**Školsko natjecanje iz informatike**

Srednja škola

Prva podskupina (1. i 2. razred)

18. siječnja 2018.

# RJEŠENJA ZADATAKA

Ministarstvo znanosti i obrazovanja

Agencija za odgoj i obrazovanje

HRVATSKA ZAJEDNICA TEHNIČKE KULTURE

HRVATSKI SAVEZ INFORMATIČARA

Zadatak se sastoji od dvaju manjih zadataka:

- **Pronalazak dimenzija tablice** vršimo ispitivanjem svih mogućnosti, tj. svih mogućih rastava N = R x S pri čemu je N duljina zadane riječi. Za svaki mogući broj redaka R (koje biramo for petljom) provjeravamo je li N djeljiv sa R i ako jest, računamo broj stupaca S i razliku S – R te u pomoćnoj varijabli pamtimo dimenzije (R, S) koje imaju najmanju pronađenu razliku.
- **Ispis riječi u tablicu** vršimo prolaskom po znakovima dane riječi, određujući polje u koje treba upisati taj znak uz pomoć dijeljenja rednog broja znaka s brojem stupaca i razlikujući dva slučaja ovisno o parnosti dobivenog retka.

```
#!/usr/bin/python3
def rs(n):
    best = (1, n)
    for r in range(2, n):
        if n % r == 0:
            s = n // r
            if s >= r and best[1] - best[0] > s - r:
                best = (r, s)
    return best

rijec = input()
r, s = rs(len(rijec))
a = [['.' for j in range(s)] for i in range(r)]
for i, znak in enumerate(rijec):
    redak = i // s
    if redak % 2 == 0:
        stupac = i % s
    else:
        stupac = s - 1 - i % s
    a[redak][stupac] = znak
for i in range(r):
    print(''.join(a[i]))
```

Ovo je zadatak u kojemu je prilično jasno što treba činiti, ali treba razmisliti kako to jednostavno i što elegantnije implementirati. Jedno moguće rješenje sastoji se od sljedećih koraka:

- Za svaki ton imamo string stanja odgovarajućih rupa koji definiramo uz pomoć slike iz teksta zadatka.
- Prolazimo redom po tonovima melodije i za trenutačni ton pronalazimo njegov redni broj u nizu tonova oktave.
- Uz pomoć tog broja dohvaćamo string odgovarajućih rupa, uspoređujemo ga s istim stringom za prethodni ton, te povećavamo rješenje za broj različitih znakova.

```
#!/usr/bin/python3
tonovi = [
    ['C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B'],
    ['C', 'Db', 'D', 'Eb', 'E', 'F', 'Gb', 'G', 'Ab', 'A', 'Bb', 'B']
    ]
pozicije = [
    'x xxx xxxx',
    'x xxx xxxo',
    'x xxx xxx.',
    'x xxx xxo.',
    'x xxx xx..',
    'x xxx x.xx',
    'x xxx .xx.',
    'x xxx ....',
    'x xx. xxo.',
    'x xx. ....',
    'x x.x x...',
    'x x.. ....',
        ]
n = int(input())
trenutacno = pozicije[0]
tezina = 0
for ton in input().split():
    for niz in tonovi:
        if ton in niz:
            i = niz.index(ton)
    tezina += len([j for j in range(10)\
            if pozicije[i][j] != trenutacno[j]])
    trenutacno = pozicije[i]
print(tezina)
```

Zadatak je moguće riješiti pretraživanjem u širinu (tzv. BFS), koje zbog neefikasnosti tj. prekoračenja vremenskog ograničenja u ovom zadatku osvaja 9/10 test podataka. To rješenje ostavljamo čitatelju za vježbu. Ovdje opisujemo vrlo efikasno rješenje koje radi i za daleko veće brojeve N.

Rješenje se temelji na promatranju niza najbolje ocijenjenih zaposlenika u konačnom nizu, tj. onih čije ocjene trebaju biti N, N – 1, N – 2, N – 3, i tako dalje. Ono što sigurno možemo učiniti jest kazniti sve zaposlenike točno tim redoslijedom. Time dobivamo upravo tražene ocjene, ali ne nužno s najmanjim brojem kažnjavanja.

Primijetimo da iz tog redoslijeda kažnjavanja možemo izostaviti prvog zaposlenika – njegova će se ocjena povećati točno onoliko puta koliko ima bolje ocijenjenih zaposlenika, što znači da će na kraju biti upravo N, što i želimo.

Na isti način zaključujemo da iz tog redoslijeda kažnjavanja možemo izostaviti i drugog zaposlenika (koji treba dobiti ocjenu N – 1) ako će se njegova ocjena povećati ispravan broj puta. To će biti slučaj ako je zaposlenik iz prethodnog odlomka inicijalno ocijenjen bolje od njega.

U tom slučaju, mogli bismo izostaviti i trećeg zaposlenika (koji treba dobiti ocjenu N – 2) ako će se njegova ocjena povećati ispravan broj puta, a to će biti slučaj ako su oba zaposlenika iz prethodnih odlomka inicijalno ocijenjena bolje od njega.

Dakle, prolazimo po zaposlenicima navedenim redoslijedom dok god vrijedi uvjet da su prethodno posjećeni zaposlenici ocijenjeni bolje od trenutačnog. Kada to prestane vrijediti, kažnjavamo redom tog zaposlenika i sve iduće.

```
#!/usr/bin/python3
n = int(input())
a = list(map(int, input().split()))
b = list(map(int, input().split()))
last = n + 1
for k in range(n, 0, -1):
    i = b.index(k)
    if a[i] < last:
        last = a[i]
    else:
        last = 0
        print(i + 1)
```