Vaje iz alternativnih modelov računanja

Sergio Cabello

9. november 2010

naslov: Vaje iz alternativnih modelov računanja

izdaja: prva izdaja

založnik: samozaložba

avtor: Sergio Cabello

leto izida: 2010 (v Ljubljani)

natis: elektronsko gradivo
http://www.fmf.uni-lj.si/~cabello/gradiva/vajeamr.pdf

CIP - Kataložni zapis o publikaciji
Narodna in univerzitetna knjižnica, Ljubljana

510.5(075.8)(076.1)

CABELLO, Sergio
Vaje iz alternativnih modelov računanja [Elektronski vir] / Sergio Cabello. - 1. izd. - El. knjiga. - V Ljubljani : samozal., 2010

Način dostopa (URL): http://www.fmf.uni-lj.si/~cabello/gradiva/vajeamr.pdf

ISBN 978-961-276-042-7

253228544

Kazalo

Priporočam naslednje monografije, ki pokrijejo vsaka svoj del predmeta:

- J. JaJa. Introduction to parallel algorithms. Addison-Wesley, 1992.
- R. Motwani in P. Raghavan. Randomized Algorithms. Cambridge University Press, 2001.
- M. Mitzenmacher in E. Upfal. Probability and Computing. Cambridge University Press, 2005.
- J. Kleinberg in É. Tardos. Algorithm Design. Addison-Wesley, 2005.
- V.V. Vazirani. Approximation Algorithms. Springer-Verlag, 2001.

Vse skupaj obravnavajo seveda precej več tém, kot jih obravnavamo pri predmetu. Na spletni strani Permute-With-Allso na voljo tudi skripta in prosojnice, ki so bolj neformalne in zato včasih lažje razumljive. Poiščite! V najslabšem primeru lahko najdete druge zanimive stvari.

Zahvala. Zahvaljujem se Gašperju Fijavžu za številne popravke. Seveda pa je avtor odgovoren za vse preostale napake.

1 Naključni algoritmi

1.1. Podprogram NepoštenKovanec($\cdot$) vrne '1' s verjetnostjo p in '0' z verjetnostjo $1-p$. Vrednost $p \in (0,1)$ ni znana. Opiši naključni algoritem PoštenKovanec($\cdot$), ki enakomerno vrača '1' oziroma '0' in kot naključen podprogram uporablja samo NepoštenKovanec. Naj bo N slučajna spremenljvka, ki pove, kolikokrat pokliče PoštenKovanec podprogram NepoštenKovanec. Kakšno je matematično upanje $\mathbb{E}[N]$?

1.2. Delamo naslednji poskus. Imamo n košar $B_1, B_2, \ldots, B_n$. Na vsakem koraku dodamo novo žogo v naključno izbrano košaro izmed $B_1, B_2, \ldots, B_n$. Naj bo X število korakov do dogodka 'vsaka košara vsebuje vsaj eno žogo'. Kakšno je matematično upanje slučajne spremenljevke X? (Namig: opazuj slučajno spremenljvko X_i, ki pove, koliko žog porabimo, da gremo od dogodka 'i praznih košar' do dogodka '$(i-1)$ praznih košar'.)

1.3. Denimo, da je rojstni dan naključnega človeka enakomerno porazdeljen po dneh v letu (prestopna leta zanemari). Recimo, da je v isti sobi k naključno izbranih ljudi. Naj bo X število parov ljudi v sobi, ki imajo isti rojstni dan. Za katere k je pričakovana vrednost spremenljevke X najmanj 1?

Kaj se dogaja na planetu Kriptonu, kjer ima vsako leto n dni?

1.4. Problem Max-Cut je naslednja optimizacijska naloga: v grafu G iščemo prerez, ki ima največjo moč. Problem Max-Cut je NP-težak. Opiši naključni algoritem, ki vrne prerez s pričakovano močjo najmanj $|E(G)|/2$.
Ali to pomeni, da vedno obstaja prerez s močjo najmanj $|E(G)|/2$?

1.5. Delamo naslednji poskus. Na začetku imamo košaro, ki vsebuje natanko 1 belo žogo in 1 črno žogo. Nato n krat ponovimo naslednji korak: izberemo naključno žogo iz košare in jo damo nazaj v košaro z dodatno žogo iste barve. V košari je na koncu $n+1$ žog. Pokaži, da je število belih žog enakomerno porazdeljeno na $\{1, \ldots, n\}$.

1.6. *k-mediana* množice števil S je element m iz S, ki zadošča $k = |\{a \in S \mid a \leq m\}|$. Opazujemo naslednji algoritem, ki vrne k-mediano:

Algorithm Find($S; k$)
Input: množica S z n različnimi števili; število k med 1 in n
Output: k-mediana množice S
1. Enakomerno izberemo naključen element y iz S;
2. $S_1 = \{x \in S \mid x < y\}$;
3. $S_2 = \{x \in S \mid x > y\}$;
4. **if** $|S_1| \geq k$
5. **then return** Find(S_1; k);
6. **if** $|S_1| = k-1$
7. **then return** y;
8. **if** $|S_1| < k-1$
9. **then return** Find(S_2; $k - |S_1| - 1$);

Pokaži, da je pričakovana časovna zahtevnost algoritma $O(n)$.
(Namig: zapiši rekurzivno enačbo za časovno zahtevnost.)

1.7. Opazujemo naslednji (okorni) algoritem, ki izračuna najmanjši interval, ki vsebuje n podanih števil:

Algorithm INTERVAL(S)
Input: množica S z n števili
Output: najmanjši interval, ki vsebuje S
1. **if** $|S| \leq 2$
2. **then return** najmanjši interval, ki vsebuje S;
3. sestavimo naključno podmnožico $S' \subseteq S$, kjer neodvisno dodamo vsak element iz S s verjetnostjo $\frac{1}{\sqrt{n}}$;
4. $[a, b]$:=INTERVAL(S');
5. $S_I := S \setminus [a, b]$;
6. **return** INTERVAL($S_I \cup \{a, b\}$);

Pokaži, da je pričakovana časovna zahtevnost algoritma $O(n)$.
(Namig: kakšna je pričakovana vrednost za $|S'|$ in $|S_I|$?)

1.8. Opazujemo naslednjo varianto algoritma INSERTIONSORT, pri čemer naredimo na začetku naključno permutacijo:

Algorithm RANDINSERTIONSORT
Input: Tabela $A[1 \ldots n]$ različnih števil
Output: $A[1 \ldots n]$ vsebuje originalna števila urejena po velikosti
1. naključno permutiramo elemente množice A, pri čemer ima vsaka permutacija isto verjetnost;
2. **for** $j \leftarrow 2$ **to** n
3. **do** $key \leftarrow A[j]$;
4. $i \leftarrow j - 1$;
5. **while** $(i > 0)$ **and** $(A[i] > key)$
6. **do** $A[i + 1] \leftarrow A[i]$;
7. $i \leftarrow i - 1$;
8. $A[i + 1] \leftarrow key$;

Naj bo X slučajna spremenljvka, ki pove, kolikokrat delamo primerjavo $(A[i] > key)$ v vrstici 5 algoritma. Kakšna je pričakovana vrednost spremenljvke X? Analiza mora biti natančna in brez O-notacije. (Privzameš lahko, da v vrstici 5 algoritem *ne* dela primerjave $(A[i] > key)$, ko je $(i \leq 0)$.)

1.9. Opazujemo naslednji algoritem:

Algorithm BREZIMENA
Input: Naravno število $n > 1$
1. $korak \leftarrow 0$;
2. $K \leftarrow$ košara, ki vsebuje natanko eno belo in eno črno žogo;
3. **while** K vsebuje $< n$ belih žog
4. **do** $\check{Z} \leftarrow$ žoga izbrana naključno enakomerno iz košare K;
5. **if** $\check{Z}$ je žoga barve X;
6. **then** damo $\check{Z}$ nazaj v košaro K in damo še eno novo belo žogo v košaro K;
7. **else** damo $\check{Z}$ nazaj v košaro K;
8. $korak \leftarrow korak + 1$;

(a) Koliko je pričakovana vrednost slučajne spremenljke *korak* na koncu izvajanja algoritma, če je X bela barva.

(b) Koliko je pričakovana vrednost slučajne spremenljke *korak* na koncu izvajanja algoritma, če je X črna barva.

1.10. Oznaka $[n]$ naj pomeni $\{1, \dots n\}$. Naj bo $\sigma : [n] \to [n]$ permutacija enakomerno izbrana iz množice permutacij.

(a) Naj bo X število negibnih točk permutacije, $X = |\{i \in [n] \mid \sigma(i) = i\}|$. Kakšno je matematično upanje spremenljivke X in spremenljivke X^2?

(b) Koliko je

$$\mathbb{E}\left[\sum_{i\in[n]} |\sigma(i) - i|\right] ?$$

1.11. Opazujemo naslednji algoritem:

Algorithm BrezImena2
Input: Naravno število $n > 1$
1. $korak \leftarrow 0$;
2. $i_{korak} \leftarrow n$;
3. **while** $i_{korak} \neq 1$
4. **do** $i_{korak+1} \leftarrow$ element izbran naključno enakomerno med $\{1, \dots, i_{korak}\}$;
5. $korak \leftarrow korak + 1$;

Pokaži, da je pričakovana vrednost spremenljvke *korak* na koncu algoritma $O(\log n)$.

1.12. Poiskati želimo algoritem, ki vrne enakomerno naključno permutacijo elementov tabele $A[1 \dots n]$. Imamo podprogram Random(a, b), ki izbere enakomerno naključno celo število na intervalu $[a, b]$. Opazujemo naslednji algoritem:

Algorithm Permute-With-All(A)
1. **for** $i = 1$ **to** n
2. **do** swap $A[i] \longleftrightarrow A[$Random$(1, n)]$;

(a) Dokaži, da Permute-With-All ne izbere naključne permutacije enakomerno.

(b) Opiši algoritem, ki pravilno vrne enakomerno naključno permutacijo v linearnem času. Dokaži pravilnost.

1.13. Imamo podprogram RandomBit(), ki enakomerno vrne naključen bit. Torej velja

$$\Pr[\text{RandomBit}() = 0] = \Pr[\text{RandomBit}() = 1] = \tfrac{1}{2}.$$

Z uporabo psevdokode opiši algoritem Random(a, b), ki vrne enakomerno izbran naključni element iz $\mathbb{Z} \cap [a, b]$, kjer sta a in b naravni števili. Znotraj algoritma Random lahko dobimo naključne podatke le z uporabo RandomBit(). Kakšna je pričakovana časovna zahtevnost algoritma?

1.14. Naj bodo $a_1, \dots, a_n$ naključna realna števila, ki so izbrana enakomerno na intervalu $[0, 1]$. Opiši algoritem, ki uredi $a_1, \dots, a_n$ v pričakovanem linearnem času. Algoritem

lahko uporablja funkcijo *floor*. (Namig: poglej kvadrat števila elementov na intervalu $[i/n, (i+1)/n]$.)

1.15. Na predavanjih smo spoznali naključni algoritem Min-Cut za problem iskanja minimalnega prereza. Ta algoritem spremenimo na naslednji način: na vsakem koraku namesto izbora naključne povezave za skrčitev (*contraction*) izberemo dve naključni točki grafa in ju identificiramo. Pokaži, da obstajajo grafi, za katere je verjetnost, da tako popravljeni algoritem vrne minimalni prerez, eksponentno majhna. (Namig: $\binom{n}{n/2}$ je približno $2^n/\sqrt{n}$.)

1.16. Na predavanjih smo spoznali naključni algoritem Min-Cut za problem iskanja minimalnega prereza. Videli smo, da vrne Min-Cut minimalni prerez s verjetnostjo najmanj $2/(n(n-1))$, kjer je n število točk grafa. Kakšno je število skrčitev in verjetnost napake za naslednje variante.

(a) Min-Cut ponovimo t krat in vrnemo najboljši prerez, ki ga smo našli. (Namig: spomni se, da je $1 - x \leq e^{-x}$.)

(b) Povezave skrčimo po algoritmu Min-Cut, dokler ne dobimo grafa s k točkami, kjer je $k \leq n$. Potem naredimo ℓ kopij tega grafa, ki ima k točk, in na vsaki kopiji posebej uporabimo algoritem Min-Cut ter vrnemo najboljši najden prerez.

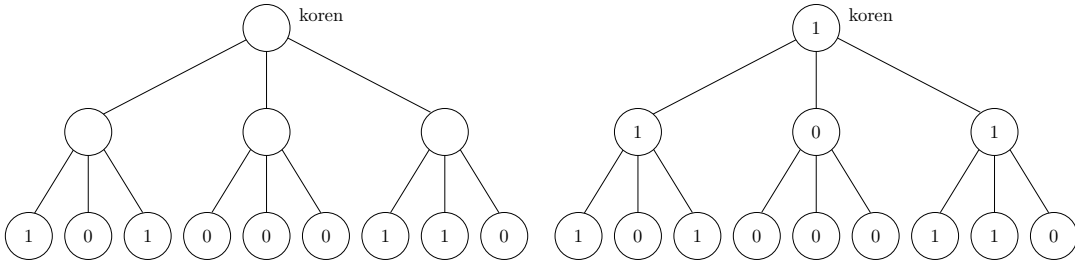
1.17. Opazujemo naslednji enostavni primer hazardne igre. Na začetku imamo dohodek nič. Igramo n krogov. V vsakem krogu se dohodek poveča za 1 s verjetnostjo 1/3 ali se zmanjša za 1 s verjetnostjo 2/3. Pokaži, da obstaja absolutna zgornja meja (neodvisna od n) za pričakovano število krogov, v katerih imamo pozitiven dohodek.

1.18. Naj bo T drevo s korenom. Vsi listi drevesa so na nivoju h in vsako notranje vozlišče drevesa T ima natanko 3 sinove. Torej ima drevo T natanko 3^h listov. Pišimo $n = 3^h$. Vsak list drevesa ima podano vrednost 0 ali 1. Situacija je predstavljena na levi strani slike 1.1.

Vrednost notranjega vozlišča je definirana rekurzivno po večinskem pravilu: ima vrednost 0, če imata dva ali trije sinovi vrednost 0, sicer ima vrednost 1. Zanima nas vrednost korena. Poglej desno stran slike 1.1.

Opiši naključni algoritem, ki izračuna vrednost korena in pogleda pričakovano podlinearno število listov. Pričakovano število listov, pri katerih algoritem pogleda vrednost, naj bo $O(n^c)$, pri čemer je konstanta $c < 1$.

Namig: ko vemo, da imata 2 sinova vrednost 1, vrednost tretjega sina ni pomembna.



Slika 1.1: Slika za vajo 1.18.

1.19. Pokaži, da je **ZPP**=**RP**$\cap$ co-**RP**.

1.20. Pokaži, da je **BPP**=co-**BPP**.

1.21. Podana je tabela $A[1 \cdots n]$ števil. Opiši podlinearni algoritem tipa Monte Carlo, ki zadošča naslednji zahtevi: če je najmanj $n/2$ elementov iz A negativnih, potem algoritem vrne 'Več kot ena polovica je negativnih.'; če je največ $n/4$ elementov iz A negativnih, potem algoritem vrne 'Manj kot ena četrtina je negativnih'; sicer algoritem vrne karkoli, lahko tudi napačen odgovor. Verjetnost napake algoritma naj bo največ δ, kjer je $\delta < 1$ vhodni parameter.

1.22. Zanima nas naslednji odločitveni problem: ali sta dve podani multimnožici števil enaki. Določi polinom za vsako multimnožico na tak način, da sta polinoma enaka natanko tedaj, ko sta multimnožici enaki. Nato opiši algoritem tipa Monte Carlo, ki določi, ali sta dve podani multimnožici števil enaki.

1.23. Opazujemo ne nujno pošten kovanec. Ko ga mečemo, grb pade z verjetnostjo $p \in (0,1)$, kjer p ni znan. Radi bi ocenili vrednost p. Kovanec vržemo m krat in uporabljamo cenilko (*estimator*) $\tilde{p} = h/m$, kjer je h število grbov. Kakšen mora biti m, da velja $\Pr[|\tilde{p} - p| \leq \varepsilon p] \geq 1 - \delta$? (Vrednost m bo odvisna od δ, ε, p.) Kaj se dogaja, ko gre $p \to 0$?

1.24. Delamo naslednji poskus. Imamo n košar $B_1, B_2, \ldots, B_n$. Naslednji korak ponovimo k krat: novo žogo damo v košaro, ki jo izberemo enakomerno naključno. Naj bo X_i število žog v košari B_i na koncu. Naj bo $X_{min} = \min\{X_1, \ldots, X_n\}$. Določi vrednost $k = k(n)$, za katero velja

$$\Pr[X_{min} < 2 \ln n] \leq \frac{2}{n}.$$

Ali pri tako izbranem $k = k(n)$ velja $\mathbb{E}[X_{min}] = \Omega(\log n)$?

1.25. Naj bo $S_0 = \{1, \ldots, n\}$, kjer je $n = 2^k$ za neko naravno število k. Za vsak $i \in \mathbb{N}$ definiramo množico S_i takole: začnemo z $S_i = \emptyset$ in potem vsak element iz S_{i-1} neodvisno kopiramo v S_i z verjetnostjo $1/2$. Naj bo

$$Z = \sum_{i=0}^{k} |S_i|.$$

(a) Pokaži, da je za dovolj velik n verjetnost $\Pr[Z > 10n] < 1/100$.

(b) Naj bo $t = \min\big(\{i \in \mathbb{N} \mid S_i = \emptyset\} \cup \{n^3\}\big)$. Ali velja $\mathbb{E}[t] = O(\log n)$?

1.26. Pošteno igralno kocko s številkami $\{1, 2, \ldots, 6\}$ vržemo n krat. Naj bo X vsota števil, ki smo jih dobili na kocki. Vemo, da je $\mathbb{E}[X] = 7n/2$. Z uporabo meje Černova pokaži naslednjo trditev: za vsako konstanto $\delta > 0$ obstaja konstanta $c = c(\delta) > 0$, ki zadošča

$$\Pr\Big[X \in \big[\tfrac{7}{2}n - c\sqrt{n}, \tfrac{7}{2}n + c\sqrt{n}\big]\Big] > 1 - \delta.$$

1.27. Naj bodo $X_1, X_2, \ldots, X_n$ neodvisne slučajne spremenljvke z zalogo $\{-1, 1\}$. Ni nujno, da so enakomerne. Naj bo $X = \sum_{i=1}^{n} X_i$. Denimo, da velja $\mathbb{E}[X] = 0$. Pokaži mejo tipa Černova za $\Pr[|X| \geq t]$, ki velja za poljuben $t \in (0, n)$.

1.28. Naj bo A matrika velikosti $n \times n$ s koeficienti iz $\{0, 1\}$. Zanima nas vrednost

$$\phi(A) = \min\{||Ab||_\infty \mid b \in \{-1, 1\}^n\},$$

kjer je $||c||_\infty$ največja absolutna vrednost koeficientov vektorja c:

$$||(c_1\, c_2\, \cdots\, c_n)^T||_\infty = \max_{i=1\cdots n} |c_i|.$$

Pokaži, da velja $\phi(A) = O(\sqrt{n \log n})$. Namig: izberi naključni vektor $b \in \{-1, 1\}^n$ in uporabi mejo Černova iz vaje 1.27.

1.29. Opazujemo naslednji algoritem:

Algorithm BrezImena3
Input: Naravno število k
1. $x = 0$;
2. **for** $i = 1$ **to** k
3. **do** vržemo pošteno igralno kocko
4. **if** dobimo 1
5. **then** izberemo točko $(a, b) \in [0, 2] \times [0, 2]$ enakomerno naključno;
6. **if** $a^2 < b$
7. **then** $x = x + 1$
8. **return** $rez = x/k$

(a) Določi $\mathbb{E}[rez]$.

(b) Izračunaj vrednost parametra k, za katerega velja $\Pr\left\{|rez - \mathbb{E}[rez]| < 10^{-5}\right\} \geq 0.99$.

(c) Kaj bi spremenili v algoritmu, če želimo $\mathbb{E}[rez] = \ln 2$? Namig: $\int \frac{1}{x} dx = \ln x$.

1.30. Oznaka $[n]$ naj pomeni $\{1, \ldots n\}$. Predpostavimo, da podprogram Random(k) vrne naključno celo število med 1 in k enakomerno. Opazujemo naslednji algoritem, ki vrne *injektivno* funkcijo $f : [n] \to [k]$ pri nekem $k \geq n$.

Algorithm Vaja1
Input: n, k, pri čemer je $n \leq k$
1. **for** $i = 1, 2, \ldots, n$
2. **do** $f(i)$:=Random(k);
3. **while** $f(i) \in \{f(j) \mid j < i\}$;
4. **do** $f(i)$:=Random(k);

Naj bo X slučajna spremenljvka, ki pove, kolikokrat smo klicali Random(k).

(a) Če je $k = n$, koliko je $\mathbb{E}[X]$?

(b) Kako je $\mathbb{E}[X]$ odvisno od parametra k?

(c) Pokaži, da je v primeru $k = 2n$, verjetnost $\Pr[X > 2 \cdot \mathbb{E}[X]]$ zelo zelo majhna.

1.31. Imamo n nalog, ki jih moramo razdeliti med m strojev, pri čemer m deli n. Posamezen stroj opravi posamezno nalogo v eni uri z verjetnostjo p oziroma v k urah z verjetnostjo $1-p$. Vsak stroj dobi naključno podmnožico n/m nalog. Naj C označuje čas potreben za dokončanje vseh nalog. Z uporabo meje Černova pokaži zgornjo in spodnjo mejo vrednosti slučajne spremenljevke C (z veliko verjetnostjo).

1.32. Graf G z n točkami konstruiramo naključno na naslednji način: za vsak par točk u, v grafa dodamo povezavo uv s verjetnostjo $p = p(n)$. Za katere vrednosti p je število pričakovanih 5-klik enako 1? Ali je, pri tako izbranem p, število povezav naključnega grafa G skoncentrirano (*concentrated*) okoli matematična upanja?

1.33. Imamo 10 košar in n žog. Kot običajno vržemo žogo v naključno košaro, pri čemer ima vsaka košara isto verjetnost. Naj bo X število žog v košari, ki ima največje število žog. Naj bo Y število žog v košari, ki ima najmanjšje število žog. Pokaži, da za vsak $\varepsilon > 0$ obstaja vrednost $c = c(\varepsilon) > 0$, ki zadošča

$$\Pr\left[X - Y \geq c\sqrt{n}\,\right] \leq \varepsilon.$$

(Namig: opazuj vsak par košar posebej.)

1.34. Podana je ogromna multimnožica S, ki vsebuje n števil. Poiskati želimo približek mediane S. Pravimo, da je število x *ε-aproksimacija mediane* za multimnožico S, če imata $\{s \in S \mid s \leq x\}$ in $\{s \in S \mid s \geq x\}$ moč najmanj $(1/2 - \varepsilon)n$. Denimo, da je ε pozitiven in dovolj majhen.

Opazujemo naslednji algoritem. Podmnožico $R \subseteq S$ konstruiramo takole: k krat izberemo naključen element množice S in ga dodamo v R. (R je multimnožica in lahko posamezen element vsebuje večkrat.) Nato izračunamo mediano m multimnožice R in vrnemo m. Število k moramo izbrati tako, da pravilno najdemo ε-aproksimacijo mediane.

(a) Naj bo x_ℓ element iz S, ki je v položaju $(1/2-\varepsilon)n$, ko je S urejena od najmanjšega do največjega elementa (*increasingly*). Naj bo X število elementov množice R, ki so manjši kot x_ℓ. Pokaži, da velja $\Pr[X > k/2] \leq 1/200$ za $k = \Theta(\varepsilon^{-2})$.

(b) Določi natančno vrednost parametra k, za katerega algoritem vrne (0.05)-aproksimacijo mediane za S z verjetnostjo najmanj 0.99.

(c) Za podane parametre $\delta \in (0, 1)$ in ε izračunaj vrednost $k = k(\varepsilon, \delta)$ za to, da vrne algoritem ε-aproksimacijo mediane za S z verjetnostjo najmanj $1 - \delta$.

1.35. Podana je ogromna množica S, ki vsebuje n različnih elementov. Zanima nas konstrukcija naključne podmnožice $R \subset S$, ki vsebuje natančno $\sqrt{n}$ elementov. Privzameš lahko, da je $\sqrt{n}$ celo število.

Eksperimentiramo s naslednjim algoritmom, ki ima dva dela. Prvi del poteka takole: začnemo s prazno množico R in nato vsak element $s \in S$ neodvisno dodamo v R s verjetnostjo $n^{-1/2} + cn^{-3/4}$, kjer je c ustrezno izbrana konstanta. V drugem delu vrne algoritem 'Napako', če ima R manj kot $\sqrt{n}$ elementov. Drugače pa algoritem odstrani $|R| - \sqrt{n}$ naključnih elementov iz R in vrne rezultat.

(a) Kakšna je pričakovana moč množice R na koncu prvega dela?

(b) Določi konstanto c, za katero je verjetnost napake v algoritmu manjša kot $1/100$.

(c) Predpostavimo, da algoritem ne vrne 'Napake'. Pokaži, da je število elementov, ki jih algoritem na drugem koraku odstrani, z veliko verjetnostjo $O(n^{1/4}\sqrt{\log n})$.

(d) Predpostavimo, da algoritem ne vrne 'Napake'. Pokaži, da je pričakovano število elementov, ki jih algoritem na drugem koraku odstrani, $O(n^{1/4}\sqrt{\log n})$.

1.36. Na trgu je nova zbirka n različnih znamk. Novo znamko dobimo, ko kupimo kuverto. Vsaka kuverta vsebuje znamko, ki je izbrana enakomerno naključno med znamkami v kompletu. V vaji 1.2 smo videli, da je pričakovano število kupljenih kuvert do dokončanja kompleta $\Theta(n \log n)$.

Pričakujemo, da bo v prihodnosti vsak komplet znamk zelo dragocen. Torej se odločimo, da bomo kupili dovolj kuvert, da dobimo veliko kompletov znamk. Pokaži, da z veliko verjetnostjo lahko dobimo $100 \log n$ kompletov, če kupimo $10^5 n \log n$ kuvert.

1.37. Imamo Markovsko verigo s stanji $\{1, 2, 3, 4\}$ in prehodno matriko

$$P = \begin{bmatrix} 0 & 3/10 & 1/10 & 3/5 \\ 1/10 & 1/10 & 7/10 & 1/10 \\ 1/10 & 7/10 & 1/10 & 1/10 \\ 9/10 & 1/10 & 0 & 0 \end{bmatrix}.$$

- Izračunaj stacionarno porazdelitev.
- Izračunaj verjetnost, da imamo stanje 4 po 32 korakih, če smo začeli v stanju 1.
- Izračunaj verjetnost, da imamo stanje 4 po 128 korakih, če je stanje na začetku izbrano naključno enakomerno.

1.38. k-barvanje grafa G je preslikava $b : V(G) \to \{1, \ldots, k\}$. Graf G je k-obarvljiv, če obstaja takšno k-barvanje, da nobena povezava ni enobarvna. Graf G je $(\triangle, k)$-obarvljiv, če obstaja takšno k-barvanje, da noben trikotnik (3-klika) ni enobarven. Naj bo G 3-obarvljiv graf.

(a) Pokaži, da je graf G $(\triangle, 2)$-obarvljiv.

(b) Opazuj naslednji algoritem, ki vrne 2-barvanje grafa G, ki zadošča, da noben trikotnik (3-klika) ni enobarven: dokler obstaja enobarven trikotnik v G, algoritem izbere enobarven trikotnik in spremeni barvo naključne točke takega trikotnika. Določi zgornjo mejo za pričakovano število korakov algoritma.

1.39. Matrika P je dvojno stohastična matrika (*doubly stochastic*), če je vsota koeficientov vsake vrstice in vsakega stolpca enaka 1. Imamo Markovsko verigo, pri čemer je prehodna matrika dvojno stohastična. Pokaži, da je enakomerna porazdelitev stacionanarna.

1.40. Imamo Markovsko verigo z n stanji, stacionarno porazdelitvjo $\bar{\pi} = (\pi_1, \ldots, \pi_n)$ in prehodno matriko P. Recimo, da začnemo verigo v času 0 in naredimo m korakov. Naj bo $X_0, X_1, X_2, \ldots, X_m$ zaporedje stanj, ki ga dobimo. Zanima nas obrnjeno zaporedje stanj $X_m, X_{m-1}, \ldots, X_0$.

- Za podano stanje X_{k+1} pokaži, da je stanje X_k neodvisno od $X_{k+2}, X_{k+3}, \ldots, X_m$.
- Pokaži, da zadošča prehodna matrika obrnjenega zaporedja Q enakosti $Q_{i,j} = \pi_j P_{j,i}/\pi_i$.

1.41. Imamo Markovsko verigo na stanjih $\{1, \ldots, n\}$. Za prehodno matriko P za $i < n$ velja $P_{i,i+1} = 1/2$ in $P_{i,1} = 1/2$, pa tudi $P_{n,n} = 1/2$ in $P_{n,1} = 1/2$. Izračunaj stacionarno porazdelitev.

1.42. Imamo 2 košari (L in D) in n žog, ki so porazdeljene med L in D. Na vsakem časovnem koraku izberemo enakomerno naključno žogo in jo damo v drugo košaro. To je, če je izbrana žoga v L, potem jo damo v D, in obratno. Naj bo X_t število žog v košari L na časovnem koraku t. $X_1, X_2, \ldots$ je Markovska veriga.

(a) Opiši prehodno matriko verige.

(b) Poišči stacionarno porazdelitev verige. (Namig: če razumeš postopek, lahko odgovor uganiš in dokažeš njegovo pravilnost. V rešitvi nastopajo binomski koeficienti $\binom{n}{j}$.)

1.43. Imamo košaro, ki vsebuje n žog. Nekatere izmed žog so bele, druge so črne. Na posameznem koraku naredimo naslednje: izberemo naključno žogo iz košare in jo zamenjamo z novo belo žogo z verjetnostjo $2/3$ in s črno žogo z verjetnostjo $1/3$. Košara ima vedno n žog. Naj bo X_t število **belih** žog v košari na koncu koraka t. Zaporedje $X_0, X_1, X_2, \ldots$ je Markovska veriga, kjer je X_0 število belih žog na začetku.

(a) Napiši prehodno matriko Markovske verige.

(b) Pokaži, da je $\pi_i = \binom{n}{i} \frac{2^i}{3^n}$ stacionarna porazdelitev verige. (Namig: koliko je $(1+2)^n$?)

(c) Zakaj lahko porazdelitev opisano v točki (b) kar uganemo?

(d) Predpostavimo, da je $n = 3$ in da so v začetku vse žoge črne. Določi zgornjo mejo za pričakovano vrednost slučajne spremenljvke $\min\{t \mid X_t = 3\}$.

1.44. Imamo 3 košare K_0, K_1, K_2. V košari K_0 sta 2 žogi. Košari K_1 in K_2 sta prazni. Na vsakem koraku izberemo eno naključno žogo. Če jo izberemo iz košare K_i, jo damo v košaro $K_{i+1 \bmod 3}$. Stanje na koncu koraka t opišemo s trojko $X_t = (a_0, a_1, a_2) \in \{0, 1, 2\}^3$, ki zadošča $a_0 + a_1 + a_2 = 2$. Začetno stanje je $X_0 = (2, 0, 0)$. Zaporedje $X_0, X_1, X_2, \ldots$ je Markovska veriga.

(a) Napiši prehodno matriko Markovske verige.

(b) Izračunaj stacionarno porazdelitev verige.

(c) Naj bo Y_t število žog v košari K_0 na koncu koraka t, in naj bo $Z_t = \sum_{j=0}^{j=t} Y_j$. Izračunaj vrednost $\lim_{t\to\infty} (Z_t/t)$.

1.45. V povezanem grafu G naredimo naključen obhod.

(a) Pokaži, da je stacionarna porazdelitev enaka

$$\pi_v = \frac{\deg(v)}{2|E(G)|} \quad \text{za vsako točko } v \in V(G).$$

(b) Pokaži, da za vsako točko $v \in V(G)$ velja

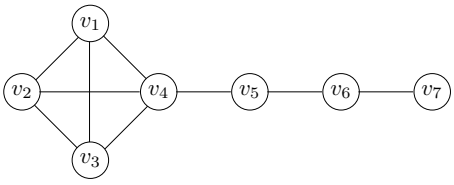
$$h_{v,v} = \frac{2|E(G)|}{\deg(v)}.$$

(c) Pokaži, da za vsako povezavo $uv \in E(G)$ velja $h_{v,u} < 2|E(G)|$. (Namig: izračunaj $h_{v,v}$ z uporabo sosedov točke v.)

(c) Pokaži, da je čas pokritja (*cover time*) (pričakovani čas do obiska vseh točk) največ $4|V(G)| \cdot |E(G)|$. (Namig: uporabi vpeto drevo.)

1.46. Naj bo G_n naslednji graf:

- $V(G_n) = \{v_1, v_2, \ldots, v_{2n-1}\}$,
- točke $v_1, \ldots, v_n$ inducirajo poln podgraf (so vse paroma sosede),
- točke $v_n v_{n+1} v_{n+2} \ldots v_{2n-1}$ inducirajo pot, drugih povezav ni.



Na sliki je graf G_4. Opazujemo naključni obhod v grafu G_n, kjer se na vsakem koraku premaknemo v enakomerno izbranega naključnega soseda.

(a) Kakšna je stacionarna porazdelitev?

(b) Naj bo Z število korakov potrebnih za obisk vseh točk, če začnemo obhod v točki v_{2n-1}. Pokaži, da je $\mathbb{E}[Z] = O(n^2)$.

(c) Naj bo Y_t slučajna spremenljvka, ki pove indeks točke na koraku t. Na primer, če je na desetem koraku obhod v točki v_4, potem je $Y_{10} = 4$. Naj bo $Z_t = \sum_{j=0}^{t} Y_t$. Izračunaj vrednost $\lim_{t\to\infty} (Z_t/t)$.

1.47. Muca in miška neodvisno in naključno hodita v grafu G, ki je povezan in ni dvodelen (*bipartite*). Obhoda začneta v istem času v različnih točkah in naredita en korak v vsakem časovnem koraku. Kot si predstavljate, muca poje miško, če se v istem časovnem trenutku srečata v isti točki. Dokaži zgornjo mejo $O(|E(G)|^2 \cdot |V(G)|)$ za pričakovano število korakov do trenutka, ko poje muca miško. (Namig: poglej Markovsko verigo na stanjih (a, b), kjer je a položaj muce in b položaj miške, in uporabi vajo 1.45.)

2 Aproksimacijski algoritmi in $\#P$

2.1. Zanima nas problem TSP v ravnini z evklidsko razdaljo: vhod je množica točk P, cena povezave uv pa je enaka evklidski razdalji med točkama u in v. Opazujemo naslednji algoritem: če so $p_1, p_2, \ldots, p_n$ točke vhoda urejene po x-koordinatah (predpostavimo da so vse x-koordinate različne), algoritem vrne obhod

$$p_1 \to p_2 \to p_3 \to \cdots \to p_n \to p_1.$$

Pokaži, da tak algoritem ni k-aproksimacijski algoritem za nobeno konstanto k.

2.2. Barvanje grafa G je preslikava $b : V(G) \to \mathbb{N}$, ki zadošča $b(u) \neq b(v)$ za vsako povezavo $uv \in E(G)$. Graf želimo obarvati z najmanjšim možnim številom barv, to je, minimizirati moč množice $b(V(G))$. Uporabljamo naslednji požrešni algoritem. Naj bodo $v_1, v_2, \ldots, v_n$ točke grafa G. Za vsak $i = 1, \ldots, n$ naj bo G_i podgraf grafa G induciran na točkah $v_1, v_2, \ldots, v_i$. Za vsak $i = 1, \ldots, n$ iterativno definiramo barvo točke v_i kot

$$b(v_i) = \min\{n \in \mathbb{N} \mid b(\cdot) \text{ je barvanje grafa } G_i\}.$$

Pokaži, da tak algoritem ni k-aproksimacijski algoritem za nobeno konstanto k.

2.3. Za vsak $k \in \mathbb{N}$ definiramo naslednji optimizacijski problem:

> k-LargestCover
> Vhod: končna množica S in družina podmnožic $\mathcal{F} \subseteq 2^S$, kjer je vsaka podmnožica $X \in \mathcal{F}$ moči k.
> Naloga: maksimiziraj $|\mathcal{G}|$ pri pogojih: $\mathcal{G} \subseteq \mathcal{F}$ in podmnožice poddružine $\mathcal{G}$ so paroma disjunktne.

(a) Kakšen problem je 2-LargestCover?

(b) Opiši k-aproksimacijski algoritem za problem k-LargestCover.

2.4. Zanima nas naslednji optimizacijski problem:

> BinPacking
> Vhod: Seznam naravnih števil $a_1, \ldots, a_n \in \mathbb{N}$ in vrednost $B \in \mathbb{N}$. Za vsak i naj velja $a_i \leq B$.
> Naloga: minimiziraj vrednost k pri pogojih: $I_1, I_2, \ldots, I_k \subseteq [n]$, $[n]$ je disjunktna unija $I_1, \ldots, I_k$ in za vsak $j = 1, \ldots, k$ naj velja $\sum_{i \in I_j} a_i \leq B$.

Elemente velikosti $a_1, \ldots, a_n$ razdelimo elementov na košare velikosti B in pri tem želimo minimizirati število košar, ki jih rabimo.

(a) Predpostavimo, da P≠NP. Pokaži, da za ta problem ne obstaja α-aproksimacijski algoritem za nobeno konstanto $\alpha \in (1, 3/2)$.

(b) Uporabljamo algoritem s strategijo *first-fit*: na koraku i ($i = 1, \ldots k$) dodamo indeks i v prvo skupino I_j, ki ima dovolj prostora za a_i. To je, indeks i je dodan v množico I_j, če trenutno velja

$$j = \min\{j' \mid \sum_{\ell \in I_{j'}} a_\ell \leq B - a_i\}.$$

Pokaži, da je tak algoritem 2-aproksimacijski.

2.5. Zanima nas naslednji optimizacijski problem:

SeveralValuableBins
Vhod: n elementov vrednosti $a_1, a_2, \ldots, a_n \in \mathbb{N}$ in število $B \in \mathbb{N}$. Velja $\sum_i a_i \geq B$.
Naloga: naredi čim večje število skupin elementov, kjer je vsota vrednosti vsake skupine najmanj B.

(a) Prevedi problem SeveralValuableBins na ILP.

(b) Opiši 2-aproksimacijski algoritem za problem SeveralValuableBins.

(c) Pokaži: če obstaja (5/3)-aproksimacijski algoritem za SeveralValuableBins, potem je P=NP.

2.6. V dvodelnem grafu $G = (U, V, E)$ naj E označuje njegovo množico povezav, U in V pa naj bosta oba barvna razreda njegovih točk. Za $k \in \mathbb{N}$ definiramo naslednji optimizacijski problem:

k-CompatibleMatching
Vhod: Dvodelni grafi $G_1 = (U, V, E_1)$, $G_2 = (U, V, E_2)$, $\ldots$, $G_k = (U, V, E_k)$ z istimi točkami.
Naloga: maksimiziraj $|F|$ pri pogojih: $F \subseteq E_1 \cup \cdots \cup E_k$ in pri čemer je za vsak $i = 1, \ldots k$ množica povezav $F \cap E_i$ prirejanje.

(a) Opiši k-aproksimacijski algoritem za problem k-CompatibleMatching.

(b) Opiši (3/2)-aproksimacijski algoritem za problem 2-CompatibleMatching.

2.7. V dvodelnem grafu $G = (U, V, E)$ naj E označuje njegovo množico povezav, U in V pa naj bosta oba barvna razreda njegovih točk. Za $k \in \mathbb{N}$ definiramo naslednji optimizacijski problem:

k-CompatibleMaxDegree2
Vhod: Dvodelni grafi $G_1 = (U, V, E_1)$, $G_2 = (U, V, E_2)$, $\ldots$, $G_k = (U, V, E_k)$ z istimi točkami.
Naloga: maksimiziraj $|F|$ pri pogojih: $F \subseteq E_1 \cup \cdots \cup E_k$ pri čemer ima za vsak $i = 1, \ldots k$ graf $(U, V, E_i \cap F)$ maksimalno stopnjo največ 2.

(a) Prevedi problem k-CompatibleMaxDegree2 na ILP.

(b) Opiši $(2k)$-aproksimacijski algoritem za problem k-CompatibleMaxDegree2.

2.8. Zanima nas problem (+1)-TSP. Vhod problema (+1)-TSP je vhod navadnega problema TSP, kjer imajo cene/razdalje naslednje lastnosti: $d_{ij} = d_{ji}$ za vsak i, j (simetrija), $d_{ij} \leq 1 + d_{ik} + d_{kj}$ za vsak i, j, k (približna trikotniška neenakost), in $d_{ij} \in \mathbb{N}$ za vsak i, j (cene so naravna števila). Iščemo aproksimacijski algoritem za problem (+1)-TSP. Uporabljamo 2-aproksimacijski algoritem za metrični TSP (ali $\triangle$-TSP), ki ga smo spoznali na predavanjih. Ni jasno in mogoče ni res, da je tak algoritem 2-aproksimacijski za (+1)-TSP. Ali je tak algoritem k-aproksimacijski za kakšno konstanto k?

2.9. Zanima nas metrični TSP ($\triangle$-TSP), kjer je d_{ij} cena/razdalja med točko i in točko j. Naslednji algoritem je znan kot Christofidesov postopek:

- poiščemo minimalno vpeto drevo T v polnem grafu K_n z cenami d_{ij};
- z *Odd* označimo množico točk, ki imajo v drevesu T liho stopnjo;
- konstruiramo podgraf H grafa K_n induciran na množici *Odd*, z istimi cenami na povezavah d_{ij};
- v H poiščemo popolno prirejanje M z minimalno vsoto cen;
- konstruiramo Eulerjev obhod $W = v_1v_2v_3\cdots v_{2n}$ v grafu $T + M$;
- v obhodu $W = v_1v_2v_3\cdots v_{2n}$ odstranimo vse večkratne ponovitve točk. Naj bo Ω obhod, ki ga dobimo. (Na primer, iz obhoda $W = v_1v_2v_3v_2v_1v_4v_2v_1$ pridelamo obhod $\Omega = v_1v_2v_3v_4v_1$.)
- vrnemo Ω.

Vsak korak algoritma porabi polinomski čas. Ali je res, da vrne algoritem dopustno rešitev? Zakaj? Dokaži, da je Christofidesov algoritem (3/2)-aproksimacijski. (Namig: kaj lahko rečemo o dolžini prirejanja M?)

2.10. Na predavanjih smo spoznali 2-aproksimacijski algoritem za metrični TSP ($\triangle$-TSP). Pokaži, da algoritem ni $(2-\varepsilon)$-aproksimacijski za nobeno konstanto $\varepsilon > 0$. (Namig: poišči protiprimer s cenami 1 in 2.)

2.11. Zanima nas naslednji problem razporeda (*scheduling*). Imamo n nalog in m identičnih strojev $M_1, \ldots, M_m$. Vsak izmed strojev za nalogo j rabi čas $t_j > 0$. Strojem želimo dodeliti naloge na tak način, da minimiziramo čas končanja zadnjega stroja. To je, če je $A(i)$ je množica nalog stroja M_i, potem stroj M_i konča v času

$$T_i = \sum_{j \in A(i)} t_j,$$

minimizirati pa želimo vrednost $\max\{T_1, T_2, \ldots, T_m\}$.

(a) Opazujmo naslednji požrešni algoritem: za vsak $j = 1, \ldots, n$, nalogo j dodelimo stroju, ki bi trenutno dodelitev nalog $1, \ldots, (j-1)$ končal najhitreje. Pokaži, da je tak algoritem 2-aproksimacijski.

(b) Pokaži, da je algoritem točke (a) $(2 - 1/m)$-approksimacijski.

(c) Pokaži, da je aproksimacijski faktor $(2 - 1/m)$ tesen za algoritem točke (a). To je, dokaži, da obstaja vhod problema, za katero algoritem vrne dodelitev, ki porabi vsaj $(2-1/m)$ krat več časa kot optimalna dodelitev. (Namig: poskusi $n = (m(m-1)+1$ nalog.)

(d) Recimo, da je strojev 10, za vsako nalogo naj velja $1 \leq t_i \leq 50$, vsota trajanja vseh nalog pa naj bo najmanj 3000. Pokaži, da je algoritem točke (a) potem (7/6)-aproksimacijski.

(e) Opazujmo alternativni algoritem, ki je znan kot *ordered scheduling algorithm*. Najprej uredimo naloge glede časa izvajanja od najdaljšega do najkrajšega. Nato uporabimo požrešni algoritem odstavke (a) s to urejenostjo. Pokaži, da je tak algoritem (3/2)-aproksimacijski. (V resnici je tak algoritem (4/3)-aproksimacijski, dokaz pa je precej težji.)

2.12. Zanima nas problem 0/1-Nahrbtnik:

> 0/1-Nahrbtnik
> Vhod: elementi $e_1, e_2, \ldots e_n$, kjer ima posamezen element e_i ceno $c_i \in \mathbb{N}$ in velikost $v_i \in \mathbb{N}$. Velikost nahrbtnika V.
> Naloga: maksimiziraj $\sum_{i \in I} c_i$ pri pogojih $I \subset \{1, \ldots, n\}$ in $\sum_{i \in I} v_i \leq V$.

Predpostavimo, da je $v_i \leq V$ za vsak i in da je $\sum_i v_i > V$. Predpostavimo, da so elementi urejeni po relativni ceni glede na velikost, naj za vsak i velja $c_i/v_i \geq c_{i+1}/v_{i+1}$.

(a) Uporabljamo naslednji požrešni algoritem: na koraku i damo element e_i v nahrbtnik če je prostora še dovolj. Pokaži, da tak algoritem ni α-aproksimacijski algoritem za nobeno konstanto α.

(b) Uporabljamo naslednji algoritem. Poiščemo prvi indeks k, ki zadošča $\sum_{i=1}^{k} v_i > V$, in potem izberemo najboljšo rešitev med $I_1 = \{1, \ldots, k-1\}$ in $I_2 = \{k\}$. Pokaži, da je tak algoritem 2-aproksimacijski. (Namig: Pokaži, da je vrednost optimalne rešitve največ $\sum_{i=1}^{k} c_i$.)

2.13. Zanima nas problem k-MultiwayCut:

> k-MultiwayCut
> Vhod: graf G in točke $s_1, s_2, \ldots, s_k \in V(G)$.
> Naloga: minimiziraj $|A|$ pri pogojih: $A \subseteq E(G)$ in graf $G - A$ nima nobenih poti med s_i in s_j, za vsak i, j, $i \neq j$.

Opazujemo naslednji algoritem. Za vsak s_i izračunamo množico povezav B_i, ki ima najmanjšo moč in loči s_i od $\{s_1, \ldots, s_k\} \setminus s_i$. To je, $G - B_i$ nima poti med s_i in s_j za noben $s_j \neq s_i$. Vsak B_i lahko izračunamo v polinomskem času; za nalogo ni pomembno, kako to naredimo. Algoritem vrne $\bigcup_i B_i$.

(a) Pokaži, da je tak algoritem 2-aproksimacijski.

(b) Pokaži, da tak algoritem ni 9/5-aproksimacijski.

(c) Opiši (3/2)-aproksimacijski algoritem za problem 3-MultiwayCut.

2.14. Opiši 6-aproksimacijski algoritem za problem konstrukcije največje neodvisne množice v *ravniskih* grafih. (Množica točk U grafa G je neodvisna, če za vsaki točki $u, v \in U$ velja $uv \notin E(G)$.)

2.15. Zanima nas problem VertexCover: dan je graf G, iščemo pa najmanjšo množico točk grafa, ki pokrijejo vsako povezavo grafa. Nekdo priporoča naslednji algoritem: naj bo T vpeto drevo pregleda grafa G v globino. Vrnemo množico točk S, ki niso listi v T. Pokaži, da je priporočeni algoritem 2-aproksimacijski algoritem za VertexCover. (Namig: najdi veliko prirejanje v grafu G.)

2.16. V tej nalogi imajo vsi kvadrati stranice vzporedne koordinatnima osema. Naj bo R množica kvadratov v ravnini, kjer ima vsak kvadrat ploščino največ 1. Predpostavimo, da je vsota ploščin kvadratov množice R vsaj 1. *Pakiranje* je takšna podmnožica kvadratov $S \subset R$ skupaj s položaji kvadratov iz S, da so vsi kvadrati znotraj kvadrata $[0, 1]^2$ in so notranjosti kvadratov paroma disjunktne. Zanima nas pakiranje, ki maksimizira pokrito

ploščino. Opiši 4-aproksimacijski algoritem za tak problem. (Namig: opiši algoritem, ki vrne pakiranje, ki pokrije ploščino najmanj 1/4.)

2.17. Zanima nas optimizacijski problem k-Covering:

> k-Covering
> Vhod: končna množica realnih števil $X = \{x_1, x_2, \ldots x_n\} \subset \mathbb{R}$.
> Naloga: maksimiziraj $|(I_1 \cup I_2 \cup \ldots I_k) \cap X|$, kjer so $I_1, I_2, \ldots, I_k$ enotski intervali.

Uporabljamo naslednji požrešni algoritem: na koraku i ($i = 1 \ldots k$) izberemo enotski interval I_i, ki maksimizira $|I_i \cap (X \setminus (I_1 \cup \cdots \cup I_{i-1})|$.

(a) Pokaži, da tak algoritem ne rešuje problema k-Covering optimalno.

(b) Pokaži, da je tak algoritem (4/3)-aproksimacijski, ko je $k = 2$.

(c) Pokaži, da je tak algoritem (27/19)-aproksimacijski, ko je $k = 3$.

2.18. Za vsako točko $x \in \mathbb{R}^2$ in vsako končno množico točk $Q \subset \mathbb{R}^2$ definiramo $d(x, Q) = \min_{q \in Q} d(x, q)$. Optimizacijski problem Center je definiran takole:

> Center
> Vhod: končna množica $P \subset \mathbb{R}^2$ in naravno število k.
> Naloga: izberi $Q \subseteq P$ moči k, ki minimizira $\max_{p \in P} d(p, Q)$.

Naloga razumemo na naslednji način: minimizirati želimo polmer k diskov, ki imajo središča v točkah množice P in pokrijejo P. Problem Center je NP-težak.

Opazujemo naslednji algoritem za center:

Algorithm Greedy
Input: P, k (denimo $|P| < k$)
1. izberemo poljubno točko q_1 iz P;
2. $Q = \{q_1\}$;
3. **for** $i = 2$ **to** k
4. **do** naj bo q_i točka, ki maksimizira $d(q, Q)$ med točkami $q \in P$;
5. $Q = Q \cup \{q_i\}$;
6. **return** Q;

(a) Pokaži, da je algoritem Greedy 2-aproksimacijski algoritem.

(b) Pokaži, da algoritem Greedy ni (1.9)-aproksimacijski algoritem za $k = 4$.

2.19. Zanima nas problem LongestPath: podan je graf G, iščemo pa najdaljšo pot v G, ki je seveda brez ponovljenih točk. Predpostavimo, da P≠NP. Pokaži, da za problem LongestPath ne obstaja FPTAS.

2.20. Naj bo $G = (V, E)$ poljuben graf. Za vsako celo število $\ell \geq 1$ definiramo graf $G^{(\ell)} = (V^{(\ell)}, E^{(\ell)})$ na naslednji način: $V^{(\ell)}$ je množica ℓ-teric točk iz V, dve ℓ-terici $(v_1, v_2, \ldots, v_\ell)$ in $(w_1, w_2, \ldots, w_\ell)$ pa sta sosedi natanko tedaj, ko je za vsak i točka v_i bodisi soseda točke w_i ali pa sta v_i in w_i enaki. Preveri, da je $G^{(1)} = G$.

Zanima nas optimizacijski problem iskanja največje klike v podanem grafu, ki ga imenujemo Max-Klik.

(a) Naj bo M_ℓ moč največje klike v grafu $G^{(\ell)}$. Pokaži, da velja $M_\ell = (M_1)^\ell$.

(b) Pokaži naslednjo trditev: če obstaja k-aproksimacijski algoritem za problem Max-Klik za neko konstanto k, potem obstaja FPTAS za problem Max-Klik.

(c) Predpostavimo, da P$\neq$ NP. Pokaži, da ne obstaja k-aproksimacijski algoritem za problem Max-Klik za nobeno konstanto k.

2.21. Naj bo P množica n točk v ravnini. *Steinerjevo drevo* T za P je drevo, ki zadošča $V(T) \subset \mathbb{R}^2$ in $P \subseteq V(T)$. 'Dodatne' točke $V(T) \setminus P$ imenujemo Steinerjeve točke. Dolžina povezave uv je dolžina daljice $\overline{uv}$, in dolžina drevesa je vsota dolžine povezav. Naj bo $T_{Ste}(P)$ Steinerjevo drevo za P z minimalno dolžino. Problem iskanja $T_{Ste}(P)$ je NP-težak.

Točke na ravnine definirajo 'navadno' vpeto drevo $T_{min}(P)$: opazujmo poln graf z množico točk P, kjer je cena povezave uv dolžina daljice $\overline{uv}$, in naj bo $T_{min}(P)$ minimalno vpeto drevo tega grafa.

(a) Pokaži, da je $T_{Ste}(P)$ lahko strogo krajši kot $T_{min}(P)$.

(b) Pokaži, da je algoritem, ki izračuna $T_{min}(P)$ in ga vrne, 2-aproksimaciski algoritem za problem iskanja $T_{Ste}(P)$.

2.22. Naj bo P množica n točk v ravnini. Diameter $\Delta(P)$ množice P je definiran kot

$$\Delta(P) = \max_{p,q \in P, p \neq q} |pq|,$$

kjer je $|pq|$ dolžina daljice $\overline{pq}$. Diameter $\Delta(P)$ lahko izračunamo v času $O(n^2)$. Zanima nas FPTAS, ki porabi linearni čas za vsako konstanto ε.

- Opiši linearni 2-aproksimacijski algoritem.
- Opiši $(1+\varepsilon)$-aproksimacijski algoritem, ki porabi linearni čas za vsako konstanto $\varepsilon > 0$. Lahko predpostaviš, da imaš na voljo funkcijo celi del ($\lfloor \cdot \rfloor$). (Namig: uporabi prvo točko in mrežo z velikostjo odvisno od ε.)

2.23. Opiši 2-aproksimacijski algoritem za naslednji problem:

LinearOrdering
Vhod: usmerjen graf $G = (V, A)$.
Naloga: poišči bijektivno preslikavo $\pi : V \to \{1, \cdots, |V|\}$, ki maksimizira moč množice

$$\{\overrightarrow{uv} \in A \mid \pi(u) < \pi(v)\}.$$

2.24. Problem pod drobnogledom je barvanje grafov na n točkah.

(a) Pokaži, da je vsak graf z maksimalno stopnjo Δ tudi $(\Delta+1)$-obarvljiv. Opiši algoritem, ki v polinomskem času vrne $(\Delta+1)$-barvanje.

(b) Opiši algoritem, ki reši naslednjo nalogo v polinomskem času: vhod je graf G, ki je 3-obarvljiv. Poiskati želimo $O(\sqrt{n})$-barvanje grafa G. (Namig: Če je nek graf 2-obarvljiv, ga lahko 2-pobarvamo v polinomskem času.)

2.25. Zanima nas naslednji problem razporeda (*scheduling*). Opraviti moramo n nalog, na voljo pa imamo 2 identična stroja M_1, M_2. Naloga j porabi čas $t_j > 0$, kjer je t_j celo število. Strojema želimo dodeliti naloge na tak način, da minimiziramo čas dokončanja zadnjega stroja. To je, če je $A(i)$ je množica nalog stroja M_i, potem konča stroj M_i v času

$$T_i = \sum_{j \in A(i)} t_j,$$

minimizirati pa želimo vrednost $\max\{T_1, T_2\}$.

(a) Opiši algoritem, ki uporablja dinamično programiranje in najde optimalno rešitev v času $O(n(T^*)^2)$, kjer je T^* čas optimalne rešitve.

(b) Opiši FPTAS za problem.

2.26. Zanima nas naslednji problem razporeda (*scheduling*). Opraviti moramo n nalog, na voljo imamo tri identične stroje M_1, M_2, M_3. Naloga j porabi čas $t_j > 0$, kjer je t_j celo število. Če je $A(i)$ je množica nalog stroja M_i, potem konča stroj M_i v času

$$T_i = \sum_{j \in A(i)} t_j.$$

Želimo minimizirati vrednost $T_1^2 + T_2^2 + T_3^2$.

(a) Opiši algoritem, ki uporablja dinamično programiranje in najde optimalno rešitev v psevdopolinomskem času.

(b) Opiši FPTAS za problem.

(c) Opiši PTAS, ki uporablja strukturiranje izhoda.

2.27. Gledamo problem 0/1-KnapsackWithTaxes, ki je definiran takole. Imamo množico X z n elementi. Vsak element $x \in X$ ima velikost $v(x)$ in ceno $c(x)$. Velikosti in cene so cela števila. Tudi imamo nahrbtnik s prostornino W. Nekatere elemente X pospravimo v nahrbtnik in gremo na pot v drugo državo. Na žalost carina na meji ne dela kot bi morala, in bo zaplenila element iz nahrbtnika, ki ima največjo ceno. Maksimizirati želimo vsoto vrednosti elementov, ki bodo ostali v nahrbnitku. Bolj formalno:

$$\begin{array}{ll} \max & \sum_{x \in Y} c(x) - \max_{x \in Y} c(x) \\ \text{p.p.} & Y \subseteq X \\ & \sum_{x \in Y} v(x) \leq W \end{array}$$

Opiši FPTAS za problem 0/1-KnapsackWithTaxes.

2.28. Naj bo G_n multigraf na točkah $\{1, 2, \ldots, n\}$, ki ima dve povezavi e_i, e_i' med točkama i in $i+1$ za vsak $i = 1, 2, \ldots, n-1$. Označimo $s = 1$ in $t = n$. Vsaka povezava e grafa G_n ima dolžino $\ell(e) \in \mathbb{N}$ in ceno $c(e) \in \mathbb{N}$. Opazujemo naslednji optimizacijski problem: podana je vrednost $L > 0$, želimo pa minimizirati ceno poti od točke s do točke t pri pogoju, da ima pot dolžino največ L. Naj bo C^* cena optimalne rešitve.

(a) Opiši algoritem, ki optimalno rešuje ta problem v času $O(n^2 C^*)$.

(b) Opiši algoritem, ki optimalno rešuje ta problem v času $O(n^2 L)$.

(c) Recimo, da poznamo vrednost C_0, ki zadošča $C_0 \leq C^* \leq n \cdot C_0$. Opiši PTAS za problem. (PTAS lahko dobi C_0 pri vhodu.)

(d) Opiši PTAS za tak problem.

2.29. Naj bo S množica n enotskih kvadratov v ravnini, ki imajo robove vzporedne koordinatnima osema. Predpostavimo, da so kvadrati v splošnem položaju: x-koordinate in y-koordinate robov kvadratov iz S so same različne. Kvadrati se torej ne dotikajo, lahko pa se sekajo. Množica S definira naslednji graf G_S: vsak kvadrat $s \in S$ definira vozlišče grafa G_S, kvadrata s in s' pa sta soseda natanko tedaj, ko se sekata. Zanima nas največja neodvisna množica grafa G_S. To geometrijsko pomeni, da iščemo največjo podmnožico $R \subseteq S$, kjer se nobena dva kvadrata iz R ne sekata. Ta problem je NP-težak. Iščemo PTAS.

Naj bo k naravno število večje kot 2; k določimo kasneje. Za vsak $a = 0, 1, 2, \ldots, k-1$ definiramo naslednjo množico navpičnih premic

$$V_a = \{\{x = i \cdot k + a\} \mid i \in \mathbb{Z}\}.$$

Naj bo H_b množica vodoravnih premic, ki jo dobimo iz V_b z rotacijo $\pi/2$ v smeri urnega kazalca. Naj bo $S(a, b)$ podmnožica kvadratov s S, ki *ne sekajo* niti V_a niti H_b.

(a) Preveri, da razumeš definicije H_a, V_b in $S(a, b)$.

(b) Pokaži naslednjo trditev: če je S znotraj kvadrata velikosti $k \times k$, potem lahko izračunamo optimalno rešitev v času $O(n^{2+k^2})$.

(c) Pokaži, da lahko izračunamo optimalno rešitev za kvadrate $S(a, b)$ v času $O(n^{2+k^2})$.

(d) Naj bo $Sol(a, b)$ optimalna rešitev za kvadrate $S(a, b)$. Naj bo R najboljša rešitev med $Sol(a, b)$, kjer je $a, b \in \{0, 1, \ldots, k-1\}$. Naj bo R^* optimalna rešitev za S. Pokaži, da je $|R^*| - |R| \leq (4/k)|R^*|$. (Namig: Dirichletov princip)

(d) Opiši PTAS.

(Namig: če je pretežko, naprej razmisli o podobnem algoritmu za eno dimenzijo.)

2.30. Recimo, da imamo orakelj, ki za vsak podan graf v polinomskem času prešteje število njegovih 4-barvanj. Opiši algoritem, ki zna z uporabo zgornjega oraklja v polinomskem času prešteti, koliko 3-barvanj ima vhodni graf. Ali je prevedba varčna (*parsimonious*)?

2.31. Naj bodo $S_1, S_2, \ldots, S_m$ podmožice končne množice U. Za vsak $1 \leq i \leq m$ poznamo moč $|S_i|$. Želimo (ε, δ)-aproksimacijo velikosti množice $S = \bigcup_i S_i$. Predpostavimo, da imamo algoritem, ki pri vhodu i vrne element izbran naključno enakomerno iz S_i. Ravno tako imamo algoritem, ki pri vhodu $x \in U$ vrne

$$c(x) = |\{i \mid x \in S_i\}|.$$

Naslednji poskus ponovimo t krat: v koraku j izberemo naključno podmnožico S_j, kjer je verjetnost, da je S_i izbrana, enaka

$$\frac{|S_i|}{\sum_k |S_k|},$$

in potem iz tako izbrane množice S_j izberemo še njen naključni element x_j. Če uporabljamo cenilko

$$\left(\frac{1}{t}\sum_{j=1}^{t}\frac{1}{c(x_j)}\right)\left(\sum_{i=1}^{m}|S_i|\right)$$

za $|S|$, kako veliko mora biti t, da je cenilka (ε, δ)-aproksimacija za $|S|$.

3 Vzporedni algoritmi

3.1. Podana je tabela $A[1 \ldots n]$, ki vsebuje n celih števil. Konstruiraj tabelo $S[1 \ldots n]$, kjer je $S[i]$ vsota prvih i elementov tabele $A[\,]$. Torej, $S[i] = A[1] + A[2] + \cdots + A[i]$. Opiši vzporedni algoritem v modelu EREW, ki rešuje ta problem v času $O(\log n)$ in potrebuje $O(n/\log n)$ procesorjev.

3.2. Podana je tabela $A[1 \ldots n]$, ki vsebuje n elementov. Podana je tudi tabela $L[1 \ldots n]$, kjer je $L[i] \in \{1, 2, \ldots, k\}$ oznaka elementa $A[i]$. Predpostavimo, da je k vnaprej izbrana konstanta, denimo $k = 196$. Konstruiraj tabelo $B[1 \ldots n]$, ki vsebuje elemente iz $A[1 \ldots n]$ urejene po oznakah: elementi z oznako '1' na začetku, nato elementi z oznako '2', vse do elementov z oznako 'k'. Opiši vzporedni algoritem v modelu EREW, ki rešuje ta problem v času $O(\log n)$ in potrebuje $O(n/\log n)$ procesorjev.

3.3. Podana je tabela $A[1 \ldots n]$, ki vsebuje n elementov, in urejena tabela $J[0 \ldots s]$, ki zadošča $J[i] < J[i+1]$ za vsak i. Naj velja tudi $J[0] = 1$ in $J[s] = n$. Konstruiraj tabelo $B[1 \ldots n]$, ki za vsak ℓ in i, ki ustrezata $J[i-1] < \ell \leq J[i]$ in $2 \leq i \leq n$, zadošča $B[\ell] = A[J[i]]$. Na primer, če je vhod $A = [1, 2, 3, 4, 5, 6, 7, 8]$ in $J = [1, 3, 5, 8]$, potem mora biti izhod $B = [3, 3, 3, 5, 5, 8, 8, 8]$. Opiši vzporedni algoritem v modelu EREW, ki rešuje ta problem v času $O(\log n)$ in potrebuje $O(n/\log n)$ procesorjev.

3.4. Podana je tabela $A[1 \ldots n]$, ki vsebuje n celih števil, in tabela $B[1 \ldots n]$, pri čemer je vsak $B[i] \in \{\top, \bot\}$. Naj velja tudi $B[1] = \top$. Konstruiraj tabelo $\sigma[1 \ldots n]$, pri čemer je $\sigma[i]$ vsota elementov $A[\,]$ na levo od položaja i do vključno prvega položaja j, ki zadošča $B[j] = \top$. Kot primer, če je $A = [1, 2, 3, 4, 5, 6, 7]$ in $B = [\top, \bot, \top, \top, \bot, \bot, \top]$, potem mora biti izhod $\sigma = [1, 3, 3, 4, 9, 15, 7]$. Opiši vzporedni algoritem v modelu EREW, ki rešuje ta problem v času $O(\log n)$ in potrebuje $O(n/\log n)$ procesorjev.

3.5. Podana je tabela $A[1 \ldots n]$, ki vsebuje n različnih točk v ravnini. Štirikotnik S je podan kot tabela oglišč $S[1 \ldots 4]$. Konstruiraj tabelo $B[1 \ldots n]$, ki na začetku vsebuje točke iz A, ki so znotraj štirikotnika S. Opiši vzporedni algoritem, ki rešuje ta problem v času $O(\log n)$ in potrebuje $O(n)$ operacij. Kakšen PRAM model računanja uporabljaš?

3.6. Dano je celo število x. Konstruiraj tabelo potenc celega števila x, $B[1 \ldots n]$, $B[i] = x^i$. Opiši vzporedni algoritem, ki rešuje ta problem v času $O(\log n)$ in potrebuje $O(n)$ operacij. Kakšen PRAM model računanja uporabljaš?

3.7. Imamo cikel C na vozliščih $v_0, v_1, \ldots, v_{n-1}$ s povezavami $v_{i-1}v_{i \bmod n}$ za vsak i. Dana je tudi tabela $E[1 \ldots s]$, kjer je vsak $E[j]$ dodatna povezava. Pri tem velja, da je vsaka točka v_i krajišče največ ene dodatne povezave iz $E[\,]$. Cikel C narišemo v ravnini brez presečišč. Zanima nas, ali lahko narišemo tudi vse povezave iz $E[\,]$ znotraj C brez presečišč. Opiši vzporedni algoritem, ki rešuje tak problem v času $O(\log n)$ in potrebuje $O(n)$ operacij. Kakšen PRAM model računanja uporabljaš?

3.8. Podana je tabela $A[1 \ldots n]$ celih števil in tabela $Z[1 \ldots n]$ pozitivnih celih števil, pri čemer velja $Z[i] \leq i$. Izračunaj tabelo $M[1 \ldots n]$, kjer je $M[i]$ maksimalno število v podtabeli $A[Z[i] \ldots i]$, $M[i] = \max_{j=Z[i]}^{i} A[j]$. (Črka Z pomeni 'začetek'.) Opiši algoritem, ki rešuje ta problem v $O(\log n)$ času in potrebuje $O(n)$ procesorjev.

3.9. Opiši algoritem, ki v času $O(\log n)$ izračuna produkt AB, kjer sta A in B $n \times n$ matriki. Kakšen PRAM model računanja uporabljaš?

3.10. Podana je tabela $A[1 \ldots n]$, pri čemer je $A[i] \in \{0, 1\}$ za vsak i. Opiši algoritem v modelu CRCW, ki v času $O(1)$ najde prvi indeks k, ki zadošča $A[k] = 1$. Število operacij algoritma naj bo linearno. Namig: deli in vladaj na $\sqrt{n}$ podproblemov.

3.11. Podana je tabela $A[1 \ldots n]$ celih števil. Zanima nas vrednost $\max_i A[i]$. Opiši algoritem, ki ta problem rešuje v času $O(1)$ z uporabo $n^{1.01}$ procesorjev. Kakšen PRAM model računanja uporabljaš?

3.12. Podana je tabela $A[1 \ldots n]$ celih števil. Opiši algoritem, ki v času $O(1)$ odloči, ali je maksimalni element ponovljen.

3.13. Podana je tabela $A[1 \ldots n]$. Opiši algoritem v modelu CRCW, ki z uporabo polinomskega števila procesorjev v konstantem času odloči, ali je vsak element iz tabele A v njej zapisan natanko dvakrat. Na primer, za vhod $A[c, a, b, c, b, a]$ mora biti odgovor 'DA', za vhoda $A[a, a, a, a, b, b]$ ali $A[a, b, c, a, b]$ mora biti odgovor 'NE'.

3.14. Podana je tabela $A[1 \ldots n]$ celih števil.

(a) Opiši algoritem v modelu EREW, ki z uporabo polinomskega števila procesorjev v času $O(\log n)$ odloči, ali so vsi elementi iz tabele A enaki.

(b) Opiši algoritem v modelu EREW, ki z uporabo polinomskega števila procesorjev v času $O(\log n)$ odloči, ali ima vsak element iz tabele A isto število ponovitev v A. Na primer, za vhoda $A[3, 1, 2, 3, 2, 1]$ ali $A[1, 1, 1, 2, 2, 3, 3, 2, 3]$ mora biti odgovor 'DA' in za vhoda $A[1, 1, 1, 1, 2, 2]$ ali $A[1, 2, 3, 1, 2]$ mora biti odgovor 'NE'.

3.15. Podani sta urejeni tabeli $A[1 \ldots n]$ in $B[1 \ldots n]$ različnih celih števil: $A[1] < A[2] < \cdots < A[n]$ in $B[1] < B[2] < \cdots < B[n]$. Zanima nas konstrukcija *urejene* tabele $C[1 \ldots 2n]$, ki vsebuje vse elemente iz A in B.

(a) Opiši algoritem v modelu CRCW, ki ta problem reši v času $O(1)$.

(b) Opiši algoritem v modelu CREW, ki ta problem reši v času $O(\log n)$.

(c) Opiši algoritem v modelu EREW, ki ta problem reši v času $O(\log^2 n)$.

(d) Pokaži, da je problem urejanja različnih celih števil v razredu NC.

(e) Če imata A in B lahko ponovljene elemente, kaj moramo spremeniti algoritmu, ki reši nalogo (a)?

3.16. Podana je urejena tabela $A[1 \ldots n]$ različnih celih števil, $A[1] < A[2] < \cdots < A[n]$. Podano je tudi celo število x, ki zadošča $x \leq A[n]$. Zanima nas $rank(x : A) = \max\{i \in \{1, \ldots, n\} \mid A[i] \leq x\}$. Opiši algoritem v modelu CRCW, ki rešuje tak problem v času $O(\log n / \log p)$, kjer je p število procerjev.

3.17. Uporabimo model CRCW. Vhod je tabela $A[1 \ldots n]$ različnih celih števil. Konstruiraj tabelo $B[1 \ldots n]$, kjer je $B[i]$ maksimalni indeks elementa iz $A[1 \ldots i]$, ki je večje kot $A[i]$. Če $A[i]$ je največji element v $A[1 \ldots i]$, potem naj bo $B[i] = 0$. Na primer, za vhod $A = [3, 7, 5, 2, 4, 8, 1]$ mora biti izhod $B = [0, 0, 2, 3, 3, 0, 6]$.

(a) Zapiši psevdokodo algoritma, ki rešuje ta problem v konstantem času in potrebuje polinomsko število procesorjev.

(b) Opiši algoritem, ki rešuje ta problem v času $O(\log n)$ in potrebuje linearno število procesorjev.

3.18. Drevo s korenom opišemo s tabelo $P[1 \dots n]$. Pri tem za koren r velja $P[r] = r$, sicer pa $P[i]$ označuje očeta elementa i. Opiši algoritem, ki izračuna 2-barvanje drevesa v času $O(\log n)$.

3.19. Drevo s korenom opišemo s tabelo $P[1 \dots n]$. Pri tem za koren r velja $P[r] = r$, sicer pa $P[i]$ označuje očeta elementa i. Opiši algoritem, ki izračuna v času $O(\log n)$ število listov drevesa. (Ali je lahko koren drevesa list?)

3.20. Podana je tabela $A[1 \dots n]$, ki vsebuje cela števila med 1 in k, pri čemer je k vnaprej znana vrednost. Konstruiraj tabelo $B[1 \dots n]$ na tak način, da je $B[1], B[2], \dots B[n]$ permutacija števil $\{1, 2, \dots n\}$ in je $A[B[i]] \leq A[B[i+1]]$ za vsak $i = 1, 2, \dots, n-1$. Na primer, če je vhod $k = 4$ in $A = [1, 2, 3, 4, 1, 1, 3, 3]$, potem je ustrezen izhod $B = [1, 5, 6, 2, 3, 7, 8, 4]$.

(a) Opiši algoritem v modelu CRCW, ki vrne v času $O(k \log n)$ tabelo $B[\,]$.

(b) Opiši algoritem v modelu CRCW, ki vrne v času $O(\log k \log n)$ tabelo $B[\,]$.

Naj bo T drevo s korenom r in n vožlišči. Drevo s korenom opišemo s tabelo $P[1 \dots n]$. Pri tem za koren r velja $P[r] = r$, sicer pa $P[i]$ označuje očeta elementa i. Poišči permutacijo $\sigma[1 \dots n]$ elementov $\{1, 2, \dots n\}$, ki zadošča: $\sigma[1] = r$ poleg tega pa velja $i < j$, če je $\sigma(i)$ oče vozlišča $\sigma(j)$. Na primer, če je $n = 6$ in $P = [4, 1, 4, 4, 1, 3]$, potem je možna rešitev $\sigma = [4, 1, 3, 5, 2, 6]$.

(c) Opiši algoritem v modelu CRCW, ki v času $O(\log^2 n)$ vrne takšno permutacijo σ.

3.21. Vozlišča drevesa s korenom T označimo s $1, 2, \dots, n$. Drevo je podano s tabelo $P[1 \dots n]$, kjer je $P[i]$ oče vozlišča i. Za koren r pa velja $P[r] =$NULL. Izračunati želimo tabelo $B[1 \dots n][1 \dots n]$, ki zadošča

$$B[i][j] = \begin{cases} 1 & \text{če je } j \text{ na poti od } i \text{ do korena;} \\ 0 & \text{sicer.} \end{cases}$$

(a) Opiši algoritem s psevdokodo, ki rešuje ta problem v času $O(\log n)$ in potrebuje polinomsko število procesorjev. (Namig: skakanje kazalcev za neko operacijo, ki jo lahko izračunamo v $O(1)$ času z uporabo polinomskega števila procesorjev.)

(b) Denimo, da je drevo T seznam. Opiši algoritem v modelu EREW, ki rešuje ta problem v času $O(\log n)$ z uporabo $O(n^2/\log n)$ procesorjev.

3.22. Vozlišča drevesa s korenom T označimo s $1, 2, \dots, n$. Drevo je podano s tabelo $P[1 \dots n]$, kjer je $P[i]$ oče vozlišča i. Za koren r pa velja $P[r] =$NULL. Drevo T je posebnega tipa: vsako vozlišče, ki ni koren, ima največ enega sina. Koren lahko ima več sinov. Podana je tudi tabela $w[1 \dots n]$, kjer je $w[i]$ teža vozlišča i.

(a) Opiši algoritem v modelu CRCW, ki v času $O(\log n)$ izračuna število sinov korena.

(b) Opiši algoritem v modelu CRCW, ki v času $O(\log n)$ izračuna za vsako vozlišče i vrednost

$$\phi(i) = \sum_{\substack{u \in V(T) \text{ ni na nobeni} \\ \text{poti od } i \text{ do korena}}} w(u).$$

3.23. Naj bosta $A[1 \ldots n]$ in $B[1 \ldots n]$ tabeli bitov, ki hranita n-bitov števil a in b.

(a) Opiši algoritem v modelu EREW, ki v času $O(\log n)$ odloči, ali je $a < b$, $a = b$ ali $a > b$.

(b) Opiši algoritem v modelu CRCW, ki v času $O(1)$ odloči, ali je $a < b$, $a = b$ ali $a > b$.