## INSTRUKCIJE

Uz pomoć dvije ugnježđene petlje generiramo niz koji se sastoji redom od jednog broja 1, dva broja 2, tri broja 3 itd.

Nakon toga, zbroj brojeva između A-tog i B-tog broja jednostavno pronalazimo zbrajajući brojeve koji pišu u nizu na tim lokacijama.

## PJESMA

Dodijelimo svakoj riječi u naslovu pjesme oznaku koja govori je li se riječ pojavila u tekstu pjesme.

Za svaku riječ iz teksta pjesme tražimo postoji li u naslovu pjesme ta riječ, te ako se nije dosad pojavila označimo je i povećamo brojač riječi koje su se pojavile za jedan.

U trenutku kada brojač postane veći od ili jednak N/2, prekidamo algoritam i ispisujemo redni broj zadnje riječi iz teksta pjesme.

## LAGNO

Za svako prazno polje računamo broj bijelih kamenčića koje možemo pretvoriti u crne ako stavimo kamenčić na to polje. To radimo tako da postavimo brojač kamenčića na 0, te za svaki od osam smjerova (gore, dolje, lijevo, desno i četiri dijagonalna smjera) radimo sljedeće:

Pomičemo se u odabranom smjeru sve dok ima bijelih kamenčića na promatranom polju. Ako se nakon tog niza bijelih kamenčića na promatranom polju nalazi crni kamenčić tada povećamo brojač kamenčića za ukupan broj bijelih kamenčića u nizu.

## NIKOLA

Modelirajmo zadatak kao traženje najkraćeg puta u grafu u kojem stanje predstavljamo parom (*polje*, *skok*), gdje *polje* predstavlja polje na kojem se Nikola nalazi, a *skok* predstavlja duljinu prethodnog skoka.

Iz stanja (*polje*, *skok*) možemo prijeći u stanje (*polje* - *skok*, *skok*) skokom unatrag, te u stanje (*polje* + *skok* + 1, *skok* + 1) skokom unaprijed, ako tim skokovima ne bismo ispali iz niza.

Vidimo da je graf acikličan, odnosno da ne postoji niz prijelaza kojim bi se vratili u trenutno stanje, jer skokovi unaprijed povećavaju parametar *skok*. Zbog toga duljinu najkraćeg puta možemo izračunati metodom dinamičkog programiranja.

Neka funkcija opt( *polje*, *skok* ) određuje najmanji ukupni iznos pristojbi koje Nikola treba platiti da dođe s polja *polje* na polje N, ako je prethodni skok bio duljine *skok*.

Funkciju izračunavamo ovako:

- za *polje* < 1 ili *polje* > N vrijedi opt( *polje*, *skok* ) = ∞
- za polje = N vrijedi opt( *polje, skok* ) = pristojba[N]
- inače vrijedi opt( *polje*, *skok* ) = pristojba[*polje*] +
  min{ opt( *polje*-*skok*, *skok* ), opt( *polje*+*skok*+1, *skok*+1 ) }

Rješenje, odnosno najmanji ukupni iznos pristojbi koje Nikola treba platiti da dođe s polja 1 na polje N tada iznosi opt( 2, 1).