19. srednješolsko tekmovanje ACM v znanju računalništva

Šolsko tekmovanje

26. januarja 2024

NASVETI ZA TEKMOVALCE

Naloge na tem šolskem tekmovanju pokrivajo širok razpon težavnosti, tako da ni nič hudega, če ne znaš rešiti vseh.

Nekatere naloge so tipa **napiši program** (ali **napiši podprogram**), nekatere pa tipa **opiši postopek**. Pri slednjih ti ni treba pisati programa ali podprograma v kakšnem konkretnem programskem jeziku, ampak lahko postopek opišeš tudi kako drugače: z besedami (v naravnem jeziku), psevdokodo (glej spodaj), diagramom poteka itd. Glavno je, da je tvoj opis dovolj natančen, jasen in razumljiv, tako da je iz njega razvidno, da si dejansko našel in razumel pot do rešitve naloge.

Psevdokodi pravijo včasih tudi strukturirani naravni jezik. Postopek opišemo v naravnem jeziku, vendar opis strukturiramo na podoben način kot pri programskih jezikih, tako da se jasno vidi strukturo vejitev, zank in drugih programskih elementov.

Primer opisa postopka v psevdokodi: recimo, da imamo zaporedje besed in bi ga radi razbili na več vrstic tako, da ne bo nobena vrstica preširoka.

```
naj bo trenutna vrstica prazen niz;
pregleduj besede po vrsti od prve do zadnje:
    če bi trenutna vrstica z dodano trenutno besedo (in presledkom
    pred njo) postala predolga,
        izpiši trenutno vrstico in jo potem postavi na prazen niz;
    dodaj trenutno besedo na konec trenutne vrstice;
če trenutna vrstica ni prazen niz, jo izpiši;
```

(Opomba: samo zato, ker je tu primer psevdokode, to še ne pomeni, da moraš tudi ti pisati svoje odgovore v psevdokodi.)

Če pa v okviru neke rešitve pišeš izvorno kodo programa ali podprograma, obvezno poleg te izvorne kode v nekaj stavkih opiši, kako deluje (oz. naj bi delovala) tvoja rešitev in na kakšni ideji temelji.

Pri ocenjevanju so vse naloge vredne enako število točk. Svoje odgovore dobro utemelji. Prizadevaj si predvsem, da bi bile tvoje rešitve pravilne, ob tem pa je zaželeno, da so tudi čim bolj učinkovite (take dobijo več točk kot manj učinkovite). Za manjše sintaktične napake se načeloma ne odbije veliko točk. Priporočljivo in zaželeno je, da so tvoje rešitve napisane pregledno in čitljivo. Če je na listih, ki jih oddajaš, več različic rešitve za kakšno nalogo, jasno označi, katera je tista, ki naj jo ocenjevalci upoštevajo.

Če naloga zahteva branje ali obdelavo vhodnih podatkov, lahko tvoja rešitev (če v nalogi ni drugače napisano) predpostavi, da v vhodnih podatkih ni napak (torej da je njihova vsebina in oblika skladna s tem, kar piše v nalogi).

Nekatere naloge zahtevajo branje podatkov s standardnega vhoda in pisanje na standardni izhod. Za pomoč je tu nekaj primerov programov, ki delajo s standardnim vhodom in izhodom:

- Program, ki prebere s standardnega vhoda dve števili in izpiše na standardni izhod njuno vsoto:

```
program BranjeStevil;
var i, j: integer;
begin
  ReadLn(i, j);
  WriteLn(i, ' + ', j, ' = ', i + j);
end. {BranjeStevil}
```

```
#include <stdio.h>
int main() {
  int i, j; scanf("%d %d", &i, &j);
  printf("%d + %d = %d\n", i, j, i + j);
  return 0;
}
```

- Program, ki bere s standardnega vhoda po vrsticah, jih šteje in prepisuje na standardni izhod, na koncu pa izpiše še skupno dolžino:

```
program BranjeVrstic;
var s: string; i, d: integer;
begin
  i := 0; d := 0;
  while not Eof do begin
    ReadLn(s);
    i := i + 1; d := d + Length(s);
    WriteLn(i, '. vrstica: "', s, '"');
  end; {while}
  WriteLn(i, ' vrstic, ', d, ' znakov.');
end. {BranjeVrstic}
```

```
#include <stdio.h>
#include <string.h>
int main() {
  char s[201]; int i = 0, d = 0;
  while (gets(s)) {
    i++; d += strlen(s);
    printf("%d. vrstica: \"%s\"\n", i, s);
  }
  printf("%d vrstic, %d znakov.\n", i, d);
  return 0;
}
```

Opomba: C-jevska različica gornjega programa predpostavlja, da ni nobena vrstica vhodnega besedila daljša od dvesto znakov. Funkciji gets se je v praksi bolje izogibati, ker pri njej nimamo zaščite pred primeri, ko je vrstica daljša od naše tabele s. Namesto gets bi bilo bolje (in varneje) uporabiti fgets ali fscanf; vendar pa za rešitev naših tekmovalnih nalog zadošča tudi gets.

- Program, ki bere s standardnega vhoda po znakih, jih prepisuje na standardni izhod, na koncu pa izpiše še število prebranih znakov (ne vštevši znakov za konec vrstice):

```
program BranjeZnakov;
var i: integer; c: char;
begin
  i := 0;
  while not Eof do begin
    while not Eoln do
      begin Read(c); Write(c); i := i + 1 end;
    if not Eof then begin ReadLn; WriteLn end;
  end; {while}
  WriteLn('Skupaj ', i, ' znakov.');
end. {BranjeZnakov}
```

```
#include <stdio.h>

int main() {
  int i = 0, c;
  while ((c = getchar()) != EOF) {
    putchar(c); if (i != '\n') i++;
  }
  printf("Skupaj %d znakov.\n", i);
  return 0;
}
```

Še isti trije primeri v pythonu:

```
# Branje dveh števil in izpis vsote:
import sys

a, b = sys.stdin.readline().split()
a = int(a); b = int(b)
print "%d + %d = %d" % (a, b, a + b)
```

```
# Branje standardnega vhoda po vrsticah:
import sys

i = d = 0
for s in sys.stdin:
  s = s.rstrip('\n') # odrežemo znak za konec vrstice
  i += 1; d += len(s)
  print "%d. vrstica: \"%s\"" % (i, s)
print "%d vrstic, %d znakov." % (i, d)
```

```
# Branje standardnega vhoda znak po znak:
import sys

i = 0
while True:
  c = sys.stdin.read(1)
  if c == "": break # EOF
  sys.stdout.write(c)
  if c != '\n': i += 1
print "Skupaj %d znakov." % i
```

Še isti trije primeri v javi:

```
// Branje dveh števil in izpis vsote:
import java.io.*;
import java.util.Scanner;

public class Primer1
{
   public static void main(String[] args) throws IOException
   {
      Scanner fi = new Scanner(System.in);
      int i = fi.nextInt(); int j = fi.nextInt();
      System.out.println(i + " + " + j + " = " + (i + j));
   }
}
```

```
// Branje standardnega vhoda po vrsticah:
import java.io.*;

public class Primer2
{
   public static void main(String[] args) throws IOException
   {
      BufferedReader fi = new BufferedReader(new InputStreamReader(System.in));
      int i = 0, d = 0; String s;
      while ((s = fi.readLine()) != null) {
         i++; d += s.length();
         System.out.println(i + ". vrstica: \"" + s + "\""); }
      System.out.println(i + " vrstic, " + d + " znakov.");
   }
}
```

```
// Branje standardnega vhoda znak po znak:
import java.io.*;

public class Primer3
{
   public static void main(String[] args) throws IOException
   {
      InputStreamReader fi = new InputStreamReader(System.in);
      int i = 0, c;
      while ((c = fi.read()) >= 0) {
         System.out.print((char) c); if (c != '\n' && c != '\r') i++; }
      System.out.println("Skupaj " + i + " znakov.");
   }
}
```

19. srednješolsko tekmovanje ACM v znanju računalništva

Šolsko tekmovanje

26. januarja 2024

NALOGE ZA ŠOLSKO TEKMOVANJE

Svoje odgovore dobro utemelji. Če pišeš izvorno kodo programa ali podprograma, **OBVEZNO** tudi v nekaj stavkih z besedami opiši idejo, na kateri temelji tvoja rešitev. Če ni v nalogi drugače napisano, lahko tvoje rešitve predpostavljajo, da so vhodni podatki brez napak (da ustrezajo formatu in omejitvam, kot jih podaja naloga). Zaželeno je, da so tvoje rešitve poleg tega, da so pravilne, tudi učinkovite (bolj učinkovite rešitve dobijo več točk). Nalog je pet in pri vsaki nalogi lahko dobiš od 0 do 20 točk.

Rešitve bodo objavljene na `https://rtk.ijs.si/`.

1. Podatkovni center

V podatkovnem centru se nahajajo omare s strežniki, ki so oštevilčene od 1 do n. Na vratih v center, ki so označena z 0, in vratih za vsako omaro je nameščen senzor, ki sporoča varnostnemu sistemu, kdaj so bila vrata odprta ali zaprta. Varnostni sistem je treba dopolniti tako, da javi napako, če:

- se bodo odprta vrata odprla ali zaprta vrata zaprla,
- je katera od omar odprta, ko se vhodna vrata zaprejo.

Ko se vhodna vrata v center zaprejo, je na sistem poslana datoteka s številom vrat v centru (vključuje vhodna vrata) in vsakim dogodkom, ki so ga senzorji zaznali. Vsak dogodek je v svoji vrstici in je opisan z dvema številoma (ločenima s presledkom): številko vrat (od 0 do n, pri čemer 0 pomeni vhodna vrata, ostale pa so omare) in številom 0 ali 1, ki pove, ali so se vrata odprla (1) ali zaprla (0).

Napiši program, ki bo preveril podatke v datoteki in izpisal sporočilo ob nepravilnem dogodku. Če se vhodna vrata zaprejo brez napake, mora program izpisati številke vseh omar, ki so bile pred tem vsaj enkrat odprte. Tvoj program lahko bere podatke s standardnega vhoda ali pa iz datoteke `vhod.txt` (karkoli ti je lažje). Predpostaviš lahko, da je $n \leq 100$, da so bila pred prvim dogodkom v datoteki vsa vrata zaprta in da se zapiranje vhodnih vrat pojavi v zadnji vrstici datoteke, prej pa ne.

Primer datoteke (komentarji na desni niso del datoteke):

Datoteka	Komentar
`4`	število vrat ($n+1$)
`0 1`	vhodna vrata se odprejo
`1 1`	omara 1 se odpre
`3 1`	omara 3 se odpre
`1 0`	omara 1 se zapre
`3 0`	omara 3 se zapre
`2 1`	omara 2 se odpre
`2 0`	omara 2 se zapre
`0 0`	vhodna vrata se zaprejo

Pri tej vhodni datoteki bi moral program ugotoviti, da ni bilo napak, vsaj enkrat odprte pa so bile omare 1, 2 in 3.

2. Plesalci

Na odru je n plesalcev, ki plešejo ples v vrsti. Mesta, kjer stojijo pred nastopom, so oštevilčena s števili od 1 do n. Ples je razdeljen na m dejanj, vsako dejanje pa lahko opišemo s seznamom, kjer i-to število v njem pove, na katero mesto se v tem dejanju premakne oseba, ki je ob začetku dejanja stala na mestu i. Ker so plesalci izkušeni, lahko celotno dejanje izvedejo vsi naenkrat.

Po plesu plesalci obdržijo vrstni red in se pripravijo na ponovno izvedbo plesa pred novo množico ljudi. **Napiši program** (ali podprogram oz. funkcijo), ki izračuna, kolikokrat morajo izvesti ples, da se bo (ob koncu zadnje izvedbe celotnega plesa) vrstni red povrnil na prvotnega. Predpostaviš lahko, da sta m in n manjša od 1000 in da tudi potrebno število izvajanj plesa ne bo večje od 1000. Zaželeno je, da je tvoj postopek kolikor toliko učinkovit. Podrobnosti tega, v kakšni obliki tvoj (pod)program dobi ali prebere vhodne podatke, si izberi sam in jih v svoji rešitvi tudi opiši.

Primer: recimo, da imamo $n = 5$ plesalcev in $m = 2$ dejanji, opisani s seznamoma $[3, 4, 1, 5, 2]$ in $[4, 3, 2, 1, 5]$. Potem plesalec, ki je bil ob začetku plesa na mestu 1, pride po prvem dejanju na mesto 3, od tam pa v drugem dejanju na mesto 2. Ob drugi izvedbi plesa pride ta plesalec v prvem dejanju z mesta 2 na mesto 4, v drugem dejanju pa od tam na mesto 1. Podobno bi se dalo razmišljati še naprej in tudi za druge plesalce. Pri tem konkretnem primeru se izkaže, da je treba ples izvesti šestkrat, preden pridejo vsi plesalci nazaj na svoj začetni položaj.

3. Nizi

Napiši podprogram (oz. funkcijo) `NastejNize(s, k)`, ki kot parametra dobi niz s in celo število k. Predpostaviš lahko, da bo niz s sestavljen le iz malih črk angleške abecede, pri čemer se nobena ne pojavi v s več kot enkrat. Če dolžino s-ja označimo z n, bo veljalo $1 \leq n \leq k \leq 30$. Tvoj podprogram naj izpiše vse take nize t, ki so dolgi natanko k znakov, vsi ti znaki so iz niza s in vsak znak niza s se pojavi vsaj enkrat v nizu t. Nize lahko izpišeš v poljubnem vrstnem redu (vsakega natanko enkrat), zaželeno pa je, da je tvoja rešitev učinkovita in ne generira nizov po nepotrebnem.

Primer: če dobimo $s =$ `ga` in $k = 3$, moramo izpisati šest nizov: `aag`, `aga`, `gaa`, `agg`, `gag`, `gga` (ne nujno v tem vrstnem redu).

4. Zavarovanje

Pošta želi pridobiti zaupanje javnosti in se zato odloči, da bo pošiljke zavarovala in v primeru, ko jih ne dostavi pravočasno, pošiljateljem izplačala odškodnino.

Želijo pa zavarovati tudi manj vredne pošiljke, zato se odločijo nastaviti neko minimalno vrednost odškodnine o, ki bo v primeru prepozne dostave izplačana ne glede na vrednost pošiljke. Pošta torej v primeru prepozne dostave pošiljke izplača maksimum minimalne vrednosti odškodnine in vrednosti pošiljke.

Pošto zanima, kako izbira minimalne odškodnine o vpliva na znesek, ki ga morajo v najslabšem primeru (če izgubijo vso pošto v sistemu) izplačati pošiljateljem. **Opiši postopek** (ali napiši program ali podprogram, če ti je lažje), ki izračuna ta znesek za različne vrednosti o. Kot vhodne podatke tvoj postopek dobi: število pošiljk v sistemu (n); vrednosti teh pošiljk ($v_1, \ldots, v_n$); število minimalnih odškodnin, o katerih pošta razmišlja (m); in višine teh minimalnih odškodnin ($o_1, \ldots, o_m$). Tvoj postopek naj za vsako o_i (za $i = 1, \ldots, m$) izračuna skupno vrednost plačane odškodnine, če se izgubi vseh n pošiljk. Zaželeno je, da je tvoj postopek čim učinkovitejši.

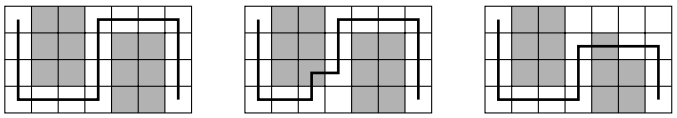
Primer: če je $o = 10$ in imamo tri pošiljke z vrednostmi 1, 20 in 25, za odškodnino velja:

- odškodnina 1. pošiljke je 10, saj je vrednost pošiljke 1, kar je manjše od minimalne odškodnine;
- odškodnina 2. pošiljke je 20, saj je vrednost pošiljke 20, kar je večje od minimalne odškodnine;
- odškodnina 3. pošiljke je 25, saj je vrednost pošiljke 25, kar je večje od minimalne odškodnine.

V najslabšem primeru (če izgubi vse pošiljke) mora torej pošta plačati $10 + 20 + 25 = 55$ enot denarja.

5. Najkrajše poti

Dana je pravokotna karirasta mreža, ki ima h vrstic in w stolpcev. Nekatera polja na mreži so prehodna, na drugih pa so ovire (in so zato neprehodna). Po čim krajši poti želimo priti od zgornjega levega do spodnjega desnega kota mreže s premiki gor, dol, levo in desno. Polji v zgornjem levem in spodnjem desnem kotu sta zagotovo prehodni. Na poti lahko eno oviro odstranimo, da si morebiti skrajšamo razdaljo (ni pa to obvezno). **Opiši postopek** (ali napiši program ali podprogram, če ti je lažje), ki kot vhodni podatek dobi opis mreže in izračuna dolžino najkrajše take poti (ali pa ugotovi, da taka pot sploh ne obstaja). Podrobnosti tega, kako je mreža opisana, si izberi sam in jih v svoji rešitvi tudi opiši.



Primer: gornje slike kažejo tri primere mrež, pri čemer beli kvadratki predstavljajo prehodna polja, sivi pa neprehodna. Debela črta kaže eno od možnih najkrajših poti; kjer prečka sivo polje, to pomeni, da oviro na njem odstranimo. Na levi in srednji sliki je obakrat ista mreža, pri kateri obstaja več enako dobrih najkrajših poti (dolgih po 15 korakov); tu sta prikazani dve izmed njih. Na desni sliki je drugačna mreža, pri kateri je najkrajša pot dolga 13 korakov (tu prikazana pot je tudi edina te dolžine).