Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Igor Rožanc

Algoritmi in podatkovne strukture 1 –APS1

1. LABORATORIJSKE VAJE

(18. – 22. oktober 2021)

2. letnik

BVS Računalništvo in informatika

Študijsko leto 2021/22

VSEBINA

1. Uvodna predstavitev
2. Teorija 1: Algoritmi
   - Računski problemi
   - Vrste računskih problemov
   - Opis algoritmov
   - Sled izvedbe algoritma
   - Metode snovanja algoritmov
3. Kviz: Algoritmi

1. naloga

Problem: Izmed vseh udeležencev vaj želimo poiskati tistega, ki bo prvi slavil rojstni dan.

a) Ali je ta problem računski? Zakaj?
b) Ali je problem dobro definiran? Če ni, popravi opis!
c) Napišite tri primere nalog za ta problem!
d) Koliko je možnih nalog za ta problem?

2. naloga

Enak problem kot pri prvi nalogi.

a) Kaj je vhod in kaj izhod za algoritem, ki rešuje problem?
b) Kakšna je formalna definicija problema?
c) Opišite algoritem v naravnem jeziku!
d) Ali obstaja več takih algoritmov?
e) Za kakšne vrste problem gre?
f) Spremenite problem, da bo:

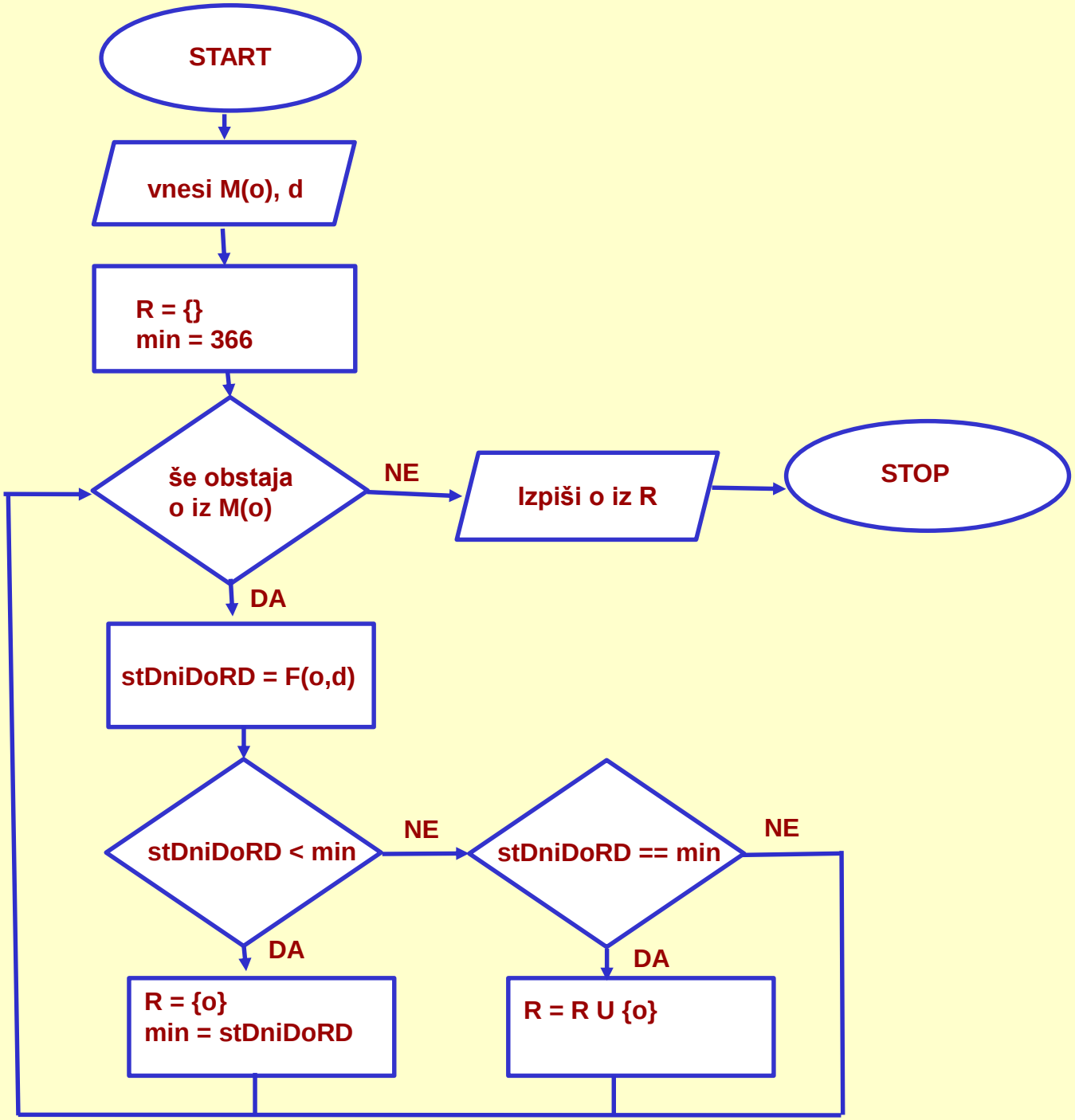
- ***Iskalni***
- ***Odločitveni***
- ***Preštevalni***
- ***Naštevalni***
- ***Optimizacijski***

3. naloga

Enak problem kot pri prvi nalogi.

a) Opišite algoritem še z:

- *diagramom poteka*
- *psevdokodo*
- *programskim jezikom Java*


START
vnesi M(o), d
R = {}
min = 366
še obstaja
o iz M(o)
NE
Izpiši o iz R
STOP
DA
stDniDoRD = F(o,d)
stDniDoRD < min
NE
stDniDoRD == min
NE
DA
DA
R = {o}
min = stDniDoRD
R = R U {o}

```
funkcija prviRD(M(o), d) {
   R={};
   min = 366;
   for (vsak o iz M(o)) {
      stDniDoRD = F(o, d);
      // stDniDoRD =
      //število dni od datuma d do rojstnega dneva osebe o
      if (stDniDoRD < min) {
         R={o};
         min=stDniDoRD;
       }
       else if (stDniDoRD==min) {
           R=R U {o};
        }
     }
     izpiši prvi o iz R;
}
```

```
// razred Datum: int dan, int mesec, int leto
int zapDan() {
   final int[] DOL_MES = {31,29,31,30,31, …., 31};
   int st = dan;
   for (int m=1; m<mesec;m++)
      st+=DOL_MES[m-1];
    return st;
}

// razred Oseba: String ime, String priimek, Datum datumR
int funStDni(Datum d) {
  return (LETO_DNI + datumR.zapDan() – d.zapDan()) % LETO_DNI;
}


public static final int LETO_DNI = 366;

public static Oseba prviRD(List<Oseba> m, Datum d) {
   List<Oseba> r = new ArrayList<>();
   int min = LETO_DNI, stDniDoRD;
   for (Oseba o : m) {
      stDniDoRD = o.funStDni(d);
      if (stDniDoRD < min) {
         r.clear();
         r.add(o);
         min=stDniDoRD;
       }
       else if (stDniDoRD==min) {
           r.add(o);
        }
     }
     return r.get(0);
}
```

4. naloga

Problem: Denimo, da je n mest medsebojno povezanih med seboj z letalskimi povezavami po principu vsak z vsakim. Iz prvega mesta želimo obiskati vsa ostala mesta tako, da nobenega mesta ne obiščemo več kot enkrat.

a) Narišite model (naloge) problema!
b) Za kakšen vrste problem gre?
c) Koliko rešitev obstaja?
d) Opišite preprost algoritem v psevdokodi!

```
funkcija poletiDoVsehMest( M(m), pm ) {
   O = {pm};
   N = M(m) \ {pm};
   R = pm;
   dokler (N ni prazen) {
      izberi mesto iz N
      O = O U {mesto};
      N = N \ {mesto};
      R = R + “, ”+ mesto;
    }
    izpiši R;
}
```

5. naloga

Problem iz naloge 4 spremenimo tako, da dodamo podatek o razdalji med mesti, iščemo pa najkrajšo pot (iz prvega mesta do vseh ostalih brez ponovitev)!

a) Za kakšen vrste problem gre sedaj?
b) Koliko rešitev obstaja za konkretno nalogo?
c) Koliko rešitev obstaja v splošnem?

5. naloga

Teorija 1: Naloga za oddajo preko Učilnice

Rok: ponedeljek, 25. oktobra 2021 do 23.59

a) Določite konkretno nalogo velikosti 6!

- ***Narišite model za to nalogo!***

b) Opišite poljuben algoritem v psevdokodi!

- ***Zamislite si ga sami, lahko zelo neoptimalen algoritem.***
- ***Bisteven je primeren opis s psevdokodo.***

c) Predstavite sled tega algoritma!

- ***Popolna predstavitev sledi je veliko preobsežna.***
- ***Predstavite le nekaj primerov in nakažite izbor ustreznega rezultata.***

6. naloga

Kratko opišite naslednje metode snovanja algoritmov in zanje podajte en tipičen primer problema:

a) Groba sila
b) Izčrpno preiskovanje
c) Sestopanje
d) Metoda razveji in omeji
e) Požrešna metoda
f) Metoda deli in vladaj

Kviz: Algoritmi

Kaj od naštetega velja za algoritem?

Izberite enega ali več odgovorov:

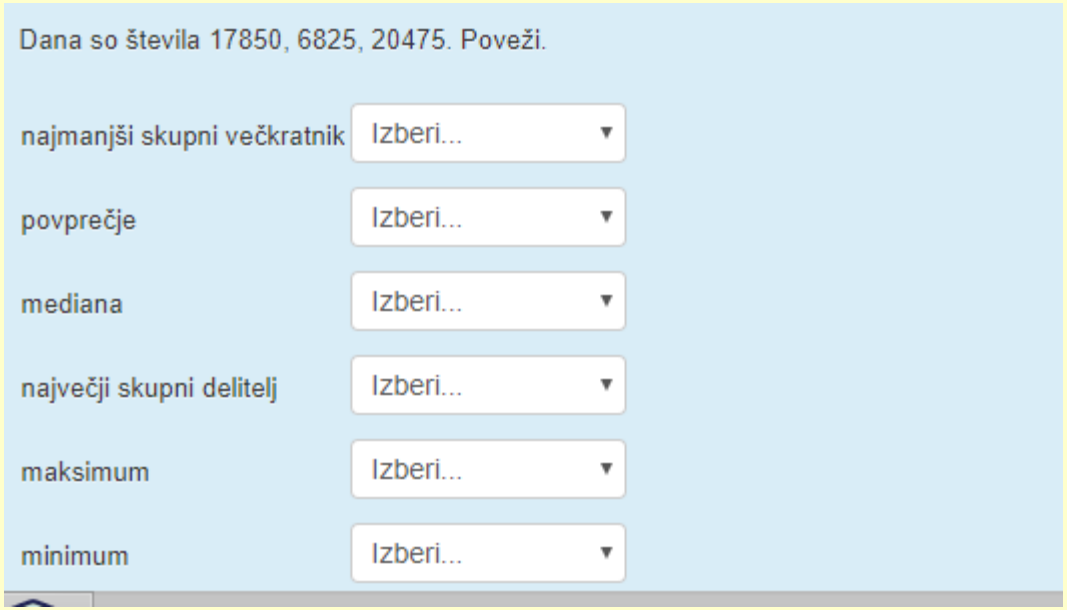
- [ ] je jasen in nedvoumen postopek
- [ ] ga lahko izvaja le računalnik
- [ ] je posebna vrsta ritma za petje računalniških duhovnih pesmi
- [ ] je mehaničen postopek

Kviz

Kaj lahko odgovori algoritem pri naslednjih vrstah problemov?

odločitveni problem	Izberi...
(konstrukcijski) optimizacijski problem	Izberi...
preštevalni problem	Izberi...

Kviz


Dana so števila 17850, 6825, 20475. Poveži.
najmanjši skupni večkratnik
Izberi...
povprečje
Izberi...
mediana
Izberi...
največji skupni delitelj
Izberi...
maksimum
Izberi...
minimum
Izberi...

Kviz

S katerimi od naštetih načinov je moč (ni pa nujno enostavno) opisovati algoritme?

Izberite enega ali več odgovorov:

- [ ] slovenščina
- [ ] krogi v žitu
- [ ] lambda račun
- [ ] Morsejeva koda
- [ ] JavaScript

Kviz

Največji skupni delitelj dveh praštevil je:

Odgovor:

Kviz

V algoritmi "Eratostenovo sito" najprej iz seznama odstranimo vsa soda števila (razen 2).

Izberite en odgovor:

- Drži
- Ne drži

Kviz

Urejanje seznama 3, 1, 3, 1, 5, 9 je **računski problem**?

Izberite en odgovor:

- Drži
- Ne drži

Kviz

Sled algoritma je izpis vrednosti pomembnih spremenljivk in drugih podatkov tekom izvajanja.

Izberite en odgovor:

- Drži
- Ne drži

Kviz

S pomočjo indukcije dokaži pravilnost dvojiškega iskanja. Poveži.

```
fun binarySearch(a, left, right, key) is
    if left > right then return -1
    mid = left + (right - left) / 2
    if (key < a[mid]) then
        return binarySearch(a, left, mid - 1)
    if (key > a[mid]) then
        return binarySearch(a, mid + 1, right)
    return mid
```

Če *key* = *a*[*mid*], potem vrnemo *mid*, kar je pravilno, ker je *mid* indeks elementa z vrednostjo *key*. Izberi...

Če *binarySearch* deluje pravilno za *n*, potem deluje pravilno tudi za *n* + 1. Izberi...

Če je left > *right*, potem je dolžina tabele *n* = *right* - *left* + 1 <= 0. Algoritem vrne -1, kar je pravilno, ker v prazni tabeli, ne more biti iskanega elementa. Izberi...

Če *left* <= *right*, potem je dolžina tabele *n* > 0 (vsebuje vsaj en element). Pri tem ločimo tri primere. Izberi...

Primer, ko je *key* > *a*[*mid*] je simetričen primeru *key* < *a*[*mid*]. Naredi za vajo sam. Izberi...

Algoritem *binarySearch* deluje pravilno (vrne -1, če iskanega elementa v tabeli ni, sicer pa vrne indeks elementa) za vse dolžine tabele *n* = *right* - *left* + 1. Izberi...

Če *key* < *a*[*mid*], potem *key* < *a*[*i*] za vsak *i* >= *mid*, ker je tabela urejena. Torej lahko element iščemo v *a*[*left*, ..., *mid* - 1]. Dolžina te tabele je *mid* - 1 - *left* + 1 = (*right* - *left*) / 2, kar je manjše od dolžine prvotne tabele, ki je *right* - *left* + 1. Iskani element je torej v manši podtabeli, za katero pa algoritem pravilno deluje po induktivni predpostavki. Izberi...

Kviz

Algoritem, ki na vhodu prejme število, testiramo s testom, ki traja toliko časa, kolikor je enic v dvojiški predstavitvi števila. Npr. 5 = 101b, torej test za število 5 traja 2 sekundi. Želimo testirati vsa 8 bitna števila, koliko sekund bo trajalo celotno testiranje?

Odgovor:

Kviz

Iz števil od 1 do 100 zgeneriramo vsa možna zaporedja (različnih števil) dolžine 4. Koliko je takih zaporedij?

Odgovor:

Kviz

Zančna invarianta za iskanje minimuma v tabeli elementov je naslednja trditev: v *i*-ti (začenši z 0) iteraciji je **trenutni minimum** *min* enak _____ vrednosti izmed prvih *i* + 1 elementov tabele.

Primer algoritma:

```
min = MAX
for i = 0 to n - 1 do
    if a[i] < min then min = a[i]
```

Kviz

Naslednja funkcija rekurzivno računa Fibbonacijeva števila.

```
fun fib(n) is
    return fib(n-1) + fib(n+2)
```

Izberite en odgovor:

- Drži
- Ne drži

Kviz

Dober nabor testnih primerov za testiranje pravilnosti algoritma vključuje robne primere.

Izberite en odgovor:

- Drži
- Ne drži