Novada olimpiāde - 2020

Cik labo?

Sākumā vajag noskaidrot, kuri skaitļi ir palikuši pēc visu gājienu izdarīšanas. No dotā skaitļu intervāla mēs izņemam vairākus skaitļu intervālus, kas savā starpā var pārklāties. Lai noskaidrotu, kuri skaitļi ir palikuši, sakārtojam visu gājienu intervālu galapunktus augošā secībā (zinot, vai konkrētais skaitlis ir intervāla sākums vai beigas) un tad, ejot cauri šim sakārtotajam masīvam, var uzturēt skaitītāju, kas nosaka vai šajā brīdī visi gājienu intervāli ir beigušies. Ja visi gājienu intervāli ir beigušies, tad tas nozīmē, ka šajā brīdī sākas skaitļu intervāls, kuri ir palikuši.

Tātad atlicis noskaidrot cik labo skaitļu ir dotā skaitļu intervālā $[a, b]$. Sākumā noskaidrosim, cik ir labo skaitļu intervālā $[1, c]$. Tā kā labi ir tie skaitļi, kas dalās ar M_1 vai M_2, tad labo skaitļu skaits ir $L(c) = \frac{c}{M_1} + \frac{c}{M_2} - \frac{c}{MKD(M_1,M_2)}$. To skaitļu, kas dalās gan ar M_1, gan M_2, skaits ir jāatņem, jo tas ir ieskaitīts divas reizes - abos iepriekšējos saskaitāmajos. Tad, lai noskaidrotu cik ir labo skaitļu intervālā $[a, b]$, jāaprēķina $L(b) - L(a-1)$.

Risinājuma izpildes laika sarežģītība ir $O(GlogG)$.

Latīņu kvadrāti

Ielasot dotā rūtiņu laukuma vērtības, pārkodēsim simbolus par skaitļiem. Piemēram, simbolus, kas atbilst cipariem, par vienciparu skaitļiem no 0 līdz 9, lielos burtus - par skaitļiem no 10 līdz 35, bet mazos - par skaitļiem no 36 līdz 61. Tagad katrā laukuma rūtiņā varam izveidot 62 bitus garu masīvu, kur katrā bitā glabājas informācija vai iepriekšējās secīgās n šīs rindas rūtiņās attiecīgais simbols bija pārstāvēts, vai nē. Turklāt, lai savāktu šo informāciju, katrā pozīcijā pietiek paņemt ierakstu no iepriekšējās pozīcijas un veikt divas izmaiņas - pievienot informāciju par jauno bitu un nodzēst par to, kas tagad "izbrauc no redzamības apgabala".

Tagad varam caurskatīt rindas un katrā pozīcijā atzīmēt vērtību A_{ij} - cik ir tādas rindas pēc kārtas, kuru secīgo n rūtiņu segmenta, kas beidzas rūtiņā $(i; j)$, saturs (vērtību komplekts) ir vienāds.

Tādā pat veidā varam aprēķināt šos lielumus, veicot laukuma caurskati pa kolonnām. Finālā iegūsim vērtības B_{ij} - cik ir tādas kolonnas pēc kārtas, kuru secīgo n rūtiņu segmenta, kas beidzas rūtiņā $(i; j)$, saturs (vērtību komplekts) ir vienāds.

Beigās atliek katrai rūtiņai $(i; j)$ izanalizēt saistībā ar to uzkrāto informāciju. Ja bitu masīvu ieraksts abos virzienos sakrīt, ja katrā no virzieniem ir vismaz n vienādi ieraksti un atzīmēto bitu skaits ir tieši n, tad šādā rūtiņā beidzas derīgs latīņu kvadrāts.

Atrisinājuma izpildes laika sarežģītība ir $O(mk)$.

Lakatiņa summa

Reducēšana. Sākumā reducēsim uzdevumu uz vieglāku. Pieņemsim, ka mums ir funkcija $F(T)$, kas dotam nenegatīvam skaitlim T izrēķina lielāko skaitli, kura lakatiņa summa nepārsniedz T (ja $T = 0$, tad $F(T) = 0$).

Tad atradīsim $X = F(S_{\text{maz}} - 1)$ un $Y = F(S_{\text{liel}})$:

- Ja $X = Y$, tad lielākais skaitlis, kura lakatiņa summa nepārsniedz S_{liel}, ir stingri mazāks par S_{maz}, un atbilde ir $(0,\ 0)$, jo nav neviena skaitļa, kura lakatiņa summa atrodas intervālā $[S_{\text{maz}}, S_{\text{liel}}]$.
- Pretējā gadījumā atbilde ir $(X + 1, Y)$, jo $X + 1$ ir pirmais skaitlis, kura lakatiņa summa ir vismaz S_{maz}.

Risinājums. Nepieciešams implementēt funkciju F. Aplūkosim vēlreiz skaitļa lakatiņa summas definīciju

$$S(A) = \overline{a_n a_{n-1} \dots a_2 a_1} + \overline{a_n a_{n-1} \dots a_2} + \overline{a_n a_{n-1} \dots a_3} + \dots + \overline{a_n a_{n-1}} + a_n$$

Ievērosim, ka varam to pārrakstīt citā formā:

$$S(A) = \overline{\underbrace{a_n a_n \dots a_n a_n}_{n \text{ cipari}}} + \overline{\underbrace{a_{n-1} \dots a_{n-1}}_{n-1 \text{ cipars}}} + \overline{\underbrace{a_{n-2} a_{n-2} \dots a_{n-2}}_{n-2 \text{ cipari}}} + \dots + \overline{a_2 a_2} + a_1$$

Tāpēc, ja gribam izrēķināt $F(T)$, tad varam rīkoties pēc alkatīga principa. Uzbūvēsim tādu A, ka $F(T) = S(A)$. Apzīmēsim $A = \overline{a_n a_{n-1} \dots a_2 a_1}$. Lai A būtu pēc iespējas lielāks, pirmkārt, ciparu skaitam n jābūt pēc iespējas lielākam. Savukārt, ja zināms n, tad vērtībai a_n jābūt pēc iespējas lielākai. Līdz ar to varam pārlasīt n (no 0 līdz 17, jo S_{liel} nav lielāks par $\underbrace{11 \dots 1}_{17 \text{ cipari}}$) un a_n (no 1 līdz 9), un izvēlamies tādus, ka

$$a_n \cdot \underbrace{11 \dots 1}_{n \text{ cipari}} \le T$$

un n, a_n ir pēc iespējas lielāki. Tad atkārtojam šo procesu skaitlim $T - a_n \cdot \underbrace{11 \dots 1}_{n \text{ cipari}}$, tādā veidā atrodot visus skaitļa A ciparus.

Piemēram, izrēķināsim $F(5436)$. Lielākais skaitlis, kas sastāv no visiem vienādiem cipariem un nepārsniedz 5436, ir 4444. Tad paliek skaitlis $5436 - 4444 = 992$. Lielākais skaitlis no vienādiem cipariem, kas nepārsniedz 992, ir 888. Paliek skaitlis $992 - 888 = 104$. Nākamais skaitlis no vienādiem cipariem tad ir 99. Pēdējais skaitlis ir $104 - 99 = 5$. Līdz ar to $F(5436) = 4895$.

Ātrdarbība. Tiek divreiz izsaukta funkcija F, kuras izpildes laika ātrdarbība atkarībā no realizācijas ir vai nu $O(\log(S_{\text{liel}})^2)$, vai arī $O(\log(S_{\text{liel}}))$.

Neder kā summa

Aplūkosim uzdevuma "Lakatiņa summa" risinājumu un tajā definēto funkciju $F(T)$ – lielāko skaitli, kura lakatiņa summa nepārsniedz T.

Atkal izrēķināsim divus skaitļus $X = F(S_{\text{maz}} - 1)$ un $Y = F(S_{\text{liel}})$. Visi skaitļi, kuru lakatiņa summa atrodas starp S_{maz} un S_{liel}, ieskaitot, ir visi skaitļi intervālā $(X, Y]$, jo var pamanīt, ka skaitļa lakatiņa summa pieaug, ja pieaug skaitlis. Līdz ar to atbilde uz uzdevumu ir skaitļu skaits intervālā $[S_{\text{maz}}, S_{\text{liel}}]$, no kura ir atņemts skaitļu skaits intervālā $(X, Y]$. Tātad atbilde uz uzdevumu ir

$$(S_{\text{liel}} - S_{\text{maz}} + 1) - (Y - X).$$

Ātrdarbība. Līdzīgi kā uzdevumā "Lakatiņa summa", izpildes laika ātrdarbība ir vai nu $O(\log(S_{\text{liel}})^2)$, vai arī $O(\log(S_{\text{liel}}))$.

"Pēdējās formulas" autosacīkstes

Šis uzdevums ir visai tehnisks - nepieciešams precīzi realizēt uzdevuma tekstā aprakstīto. Pēc kārtas pa rindām ir jāievada dati par kārtējo sportistu, sacensību rezultāti (iegūtā vieta vai informācija par nepiedalīšanos/nefinišēšanu) korekti

jāpārvērš punktos un, ja punktu skaits šobrīd ir lielākais, informācija par šo sportistu un iegūtajiem punktiem jāsaglabā.

Saprotams, ka atsevišķā mainīgā jāglabā šobrīd lielākais kāda sportista iegūtais punktu skaits P_{maks}.

Uzdevumu mazliet sarežģī nepieciešamība izvadīt **visu** maksimālo punktu skaitu ieguvušo sportistu numurus. Ja viena sportista gadījumā tas būtu vienkāršs veselu skaitļu tipa mainīgais, tad tagad numuru saglabāšanai nepieciešams izmantot kādu datu struktūru - piemēram, masīvu.

Tad, atkarībā no aprēķinātā punktu skaita *P* vai nu šim masīvam jāpievieno sportista numurs (ja $P = P_{maks}$), vai arī masīvs jānotīra, tam jāpievieno sportista numurs un P kļūst par jauno P_{maks}.

Ja $P < P_{maks}$, tad nekas nav jādara - šis sportists nevar būt starp labākajiem kopvērtējumā.

Pēc visu sportistu datu apstrādes nepieciešams izvadīt P_{maks}, masīva elementu (sportistu numuru) skaitu un pēc tam arī pašus sportistu numurus.

Torņi

Uzdevuma būtība: Ir V vaicājumi, kur katrā var būt viena no darbībām:

- Uzbūvē torni augstumā h,
- Nojauc vienu no jau uzbūvētiem torņiem augstumā h,
- Noskaidro, kāds ir k-tā augstākā torņa augstums.

Risinājums:

Binārā meklēšana segmentu kokā vai *treap* datu struktūrā.

Apskatīsim risinājumu, izmantojot segmentu koku.

Lai nebūtu jāraksta dinamisks segmentu koks un samazinātu izpildes laika sarežģītību binārajā meklēšanā, vispirms veiksim torņu augstumu vērtību kompresiju:

- No ievaddatiem nolasām visus dažādos torņu augstumus, kas parādās vaicājumos, tos sakārtojam nedilstošā secībā izslēdzot atkārtojumus (nav svarīgi, cik torņi ir ar vienādu augstumu). Tad turpmāk torni ar augstumu h uzdevumā aizvietojam ar tā pozīciju šajā sakārtotajā masīvā. Jauno torņa augstumu var efektīvi atrast izmantojot bināro meklēšanu vai iebūvētās datu struktūras, kā `std::map`. Pārveidojot augstumus uz intervālu $[1; V]$, tiek saglabātas savstarpējās īpašības par to, kuri torņi ir lielāki, mazāki par citiem, padarot torņu augstumu vērtības mazākas un daudz vieglāk apstrādājamas.

Lai atrisinātu uzdevumu:

1. Uzkonstruē segmentu koku intervālā $[1; V]$, kas glabās uzbūvēto torņu skaitu.
2. Apstrādā pieprasījumu:
   - Uzbūvē torni augstumā h:
     - Segmenta kokā pozīcijā h pieskaita 1. Izpildes laika sarežģītība: $O(logV)$
   - Nojauc torni augstumā h:
     - Segmenta kokā pozīcijā h atņem 1. Izpildes laika sarežģītība: $O(logV)$
   - Atrod k-to šobrīd augstāko uzbūvēto torni:
     - Ja intervālā $[1; V]$ ubūvēto torņu skaits ir mazāks par k, tad atbilde neeksistē.
     - Citādi veic bināro meklēšanu uz uzbūvēto torņu skaitu:

– Risinājums ar sarežģītību $O(\log^2 V)$
  - Binārās meklēšanas iterācijā pārbaudot vērtību h no segmentu koka noskaidro uzbūvēto torņu skaitu intervālā $[1; h]$ un izdara secinājumus.

Var pamanīt, ka var izmantot segmentu koka uzbūves binārās īpašības, lai optimizētu bināro meklēšanu.

– Risinājums ar sarežģītību $O(\log V)$

Apstaigā segmentu koku sākot ar saknes intervālu $[1; V]$:

- Ja intervāla izmērs ir 1 (tam var piederēt torņi ar augstumu h', tad vaicājuma atbilde atrasta, tā ir h').
- Citādi apskata intervāla "bērnu" intervālus:
- Ja no "bērniem" intervālā pa labi uzbūvēto torņu skaits cnt_R ir lielāks, vienāds par meklēto skaitu k, tad rekursīvi apstaigā labējo intervālu.
- Citādi rekursīvi apstaigā intervālu pa kreisi, vērtību k aizstājot ar $k - cnt_R$, jo tika noskaidrots, ka cnt_R torņi ir augstāki, un tie vairs netiks apskatīti intervālā pa kreisi.

Izpildes laika sarežģītība:

Koordinašu kompresija: $O(V \times \log V)$

Segmentu koka uzkonstruēšana: $O(V \times \log V)$

Katrs pieprasījums: $O(\log V)$

Visu pieprasījumu apstrāde: $O(V \times \log V)$

Kopējā risinājuma izpildes laika sarežģītība: $O(V \times \log V)$