Gudrie sienāži

Sienāzis 0. Sākumā ievērosim, ka sienāzis ar numuru 0 paliks uz savas vietas. Līdz ar to, lai sakārtotos, visiem sienāžiem, kas ir pa kreisi no x_0, ir noteikti jāpalecās pa labi tā, lai nonāktu pa labi no x_0. Izdarīsim to un tālāk pieņemsim, ka $x_0 = 0$. (Atbildei beigās pieskaitīsim tikko izdarīto kopīgo lēcienu skaitu.)

Lēcienu skaits. Ņemsim vērā, ka katram sienāzim ir izdevīgi kustēties vienmēr tikai vienā virzienā, ceļā uz savu galapunktu (nemainīt lēcienu virzienu).

Parādīsim, ka nevienam sienāzim nekad nav jēgas lekt vairāk par n reizēm. Aplūkosim jebkuru korektu sienāžu izkārtojumu $p_1, p_2, \dots, p_n$. Paņemsim šo pašu izkārtojumu p, bet 1. sienāzi ieliksim koordinātē n, ja $p_1 > n$. Šis arī ir korekts izkārtojums, jo 1. sienāzis ir pa kreisi no visiem pārējiem (izņemot 0.), un mēs tikai pabīdījām to vēl pa kreisi. Tā kā katra sienāža sākumkoordināte nav lielāka par n, tad 1. sienāzim jebkurā gadījumā vajag mazāk par n soļiem, lai tajā aizlektu. Kopējais lēcienu skaits nepalielinājās, jo ja $p_1 > n$, tad 1. sienāzim jau vajadzēja $p_1 - n > 0$ soļus, lai izlektu ārā no intervāla $[1, n]$. Nosauksim šo izkārtojumu par $p(1)$.

Tagad aplūkosim 2. sienāzi. Ja $p_2 > n + 2$, tad aplūkosim izkārtojumu $p(2)$, kurš atšķiras no $p(1)$ tikai ar to, ka p_2 ir intervālā $[n, n + 2]$ (tā kā 2. sienāža lēciena garums ir 2, tad viņa pozīcija šajā intervālā ir viennozīmīga). Ja $p_2 > n + 2$, tad lai $p(2) = p(1)$. Citiem vārdiem, tagad 2. sienāzis atrodas kaut kur intervālā $[1, n + 2]$. Tā kā 2. sienāža lēciena garums ir 2, viņam vajag ne vairāk kā $\frac{n+2}{2} \leq n$ lēcienus, lai pārvietotos jebkurā pozīcijā šajā intervālā. Atkal, kopējais lēcienu skaits nepalielinājās, jo ja $p_2 \geq n + 3$, tad 2. sienāzis pielietoja papildus soļus, lai izlektu no intervālā $[1, n + 2]$, kurā viņš sākumā atrodas.

Šadu procesu atkārtojam, i-to sienāzi "iebīdot" intervālā $[1, (n-1) + 1 + 2 + 3 + \cdots + i] = \left[1, n - 1 + \frac{i(i+1)}{2}\right]$, iegūstot izkārtojumu $p(i)$, un nepalielinot kopējo lēcienu skaitu. Lai i-tajam sienāzim nokļūtu jebkurā vietā šajā intervālā no sākumpozīcijas, viņam vajag ne vairāk kā $\frac{n-1+\frac{i(i+1)}{2}}{i} = \frac{n-1}{i} + \frac{i+1}{2} \leq n$ lēcienus (pierādīt šo nevienādību atstāsim kā vingrinājumu :)). Tātad, lai nonāktu līdz korektam izkārtojumam $p(n)$, kopā vajag veikt ne vairāk lēcienus, kā lai nokļūtu uz p, un katram sienāzim vajag patērēt ne vairāk kā n lēcienus.

Dināmiskā programmēšana. Izmantosim pēdējo novērojumu, lai izdomātu algoritmu, kas risina uzdevumu. Rēķināsim tabulu $dp[i][k]$ – kāds ir mazākais kopējais lēcienu skaits, lai pirmie i sienāži būtu korekti izvietoti savā starpā, un i-tais sienāzis būtu novietots koordinātē $d(i,k) = (x_i \bmod i) + ik$. Tad atbilde būs minimums no visām vērtībām $dp[n-1][k]$.

Skaidrs, ka katram k izpildās $dp[1][k] = |x_1 - k|$ (izņemot $k = 0$, jo pozīcijā 0 jau atrodas 0. sienāzis). Parādīsim, kā izrēķināt vērtību $dp[i][k]$, ja visas vērtības priekš $dp[i-1][k]$ jau ir izrēķinātas. Tad $dp[i][k]$ ir vienāds ar sekojošo divu vērtību summu:

- $\frac{|x_i - d(i,k)|}{i}$, jo no x_i vajag aizlekt uz $d(i,k)$;
- minimums no $dp[i-1][j]$ visiem j, kuriem $d(i-1,j) < d(i,k)$, jo korektajā izkārtojumā visi sienāži ar numuriem no 1 līdz $i-1$ ir izvietoti korekti, papildus $(i-1)$-jam sienāzim ir jābūt pa kreisi no i-tā.

Ātrdarbība. Cik ātri mēs varam izrēķināt vienu vērtību $dp[i][k]$ ar šādu metodi? Mums vajag atrast minimumu starp $dp[i-1][j]$, kuru var būt līdz pat n. Savukārt jāsarēķinā ir aptuveni n^2 vērtības (i ir no 1 līdz $n-1$, un k ir no 0 līdz n). Tas nozīmē, ka kopā japatērē $O(n^3)$ laiks, lai izrēķinātu šo tabulu.

Lai pilnībā atrisinātu uzdevumu, paātrināsim šo risinājumu līdz kvadrātiskam darbības laikam. Kad esam sarēķinājuši $dp[i][k]$, pieglābāsim to indeksu j, kuram $d(i-1,j) < d(i,k)$ un $dp[i-1][j]$ ir minimāls. Kad rēķinam $dp[i][k+1]$, paņemsim minimumu no $dp[i-1][j]$ un visiem j', kuriem $d(i-1,j) < d(i-1,j')$ un $d(i-1,j') < d(i,k+1)$. Šajā gadījumā j' tiek pārlasīts sākot jau no j, nevis no 0. Tādējādi, priekš visu $dp[i][k]$ rēķināšanas mums jāaplūko katru $dp[i-1][j]$ tikai vienu reizi, un sarežgītība kļūst $O(n^2)$. Šo tehniku sauc par *divu norāžu* (*two pointers*) metodi.

Ugunsgrēks

Laukumu aizdegšanās. Sākumā sarēķināsim $u[x][y]$ – laiku, kad pirmo reizi aizdegsies laukums ar koordinātēm (x,y). Šīs vērtības var sarēķināt laikā $O(nm)$ ar *meklēšanu plašumā* (*breadth first search*), sākot staigāt no rūtiņām, kurās sākumā jau deg uguns.

Bēgšanas laiki. Ievērosim, ka ceļš no jebkura laukuma uz kādu patversmi (laukums, kas nevar aizdegties) nevar būt garāks par nm. Katram laikam t no nm līdz 0 sarēķināsim sarakstu ar tādiem laukumiem, ka sākot ceļu laikā t, no laukuma var aizbēgt uz kādu patversmi, bet nevar, sākot laikā $t+1$. Nosauksim šādu sarakstu par $s[t]$. Tad atbilde uz uzdevumu būs tādu laukumu skaits, kurā sākumā ir cilvēks, un tas piedalās kādā no sarakstiem $s[t]$ kādam t. (Ja laukums nepiedalās nevienā no sarakstiem, tad no tā nav iespējams nekad aizbēgt uz patversmi). Glabāsim būla masīvu $v[x][y]$, kas apzīmē, vai laukums (x,y) jau tika ielikts kādā sarakstā (sākumā aizpildīts ar nullēm).

Pašā sākumā sarakstā $s[nm]$ ieliksim visus laukumus, kuros ir patvērsme (attiecīgi uzstādam atbilstošos $v[x][y]$ par 1). Tagad pārlasīsim laiku t no nm līdz 1. Fiksētam t pārlasam laukumus (x,y) no saraksta $s[t]$. Pārlasam visus četrus (x,y) kaimiņus. Pieņemsim, ka aplūkojam vienu no tā kaimiņiem (x',y') (kurā nav uguns vai klintis), un ka $v[x'][y']$ ir 0 (šis laukums vēl nav nevienā no sarakstiem). Tad, tā kā t ir pēdējais laiks, kad bēgt no (x,y), no laukuma (x',y') vajag bēgt ne vēlāk kā laikā $t-1$ (jo citādi šis laukums būtu jau ievietots kādā no sarakstiem iepriekš). No citas puses, no laukuma (x',y') vajag arī bēgt pirms tas aizdegas, t.i., ne vēlāk kā laikā $u[x'][y']-1$. Līdz ar to ieliekam laukumu (x',y') sarakstā $s[\min(t-1, u[x][y]-1)]$, un uzstādam $v[x'][y']$ uz 1.

Tādā veidā mēs katram laukumam uzzinam pēdējo brīdi, kad no tā bēgt, lai izglabtos. Tas arī pasaka, no kuriem laukumiem ir iespējams izglābties. Tā kā katrs laukums tiek ielikts sarakstā maksimums vienu reizi, kopā visi saraksti $s[t]$ aizņem $O(nm)$ atmiņas. Sarakstus $s[t]$ ērti implementēt ar dināmiskajiem masīviem (piemēram, vector C++). Algoritms arī strādā laikā $O(nm)$.

Īpašie gadījumi. Augstāk aprakstītajā risinājumā netiek apstrādāts interesants gadījums, kad cilvēks ir pilnībā norobežots ar klintīm gan no uguns, gan no patversmēm. Arī šajā gadījumā cilvēks izglābjas! Šādas vietas ir jāatrod atsevišķi – var palaist divas meklēšanas plašumā no degpunktiem un no patversmēm; tad atbildei ir papildus jāpieskaita tie cilvēki, kuri nav sasniedzami no abu šādu veidu laukumiem.