Opisi algoritama

Zadatke, testne primjere i rješenja pripremili: Fabijan Bošnjak, Nikola Dmitrović, Marin Kišić, Josip Klepec, Daniel Paleka, Ivan Paljak, Tonko Sabolčec i Paula Vidas. Primjeri implementiranih rješenja su dani u priloženim izvornim kodovima.

Zadatak: Koeficijent

Pripremio: Nikola Dmitrović
Potrebno znanje: naredba učitavanja i ispisivanja

Nekoliko je različitih načina na koje možemo riješiti ovaj zadatak. Najjednostavniji način je ispis izraza oblika $(N-1)+1$. Uočite da ispis razmaka, koji Python obavezno dodaje, nije dozvoljen što nas dovodi do činjenice da u naredbi ispisa trebamo koristiti svojstva separatora `sep`.

Programski kod (pisan u `Python 3`):

```
N = int(input())
print(N-1,'+',1,sep = '')
# može i ovako.. print(N-1,1,sep='+')
```

Drugačije rješenje moglo je ići u smjeru kreiranja stringa oblika $1+1+\cdots+1$ s ukupno N jedinica.

Programski kod (pisan u `Python 3`):

```
N = int(input())
s = '1'
for i in range(N-1):
    s += '+1'
print(s)
```

Zadatak: Hajduk

Pripremio: Fabijan Bošnjak
Potrebno znanje:

Zadatak: Preokret

Pripremio: Nikola Dmitrović
Potrebno znanje: naredba ponavljanja, rad s nizovima

Zadatak se sastoji od tri dijela. Za svakog natjecatelja po nešto. Krenimo redom. Da bi odrediti koliko je koji tim postigao golova dovoljno je učitavati zadane vrijednosti i brojati koliko se puta pojavio broj 1, a koliko broj 0.

Programski kod (pisan u `Python 3`):

```
N = int(input())
city = protivnik = 0
for i in range(N):
    gol = int(input())
    if gol == 1:
      city += 1
    else:
      protivnik += 1
print(city, protivnik)
```

Nastavimo dalje. Da bi odredili i koliko se puta dogodio neriješen rezultat trebamo pratiti kada je broj postignutih golova jednak, tj. kada će razlika postignutih golova biti jednaka nuli.

Programski kod (pisan u `Python 3`*):*

```
N = int(input())
city = protivnik = 0
nerijeseno = 1 # zbog 0:0
for i in range(N):
    gol = int(input())
    if gol == 1:
      city += 1
    else:
      protivnik += 1
    nerijeseno += (city == protivnik)
print(city, protivnik)
print(nerijeseno)
```

Za treći dio zadatka uočimo da se preokret sastoji od tri dijela. Prvo, jedan od timova gubi i počne postizati golove. Drugo, nakon niza postignutih golova dođe do neriješenog rezultata. Treće, tu ne stane već nastavi davati golove. Znači, razlika postignutih golova prvo pada do nule i onda nastavi rasti. Ili obrnuto, ovisno jel li City pravi preokret ili njegov protivnik. Jednu od implementacija rješenja možete pronaći u priloženim izvornim kodovima.

Zadatak: Grudanje

Pripremio: Marin Kišić
Potrebno znanje:

Zadatak: Drvca

Pripremili: Marin Kišić i Josip Klepec
Potrebno znanje:

Zadatak: Lampice

Pripremio: Tonko Sabolčec
Potrebno znanje:

Zadatak: Sob

Pripremili: Paula Vidas i Daniel Paleka
Potrebno znanje: matematika, pohlepni algoritmi

Uređeni par (a, b) zvat ćemo *dobrim* ako vrijedi $a \,\&\, b = a$.

Prvi podzadatak možemo riješiti tako da $a \in A$ uparimo sa onim $b \in B$ za kojeg vrijedi $b \bmod N = a$.

Drugi podzadatak možemo riješiti sljedećim algoritmom: Neka su $i_1 > i_2 > ... > i_k$ pozicije jedinica u binarnom zapisu od N. Uparit ćemo najmanjih 2^{i_1} elemenata skupova A i B tako da uparimo one a i b za koje vrijedi $a \equiv b \mod 2^{i_1}$. Zatim uzmemo sljedećih 2^{i_2} najmanjih elemenata i uparimo one koji su jednaki modulo 2^{i_2}, itd. Dokaz da su odabrani parovi dobri ostavljamo čitatelju za vježbu.

Treći podzadatak mogao se riješiti na više načina. Jedan mogući način je da napravimo bipartitni graf sa čvorovima iz skupova A i B te dodamo bridove između svih dobrih parova. Na dobivenom grafu napravimo algoritam za uparivanje na bipartitnom grafu (*bipartite matching*), na primjer u složenosti $\mathcal{O}(NE)$, pri čemu je E broj bridova u grafu. Primjetimo da broj bridova možemo ograničiti sa $E < 3^{10} = 59049$.

Drugi način koristi sljedeći pohlepni algoritam: Prolazimo kroz elemente skupa A od većih prema manjima i trenutni element uparimo sa najmanjim još neuparenim elementom skupa B s kojim ga smijemo upariti.

Ako pokrenemo taj algoritam na nekoliko primjera, možemo uočiti sljedeću pravilnost: Neka se najveći element skupa A, tj. $N-1$, upari sa $b \in B$. Tada se upare i $N-1-t$ sa $b-t$ za svaki $t \in \{1, 2, ..., b-M\}$. Nakon što maknemo uparene elemente dobili smo isti zadatak, sada za skupove $A' = \{0, 1, ..., N-1-(b-M)-1\}$ i $B' = \{b+1, b+2, ..., M+N-1\}$. Ovo rješenje možemo implementirati u složenosti $\mathcal{O}(N)$.

Dokaz prethodne tvrdnje:
Neka je $a = N-1$ (radi ljepših oznaka) i b, kao i prije, najmanji element skupa B za kojeg vrijedi $a \,\&\, b = a$. Indeksom i ćemo označavati znamenku težine 2^i. Ako je $b = M$ nemamo što za dokazivati, pa pretpostavimo da je $b > M$ i označimo $k = b - M$. Neka je i pozicija najmanje značajne jedinice u b. Očito mora biti $a_j = b_j = 0$ za $j < i$. Kada bi bilo $a_i = 0$ onda bi vrijedilo $a \,\&\, (b-1) = a$ pa b ne bi bio najmanji element od B koji se može upariti sa a. Dakle, $a_i = b_i = 1$. Sada je očito da je $(a-t, b-t)$ dobar par za $t \in \{1, 2, ..., 2^i\}$. Ako je $k \leq 2^i$ gotovi smo, inače promatramo sljedeću najmanje značajnu jedinicu u b i induktivno ponavljamo isti postupak. Preostaje još pokazati da uvijek postoji neki $b \in B$ s kojim se a može upariti, no to ostavljamo čitateljici za vježbu.