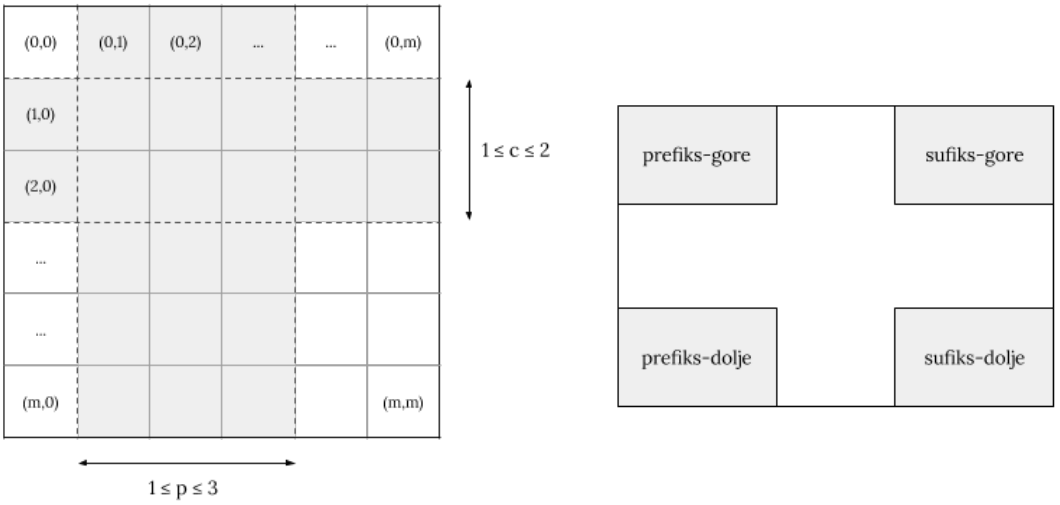
Opisi algoritama

Zadatke, testne primjere i rješenja pripremili: Domagoj Bradač, Marin Kišić, Adrian Satja Kurdija, Vedran Kurdija, Ivan Paljak, Tonko Sabolčec i Paula Vidas. Primjeri implementiranih rješenja dani su u priloženim izvornim kodovima.

Zadatak: Baloni

Autor: Adrian Satja Kurdija
Pripremili: Tonko Sabolčec, Adrian Satja Kurdija i Domagoj Bradač
Potrebno znanje: sweep-line algoritmi, računanje površine unije/presjeka

Kombinacije plavih i crvenih balona možemo prikazati $(m+1) \times (m+1)$ tablicom (npr. redci označavaju broj crvenih, a stupci broj plavih balona). Primjetite da za svaku izjavu skup parova crvenih i plavih balona ima oblik "plusa" u tablici.



Rješenje je tada površina presjeka svih takvih pluseva. No, lakše će biti izračunati površinu nezadovoljavajućih parova balona i tu vrijednost oduzeti od ukupnog broja polja u tablici — $(m+1)^2$. Zanima nas površina unije rubnih pravokutnika koje možemo podijeliti u 4 skupine (vidi gornju desnu sliku).

Za početak ćemo zasebno izračunati površinu unije za svaku od 4 skupine. Na primjeru gornje lijeve skupine, u vrijednost `prefiks_gore[i]` bit će zapisan broj polja unije u i-tom stupcu. Niz možemo konstruirati tako da postupno prolazimo od posljednjeg stupca tablice prema početnom te pamtimo trenutnu maksimalnu visinu pravokutnika iz te skupine.

Analogno određujemo vrijednosti za preostale skupine pravokutnika. Zatim ćemo izračunati uniju gornjih pravokutnika (tako da za svaki stupac i odredimo `gore[i] = max(prefiks_gore[i], sufiks_gore[i])`) i analogno tome uniju donjih pravokutnika, `dolje[i]`. Konačno, broj nezadovoljavajućih polja za i-ti stupac dobivamo kao `min(m + 1, gore[i] + dolje[i])`, a iz toga lako dobivamo ukupan broj zadovoljavajućih parova. Složenost ovakvog algoritma iznosi $\mathcal{O}(m)$ i dovoljna je za prva tri podzadatka.

Posljednji podzadatak rješavamo koristeći tehniku *sažimanja* (primijetite da ćemo imati $\mathcal{O}(n)$ *različitih* stupaca, stoga ih možemo grupirati tako da pojedino polje tablice gledamo kao uzastopni niz stupaca s istim *svojstvima*). Složenost tada iznosi $\mathcal{O}(n \log n)$.

Zadatak: Bitstring

Inspiracija: Rusija 2015., zadatak Пингвиноведение[1]
Pripremio: Adrian Satja Kurdija
Potrebno znanje: dinamičko programiranje, strukture podataka

Trebamo pronaći niz koji se sastoji od najviše k blokova istih znamenki, a među njima onaj koji se od početnog razlikuje u minimalnom broju pozicija. Ako duljina izvornog niza nije velika, tada možemo proći kroz sve moguće nizove u $O(2^n)$ i provjeriti jesu li podijeljeni u potreban broj blokova. Za one koji jesu pamtimo minimalnu cijenu – broj znakova u kojima se taj niz razlikuje od izvornika. Ovo rješenje nosi 20 bodova.

Neka $ans[i][j]$ označava optimalan odgovor za prefiks izvornog niza duljine i, ako ga podijelimo u j blokova.
Inicijalizacija: prazan prefiks može se podijeliti na bilo koji broj blokova, cijena bilo kojeg dijeljenja je 0.
Prijelaz: iteracijom po pozicijama $x \leq i$ određujemo mjesto početka posljednjeg bloka, a zatim

$$ans[i][j] = \min_{x=1}^{i}(ans[x-1][j-1] + trosak[x][i]),$$

gdje $trosak[x][i]$ odgovara optimalnom razdvajanju intervala $[x, i]$ u jedan blok, a računa se kao manji od broja nula i broja jedinica u tom intervalu.
Odgovor na problem sadržan je u $ans[n][k]$. Ovo rješenje nosi 40 bodova i ima složenost $O(N^2K)$.

Neka $ans[i][j][c]$ označava optimalan odgovor ako prefiks duljine i podijelimo na j blokova, a zadnji blok čine znakovi c.
Prijelaz: sljedeći znak može započeti novi blok ili nastaviti stari, tj.

$$ans[i][j][c] = (s[i] \neq c) + \min(ans[i-1][j][c], ans[i][j-1][0], ans[i][j-1][1])$$

Ovo rješenje nosi 60 bodova i ima složenost $O(NK)$.

Zamislimo da smo cijeli izvorni niz pretvorili u jedan blok nula. Sada ćemo odabrati neke blokove (najviše $k/2$ njih) koji će se sastojati od jedinica. Odaberemo li neki blok i promijenimo ga u jedinice, cijena particije mijenja se za $-$broj jedinica + broj nula. Dakle, ako izvorni niz brojeva 0 i 1 zamijenimo nizom brojeva -1 i 1, potrebno je riješiti sljedeći problem: u nizu treba odabrati najviše $k/2$ disjunktnih intervala s maksimalnim zbrojem.

Pretpostavimo da u nizu trebamo odabrati k disjunktnih intervala s maksimalnim zbrojem. Za $k = 1$ potrebno je pronaći jedan interval s maksimalnim zbrojem, to je standardni problem. Odgovor ćemo konstruirati postupno: pretpostavimo da smo već odabrali k intervala, a želimo odabrati $k + 1$. Tvrdimo da treba ili dodati interval disjunktan sa svima već odabranima, ili jedan od odabranih intervala podijeliti na dva dijela, "izrezujući" neki dio iz njega. Za obje svrhe potrebno je moći pronaći podinterval s maksimalnim/minimalnim zbrojem na intervalu, što možemo tournament stablom. Budući da u svakom koraku postoji $O(k)$ intervala za navedenu provjeru, ovo rješenje može se implementirati u složenosti $O(k^2 + n \log n)$ i nosi 80 bodova.

Da bismo optimirali rješenje, valja primijetiti da se naše mogućnosti vrlo malo mijenjaju iz koraka u korak: za upit se pojavljuju tri nova intervala a uklanja se jedan. Dakle, s pomoću strukture prioritetnog reda možemo odabrati optimalnu radnju za svaki korak. Ovo je rješenje složenosti $O(n \log n)$.

[1] http://neerc.ifmo.ru/school/archive/2014-2015/ru-olymp-roi-2015-day1.pdf

Zadatak: Kalendar

Autor: prof. dr. sc. Željko Ilić (FER)
Pripremio: Adrian Satja Kurdija

Zadatak rješavamo simulacijom po danima. Održavamo trenutačni datum i trenutačnu oznaku datuma na satu. U while petlji najprije pomičemo oznaku dana na satu za +1 (osim ako je 31, kada je pomičemo na 1), a potom trenutačni datum uvećavamo za jedan, te provjeravamo podudaranje točnog datuma i onog na satu. U pomoćnoj varijabli pamtimo je li sat već počeo "kasniti" i zaustavljamo petlju kada se nakon kašnjenja dogodi podudaranje. Pri uvećavanju trenutačnog datuma treba voditi računa o prijelazu iz mjeseca u mjesec, kada je važno koliko mjesec ima dana i ako je veljača, je li godina prijestupna.

Zadatak: Mnogokut

Autor: Paula Vidas
Pripremili: Adrian Satja Kurdija i Paula Vidas
Potrebno znanje: metoda dvaju pokazivača (two pointers)

Trojku (različitih) vrhova mnogokuta zvat ćemo *dobrom* ako je moguće dani mnogokut saviti u trokut s tim vrhovima. U suprotnom, kažemo da je trojka *loša*.

20% bodova moguće je ostvariti jednostavnim *brute force* algoritmom. Za svaku trojku vrhova provjerimo je li dobra. Trojka je dobra ako i samo ako za duljine stranica, tj. zbrojeve duljina štapića između vrhova, vrijedi da je duljina svake stranice manja od zbroja duljina druge dvije. Algoritam je moguće implementirati u složenosti $\mathcal{O}(n^3)$.

Za ostatak rješenja ključna ideja je sljedeća: umjesto da brojimo dobre trojke, brojat ćemo loše. Na kraju od ukupnog broja trojki, koji je jednak $\binom{n}{3}$, samo oduzmemo broj loših.

Trojka je loša ako i samo je duljina najveće stranice veća ili jednaka zbroju duljina druge dvije stranice. To vrijedi ako i samo ako je duljina najveće stranice veća ili jednaka polovici opsega.

Fiksirajmo vrhove najdulje stranice. Ako su zbrojevi duljina s obje strane jednaki, treći vrh možemo odabrati proizvoljno. Inače, treći vrh može biti bilo koji vrh na "kraćoj" strani. Ovaj pristup moguće je implementirati u složenosti $\mathcal{O}(n^2)$, što je dovoljno za 50% bodova.

Sada ćemo opisati puno rješenje. Za vrh x označimo s $f(x)$ prvi vrh u smjeru kazaljke na satu za koji je zbroj duljina štapića od x do $f(x)$ veći ili jednak polovici opsega. Neka je k jednak broju vrhova strogo između x i $f(x)$ (u smjeru kazaljke na satu). Tada vrijedi da je broj loših trojki u kojima je x jedan od vrhova najdulje stranice, a drugi vrh je onaj koji mu je bliži u smjeru kazaljke na satu, jednak $\binom{n-k-1}{2}$. Naime, za druga dva vrha možemo uzeti bilo koje vrhove između $f(x)$ i x, uključujući $f(x)$, ali naravno ne i x.

Kako efikasno naći $f(x)$? Koristit ćemo tzv. metodu dva pokazivača. Vrhove mnogokuta označimo s $1, 2, \ldots n$ u smjeru kazaljke na satu. Na početku oba pokazivača pokazuju na vrh 1. Pomičemo drugi pokazivač u smjeru kazaljke na satu sve dok je zbroj duljina između pokazivača manji od polovice opsega. Kad stanemo, našli smo $f(1)$. Pomaknimo sad prvi pokazivač u vrh 2. Primjetimo da se $f(2)$ ne može nalaziti strogo između 2 i $f(1)$. Opet pomičemo drugi pokazivač dok zbroj ne postane veći ili jednak polovici opsega, odnosno dok ne nađemo $f(2)$. Zatim isti postupak ponavljamo redom za sve vrhove $3, 4, \ldots n$. Na kraju, prvi pokazivač je napravio jedan, a drugi pokazivač najviše dva kruga oko mnogokuta, pa je složenost ovog rješenja $\mathcal{O}(n)$.

Alternativno, $f(x)$ možemo naći binarnim pretraživanjem. U tom slučaju složenost je $\mathcal{O}(n \log n)$.