15. veljače 2019.

2019 Natjecanje iz informatike

OPISI ALGORITAMA

Agencija za odgoj i obrazovanje
Education and Teacher Training Agency

HRVATSKI SAVEZ INFORMATIČARA

Ministarstvo znanosti, obrazovanja i sporta

5.1. Zadatak: Postaja Autor: Nikola Dmitrović

Odredimo najprije mogu li oba znanstvenika ući. Za to Z mora biti manji ili jednak 8 i K+X+Z<=1000. Ako ne mogu obojica onda u else dijelu if-ovima odredimo može li lakši od njih.

5.2. Zadatak: Šibice Autor: Nikola Dmitrović

Odredimo znamenke na sljedeći način:

jedinice=n%10
desetice=(n//10)%10
stotice=(n//100)%10
Sada if naredbama pribrojimo u rješenje pripadajući broj šibica.

Za odgovor na drugo pitanje gornji postupak trebamo napraviti unutar neke for petlje koja će ići po svim brojevima i za svaki od njih određivati koliko šibica treba da se taj broj prikaže.

5.3. Zadatak: Bombanje Autor: Nikola Dmitrović

Za odgovor na prvo pitanje prebrojimo koliko puta smo učitali da je Zg=Zi.

Drugi dio zadatka možemo rješiti tako da u jednoj varijabli pratimo koliko smo bombona zaradili, u drugoj koliko trebamo dati ako sljedeći put krivo odgovorimo, a u treći koliko ćemo dobiti ako sljedeći put točno odgovorimo.

6.1. Zadatak: Fašnik Autor: Nikola Dmitrović

Zadatak možemo rješiti if naredbama međutim ljepše rješenje je while petljom. Povećavat ćemo manji broj tako dugo dok Petra ima dovoljno bombona i dok Ana i Biba nemaju isti broj bombona.

Npr.

ako je A<B

```
while A<B and P-1>B:
    A += 1
    P -= 1
```

6.2. Zadatak: Umjetnica Autor: Nikola Dmitrović

Prvo pitanje je kao zadatak 5.2. Šibice. Za drugo pitanje imat ćemo pomoćnu varijablu u koju ćemo spremiti odgovor. Kada prolazimo po svim brojevima, ako se trenutni broj može prikazati s točno S šibica i ako je <= K, spremimo odgovor u našu pomoćnu varijablu.

6.3. Zadatak: Testomanija Autor: Nikola Dmitrović

Koliko je ukupno podzadataka točno riješeno?

Za ovaj dio, trebamo usporediti službena rješenja (SR) sa ponuđenim rješenjima (PR) za svaki podzadatak. Ako su SR i PR jednaki, tada je podzadatak točno riješen.

Koliko se ukupno bodova dobilo nakon vrednovanja svih N zadataka?

Za svaki zadatak, prvo trebamo odrediti broj točno riješenih podzadataka. Ovisno o broju točno riješenih podzadataka dodjeljujemo bodove prema pravilima zadatka:

4 točna: 10 bodova

3 točna: 6 bodova

2 točna: 3 boda

1 točan: 1 bod

Ukupno bodova je zbroj bodova za sve zadatke.

Koliko se najviše bodova može imati nakon promjene pogrešnih odgovora?

Ovdje, osim što želimo znati koliko bodova se dobilo, želimo znati i koliko bismo bodova mogli dobiti ako promijenimo nekoliko odgovora. Da bismo to učinili, prvo moramo pronaći one zadatke u kojima promjena jednog ili više odgovora može donijeti najviše dodatnih bodova. Prioritet bi trebao biti:

a) Zadaci gdje je 3 podzadatka točno, a 1 netočan - promjenom ovog netočnog odgovora dobit ćemo dodatnih 4 boda (10-6).

b) Zadaci gdje su 2 podzadatka točna, a 2 netočna - promjenom jednog netočnog odgovora dobit ćemo dodatnih 3 boda (6-3).

c) Zadaci gdje je 1 podzadatak točan, a 3 netočna - promjenom jednog netočnog odgovora dobit ćemo dodatnih 2 boda (3-1).

d) Zadaci gdje su svi podzadaci netočni nemaju prioritet jer promjenom jednog odgovora dobivamo samo 1 dodatni bod.

Počinjemo s a) i nastavljamo slijedom prema d) dok ne potrošimo sve promjene (K).

7.1. Zadatak: Trgovina — Autor: Marin Kišić

Kako se u svakih 10 uzastopnih nenegativnih brojeva pojavljuju sve moguće znamenke, dovoljno je bilo proći kroz brojeve X-10, X-9, ... , X, X+1, X+2, …, X+10 te pronaći traženi broj.

Programski kod (pisan u Python 3)

```
x = int(input())
y = int(input())
rj = -1
```

```
mini = 1111111
for i in range(x - 10, x + 11):
    if (i >= 0 and i % 10 == y and max(i - x, x - i) < mini):
        rj = i
        mini = max(i - x, x - i)
print (rj)
```

Potrebno znanje: for petlja, naredba odlučivanja

Kategorija: ad hoc

7.2. Zadatak: Look — Autor: Stjepan Požgaj

Da bismo riješili zadatak, moramo napisati dvije funkcije: jednu koja generira n-ti broj "pogledaj i reci" slijeda koji počinje s brojem X i drugu koja pretražuje početne brojeve kako bi pronašla odgovarajući broj slijeda koji sadrži broj Y.

1. Generiranje n-tog broja slijeda

Da bismo generirali slijed, moramo:

1. Prolaziti kroz svaki znak broja.
2. Brojati koliko uzastopnih puta se svaka znamenka pojavljuje.
3. Formirati novi broj na temelju broja uzastopnih znamenki i same znamenke.

2. Pronalaženje početnog broja slijeda za broj Y

Za ovaj dio, moramo proći kroz sve moguće dvoznamenkaste brojeve kao početni broj slijeda. Za svaki početni broj, generiramo slijed dok ne dođemo do broja s više ili jednakim brojem znamenki kao Y, a zatim uspoređujemo da vidimo je li to broj Y. Ako jest, vraćamo taj početni broj. Ako postoji više odgovarajućih brojeva, vraćamo najmanji.

7.3. Zadatak: Strah — Autor: Vedran Kurdija

Za početak dodijelimo svakoj od 5 mogućih vrsta predmeta po jedan indeks. Indeks vrste neka bude mjesto na kojem se ta vrsta nalazi u nizu vrsta sortiranom po abecedi. Dakle, indeks kacige bit će 1, indeks koplja bit će 2, indeks mača bit će 3, indeks oklopa bit će 4, a indeks štita bit će 5.

Za točan prvi redak ispisa bilo je potrebno pronaći vrstu predmeta koju je Đuro ponudio u najviše primjeraka. Kao rješenje ovog dijela zadatka prebrojat ćemo koliko puta se među Đurinim predmetima pojavila svaka od 5 mogućih vrsta predmeta i od njih ispisati onu koja se pojavila najviše puta.

Jednom ćemo for petljom ići od 1 do 5 (uključivo), obilazeći na taj način indekse svih vrsta. Za svaku od 5 vrsta iz prve for petlje, dodatnom ćemo (ugniježđenom) for petljom prebrojati koliko puta se ta vrsta nalazi među predmetima koje je ponudio Đuro. Također, cijelo ćemo vrijeme pamtiti koja se vrsta dosad pojavila najviše puta (u pomoćnoj string varijabli), kao i koliko se puta pojavila (u pomoćnoj int varijabli). Kada za neku od 5 vrsta iz prve for petlje drugom for petljom prebrojimo koliko se puta pojavila među Đurinih N predmeta, usporedit ćemo taj broj sa brojem pojavljivanja dosadašnje najponuđenije vrste

(imamo ga u pomoćnoj varijabli) te ako se trenutna vrsta pojavila više puta, ažurirati naše dvije pomoćne varijable.

Kao pripremu za drugi redak ispisa grupirat ćemo sve Đurine predmete po vrstama. U tu svrhu možemo stvoriti 5 nizova za snage, a 5 za mase predmeta, po jedan za svaku vrstu. U prvi će niz snaga redom ići snage svih ponuđenih kaciga, u drugi će niz snaga redom ići snage svih ponuđenih kopalja, i tako do petog niza, a analogno ćemo pohraniti i mase.

Nakon što smo grupirali Đurine predmete po vrstama, na sve ćemo načine pokušati odabrati predmete koje je Jura kupio te od svih mogućih opcija odabrati onu koja ima najmanju masu, a nadvladava Jurin strah. Učinit ćemo to korištenjem 5 ugniježđenih for petlji.

Prva će for petlja na sve moguće načine odabrati neki od Đurinih predmeta koji ima indeks vrste jednak 1, tj. kacigu, druga će for petlja za tu odabranu kacigu na sve moguće načine odabrati neki od Đurinih predmeta s indeksom vrste jednakim 2, tj. koplje, i tako dalje do indeksa 5. Ovime želimo isprobati sve moguće načine na koje se Jura mogao naoružati kako bismo od njih odabrali najpovoljniji. No, primijetimo da ovaj pristup pretpostavlja da je Jura kupio po jedan predmet od svake vrste (jer svaka for petlja odabire po jedan predmet), a moguće je da Jura predmet neke vrste nije morao kupiti, kao i da predmete neke vrste Đuro uopće nema u ponudi. Ovom ćemo problemu doskočiti tako da ćemo za svaku od 5 vrsta, u njezin niz snaga, kao i u njezin niz masa, dodati jednu nulu na kraj. Ovime smo efektivno za svaku od 5 vrsta predmeta dodali u Đurinu ponudu jedan "imaginarni" predmet mase 0 i snage 0. Sada će naše for petlje pri ispitivanju svih mogućih odabira uzimanjem tog "imaginarnog" predmeta simulirati situaciju kada Jura predmet te vrste nije uopće uzeo. Primijetimo da taj "imaginarni" predmet sigurno ne mijenja situaciju pri izračunu, budući da ne mijenja niti Jurinu masu niti Jurinu snagu, tako da smo mogli zamisliti da ga je Đuro ponudio bez da time promijenimo konačno rješenje.

Kao i u rješenju za prvi redak ispisa, koristit ćemo pomoćnu varijablu koja će nam tijekom ovog procesa pamtiti kolika je bila dosad najmanja masa koju smo pronašli, a da nadvladava Jurin strah. Na kraju će u toj varijabli biti pohranjeno rješenje.

Programski kod (pisan u Python 3)

```
#!/usr/bin/python3

n, strah = map(int, input().split())

indeks_predmeta = {
        'kaciga': 0,
        'koplje': 1,
        'mac': 2,
        'oklop': 3,
        'stit': 4,
        }

snage = [[] for i in range(5)]

mase = [[] for i in range(5)]

for _ in range(n):
    predmet, snaga, masa = input().split()
```

```
    i = indeks_predmeta[predmet]
    snage[i].append(int(snaga))
    mase[i].append(int(masa))

# Prvi dio zadatka.
br_primjeraka = [len(snage[i]) for i in range(5)]
max_primjeraka = 0
for i, predmet in enumerate(['kaciga', 'koplje', 'mac', 'oklop', 'stit']):
    if br_primjeraka[i] > max_primjeraka:
        max_primjeraka = br_primjeraka[i]
        najcesci_predmet = predmet
print(najcesci_predmet)

# Imaginarni primjerak (ako ne uzima taj predmet).
for i in range(5):
    snage[i].append(0)
    mase[i].append(0)
    br_primjeraka[i] += 1

# Prolazimo svim mogucnostima.
min_masa = float('inf')
for a in range(br_primjeraka[0]):
    for b in range(br_primjeraka[1]):
        for c in range(br_primjeraka[2]):
            for d in range(br_primjeraka[3]):
                for e in range(br_primjeraka[4]):
                    ukupna_snaga = snage[0][a] + snage[1][b] +\
                            snage[2][c] + snage[3][d] + snage[4][e]
                    ukupna_masa = mase[0][a] + mase[1][b] +\
                            mase[2][c] + mase[3][d] + mase[4][e]
                    if ukupna_snaga >= strah:
                        min_masa = min(min_masa, ukupna_masa)
print(min_masa)
```

Potrebno znanje: petlje, nizovi

Kategorija: ad hoc

8.1. Zadatak: Drevni Autor: Josip Klepec

Da bismo riješili ovaj problem, moramo napraviti dvije glavne provjere za svaku riječ:

Provjera sadrži li riječ određeno ime ("mirko" ili "slavko").

Nakon preuređivanja riječi (dio gdje se riječ dijeli na dva dijela, a zatim preuređuje), provjera sadrži li novostvorena riječ to ime.

Zatim možemo usporediti broj pojavljivanja oba imena u originalnim i preuređenim riječima i donijeti odluku na temelju tih brojeva.

8.2. Zadatak: Rebeka Autor: Josip Klepec

Najprije pogledajte zadatke i rješenja Sheldon i Umjetnica iz petog, odnosno šestog razreda. Odgovor na prvo pitanje možete pronaći u prijašnjim zadacima. Odgovor na drugo pitanje i treće pitanje ćemo dobiti iz jednog pomoćnog niza. U taj pomoćni niz spremimo parove (broj šibica koje mi trebaju da prikažem x, x). Sortiramo taj niz. Sada odgovor na drugo pitanje je K-ti broj u tom nizu, dok je odgovor na treće pitanje suma prvih K brojeva u tom nizu.

Primjetimo da je odgovor na 4. pitanje ispisati s/2 jedinica ako je s paran, a ako je neparan 7 pa onda (s-1)/2 jedinica.

8.3. Zadatak: Ruka Autor: Mislav Bradač

U prvoj skupini test primjera vrijedilo je da u zadanom kodu robotske ruke nema naredbe REPEAT. Da bismo riješili te primjere bilo je potrebno u jednom nizu pamtiti za svaku poziciju broj loptice koja se tamo nalazi te u jednoj varijabli poziciju ruke (pos), a u drugoj broj loptice (in_hand) koju trenutno drži. Kretanje ruke možemo simulirati naredbu po naredbu programa. Pomicanje ruke se svodi na uvećavanje ili smanjivanje varijable pos te provjeru je li vrijednost postala veća od broja loptica, odnosno je li postala manja od nule. Naredbu SWAP simuliramo tako da zamijenimo vrijednost polja s lopticama na trenutnoj poziciji ruke s vrijednosti varijable in_hand. Rješenje će nam se na kraju simulacije programa robotske ruke nalaziti u nizu u kojem smo pamtili brojeve na loptica za svaku poziciju.

U drugoj skupini primjera u kodu postoji naredba REPEAT, no garantirano je da će se u njezinom tijelu nalaziti samo jedna naredba. Te primjere riješavamo na sličan način kao i primjere iz prošle skupine, osim što ćemo naredbu unutar petlje izvesti onoliko puta koliko je koraka petlje. Alternativno se je moglo izmijeniti program tako da naredbu unutar petlje zamijenimo nekom drugom ili da ju izbrišemo. Točnije, naredbu SWAP možemo izbrisati ako se nalazi unutar petlje s parnim brojem ponavljanja, a naredbu LEFT i RIGHT ćemo transformirati tako da broj pomaka pomnožimo brojem ponavljanja petlje. Nakon tih transformacija program možemo izvesti kao da petlji uopće nema.

U posljednoj skupini primjera unutar petlje se može nalaziti proizvoljan broj naredbi. Te primjere rješavamo tako da izvršavamo naredbu po naredbu programa. Nakon izvršavanja naredbe SWAP, LEFT ili RIGHT pomičemo se na iduću naredbu. U trenutku dolaska do naredbe REPEAT u varijabli loop_count ćemo zapamtiti broj ponavljanja petlje, a u varijabli loop_start broj trenutne linije te ćemo se pomaknuti na iduću naredbu. Kod dolaska do naredbe ENDREPEAT moramo smanjiti vrijednosti varijable loop_count te u slučaju da je vrijednost veća od nula skočiti na liniju loop_start+1, a ako je nula

možemo se pomaknuti na iduću naredbu. Time smo uspješno simulirali i petlju programa programske ruke.

Programski kod (pisan u Python 3)

```
#!/usr/bin/env python3
n = int(input())
m = int(input())
code = [input() for i in range(m)]
a = [i + 1 for i in range(n)]
pos = 0
in_hand = 0
loop_start = -1
loop_count = 0
i = 0
while i < m:
    line = code[i].split()
    if line[0] == "LIJEVO":
            pos -= int(line[1])
            pos = max(0, pos)
            i += 1
    elif line[0] == "DESNO":
            pos += int(line[1])
            pos = min(n - 1, pos)
            i += 1
    elif line[0] == "ZAMJENA":
            in_hand, a[pos] = a[pos], in_hand
            i += 1
    elif line[0] == "REPEAT":
            loop_start = i + 1
            loop_count = int(line[1])
            i += 1
    elif line[0] == "ENDREPEAT":
            loop_count -= 1
        if loop_count == 0:
            i += 1
        else:
            i = loop_start
```

```
    else:
            assert False, "kriva naredba"
print(" ".join(map(str, a)))
```

Potrebno znanje: petlje, nizovi

Kategorija: ad hoc