Vaje pri PSA1

Jure Slak, študijsko leto 2020/21

Naloga 0. Časovna zahtevnost ni isto kot čas izvajanja. Učinkovitost ni skalabilnost.

Naloga 1. Zapiši v O notaciji:

$$4n + \log n + 4/n, 2^{\log n+1}, 5n^3 + 2^{6\log_{10} n}, \lceil\sqrt{n}\rceil + 12\log n^3, \log^2 n + \log(n^2)$$

Naloga 2. Postavi zahtevnosti v pravi vrstni red, da velja relacija $=$.

$$O(n^2) = O(2^n) = O(n!) = O(2^{2^n}) = O(\log n) = O(\log\log n) = O(\sqrt{n}) = \\ = O(2^{n\log n}) = O(\log(n!)) = O(1) = O(n\log n) = O(\log_2 n)$$

Naloga 3. Napiši algoritem za iskanje maksimuma seznama. Napiši še en (drugačen) algoritem za iskanje maksimuma seznama. Napiši še en (drugačen) algoritem za iskanje maksimuma seznama.

Primerjaj algoritme po različnih lastnostih:

- Časovna zahtevnost
- Prostorska zahtevnost
- Paralelizabilnost
- Online / Offline

Naloga 4. Analiziraj urejanje z izbiranjem:

```
def selection_sort(s):
    for last_idx in range(len(s)-1, 0, -1):
        max_idx = 0
        for i in range(1, last_idx+1):
            if s[i] > s[max_idx ]:
                max_idx = i
        s[last_idx], s[max_idx] = s[max_idx], s[last_idx]
```

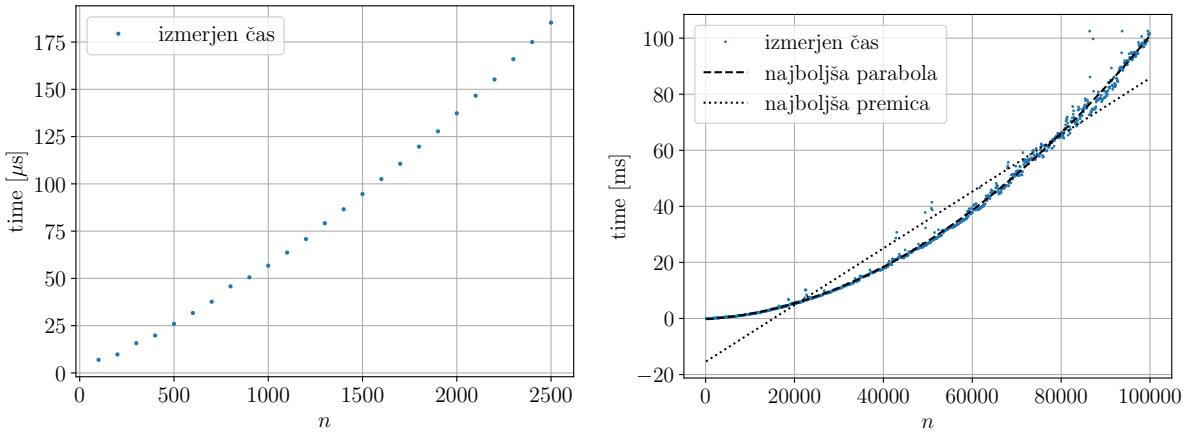
Naloga 5. Navedi osnovne podatkovne strukture v Pythonu in časovne zahtevnosti standardnih operacij na njih.

Naloga 6. Analiziraj časovno zahtevnost funkcij `fib1` in `fib2` ob predpostavki, da je operacija $+$ izvedljiva v $O(1)$.

```
def fib1(n):
    if n == 0: return 0
    if n == 1: return 1
    return fib1(n-2) + fib1(n-1)
```

```
def fib2(n):
    a, b = 0, 1
    if n == 0: return a
    for i in range(1, n):
        a, b = b, a+b
    return b
```

Naloga 7. Analiziraj časovno zahtevnost funkcije `fib2` z upoštevanjem časovne zahtevnosti operacije $+$. Namig: slika 1.



Slika 1: Čas izvajanja funkcije `fib2` glede na n.

Naloga 8. Izračunaj `7 & 9`, `7 | 9`, `7 ^ 9`, `~3`, `7 << 2`, `7 >> 2`. Kako bi preveril ali je i-ti bit prižgan? Kako bi ga nastavil na 1? Kako bi ga nastavil na nič?

Naloga 9. Napiši algoritem za izračun a^n; $a, n \in \mathbb{N}$, pri predpostavki, da je množenje dveh števil $O(1)$. Analiziraj časovno zahtevnost. Ali ga lahko izboljšaš? Lahko pogoje za a sprostimo?

Naloga 10. Na predavanjih ste spoznali algoritem `karatsuba` za množenje števil. Kaj je bitni zapis števila -1? Kaj pa -3? Dokaži, da je bitni zapis števila $-a$ enak $\sim a + 1$.

```
def karatsuba(m, n, b):
    if b == 0: return m*n
    m1, m2 = m >> b, m & ~(-1 << b)
    n1, n2 = n >> b, n & ~(-1 << b)
    mn1 = karatsuba(m1, n1, b//2)
    mn2 = karatsuba(m2, n2, b//2)
    mn0 = karatsuba(m1+m2, n1+n2, b//2) - mn1 - mn2
    return (mn1 << 2*b) + (mn0 << b) + mn2
```

Pomnoži števili $\texttt{10011011}_2$ in $\texttt{10111010}_2$. Kaj izberemo za b?

$(155 \cdot 186 = 28830 = 111000010011110_2)$

Naloga 11. Opiši in analiziraj algoritem za urejanje seznama n celih števil x_i, za katere velja $x_i \in [0, M]$. Zakaj ima lahko tako časovno zahtevnost? Kdaj se splača uporabiti ta algoritem? Ali lahko razviješ stabilno verzijo?

Naloga 12. Opiši in analiziraj algoritem za urejanje seznama n nizov iz malih črk angleške abecede. Zakaj ima lahko tako časovno zahtevnost? Kdaj se splača uporabiti ta algoritem?

Naloga 13. Med spodnjimi algoritmi izberi najboljšega, pri čemer je n velikost vhodnega problema.

- Algoritem A reši problem tako, da ga razdeli na pet podproblemov velikosti $n/2$, rekurzivno reši vsak podproblem, nato pa v linearnem času združi rešitve.
- Algoritem B reši problem tako, da rekurzivno reši dva podproblema velikosti $n-1$, nato pa rešitvi združi v konstantnem času.
- Algoritem C reši problem tako, da ga razdeli na devet podproblemov velikosti $n/3$, rekurzivno reši vsak podproblem, nato pa rešitve združi v času $O(n^2)$.

Naloga 14. Opiši in analiziraj dva algoritma za iskanje para najbližjih točk v ravnini. Posploši na $\mathbb{R}^d$.

Zanimivost: drugi D&C algoritmi: iskanje lastnih vrednosti simetrične matrike, izračun hitre Fourierove transformacije (FFT).

Naloga 15. Imamo k urejenih seznamov $A_1, \ldots, A_k$ dolžine n, ki jih želimo zliti v en sam urejen seznam dolžine kn. Kakšna je časovna zahtevnost, če jih zlivamo enega za drugim? Razvij učinkovitejši algoritem, ki uporablja metodo deli in vladaj.

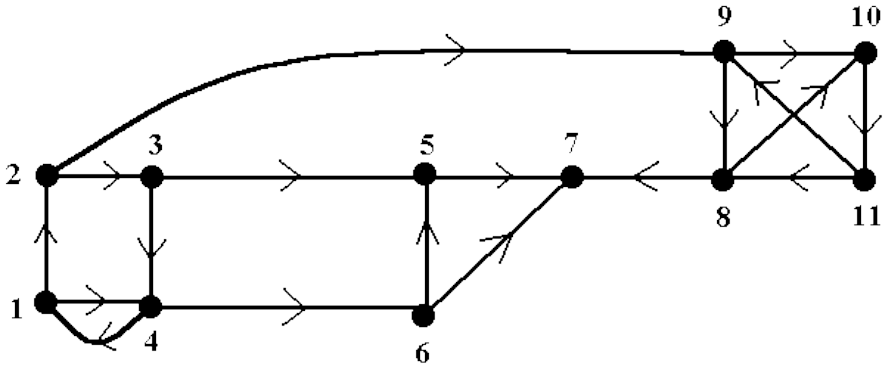
Naloga 16. Naj bo a urejen seznam celih števil dolžine n. Pokaži, da za določitev takega indeksa i, da za dano število x velja $a[i] = x$, potrebujemo $\Omega(\log n)$ korakov, če lahko do seznama dostopamo samo s primerjavami. Dokaži, da lahko mejo dosežemo.

Naloga 17. Pokaži delovanje vrste in sklada na primeru. Podaj idejo implementacije vrste in sklada s pomočjo seznama fiksne dolžine. Predpostaviti smeš, da v vrsti ali skladu nikoli ne bo več kot N elementov. Kaj pa, če te predpostavke nimaš? Kako uporabljamo vrsto in sklad v Pythonu/Javi/C/C++?

Naloga 18. Pogosto imamo pri programiranju opravka z (2D) mrežami (grid) in implicitno podanih grafih na njih. Razvij pristop za delo z mrežami, ki se dobro generalizira na različno definirane sosednosti, dane s končno množico sosedov (4-sosednost, 8-sosednost, konj-sosednost, ciklične verzije). Koliko vozlišč in povezav imajo ti grafi? Kaj pa če je množica sosedov drugačna (npr. kraljica).

Naloga 19. Implementiraj "bucket fill" operacijo, ki je na voljo v vsakem programu za urejanje slik. Katere probleme še lahko rešimo z istim algoritmom?

Naloga 20. Zapiši primerno predstavitev grafa spodaj in na njem izvedi algoritem `dfs`. Posploši na grafe z več komponentami. Preštej jih.



Naloga 21. Zapiši DFS in ga spremeni v verzijo, ki ne potrebuje rekurzije.

Naloga 22. V računalništvu pogosto naletimo na situacijo, ko nek program ali paket za svoje delovanje potrebuje druge programe, ki spet potrebujejo druge programe, itd. Problem nastane, če se zgodi, da nek program za svoje delovanje potrebuje drug program, ki pa spet (lahko posredno) potrebuje prvega. Standardna distribucija Linux-a ima npr. 32 000 paketov, od katerih ima vsak potrebuje nekaj drugih paketov. Recimo, da sedaj nek paket p dodatno doda paket q kot dependency. Razvij algoritem, ki bo preveril, ali to povzroči kakšen cikel?[1]

Namig: problem najprej prevedi v problem iz teorije grafov.

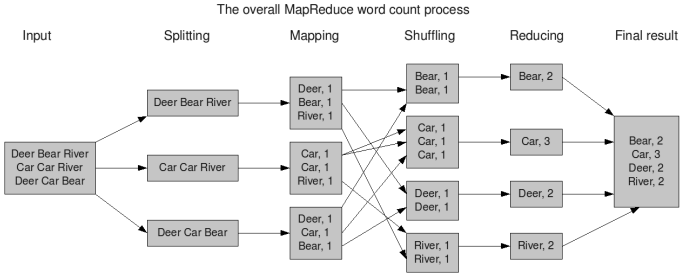
Kako bi preveril, ali dan graf vsebuje cikel?

Naloga 23. Opiši algoritem za reševanje labirintov na mreži. Razvij algoritem za generiranje naključnih labirintov na mreži s pomočjo `dfs`.

Naloga 24. Kako bi na roke poiskal topološko ureditev DAGa? Zapiši ta algoritem in analiziraj njegovo časovno zahtevnost.

Naloga 25. Razvij algoritem, ki ugotovi, ali ima DAG enolično topološko ureditev. Kako bi preštel vse možne ureditve?

Naloga 26. Pri izvajanju kompleksnih paralelnih procesov, kot npr. MapReduce[2] imamo več vhodov, nekaj vmesnih izračunov v enem ali več korakih in enega ali več izhodov. Najmanj koliko sekvenčnih korakov je potrebnih za izračun rezultata?[3]



[1] V praksi so v primeru Linux paketov dovoljeni tudi cikli.

[2] `https://en.wikipedia.org/wiki/MapReduce`

[3] slika vzeta z: `http://www.milanor.net/blog/an-example-of-mapreduce-with-rmr2/`

Naloga 27. Razvij algoritem, ki poišče najdaljšo pot v DAGu. Analiziraj njegovo časovno zahtevnost.

Naloga 28. Razvij algoritem, ki poišče število poti med s in t v DAGu.

Naloga 29. Spomni se Kosajarovega algoritma za iskanje močno povezanih komponent in razvij alternativen in v praksi učinkovitejši algoritem.

Naloga 30. Reši problem 2SAT. Dana je logična formula v KNF, pri čemer so disjunkcije kvečjemu dvočlenske, npr.

$$(x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_3) \wedge (x_1 \vee x_2) \wedge (\bar{x}_3 \vee x_4) \wedge (\bar{x}_1 \vee x_4)$$

Ali obstaja nabor vrednosti $x_1, \ldots, x_n$, da je resnična?

Naloga 31. Izvedi BFS na izbranem primeru (npr. Petersenov graf) in nariši BFS drevo.

```
from collections import deque
def bfs(G):
    n = len(G)
    visited = [False] * n
    for i in range(n):
        if not visited[i]:
            q = deque([i])
            while q:
                c = q.popleft()
                if visited[c]: continue
                visited[c] = True
                # obravnavaj c
                for u in G[c]:
                    if not visited[u]:
                        q.append(u)
```

Naloga 32. Modificiraj BFS tako, da bo poiskal dolžino najkrajše poti med s in t v usmerjenem grafu. Kaj pa če želimo dejansko pot?

Naloga 33. S pomočjo BFS razvij algoritem, ki v času $O(V + E)$ poišče najkrajšo pot v grafu, kjer so povezave dolžine 1 ali 2.

Naloga 34. (RTK 2016, 2. skupina) Mirko je navdušen žužkofil. Doma ima terarij z n žužki, oštevilčenimi z števili od 1 do n, toda ne pozna njihovega spola. Kljub temu pa trdi, da so gotovo vsi žužki heteroseksualni. To želi dokazati tako, da en mesec strmi v terarij in si beleži interakcije med žužki. Natančno opiši postopek, ki sprejme seznam interakcij med žužki, in pove, ali obstaja taka razporeditev spolov, da so bile vse interakcije heteroseksualne. Žužki se vedno razmnožujejo samo v parih in noben žužek ni nikoli v interakciji sam s sabo. Žužki so lahko samo dveh spolov in spola ne spreminjajo med mesecem opazovanja.

Naloga 35. Izračunaj število najkrajših poti od s do t.

Naloga 36. Poišči število trikotnikov v neusmerjenem grafu G.

Naloga 37. Izvedi Dijsktrov algoritem na izbranem grafu.

Naloga 38. Analiziraj predpostavke Dijkstrovega algoritma. Ali so vse nujne? Poskusi jih sprostiti.

Naloga 39. Poskusi problem najdaljše poti v grafu prevesti na problem najkrajše poti v grafu preko Dijkstrovega algoritma. Obupaj in pokaži, da je problem najdaljše poti vsaj tako težak kot problem Hamiltonove poti.

Naloga 40. Implementiraj kopico s seznamom. Izvedi zaporedje push in pop operacij na primeru. Opiši časovne zahtevnosti operacij in opiši urejanje s kopico.

Naloga 41. V računalniških igricah imamo pogosto dogodke (kot npr. status efekte), ki se zgodijo ali pa se bodo zgodili v kratki prihodnosti, npr. zadane te goreča puščica in moraš takoj zagoreti, čez 5 sekund pa se ogenj pogasi. Opiši podatkovno strukturo, ki hrani te dogodke, tako da se na vsaki iteraciji glavne zanke lahko učinkovito obravnava vse dogodke, ki bi se morali zgoditi med zadnjo in predzadnjo iteracijo ter doda morebitne nove.

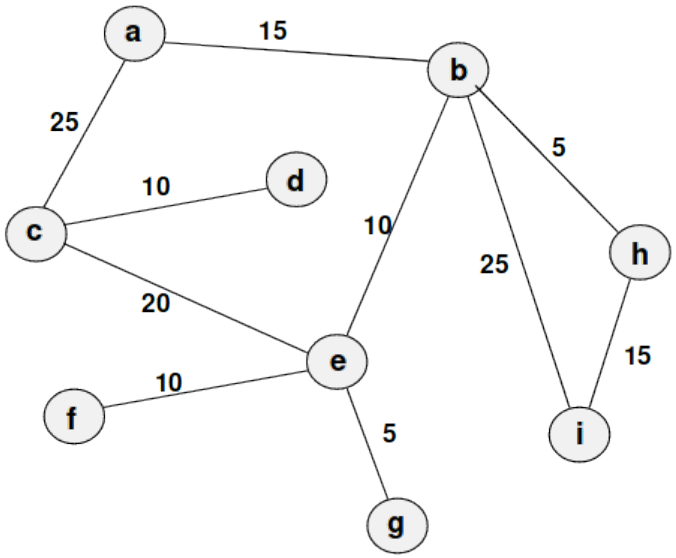
Naloga 42. Posploši Dijkstrov algoritem na grafe, ki imajo uteži tudi na vozliščih.

Naloga 43. Oglej si, kako Dijkstra išče najkrajše poti na primeru sveta, predstavljenega s kvadratno mrežo, z ovirami. Razvij algoritem za iskanje najkrajše poti, imenovan A^*, ki v praksi deluje bolje.

Naloga 44. Dane so 4 točke v ogliščih kvadrata v ravnini. Poveži jih med seboj, tako da bo med vsakim dvema obstajala pot in da bo skupna dolžina poti čim krajša.

Opomba: Steinerjevo razmerje $\rho = \sup_{\mathcal{P}} \frac{|\mathrm{SMT}(\mathcal{P})|}{|\mathrm{MST}(\mathcal{P})|}$ za ravnino je še odprt problem (Gilbert–Pollak Conjecture).

Naloga 45. Na primeru grafa G najdi najmanjše vpeto drevo s pomočjo Kruskalovega algoritma.



Naloga 46. Naredi enako s Primovim algoritmom.

Naloga 47. Imejmo 8 elementov, [0, 1, 2, 3, 4, 5, 6, 7], ki naj bodo vsak v svoji množici z rangom 1. Izvedi naslednje operacije:
`union(1, 7), union(2, 3), union(3, 4), union(5, 1), union(7, 4)`

Naloga 48. Dana je mreža s črnimi in belimi celicami. Razvij podatkovno strukturo, ki hrani informacijo o povezanih skupinah črnih celic in omogoča njihovo združevanje. Analiziraj časovno zahtevnost in simuliraj na 4×4 mreži. Posploši na barvne slike. Uporabi na primeru postavljanja zgradb v računalniških igricah.

Naloga 49. Zakodiraj stavek `A GREMO LAHKO ŽE DOMOV`? Koliko prostora smo pridobili?

Naloga 50. Odkodiraj prej skonstriuran stavek. Kaj opaziš? Razmisli o metodah za reševanje tega problema. Kakšna sta časovna in prostorska zahtevnost dekodiranja? Pri dekodirnih algoritmih je zelo pomembno, da so streaming algoritmi, ali ta ustreza?

Naloga 51. Predpostavi, da so simboli že urejeni po frekvencah. Konstruiraj Huffmanovo drevo v $O(n)$ časa. Ali je časovna zahtevnost tega algoritma zelo pomembna?

Naloga 52. Konstruiraj primer, ko se Huffmanovo kodiranje prevede nazaj na bločno kodiranje.

Naloga 53. Dana je spodnjetrikotna matrika števil A. Najdi največjo možno vsoto števil, pri čemer lahko seštevaš števila tako, da začneš pri edinem elementu v prvi vrstici in se pri prehodu v naslednjo vrstico premakneš direktno navzdol ali diagonalno navzdol v desno. Kakšna je časovna zahtevnost izračuna?

Naloga 54. Na klopco želimo posestni n politikov, ki so rdeči, beli ali roza. Rdeči in beli se med seboj ne marajo, tako da jih ne smemo posesti skupaj. Na koliko načinov lahko to naredimo? Kakšna je časovna zahtevnost izračuna?

Naloga 55. V gostilni morate plačati račun za x eur. Na koliko načinov lahko to naredite z bankovci in kovanci, če imate neomejeno zalogo vseh kovancev in bankovcev. Kaj pa če imate dane neke fiksne zaloge? Kakšna je časovna zahtevnost izračuna?

Naloga 56. Dana je kvadratna mreža z ovirami. Na koliko načinov je možno priti od levega spodnjega kota do zgornjega desnega, če se lahko premikam gor, desno in diagonalno?

Naloga 57. Imamo šahovnico s 4 vrsticami in n stolpci, pri čemer je na vsakem polju napisana številka. Na voljo imamo še $2n$ kamenčkov, ki jih želimo postaviti na polja šahovnice tako, da bo vsota pokritih števil čim večja. Ni potrebno, da postavimo vse kamenčke, na vsakem polju pa je lahko največ en kamenček. Pri tem imamo omejitev, da dveh kamenčkov ne smemo postaviti na polji s skupnim robom (lahko pa ju postavimo na polji s skupnim ogliščem). Zapiši algoritem, ki s pomočjo dinamičnega programiranja v času $O(n)$ izračuna optimalno postavitev kamenčkov. Posploši iz 4ih na k vrstic in oceni časovno zahtevnost.