

### ZBROJ

Kako rezultat ne stane nužno u 32-bitni cjelobrojni tip podataka, gradit ćemo ga u string odmah prilikom računanja. Službena rješenja u različitim jezicima pokazuju različite pristupe implementaciji, dijelom uvjetovane mogućnostima jezika:

- Rješenje u jeziku C učitava brojeve kao stringove, izvodi zbrajanje zdesna nalijevo i gradi rezultat unazad
- Rješenje u Pascalu učitava brojeve kao integere i također gradi rezultat unazad, ali je kôd jednostavniji zbog većih mogućnosti rada sa stringovima
- Rješenje u jeziku C++ učitava brojeve kao stringove i nadopunjava ih nulama slijeva da budu jednake duljine, a zatim zbraja slijeva nadesno, što smijemo jer nema zavisnosti među rezultatima zbrajanja pojedinih znamenki

### VLAK

Za svaki vagon pamtimo:

- Ukupan broj putnika u vagonu
- Za svako slovo, koliko putnika čije ime počinje tim slovom se nalazi u tom vagonu

Na temelju ovih podataka možemo jednoznačno odrediti za svakog putnika u koji vagon ulazi te zatim povećati odgovarajuće brojeve za taj vagon. Na kraju ispišemo ukupan broj putnika u svakom vagonu.

### TETRIS

Svaku od figura u nekoj od njezinih rotacija možemo zapisati kao niz brojeva: zamislimo da ona leži u praznom polju, pa za svaki stupac zapišemo broj najnižeg retka koji zauzima u tom stupcu.

Tako ćemo npr. za figuru broj 5 u njene četiri rotacije dobiti sljedeća četiri zapisa:

$\{1, 1, 1\}$ ,  $\{1, 2\}$ ,  $\{2, 1, 2\}$ ,  $\{2, 1\}$ .

Za niz A kažemo odgovara nizu B ako se niz B može dobiti tako da svakom elementu niza A dodamo neki broj C. Tako npr. niz  $\{2, 1, 2\}$  odgovara nizu  $\{5, 4, 5\}$ , ali ne odgovara nizovima  $\{4, 5, 4\}$  i  $\{5, 4\}$ .

Primjećujemo da se figura u nekoj od rotacija može ubaciti u polje, ako podniz visina stupaca na koje će figura leći odgovara zapisu figure u toj rotaciji.

Za svaki zapis figure prebrojimo koliko postoji podniza u nizu stupaca koji odgovaraju zapisu figure – na toliko načina možemo figuru baciti u polje.

Kako ne bismo više puta pribrojali istu konfiguraciju, trebamo još obratiti pozornost na to da figuru broj 2 ne možemo rotirati (sve četiri rotacije su jednake), a figure brojeva 1, 3 i 4 možemo samo jednom rotirati za 90 stupnjeva (rotacijom za 180 stupnjeva figura se vraća u početno stanje).

Rješenje u jeziku C++ prikazuje ovaj pristup, a u rješenjima u C-u i Pascalu su ručno raspisani uvjeti u kojima se figura može smjestiti u polje.

### KRUG

Za početak, napravimo Stankovu transformaciju  $K$  puta da dobijemo konačni krug. Zanima nas koliko je različitih početnih krugova moglo dati dobiveni konačni, što znači da iz konačnog kruga radimo obrnutu transformaciju  $K$  puta.

Obrnuta transformacija nije jednoznačna, ali je najviše dvoznačna. Naime, ako pretpostavimo neki element iz prethodnog kruga, tad je tim elementom i trenutnim krugom jednoznačno određen prethodni krug. Kako taj jedan element možemo izabrati na dva načina, slijedi da postoje najviše dva moguća prethodna kruga. Napravimo li rekurzivno obrnutu transformaciju  $K$  puta, dobit ćemo  $2^K$  mogućih početnih krugova.

Potrebno je još izbaciti jednake krugove. Hoćemo li to raditi na kraju ili za vrijeme generiranja je implementacijski detalj, međutim način na koji ćemo provjeravati jesu li krugovi različiti bitno utječe na ukupnu složenost algoritma:

- Uspoređujemo li na kraju naivno svaki krug sa svakim drugim, rotirajući pritom oba kruga na svih  $N$  načina, složenost je u najgorem slučaju  $O(2^K \cdot N^3)$ .
- Primijetimo li da je dovoljno rotirati samo jedan krug, a drugi držati fiksni, složenost se smanjuje na  $O(2^K \cdot N^2)$ .
- Za svaki krug možemo definirati tzv. normalnu ili kanonsku formu, npr. leksikografski najmanji niz od svih nizova koje dobivamo rotacijom. Sad je za jednakost krugova dovoljno provjeriti jednakost njihovih kanonskih formi. Ukoliko krugove ubacujemo u skup implementiran balansiranim binarnim stablom (npr. klasa `set` u jeziku C++), složenost je  $O(2^K \cdot (N + \log(2^K)) \cdot N) = O(2^K \cdot (N + K) \cdot N)$ .
- Ukoliko koristimo efikasniju strukturu podataka, npr. trie, složenost se može smanjiti na  $O(2^K \cdot N^2)$  ili čak  $O(2^K \cdot N)$  s hash tablicom.

Iako neke od navedenih složenosti izgledaju velike, ograničenja u zadatku i konstante skrivene asimptotskom notacijom su male pa je većina rješenja dovoljno brza.