3. ožujka 2021.

2021 Natjecanje iz informatike

Županijska razina 2021/ 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: Hvar — Autor: Nikola Dmitrović

Kako periodi mogu biti različiti, nužno ih je, prije uspoređivanja, svesti na isti period. Očito je da će to biti dani. Jednostavnim uspoređivanje odredit ćemo tražene vrijednosti iz teksta zadatka.

Programski kod (pisan u Python 3)

```
X = int(input())
HP = int(input())
Y = int(input())
VP = int(input())
period = [1, 7, 28, 336]
hvar = X * period[HP - 1]
vis = Y * period[VP - 1]
if hvar > vis:
    print('HVAR')
    print(hvar)
else:
    print('VIS')
    print(vis)
```

Potrebno znanje: naredba odlučivanja

Kategorija: ad hoc

5.2. Zadatak: Klinac — Autor: Nikola Dmitrović

Prvo ćemo za svaki od brojeva G i K odrediti nove brojeve koji imaju samo prvu, samo prve dvije, te samo prve tri znamenke. Uspoređivanjem tih brojeva otkrit ćemo koliko je poklapanje brojeva G i K te odrediti koliko je bodova dobio na kojem pitanju. Ove korake treba ponoviti N puta.

Programski kod (pisan u Python 3)

```
N = int(input())
ukupno = 0
for i in range(N):
    G = int(input())
    K = int(input())
    G1 = G // 1000
    G12 = G // 100
    G123 = G // 10
    K1 = K // 1000
    K12 = K // 100
    K123 = K // 10
```

```
    if G == K:
        ukupno += 4
    elif G123 == K123:
        ukupno += 3
    elif G12 == K12:
        ukupno += 2
    elif G1 == K1:
        ukupno += 1
print(ukupno)
```

Potrebno znanje: naredba ponavljanja

Kategorija: rad sa znamenkama

5.3. Zadatak: Nepa — Autor: Nikola Dmitrović

Zadatak pruža velike mogućnosti za rješavanje pojedinih dijelova zadatka i dobivanje dijela bodova.

Prvi podzadatak:

Je li red sadrži parne ili neparne brojeve ovisi o parnosti broja N. Ako je broj N paran, u retku su zapisani parni brojevi, a ako je neparan onda su u retku zapisani neparni brojevi.

Drugi podzatak:

Promatranjem slučajeva možemo uočiti da se su razlike brojeva po redcima redom: 2. red - 2, 3. red - 4, 4. red - 6, 5. red - 8, 6. red - 10...Uočavamo da je formula koja opisuje odnos retka i razlike 2 *N – 2.

Treći podzadatak:

Promatranjem slučajeva možemo uočiti da su zadnji brojevi u redcima redom: 1, 4, 7, 12, 17, 24, 31, 40.. Uočimo da su razlike tih broja redom 3, 3, 5, 5, 7, 7, 9, 9.. Sada lako možemo odrediti koji je posljednji broj u retku.

Četvrti podzadatak:

Zbroj brojeva u N-tom retku možemo dobiti simulacijom zbrajanja brojeva po retku.

Programski kod (pisan u Python 3)

```
N = int(input())
neparni = 1
parni = 2
zbroj = 0
# prvo pitanje
if N % 2 == 0:
    print('P')
else:
    print('N')
```

```
# drugo pitanje
print(2 * N - 2)
# treće pitanje
p = 0
pomak = 1
for i in range(N):
    p += pomak
    if i % 2 == 0:
        pomak += 2
print(p)
# četvrto pitanje
for i in range(N):
    zbroj = 0
    if i % 2 == 0:
        for j in range(i+1):
            zbroj += neparni
            neparni += 2
    else:
        for j in range(i+1):
            zbroj += parni
            parni += 2
print(zbroj)
```

Potrebno znanje: naredba ponavljanja

Kategorija: ad hoc i simulacija

6.1. Zadatak: Double — Autor: Nikola Dmitrović

Prvo treba provjeriti koliko je od pet zadanih brojeva dvoznamenkasto. Ako ih više od tri tada treba ispisati riječ DA i zbroj tri najveća broja. Ako nije, tada prvo treba brojeve koji su manji od 10 promijeniti u razliku koja nedostaje do 10, a brojeve veće od deset pretvoriti u nule. Nakon togatreba ispisati riječ NE i zbroj tri najmanja takva broja.

Programski kod (pisan u Python 3)

```
NP = int(input())
S = int(input())
A = int(input())
L = int(input())
B = int(input())
triple = [P, S, A, L, B]
```

```
ukupno = 0
for i in triple:
    if 9 < i < 100:
        ukupno += 1
if ukupno >= 3:
    print('DA')
    triple = sorted(triple)
    print(triple[2] + triple[3] + triple[4])
else:
    print('NE')
    for i in range(5):
        triple[i] = max(0, 10 - triple[i])
    triple = sorted(triple)
    print(triple[0] + triple[1] + triple[2])
```

Potrebno znanje: naredba odlučivanja

Kategorija: adhoc

6.2. Zadatak: Pane — Autor: Nikola Dmitrović

Vidi 5.3 Nepa.

6.3. Zadatak: Buffon — Autor: Josip Klepec

Štapiće možemo pamtiti pomoću 4 niza (r1, s1, r2, s2).

Najbitniji dio zadatka je za 2 štapića odgonetnuti dijele li njihove dužine zajedničku točku.

Rastavljamo na 3 slučaja:

- obje dužine su okomite;
- obje dužine su vodoravne;
- jedna dužina je okomita, druga vodoravna.

Prva dva slučaja svode se na provjeru jesu li dužine u istom retku (stupcu) te imaju li zajednički stupac (redak). Problem je sličan problemu presjeka 2 intervala.

Zadnji slučaj je provjera je li stupac okomite dužine između stupaca u kojem se nalazi vodoravna dužina te je li redak vodoravne dužine između redaka okomite dužine. (dvije ugnježđene for petlje).

Potrebno znanje: Naredbe odlučivanja, petlje, nizovi

Kategorija: ad hoc

7.1. Zadatak: Glasanje Autor: Vedran Kurdija

U zadatku je potrebno simulirati opisani postupak. U pomoćnom nizu pamtit ćemo raspone svih redova. U pomoćnoj matrici ili nizu vektora, za svaki ćemo red pamtiti koji se nastavnici u njemu nalaze. For petljom ćemo unositi nastavnike redom koji dolaze. Za svakog nastavnika proći ćemo po njegovom prezimenu i pronaći početna slova svih njegovih prezimena. U svrhu pronalaska možemo primijetiti da svako početno slovo prezimena slijedi crticu ili započinje cijelo prezime. Također, svako je početno slovo veliko.

Za svako pronađeno početno slovo nekog prezimena trenutnog nastavnika, proći ćemo po svim redovima abecedno te pronaći kojem redu ono pripada uspoređivanjem znakova početka i kraja reda sa početnim slovom prezimena. Kada pronađemo pripadajući red, provjerit ćemo nalazi li se u njemu manje ljudi nego u redu koji je za dosad obrađena prezimena tog nastavnika imao najmanje ljudi. Red s dosad najmanje ljudi možemo pamtiti u pomoćnoj varijabli, koju onda osvježavamo. Ako je broj ljudi u pripadajućem redu trenutnog prezimena i redu s najmanje ljudi za dosad obrađena prezimena tog nastavnika jednak, valja nam usporediti redove i odabrati manji te njime osvježiti pomoćnu varijablu.

Red u koji se smjestio zadnji nastavnik ispisujemo unutar for petlje za obradu nastavnika dodavanjem if naredbe koja provjerava radi li se o zadnjem nastavniku. Nakon toga, za ispis drugog reda, korištenjem pomoćne varijable pronalazimo red sa najmanje ljudi, sortiramo ga te ispisujemo. Za detalje pogledati izvorni kod.

Potrebno znanje: stringovi, for petlja, polja, naredba grananja

Kategorija: Simulacije

7.2. Zadatak: Mikado Autor: Josip Klepec

Vidi pod 6.3.

7.3. Zadatak: Babuška Autor:Vedran Kurdija

U zadatku je potrebno simulirati opisani postupak. Za jednostavan prikaz babuški u memoriji uvest ćemo dva pomoćna niza iznad i ispod, u kojima ćemo za svaku babušku pamtiti koja se babuška nalazi neposredno iznad nje, a koja neposredno ispod nje. Ako se iznad ili ispod ne nalazi nijedna babuška, na tim mjestima pisat će 0. Na početku su svi elementi spomenutih nizova jednaki 0.

For petljom ćemo redom unositi promjene te za svaku promjenu osvježiti pripadajuće elemente u nizovima iznad i ispod na sljedeći način:

```
ispod[iznad[x]] = 0;
iznad[ispod[y]] = 0;
iznad[x] = y;
ispod[y] = x;
```

Nakon provedenih promjena, korištenjem pomoćne varijable prebrojat ćemo koliko babuški u nizu iznad ima vrijednost 0, jer su to babuške koje se ne nalaze unutar neke druge babuške. Vrijednost ove pomoćne varijable rješenje je za prvi redak.

Za drugi redak proći ćemo po svim babuškama te ispisati -1 za one koje u nizu iznad imaju vrijednost 0. Primijetimo da je broj pojavljivanja broja -1 u drugom retku ispisa jednak prvom retku ispisa. Za babuške koje imaju neku drugu babušku iznad sebe, to jest u nizu iznad nemaju vrijednost 0, korištenjem while petlje "popet" ćemo se od njih do babuške koja iznad sebe nema nikoga, to jest u nizu iznad ima vrijednost 0:

```
while (iznad[trenutna] != 0)
    trenutna = iznad[trenutna];
```

Potrebno znanje: for petlja, while petlja, polja

Kategorija: Simulacije

8.1. Zadatak: Poravnanja — Autor: Stjepan Požgaj

U zadatku moramo odgovoriti na 3 pitanja: koja je najveći zbroj brojeva u stupcu ako su nizovi poravnati ulijevo, u sredinu ili udesno?

Da bismo odredili koji je najveći zbroj kad su stupci poravnati ulijevo dovoljno je for petljom proći po svim mogućim udaljenostima od početka, zbrojiti elemente iz nizova čija je duljina veća ili jednaka trenutnoj udaljenosti i uzeti najveći od takvih zbrojeva.

Slučaju kad nizovi poravnati udesno možemo riješiti analogno, no autoru se čini da je još bolje napisati funkciju koja prima nizove i odgovara na prvo pitanje i onda nju iskoristiti i za treće pitanje tako da funkciji proslijedimo nizove čije smo elemente prethodno poslagali u obrnutom redoslijedu.

Najteži slučaj je kad su nizovi poravnati u sredinu. No i u tom slučaju možemo elegantno iskoristiti ranije spomenutu funkciju. Nizove raspolovimo, lijevim polovicama obrnemo redoslijed elemenata i za takve nizove, posebno za lijeve i posebno za desne polovice pozovemo funkciju iz prvog dijela i kao rješenje uzmemo veći od ta dva rezultata.

Potrebno znanje: naredba grananja, for petlja, polja

Kategorija: ad-hoc

8.2. Zadatak: Čarape — Autor: Josip Klepec

Primijetimo da pohlepni algoritam koji sparuje čarape nasumično ne daje uvijek najbolje rješenje. Zapravo želimo da što više veličina čarapa ima tu jednu "čarapu viška" koja nije uparena jer će na taj način traženi broj biti maksimalan. Za početak grupiramo čarape po veličini. Odnosno za svaku veličinu izračunamo koliko ima čarapa te veličine. Nakon toga uparivat ćemo parove sljedećim postupkom sve dok nemamo dovoljno parova.

- Ako za par koji biramo sljedeći imamo više mogućnosti (više različitih veličina) izabrat ćemo onu koja i dalje nakon uparivanja može imati jednu čarapu viška (odnosno veličinu čarapa takvu da ima još barem 3 čarape te veličine).
- Ako takve nema svejedno je koji par uparimo. Nakon uparivanja smanjimo broj čarapa te veličine za 2.

Potrebno znanje: petlje, naredbe odlučivanja

Kategorija: ad hoc

8.3. Zadatak: Legić Autor: Gabrijel Jambrošić

Zadatak rješavamo tako da za svaki upit simuliramo sve operacije, a pritom održavamo sve moguće pozicije upitanog broja xi. To možemo tako da na početku i-tog upita sve kocke na kojima je broj xi označimo s 1, a sve ostale s 0. Zatim kod svake operacije premjestimo odgovarajuće kocke na odredišni toranj i održavamo kocke na kojima je 1 (na pozicije tih kocaka je moguće dovesti traženi broj iz upita). Prilikom prebacivanja kocaka ćemo neku kocku označiti jedinicom ako na njeno mjesto može doći kocka koja je također označena jedinicom (na jedan ili drugi način prebacivanja), a inače nulom. Tada možemo odabrati upravo taj način prebacivanja i dovesti kocku s traženim brojem na to mjesto.

Nakon izvršenih svih operacija dovoljno je pogledati nalazi li se na upitanom mjestu kocka označena s 0 ili s 1, na sve kocke koje su označene s 1 možemo dovesti traženi broj.

Složenost algoritma: O(Q*M*N*maxH)

Potrebno znanje: naredba grananja, for petlja, polja

Kategorija: Simulacije