2018 iz informatike Natjecanje

15. ožujka 2018.

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: Vrijeme Autor: Nikola Dmitrović

Preporučam da pročitate zadatak Štoperica i pogledate video koji objašnjava njegovo rješenje. Sličnim pristupom možemo riješiti ovaj zadatak. For petljama na sve moguće načine probamo sve kombinacije S i M, za svaku kombinaciju napravimo broj koji bi Ivica izgovorio da je toliko sati i ako je taj broj jednak broju u inputu ispišemo S:M.

5.2. Zadatak: Lokve Autor: Nikola Dmitrović

Kako bismo odgovorili na prvo pitanje, potrebno je započeti s lokacijom P i dodati korake X dok ne dođemo do kraja ceste (to jest do N). Tijekom svakog koraka, provjeravamo da li se lokacija glavnog lika nalazi na jednoj od lokvi. Ako je tako, povećavamo brojač za to.

Kako bismo odgovorili na drugo pitanje, moramo izračunati korak koji omogućuje liku da preskoči sve lokve. Počinjemo s korakom duljine 1 i radimo postupak kao za prvo pitanje samo sada nam duljina koraka nije X nego trenutna duljina koraka. Trenutnu duljinu koraka povećavamo sve dok lik ne može preskočiti sve lokve.

5.3. Zadatak: Slijedi Autor: Nikola Dmitrović

Zadatak rješavamo simulacijom uvjeta u tekstu zadatka. Možda najelegantnije rješenje je najprije učitati sve, zatim for petljom prolaziti po zadacima. Za pojedini zadatak možemo imati dvije while petlje. Prva radi tako dugo dok je učenik točno odgovarao. Za svaki točan odgovor odmah povečamo i rješenje za jedan i u neku varijablu odmah pamtimo zadnji točno odgovoren podzadatak. Nakon toga druga while petlja je zapravo identična prvoj samo trebamo dodati još jedna uvjet - razlika, tj. greška na trenutnom podzadatku treba biti jednaka kao i kod prve greške.

6.1. Zadatak: Autobus Autor: Nikola Dmitrović

U ovom zadatku najlakše nam je računati sve ako sva vremene preračunamo u minute. Tada možemo imati neku varijablu trenutno_vrijeme koju ćemo povećavati za X sve dok trenutno_vrijeme+X<Lukin dolazak na kolodvor. Nakon tih povećanja u trenutno_vrijeme nam piše zadnji bus na koji je Luka zakasnio, a trenutno_vrijeme+X nam prikazuje vrijeme polaska bus na koji je Luka uranio. Naravno, vremena treba ispisati u traženom formatu.

6.2. Zadatak: Štoperica Autor: Marin Kišić

Štoperica - Državno 2019

https://www.youtube.com/watch?v=VuTODWgfNNw

Pogledati rješenje na priloženom linku.

6.3. Zadatak: Crompiri Autor: Stjepan Požgaj

Kako bismo pratili ukupan broj krumpira koje je svaki natjecatelj ogulio tokom cijelog natjecanja, napravimo listu naziva "ukupnoOguljeno". Ova lista će se ažurirati svaki dan.

Sada dolazi ključni dio. Svaki dan, moramo provjeriti tko je ogulio najviše krumpira. Ovaj natjecatelj će biti prvi kandidat za posao. Ako je ovaj natjecatelj već zaposlen, moramo provjeriti tko je do tog trenutka ukupno ogulio najviše krumpira i nije već zaposlen. Da bismo pratili koji su natjecatelji već zaposleni, koristimo drugu listu pod imenom "zaposleni".

Dakle, svaki dan, slijedimo ovaj postupak:

Pogledamo tko je tog dana ogulio najviše krumpira.

Ako ovaj natjecatelj nije već zaposlen, odabiremo ga.

Ako je već zaposlen, gledamo ukupnu listu "ukupnoOguljeno" kako bismo vidjeli tko je do sada ukupno ogulio najviše krumpira i još uvijek nije zaposlen.

Ponavljamo ovaj postupak dok ne pronađemo natjecatelja koji će biti zaposlen tog dana.

Na kraju ovog postupka, imat ćemo listu svih zaposlenih natjecatelja, u redoslijedu u kojem su zaposleni.

7.1. Zadatak: Osmerokut — Autor: Mislav Bradač

For petljom možemo automatizirati računanje dijagonala, dok za ispis jednostavno napišemo formule.

```
for i in range(4):
        if i % 2: d.append(p[i] * p[(i + 3) % 8])
        else: d.append(p[i] * p[(i - 3 + 8)%8])

print(p[0] * p[7] + d[0] + d[2],end=' ')
print(p[0] * p[1] + d[0] + d[2] + d[1],end=' ')
print(p[1] * p[2] + d[1] + d[2])

print(p[6] * p[7] + d[0] + d[2] + d[3],end=' ')
print(d[0] + d[2] + d[1] + d[3], end = ' ')
print(p[2] * p[3] + d[1] + d[2] + d[3])

print(p[6] * p[5] + d[0] + d[3], end = ' ')
print(p[5] * p[4] + d[3] + d[0] + d[1], end = ' ')
print(p[4] * p[3] + d[3] + d[1])
```

7.2. Zadatak: Guljenje — Autor: Stjepan Požgaj

Vidi 6.3.

7.3. Zadatak: Puzzle — Autor: Mislav Bradač

Možemo for petljama prolaziti po slagalici od gore lijevo do dolje desno i na svakoj poziciji pokušati staviti neku od puzzli koju još nismo nigdje stavili. Kako bismo stavili puzzlu, s dodatne dvije for petlje prolazimo po puzzli i provjeravamo imamo li negdje kontradikciju, tj. ako je u puzzli 0, a na slagalici 1, tada ne mozemo postaviti puzzlu na tu poziciju (analogno i za 1 na puzzli). Dodatna komplikacija koja se može pojaviti je da je rupa prevelika. To možemo rješiti tako da privremeno postavimo puzzlu u slagalicu, zatim za svako polje puzzle koje je 1 provjerimo susjedna polja, ako postoji neko polje koje je trebalo biti pokriveno tada ipak ne možemo postaviti puzzlu na trenutnu poziciji. Ako puzzlu možemo staviti na tu poziciji, u neki pomoćni niz zapamtimo da smo stavili tu puzzlu te u slagalici sva polja koja puzzla prekriva označimo s 1.

8.1. Zadatak: Kameni — Autor: Josip Klepec

Zadatak rješavamo simulacijom rušenja zida. Imat ćemo matricu znakova koja predstavlja zid. Za svako bušenje mi zapravo dobijemo poziciju na rubu zida s lijeve ili desne strane. Nakon toga radimo sljedeći postupak. Ako je na zadanoj poziciji tvrdi kamen ili smo izašli van zida, prekidamo postupak i idemo na iduće bušenje. Inače sve redove iznad naše pozicije u trenutnom stupcu pomaknemo za jedan red prema dolje, a na mjesto najgornjeg reda upišemo znak 'X'. Nakon pomicanja redova pomaknemo poziciju u zadanom smjeru (lijevo/desno). Na kraju prebrojimo sve ne 'X' znakove u matrici.

8.2. Zadatak: Sretno — Autor: Vedran Kurdija

Za rješenje ovog zadatka treba nam pomoćni niz u kojem ćemo pamtiti prvo vrijeme kada je frizer slobodan. Sada za svaku mušteriju možemo proći po svim frizerima i za svakog frizera odredimo vrijeme kada bi trenutna mušterija bila gotova kod tog frizera prema formuli gotov = max(dolazak, slobodan[j]) + p[j] * z; Iz tih informacija sada možemo odrediti kod kojeg frizera će mušterija otići. Nakon što smo odredili frizera, trebamo postaviti slobodan[odabrani_frizer] na vrijeme kada će mušterija biti gotova kod tog frizera.

8.3. Zadatak: Buzdovan — Autor: Stjepan Požgaj