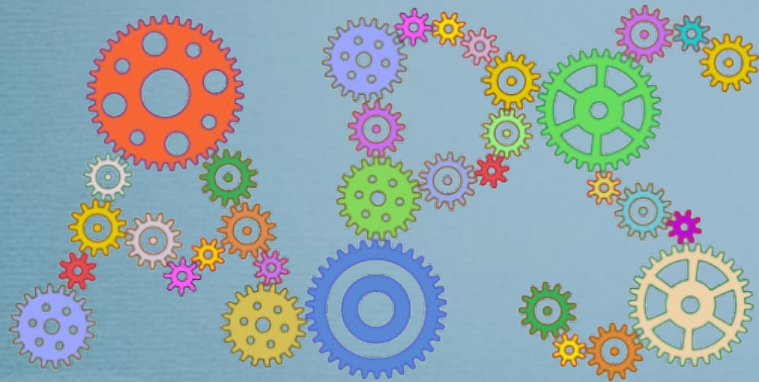


Algoritmi in podatkovne strukture 1

Visokošolski strokovni študij Računalništvo in informatika

Požrešni
algoritmi



Požrešna metoda

- Ideja metode

- postopna gradnja končne rešitve
- pričnemo s »prazno« delno rešitvijo
- **zaporedoma** dopolnjujemo rešitev
- na vsakem **koraku** izberemo v tistem **trenutku** najbolj **obetaven** način dopolnitve rešitve
 - npr. največje povečanje kriterijske funkcije
 - kratkovidnost pogleda
- dokler ne dobimo končne rešitve

Menjava kovancev

- Klasičen primer
 - s kovanci vrednosti
 - 200, 100, 50, 20, 10, 5, 2, 1 centov
 - kovancev posamezne vrednosti je dovolj
 - sestavi skupno vrednost k centov
 - uporabi čim manj kovancev

Menjava kovancev

- Požrešna rešitev
 - po vrsti po padajoči vrednosti kovanca
 - izbiramo kovance dokler je skupna vrednost $\leq k$
- Ali *požrešni princip* vedno deluje?
 - s kovanci vrednosti 10, 8, 1 eur in sestavi 17 eur
- Kdaj deluje *požrešni princip*?

Požrešna metoda

- Prednosti
 - algoritmi lahki za snovanje
 - pogosto lahko zasnujemo več različnih požrešnih algoritmov
 - algoritmi lahki za implementacijo
 - edini „zapleteni“ korak je izbira nove komponente
 - algoritmi so učinkoviti (hitri)
 - polinomski z majhno stopnjo
 - analiza časovne zahtevnosti je enostavna

Požrešna metoda

- Slabosti

- požrešni algoritem morda ne deluje
 - ne najde vedno **natančne** rešitve
 - na vsakem koraku upoštevamo le trenutni scenarij, ne upoštevamo globalnega scenarija
- pravilnost algoritma težko dokazati
 - algoritem je pravilen, če vedno vrne natančno rešitev
 - kadar algoritem deluje, je dokaz njegove pravilnosti dokaj zapleten

Požrešna metoda

- Približni algoritem

- algoritem **ne vrača natančne** rešitve ampak **približno**

- z vidika natančnosti torej ni pravilen

- sprostimo omejitve natančnosti

- požrešni algoritmi navadno vračajo **dopustne rešitve**

- za reševanje optimizacijskih problemov

- takšni rešitvi pravimo **približna rešitev**

- včasih smo s tem zadovoljni

- reševanje *NP*-težkih problemov

- hitra (polinomska) približna rešitev

- za natančno rešitev (če $P \neq NP$) potrebujemo eksponento časa

Požrešna metoda

- *NP*-težki problemi
 - poiščemo **natančni eksponentni** algoritem
 - groba sila, izčrpno preiskovanje, sestopanje, ...
 - vedno vrne natančno rešitev
 - eksponentna časovna zahtevnost
 - poiščemo **približni polinomski** algoritem
 - požrešna metoda, ...
 - ne vrne vedno natančne rešitve (lahko je približna)
 - vedno vrne dopustno rešitev
 - polinomska časovna zahtevnost

Razporeditev datotek na trak

- Opis problema

- dano množico datotek
 - vsaka datoteka ima neko dolžino
- razporedi na trak
- tako, da bo povprečni čas dostopa datoteke
 - vedno začnemo na začetku traku
- najmanjši



Razporeditev datotek na trak

- Definicija problema

- naloga

- n datotek: $1, 2, \dots, n$
 - dolžine datotek: l_1, l_2, \dots, l_n
 - dolžina $l(i) = l_i$

- dopustna rešitev

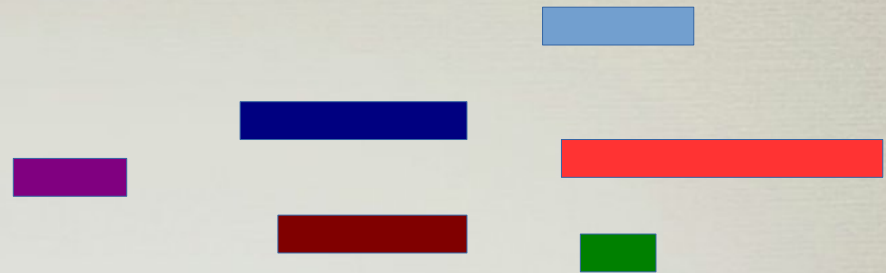
- zaporedje datotek: $s = (s_1, s_2, \dots, s_n)$

- cilj

- najmanjši povprečni čas branja

$$t_i(s) = c \cdot \sum_{j=1}^i l(s_j)$$

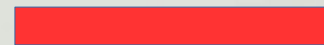
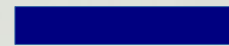
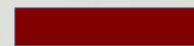
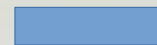
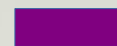
$$\bar{t}(s) = \frac{1}{n} \sum_{i=1}^n t_i(s)$$



Razporeditev datotek na trak

- Požrešni algoritem

- izbira najkrajše datoteke

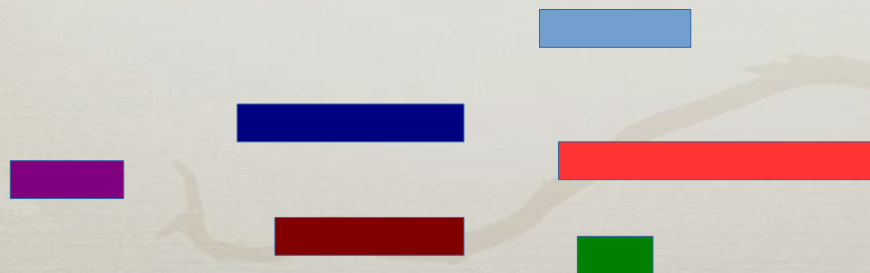


- datoteke urejene po dolžini

- dolžine datotek: $l_1 \leq l_2 \leq \dots \leq l_n$

- neurejene datoteke

- dolžine datotek: l_1, l_2, \dots, l_n



Razporeditev datotek na trakove

- Posplošitev prejšnjega problema
 - n datotek
 - m trakov
- Požrešni algoritem
 - izbira najkrajše datoteke
 - zapis na najmanj zaseden trak
 - datoteke torej zapisujemo na trakove po vrsti

Požrešna metoda

- Kako dokažemo pravilnost algoritma?
 - optimalna podstruktura
 - optimalna rešitev problema vsebuje optimalne rešitve podproblemov
 - požrešna lastnost
 - požrešna izbira vodi v optimalnost



Primeri požrešnih algoritmov

- Natančni polinomski algoritmi
 - najcenejše poti v grafu
 - Dijkstrov algoritem
 - najcenejše vpeto drevo
 - Primov algoritem
 - Kruskalov algoritem
 - predponsko kodiranje
 - Huffmanov algoritem

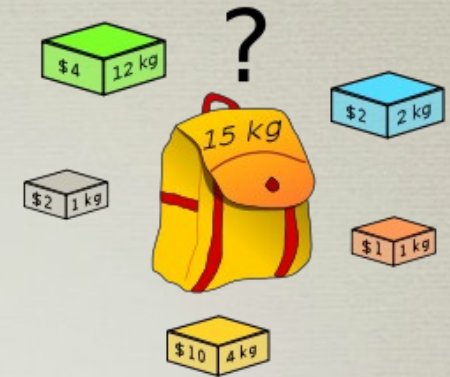
Primeri požrešnih algoritmov

- Približni polinomski algoritmi
 - trgovski potnik
 - vedno obiščemo najbližjega soseda
 - največja klika, največja neodvisna množica, največja dominantna množica
 - uredimo vozlišča po stopnjah, izbiramo dokler ne dobimo dopustne rešitve
 - najmanjše vozliščno pokritje
 - ALG1: uredimo vozlišča po stopnjah in izbiramo dokler ne dobimo rešitve
 - ALG2: izbiranje povezav (poiščemo maksimalno ujemanje)

Problem nahrbtnika

- Definicija problema

- nahrbtnik prostornine V
- n predmetov oštevilčenih od 1 do n
- za vsak predmet $i \in \{1, 2, \dots, n\}$ poznamo
 - c_i – cena / vrednost predmeta i , $c_i > 0$
 - v_i - velikost predmeta i , $0 < v_i \leq V$

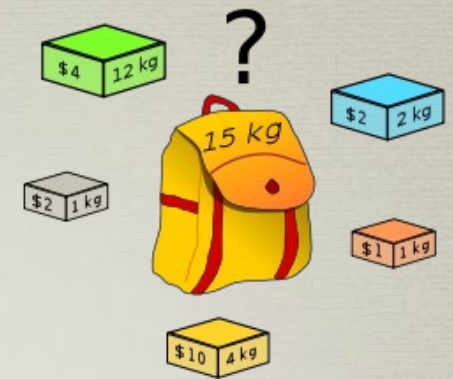


- Poišči nabor predmetov

- katerih skupna velikost ne presega prostornine nahrbtnika
- katerih skupna vrednost je največja

Problem nahrbtnika

- Dve različici problema
 - preprosti, navadni ali zvezni nahrbtnik
 - predmete lahko režemo
 - dvojiški ali 0/1 nahrbtnik
 - predmetov ne smemo rezati



Preprosti nahrbtnik

- Definicija problema

- naloga

- prostornina V , n predmetov, ki jih lahko **režemo**
 - vrednosti (c_1, c_2, \dots, c_n)
 - velikosti (v_1, v_2, \dots, v_n)

- dopustna rešitev

- deleži (x_1, x_2, \dots, x_n) , kjer $0 \leq x_i \leq 1$

- omejitev

- skupna velikost ne presega prostornine nahrbtnika

$$\sum_{i=1}^n x_i v_i \leq V$$

- cilj

- katerih skupna vrednost je največja

$$\text{maxi} \sum_{i=1}^n x_i c_i$$

Preprosti nahrbtnik

- Požrešni algoritem
 - zaporedoma izbiramo predmete
 - upoštevamo omejitve
 - izberemo predmet po nekem kriteriju
 - primer
 - z različnimi kriteriji
 - kateri kriterij vodi do optimalnosti?
 - algoritem
 - zahtevnost
 - pravilnost