OJ 534

Frogger

Prof. Edson Alves

Faculdade UnB Gama

Freddy Frog is sitting on a stone in the middle of a lake. Suddenly he notices Fiona Frog who is sitting on another stone. He plans to visit her, but since the water is dirty and full of tourists' sunscreen, he wants to avoid swimming and instead reach her by jumping.

Unfortunately Fiona's stone is out of his jump range. Therefore Freddy considers to use other stones as intermediate stops and reach her by a sequence of several small jumps.

To execute a given sequence of jumps, a frog's jump range obviously must be at least as long as the longest jump occuring in the sequence.

O sapo Freddy está sentado em uma pedra no meio do lago. De repente ele observa que a sapa Fiona está sentada em outra pedra. Ele planeja visitá-la, mas como a água está suja e cheia de protetor solar dos turistas, ele não quer nadar e pretende alcançá-la pulando.

Infelizmente a pedra de Fiona está fora do alcance de seu pulo. Deste modo Freddy considera usar outras pedras como paradas intermediárias e chegar a ela por meio de vários saltos pequenos.

Para executar uma sequência de saltos, obviamente o alcance do pulo de um sapo deve ser, no mínimo, tão longo quanto o salto mais longo da sequência.

The frog distance (humans also call it minimax distance) between two stones therefore is defined as the minimum necessary jump range over all possible paths between the two stones.

You are given the coordinates of Freddy's stone, Fiona's stone and all other stones in the lake. Your job is to compute the frog distance between Freddy's and Fiona's stone.

A distância do sapo (humanos também a chamam distância minimax) entre duas pedras é definida como o alcance mínimo de salto necessário para, a partir da saída, chegar a uma determinada pedra, considerados todos os caminhos possíveis entre estas duas pedras.

Você receberá as coordenadas das pedras onde estão Freddy e Fiona e também de todas as outras pedras que estão no lago. Seu trabalho é calcular a distância do sapo entre as pedras de Freddy e Fiona.

Input

The input file will contain one or more test cases. The first line of each test case will contain the number of stones n $(2 \leq n \leq 200)$. The next n lines each contain two integers x_i, y_i $(0 \leq x_i, y_i \leq 1000)$ representing the coordinates of stone #i. Stone #1 is Freddy's stone, stone #2 is Fiona's stone, the other $n-2$ stones are unoccupied. There's a blank line following each test case. Input is terminated by a value of zero (0) for n.

Output

For each test case, print a line saying '`Scenario #`x' and a line saying '`Frog Distance =` y' where x is replaced by the test case number (they are numbered from 1) and y is replaced by the appropriate real number, printed to three decimals. Put a blank line after each test case, even after the last one.

Entrada

A entrada é composta por um ou mais casos de teste. A primeira linha de cada caso contém o número de pedras n $(2 \leq n \leq 200)$. As próximas n linhas contém, cada uma, dois inteiros x_i, y_i $(0 \leq x_i, y_i \leq 1000)$ representando as coordenadas da pedra #i. Freddy está na pedra #1, Fiona na pedra #2 e as $n - 2$ outras pedras estão desocupadas. Há uma linha em branco entre os casos de teste. A entrada termina com o valor (0) para n.

Saída

Para cada caso de teste, imprima uma linha com a mensagem '`Scenario #`x' e uma linha com a mensagem '`Frog Distance =` y' onde x é o número do caso de teste (os testes são numerados a partir de 1) e y é a distância do sapo, com três casas decimais. Imprima uma linha em branco após cada caso de teste, inclusive o último.

Exemplo de entrada e saída

Exemplo de entrada e saída

```
2
```

Exemplo de entrada e saída

```
2 ⟵ # de pedras
```

Exemplo de entrada e saída

```
2
```

Exemplo de entrada e saída

```
2

0 0
```

Exemplo de entrada e saída

```
2

0 0 ⟵ pedra do Freddy
```

Exemplo de entrada e saída

2

0 0

Exemplo de entrada e saída

2

0 0

3 4

Exemplo de entrada e saída

2

0 0

3 4 ⟵ *pedra da Fiona*

Exemplo de entrada e saída

2

0 0

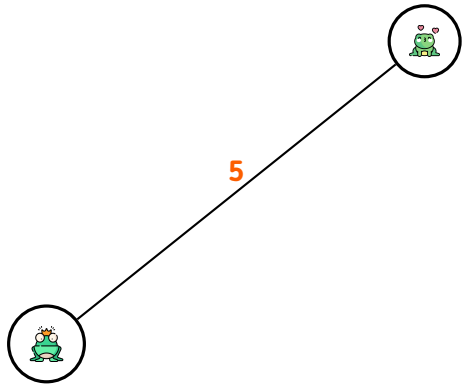
3 4

Exemplo de entrada e saída

2

0 0

3 4



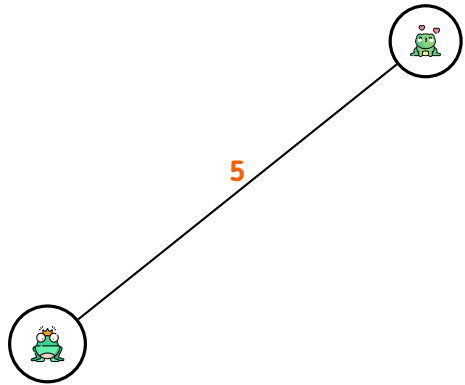
Exemplo de entrada e saída

2

0 0

3 4

5.000



Exemplo de entrada e saída

Exemplo de entrada e saída

```
3
```

Exemplo de entrada e saída

```
3
```

```
17 4
```

Exemplo de entrada e saída

3

17 4

Exemplo de entrada e saída

3

17 4

19 4

Exemplo de entrada e saída

3

17 4

19 4

Exemplo de entrada e saída

3

17 4

19 4

18 5

Exemplo de entrada e saída

3 17 4 19 4 18 5	3

Exemplo de entrada e saída

```
3
17 4
19 4
18 5
```

```
3
```

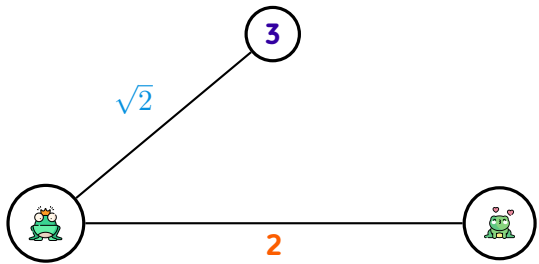
Exemplo de entrada e saída

3

17 4

19 4

18 5



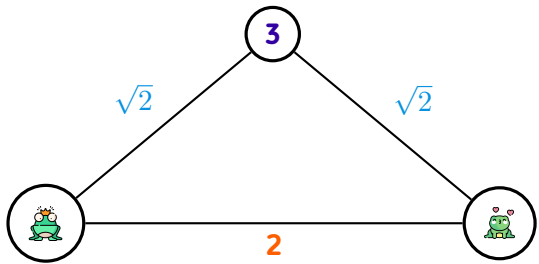
Exemplo de entrada e saída

3

17 4

19 4

18 5



Exemplo de entrada e saída

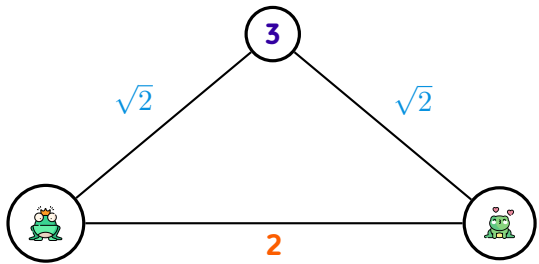
3

17 4

19 4

18 5

1.414



Solução

Solução

- ⋆ **As pedras são os vértices do grafo G**

Solução

- ⋆ **As pedras são os vértices do grafo G**
- ⋆ **Os pesos das arestas são as distâncias euclidianas entre as pedras**

Solução

- ⋆ **As pedras são os vértices do grafo G**
- ⋆ **Os pesos das arestas são as distâncias euclidianas entre as pedras**
- ⋆ **Seja $\mathrm{jump}[u]$ o alcance de salto mínimo para chegar a u a partir de 1**

Solução

- ⋆ **As pedras são os vértices do grafo** G
- ⋆ **Os pesos das arestas são as distâncias euclidianas entre as pedras**
- ⋆ **Seja** $\text{jump}[u]$ **o alcance de salto mínimo para chegar a** u **a partir de** 1
- ⋆ **Naturalmente,** $\text{jump}[1] = 0$

Solução

- ⋆ **As pedras são os vértices do grafo** G
- ⋆ **Os pesos das arestas são as distâncias euclidianas entre as pedras**
- ⋆ **Seja** $\mathsf{jump}[u]$ **o alcance de salto mínimo para chegar a** u **a partir de** 1
- ⋆ **Naturalmente,** $\mathsf{jump}[1] = 0$
- ⋆ **Para cada aresta** (u, v)**,** $\mathsf{jump}[u] = \min\{\mathsf{jump}[u], \max(\mathsf{jump}[v], \mathsf{dist}(u, v))\}$

```
double solve(int N, const vector<ii>& ps) {
    for (int i = 1; i <= N; ++i)
        for (int j = i + 1; j <= N; ++j)
        {
            auto [x, y] = ps[i];
            auto [a, b] = ps[j];

            dist[i][j] = dist[j][i] = hypot(x - a, y - b);
        }

    vector<double> jump(N + 1, oo);
    jump[1] = 0.0;

    for (int k = 1; k <= N - 1; ++k)
        for (int i = 1; i <= N; ++i)
            for (int j = 1; j <= N; ++j)
                jump[j] = min(jump[j], max(jump[i], dist[i][j]));

    return jump[2];
}
```