2018 Natjecanje iz informatike

9. veljače 2018.

Županijska razina 2018 / Osnovna škola (5. i 6. i 7. i 8. razred)

Primjena algoritama OŠ

OPISI ALGORITAMA

Agencija za odgoj i obrazovanje
Education and Teacher Training Agency

HRVATSKI SAVEZ INFORMATIČARA

Ministarstvo znanosti, obrazovanja i sporta

HRVATSKA ZAJEDNICA TEHNIČKE KULTURE

5.1. Zadatak: Modro

Autor: Nikola Dmitrović

Ukupan broj golova koji su postigli Vukodlaci dobije se kao zbroj golova koje su postigli u prvom i u drugom dijelu utakmice. Kao što se u tekstu zadatka garantira, taj će broj uvijek biti veći ili jednak od broja golova koje su Vilenjaci postigli u prvom dijelu utakmice. Razlika tih brojeva je traženi broj iz teksta zadatka.

Programski kod (pisan u Python 3.x)

```
Vu = int(input())
Vi = int(input())
X = int(input())
Vu = Vu + X
print(Vu - Vi)
```

Potrebno znanje: naredba učitavanja, matematička operacija oduzimanja

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
96	**61**	**67.45%**

5.2. Zadatak: Pink

Autor: Nikola Dmitrović

Zamislimo da smo u Zagrebu na dan D u vrijeme X. Da nema vremenske razlike između polaznog grada i Zagreba, kada bi avion trebao poletjeti da bi nakon L sati leta stigao u Zagreb? Odgovor je logičan, L sati prije X-tog sata. Jedini je problem što smo možda prešli iz današnjeg dana u dan prije.

Prije nego to provjerimo, trebamo odrediti vrijeme polijetanja aviona (X-L) prema lokalnom vremenu, tj. odraditi korekciju vremena s vremenskom razlikom (X-L+R).

Dobiveno vrijeme trebamo usporediti s 24 i s nula kako bi provjerili jesmo li prešli u dan prije ili u dan poslije. Pazi, zbog ograničenja na ulazne podatke moguće je otići dva dana prije dana D.

Programski kod (pisan u Python 3.x)

```
D = int(input())
X = int(input())
L = int(input())
R = int(input())
polazak = (X - L) + R
if polazak < 0: # 0 je još u trenutnom danu
    print(D - abs(polazak // 24))
    print(24 - (abs(polazak) % 24))
elif polazak >= 24: # 24 je sljedećem danu
```

```
    print(D + 1)
    print(polazak - 24)
else:
    print(D)
    print(polazak)
```

Zadatak možemo općenito riješiti tako da odredimo koliko je ukupno sati prošlo od početka mjeseca i nakon navedenih korekcija ponovo odrediti dan i sat.

Programski kod (pisan u Python 3.x)

```
D = int(input())
X = int(input())
L = int(input())
R = int(input())
minuta = D * 24 + X - L + R
print(minuta // 24)
print(minuta % 24)
```

Potrebno znanje: naredba odlučivanja

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
96	**4**	**24.40%**

5.3. Zadatak: Latinum Autor: Nikola Dmitrović

Tri dijela zadatka, tri šanse za dobijanje bodova.

Pitanje #1: Ukupan broj listića dobijemo po formuli: X*20*100 + Y*100 + Z

Pitanje #2: Prvo ponude trebamo iskazati kao broj ponuđenih listića te ih usporediti. Možemo ih uspoređivati i redom po polugama, trakama i listićima ali je prvi način jednostavniji.

Pitanje #3: Kako smo već odredili vrijednosti ponuda iskazane u listićama, oduzmimo te dvije vrijednosti i odredimo kolika je ta razlika iskazana u polugama latinuma. I dodajmo jedan da nadvisimo pobjedničku ponudu.

Programski kod (pisan u Python 3.x)

```
X = int(input())
Y = int(input())
Z = int(input())
RP = int(input())
RT = int(input())
RL = int(input())
```

```
ZP = int(input())
ZT = int(input())
ZL = int(input())

# pitanje #1
print(X * 20 * 100 + Y * 100 + Z)

# pitanje #2
rom = RP * 20 * 100 + RT * 100 + RL
zek = ZP * 20 * 100 + ZT * 100 + ZL
if rom > zek:
    print("ROM")
else:
    print("ZEK")

# pitanje #3
print (abs(rom - zek) // 2000 +1)
```

Potrebno znanje: naredba odlučivanja

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
96	**5**	**28.10%**

6.1. Zadatak: Zvijezda — Autor: Nikola Dmitrović

Višestrukom naredbom odlučivanja i dobrim postavljanjem uvjeta možemo odrediti kojoj skupini pripada zadana zvijezda. Nužno je dobro paziti na rubne temperature.

Programski kod (pisan u Python 3.x)

```
T = int(input())
if T >= 29727:
    print("O")
elif T >= 9727:
    print("B")
elif T >= 7227:
    print("A")
elif T >= 5727:
```

```
    print("F")
elif T >= 4927:
    print("G")
elif T >= 3427:
    print("K")
elif T >= 2127:
    print("M")
```

Potrebno znanje: složena naredba odlučivanja

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
111	**80**	**86.89%**

6.2. Zadatak: Planet — Autor: Nikola Dmitrović

Kako vrijeme isto protječe na svim planetima, možemo broj zemaljski dana pretvoriti u broj sati te djeljenjem s duljinom dana za zadanoj planeti odrediti koliko je to vrijeme iskazano dana.

Da bi odredili traženi broj planeta, za svaku od njih trebamo odrediti koliko je Katarina stara na njima i provjeriti je li taj broj veći od X.

Programski kod (pisan u Python 3.x)

```
N = int(input())
P = int(input())
X = int(input())
planeti = [1408, 5832, 24, 25, 10, 11, 17, 16]

if P == 1:
    K = (N * 24) // planeti[0]
elif P == 2:
    K = (N * 24) // planeti[1]
elif P == 3:
    K = (N * 24) // planeti[2]
elif P == 4:
    K = (N * 24) // planeti[3]
elif P == 5:
    K = (N * 24) // planeti[4]
elif P == 6:
```

```
    K = (N * 24) // planeti[5]
elif P == 7:
    K = (N * 24) // planeti[6]
elif P == 8:
    K = (N * 24) // planeti[7]
print(K)
#drugi dio
koliko = 0
for i in planeti:
    if ((N * 24) // i) > X:
        koliko += 1
print(koliko)
```

Potrebno znanje: naredba odlučivanja, naredba ponavljanja (opcija), niz (opcija)

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
111	**35**	**54.29%**

6.3. Zadatak: Katar — Autor: Nikola Dmitrović

Prvi dio zadatka možemo riješiti simulirajući uvjete iz teksta zadatka. Zamislimo niz od N klima (doslovno, a i u programerskom smislu) koje su u startu sve ugašene. X puta ćemo daljinski usmjeriti prema nekoj od njih pazeći u kojem su odnosu signal koji daljinski šalje i stanje klime (radi/ne radi). Nakon svakog slanja trebamo promjeniti signal.

Programski kod (pisan u Python 3.x)

```
N = int(input())
X = int(input())

klime = [0] * N
signal = 1

for i in range(X):
    Si = int(input())

    if signal == 1 and klime[Si-1] == 0: klime[Si-1] = 1
    if signal == -1 and klime[Si-1] == 1: klime[Si-1] = 0
```

```
    signal *= -1
print(sum(klime))
```

Rješenje drugog dijela zadatka ovisi o stanju u kojem je ostao daljinski. Ako je sljedeći signal koji će daljinski poslati 'G' tada ćemo za svaku od ugašenih klima trebati točno dva signala (prvim gasimo ugašenu klimu, drugim je palimo), a ako je sljedeći signal 'P' tada ćemo za prvu klimu iskoristiti jedan signal (upali je), a za svaku sljedeću po dva.

```
# drugi dio
print(2 * (N - sum(klime)) - ((X + 1) % 2))
```

Potrebno znanje: naredba ponavljanja, liste/nizovi

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
111	**9**	**13.66%**

7.1. Zadatak: Rok — Autor: Nikola Dmitrović

Svaki od N datuma trebamo usporediti s današnjim datumom i za svakog ispisati odgovarajuću poruku. To možemo postići uspoređujući redom godine, pa mjesece i na kraju dane. Za drugi dio zadatka trebamo pronaći datum koji ima najmanju godinu, pa tada najmanji mjesec i na kraju najmanji dan.

Programski kod (pisan u Python 3.x)

```
N = int(input())
D, M, G = map(int, input().split())
L = [2020, 12, 30]

for i in range(N):
    Di, Mi, Gi = map(int, input().split())
    ok = 1
    if Gi < G:
        ok = 0
    elif Gi == G:
        if Mi < M:
            ok = 0
        elif Mi == M:
            if Di < D:
                ok = 0
    if ok:
```

```
        print('DA')
    else:
        print('NE')

    if [Gi, Mi, Di] < L:
        L = [Gi, Mi, Di]

print(L[2], L[1], L[0])
```

Potrebno znanje: naredba odlučivanja, naredba ponavljanja, algoritam traženje minimalnog elementa

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
57	**11**	**41.84%**

7.2. Zadatak: Temperatura — Autor: Marin Kišić

Na početku for petljom ćemo fiksirati neki dan i provjeriti jesu li baku boljela leđa na taj dan. Provjeru ćemo napraviti s još jednom for petljom koja ide od i-1 prema 0 tako dugo dok je A[j]=B[j] (j je varijabla druge for petlje). Kada smo našli najveći j takav da je A[j] != B[j], onda provjerimo prema uvjetu iz zadatka jesu li baku boljela leđa na dan i te ako jesu povećamo rješenje za 1.

Programski kod (pisan u C++)

```
#include <cstdio>
const int MAXN = 105;
int n, a[MAXN], b[MAXN];

int main() {
  scanf("%d",&n);
  for (int i = 0; i < n; i++) scanf("%d",&a[i]);
  for (int i = 0; i < n; i++) scanf("%d",&b[i]);

  int rj = 0;
  for (int i = 0; i < n; i++) {
    if (a[i] == b[i]) continue;
    for (int j = i - 1; j >= 0; j--) {
      if (a[j] != b[j]) {
        if (a[j] < b[j] && a[i] > b[i]) rj++;
```

```
        if (a[j] > b[j] && a[i] < b[i]) rj++;
        break;
      }
    }
  }
  printf("%d\n",rj);
  return 0;
}
```

Potrebno znanje: nizovi/liste

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
57	**11**	**23.16%**

7.3. Zadatak: Robot — Autor: Stjepan Požgaj

Za odgovor na prvo pitanje moramo samo simulirati robotovo kretanje po tablici te ispisivati slova na kojem se nalazi.

Odgovor na drugo pitanje najteži je dio ovog zadatka. Robot kreće s polja u X-tom retku i Y-tom stupcu. Svoj put završava u trenutku kad od prikupljenih slova može dobiti riječ K. Da bismo znali koji je to trenutak moramo u svakom koraku znati koja je najveća duljina prefiksa riječi K koju možemo dobiti. Ako je prije nekog koraka ta duljina jednaka X, tada tu duljinu povećavamo za jedan ako je X+1 slovo riječi K jednako slovu koje smo pokupili u tom koraku.

Za dodatne implementacijske detalje pogledajte službeno rješenje.

Programski kod (pisan u C++)

```
#include <iostream>
#include <cstdio>
#include <string>
#include <vector>
using namespace std;

#define FOR(i, a, b) for(int i = (a); i < (b); i++)
#define REP(i, n) FOR(i, 0, n)
#define TRACE(x) cerr << #x << " = " << x << endl

typedef long long int llint;
```

```
typedef pair<int, int> par;
#define X first
#define Y second
const int smjerX[4] = {0, 1, 0, -1};
const int smjerY[4] = {1, 0, -1, 0};
const int MAXN = 20, INF = 1e9;
int n, m;
int imam[4];
char mat[MAXN][MAXN];
vector<string> rijeci;
vector<int> naredbe;

int moze(par t) {
  if(t.X < 0 || t.X >= n || t.Y < 0 || t.Y >= m) return 0;
  if(mat[t.X][t.Y] == '#') return 0;
  return 1;
}

void ispisi_prvih_deset(par t) {
  int it = 0;
  for(int i = 0; i < 10; i++) {
    if(it == (int) naredbe.size()) it = 0;
    par kamo = par(t.X + smjerX[naredbe[it]], t.Y + smjerY[naredbe[it]]);
    it++;
    if(moze(kamo)) {
      t = kamo;
      cout << mat[t.X][t.Y];
    }
    else i--;
  }
  cout << endl;
}

int provjera(par t) {
  REP(i, 4)
    if(imam[i])
      if(moze(par(t.X + smjerX[i], t.Y + smjerY[i])))
```

```
        return 1;
  return 0;
}

int f(par t, int naj) {
  if(mat[t.X][t.Y] == '#') return INF;
  if(!provjera(t)) return INF;
  vector<int> niz((int) rijeci.size(), 0);
  int it = 0;
  for(int i = 0; i < naj + 5; i++) {
    if(it == (int) naredbe.size()) it = 0;
    par kamo = par(t.X + smjerX[naredbe[it]], t.Y + smjerY[naredbe[it]]);
    it++;
    if(moze(kamo)) {
      t = kamo;
      char slovo = mat[t.X][t.Y];
      REP(j, (int) niz.size()) {
        if(rijeci[j][niz[j]] ==  slovo) {
          niz[j]++;
          if(niz[j] == (int)rijeci[j].size()) return i + 1;
        }
      }
      if(!provjera(t)) return INF;
    }
    else i--;
  }
  return INF;
}

int main() {
  par poc;
  int broj_rijeci;
  string koraci;
  cin >> n >> m;
  REP(i, n) REP(j, m) cin >> mat[i][j];
  cin >> poc.X >> poc.Y;
  poc.X--, poc.Y--;
```

```
  cin >> koraci;
  REP(i, (int) koraci.size()) {
    char c = koraci[i];
    int sad = 0;
    if(c == 'R') sad = 0;
    if(c == 'D') sad = 1;
    if(c == 'L') sad = 2;
    if(c == 'U') sad = 3;
    naredbe.push_back(sad);
    imam[sad] = 1;
  }
  //cin >> broj_rijeci;
  broj_rijeci = 1;
  REP(i, broj_rijeci) {
    string s;
    cin >> s;
    rijeci.push_back(s);
  }
  ispisi_prvih_deset(poc);
  int x = f(poc, INF);
  cout << x << endl;
  return 0;
}
```

Potrebno znanje: matrice

Kategorija: simulacija

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
57	**4**	**13.16%**

8.1. Zadatak: Dobrila — Autor: Nikola Dmitrović

Zadatak možemo riješiti na razne načine igrajući se sa strukturama podataka koje su nam na raspolaganju. Pokažimo jedan način koji uključuje rječnike u Pythonu.

Programski kod (pisan u Python 3.x)

```
S = input()
L = S.split('#')

novcanice = {'DOBRILA':10, 'JELACIC':20, 'GUNDULIC':50}
novcanice.update({'MAZURANIC':100,'RADIC':200, 'MARULIC':500, 'STARCEVIC':1000})

likovi = novcanice.keys()

ukupno = 0
for i in L:
    for j in likovi:
        if i.find(j) == 0:
            ukupno += novcanice[j]
        if i.find(j) > 0:
            ukupno += int(i[:i.find(j)])*novcanice[j]
print(ukupno)
```

Potrebno znanje: string

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
53	**14**	**51.32%**

8.2. Zadatak: Milan — Autor: Vedran Kurdija

Za početak ćemo spremiti sve Milanove izjave kako bismo ih kasnije mogli koristiti. Proći ćemo po svim datumima od 9.2.2018. do 30.12.2100. te prebrojati koliko ih zadovoljava sve izjave. Po datumima možemo iterirati pomoću tri varijable: dan, mjesec i godina koje u svakom koraku iteracije ažuriramo kako bismo prešli na sljedeći dan. Za svaki od datuma možemo napraviti pomoćni niz u koji ćemo spremiti koliko puta se svaka od deset znamenaka pojavljuje u datumu. Nakon toga ćemo proći po svim Milanovim izjavama i provjeriti jesu li zadovoljene. Ako su sve izjave zadovoljene, uvećavamo rješenje za jedan. Za implementacijske detalje pogledajte kod.

Programski kod (pisan u C++)

```
#include <cstring>
```

```
#include <iostream>
using namespace std;

const int maxn = 55;
int n;
int oblik[maxn];
int x[maxn], k[maxn];
int br_znam[15];

void prebroji(int broj) {
    while (broj > 0) {
        int znam = broj % 10;
        br_znam[znam]++;
        broj /= 10;
    }
}
int main (void) {
    cin >> n;
    for (int i = 0; i < n; i++)
        cin >> oblik[i] >> x[i] >> k[i];
    int dan = 9;
    int mjesec = 2;
    int godina = 2018;
    int rjesenje = 0;
    while (godina < 2101) {
        memset(br_znam, 0, sizeof br_znam);
        prebroji(dan);
        prebroji(mjesec);
        prebroji(godina);
        bool ok = 1;
        for (int i = 0; i < n && ok == 1; i++) {
            if (oblik[i] == 1 && br_znam[x[i]] < k[i])
                ok = 0;
            if (oblik[i] == 2 && br_znam[x[i]] != k[i])
                ok = 0;
            if (oblik[i] == 3 && br_znam[x[i]] > k[i])
                ok = 0;
```

```
        }
        if (ok)
            rjesenje++;
        dan++;
        if (dan > 30) {
            dan = 1;
            mjesec++;
            if (mjesec > 12) {
                godina++;
                mjesec = 1;
            }
        }
    }
    cout << rjesenje << endl;
    return 0;
}
```

Potrebno znanje: for ili while petlja, nizovi

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
53	**6**	**18.11%**

8.3. Zadatak: Pepper Autor: Stjepan Požgaj

Za odgovor na prvo pitanje moramo samo simulirati robotovo kretanje po tablici te ispisivati slovo po slovo na kojem se nalazi.

Odgovor na drugo pitanje najteži je dio ovog zadatka. Robot kreće s polja u X-tom retku i Y-tom stupcu. Svoj put završava u trenutku kad od prikupljenih slova može dobiti riječ K. Da bismo znali koji je to trenutak moramo u svakom koraku znati koja je najveća duljina prefiksa riječi K koju možemo dobiti. Ako je prije nekog koraka ta duljina jednaka X, tada tu duljinu povećavamo za jedan ako je X+1 slovo riječi K jednako slovu koje smo pokupili u tom koraku.

Za odgovor na treće pitanje moramo pokušati odgovoriti na drugo pitanje krećući iz svake pozicije. Tu moramo paziti na slučaj u kojem nakon bilo koliko koraka ne možemo dobiti riječ K.

Za dodatne implementacijske detalje pogledajte službeno rješenje.

Programski kod (pisan u C++)

```
#include <iostream>
#include <cstdio>
```

```
#include <string>
#include <vector>
using namespace std;

#define FOR(i, a, b) for(int i = (a); i < (b); i++)
#define REP(i, n) FOR(i, 0, n)
#define TRACE(x) cerr << #x << " = " << x << endl

typedef long long int llint;
typedef pair<int, int> par;
#define X first
#define Y second
const int smjerX[4] = {0, 1, 0, -1};
const int smjerY[4] = {1, 0, -1, 0};
const int MAXN = 20, INF = 1e9;
int n, m;
int imam[4];
char mat[MAXN][MAXN];
vector<string> rijeci;
vector<int> naredbe;

int moze(par t) {
  if(t.X < 0 || t.X >= n || t.Y < 0 || t.Y >= m) return 0;
  if(mat[t.X][t.Y] == '#') return 0;
  return 1;
}

void ispisi_prvih_deset(par t) {
  int it = 0;
  for(int i = 0; i < 10; i++) {
    if(it == (int) naredbe.size()) it = 0;
    par kamo = par(t.X + smjerX[naredbe[it]], t.Y + smjerY[naredbe[it]]);
    it++;
    if(moze(kamo)) {
      t = kamo;
      cout << mat[t.X][t.Y];
    }
```

```
    else i--;
  }
  cout << endl;
}

int provjera(par t) {
  REP(i, 4)
    if(imam[i])
      if(moze(par(t.X + smjerX[i], t.Y + smjerY[i])))
        return 1;
  return 0;
}

int f(par t, int naj) {
  if(mat[t.X][t.Y] == '#') return INF;
  if(!provjera(t)) return INF;
  vector<int> niz((int) rijeci.size(), 0);
  int it = 0;
  for(int i = 0; i < naj + 5; i++) {
    if(it == (int) naredbe.size()) it = 0;
    par kamo = par(t.X + smjerX[naredbe[it]], t.Y + smjerY[naredbe[it]]);
    it++;
    if(moze(kamo)) {
      t = kamo;
      char slovo = mat[t.X][t.Y];
      REP(j, (int) niz.size()) {
        if(rijeci[j][niz[j]] ==  slovo) {
          niz[j]++;
          if(niz[j] == (int)rijeci[j].size()) return i + 1;
        }
      }
      if(!provjera(t)) return INF;
    }
    else i--;
  }
  return INF;
}
```

```
int main() {
  par poc;
  int broj_rijeci;
  string koraci;
  cin >> n >> m;
  REP(i, n) REP(j, m) cin >> mat[i][j];
  cin >> poc.X >> poc.Y;
  poc.X--, poc.Y--;
  cin >> koraci;
  REP(i, (int) koraci.size()) {
    char c = koraci[i];
    int sad = 0;
    if(c == 'R') sad = 0;
    if(c == 'D') sad = 1;
    if(c == 'L') sad = 2;
    if(c == 'U') sad = 3;
    naredbe.push_back(sad);
    imam[sad] = 1;
  }
  //cin >> broj_rijeci;
  broj_rijeci = 1;
  REP(i, broj_rijeci) {
    string s;
    cin >> s;
    rijeci.push_back(s);
  }
  ispisi_prvih_deset(poc);
  int x = f(poc, INF);
  cout << x << endl;
  par rj = poc;
  int sad = x;
  REP(i, n)
    REP(j, m) {
      int t = f(par(i, j), sad);
      if(t < sad || (t == sad && i < rj.X) || (t == sad && i == rj.X && j <
rj.Y)) {
```

```
            sad = t;
            rj = par(i, j);
          }
        }
    rj.X++, rj.Y++;
    cout << rj.X << " " << rj.Y << endl;
    return 0;
}
```

Potrebno znanje: matrice

Kategorija: simulacija

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
53	**1**	**11.97%**