Cidade ideal

Leonardo, como muitos outros cientistas e artistas italianos de sua época, era extremamente interessado em planejamento de cidades e design urbano. Ele tinha como objetivo modelar uma cidade ideal: confortável, espaçosa e racional no uso de seus recursos, muito diferente das cidades estreitas e claustrofóbicas da idade média.

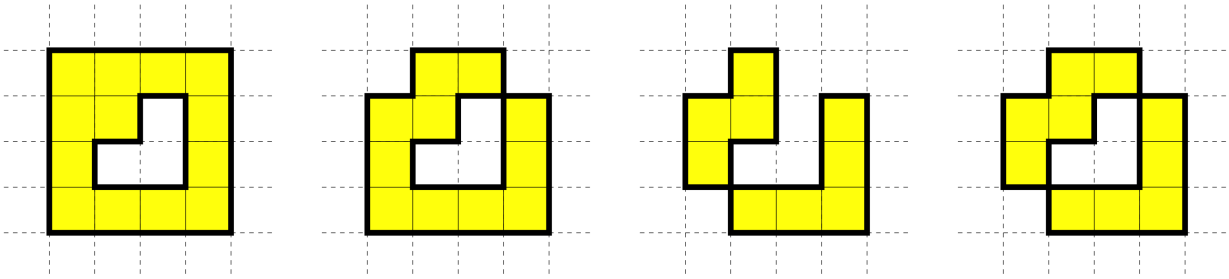
A cidade ideal

Uma cidade é feita de N blocos dispostos em um quadriculado com infinitas células. Cada célula é identificada por um par de coordenadas (linha, coluna). Dada uma célula (i, j), as células adjacentes são: (i - 1, j), (i + 1, j), (i, j - 1) e (i, j + 1). Cada bloco, quando colocado no quadriculado, cobre exatamente uma célula. Um bloco pode ser colocado sobre uma célula (i, j) se e somente se $1 \leq i, j \leq 2^{31} - 2$. Nós usaremos as coordenadas das células também para referenciar os blocos sobre elas. Dois blocos são adjacentes se eles forem colocados em células adjacentes. Em uma cidade ideal, todos os blocos são conectados de tal forma que não existem "buracos" dentro de suas fronteiras, isto é, as células devem satisfazer ambas as condições a seguir.

- Para quaisquer duas células *vazias*, existe ao menos uma sequência de células *vazias* adjacentes conectando-as.
- Para quaisquer duas células *não vazias*, existe ao menos uma sequência de células *não vazias* adjacentes conectando-as.

Exemplo 1

Nenhuma das configurações abaixo representam cidades ideais: as duas primeiras da esquerda não satisfazem a primeira condição, a terceira não satisfaz a segunda condição e a quarta não satisfaz nenhuma das condições.



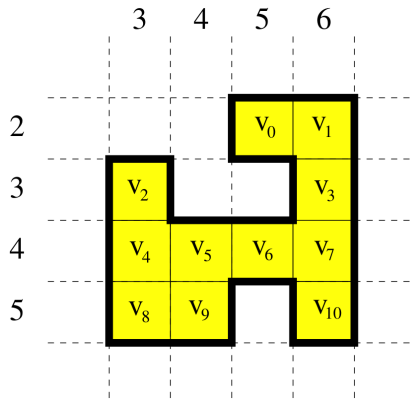
Distância

Ao atravessar uma cidade, um *passo* indica ir de um bloco para um bloco adjacente. Células vazias

não podem ser atravessadas. Sejam v_0, v_1, …, v_{N-1} as coordenadas de blocos colocados no quadriculado. Para quaisquer dois blocos com coordenadas v_i e v_j, sua distância $d(v_i, v_j)$ é o menor número de passos necessários para ir de um ao outro.

Exemplo 2

A configuração a seguir representa uma cidade ideal feita com N = 11 blocos nas coordenadas 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), e v_{10} = (5, 6). Por exemplo, $d(v_1, v_3) = 1$, $d(v_1, v_8) = 6$, $d(v_6, v_{10}) = 2$, e $d(v_9, v_{10}) = 4$.



Enunciado

Sua tarefa é, dada uma cidade ideal, escrever um programa que calcule a soma das distâncias entre todos os blocos v_i e v_j, onde $i < j$. Formalmente, seu programa deve computar o valor do seguinte somatório:

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

Especificamente, você tem que implementar uma rotina `DistanceSum`(N, X, Y) que, dados N e dois vetores X e Y que descrevem uma cidade, calcula a fórmula acima. Ambos, X e Y possuem tamanho N; o bloco i está na coordenada (X[i], Y[i]) para $0 \le i \le N - 1$, e $1 \le X[i], Y[i] \le 2^{31} - 2$. Como o resultado pode ser muito grande para ser representado usando 32 bits, você deve devolver o valor módulo 1 000 000 000 (um bilhão).

No exemplo 2, existem 11 × 10 / 2 = 55 pares de blocos. A soma das distâncias entre todos os pares de blocos é 174.

Subtarefa 1 [11 pontos]

Você pode supor que $N \le 200$.

Subtarefa 2 [21 pontos]

Você pode supor que $N \le 2\,000$.

Subtarefa 3 [23 pontos]

Você pode supor que $N \leq 100\,000$.

Adicionalmente, as seguintes condições ocorrem: dadas quaisquer duas células não vazias i e j tais que X[i] = X[j], toda célula entre elas é não vazia também; dadas duas células não vazias i e j tais que Y[i] = Y[j], toda célula entre elas é não vazia também.

Subtarefa 4 [45 pontos]

Você pode supor que $N \leq 100\,000$.

Detalhes de implementação

Você deve submeter exatamente um arquivo chamado `city.c`, `city.cpp` ou `city.pas`. Este arquivo deve implementar a função descrita acima usando a seguinte assinatura.

Programas em C/C++

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

Programas em Pascal

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

Essa função deve comportar-se como descrito acima. Claro que você é livre para implementar outras funções para seu uso interno. Sua submissão não deve interagir de qualquer modo com a entrada e saida padrão, nem com qualquer outro arquivo.

Avaliador fornecido

O avaliador fornecido com a tarefa respeita o seguinte formato de entrada:

- linha 1: N;
- linhas 2, …, N + 1: X[i], Y[i].

Limite de tempo de execução e memória

- Limite de tempo de execução: 1 segundo.
- Limite de memória: 256 MiB.