

Super Marcos Bros

CIn UFPE

STATEMENT

SUBMIT

SUBMISSIONS

STATISTICS

Marcos é um aluno de computação que adora estudar sobre jogos e suas teorias. Lembrando dos bons tempos em que jogava [Super Mario Bros](#), no seu Nintendinho, ele decidiu criar uma variação do jogo.

Nessa variação, cada fase corresponde a uma grade 2D, sendo a posição de cada obejeto dada pelas suas coordenadas (X, Y) . O personagem *Super Marcos* começa a fase numa posição inicial (X_0, Y_0) e o seu objetivo é alcançar a bandeira que se encontra na posição final (X_f, Y_f) . Para tal, ele deve efetuar saltos entre diferentes plataformas espalhadas pelo cenário.

Para saltar do ponto $P_i=(X_i, Y_i)$ para o ponto $P_j=(X_j, Y_j)$, nosso herói precisa realizar um esforço, ou seja, gastar uma quantidade de energia correspondente ao quadrado da distância em linha reta entre esses pontos, isto é

$$E(P_i, P_j) = (X_j - X_i)^2 + (Y_j - Y_i)^2.$$

Um detalhe a ser observado, no entanto, é que nem sempre é possível realizar o salto entre dois pontos **A** e **B** do cenário, por exemplo, se eles estiverem excessivamente afastados. Também pode ser que, ao saltar do ponto **A** para o ponto **B**, o ponto **A** saia da tela devido à rolagem da camera, não sendo possível retornar a ele.

Por fim, alguma plataformas têm moedas que fazem com que, ao se saltar para elas, a energia seja recuperada, ao invés de gasta. Por exemplo, se o personagem começa na posição $S=(0, 0)$ e temos duas plataformas nas posições $A=(-3, 4)$ e $B=(10, 10)$. Então os saltos $S \rightarrow A \rightarrow B$ gastam **230** pontos de energia total, ao passo que saltar diretamente $S \rightarrow B$ exige **200** pontos de energia. Porém, se a plataforma **A** tiver uma moeda, então saltar de **S** para **A** faz com que o Super Marcos *ganhe* **25** pontos de energia, de forma que o percurso $S \rightarrow A \rightarrow B$ passa a ter um gasto líquido de apenas **180** pontos de energia.

Para obter a pontuação máxima, o jogador deverá atravessar a fase realizando uma combinação de saltos de menor custo energético total. Nosso objetivo é determinar esses percursos ótimos.

Observação

As moedas reaparecem automaticamente sempre que o personagem salta para fora das plataformas que as contêm.

Time Limit

1 second

Memory Limit

768 MB

Output Limit

4 MB

Input Specification

A entrada inicia com uma linha

T

correspondente ao número de casos de teste, cada um representando uma fase do jogo. Logo após, temos as especificações dos diferentes casos.

Cada caso começa com uma linha

N

correspondente ao número de pontos do cenário, numerados de 0 a **N-1**. Em seguida temos **N** linhas, cada uma com dois inteiros

X0 Y0
X1 Y1
...
Xf=X[N-1] Yf=Y[N-1]

correspondentes às coordenadas dos **N** pontos. O ponto $P_0=(X_0, Y_0)$ corresponde à posição inicial do personagem, e o ponto $P[N-1]$ corresponde à posição final da fase. Os demais pontos correspondem a plataformas de salto.

Em seguida temos uma linha de inteiros

M I[0] I[1] ... I[M-1]

onde **M** é quantidade de plataformas que possuem moedas, e $0 < I[0] < I[1] \dots < I[M-1] < N-1$ são números dessas plataformas.

A seguir, para cada ponto $j=0, 1, \dots, N-2$, temos uma linha

Dj Vj[0] Vj[1] ... Vj[Dj-1]

onde **Dj** é a quantidade de pontos alcançáveis através de um salto a partir do ponto P_j , e $V_j[0] \dots V_j[Dj-1]$ são os números desses pontos.

Ao final de cada caso temos uma linha em branco.

Output Specification

Para cada uma das fases, deve ser impressa uma linha

E I[0] I[1] ... I[Q-1]

onde

- $E \geq 0$ é o gasto energético mínimo necessário para atravessar a fase
- $I[0]=0 \ I[1] \dots I[Q-1]=N-1$ corresponde ao percurso ótimo seguido pelo Super Mario.

Importante

Caso haja mais de um percurso ótimo, deve ser impresso o caminho com menos arestas. Persistingo o empate, deve ser escolhido o menor caminho em ordem lexicográfica reversa. Por exemplo, cado haja dois percursos ótimos **0-4-2-5-7-9** e **0-3-1-5-7-9**, o segundo deve ser escolhido já que

9=9
7=7
5=5
1<2

Devido às moedas de recuperação de energia, algumas fases podem permitir que o personagem realize uma subsequência cíclica de saltos de custo energético negativo, o que o permitiria acumular energia infinitamente, não havendo portanto um percurso gasto energético mínimo. Nesses casos , o programa deve imprimir uma linha

LOOP

Sample Input #1

1 10
2 10
3 0 0
4 19 2
5 12 4
6 7 5

Sample Output #1

1 125 0 8 3 4 2 5 6 9
2 171 0 7 1 3 8 9
3 120 0 5 3 1 7 6 8 9
4 175 0 5 6 4 9
5 192 0 5 6 8 3 9
6 137 0 1 7 3 2 8 9

Sample Input #2

1 10
2 20
3 0 0
4 27 1
5 33 6
6 26 1

Sample Output #2

1 31 0 11 15 13 18 14 17 2 5 16
2 40 0 9 12 2 13 15 4 17 1 6 14
3 117 0 9 10 7 11 5 8 15 3 13 16
4 12 0 15 8 17 6 18 5 3 1 10 11
5 -150 0 16 17 15 1 11 5 9 8 14
6 26 0 10 4 9 17 15 3 5 16 6 13

Sample Input #3

1 10
2 30
3 0 0
4 74 4
5 2 6
6 7 13

Sample Output #3

1 LOOP
2 LOOP
3 LOOP
4 214 0 22 10 25 11 21 7 15 13 1
5 -45 0 24 5 11 7 22 15 20 23 27
6 434 0 20 27 15 24 21 7 16 1 8