11. ožujka 2022.

2022 Natjecanje iz informatike

Županijska razina 2022/ 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

5.1. Zadatak: Semafor — Ideja: Nikola Dmitrović

Postoje samo četiri testna primjera koja treba provjeriti.

Programski kod (pisan u Python 3)

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

if N == 1 and K == 2:
    print(2)
if N == 2 and K == 1:
    print(3)
if N == 2 and K == 3:
    print(1)
if N == 3 and K == 2:
    print(2)
```

Potrebno znanje: naredba odlučivanja

Kategorija: ad hoc

5.2. Zadatak: Wordle — Ideja: Nikola Dmitrović

Prvo odredimo znamenke broja K. Za svaki od N brojeva koje Vito izgovori odredimo njegove znamenke. Provjerom odnosa tih znamenki kreiramo traženi ispis.

Programski kod (pisan u Python 3)

```
K = int(input()) # Teo broj
N = int(input()) # broj pokušaja

Kj = K % 10; K = K // 10
Kd = K % 10; K = K // 10
Ks = K % 10; K = K // 10
Kt = K

for i in range(N):
    X = int(input()) # Vitin pokušaj

    Xj = X % 10; X = X // 10
    Xd = X % 10; X = X // 10
```

```
    Xs = X % 10; X = X // 10
    Xt = X
    # provjera odnosa tisućica
    if Xt == Kt:
        print('G', end = '')
    elif Xt == Kd or Xt == Ks or Xt == Kj:
        print('Y', end = '')
    else:
        print('R', end = '')
    # provjera odnosa stotica
    if Xs == Ks:
        print('G', end = '')
    elif Xs == Kt or Xs == Kd or Xs == Kj:
        print('Y', end = '')
    else:
        print('R', end = '')
# provjera odnosa desetica
    if Xd == Kd:
        print('G', end = '')
    elif Xd == Kt or Xd == Ks or Xd == Kj:
        print('Y', end = '')
    else:
        print('R', end = '')
# provjera odnosa jedinica
    if Xj == Kj:
        print('G', end = '')
    elif Xj == Kt or Xj == Ks or Xj == Kd:
        print('Y', end = '')
    else:
        print('R', end = '')
```

Potrebno znanje: naredba ponavljanja, naredba odlučivanja

Kategorija: ad hoc

5.3. Zadatak: Slavlje

Ideja: Nikola Dmitrović

Zadatak se sastoji od četiri dijela. Dosjetka koju napravimo je da odredimo koji je današnji datum dan od početka godine (*datum = (B - 1) * 7 + A*). Dalje se samo implementiraju uvjeti iz zadatka pazeći na rubne slučajeve.

Programski kod (pisan u Python 3)

```
A = int(input())
B = int(input())
X = int(input())

# prvo pitanje: koji je danas dan?
datum = (B - 1) * 7 + A
D = datum % 28
M = datum // 28 + 1
if D == 0:
    D = 28
    M -= 1
print(D, M)

# drugo pitanje: datum kada je rođendan

datum += X
D1 = datum % 28
M1 = datum // 28 + 1
if D1 == 0:
    D1 = 28
    M1 -= 1
if M1 > 12: M1 -= 12
print(D1, M1)

# treće pitanje: koji je to dan u tjednu

dan_tjedan = D1 % 7 - 1
if dan_tjedan == -1: dan_tjedan = 6
if dan_tjedan == 0: print('PONEDJELJAK')
if dan_tjedan == 1: print('UTORAK')
if dan_tjedan == 2: print('SRIJEDA')
```

```
if dan_tjedan == 3: print('CETVRTAK')
if dan_tjedan == 4: print('PETAK')
if dan_tjedan == 5: print('SUBOTA')
if dan_tjedan == 6: print('NEDJELJA')

# četvrto pitanje: rođendanski tjedan

D2 = D1 - dan_tjedan
D3 = D1 + (6 - dan_tjedan)
print(D2, M1, D3, M1)
```

Potrebno znanje: naredba odlučivanja

Kategorija: ad hoc

6.1. Zadatak: Wordle Ideja:

Vidi 5.2.

6.2. Zadatak: Veselje Ideja:

Vidi 5.3

6.3. Zadatak: Vampir Ideja: Nikola Dmitrović

Za svaki troznamenkasti broj *i* provjerimo je djelitelj broja N te ako je, odredimo par brojeva (i, j) koji pomnoženi daje N. Broj N i umnožak i*j pretvorimo u niz znakova te svaki taj niz sortiramo. Ako su sortirane vrijednosti jednake, pronašli smo traženi par za koji još trebamo provjeriti ima li najveći zbroj znamenki. Ako ne pronađemo traženi par brojeva, onda trebamo ispisati zbroj znamenki broja N.

Programski kod (pisan u Python 3)

```
N = int(input())
A = B = 0
zbroj = 999+999

for i in range(100, 1000):
    j = N / i
    if 100 <= j <= 999 and j - int(j) == 0:
        s1 = str(N)
        s2 = str(i) + str(int(j))

        if ''.join(sorted(s1)) == ''.join(sorted(s2)):
```

```
            if i + int(j) < zbroj:
                A = i
                B = int(j)
                zbroj = i + j

if A == 0 and B == 0:
    zbroj = 0
    for i in str(N):
        zbroj += int(i)
    print(zbroj)
else:
    print(A, B)
```

Potrebno znanje: string

Kategorija: ad hoc

7.1. Zadatak: Umnožak — Ideja: Stjepan Požgaj

Ovo je primjer zadatka koji se može riješiti na više različitih načina. Jedan pristup bi bio da za svaki od tri broja probamo promijeniti neku od njegovih znamenki, nakon čega je potrebno provjeriti dobiva li se nakon promjene ispravni izraz.

U rješenju koje je po mišljenju autora jednostavnije imamo tri slučaja. U prvom slučaju pretpostavimo da su brojevi A i B ispravni, tj. da je potrebno promijeniti točno jednu znamenku broja C. Da bismo saznali koja je to znamenka moramo provjeriti razlikuje se umnožak brojeva A i B od C u točno jednoj znamenki. Slično radimo i u slučajevima kad je promijenjena znamenka nekog od prva dva broja.

Potrebno znanje: naredba grananja, rad sa znamenkama

Kategorija: ad hoc

7.2. Zadatak: Rotacije — Ideja: Gabrijel Jambrošić

Kada se na početnoj tablici nalazi točno jedan žeton postoje dva slučaja koja treba razlikovati:

1. Žeton je u sredini tablice (moguće jedino ako je N neparan)

2. Žeton nije u sredini tablice

U prvom slučaju tablica već zadovoljava uvjete iz zadatka jer izgleda isto kako god ju okrenuli.

U drugom slučaju treba dodati još tri žetona na pozicije koje odgovaraju poziciji početnog žetona za tri preostale rotacije.

Ovo rješenje ostvaruje 20 bodova.

Za sve bodove možemo poopćiti prošlo rješenje: Za svaki žeton na koji naiđemo promotrimo odgovarajuće pozicije tog žetona za preostale tri rotacije. Na svakoj od tih pozicija treba se nalaziti žeton pa ga dodajemo ako on već ne postoji. Moramo paziti da iste pozicije ne brojimo više puta, a to možemo ostvariti tako da promatramo samo jedan kvadrant tablice ili bilježimo pozicije na koje smo već postavili žeton.

Vremenska složenost algoritma: O(N^2)

Potrebno znanje: dvodimenzionalne liste

Kategorija: ad hoc

7.3. Zadatak: Ogrlice — Ideja: Stjepan Požgaj

Potrebno je na pametan način odrediti koje su ogrlice međusobno jednake. Vidimo da su dvije ogrlice jednake ako su jednake nakon što jednu zarotiramo u smjeru kazaljke na satu za neki broj. Isto tako, prije tog rotiranja možemo ogrlicu polegnuti na suprotnu stranu, što je ekvivalentno tome da slova iz riječi iz ulaza koja predstavlja ogrlicu zapišemo u obrnutom poretku.

Cilj nam je za svaku riječ iz ulaza odrediti sve njezine moguće reprezentacije i onda uzeti neku jedinstvenu koja će ju predstavljati. Sve riječi moguće je generirati na prije opisani način: iz riječi izvadimo sve moguće rotacije originalne i okrenute riječi. Tako za svaku od N riječi dobivamo listu od 2*K riječi. Za uzeti jednog od tih 2*K predstavnika najbolje je uzeti leksikografski najmanju riječ, tj. prvu po abecedi. Kad za svaku ogrlicu imamo predstavnika, samo treba odrediti koliko je među tim predstavnicima različitih riječi. To se ponovno može napraviti pomoću sortiranja ili pak možemo koristiti neku strukturu poput rječnika ili skupa.

Potrebno znanje: for petlja, sortiranje

Kategorija: ad hoc

8.1. Zadatak: Autobus — Ideja: Patrick Pavić

Označimo početan broj putnika u autobusu s x. Primijetimo da nakon izlaska određenog broja putnika na prvoj stanici u autobusu se nalazi x – B1 putnika, pa nakon ulaska x – B1 + A1 putnika. Analogno nakon druge stranice x – B1 + A1 – B2 nakon izlaska te x – B1 + A1 – B2 nakon ulaska itd. Neka je mi najmanji od tih 2N brojeva koji će se pojaviti uz x, a mx najveći. Zbog uvjeta u kojem autobus nikada ne smije biti prazan vrijedi x + mi >= 0 tj. x >= -mi. Zato što je m najmanji takav broj, uvijek će vrijediti da je broj putnika u autobusu nenegativan. Analogno neka je mx najveći broj koji se pojavljivao uz x. Tada zbog ograničenog broja putnika mora vrijediti x + mx <= L tj. X <= L – mx. Sada znamo da za početni broj putnika u autobusu vrijedi -mi <= x <= L – mx. Kako je uvjet zadatka da uvijek postoji barem jedno rješenje, uvijek će vrijediti -mi <= L – mx. Sada kako bismo ispisali konačan broj putnika, za najmanji mogući potrebno je ispisati -mi – B1 + A1 – B2 + A2 – B3 + A3..., a za najveći L – mx – B1 + A1 – B2 + A2 – B3 + A3 ...

Potrebno znanje: naredba ponavljanja

Kategorija: ad hoc

8.2. Zadatak: Plusevi — Ideja: Patrick Pavić

Zadatak se može riješiti simuliranjem stavljanja pluseva. Potrebno je primijetiti da promatramo li polje u najvišem retku te najljevijem stupcu, postoji točno jedan način na koji možemo postaviti plus na njega tj. na taj način da gornji kraj plusa prekrije to polje. Kada bismo mogli na drugačiji način prekriti to polje, ono ne bi bilo u najvišem retku te najlijevijem stupcu. Sada možemo običnim prolaskom po matrici idući po retcima od gore prema dolje te zatim po stupcima s lijeva na desno probati popuniti tablicu. Ukoliko naiđemo na polje koje ne možemo popuniti, iz nužnosti postupka slijedi da rješenje ne postoji. Primijetite da također ako rješenje postoji, ono je uvijek jedinstveno.

Potrebno znanje: naredba grananja, for petlja, polja

Kategorija: ad hoc

8.3. Zadatak: Smijemo! — Ideja: Patrick Pavić

Zadatak je također moguće promatrati kao vrstu problema nazvanu Steinerova stabla. Više o tome na sljedećoj poveznici https://en.wikipedia.org/wiki/Rectilinear_Steiner_tree. Primijetite da kako god popunili matricu, uvijek će postojat najviše tri dodatna polja, ne nalaze se u početnim fontanama, koja sadrže više od dva susjedna polja. U suprotnom postoji bolje rješenje. Stoga moguće je probati na svaki mogući način postaviti te tri fontane, te potom pospajati ih na svaki od 3 moguća redoslijeda i zatim svaku od već postojećih fontana spojiti na neku od te tri dodatne fontane. Također potrebno je promatrati slučaj kada dodajemo dvije dodatne fontane, jednu dodatnu fontanu te nijednu dodatnu fontanu, no razdvajanje tih slučajeva moguće je izbjeći pažljivom implementacijom. Slijedi da je konačna složenost O(R ^ 4 S ^ 4).

Potrebno znanje: naredba grananja, for petlja, polja

Kategorija: ad hoc