog za to je što za bilo koji $\epsilon \leq \frac{1}{2}$ možemo povećati uspješnost algoritma na više od $\frac{1}{2}$, nezavisno ga pokrećući n puta s istim ulazom x, gdje je n konstantan prirodan broj.

Postavlja se pitanje može li se Las Vegas algoritam iz definicije 3.1. transformirati u algoritam iz definicije 3.2. i obratno. Odgovor je potvrdan i prvo ćemo pokazati kako transformirati algoritam koji dopušta odgovor „?" u algoritam čiji su izlazi točni rezultati.

Propozicija 3.3. *Neka je A Las Vegas algoritam koji dopušta odgovor „?" s ograničenom vjerojatnošću. Moguće je transformirati algoritam A u Las Vegas algoritam A' koji ne dopušta odgovor „?".*

Dokaz. Neka je A algoritam koji dopušta odgovor „?". Transformirat ćemo algoritam A u algoritam A' koji se ponovno pokreće s istim ulazom u slučajevima gdje bi A odgovorio s „?".

Ovaj princip možemo opisati i pomoću stabla izračunavanja. Neka je $T_{A,w}$ stablo izračunavanja algoritma A za ulaz w. Tada stablo izračunavanja za A', $T_{A',w}$ dobivamo „lijepljenjem" stabla $T_{A,w}$ na sve listove stabla $T_{A,w}$ gdje je izlaz „?". Postupak ponavljamo sve dok ima listova s izlazom „?". □

Mana novonastalog algoritma A' je što može sadržavati beskonačna izračunavanja. Naime, postupak se ponavlja sve dok postoje listovi u stablu izračunavanja gdje je izlaz „?", a vjerojatnost izlaza „?" je ista u svakom pokretanju. Drugim riječima, ne znamo hoće li algoritam stati. S druge strane, ako stane, sigurni smo da je rezultat točan.

Zanima nas koliko se isplati raditi ovu transformaciju, tj. kolika je vjerojatnost da algoritam A' uopće stane i kolika je očekivana vremenska složenost algoritma A' u odnosu na algoritam A.

Neka je $\text{Time}_A(w)$ vremenska složenost algoritma A u najgorem slučaju. Ako zamislimo stablo izračunavanja $T_{A,w}$, $\text{Time}_A(w)$ bi bila njegova dubina. Stoga, vjerojatnost da A' stane u $\text{Time}_A(w)$ je najmanje $\frac{1}{2}$ (sjetimo se definicije 3.2.). Tom logikom, vjerojatnost da A' stane u $2 \cdot \text{Time}_A(w)$ je $\frac{3}{4}$, budući da ponovno pokreće algoritam A na svakom listu s izlazom „?". To znači da nakon $k \cdot \text{Time}_A(w)$ algoritam A' stane s vjerojatnošću najmanje $1 - \frac{1}{2^k}$, jer je $\frac{1}{2^k}$ gornja ograda za vjerojatnost izlaza „?" u k nezavisnih pokretanja algoritma A. Primijetimo da vjerojatnost da A' stane teži k 1 s povećanjem vremena što je prilično dobar rezultat.

Sada ćemo analizirati očekivanu vremensku složenost algoritma A'.

Bez smanjenja općenitosti možemo pretpostaviti da sva izračunavanja algoritma A s izlazom „?" imaju maksimalnu duljinu, tj. $\text{Time}_A(w)$. Definiramo skup Set_i za

svaki $i \in \mathbb{N} \setminus \{0\}$ na sljedeći način:

$$\text{Set}_i = \{C \in S_{A',w} \mid (i-1) \cdot \text{Time}_A(w) < \text{Time}(C) < i \cdot \text{Time}_A(w)\}$$

Skup Set_i je skup svih izračunavanja algoritma A' s ulazom w koja stanu u točno i-tom pokretanju algoritma A. Očito vrijede sljedeće tvrdnje:

$$S_{A',w} = \bigcup_{i=1}^{\infty} \text{Set}_i \tag{3.1}$$

$$\text{Set}_r \cap \text{Set}_s = \emptyset, \quad \text{za } r \neq s$$

Već smo ustanovili da je vjerojatnost da izračunavanje stane u prvih k pokretanja algoritma A najmanje $1 - \frac{1}{2^k}$:

$$\sum_{C \in \cup_{i=1}^{k} \text{Set}_i} \text{Prob}(\{C\}) \geq 1 - \frac{1}{2^k}, \quad \text{za } k \in \mathbb{N} \setminus \{0\}$$

Direktna posljedica te tvrdnje je da je vjerojatnost da izračunavanje stane u točno i-tom pokretanju algoritma A najviše $\frac{1}{2^{i-1}}$:

$$\sum_{C \in \text{Set}_i} \text{Prob}(\{C\}) \leq \frac{1}{2^{i-1}}, \quad \text{za } i \in \mathbb{N} \setminus \{0\} \tag{3.2}$$

Koristeći definiciju očekivane vremenske složenosti dobivamo:

$$\begin{aligned} \text{Exp-Time}_{A'}(n) &= \sum_{C \in S_{A',w}} \text{Time}_{A'}(C) \cdot \text{Prob}(\{C\}) \\ &\overset{(3.1)}{=} \sum_{i=1}^{\infty} \sum_{C \in \text{Set}_i} \text{Time}_{A'}(C) \cdot \text{Prob}(\{C\}) \end{aligned}$$

Primijetimo da vrijedi $\text{Time}_{A'}(C) \leq i \cdot \text{Time}_A(w)$ za svaki $C \in \text{Set}_i$ pa stoga možemo pisati:

$$\begin{aligned}
\text{Exp-Time}_{A'}(n) &\leq \sum_{i=1}^{\infty} \sum_{C \in \text{Set}_i} i \cdot \text{Time}_A(w) \cdot \text{Prob}(\{C\}) \\
&= \sum_{i=1}^{\infty} i \cdot \text{Time}_A(w) \cdot \sum_{C \in \text{Set}_i} \text{Prob}(\{C\}) \\
&\overset{(3.2)}{\leq} \sum_{i=1}^{\infty} i \cdot \text{Time}_A(w) \cdot \frac{1}{2^{i-1}} \\
&= \text{Time}_A(w) \cdot \sum_{i=1}^{\infty} \frac{i}{2^{i-1}} \\
&= \text{Time}_A(w) \cdot 2 \cdot \sum_{i=1}^{\infty} \frac{i}{2^i} \\
&= 4 \cdot \text{Time}_A(w)
\end{aligned}$$

A to upravo znači da je $\text{Exp-Time}_{A'}(n) \in \mathcal{O}(\text{Time}_A(n))$

Rekli smo da se dva modela Las Vegas algoritama mogu transformirati jedan u drugi. Pokazali smo kako transformirati algoritam koji dopušta odgovor „?" u algoritam koji ne dopušta. Sada ćemo pokazati kako transformirati algoritam koji ne dopušta odgovor „?" u onaj koji dopušta. No prvo odgovorimo na pitanje zašto bi takva transformacija bila korisna. U svrhu toga razmotrimo sljedeći primjer.

Pretpostavimo da imamo stablo izračunavanja s mnogo kratkih izračunavanja, ali postoji manji broj relativno dugačkih izračunavanja, možda čak i beskonačnih. U tom slučaju kada izračunavanje traje predugo, možemo pretpostaviti da se izvodi jedno od dugačkih izračunavanja i odlučiti prekinuti algoritam te vratiti „?" kao odgovor. Glavno pitanje je koja je gornja granica za vremensku složenost algoritma prije nego ga odlučimo prekinuti, a da vjerojatnost izlaza „?" bude manja od $\frac{1}{2}$ u skladu s definicijom 3.2. Odgovor nam daje sljedeći teorem:

Teorem 3.4. *Neka je A algoritam u skladu s definicijom 3.1. Ako ga želimo transformirati u algoritam B prema definiciji 3.2., tada je gornja granica za zaustavljanje algoritma A:* $2 \cdot \text{Exp-Time}_A(w)$.

Dokaz. Pretpostavimo suprotno, tj. da algoritam B dobiven na gore opisani način ne zadovoljava definiciju 3.2. Tada vrijedi:

$$\text{Prob}(B(w) = \text{"?"}) > \frac{1}{2} \tag{3.3}$$

Definirajmo skup:

$$\text{Set}_{A,w}(\text{"?"}) = \{C \in S_{A,w} \mid \text{Time}(C) > 2 \cdot \text{Time-Exp}_A(w)\} \subset S_{A,w}$$

To je skup svih izračunavanja algoritma A čija je vremenska složenost veća od očekivane vremenske složenosti algoritma A. Drugim riječima, to su izračunavanja koja u algoritmu B vraćaju „?”.

Sada izraz (3.3) možemo zapisati kao:

$$\sum_{C \in S_{A,w}(\text{"?"})} \text{Prob}(\{C\}) > \frac{1}{2} \tag{3.4}$$

Definirajmo i skup svih izračunavanja algoritma A čija je vremenska složenost manja ili jednaka očekivanoj vremenskoj složenosti algoritma A. Drugim riječima, definiramo skup svih izračunavanja iz A koja bi u algoritmu B vraćala točan rezultat:

$$S_{A,w}(F(w)) = S_{A,w} - S_{A,w}(\text{"?"}) \tag{3.5}$$

Prema definiciji skupa $S_{A,w}(?)$, vrijedi sljedeće:

$$\text{Time}(C) \geq 2 \cdot \text{Exp-Time}_A(w) + 1, \quad \text{za svaki } C \in S_{A,w}(\text{"?"}) \tag{3.6}$$

Dakle,

$$\begin{aligned}
\text{Exp-Time}_A(w) &\overset{\text{def.}}{=} \sum_{C \in S_{A,w}} \text{Time}_A(C) \cdot \text{Prob}(\{C\}) \\
&\overset{(3.5)}{=} \sum_{C \in S_{A,w}(\text{"?"})} \text{Time}_A(C) \cdot \text{Prob}(\{C\}) + \sum_{C \in S_{A,w}(F(w))} \text{Time}_A(C) \cdot \text{Prob}(\{C\}) \\
&\overset{(3.6)}{>} \sum_{C \in S_{A,w}(\text{"?"})} (2 \cdot \text{Exp-Time}_A(w) + 1) \cdot \text{Prob}(\{C\}) + 0 \\
&= (2 \cdot \text{Exp-Time}_A(w) + 1) \cdot \sum_{C \in S_{A,w}(\text{"?"})} \text{Prob}(\{C\}) \\
&\overset{(3.4)}{>} (2 \cdot \text{Exp-Time}_A(w) + 1) \cdot \frac{1}{2} \\
&= \text{Exp-Time}_A(w) + \frac{1}{2}
\end{aligned}$$

Dobili smo očitu kontradikciju. Dakle, negacija (3.3) je istinita.

$$\text{Prob}(B(w) = \text{"?"}) \leq \frac{1}{2}$$

Time smo pokazali da je algoritam B Las Vegas algoritam u skladu s definicijom 3.2. □

3.1.1 Random - select algoritam (RSEL)

Jasno je da je vjerojatnosni Quicksort algoritam iz Poglavlja 2.2.1 jedan primjer Las Vegas algoritama budući da uvijek vraća točan rezultat. Sada ćemo pogledati još

jedan primjer Las Vegas algoritma koji ne dopušta odgovor „?" te uvijek daje točan rezultat. Riječ je o Random - select algoritmu (RSEL).

Algoritam RSEL(A, k) vraća k-ti po veličini element iz skupa A. Slijedi njegov pseudokod:

- *Ulaz:* $A = \{a_1, a_2, \ldots a_n\}$ i $k \in \{1, \ldots, n\}$, gdje $n \in \mathbb{N}\backslash\{0\}$.
- *Prvi korak:*
  - ako $n = 1$, vrati a_1
  - inače slučajnim odabirom izaberi $i \in \{1, 2, \ldots n\}$
- *Drugi korak:* Rastavi skup A na dva podskupa:
  - $A_< = \{b \in A \,|\, b < a_i\}$
  - $A_> = \{c \in A \,|\, c > a_i\}$
- *Treći korak:*
  - ako $|A_<| > k$ pozovi RSEL$(A_<, k)$
  - ako $|A_<| = k - 1$, vrati a_i
  - inače pozovi RSEL$(A_>, k - |A_<| - 1)$

Jasno je da algoritam RSEL vraća točan rezultat u svakom pokretanju. Pogledajmo njegovu složenost u najboljem i najgorem slučaju, te njegovu očekivanu složenost.

Isto kao kod vjerojatnosnog Quicksort algoritma, složenost u najgorem slučaju dobivamo ako za pivot element uvijek biramo najveći ili najmanji element. Na sličan način kao u Propoziciji 2.2, pokaže se da je tada složenost jednaka $\mathcal{O}(n^2)$. Budući da algoritam RSEL bira pivote na slučajan način, izračunajmo koja je vjerojatnost da izračunavanje bude najgore složenosti.

Propozicija 3.5. *Neka je n broj elemenata u skupu A. Tada, vjerojatnost da RSEL(A, k) ima složenost $\mathcal{O}(n^2)$ je jednaka $\frac{2^{n-1}}{n!}$.*

Dokaz. Neka je ϵ_k događaj izabiranja najvećeg ili najmanjeg elementa iz skupa u trenutku kada je u skupu k elemenata. Primijetimo da je $\mathrm{P}(\epsilon_k) = \frac{2}{k}$ za $k > 1$, te je jednaka 1, ako je $k = 1$. Također primijetimo da su ti događaji međusobno nezavisni. Neka je ϵ izračunavanje s najgorom složenošću.

Već smo spomenuli da će algoritam imati najgoru složenost ako je svaki put za pivota izabran najveći ili najmanji element. Dakle, događaj ϵ možemo shvatiti kao presjek svih događaja ϵ_i, gdje je ϵ_i događaj izabiranja najvećeg ili najmanjeg elementa iz skupa u trenutku kada je u skupu i elemenata.

$$\epsilon = \bigcap_{i=1}^{n} \epsilon_i$$

Nas zanima koliko iznosi vjerojatnost $\mathrm{P}(\epsilon)$.

$$\mathrm{P}(\epsilon) = P\left(\bigcap_{i=1}^{n} \epsilon_i\right) \overset{nez.}{=} \prod_{i=1}^{n} \mathrm{P}(\epsilon_i) =$$
$$= \mathrm{P}(\epsilon_1) \cdot \prod_{i=2}^{n} \mathrm{P}(\epsilon_i) = \prod_{i=2}^{n} \frac{2}{i} = \frac{2^{n-1}}{n!}$$

□

S druge strane, kada za pivota uvijek izabiremo srednji element po veličini, skup A rastavljamo na dva skupa, oba s $\frac{n}{2}$ elemenata. Pri tome smo napravili $n-1$ usporedbi svakog elementa s pivotom. Koristeći strategiju podijeli pa vladaj dobivamo sljedeću rekurziju:

$$T(n) = T\left(\frac{n}{2}\right) + n - 1$$

Koristeći Glavni teorem, tj. Teorem 2.3, zaključujemo da je najbolja složenost algoritma RSEL jednaka $\mathcal{O}(n)$, budući da u ovom slučaju imamo situaciju gdje je $1 = a < c = 2$, gdje su a i c oznake iz Glavnog teorema.

Propozicija 3.6. *Očekivana vremenska složenost Random - select algoritma je* $\mathcal{O}(n)$.

Dokaz. Neka je $T_{A,k}$ slučajna varijabla koja broji usporedbe tijekom izvršavanja Random - select algoritma s ulazom (A, k). Neka je $|A| = n$. Želimo naći gornju ogradu za očekivani broj usporedbi $\mathrm{E}[T_{A,k}]$. Budući da $\mathrm{E}[T_{A,k}]$ ovisi o broju elemenata u skupu A, koristit ćemo notaciju $T_{n,k}$ umjesto $T_{A,k}$.

Neka je

$$T_n = \max\{T_{n,k} \mid 1 \leq k \leq n\}$$

U prvom ponavljanju prvog, drugog i trećeg koraka dogodi se $n-1$ usporedbi, budući da pivot element uspoređujemo sa svim ostalim elementima iz skupa A. Pivot je izabran na slučajan način, pa je vjerojatnost da je izabrani pivot, a_i, j-ti najmanji element jednaka $\frac{1}{n}$, za svaki $j \in \{1, \ldots, n\}$. Skup A rastavljamo na $|A_<|$ i $|A_>|$. U slučaju kada je $|A_<| > k$, nastavljamo raditi s tim skupom, te u njemu tražimo k-ti najmanji element. Slučajna varijabla koja broji te usporedbe je $T_{|A_<|,k}$, a inače nastavljamo raditi sa skupom $|A_>|$ te u njemu tražimo $k-|A_<|-1$-ti po redu najmanji element. Primijetimo da je to ukupno k-ti najmanji. Pripadajuća slučajna varijabla je tada $T_{|A_>|,k-|A_<|-1}$. Stoga imamo:

$$T_{n,k} \leq n - 1 + \max\{T_{|A_<|,k}, T_{|A_>|,k-|A_<|-1}\}$$

Pretpostavimo da smo za pivota izabrali j-ti najmanji element, u tom slučaju je $|A_<| = j-1$, a $|A_>| = n-j$:

$$T_{n,k} \leq n - 1 + \max\{T_{j-1,k}, T_{n-j,k-j}\}$$

Koristeći definiciju slučajne varijable T_n imamo:

$$T_{n,k} \leq n - 1 + \max\{T_{j-1}, T_{n-j}\}$$

Dobivamo rekurziju:

$$\mathrm{E}[T_n] \leq n - 1 + \frac{1}{n}\sum_{j=1}^{n-1} \max\{\mathrm{E}[T_{j-1}], \mathrm{E}[T_{n-j}]\}$$

Očekivani broj usporedbi je veći, što je broj elemenata u skupu veći:

$$\mathrm{E}[T_n] \leq n - 1 + \frac{1}{n}\sum_{j=1}^{n-1} \mathrm{E}[T_{\max\{j-1,n-j\}}]$$

Primijetimo da $\max\{j-1,\, n-j\} \in \{\lceil \frac{n}{2} \rceil, \ldots, n\}$, pa zato vrijedi:

$$\mathrm{E}[T_n] \leq n - 1 + \frac{2}{n}\sum_{l=\lceil \frac{n}{2} \rceil}^{n-1} \mathrm{E}[T_l]$$

Matematičkom indukcijom dokažimo da je $\mathrm{E}[T_n] \leq 5n$.

- *Bazni slučaj:* $n = 1$

$$\mathrm{E}[T_1] = 1 - 1 = 0 \leq 5$$

- *Pretpostavka:* Pretpostavimo da tvrdnja vrijedi za sve prirodne brojeve $m < n$.
- *Korak:* Koristeći pretpostavku, dokažimo da tvrdnja vrijedi za n.

$$\begin{aligned}\mathrm{E}[T_n] &\overset{\text{def.}}{\leq} n - 1 + \frac{2}{n}\sum_{l=\lceil \frac{n}{2} \rceil}^{n-1} \mathrm{E}[T_l] \\ &\overset{\text{pretp.}}{\leq} n - 1 + \frac{2}{n}\sum_{l=\lceil \frac{n}{2} \rceil}^{n-1} 5 \cdot l\end{aligned}$$

Koristimo činjenicu: $\sum_{l=\lceil \frac{n}{2} \rceil}^{n-1} l = \sum_{l=1}^{n-1} l - \sum_{l=1}^{\lceil \frac{n}{2} \rceil - 1} l$

$$\mathrm{E}[T_n] \leq n - 1 + \frac{10}{n}\left(\sum_{l=1}^{n-1} l - \sum_{l=1}^{\lceil \frac{n}{2} \rceil - 1} l\right)$$

Koristimo formulu za zbroj prvih n prirodnih brojeva: $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

$$\begin{aligned}\mathrm{E}[T_n] &\leq n - 1 + \frac{10}{n}\left(\frac{n(n-1)}{2} - \frac{\left(\lceil \frac{n}{2} \rceil - 1\right) \cdot \lceil \frac{n}{2} \rceil}{2}\right) \\ &= n - 1 + 5(n-1) - 5\left(\left\lceil \frac{n}{2} \right\rceil - 1\right) \cdot \frac{\lceil \frac{n}{2} \rceil}{n}\end{aligned}$$

Budući je $\lceil \frac{n}{2} \rceil \geq \frac{n}{2}$, vrijedi i $\frac{\lceil \frac{n}{2} \rceil}{2} \geq \frac{\frac{n}{2}}{2} = \frac{1}{2}$:

$$\begin{aligned}\mathrm{E}[T_n] &\leq n - 1 + 5(n-1) - 5\left(\left\lceil \frac{n}{2} \right\rceil - 1\right) \cdot \frac{1}{2} \\ &\leq 5 \cdot n \in \mathcal{O}(n)\end{aligned}$$

Time smo dokazali da je očekivana složenost random - select algoritma jednaka $\mathcal{O}(n)$. □

3.1.2 Protokol $\mathbf{LV}_{10}$

Sada ćemo navesti primjer Las Vegas algoritma koji dopušta odgovor „?" te ćemo dokazati da je u skladu s definicijom 3.2. Koristit ćemo pretpostavke, funkcije i skupove koje smo već definirali kod Protokola R_k u poglavlju 2.1.1.

- *Početna situacija:* Pretpostavimo da imamo dva računala R_I i R_{II}. R_I ima spremljeno 10 strigova x_1, $x_2, \ldots$ x_{10}, a računalo R_{II} ima spremljeno y_1, $y_2, \ldots$ y_{10}. Pretpostavljamo da x_i, $y_i \in \{0,1\}^n$ za svaki $i \in \{1, \ldots 10\}$. Protokol bi trebao odgovarati na pitanje postoji li barem jedan $i \in \{1, \ldots, 10\}$ takav da je $x_i = y_i$.
- *Prva faza:* R_I na slučajan način izabire 10 prostih brojeva iz skupa $\mathrm{PRIM}(n^2)$.
- *Druga faza:* R_I računa

  $$s_i = \mathrm{Number}(x_i) \mod p_i, \quad \text{za } i \in \{1, \ldots 10\}$$

  te šalje računalu R_{II} slučajno izabrane proste brojeve te iz njih izračunate brojeve: p_1, p_2, $\ldots$ p_{10}, s_1, s_2, $\ldots$ s_{10}
- *Treća faza:* Računalo R_{II} računa

  $$q_i = \mathrm{Number}(y_i) \mod p_i, \quad \text{za } i \in \{1, \ldots, 10\}$$

  te uspoređuje s_i i q_i za svaki $i \in \{1, \ldots, 10\}$.
  - Ako $s_i \neq q_i$ za svaki i, tada R_{II} sa sigurnošću zna da $x_i \neq y_i$ za svaki i te vraća odgovor „0".

  - U suprotnom, postoji i za koji vrijedi $s_i = q_i$. Neka je j najmanji indeks za koji vrijedi $s_j = q_j$. Tada R_{II} šalje cijeli string y_j i indeks j računalu R_I.

- *Četvrta faza:* Računalo R_I uspoređuje x_j i y_j bit po bit.
  - Ako $x_j = y_j$, R_I vraća odgovor „1".
  - Inače vraća odgovor „?".

Proučit ćemo dva slučaja: kada ne postoji i takav da $x_i = y_i$ i kada takav i postoji. Dokazat ćemo da je vjerojatnost odgovora „0" ili „1" (ovisno o slučaju) veća od $\frac{1}{2}$ te da algoritam ni u kojem slučaju ne vraća pogrešan odgovor.

Neka je $((x_1, x_2, \ldots, x_{10}), (y_1, y_2, \ldots, y_{10}))$ ulaz tako da $x_i \neq y_i$ za svaki $i \in \{1, \ldots, 10\}$. Algoritam vraća „0" ako za svaki i vrijedi

$$\text{Number}(x_i) \mod p_i \neq \text{Number}(y_i) \mod p_i \tag{3.7}$$

Iz poglavlja 2.1.1 znamo da je vjerojatnost da slučajno izabran broj $p_i \in \text{PRIM}(n^2)$ zadovoljava (3.7) najmanje $1 - \frac{2 \ln n}{n}$.

Budući da su p_i izabrani neovisno, vjerojatnost da $\text{Number}(x_i) \mod p_i \neq \text{Number}(y_i) \mod p_i$ za svaki $i \in \{1, \ldots, 10\}$ je najmanje

$$\left(1 - \frac{2 \ln n}{n}\right)^{10}$$

Stoga vrijedi:

$$\text{Prob}\left(\text{LV}_{10}((x_1, \ldots, x_{10}), (y_1, \ldots, y_{10})) = 0\right) \geq \left(1 - \frac{2 \ln n}{n}\right)^{10}$$

Koristeći binomni teorem: $(x + y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$ imamo:

$$\text{Prob}\left(\text{LV}_{10}((x_1, \ldots, x_{10}), (y_1, \ldots, y_{10})) = 0\right) \geq \sum_{i=0}^{10} (-1)^i \cdot \binom{10}{i} \cdot \left(\frac{2 \ln n}{n}\right)^i$$

Sumu rastavimo na slučaj kada je $i = 0$, te na parne i neparne indekse i:

$$\begin{aligned}&\text{Prob}\left(\text{LV}_{10}((x_1, \ldots, x_{10}), (y_1, \ldots, y_{10})) = 0\right) \\ &\quad \geq 1 + \sum_{l=1}^{5} \binom{10}{2l} \cdot \left(\frac{2 \ln n}{n}\right)^{2l} - \sum_{m=1}^{5} \binom{10}{2m-1} \cdot \left(\frac{2 \ln n}{n}\right)^{2m-1}\end{aligned}$$

Supstitucijama $i = l$ i $i = m - 1$ dobivamo:

$$\begin{aligned}&\text{Prob}\left(\text{LV}_{10}((x_1, \ldots, x_{10}), (y_1, \ldots, y_{10})) = 0\right)\\ &\quad \geq 1 + \sum_{i=1}^{5} \binom{10}{2i} \cdot \left(\frac{2\ln n}{n}\right)^{2i} - \sum_{i=0}^{4} \binom{10}{2i+1} \cdot \left(\frac{2\ln n}{n}\right)^{2i+1}\end{aligned}$$

Matematičkom indukcijom se lako pokaže da za $i \in \{1, \ldots, 4\}$ i $n \geq 15$, vrijedi:

$$\binom{10}{2i} \cdot \left(\frac{2\ln n}{n}\right)^{2i} \geq \binom{10}{2i+1} \cdot \left(\frac{2\ln n}{n}\right)^{2i+1}$$

To znači da je razlika članova u sumama za $i \in \{1, \ldots, 4\}$ pozitivna. Stoga, ako izbacimo te članove i dalje vrijedi:

$$\begin{aligned}&\text{Prob}\left(\text{LV}_{10}((x_1, \ldots, x_{10}), (y_1, \ldots, y_{10})) = 0\right)\\ &\quad \geq 1 - \binom{10}{1} \cdot \frac{2\ln n}{n} + \binom{10}{10} \cdot \left(\frac{2\ln n}{n}\right)^{10}\\ &\quad \geq 1 - \frac{20\ln n}{n} \geq \frac{1}{2}\end{aligned}$$

Zadnja nejednakost vrijedi za svaki $n \geq 215$. Time smo pokazali da ako ne postoji $i \in \{1, \ldots, 10\}$ takav da $x_i = y_i$, da je vjerojatnost odgovora „0" veća od vjerojatnosti odgovora „?", uz pretpostavku da je n dovoljno velik.

S druge strane, ako ne postoji i takav da $x_i = y_i$, ali postoji p_j takav da

$$\text{Number}(x_j) \mod p_j = \text{Number}(y_j) \mod p_j$$

tada algoritam vraća „?". Drugim riječima, u slučaju kada bi deterministički algoritam vratio „0", naš vjerojatnosni algoritam vraća „0" ili „?". Time smo dokazali da je protokol LV_{10} u tom slučaju Las Vegas algoritam prema definiciji 3.2.

Još nam preostaje slučaj kada postoji $i \in \{1, \ldots, 10\}$ takav da $x_i = y_i$.

Neka je $((x_1, \ldots, x_{10}), (y_1, \ldots, y_{10}))$ ulaz koji treba biti prihvaćen. Neka je j najmanji indeks za koji vrijedi $x_j = y_j$. Tada očito vrijedi i

$$\text{Number}(x_j) \mod p_j = \text{Number}(y_j) \mod p_j$$

Algoritam vraća „1" ako za svaki indeks $i < j$ vrijedi $\text{Number}(x_i) \mod p_i \neq \text{Number}(y_i) \mod p_i$. Ako je $j = 1$, algoritam vraća „1" jer prema pretpostavci $x_i = y_i$. No, kada je $j > 1$, vjerojatnost da za svaki indeks i, manji od j, vrijedi $\text{Number}(x_i) \mod p_i \neq \text{Number}(y_i) \mod p_i$ je najmanje

$$\left(1 - \frac{2\ln n}{n}\right)^{j-1}$$

U prvom slučaju smo dokazali da vrijedi nejednakost

$$\left(1 - \frac{2\ln n}{n}\right)^{j-1} \geq 1 - \frac{2(j-1)\cdot \ln n}{n}, \quad \text{za dovoljno velik } n$$

Taj izraz ima minimum u $j = 10$, pa vrijedi

$$\text{Prob}(\text{LV}_{10}((x_1, \ldots, x_{10}), (y_1, \ldots, y_{10})) = 1) \geq 1 - \frac{18\ln n}{n}$$

što je veće od $\frac{1}{2}$ za $n \geq 189$.

U slučaju kada postoji $j < i$ takav da $\text{Number}(x_j) \mod p_j = \text{Number}(y_j) \mod p_j$, tada algoritam vraća „?".

Time smo pokazali da, pod pretpostavkom da postoji $i \in \{1, \ldots, 10\}$ takav da $x_i = y_i$, protokol LV_{10} s većom vjerojatnošću vraća odgovor „1", nego „?" za dovoljno velik n. Te da u tom slučaju ne može vratiti „0". Dakle, protokol LV_{10} je Las Vegas algoritam prema definiciji 3.2.

3.2 Algoritmi s jednostranom greškom

Algoritme s jednostranom greškom koristimo za probleme odluke. Problem odluke je dan parom (Σ, L), gdje algoritam za danu riječ $x \in \Sigma^*$ mora odlučiti pripada li x jeziku $L \subset \Sigma^*$ ili ne. Greške su dopuštene samo za ulaze koji se nalaze u L. Drugim riječima, algoritam za svaki $x \in \Sigma^* \setminus L$ uvijek odgovara s „0", tj. $x \notin L$. Dakle, algoritam smije pogriješiti s ograničenom vjerojatnošću jedino na ulazima koji su u skupu L. U tom smislu, svaki algoritam s jednostranom greškom je i Las Vegas algoritam. Koncept algoritama s jednostranom greškom može se formalno izreći sljedećom definicijom.

Definicija 3.7. Neka je A vjerojatnosni algoritam i neka je (Σ, L) problem odluke. Kažemo da je A algoritam s jednostranom greškom za L ako

- za svaki $x \in L$, $\text{Prob}(A(x) = 1) \geq \frac{1}{2}$
- za svaki $x \notin L$, $\text{Prob}(A(x) = 0) = 1$

$\triangleleft$

Primijetimo da algoritmi s jednostranom greškom mogu pogriješiti samo u jednom slučaju. Specifično, mogu zaključiti da $x \notin L$ u slučaju kada $x \in L$. To je motivacija za naziv ove skupine algoritama.

Algoritmi s jednostranom greškom su vrlo korisni i često se koriste u praksi upravo zbog svojstva iskazanog u sljedećem teoremu.

Teorem 3.8. *Vjerojatnost greške algoritma s jednostranom greškom eksponencijalno se smanjuje s brojem ponavljanja.*

Dokaz. Neka je A algoritam s jednostranom greškom, te neka je (Σ, L) problem odluke koji razmatramo.

Neka su $a_1, \ldots, a_k$ odgovori dobiveni iz k nezavisnih pokretanja algoritma A na ulazu $x \subseteq \Sigma^*$.

Ako postoji $i \in \{1, \ldots, k\}$ takav da $a_i = 1$ tada zbog svojstva algoritma A sa sigurnošću znamo da $x \in L$.

S druge strane, ako je

$$a_1 = a_2 = \cdots = a_k = 0$$

vjerojatnost da $x \in L$ je najviše $\left(\frac{1}{2}\right)^k$ jer je vjerojatnost da algoritam pogriješi na ulazu iz L najviše $\frac{1}{2}$. Budući da imamo k nezavisnih pokretanja, dobivamo da je vjerojatnost pogreške upravo $\left(\frac{1}{2}\right)^k$. □

Sada ćemo navesti primjere vjerojatnosnih algoritama s jednostranom greškom i dokazati da zadovoljavaju definiciju 3.7.

3.2.1 Vjerojatnosni protokol za jednakost

Vjerojatnosni protokol za jednakost je zapravo pojednostavljena verzija Protokola R_k iz poglavlja 2.1.1. Ponovno koristimo pretpostavke i pojmove uvedene u tom poglavlju. Slijedi pseudokod vjerojatnosnog protokola za jednakost.

- *Početna situacija:* Računalo R_I ima spremljeno n bitova $x = x_1 \ldots x_n$, a računalo R_{II} ima spremljeno $y = y_1 \ldots y_n$.
- *Prva faza:* Računalo R_I slučajnim odabirom izabire broj $p \in \text{PRIM}(n^2)$
- *Druga faza:* Računalo R_I računa $s = \text{Number}(x) \mod p$ te šalje računalu R_{II} binarnu reprezentaciju broja s i broj p.
- *Treća faza:* Nakon što je računalo R_{II} primilo brojeve s i p, ono računa $q = \text{Number}(y) \mod p$ i čini sljedeće
  - Ako $q \neq s$, vraća $x \neq y$
  - Ako $q = s$, vraća $x = y$

Pokažimo da dizajnirani protokol zadovoljava definiciju algoritma s jednostranom greškom.

Definirajmo problem odluke (Σ, L) sa $\Sigma = \{0, 1\}$ i

$$L = \{(x, y) \,|\, x, y \in \{0,1\}^n,\ x \neq y,\ n \in \mathbb{N}\}$$

Ako $x = y$, tada $(x, y) \notin L$. Protokol u tom slučaju vraća $x = y$ jer $\text{Number}(x) \mod p = \text{Number}(y) \mod p$. Što bi značilo da za elemente izvan L, vjerojatnost greške je 0, tj. zadovoljena je druga tvrdnja iz definicije 3.7.

S druge strane, ako $x \neq y$, tada $(x, y) \in L$. Budući da je vjerojatnost izbora lošeg prostog broja iz skupa PRIME(n^2) najviše $\frac{2 \ln n}{n}$, vjerojatnost da protokol vrati $x \neq y$ je najmanje $1 - \frac{2 \ln n}{n}$, što je veće od $\frac{1}{2}$ za svai $n \geq 9$. Time je zadovoljena i prva tvrdnja iz definicije 3.7, te smo dokazali da je dizajnirani protokol primjer algoritma s jednostranom greškom.

3.2.2 Pojednostavljeni Solovay - Strassen algoritam (PSSA)

Odlučivanje je li broj prost ili ne jedan je od najpoznatijih računarskih problema. Tek 2004. godine indijski matematičari Agrawal, Kayal i Saxena su osmislili prvi deterministički, polinomno složen algoritam za testiranje prostih brojeva. No, taj algoritam se u praksi pokazao neefikasnim u usporedbi s nekim drugim eksponencijalnim algoritmima.

Pojednostavljeni Solovay - Strassen algoritam je primjer algoritma s jednostranom greškom koji određuje je li neparan broj n (uz uvjet da je $\frac{n-1}{2}$ također neparan broj) prost ili složen. Neka je C skup svih složenih brojeva. Algoritam za zadani n vraća je li on u skupu C ili nije. Slijedi pseudokod pojednostavljenog Solovay - Strassen algoritma.

- *Ulaz:* Neparan broj n za kojeg vrijedi: $\frac{n-1}{2}$ je također neparan
- *Prvi korak:* Slučajnim odabirom izaberi $a \in \{1, 2, \ldots, n-1\}$
- *Drugi korak:* Izračunaj $A = a^{\frac{n-1}{2}} \mod n$
  - Ako $A \in \{-1, 1\}$, n je prost ($n \notin C$)
  - Ako $A \notin \{-1, 1\}$, n je složen ($n \in C$)

Prije nego pokažemo da je pojednostavljeni Solovay - Strassen primjer algoritma s jednostranom greškom, trebamo izreći jednu propoziciju čiji dokaz možete naći u [1].

Propozicija 3.9. *Za svaki neparan n, za kojeg vrijedi da je $\frac{n-1}{2}$ također neparan, vrijedi*

- *Ako je n prost, tada $a^{\frac{n-1}{2}} \mod n \in \{-1, 1\}$, za svaki $a \in \{1, \ldots, n-1\}$.*
- *Ako je n složen, tada $a^{\frac{n-1}{2}} \mod n \notin \{-1, 1\}$, za barem polovinu a-ova iz skupa $\{1, \ldots, n-1\}$.*

Teorem 3.10. *Pojednostavljeni Solovay - Strassen algoritam je algoritam s jednostranom greškom.*

Dokaz. Pretpostavimo da je n prost ($n \notin C$). Tada zbog prve tvrdnje Propozicije 3.9 možemo zaključiti da je

$$a^{\frac{n-1}{2}} \equiv \pm 1 \mod n$$

Stoga je za svaki $a \in \{1, \ldots, n-1\}$ izlaz $n \notin C$, tj.

$$\text{Za svaki } n \notin C, \ \text{Prob}(\text{PSSA}(x) = 0) = 1$$

S druge strane, ako je n složen ($n \in C$). Tada zbog druge tvrdnje iz Propozicije 3.9 možemo zaključiti da je

$$\text{Za svaki } n \in C, \ \text{Prob}(\text{PSSA}(x) = 1) \geq \frac{1}{2}$$

Time smo pokazali da algoritam zadovoljava oba uvjeta iz definicije 3.7. □

3.3 Algoritmi s ograničenom greškom

Sljedeća vrsta vjerojatnosnih algoritama su algoritmi s ograničenom greškom. Počnimo s formalnom definicijom:

Definicija 3.11. Neka je F proizvoljna funkcija. Kažemo da je vjerojatnosni algoritam A algoritam s ograničenom greškom ako postoji realni broj ϵ, $0 < \epsilon \leq \frac{1}{2}$, takav da za svaki x iz domene funkcije F vrijedi

$$\text{Prob}(A(x) = F(x)) \geq \frac{1}{2} + \epsilon$$

◁

Klasa vjerojatnosnih algoritama s ograničenom greškom ima dva bitna svojstva:

- Za smanjenje vjerojatnosti greške ispod određene konstante δ, uvijek je dovoljan konstantan broj nezavisnih ponavljanja algoritma (s obzirom na duljinu ulaza) s istim ulazom.
- Bilo kakvo ublažavanje zahtjeva definicije 3.11. može dovesti do situacije u kojoj polinomno mnogo izvršavanja (opet s obzirom na duljinu ulaza) algoritma s istim ulazu ne bude dovoljno da se smanji vjerojatnost greške ispod određene konstante δ.

Primijetimo da je gornja granica za vjerojatnost greške $\frac{1}{2} - \epsilon$ definirana fiksnim brojem ϵ, tj. ne ovisi o ulazu. Upravo ta zajamčena udaljenost od vjerojatnosti $\frac{1}{2}$ za bilo koji ulaz, osigurava nam postojanje učinkovitog načina za smanjenje vjerojatnosti greške na proizvoljno mali δ.

Također, primijetimo da Las Vegas algoritmi, kao i algoritmi s jednostranom greškom, pripadaju klasi algoritama s ograničenom greškom. Primjerice Las Vegas algoritam koji ima vjerojatnost greške najviše $\frac{1}{2}$, nakon 2 ponavljanja ima vjerojatnost greške najviše $\frac{1}{4}$, pa tako zadovoljava uvjet iz definicije 3.11.

Ubuduće ćemo k-to ponavljanje algoritma A označavati s A_k.

3.3.1 Algoritam A_t

Sada ćemo analizirati brzinu smanjenja vjerojatnosti pogreške algoritama s ograničenom greškom, s obzirom na broj ponavljanja s istim ulazom.

Dakle, neka je A neki algoritam s ograničenom greškom, neka je t pozitivan cijeli broj. Definirajmo algoritam A_t sljedećim pseudokodom:

- *Ulaz:* x
- *Prvi korak:* t puta pokreni algoritam A te spremi izlaze $a_1, \ldots, a_t$
- *Drugi korak:*
  - Ako postoji a koji se pojavljuje najmanje $\lceil \frac{t}{2} \rceil$ puta u nizu $a_1, \ldots, a_t$, vrati „a”
  - inače, vrati „?”

Neka je $\epsilon > 0$ konstanta takva da za svaki ulaz x vrijedi:

$$\text{Prob}(A(x) = F(x)) \geq \frac{1}{2} + \epsilon$$

S p označimo vjerojatnost uspjeha (izlaz je točan) algoritma A s ulazom x:

$$p = p(x) = \text{Prob}(A(x) = F(x)) = \frac{1}{2} + \epsilon_x \quad \text{za svaki } \epsilon_x \geq \epsilon$$

Očito je da A_t računa krivi rezultat ili „?” jedino ako je točan rezultat bio izračunat manje od $\lceil \frac{t}{2} \rceil$ puta u t nezavisnih pokretanja algoritma A.

S $\text{pr}_i(x)$ označimo vjerojatnost da je A dobio točan rezultat u točno i pokretanja, za $i < \lceil \frac{t}{2} \rceil$. Pokušajmo naći gornju granicu za tu vjerojatnost.

Postoji $\binom{t}{i}$ načina da se izabere i od t pozicija na kojima je algoritam dobio točne rezultate. Za svaku poziciju p je vjerojatnost računanja točnog rezultata, a $1 - p$ je vjerojatnost računanja netočnog rezultata. Dakle, p^i je vjerojatnost dobivanja točnog rezultata na točno i mjesta, a $(1-p)^{t-i}$ je vjerojatnost dobivanja netočnog rezultata na preostalih $t - i$ mjesta. Stoga, vrijedi jednakost:

$$\text{pr}_i(x) = \binom{t}{i} \cdot p^i \cdot (1-p)^{t-i}$$

Koristimo $(1-p)^{t-i} = (1-p)^i \cdot (1-p)^{t-2i}$ te svojstvo umnoška baza s istim eksponentom $a^x \cdot b^x = (a \cdot b)^x$:

$$\text{pr}_i = \binom{t}{i} \cdot (p \cdot (1-p))^i \cdot (1-p)^{t-2i}$$

Koristimo definiciju vjerojatnosti p, $p = \frac{1}{2} + \epsilon_x$:

$$\begin{aligned}\mathrm{pr}_i(x) &= \binom{t}{i} \cdot \left(\left(\frac{1}{2}+\epsilon_x\right)\cdot\left(\frac{1}{2}-\epsilon_x\right)^i\right)\cdot\left(\frac{1}{2}-\epsilon_x\right)^{2\cdot\left(\frac{t}{2}-i\right)} \\ &= \binom{t}{i}\cdot\left(\frac{1}{4}-\epsilon_x^2\right)^i\cdot\left(\left(\frac{1}{2}-\epsilon_x\right)^2\right)^{\frac{t}{2}-i}\end{aligned}$$

Budući da je $\frac{1}{2} - \epsilon_x < \frac{1}{2} + \epsilon_x$, jer je $\epsilon_x > 0$, imamo:

$$\begin{aligned}\mathrm{pr}_i(x) &< \binom{t}{i}\cdot\left(\frac{1}{4}-\epsilon_x^2\right)^i\cdot\left(\left(\frac{1}{2}-\epsilon_x\right)\cdot\left(\frac{1}{2}+\epsilon_x\right)\right)^{\frac{t}{2}-i} \\ &= \binom{t}{i}\cdot\left(\frac{1}{4}-\epsilon_x^2\right)^i\cdot\left(\frac{1}{4}-\epsilon_x^2\right)^{\frac{t}{2}-i} \\ &= \binom{t}{i}\cdot\left(\frac{1}{4}-\epsilon_x^2\right)^{\frac{t}{2}}\end{aligned}$$

Po definiciji $\epsilon_x \geq \epsilon$, pa vrijedi

$$\mathrm{pr}_i(x) \leq \binom{t}{i}\cdot\left(\frac{1}{4}-\epsilon^2\right)^{\frac{t}{2}} \tag{3.8}$$

Sada možemo naći donju ogradu za vjerojatnost uspjeha algoritma A_t, koristeći gornju ogradu za $\mathrm{pr}_i(x)$. Rekli smo da A_t računa točan rezultat $F(x)$ ako se on pojavi na barem $\lceil \frac{t}{2} \rceil$ mjesta u nizu izlaza $a_1, \ldots, a_t$. Dakle, vjerojatnost uspjeha algoritma A_t je suma vjerojatnosti $\mathrm{pr}_i(x)$, gdje je $i \in \{\lceil \frac{t}{2} \rceil, \ldots, t\}$. Mi ćemo u ovom slučaju koristiti komplementarne događaje, $\mathrm{pr}_i(x)$ za $i \in \{0, \ldots, \lfloor \frac{1}{2} \rfloor\}$ i drugu tvrdnju Propozicije 1.2.

$$\begin{aligned}\mathrm{Prob}(A_t(x) = F(x)) &= 1 - \sum_{i=0}^{\lfloor \frac{t}{2} \rfloor} \mathrm{pr}_i(x) \\ &\overset{(3.8)}{>} 1 - \sum_{i=0}^{\lfloor \frac{t}{2} \rfloor} \binom{t}{i}\cdot\left(\frac{1}{4}-\epsilon^2\right)^{\frac{t}{2}} \\ &= 1 - \left(\frac{1}{4}-\epsilon^2\right)^{\frac{t}{2}} \cdot \sum_{i=0}^{\lfloor \frac{t}{2} \rfloor} \binom{t}{i}\end{aligned}$$

Budući da vrijedi $\sum_{i=0}^{t} \binom{t}{i} = 2^t$:

$$\begin{aligned}\text{Prob}(A_t(x) = F(x)) &> 1 - \left(\frac{1}{4} - \epsilon^2\right)^{\frac{t}{2}} \cdot 2^t \\ &= 1 - (1 - 4 \cdot \epsilon^2)^{\frac{t}{2}}\end{aligned}$$

Dakle, gornja granica za vjerojatnost greške algoritma A_t je $(1 - 4 \cdot \epsilon^2)^{\frac{t}{2}}$, što teži nuli kako se broj ponavljanja, tj. broj t, povećava.

Ako bismo željeli odrediti k tako da vrijedi

$$\text{Prob}(A_k(x) = F(x)) \geq 1 - \delta$$

za unaprijed izabranu konstantu δ, dovoljno je da uzmemo

$$\begin{aligned}\delta &\leq (1 - 4 \cdot e^2)^{\frac{k}{2}} \\ \ln \delta &\leq \frac{k}{2} \cdot \ln(1 - 4 \cdot e^2) \\ k &\geq \frac{2 \cdot \ln \delta}{\ln(1 - 4 \cdot \epsilon^2)}\end{aligned}$$

Ako su ϵ i δ konstante, tada je i k konstanta pa možemo zaključiti

$$\text{Time}_{A_k}(n) \in \mathcal{O}(\text{Time}_A(n))$$

Bitna posljedica gornjeg zaključka je da ako je A algoritam brži od bilo kojeg deterministčkog algoritma, tada je i A_k efikasniji od deterministčkih algoritama, ali s vjerojatnošću greške ispod fiksne konstante δ.

3.4 Algoritmi s neograničenom greškom

Postoje vjerojatnosni algoritmi koje zovemo algoritmi s neograničenom greškom.

Definicija 3.12. Neka je F proizvoljna funkcija. Kažemo da vjerojatnosni algoritam A algoritam s neograničenom greškom računa funkciju F, ako za svaki x iz domene funkcije F vrijedi:

$$\text{Prob}(A(x) = F(x)) > \frac{1}{2}$$

$\triangleleft$

Odmah se nameće pitanje koja je razlika između algoritama s ograničenom greškom i algoritama s neograničenom greškom. Vidimo da se definicije razlikuju samo zbog ϵ koji nam je bio fiksna udaljenost od $\frac{1}{2}$ za svaki ulaz x. Upravo je u tome razlika.

Kod algoritama s neograničenom greškom može se dogoditi da udaljenost između $\frac{1}{2}$ i vjerojatnosti greške teži k nuli s povećanjem duljine ulaza.

Primjerice, neka vjerojatnosni algoritam A slučajno bira između $2^{|x|}$ determinističkih strategija za svaki ulaz x. Ako većina njih daje točan rezultat, tada je A algoritam s neograničenom greškom. Dakle, dopuštamo

$$\text{Prob}A(x) = F(x)) = \frac{1}{2} + \frac{1}{2^{|x|}}$$

Vidimo da $\epsilon_x = \frac{1}{2^{|x|}}$ teži nuli, kako povećavamo duljinu ulaza.

Da bismo jasnije vidjeli razliku, analizirat ćemo koliko je ponovnih pokretanja algoritma s neograničenom greškom (A) potrebno da bismo dobili algoritam s ograničenom greškom (A_k) za kojeg vrijedi

$$\text{Prob}(A_k(x) = F(x)) \geq 1 - \delta \tag{3.9}$$

za unaprijed izabranu konstantu δ.

Budući da je analiza vjerojatnosti greške algoritma s ograničenom greškom A_t s obzirom na ϵ_x valjana i u slučaju algoritama s neograničenom greškom, možemo koristiti rezultat iz prošlog poglavlja

$$\text{Prob}(A_t(x) = F(x)) \geq 1 - (1 - 4 \cdot \epsilon_x^2)^{\frac{t}{2}}$$

Sada možemo izračunati donju ogradu za k tako da vrijedi (3.9):

$$k \geq \frac{2 \cdot \ln \delta}{\ln(1 - 4 \cdot \epsilon_x^2)}$$

Koristeći $\epsilon = \frac{1}{2^{|x|}}$ dobivamo:

$$k \geq \frac{2 \cdot \ln \delta}{\ln(1 - 4 \cdot 2^{-2 \cdot |x|})}$$

Koristimo tvrdnju: Za svaki $x \in (0, 1)$, $\frac{-x}{1-x} \leq \ln(1 - x) \geq -x$; čiji dokaz možete naći u [3]).

$$\begin{aligned} k &\geq \frac{2 \cdot \ln \delta}{-2^{-2 \cdot |x|}} \\ &= (-2 \cdot \ln \delta) \cdot 2^{2 \cdot |x|} \end{aligned}$$

Zaključujemo da ako želimo vjerojatnosni algoritam A_k, takav da vrijedi (3.9), imat ćemo eksponencijalno povećanje složenosti s obzirom na $|x|$, tj.

$$\text{Time}_{A_k}(x) \in \mathcal{O}(2^{2 \cdot |x|} \cdot \text{Time}_A(x))$$

Što ne znači da niti jedan algoritam s neograničenom greškom nije primjenjiv. Primjerice, kada bismo imali $\epsilon_x = \frac{1}{\log |x|}$, dogodit će se polinomno povećanje složenosti, što je ipak puno bolje od eksponencijalne.

3.4.1 Protokol UMC

UMC protokol primjer je algoritma s neograničenom greškom. Kao i u prošlim primjerima, pretpostavljamo da postoje dva računala R_I i R_{II}. Zatim, koristit ćemo već uvedene pojmove iz poglavlja 2.1.1, te već spomenuti jezik

$$L = \{(x, y) \,|\, x, y \in \{0,1\}^n, x \neq y, n \in \mathbb{N}\}$$

Slijedi pseudokod protokola UMC:

- *Početna situacija:* Računalo R_I ima spremljeno n bitova $x = x_1x_2 \dots x_n$, te računalo R_{II} također ima spremljeno n bitova $y = y_1y_2 \dots y_n$, $n \in \mathbb{N}\backslash\{0\}$. Ulaz (x, y) treba biti prihvaćen ako i samo ako $x \neq y$.
- *Prva faza:* Računalo R_I slučajno izabire $j \in \{1, \dots, n\}$ te šalje broj j i bit x_j računalu R_{II}.
- *Druga faza:* Računalo R_{II} uspoređuje x_j i y_j.
  - Ako $x_j \neq y_j$, prihvaća ulaz, tj. $(x, y) \in L$
  - Ako $x_j = y_j$, tada
    - R_{II} prihvaća (x, y) s vjerojatnošću $\frac{1}{2} - \frac{1}{2n}$
    - R_{II} odbacuje (x, y) s vjerojatnošću $\frac{1}{2} + \frac{1}{2n}$

Dokazat ćemo da je UMC protokol algoritam s neograničenom greškom. Kao i u prošlim primjerima, razmotrit ćemo dva slučaja: $(x, y) \in L$ i $(x, y) \notin L$.

Neka je $(x, y) \notin L$, tj. $x = y$. U tom slučaju postoji točno $2n$ izračunavanja C_{il}, gdje i označava slučajno izabran broj iz $\{1, \dots, n\}$ u prvoj fazi algoritma, a l označava odgovor protokola - 0 za odbacivanje, 1 za prihvaćanje ulaza (x, y).

Dakle, imamo vjerojatnosni prostor

$$S_{\mathrm{UMC},(x,y)} = \{C_{il} \,|\, 1 \leq i \leq n,\, l \in \{0,1\}\}$$

Budući da smo pretpostavili da $(x, y) \notin L$, ne postoji i takav da vrijedi $x_i \neq y_i$, pa ćemo završiti u drugom slučaju u drugoj fazi, gdje algoritam prihvaća (x, y) s vjerojatnošću $\frac{1}{2} - \frac{1}{2n}$, a odbija (x, y) s vjerojatnošću $\frac{1}{2} + \frac{1}{2n}$. Također, vjerojatnost da je izabran broj i iz skupa $\{1, \dots, n\}$ je upravo $\frac{1}{n}$. Dakle, imamo sljedeće vjerojatnosti, za $i \in \{1, \dots, n\}$:

$$\mathrm{Prob}(\{C_{i0}\}) = \frac{1}{n} \cdot \left(\frac{1}{2} + \frac{1}{2n}\right)$$

$$\mathrm{Prob}(\{C_{i1}\}) = \frac{1}{n} \cdot \left(\frac{1}{2} - \frac{1}{2n}\right)$$

Želimo dokazati da je vjerojatnost događaja da UMC protokol vraća „0”, u ovom slučaju, veća od $\frac{1}{2}$. Događaj gdje algoritam odbacuje ulaz je

$$A_0 = \{C_{i0} \mid 1 \leq i \leq n\}$$

Pa je vjerojatnost tog događaja jednaka

$$\begin{aligned}
\text{Prob}(A_0) &= \text{Prob}(\text{UMC odbacuje } (x, y)) \\
&= \sum_{i=1}^{n} \text{Prob}(\{C_{i0}\}) \\
&= \sum_{i=1}^{n} \frac{1}{n} \cdot \left(\frac{1}{2} + \frac{1}{2n}\right) \\
&= n \cdot \frac{1}{n} \cdot \left(\frac{1}{2} + \frac{1}{2n}\right) \\
&= \frac{1}{2} + \frac{1}{2n} > \frac{1}{2}
\end{aligned}$$

Preostaje nam slučaj kada $(x, y) \in L$. Tada postoji $j \in \{1, \ldots, n\}$ takav da vrijedi $x_j \neq y_j$. Bez smanjenja općenitosti, možemo pretpostaviti da postoji samo jedan takav indeks, budući da je to najgori slučaj za našu analizu.

Imamo izračunavanje gdje računalo R_I izabire indeks j u prvom koraku. U tom slučaju algoritam s vjerojatnošću 1 prihvaća ulaz. Označimo pripadajuće izračunavanje s C_j. Ostalih $2(n-1)$ označimo kao u prvom slučaju s C_{il}, gdje $i \in \{1, \ldots, n\} \setminus \{j\}$ označava indeks odabran u prvom koraku, te $l \in \{0, 1\}$ označava prihvaćanje ili odbijanje ulaza. Sada imamo vjerojatnosni prostor $(S_{\text{UMC},(x,y)}, \text{Prob})$, gdje je

$$S_{\text{UMC},(x,y)} = \{C_j\} \cup \{C_{il} \mid 1 \leq i \leq n,\, i \neq j,\, l \in \{0, 1\}\}$$

te koristeći istu logiku kao i u prvom slučaju imamo sljedeće vjerojatnosti, za $i \in \{1, \ldots, n\}$

$$\text{Prob}(\{C_j\}) = \frac{1}{n} \tag{3.10}$$

$$\text{Prob}(\{C_{i0}\}) = \frac{1}{n} \cdot \left(\frac{1}{2} + \frac{1}{2n}\right)$$

$$\text{Prob}(\{C_{i1}\}) = \frac{1}{2} \cdot \left(\frac{1}{2} - \frac{1}{2n}\right) \tag{3.11}$$

Zanima nas vjerojatnost događaja gdje protokol UMC prihvaća rezultat, taj događaj je

$$A_1 = \{C_j\} \cup \{C_{i1} \mid 1 \leq i \leq n,\, i \neq j\}$$

Željeli bismo dokazati da je vjerojatnost tog događaja veća od $\frac{1}{2}$:

$$\begin{aligned}\text{Prob}(A_1) &= \text{Prob}(\text{UML prihvaća } (x,y)) \\ &= \text{Prob}(\{C_j\}) + \sum_{i=1,\, i\neq j}^{n} \text{Prob}(\{C_{i1}\})\end{aligned}$$

Koristeći (3.10) i (3.11) dobivamo:

$$\text{Prob}(A_1) = \frac{1}{n} + \sum_{i=1,\, i\neq j}^{n} \frac{1}{n} \cdot \left(\frac{1}{2} - \frac{1}{2n}\right)$$

Koristeći: $\frac{1}{n} = \frac{1}{2n} + \frac{1}{2n}$ te $\frac{1}{n} \cdot \left(\frac{1}{2} - \frac{1}{2n}\right) = \frac{1}{2n} - \frac{1}{2n^2}$, te činjenicu da izraz koji sumiramo ne ovisi o indeksu i, pa je svejedno piše li $\sum_{i=1,\, i\neq j}^{n}$ ili $\sum_{i=1}^{n-1}$, dobivamo:

$$\begin{aligned}&= \frac{1}{2n} + \frac{1}{2n} + \sum_{i=1}^{n-1} \left(\frac{1}{2n} - \frac{1}{2n^2}\right) \\ &= \frac{1}{2n} + \sum_{i=1}^{n} \frac{1}{2n} - \sum_{i=1}^{n-1} \frac{1}{2n^2} \\ &= \frac{1}{2n} + \frac{1}{2} - \frac{n-1}{2n^2} \\ &= \frac{1}{2} + \frac{1}{2n^2} > \frac{1}{2}\end{aligned}$$

Dakle, dobili smo da protokol UMC, i u slučaju $(x,y) \in L$, kao i u slučaju $(x,y) \notin L$ s vjerojatnošću većom od $\frac{1}{2}$ vraća točan odgovor. Time smo dokazali da je protokol UMC algoritam s neograničenom greškom prema definiciji 3.12.

Bibliografija

[1] J. Hromkovič, *Algorithmics for Hard Problems*, Second Edition, Springer, 2004.

[2] J. Hromkovič, *Theoretical Computer Science*, Springer, 2004.

[3] J. Hromkovič, *Design and Analysis of Randomized Algorithms*, Second Edition, Springer, 2005.

[4] M. Mitzenmacher, E. Upfal, *Probability and computing: Randomized algorithms and probabilistic analysis*, Cambridge University Press, 2005.

Sažetak

U ovom diplomskom radu bavili smo se vjerojatnosnim algoritmima. Vjerojatnosni algoritmi su vrsta algoritama koji u barem jednom dijelu svog izvršavanja donose slučajnu odluku o daljnjem tijeku izvršavanja.

Prvo smo definirali dva teorijska modela. Algoritmi iz prvog modela na slučajan način biraju deterministički algoritam koji zatim koriste na danom ulazu, dok algoritmi koji pripadaju drugom modelu tijekom izvršavanja donose više slučajnih odluka. Primjer algoritma građenog po prvom modelu je Protokol R_k, a primjer algoritma građenog po drugom modelu je vjerojatnosni Quicksort.

Zatim smo se bavili podjelom vjerojatnosnih algoritama s obzirom na vjerojatnost greške.

Las Vegas algoritmi su vjerojatnosni algoritmi koji ne dopuštaju grešku. Za njih smo naveli dvije podvrste: oni koji dopuštaju odgovor „?" i oni koji ga ne dopuštaju. Dokazali smo da je podvrste ovih algoritama moguće svesti jedne na druge te smo analizirali Random - select algoritam kao primjer Las Vegas algoritma koji ne dopušta odgovor „?", te Protokol LV_{10} koji ga dopušta.

Algoritmi s jednostranom greškom dopuštaju grešku, ali samo ako je vjerojatnost greške manja od $\frac{1}{2}$ i to samo na ulazima koji bi trebali biti prihvaćeni. Naveli smo primjere algoritama s jednostranom greškom: vjerojatnosni protokol za jednakost te Pojednostavljeni Solovay - Strassen algoritam.

Algoritmi s ograničenom greškom zahtijevaju da je vjerojatnost greške za svaki ulaz manja od $\frac{1}{2} - \epsilon$, gdje je ϵ fiksni broj iz $\left(0, \frac{1}{2}\right]$. Analizirali smo brzinu smanjenja vjerojatnosti greške nekog algoritma s ograničenom greškom A s obzirom na broj nezavisnih ponavljanja s istim ulazom. Došli smo do zaključka da vjerojatnost greške na algoritmima s ograničenom greškom možemo spustiti proizvoljno nisko, uz konstantan broj nezavisnih ponavljanja.

Algoritmi s neograničenom greškom zahtijevaju samo da je vjerojatnost greške manja od $\frac{1}{2}$. Primjer jednog takvog algoritma je Protokol UMC za koji smo dokazali da zadovoljava definiciju algoritma s neograničenom greškom.

Summary

In this master's thesis we have dealt with randomized algorithms. Randomized algorithms are a type of algorithms that in at least one part of their execution make a random decision about the further course of execution.

We first define two theoretical models. The algorithms from the first model randomly select a deterministic algorithm, which they then use at a given input, while the algorithms belonging to the second model make more random decisions during execution. An example of an algorithm built on the first model is the R_k protocol, and an example of an algorithm built on the second model is the randomized Quicksort.

We then dealt with the classification of randomized algorithms with respect to the probability of error.

Las Vegas algorithms are randomized algorithms that guarantee that every compute answer is correct. We have listed two subtypes for them: those that allow the answer „?" and those that do not allow it. We have proven that the subtypes of these algorithms can be transformed to one another, and we have analyzed the Random - select algorithm as an example of a Las Vegas algorithm that does not allow the „?" answer, and the LV_{10} protocol that allows it.

One-sided error algorithms allow errors, but only if the probability of an error is less than $\frac{1}{2}$ and only at the inputs that should be accepted. We have provided examples of one-sided error algorithms: Randomized Equality Protocol and Simplified Solovay - Strassen Algorithm.

Bounded-error algorithms require that the error probability for each input is less than $\frac{1}{2} - \epsilon$, where ϵ is a fixed number from $\left(0, \frac{1}{2}\right]$. We have analyzed the rate of error reduction of a bounded-error algorithm A with respect to the number of independent repetitions with the same input. We have come to the conclusion that the probability of error on bounded-error algorithms can be lowered arbitrarily, with a constant number of independent repetitions.

Unbounded-error algorithms only require that the error probability is less than $\frac{1}{2}$. An example of such an algorithm is the UMC protocol, which we have proven to satisfy the definition of an unbounded-error algorithm.

Životopis

Rođena sam 28. kolovoza 1995. godine u Našicama. Osnovnu školu završila sam u Slatini. Nakon toga upisujem opću gimnaziju Marka Marulića u Slatini. Tijekom osnovne i srednje škole sudjelovala sam na nizu natjecanja iz područja matematike i logike, a posebno bih istaknula osvojeno treće mjesto na 52. državnom natjecanju iz matematike koje je održano u Opatiji 2011. godine. Do 2016. godine aktivno sam igrala stolni tenis te sam se natjecala u 2. hrvatskoj ženskoj ligi.

Po završetku srednjoškolskog obrazovanja, godine 2014. upisala sam Prirodoslovno–matematički fakultet u Zagrebu, smjer Matematika. Nakon tri godine upisala sam na istoimenom fakultetu diplomski sveučilišni studij Računarstvo i matematika.