De ideale stad

Leonardo was, zoals veel Italiaanse wetenschappers en artiesten in zijn tijd, erg geïnteresseerd in stadsplanning en -ontwerp. Zijn doel was een ideale stad: comfortabel, ruim, rationeel qua verbruik, heel anders dan de nauwe, claustrofobische Middeleeuwse steden.

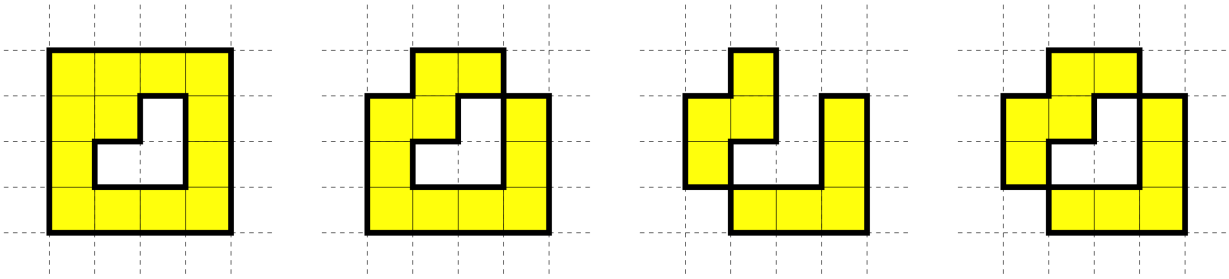
De ideale stad

De stad bestaat uit N wijken die gebouwd zijn op een oneindig rooster van cellen. Elke cel wordt aangeduid door een paar coördinaten (rij, kolom). Voor elke cel (i, j) zijn de aangrenzende cellen: (i - 1, j), (i + 1, j), (i, j - 1) en (i, j + 1). Elke wijk bedekt exact één cel op het rooster. Een wijk kan alleen in cel (i, j) geplaatst worden als en slechts als $1 \leq i,j \leq 2^{31} - 2$. We gebruiken de coördinaten van de cel om te refereren naar de wijk die erin ligt. Twee wijken zijn buren als ze in aangrenzende cellen zijn gelegen. In een ideale stad zijn alle wijken zo verbonden dat er geen "gaten" binnen de grenzen liggen; dit wil zeggen dat ze voldoen aan deze beide voorwaarden:

- Voor elk tweetal *lege* cellen bestaat er minstens één reeks van aan elkaar grenzende *lege* cellen die deze twee met elkaar verbindt.
- Voor elk tweetal *niet-lege* cellen bestaat er minstens één reeks van aan elkaar grenzende *niet-lege* cellen die deze twee met elkaar verbindt.

Voorbeeld 1

Geen van de wijk-configuraties hieronder beschrijft een ideale stad: de linkse twee voldoen niet aan de eerste voorwaarde; de derde voldoet niet aan de tweede voorwaarde, de vierde voldoet aan geen van beide voorwaarden.

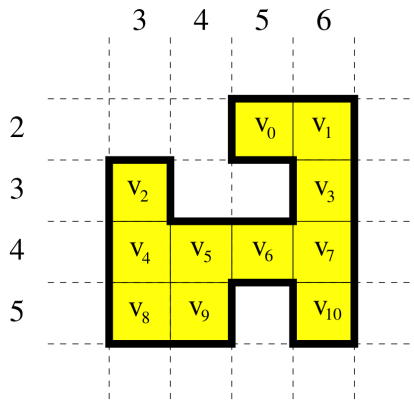


Afstand

Als je de stad wil doorkruisen kan je met een *hop* van een wijk naar een buurwijk gaan. Je kunt niet door lege cellen bewegen. Stel dat v_0, v_1, …, v_{N-1} de coördinaten zijn van N wijken op het rooster. Dan is voor elk paar van verschillende wijken op coördinaten v_i en v_j, de afstand $d(v_i, v_j)$ gedefinieerd als het kleinste aantal *hops* dat nodig is om van de ene wijk naar de andere te gaan.

Voorbeeld 2

De configuratie hieronder beschrijft een ideale stad die bestaat uit N = 11 wijken op de coordinaten v_0 = (2, 5), v_1 = (2, 6), v_2 = (3, 3), v_3 = (3, 6), v_4 = (4, 3), v_5 = (4, 4), v_6 = (4, 5), v_7 = (4, 6), v_8 = (5, 3), v_9 = (5, 4) en v_{10} = (5, 6). Dan is bijvoorbeeld: $d(v_1, v_3) = 1$, $d(v_1, v_8) = 6$, $d(v_6, v_{10}) = 2$ en $d(v_9, v_{10}) = 4$.



Opdracht

Schrijf een programma dat, gegeven een ideale stad, de som berekent van de afstanden tussen alle paren van wijken v_i en v_j waarvoor $i < j$. Formeel genoteerd, bereken de waarde van de som:

$$\sum d(v_i, v_j), \text{ waarbij } 0 \leq i < j \leq N - 1$$

Specifieker: schrijf een functie `DistanceSum`(N, X, Y) die, gegeven N en twee arrays X en Y die de stad beschrijven, de bovenstaande formule uitrekent. X en Y hebben beide lengte N. Wijk i ligt op de coördinaten (X[i], Y[i]), waarbij $0 \leq i \leq N-1$, en $1 \leq X[i],Y[i] \leq 2^{31}-2$. Omdat het resultaat te groot zou kunnen zijn om in 32 bits voorgesteld te worden, moet je het antwoord geven modulo 1 000 000 000 (één miljard).

In voorbeeld 2 zijn er 11 × 10 / 2 = 55 paren van wijken. De som van afstanden tussen alle paren van wijken is 174.

Subtaak 1 [11 punten]

Je mag ervan uitgaan dat $N \leq 200$.

Subtaak 2 [21 punten]

Je mag ervan uitgaan dat $N \leq 2\,000$.

Subtaak 3 [23 punten]

Je mag ervan uitgaan dat $N \leq 100\,000$.

Bovendien wordt aan twee bijkomende voorwaarden voldaan: voor elk paar niet-lege cellen i en j waarvoor geldt X[i] = X[j], is elke cel tussen deze cellen ook niet-leeg; en voor elk paar niet-lege cellen i en j waarvoor geldt Y[i] = Y[j], is elke cel tussen deze cellen ook niet-leeg.

Subtaak 4 [45 punten]

Je mag ervan uitgaan dat $N \leq 100\,000$.

Implementatiedetails

Je moet precies één bestand indienen, genaamd `city.c`, `city.cpp` of `city.pas`. Dit bestand moet de bovengenoemde functie implementeren volgens deze declaratie:

C/C++ programma's

```
int DistanceSum(int N, int *X, int *Y);
```

Pascal programma's

```
function DistanceSum(N : LongInt; var X, Y : array of LongInt) : LongInt;
```

Deze functie moeten werken zoals hierboven werd beschreven. Natuurlijk staat het je vrij bijkomende functies te implementeren voor intern gebruik. Je code mag op geen enkele manier interageren met standaard input/output, of met andere bestanden.

Voorbeeld-grader

De voorbeeld-grader die je krijgt in de testomgeving verwacht invoer in het volgende formaat:

- regel 1: N;
- regels 2, …, N + 1: X[i], Y[i].

Tijds- en geheugenlimieten

- Tijdslimiet: 1 seconde.
- Geheugenlimiet: 256 MiB.