Uzdevuma "Atlaides" atrisinājums

Šajā uzdevumā ir liels kārdinājums izmantot reālu skaitļu mainīgos. Taču padomāsim, vai tie patiešām ir nepieciešami. Visi ievaddatos dotie skaitļi ir naturāli – preču cenas ir dotas veselos latos un arī atlaide tiek piemērota veselos procentos. Kāda var būt cena precei, kurai piemērota atlaide? Jaunā cena ir c_i*(100-P)/100, t.i., tā ir izsakāma veselos santīmos. Tāpēc rēķināsim atbildi santīmos, kas vienmēr būs vesels skaitlis. Kad atbilde santīmos ir aprēķināta (apzīmēsim to ar A), tad izvaddatu failā jāizvada veselo latu skaits, tad komats, un tad santīmu skaits. Ja santīmu skaits ir mazāks par 10, tad nedrīkst aizmirst par nulli, kas jāizvada uzreiz aiz komata. Piemēram, to var izdarīt šādi:

- Valodā Pascal:
```
WriteLn(fout, A div 100, ',', (A div 10) mod 10, A mod 10);
```
- Valodā C vai C++:
```
printf("%I64d,%02d\n", A / 100, A % 100);
```
- Valodā C++:
```
fout << A / 100 << "," << (A / 10) % 10 << A % 10;
```

Neizmantojot reālus skaitļus, mēs izvairīsimies no iespējamām precizitātes kļūdām un problēmām ar to izvadīšanu (ņemot vērā, ka lati un santīmi jāatdala ar komatu un jāizvada divi cipari aiz komata). Nerakstīts likums ir izvairīties no reālu skaitļu lietošanas, ja bez tiem var iztikt.

Tālāk dots risinājuma pseidokods:

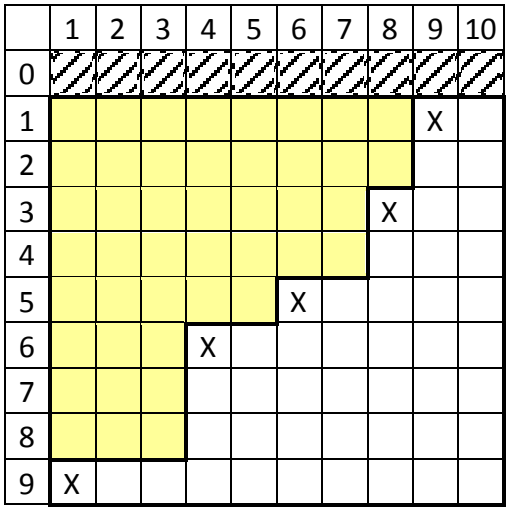
```
ielasa N, C, S, P
A ← 0

for i ← 1 to N do
    ielasa cena, skaits
    if (cena >= C) or (skaits >= S) then
        jaunā_cena ← cena * (100-P)  // cena santīmos
    else
        jaunā_cena ← cena * 100      // cena santīmos
    A ← A + jaunā_cena * skaits

izvada A
```

Kopējā cena latos var sasniegt 10^5 (lielākais iespējamais N) * 10^5 (lielākā iespējamā cena) * 10^5 (lielākais iespējamais preču skaits), t.i., 10^{15} latu jeb 10^{17} santīmu, tāpēc nepieciešams lietot 64 bitu mainīgos. Par tiem sīkāk var izlasīt šeit: http://lio.e-spiets.lv/64biti/

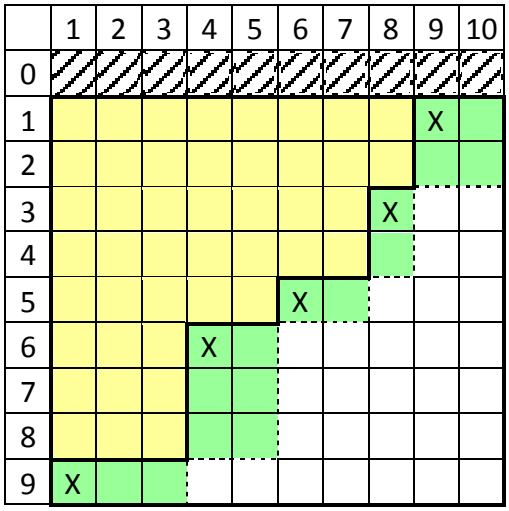
Uzdevuma “Pīrāgs” atrisinājums

Vispirms ievērosim, ka saskaņā ar uzdevuma nosacījumiem doto garumu virkne ir neaugoša, t.i., katrs nākamais ievaddatu skaitlis ir mazāks vai vienāds ar iepriekšējo. Aplūkosim šādu ievaddatu piemēru (skat. 1. zīm.):

9 10 8
8
8
7
7
5
3
3
3



1. zīmējums



2. zīmējums

Var redzēt, ka, lai iegūtu sagriezto pīrāgu, katrā no ar “X” atzīmētajām rūtiņām (sauksim tās par X-rūtiņām) noteikti jāizdara grieziens, turklāt pietiek veikt griezienus tikai šajās rūtiņās. Tātad, lai atbildētu uz pirmo jautājumu, ir jāatrod X-rūtiņu skaits. X-rūtiņas ir tajās rindās, kuru garums atšķiras no iepriekšējās rindas garuma. Lai varētu izmantot šo nosacījumu arī pirmajai rindai, varam iztēloties, ka pīrāgam ir rinda ar numuru 0 un garumu K, kā parādīts 1. zīmējumā.

Tā kā pēdējais grieziens tiek veikts kādā no X-rūtiņām, tad visās pārējās X-rūtiņās griezienam jau jābūt izdarītam pirms pēdējā grieziena. Līdz ar to pēdējā griezienā tiks izgriezts kāds no taisnstūriem, kas iekrāsoti zaļajā krāsā (skat. 2. zīm.). Katra šāda taisnstūra garums ir vienāds ar divu secīgu garumu (skaitļu, kas doti ievaddatos) starpību, bet augstums – ar skaitu, cik reižu viena un tā pati garuma vērtība atkārtojas ievaddatos. Jāizvēlas taisnstūris ar vislielāko laukumu.

Lai nebūtu jāveic speciāla apstrāde pēdējam taisnstūrim (tam, kurš atrodas viszemāk) un X-rūtiņai tā iekšpusē, varam uzskatīt, ka rindu ar numuriem no N+1 līdz R garums ir 0, kā arī ieviest rindu ar numuru R+1 un garumu -1. Šādā gadījumā risinājuma pseidokods būs īsāks:

```
griezieni ← 0
iepriekšējā_starpība ← 0
augstums ← 1
lielākais_laukums ← 0

ielasa R, K, N
iepriekšējais_garums ← K
for i ← 1 to R+1 do
    if (i <= N) then
        ielasa garums
    else if (i <= R) then
        garums ← 0
    else
        garums ← -1

    if (garums < iepriekšējais_garums) then
        if (garums >= 0) then
            griezieni ← griezieni + 1

        laukums ← iepriekšējā_starpība * augstums
        if (laukums > lielākais_laukums) then
            lielākais_laukums ← laukums
        augstums ← 1
        iepriekšējā_starpība ← iepriekšējais_garums – garums
    else
        // šajā gadījumā garums = iepriekšējais_garums
        augstums ← augstums + 1

izvada griezieni
izvada lielākais_laukums
```

Uzdevuma “Spēle” atrisinājums

Šajā uzdevumā ar **[a-b]** apzīmēsim risinājuma pseidokoda rindiņu intervālu, kurā tiek veikti tekstā minētie aprēķini. Pseidokods dots atrisinājuma beigās.

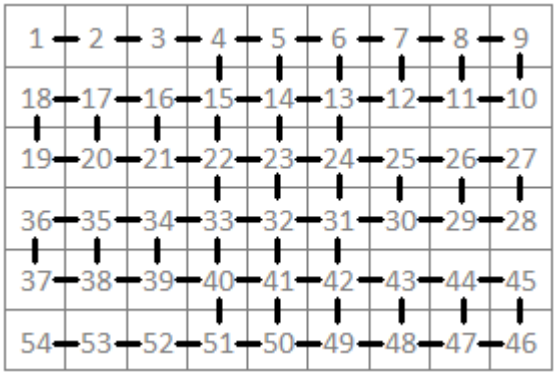
Vispirms uzskatīsim, ka S ≤ B (ja tā nav, tad apmainām šo mainīgo vērtības vietām **[2-6]**). Pirmais darbs ir noskaidrot, kurā rindā un kurā kolonnā atrodas rūtiņas S un B **[7-12]**. Ja abas rūtiņas atrodas vienā rindā, tad atbilde ir S un B vērtību starpība **[15-16]**. Citādi beigu rūtiņas rindas numurs ir lielāks par sākuma rūtiņas rindas numuru. Turpmāk analizēsim šo gadījumu.

Ja ir iespējams veikt gājienu no kādas rūtiņas uz tās blakusrūtiņu, tad savienosim šīs rūtiņas ar līniju. Tādā veidā iegūsim ceļu tīklu, kas parāda, kā iespējams pārvietoties pa rūtiņām. Piemēru ar N=6, M=9, K=12 skat. 3. zīmējumā, bet ar K=5 – 4. zīmējumā. Ar D apzīmēsim vertikālo līniju skaitu starp divām secīgām rindām. No uzdevuma nosacījumiem seko, ka starp katrām divām secīgām rindām vertikālo līniju skaits būs vienāds – tas būs D=(K+1) div 2 **[13]**, kur „div” apzīmē veselu skaitļu dalīšanu bez atlikuma (dalījuma veselo daļu).

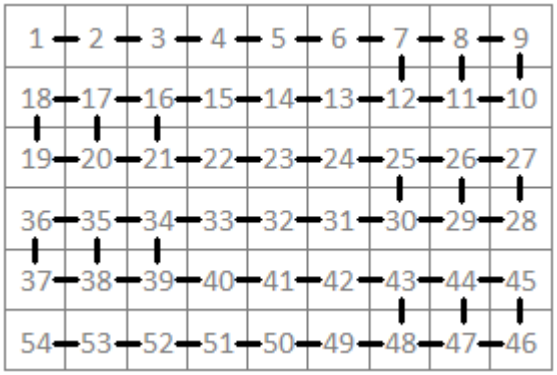
Aplūkosim divus gadījumus: 1) 2D > M **[17-24]** un 2) 2D ≤ M **[25-31]**.

Ja 2D > M (3. zīm.), tad tas nozīmē, ka būs viena vai vairākas kolonnas, kurās ir vertikālas līnijas starp katrām blakusrūtiņām (3. zīmējumā tās ir kolonnas ar numuriem no 4 līdz 6). Šo kolonnu numuri ir no M+1-D līdz D. Ja sākuma rūtiņas kolonnas numurs nav šajā intervālā, tad pirmajos gājienos kauliņš jāpārvieto horizontāli līdz šim intervālam (t.i., līdz kolonnas numurs kļūst vienāds ar M+1-D vai D). Tad jāiet uz leju līdz rindai, kurā atrodas beigu rūtiņa un tad – horizontāli līdz beigu rūtiņai **[21-23]**. Vienīgais izņēmuma gadījums, kad šāds maršruts nebūs visīsākais, ir tad, ja sākuma un beigu rūtiņas atrodas secīgās rindās un īsākajā ceļā nav nepieciešams vispirms tikt līdz kolonnai ar numuru M+1-D vai D (piemēram, ja sākuma rūtiņa ir 17 un beigu rūtiņa ir 21, sk. 3.zīm.). Šādā gadījumā vienkārši jāizvada sākuma un beigu rūtiņu x-koordinātu un y-koordinātu starpību summa **[18-19]**.

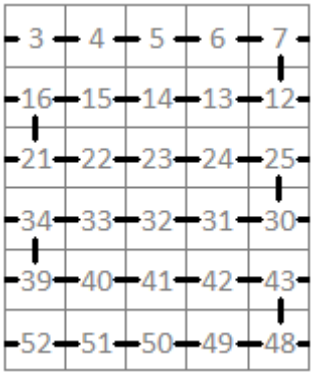
Ja 2D ≤ M (4. zīm.), tad varam ievērot, ka pārvietošanās pamatā notiek pa „čūsku", kuru veido kolonnas no D līdz M+1-D (zīmējumā – no 3 līdz 7). Ja sākuma rūtiņas kolonnas numurs nav šajā intervālā, tad sākuma rūtiņu varam horizontāli pārvietot līdz šim intervālam (t.i., līdz kolonnas numurs kļūst vienāds ar D vai M+1-D), attiecīgi palielinot veikto gājienu skaitu **[26]**. Analoģiski rīkojamies ar beigu rūtiņu **[27]**. Rezultātā abas rūtiņas atradīsies uz „čūskas". Tad kolonnas ar numuriem no 1 līdz D-1 un no M+2-D līdz M varam izmest (skat. 5. zīm.), iegūstot tabulu ar N rindām un M+2-2D kolonnām. Šīs rūtiņas pārnumurējam tā, kā redzams 6. zīm. Tā kā mums ir zināmas gan sākuma, gan beigu rūtiņas koordinātas (rindas un kolonnas numurs), tad atliek no šīm koordinātām iegūt rūtiņu numurus (jaunajā numerācijā) **[28-29]**: nepieciešamo gājienu skaits, lai nokļūtu no sākuma rūtiņas līdz beigu rūtiņai, būs vienāds ar šo rūtiņu numuru starpību **[30]**.



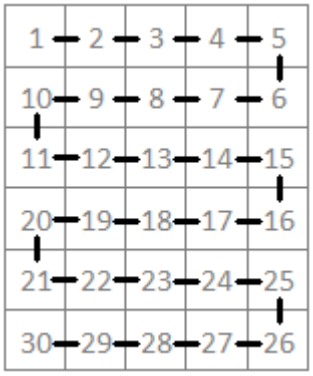
3. zīmējums



4. zīmējums



5. zīmējums



6. zīmējums

```
 1:    ielasa N, M, K, S, B
 2:    if (S > B) then
 3:        tmp ← S
 4:        S ← B
 5:        B ← tmp
 6:    end if
 7:    sy ← 1 + (S - 1) div M
 8:    sx ← 1 + (S - 1) mod M
 9:    if (sy mod 2 = 0) then sx ← M + 1 - sx
10:    by ← 1 + (B - 1) div M
11:    bx ← 1 + (B - 1) mod M
12:    if (by mod 2 = 0) then bx ← M + 1 - bx
13:    D ← (K + 1) div 2
14:    gājieni ← 0
15:    if (sy = by) then
16:        gājieni ← B - S
17:    else if (2 * D > M) then
18:        if ((by - sy = 1) and ((sx < M + 1 - D) = (sy mod 2 == 0))) then
19:            gājieni ← abs(bx - sx) + (by - sy)
20:        else
21:            pabīda(sx, M + 1 - D, D) // palīgfunkcija
22:            pabīda(bx, M + 1 - D, D) // palīgfunkcija
23:            gājieni ← gājieni + abs(bx - sx) + (by - sy)
24:        end if
25:    else // 2 * D <= M
26:        pabīda(sx, D, M + 1 - D) // palīgfunkcija
27:        pabīda(bx, D, M + 1 - D) // palīgfunkcija
28:        s_numurs ← numurs(sx, sy) // palīgfunkcija
29:        b_numurs ← numurs(bx, by) // palīgfunkcija
30:        gājieni ← gājieni + b_numurs - s_numurs
31:    end if
32:    izvada gājieni
```

```
function pabīda(x, no, līdz) // funkcija izmaina x (padotā mainīgā) vērtību
    if (x < no) then
        gājieni ← gājieni + no – x
        x ← no
    else if (x > līdz) then
        gājieni ← gājieni + x – līdz
        x ← līdz
    end if
end function

function numurs(x, y) // atgriež rūtiņas (x,y) numuru jaunajā numerācijā
    numurs ← (y–1) * (M + 2 – 2 * D) // pilno rindu skaits * čūskas
platums
    if (y mod 2 = 1) then
        numurs ← numurs + x – D + 1
    else
        numurs ← numurs + M + 2 – D – x
    return numurs
end function
```

Uzdevuma "Labākās dienas" atrisinājums

Par acīmredzamo risinājumu – pārbaudīt visus dienu intervālus un izvēlēties labāko – varēja nopelnīt 50 punktus:

```
labākā_summa ← -100000 // labākā summa jāinicializē ar vērtību, kas
ielasa N            // nepārsniedz teorētiski mazāko iespējamo atbildi
for i ← 1 to N do ielasa m[i]
for i ← 1 to N do
    summa ← 0
    for j ← i to N do
        summa ← summa + m[j]
        if (summa > labākā_summa) then labākā_summa ← summa
izvada labākā_summa
```

Šis risinājums iekļautā cikla dēļ nav pietiekoši efektīvs, lai darbotos ne vairāk par 0,2 sekundēm pie $N=10^5$.

Lai uzrakstītu labāku (efektīvāku) risinājumu, ielasīsim skaitļus pa vienam un mēģināsim noteikt, kāda ir lielākā iespējamā summa (apzīmēsim to ar max_i), ja dienu intervāls beidzas ar nupat ielasīto dienu (apzīmēsim šo dienu ar i, bet bilanci šajā dienā ar m_i). Tātad, mūs interesē lielākā no vērtībām:

$m_1+m_2+m_3+...+m_i$,

$m_2+m_3+...+m_i$,

$m_3+...+m_i$,

...

m_i .

Apzīmējot pirmo k skaitļu summu ar s_k (t.i., $s_k = m_1 + m_2 + ... + m_k$), mūs interesē lielākā no vērtībām $s_i–s_0$, $s_i–s_1$, $s_i–s_2$, ..., $s_i–s_{i–1}$ (tiek pieņemts, ka $s_0 = 0$). Citiem vārdiem, mūs interesē vismazākā no vērtībām s_0, s_1, ..., s_{i-1}. Patiesībā pietiek glabāt līdz šim vismazāko s_i vērtību (kas rēķināta no līdz šim ielasītajiem skaitļiem), ko apzīmēsim ar s_{min}.

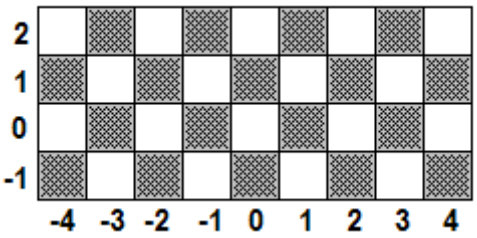
Meklētā atbilde ir lielākā no max_1, max_2, ..., max_N vērtībām. Pseidokods:

```
ielasa N
summa ← 0
s_min ← 0
labākā_summa ← -100000 // labākā summa jāinicializē ar vērtību,
                       // kas nepārsniedz teorētiski mazāko iespējamo atbildi
for i ← 1 to N do
    ielasa m_i
    summa ← summa + m_i
    max_i ← summa – s_min
    if (max_i > labākā_summa) then labākā_summa ← max_i
    if (summa < s_min) then s_min ← summa
izvada labākā_summa
```

Redzam, ka risinājumam nepieciešami vien pāris mainīgie (nav vajadzīgi masīvi). Tā kā visu skaitļu summa var būt robežās no -10^{10} līdz 10^{10}, tad arī šajā uzdevumā jālieto 64 bitu mainīgie.

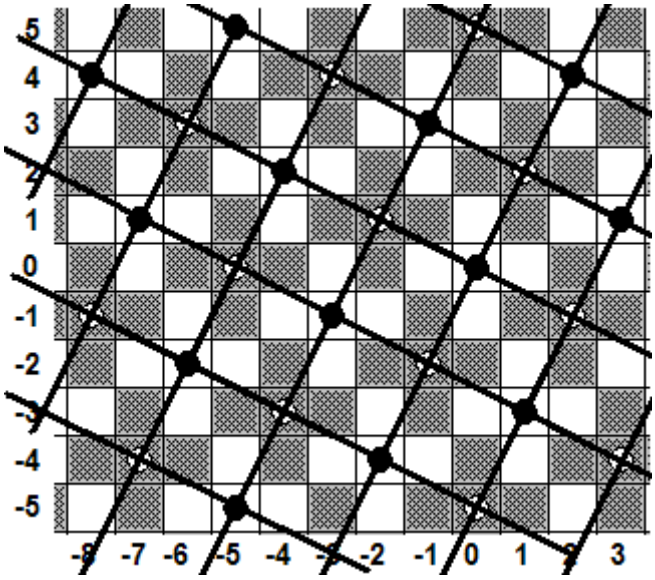
Uzdevuma “Plusiņi” atrisinājums

Vispirms aplūkosim vienkāršāku uzdevumu – atrast nepatīkamo gājienu skaitu starp divām rūtiņām, ja laukums izkrāsots šaha rūtiņu veidā:



Viegli pamanīt, ka šajā gadījumā visi gājieni ir nepatīkami, jo katrai rūtiņai blakus ir tikai pretējas krāsas rūtiņas. Šajā gadījumā nepatīkamo gājienu skaits ir vienāds ar tā saucamo Manhetenas attālumu starp šīm rūtiņām. Ja viena rūtiņa koordinātas (rinda; kolonna) ir $(r_1;k_1)$, bet otras – $(r_2;k_2)$, tad Manhetenas attālums starp tām ir $|r_1-r_2|+|k_1-k_2|$.

Kā nākamo apakšuzdevumu aplūkosim plusiņu laukumu, papildus pieņemot, ka gan sākuma, gan beigu rūtiņa atrodas kāda plusiņa centrā. Ar aplīšiem iezīmēsim šīs centra rūtiņas:



Centra rūtiņas veido regulāru rakstu, kas līdzīgs iepriekš aplūkotajām šaha laukuma rūtiņu rakstam, tikai šis jaunais režģis (iezīmēts ar taisnēm, kas iet cauri plusiņu centriem) ir pagriezts attiecībā pret vertikālo un horizontālo virzienu. Šim režģim piemīt īpašības, kas līdzīgas šaha laukuma rūtiņu režģim - katram vienas krāsas plusiņam blakus atrodas četri otras krāsas plusiņi un pagrieztajā režģī tieši šo plusiņu centri atrodas katra plusiņa centra kaimiņos. Lai pārietu no viena plusiņa uz kaimiņu plusiņu, ir nepieciešams veikt tieši vienu nepatīkamu gājienu. Ja mēs iedomātos, ka vienā gājienā drīkstam uzreiz pārvietoties starp divu kaimiņu plusiņu centriem, tad situācija būtu pilnīgi tāda pati kā iepriekš aplūkotajā uzdevumā par laukumu, kas izkrāsots šaha laukuma veidā.

Tā kā sākotnējā uzdevumā ir nepieciešams noteikt tikai nepatīkamo gājienu skaitu, tad visu uzdevumu var sadalīt trīs apakšuzdevumos:

1) atrast sākotnējās rūtiņas plusiņa centru,

2) atrast beigu rūtiņas plusiņa centru,

3) pārrēķināt koordinātas plusiņu centru režģī un aprēķināt Manhetenas attālumu starp plusiņu centriem.

Pēc dotā rūtiņas centra koordinātas ir (0; 0). Pēc zīmējuma var ievērot, ka plusiņu centri atrodas rūtiņās (2a-b, a+2b), kur a un b – veseli skaitļi.

Katrai laukuma rūtiņai ar koordinātām (r; k) iespējama tieši viena no piecām situācijām:

a) rūtiņa pati ir plusiņa centrs;

b) plusiņa centrs atrodas vienu rūtiņu augstāk - plusiņa centra koordinātas ir (r+1; k);

c) plusiņa centrs atrodas vienu rūtiņu zemāk - plusiņa centra koordinātas ir (r-1; k);

d) plusiņa centrs atrodas vienu rūtiņu pa labi - plusiņa centra koordinātas ir (r; k+1);

e) plusiņa centrs atrodas vienu rūtiņu pa kreisi - plusiņa centra koordinātas ir (r; k-1).

Tātad, lai atrastu plusiņa centru, nepieciešams ne vairāk kā piecas reizes atbildēt uz jautājumu: „Vai šī rūtiņa ir plusiņa centrs?", jeb, citiem vārdiem, „Vai iespējams atrast tādas a un b vērtības, ka rindas numurs ir 2a-b, bet kolonnas numurs ir a+2b ?". Pie kam tieši vienreiz atbilde uz šādu jautājumu būs pozitīva.

Pieņemsim, ka rūtiņas, par kuru gribam atbildēt uz jautājumu „Vai šī rūtiņa ir plusiņa centrs?", koordinātas ir (R; K). Tad nepieciešams veselos skaitļos atrisināt sistēmu:

$$\begin{cases} R = 2a - b \\ K = a + 2b \end{cases}$$

$$\begin{cases} 2K - R = 5b \\ 2R + K = 5a \end{cases}$$

$$\begin{cases} a = \dfrac{2R + K}{5} \\ b = \dfrac{2K - R}{5} \end{cases}$$

Pie kam a un b norāda plusiņa centra koordinātas „slīpajā" jeb plusiņu centru režģī – t.i., blakus atrodas plusiņi, kuriem par vienu atšķiras a vai b vērtība. Manhetenas attālumu starp plusiņu centriem aprēķina kā aprakstīts iepriekš – ja viena plusiņa centra koordinātas „slīpajā" režģī ir (a_1; b_1), bet otra – (a_2; b_2), tad attālums ir $|a_1-a_2|+|b_1-b_2|$.

Uzdevuma "Rotaļa" atrisinājums

Vispirms atrod visus skaitļa N dalītājus, ciklā no 2 līdz $\sqrt{N}$ pārbaudot, vai N dalās bez atlikuma ar šo skaitli. Ja i $(2 \le i \le \sqrt{N})$ ir N dalītājs, tad arī $\frac{N}{i}$ ir skaitļa dalītājs. Ja šādu dalītāju nav, tad vienkārši jāizvada N.

Pieņemsim, ka N dalītāji, kas lielāki par 1 un mazāki par N, ir $a_1 < a_2 < \ldots < a_k$. Uz tāfeles beigās paliks tikai pirmskaitļi, pie kam tie visi būs N dalītāji. Ja pieņem pretējo, ka skaitļu nodzēšanas/uzrakstīšanas vietā var parādīties kāds skaitlis x, kas nav N dalītājs, izsekojot nodzēšanas/uzrakstīšanas darbību virkni pretējā secībā, var parādīt, ka x ir kāda skaitļa a_i dalītājs, kas lielāks par 1 un tātad tam vajag būt arī N dalītāju virknē.

Izveidosim tabulu, kuru pakāpeniski modificējot, iegūsim beigās uz tāfeles uzrakstīto skaitļu komplektu:

Dalītājs	a_1	a_2	a_3	...	a_k
Cik reizes dalītājs beigās ir uzrakstīts uz tāfeles	c_1	c_2	c_3	...	c_k

Vispirms pieņemsim, ka visi N dalītāji arī beigās pa vienai reizei būs uzrakstīti uz tāfeles - visas c_i vērtības ir 1.

Tagad ciklā sākam apstrādāt a_i vērtības dilstošā secībā. Katram i tiek pārbaudīts, vai a_i dalās ar $a_1, a_2, \dots, a_{i-1}$. Ja a_i dalās ar a_j , tad c_j vērtība tiek palielināta par c_i. Šī pieskaitīšana nozīmē, ka katra nodzēstā a_i vietā tiks uzrakstīts a_j. Ja a_i bija kaut viens dalītājs, tad pēc šo darbību izpildes c_i tiek piešķirta vērtība 0, kas nozīmē, ka a_i no tāfeles beigās ir nodzēsts.

Kad visas a_i vērtības šādi apstrādātas, katrs c_i apzīmē, cik reizes katrs dalītājs beigās ir uzrakstīts uz tāfeles. Atliek aprēķināt visu reizinājumu $a_i c_i$ summu: $\sum_{i=1}^{k} a_i c_i$, kas arī ir meklētā atbilde.

Nodemonstrēsim algoritma darbību uzdevuma aprakstā dotajam piemēram.

N=20. Atrodam tā dalītājus un aizpildām tabulu:

Dalītājs	2	4	5	10
Cik reizes dalītājs beigās ir uzrakstīts uz tāfeles	1	1	1	1

Veic dalītāja 10 apstrādi:

Dalītājs	2	4	5	10
Cik reizes dalītājs beigās ir uzrakstīts uz tāfeles	2	1	2	0

5 ir pirmskaitlis – tabula nemainās.

Veic dalītāja 4 apstrādi:

Dalītājs	2	4	5	10
Cik reizes dalītājs beigās ir uzrakstīts uz tāfeles	3	0	2	0

2 ir pirmskaitlis – tabula nemainās.

Reizinājumu $a_i c_i$ summa:

$2 \times 3 + 4 \times 0 + 5 \times 2 + 10 \times 0 = 16.$

Algoritma ātrdarbība ir proporcionāla N dalītāju skaita k kvadrātam. Ja N nepārsniedz 10^{12}, tad k vērtība nepārsniedz 6718 (pie N=963761198400). Līdz ar to aprakstītais algoritms pārliecinoši iekļaujas testam atvēlētajā laikā.