2017 iz informatike Natjecanje

10. veljače 2017.

Županijska razina / 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: Istok

Autor: Nikola Dmitrović

Prvu rečenicu iz zadatka („Kada se licem okrenemo prema istoku, iza leđa nam je zapad. Lijevom rukom pokazujemo na sjever, a desnom na jug“) možemo jednostavno prevesti u odabrani programski jezik i tako si osigurati dio bodova koji su se mogli osvojiti na ovom zadatku.

Programski kod (pisan u Python 3.x)

```
O = input()
S = int(input())
# 0 - lijevo # 1 - desno # 2 - iza nas
if O == "I":
    if S == 0: print("S")
    if S == 1: print("J")
    if S == 2: print("Z")
```

Po uzoru na prvi dio koda trebamo pokriti i ostale kombinacije. Za to će biti najbolje da na papir nacrtamo situaciju iz prve rečenice zadatka i na taj način odredimo ostale slučajeve.

Programski kod (pisan u Python 3.x)

```
if O == "Z":
    if S == 0: print("J")
    if S == 1: print("S")
    if S == 2: print("I")
if O == "S":
    if S == 0: print("Z")
    if S == 1: print("I")
    if S == 2: print("J")
if O == "J":
    if S == 0: print("I")
    if S == 1: print("Z")
    if S == 2: print("S")
```

Potrebno znanje: naredba odlučivanja

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
117	**39**	**18,73**

5.2. Zadatak: Sto

Autor: Adrian Satja Kurdija

Brojeve učitavamo *for*-petljom koja ide od 1 do *N*. Da bismo mogli računati zbrojeve dvaju ili triju uzastopnih brojeva, nužno je pamtiti posljednja dva učitana broja A i B (u pomoćnim varijablama). Kada učitamo novi broj C, provjeravamo je li A + B + C = 100 ili B + C = 100. Ako jest, ispisujemo rješenje i prekidamo petlju (*break*) ili nekom pomoćnom varijablom označimo da je rješenje pronađeno da ne bismo poslije ispisali još neko rješenje. Nakon svakog koraka, posljednja dva učitana broja valja ažurirati: A i B poprimaju vrijednosti B i C.

Programski kod (pisan u Python 3.x)

```
n = int(input())
a, b = 0, 0
for i in range(n):
    c = int(input())
    if b + c == 100:
        print(b)
        print(c)
        break
    if a + b + c == 100:
        print(a)
        print(b)
        print(c)
        break
    a, b = b, c
```

Potrebno znanje: naredba ponavljanja

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
117	**4**	**5,89**

5.3. Zadatak: Trek

Autor: Adrian Satja Kurdija

Zadatak se može riješiti razmatrajući mnogo slučajeva, ali postoje kraća i elegantnija rješenja koja su otpornija na pogreške. Posao nam skraćuje nekoliko zaključaka.

- Dovoljno je promatrati samo nekoliko osnovnih slučajeva ovisno o tome na kojim se kružnicama (unutarnjoj ili vanjskoj) nalaze točke A i B.
- Ako se A i B obje nalaze na unutarnjoj kružnici, put će ići samo po toj kružnici.
- Ako su A i B susjedne točke na vanjskoj kružnici, put će ići po vanjskoj kružnici.

- U svim ostalim slučajevima, jednu ili obje točke A i B „spuštamo“ na unutarnju kružnicu i „spajamo“ ih putem po unutarnjoj kružnici.
- Radi jednostavnosti možemo uzeti da je A > B, zamjenjujući varijable A i B ako je A < B.
- Sada lako računamo broj hodnika između A i B koji se nalaze na istoj kružnici: uzimamo ili razliku A – B ili njezin komplement (ostatak kružnice) 7 – (A – B), ovisno što je manje.

Programski kod (pisan u Python 3.x)

```
a = int(input())
b = int(input())
if a < b:
   a, b = b, a
# Ako put ide samo po unutarnjoj:
if a <= 7 and b <= 7:
   print(10 * min(a - b, 7 - (a - b)))
# Ako put ide samo po vanjskoj:
elif a > 7 and b > 7 and (a - b == 1 or a - b == 6):
   print(30)
else:
   # S vanjske idemo na unutarnju...
   a -= 7
   put = 20
   if b > 7:
       b -= 7
       put += 20
   # ... i putujemo po unutarnjoj:
   if a < b:
       a, b = b, a
   put += 10 * min(a - b, 7 - (a - b))
   print(put)
```

Potrebno znanje: naredba odlučivanja

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
117	**0**	**14,74**

6.1. Zadatak: Utah

Autor: Nikola Dmitrović

Pretpostavimo na početku da želimo riješiti samo dio zadatka za pola bodova kako je to opisano u sekciji „Bodovanje“. Ako Utah može ostvariti pobjedu samo u slučaju kad izbacimo točnu jednu četvrtinu, tada to možemo riješiti tako da izbacimo svaku od četiri četvrtine i provjerimo kada će Utah odnijeti pobjedu.

Programski kod (pisan u Python 3.x)

```
J1, J2, J3, J4 = map(int, input().split())
B1, B2, B3, B4 = map(int, input().split())
# izbačena četvrta četvrtina
if J1 + J2 + J3 > B1 + B2 + B3:
    print(J1 + J2 + J3, B1 + B2 + B3)
# izbačena treća četvrtina
if J1 + J2 + J4 > B1 + B2 + B4:
    print(J1 + J2 + J4, B1 + B2 + B4)
# izbačena četvrt četvrtina
if J1 + J3 + J4 > B1 + B3 + B4:
    print(J1 + J3 + J4, B1 + B3 + B4)
# izbačena prva četvrtina
if J2 + J3 + J4 > B2 + B3 + B4:
    print(J2 + J3 + J4, B2 + B3 + B4)
```

Međutim, za riješiti problem u potpunosti potrebno je u rješenje ukomponirati algoritam za traženje najveće vrijednosti. Za svaku izbačenu četvrtinu treba provjeriti:

- hoće li Utah u tom slučaju pobijediti,
- ako hoće, tada treba provjeriti kojom će se to razlikom dogoditi,
- i tom slučaju provjeriti je li ta razlika veća od do tada najveće razlike (koja je u početku nula).
- Ako je veća, tada treba zapamtiti rezultat koji je doveo do toga.

Programski kod (pisan u Python 3.x)

```
J1, J2, J3, J4 = map(int, input().split())
B1, B2, B3, B4 = map(int, input().split())
razlika = 0
# izbačena četvrta četvrtina
if J1 + J2 + J3 > B1 + B2 + B3:          # ako je Utah pobijedio..
    Utah = J1 + J2 + J3                  # Utah=broj postignutih koševa Utaha
    Bulls = B1 + B2 + B3                 # Bulls=broj postignutih koševa Bullsa
    if Utah - Bulls > razlika:           # ako je razlika veća..
        razlika = Utah - Bulls           # razlika dobiva novu vrijednost
        X = Utah                         # zapamtimo broj koševa za Utah
```

```
        Y = Bulls                          # zapamtimo broj koševa za Bulls
# izbačena treća četvrtina
if J1 + J2 + J4 > B1 + B2 + B4:
    Utah = J1 + J2 + J4
    Bulls = B1 + B2 + B4
    if Utah - Bulls > razlika:
        razlika = Utah - Bulls
        X = Utah
        Y = Bulls
# izbačena druga četvrtina
if J1 + J3 + J4 > B1 + B3 + B4:
    Utah = J1 + J3 + J4
    Bulls = B1 + B3 + B4
    if Utah - Bulls > razlika:
        razlika = Utah - Bulls
        X = Utah
        Y = Bulls
# izbačena prva četvrtina
if J2 + J3 + J4 > B2 + B3 + B4:
    Utah = J2 + J3 + J4
    Bulls = B2 + B3 + B4
    if Utah - Bulls > razlika:
        razlika = Utah - Bulls
        X = Utah
        Y = Bulls
print(X, Y)
```

Zadatak se mogao riješiti i puno kraće ako bi se koristila lista i neke njene metode.

Programski kod (pisan u Python 3.x)

```
j = list(map(int, input().split()))
b = list(map(int, input().split()))
r = []
for i in range(4):
    r.append(sum(j) - j[i] - (sum(b) - b[i]))
i = r.index(max(r))
print(sum(j) - j[i], sum(b) - b[i])
```

Potrebno znanje: naredba odlučivanja, algoritam određivanja najveće vrijednosti

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
79	**13**	**18,16**

6.2. Zadatak: Jard — Autor: Nikola Dmitrović

Zadana je tablica odnosa između mjernih jedinica:

	palac	**stopa**	**jard**	**milja**
kilometar	39370.1	3280.84	1093.61	0.621371

Jasno je da su zadani samo odnosi između kilometra s jedne strane te palca, stope, jarda i milje s druge strane. Da bi odredili odnos između palca, stope, jarda i milje morat ćemo koristi kilometar kao pomoćnu mjeru. Za to nam treba malo matematike ili dobro razmišljanje.

Npr., odredimo odnos između jednog palca i jednog jarda tj. želimo izračunati:

$$1\ palac = ?\ jarda$$

Znamo da vrijedi:

$$1\ kilometar = 39370.1\ palca$$
$$1\ kilometar = 1093.61\ jarda$$

Lijeve strane su jednake, izjednačimo onda i desne strane izraza:

$$39370.1\ palca = \ 1093.61\ jarda$$

Uz malu pomoć jednostavne matematike lako odredimo da je traženi odnos:

$$1\ palac = \frac{1093.61}{39370.1}\ jarda$$
$$1\ palac = 0.027777\ jarda$$

Iz ovog jednostavnog izvoda možemo odrediti opću formulu koju možemo koristiti za određivanje svih omjera.

Zbog rečenice iz zadatka „Tvoje rješenje smatrat će se točnim ako se od službenog razlikuje za manje od 0.001 ($|tvoje - službeno| \leq 0.001$)“ dovoljno je ispisati rješenje sa svim decimalama onako kako to programski jezik omogućuje.

Programski kod (pisan u Python 3.x)

```
D = int(input())
N = int(input())
M = int(input())
odnosi = [1, 39370.1, 3280.84, 1093.61, 0.621371]
print(D * odnosi[M - 1] / odnosi[N - 1])
```

Potrebno znanje: osnovne matematičke operacije

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
79	**5**	**15,3**

6.3. Zadatak: Kviz

Autor: Adrian Satja Kurdija

Učitane brojeve spremamo u niz i za svaki broj idemo „unatrag“ po nizu, pribrajajući mu prethodne brojeve i provjeravajući je li zbroj tih *K* brojeva prost. Prostost broja provjeravamo tako da provjerimo ima li on djelitelja većih od 1 i manjih od samoga sebe. Za implementacijske detalje pogledajte donji kod.

Programski kod (pisan u Python 3.x)

```
def prost(x):
    if x == 1: return False
    for i in range(2, x):
        if x % i == 0:
            return False
    return True
n, k = map(int, input().split())
a = []
for i in range(n):
    a.append(int(input()))
    for j in range(i, -1, -1):
        if not prost(sum(a[j : i + 1])):
            continue
        if k == 0 or len(a[j : i + 1]) == k:
            for x in a[j : i + 1]:
                print(x)
            exit(0)
```

Potrebno znanje: nizovi

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
79	**2**	**6,94**

7.1. Zadatak: Milja — Autor: Nikola Dmitrović

Vidi opis zadatka 6.2. Jard.

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
66	**14**	**23.39**

7.2. Zadatak: Slaven — Autor: Nikola Dmitrović

Tijek rješavanja ovog zadatka možemo podijeliti u tri dijela. Za 21 bod možemo samo pronaći najdulji niz nepobjedivosti jer neće postojati utakmica koja bi se mogla zanemariti. Njega možemo pronaći tako da ćemo se kretati po stringu i brojati znakove koji su različiti od „L“. U onom trenutku kada dođemo do „L“ brojač u koji smo spremali broj utakmica bez poraza usporedimo s do tada najvećom duljinom i resetiramo brojač na nula.

Funkcija za određivanje najduljeg niza nepobjedivosti (pisana u Python 3.x)

```
def nepobjedivost(s):
    najveci = 0                 #najveći niz nepobjedivosti u početku je nula
    d = 0                       #d je duljina trenutnog podniza nepobjedivosti
    for i in range(len(s)):     #krećemo se po nizu utakmica
        if s[i] != 'L':         #ako znak nije „L“
            d += 1              #povećaj duljinu podniza nepobjedivosti
        else:                   #ako je „L“
            if d > najveci:     #provjeri je duljina podniza veća od „najveći“
                najveci = d     #ako jeste, pronašli smo dulji niz nepobjedivosti
            d = 0               #resetiraj duljinu podniza
    najveci = max(najveci, d)#zbog zadnjeg podniza koji ne završava s „L“
    return najveci              #vrati duljinu najvećeg niza nepobjedivosti
```

Koristeći ovu funkciju možemo sada riješiti drugi dio zadatka za još 21 bod. Kako iz sekcije „Bodovanje“ znamo da će postojati samo jedna utakmica koja bi se mogla zanemariti, tada ćemo iz originalnog stringa obrisati znak „L“ koji se pojavljuje u jednom od četiri moguća podstringa „WLW“, „WLD“, „DLW“, „DLD“. U Pythonu za to možemo iskoristiti metodu *replace*, npr. `s = s.replace("WLW","WW").`

Za cjelovito rješenje zadatka moramo pronaći način kako iz originalnog stringa obrisati po jednu utakmicu koja zadovoljava navedeni uvjet, odrediti vrijednost funkcije na takvom stringu te to ponoviti za svaku takvu utakmicu. Jedan od načina je da se krećemo po stringu i kada dođemo do znaka „L“ provjerimo koji znak mu prethodi, a koji slijedi. U povoljnim situacijama, kreiramo novi string u kojem nema tog znaka „L“ i za njega odredimo vrijednost funkcije.

Programski kod (pisan u Python 3.x)

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

```
tekme = input()
niz = nepobjedivost(tekme)
for i in range(1, N - 1):
    if tekme[i]=='L' and tekme[i-1] in ['D','W'] and tekme[i + 1] in ['D','W']:
        s = tekme[:i] + tekme[i + 1:]
        if nepobjedivost(s) > niz:
            niz = nepobjedivost(s)
print(niz)
```

Potrebno znanje: string

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
66	**7**	**24.39**

7.3. Zadatak: Tinta — Autor: Adrian Satja Kurdija

Zadatak je najefikasnije riješiti rekurzijom (što su neki natjecatelji i učinili). Ovdje dajemo jednostavnije i nešto sporije rješenje: *for*-petljom prolazimo po prirodnim brojevima koji dolaze u obzir, tj. po višekratnicima broja *K* koji imaju traženi broj znamenaka, te za svaki od njih provjeravamo (znamenku po znamenku) odgovara li njegov dekadski zapis zadanom „umrljanom“ zapisu.

Programski kod (pisan u Python 3.x)

```
a = input()
n = len(a)
k = int(input())
x = k
mogucih = 0
while x < 10**n:
   b = str(x)
   x += k
   if len(b) != n:
       continue
   for i in range(n):
       if a[i] != '*' and a[i] != b[i]:
           b = None
           break
   if b is not None:
```

```
        mogucih += 1
print(mogucih)
```

Potrebno znanje: stringovi

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
66	**3**	**14**

8.1. Zadatak: Baterija

Autor: Nikola Dmitrović

Kada Lucija uopće ne bi koristila mobitel tijekom punjenja, on bi se u X-toj minuti napunio do X * 2 posto. Za svaku minutu prije X-te minute u kojoj Lucija koristi mobitel od maksimalne napunjenosti trebamo oduzeti jedan jer se u tim minutama baterija puni jedan, a ne dva posto. Uočimo da se vrijednost varijable X nalazi u posljednjem retku ulaznih podataka zbog čega je prvo potrebno minute u kojima se mobitel koristio učitati u niz.

Programski kod (pisan u Python 3.x)

```
M = int(input())
koristenje = []
for i in range(M):
    koristenje += [int(input())]

X = int(input())
baterija = X * 2

for i in koristenje:
    if i <= X:
        baterija -= 1
print(baterija)
```

Potrebno znanje: naredba odlučivanja, naredba ponavljanja, nizovi

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
38	**31**	**35**

8.2. Zadatak: Sudar

Autor: Adrian Satja Kurdija

Da bismo riješili zadatak, trebamo implementirati nekoliko stvari.

- Simuliranje kretanja robota. Ovdje su od velike pomoći pomoćni nizovi ili rječnici *dx, dy* iz donjeg rješenja koji za pojedini smjer daju odgovarajuće pomake po *x* i *y* koordinati pa robota možemo pomicati bez *if*-ova. Simuliranje je malo zakomplicirano činjenicom da najprije moramo 5 sekundi pomicati prvog robota, potom neko vrijeme oba robota istodobno, i potom eventualno neko vrijeme robota A *ili* robota B. Za implementacijske detalje pogledajte donji kod.
- Uočavanje sudara. Primijetimo da se sudar događa kada robot A prelazi na prethodnu poziciju robota B, a robot B prelazi na prethodnu poziciju robota A. Ako pozicije *(x, y)* svakog robota

spremamo u njegov niz, lako je provjeriti je li se dogodio sudar jer su posljednje pozicije robota spremljene na krajevima odgovarajućih nizova.

- Brojenje posjećenih točaka. Ovo možemo učiniti npr. sortiranjem nizova spomenutih u prethodnoj točki i potom eliminiranjem „duplikata“ koji se u sortiranom nizu pojavljuju uzastopce. Alternativno možemo koristiti strukturu *set* (ona automatski izbacuje duplikate) ili bilježiti posjećena polja u nekoj matrici (u tom slučaju valja smisliti kako riješiti problem negativnih koordinata).

Programski kod (pisan u Python 3.x)

```
dx = {'S': 0, 'J': 0, 'I': 1, 'Z': -1}
dy = {'S': 1, 'J': -1, 'I': 0, 'Z': 0}

a = input()
b = input()
ax, ay = 0, 0
bx, by = 0, 0

A = [(0, 0)]
B = [(0, 0)]

for i in range(5):
    ax += dx[a[i]]
    ay += dy[a[i]]
    A.append((ax, ay))

a = a[5:]
sudari = 0
m = min(len(a), len(b))
for i in range(m):
    ax += dx[a[i]]
    ay += dy[a[i]]
    bx += dx[b[i]]
    by += dy[b[i]]
    if (ax, ay) == B[-1] and (bx, by) == A[-1]:
        sudari += 1
    A.append((ax, ay))
    B.append((bx, by))
```

```
a = a[m:]
b = b[m:]
for i in range(len(a)):
   ax += dx[a[i]]
   ay += dy[a[i]]
   A.append((ax, ay))
for i in range(len(b)):
   bx += dx[b[i]]
   by += dy[b[i]]
   B.append((bx, by))


print(sudari)
A = set(A)
B = set(B)
print(len(A))
print(len(B))
print(len(A | B))
```

Potrebno znanje: stringovi, sortiranje ili *set* ili matrice

Kategorija: simulacija

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
38	**10**	**31,36**

8.3. Zadatak: Kripto — Autor: Adrian Satja Kurdija

Zadatak je moguće riješiti koristeći razne pristupe. Budući da stringove treba sortirati koristeći tri kriterija, svakoj riječi možemo pridružiti *trojku* prirodnih brojeva koja sadrži vrijednosti kriterija za dotičnu riječ: *(učestalost, duljina, indeks prvog pojavljivanja)*, i onda sortirati sve te trojke jer je njih lako uspoređivati. U autorovom rješenju korišten je *dictionary* koji svakoj riječi pridružuje odgovarajuću trojku, a Python „zna“ sortirati ove trojke ulazno ili silazno. Budući da trojke sortiramo silazno (prednost ima veća prva komponenta, pa veća druga komponenta, pa veća treća komponenta), indeks prvog pojavljivanja ubačen je u trojku s negativnim predznakom da bi onaj manji ispao veći pa imao prednost. Napominjemo da je ovo samo jedno od mogućih rješenja i potičemo učenike da isprobaju i drugačije pristupe.

Programski kod (pisan u Python 3.x)

```
n = int(input())
rijeci = dict()
```

```
for i in range(n):
    a = input()
    if a in rijeci:
        rijeci[a] = (rijeci[a][0] + 1, len(a), -i)
    else:
        rijeci[a] = (1, len(a), -i)
b = sorted(rijeci.values(), reverse=True)
for val in b:
    for r in rijeci:
        if rijeci[r] == val:
            for i in range(val[0]):
                print(r)
```

Potrebno znanje: stringovi, sortiranje

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
38	**9**	**35**