Školsko natjecanje iz informatike

Srednja škola

Prva podskupina (1. i 2. razred)

25. siječnja 2019.

RJEŠENJA ZADATAKA

Ministarstvo znanosti i obrazovanja

Agencija za odgoj i obrazovanje

HRVATSKA ZAJEDNICA TEHNIČKE KULTURE

HRVATSKI SAVEZ INFORMATIČARA

Možemo koristiti funkciju koja provjerava što ćemo dobiti ako iz **i**-te epruvete pretačemo u **j**-tu. U funkciji ćemo najprije izračunati koliko se otopine prebacuje: to će biti ili cijeli sadržaj **i**-te epruvete, ili prazan prostor u **j**-toj epruveti (ukupno – trenutno), ovisno o tome što se prije potroši, tj. što je manje. Kad izračunamo prebačenu količinu otopine, lako dobivamo količine u **i**-toj i **j**-toj epruveti nakon pretakanja. Ako je bilo koja od njih jednaka N, ispisujemo pretakanje **i** → **j** te prekidamo program. Dođemo li do kraja, a prekid se nije dogodio, nema rješenja pa ispisujemo 0.

```
#!/usr/bin/python3
n = int(input())
ukupno = list(map(int, input().split()))
trenutno = list(map(int, input().split()))

def pretakanje(i, j):
    prebaceno = min(trenutno[i], ukupno[j] - trenutno[j])
    if n in (trenutno[i] - prebaceno, trenutno[j] + prebaceno):
        print(i + 1, j + 1)
        exit(0)

for i in range(3):
    for j in range(3):
        pretakanje(i, j)
print(0)
```

Zadatak rješavamo isprobavanjem svih mogućnosti za poziciju najvišeg stupca.

Nakon što odaberemo tu poziciju, s lijeve strane gradimo rastući niz stupaca na pohlepan način: dodajemo uvijek najmanji broj kocaka koji moramo dodati da bi trenutačni stupac bio viši od prethodnog. Analogno činimo s desne strane, od kraja niza prema odabranom najvišem stupcu. Na kraju dodamo potreban broj kocaka na stupac koji treba biti najviši. Ako je ukupan broj dodanih kocaka dosad najmanji, pamtimo ga kao moguće rješenje i provjeravamo sljedeću mogućnost.

Valja paziti da pri svakom pokušaju radimo s kopijom niza, tako da nam izvorni niz ostane sačuvan za sljedeće pokušaje.

```
#!/usr/bin/python3
n = int(input())
a = list(map(int, input().split()))
min_dodanih = 10**6
for k in range(1, n - 1):
    dodanih = 0
    b = a[:]
    for i in range(1, k):
        if b[i] <= b[i - 1]:
            nova_visina = b[i - 1] + 1
            dodanih += nova_visina - b[i]
            b[i] = nova_visina
    for i in range(n - 2, k, -1):
        if b[i] <= b[i + 1]:
            nova_visina = b[i + 1] + 1
            dodanih += nova_visina - b[i]
            b[i] = nova_visina
    h = max(b[k - 1], b[k + 1])
    if b[k] <= h:
        nova_visina = h + 1
        dodanih += nova_visina - b[k]
        b[k] = nova_visina
    if min_dodanih > dodanih:
        min_dodanih = dodanih
print(min_dodanih)
```

Ovo je implementacijski zadatak – prilično je jasno što treba činiti, no ideju nije posve jednostavno bez greške pretočiti u programski kod. Slično prethodnom zadatku, probat ćemo na sve načine pomaknuti figuru lijevo ili desno, te je za svaki od tih slučajeva spustiti i izbrojiti dobivene rupe, tražeći minimum.

Savjeti za lakšu implementaciju:

- Figuru ćemo pretvoriti u niz visina njezinih stupaca, a isto ćemo učiniti sa stupcima donjeg dijela ekrana (tlo). Na taj način problem svodimo na baratanje nizovima brojeva, umjesto matricama, čime smo si olakšali posao.

- Računanje broja dobivenih rupa za odabrani položaj figure činimo tako da najprije izračunamo visinu figure nakon spuštanja – to će biti najveći zbroj visine nekog stupca figure i visine tla ispod njega. Znajući tu visinu, proći ćemo po stupcima figure i od dobivene visine oduzeti visinu stupca figure i stupca tla – razliku čine rupe.

- „Crtanje“ ekrana nakon spuštanja figure možemo činiti tako da u matricu koja je na početku ispunjena točkama dodajemo stupce tla i stupce figure iz izračunatih nizova.

```
#!/usr/bin/python3
r, s = map(int, input().split())
a = []
for i in range(r):
    a.append(list(input()))

figura = [0 for j in range(s)]
dno = [0 for j in range(s)]
sirina_figure = 0
prvi_stupac_figure = -1

for j in range(s):
    for i in range(r):
        if a[i][j] == '#':
            figura[j] += 1
        else:
            break
```

```
    if figura[j]:
        sirina_figure += 1
        if prvi_stupac_figure == -1:
            prvi_stupac_figure = j

    for i in range(r - 1, -1, -1):
        if a[i][j] == '#':
            dno[j] += 1
        else:
            break

br_rupa = []
for j in range(s):
    if j + sirina_figure > s:
        break
    visina = max(dno[i] + figura[prvi_stupac_figure + i - j]
            for i in range(j, j + sirina_figure))
    rupe = sum(visina - dno[i] - figura[prvi_stupac_figure + i - j]
            for i in range(j, j + sirina_figure))
    br_rupa.append(rupe)

j = br_rupa.index(min(br_rupa))
visina = max(dno[i] + figura[prvi_stupac_figure + i - j]
        for i in range(j, j + sirina_figure))
for k in range(r - visina):
    a[k] = ['.' for i in range(s)]
for i in range(j, j + sirina_figure):
    for k in range(figura[prvi_stupac_figure + i - j]):
        a[r - visina + k][i] = '#'
for i in range(r):
    print(''.join(a[i]))
```