19. siječnja 2016.

Županijsko natjecanje / 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: Brojimo

Autor: Nikola Dmitrović[1]

Prvo trebamo odrediti korak X iz teksta zadatka. Na prvi pogled, X se može odrediti kao razlika brojeva C i B ili kao razlika brojeva B i A. Prvi način nije dobar jer postoji slučaj kada je A = 89, B = 97 i C = 1. Drugi način nije dobar jer postoji slučaj kad je A = 97, B = 1 i C = 13. Za točno određivanje koraka treba kombinirati ove dvije razlike.

Kada smo odredili korak, potrebno je odrediti sljedeći broj u slijedu brojanja. Tu treba paziti na situaciju kada je Perica krenuo brojati ispočetka.

Samo je jedan natjecatelj u potpunosti riješio ovaj zadatak. Čestitamo Patriku Modriću iz Osnovne škole Ivana Mažuranića Zagreb.

Programski kod (pisan u Python 3.x)

```
A = int(input())
B = int(input())
C = int(input())
if C > B:
    korak = C - B
else:
    korak = B - A
if C + korak > 100:
    print(1)
else:
    print(C + pomak)
```

Potrebno znanje: naredba odlučivanja

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
113	**1**	**20.84**

5.2. Zadatak: Leo

Autor: Nikola Dmitrović

Zadatak ima tri dijela. Moguće je napisati program koji odjednom rješava sva tri dijela, ali ovdje preporučujemo pisanje programa koji će zasebno riješiti svaki dio zadatka.

Koliko katova su Leo i Kiki zajedno putovali u liftu?

[1] ukupan broj osvojih bodova svih natjecatelja podijeljen s ukupnim brojem bodova koji su se mogli osvojiti

Kako lift, od trenutnog kata na kome se nalazi, najprije vozi do kata do kojeg treba proći manje katova, jasno je da će taj broj prijeđenih katova biti traženi odgovor na pitanje. Naime, do tog bližeg kata će i Leo i Kiki biti u liftu, a onda će jedan od njih izaći.

```
X = int(input())
L = int(input())
K = int(input())
if abs(X - L) <= abs(X - K):
    zajedno = abs(X - L)
else:
    zajedno = abs(X - K)
print(zajedno)
```

Kolika je ukupna udaljenost (izražena u katovima) koju je lift prešao tijekom vožnje u kojoj je i Lea i Kikija odvezao do njihovih željenih katova?

Nakon što je lift došao do bližeg kata (odgovor na 1. pitanje), krenuo je prema onom koji je u startu bio dalje. Udaljenost između katova L i K nije teško izračunati.

```
ukupno = abs(L - K) + zajedno
print(ukupno)
```

*Je li lift, od **X**-tog kata, najprije krenuo prema gore („G“) ili prema dolje („D“)?*

Ovo pitanje je bilo najteže. Ovisno o položaju katova L i K u odnosu na kat X, trebalo je procjeniti je li lift krenuo prema gore ili dolje.

```
if abs(X - L) <= abs(X - K): # L je bliži ili jednako blizu X-u od K
    if L > X:                # L je više od X
        print("G")
    else:                    # L je niže od X
        print("D")
else:                        # K je bliži
    if K > X:                # R je više od X
        print("G")
    else:                    # L je niže od X
        print("D")
```

Potrebno znanje: naredba odlučivanja

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
113	**2**	**21,23**

5.3. Zadatak: Lazanje

Autor: Nikola Dmitrović

U zadatku se pitamo: Koliko je bodova na kraju glasanja imao pobjednik, tj. onaj kuhar za koga je glasalo najviše gostiju?

Za odgovoriti na ovo pitanje dovoljno je učitati sve ulazne podatke i prebrojiti koliko se puta na ulazu pojavio znak „T“, a koliko znak „M“. Odgovor koji tražimo je veći od ta dva broja.

U kojem je trenutku glasanja postalo jasno tko je pobjednik? Tražimo oznaku gosta nakon čijeg je glasanja pobjednik imao dovoljno glasova da ga poraženi više ne može prestići.

Kako su bira između samo dva kuhara, dovoljno je uočiti da će pobjednik postati onaj koji prvi osvoji N // 2 + 1 glasova. U trenutku kada se to dogodi treba zapamtiti oznaku glasača kod kojeg se to dogodilo i zaustaviti provjeravanje tog uvjeta.

Programski kod (pisan u Python 3.x)

```
N = int(input())
tomaz = mate = 0
stop = 0 # dok je stop nula, niti jedan kuhar nije došao do N // 2 + 1 glasova
for i in range(N):
    glas = input()
    if glas == 'T':
        tomaz += 1
    else:
        mate += 1
    if tomaz == (N // 2 + 1) or mate == (N // 2 + 1):
        if stop == 0:
            stop = i + 1
if tomaz > mate:
    print(tomaz)
else:
    print(mate)
print(stop)
```

Potrebno znanje: naredba ponavljanja, naredba odlučivanja

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
113	**2**	**16,08**

6.1. Zadatak: Stella — Autor: Nikola Dmitrović

U sekciji *Ulazni podaci* definira se da je prirodan broj **N** ($10 \leq \mathbf{N} \leq 999999$, broj znamenki u broju N je paran). Iz ovog slijedi da je dovoljno kod podijeliti u tri dijela ovisno o tome kojem intervalu N pripada. Zatim treba odrediti lijevi i desni broj te odrediti traženu razliku iz teksta zadatka.

Programski kod (pisan u Python 3.x)

```
N = int(input())
if N <= 99:
    l = N // 10
    d = N % 10
    if l <= d: # može s naredbom odlučivanja
        razlika = d - l
    else:
        razlika = l - d
    # razlika = abs(l - d) # a može i pomoću apsolutne vrijednosti
if N >= 1000 and N <= 9999:
    l = N // 100
    d = N % 100
    razlika = abs(l - d)
if N >= 100000 and N <= 999999:
    l = N // 1000
    d = N % 1000
    razlika = abs(l - d)
print(razlika)
```

Potrebno znanje: naredba odlučivanja, algoritam određivanja znamenki broja

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
82	**27**	**16,95**

6.2. Zadatak: Cjenkanje — Autor: Adrian Satja Kurdija

Iz zadatka se razaznaje da nam treba while petlja koja se prekida nakon što razlika prodavačeve i kupčeve cijene postane manja od 10. U svakom koraku petlje promijenit ćemo prodavačevu ili kupčevu cijenu, ovisno o tome tko je na redu, što možemo pratiti u pomoćnoj varijabli tko_je_na_redu. Alternativno (kao u priloženom kodu #1) možemo u istom koraku petlje promijeniti i prodavačevu i

kupčevu cijenu, dozvoljavajući da nakon svake promjene prekinemo petlju ako je razlika cijena manja od 10.

Programski kod (pisan u Python 3.x)

```
# 1. RJEŠENJE
p = int(input())
k = int(input())
while True:
    # Prodavac je na redu.
    if p - k <= 10:
        print(k)
        break
    p -= 10
    # Kupac je na redu.
    if p - k <= 10:
        print(p)
        break
    k += 10
```

Programski kod (pisan u Python 3.x)

```
# 2. RJEŠENJE
P = int(input())
K = int(input())
potez = 1
while P - K > 10:
    if potez == 1:
        P = P - 10
    else:
        K = K + 10
    potez = potez * -1
if potez == 1:
    print(K)
else:
    print(P)
```

Evo jednog zanimljivog matematičkog rješenja.

Programski kod (pisan u Python 3.x)

```
# 3. RJEŠENJE
K = int(input())
```

```
P = int(input())


raz = P - K
if raz // 10 % 2:
    print(P - (raz // 10 // 2 + 1) * 10)
else:
    print(K + raz // 10 // 2 * 10)
```

Potrebno znanje: while-petlja

Kategorija: simulacija

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
82	**15**	**27.14**

6.3. Zadatak: Ispit — Autor: Dario Babojelić

Najlakše je unaprijed odrediti sve moguće brojeve bodova na ispitu. U tu svrhu stvorit ćemo polje veličine T * P. U tom polju ćemo na svaku lokaciju od 0 do T * P zapisati 0 ako taj broj bodova nije moguće ostvariti, tj. zapisati 1 ako jest. Najprije stvorimo polje gdje je na svakoj lokaciji vrijednost 0. Sada možemo pomoću dvije ugnježđene petlje isprobati sve kombinacije točnih, netočnih i neodgovorenih pitanja. Za svaku kombinaciju izračunamo broj bodova koji tom kombinacijom ostvarujemo i lokaciju u polju jednaku tom broju bodova označimo s jedinicom. Nakon što smo stvorili takvo polje lako odrađujemo ostatak zadatka. Za svaki prijateljev broj bodova x radimo sljedeće: ako je polje[x] = 1 uzmi x u obzir za maksimum, ako je polje[x] = 0 povećaj brojač neispravnih bodova za jedan.

U programskom kodu pokazat ćemo još jedan način rješavanja ovog zadatka. Ideja je pokazati mogućnosti Pythona.

Programski kod (pisan u Python 3.x)

```
P = int(input())
T = int(input())
N = int(input())
M = int(input())
L = []
for i in range(P + 1):
    for j in range(P + 1):
        for k in range(P + 1):
            if i + j + k == P:
                bodova = i * T - j * N
```

```
                L = L + [bodova]


L1 = list(map(int, input().split()))
L2 = []
koliko = 0
for i in L1:
    if i in L:
        L2 = L2 + [i]
    else:
        koliko += 1
print(koliko)
print(max(L2))
```

Potrebno znanje: nizovi, algoritam traženja maksimalnog elementa

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
82	**4**	**8,7**

7.1. Zadatak: Boks — Autor: Nikola Dmitrović

Za svakog sudca treba odrediti je glasao za boksača A ili boksača B. Zatim treba provjeriti je li neki od boksača dobio više ili jednako od dva glasa. Ako nitko nije dobio većinu glasova, treba vidjeti za koga je glasao glavni sudac. Važno je uočiti da bez obzira na to kako je glasao glavni sudac, njegov glas u takvoj situaciji određuje konačan ishod meča.

Programski kod (pisan u Python 3.x)

```
Z1 = input()
Z2 = input()
Z3 = input()
G = int(input())
L = [Z1, Z2, Z3]
A = B = 0
if Z1 == "A": A += 1
if Z1 == "B": B += 1
if Z2 == "A": A += 1
if Z2 == "B": B += 1
if Z3 == "A": A += 1
if Z3 == "B": B += 1
if A >= 2:
    print("A")
elif B >= 2:
    print("B")
else:
    print(L[G - 1])
```

Potrebno znanje: naredba odlučivanja

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
50	**16**	**30,08**

7.2. Zadatak: Usmeni — Autor: Adrian Satja Kurdija

Zapišemo li u niz zadane brojeve koje su napisali učenici od 1 do **N**, lako je simulirati postupak učiteljičinog ispitivanja: ako se trenutno nalazimo na učeniku **A**, prelazak na sljedećeg učenika vršimo promjenom varijable A na vrijednost **zapisao[A]**, tj. na broj učenika koji je zapisao učenik **A**.

U ovom zadatku, međutim, kao da postupak trebamo "obrnuti", tj. znajući posljednjeg ispitanog učenika trebamo doći do prvoga. Ići unatrag nije tako jednostavno jer često postoji više mogućnosti, npr. ako su broj 5 zapisali mnogi učenici, tada su svi oni mogli biti ispitani neposredno prije učenika 5, a i svaki od tih učenika može imati više mogućih "prethodnika", i tako dalje.

Stoga, umjesto rekonstrukcije unatrag, zadatak rješavamo "unaprijed", isprobavanjem svih mogućnosti: za svakog učenika provjerit ćemo gdje će učiteljica završiti ako krene od tog učenika i postavi **K** pitanja. Završi li učiteljica upravo na zadanom učeniku **M**, ispisujemo učenika od koga smo krenuli. Tu provjeru vršimo za sve moguće "početne" učenike od 1 do **N**.

Programski kod (pisan u Python 3.x)

```
n, k, m = map(int, input().split())
zapisao = [0]
for i in range(1, n + 1):
    zapisao.append(int(input()))
for a in range(1, n + 1):
    ispitanik = a
    for i in range(k - 1):
        ispitanik = zapisao[ispitanik]
    if ispitanik == m:
        print(a)
```

Potrebno znanje: nizovi

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
50	**3**	**10,5**

7.3. Zadatak: Cijeli — Autor: Adrian Satja Kurdija

I ovaj zadatak rješavamo isprobavanjem svih mogućnosti: za sve moguće početne pozicije **(i, j)** Markovog odabira i sve moguće početne pozicije **(r, s)** Darkovog odabira, provjeravamo koliki bi bio zbroj odabranih polja i uspoređujemo ga s dosad najvećim pronađenim zbrojem (varijablom **maks**). Na samom početku **maks** valja postaviti npr. na -50 * 2**K**, jer ako su svi brojevi u tablici negativni, maksimum može biti i negativan.

Unutar ugnježđene for-petlje koja bira sve moguće **i, j, r, s** valja zbrojiti **K** polja u **i**-tom retku počevši od polja (**i, j**) te **K** polja u **s**-tom stupcu počevši od polja (**r, s**). Od toga zbroja valja oduzeti zajedničko polje ako se odabiri "sijeku", jer bismo ga inače pribrojili dvaput. Za detalje pogledajte priloženi kod.

Programski kod (pisan u Python 3.x)

```
n, k = map(int, input().split())
```

```
a = []
for i in range(n):
    a.append([])
    redak = input().split()
    for broj in redak:
        a[-1].append(int(broj))
maks = -50 * 2 * k
for i in range(n):
    for j in range(n - k + 1):
        for r in range(n - k + 1):
            for s in range(n):
                zbroj = 0
                for x in range(k):
                    zbroj += a[i][j + x]
                    zbroj += a[r + x][s]
                if r <= i < r + k and j <= s < j + k:
                    zbroj -= a[i][s]
                maks = max(maks, zbroj)
print(maks)
```

Potrebno znanje: matrice

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
50	**2**	**8,28**

8.1. Zadatak: Care — Autor: Nikola Dmitrović

Prvo ćemo izjavu „X minuta do Y sati“ pretvoriti u minute i time odrediti koliko je minuta proteklo od početka dana. Zatim ćemo taj ukupan broj minuta pretvoriti u sate i minute. I za kraj, najveći problem je kako dobiveno rješenje prezentirati u obliku koji je definiran u tekstu zadatka.

Programski kod (pisan u Python 3.x)

```
X = int(input())
Y = int(input())
ukupno = Y * 60 - X
sati = ukupno // 60
minuta = ukupno % 60
if sati < 10 and minuta < 10:
    print('0', sati, ':0', minuta,sep = '')
elif minuta < 10:
    print(sati, ':0', minuta,sep = '')
elif sati < 10:
    print('0', sati, ':', minuta,sep = '')
else:
    print(sati, minuta, sep = ':')
```

Potrebno znanje: naredba odlučivanja, rad sa satima i minutama

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
51	**15**	**27,21**

8.2. Zadatak: Carina — Autor: Adrian Satja Kurdija

U zadatku je dovoljno za svaku osobu, redom od prve do posljednje u redu, pratiti kada ulazi i izlazi iz koje kućice. Osoba ulazi u kućicu A u trenutku kad je prethodna osoba iz nje izašla. Osoba ulazi u kućicu B u trenutku kad je prethodna osoba iz nje izašla, osim ako se trenutna osoba nije dulje zadržala u kućici A. Osoba ulazi u kućicu C u trenutku kad je prethodna osoba iz nje izašla, osim ako se trenutna osoba nije dulje zadržala u kućici B. Donje rješenje koristi varijable *a_izlaz, b_izlaz, c_izlaz* za trenutke izlazaka iz kućica A, B, C te ih ažurira za svaku novu osobu.

Programski kod (pisan u Python 3.x)

```
n = int(input())
a_izlaz = 0
b_izlaz = 0
```

```
c_izlaz = 0
for i in range(n):
   a, b, c = map(int, input().split())
   a_izlaz += a
   b_izlaz = max(a_izlaz, b_izlaz) + b
   c_izlaz = max(b_izlaz, c_izlaz) + c
print(c_izlaz)
```

Potrebno znanje: for petlja

Kategorija: simulacija

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
51	**4**	**10,84**

8.3. Zadatak: Tipkovnica — Autor: Adrian Satja Kurdija

Za svaku riječ s danog popisa provjerit ćemo je li ona "dobra", tj. je li "susjedna" zadanoj riječi (koju je Zrinka napisala). Ako jest, spremit ćemo je u niz mogućih rješenja koji ćemo na kraju ispisati sortiran abecedno.

Kako provjeravamo je li riječ "dobra"? Najprije, ako riječ nema jednak broj slova kao zadana riječ, odgovor je negativan. Inače valja proći for-petljom po duljini riječi, provjeravajući jesu li **i**-to slovo zadane riječi te **i**-to slovo trenutne riječi susjedni na tipkovnici. Ako su slova svaki put susjedna, riječ je "dobra".

Najzanimljiviji dio zadatka jest provjera jesu li dva slova susjedna na tipkovnici. Moguća su razna rješenja. Natjecatelj Jakov Manjkas koristio je *dictionary* u kojem je za svako slovo naveo njegove susjede:

```
D = {
  "Q": "WA",
  "W": "QASE",
  "E": "WSDR",
  "R": "EDFT",
  "T": "RFGY",
  "Y": "TGHU",
  "U": "YHJI",
  "I": "UJKO",
  "O": "IKLP",
  "P": "OL",
```

```
  "A": "QWSZ",
  "S": "WEDXZA",
  "D": "ERFCXS",
  "F": "RTGVCD",
  "G": "TYHBVF",
  "H": "YUJNBG",
  "J": "UIKMNH",
  "K": "IOLMJ",
  "L": "KOP",
  "Z": "ASX",
  "X": "ZSDC",
  "C": "XDFV",
  "V": "CFGB",
  "B": "VGHN",
  "N": "BHJM",
  "M": "NJK"
  }
```

i koristeći taj rječnik lako je provjeravao susjedstva. Slično je rješavao i natjecatelj Pavel Kliska, ali je u rječniku zaboravio jedno slovo i tako izgubio dva test podatka.

Jednostavnije je stoga, kao u službenom rješenju niže, promatrati tipkovnicu kao matricu kojoj samo navedemo retke, a stupci su onda definirani implicitno: prvi stupac čine slova Q, A, Z, drugi stupac W, S, X i tako dalje. Susjedstvo tada provjeravamo gledajući najprije stupce:

- Ako su slova u istom stupcu, susjedna su ako im se redci razlikuju za 1 (inače nisu).
- Ako se stupci razlikuju za 2 ili više, slova nisu susjedna.
- Ako se stupci razlikuju za 1, susjedna su:
  - ako su u istome retku, ili
  - ako su u susjednim redcima, pri čemu valja eliminirati slučaj kad su redak i stupac jednog slova manji od retka i stupca drugog slova (npr. R i G nisu susjedi).

Programski kod (pisan u Python 3.x)

```
redci = ["QWERTYUIOP", "ASDFGHJKL", "ZXCVBNM"]
def pozicija(slovo):
   for r in range(3):
       for i in range(len(redci[r])):
           if redci[r][i] == slovo:
               return r, i
```

```
a = input()
n = int(input())
rj = []
for i in range(n):
    b = input()
    if len(a) != len(b):
        continue
    blizu = True
    for i in range(len(a)):
        r1, s1 = pozicija(a[i])
        r2, s2 = pozicija(b[i])
        if s1 == s2:
            if abs(r1 - r2) == 2:
                blizu = False
        if abs(s1 - s2) == 1:
            if r1 == r2:
                continue
            if abs(r1 - r2) == 2:
                blizu = False
            elif r1 < r2 and s1 < s2 or r2 < r1 and s2 < s1:
                blizu = False
        if abs(s1 - s2) > 1:
            blizu = False
    if blizu:
        rj.append(b)
for s in sorted(rj):
    print(s)
```

Potrebno znanje: tablica, stringovi

Kategorija: ad hoc

Broj natjecatelja koji su rješavali zadatak	Broj natjecatelja koji su točno riješili zadatak	Prosječna riješenost zadatka
51	**2**	**17,45**