A - Oracle-Guided Road Network Planning

実行時間制限: 2 sec / メモリ制限: 1024 MB

ストーリー

昔々、高橋王国にはたくさんの都市が点在していましたが、道路が無く人々は都市間の移動に苦労していました。そこで国王の高橋くんは、都市をいくつかのグループに分け、グループ内の任意の2都市間が道路によって互いに行き来できるように都市を結ぶ道路を建設することで、国を発展させる計画を考えました。高橋くんはなるべく建設する道路の総長を短くしたいと考えていましたが、高橋王国には正確な地図が無く、各都市の位置がはっきりしていないという問題があった。

どのように道路を建設するべきか困った高橋くんは、王国一の占い師に助けを求めることにしました。占い師は、いくつかの都市を選ぶことで、それらの都市を互いに行き来できるような総長の最も短い道路の建設方法を知ることができます。ただし、占いの回数には限界があります。

なるべく建設する道路の総長を短くできるように、最適な占い方および道路の建設計画を考えてください。

問題文

平面上に N 個の都市があり、0 から $N-1$ までの番号が付けられている。都市 i の座標は (x_i, y_i) としてジャッジ内部では定まっているが、入力では与えない。その代わりに都市 i が存在する可能性のある座標の長方形範囲 lx_i, rx_i, ly_i, ry_i が与えられ、$lx_i \leq x_i \leq rx_i$ および $ly_i \leq y_i \leq ry_i$ を満たす。都市間の距離はユークリッド距離の小数点以下を切り捨てた整数値 $\mathrm{dist}(i, j) = \left\lfloor \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \right\rfloor$ として定義される。

長さ M の配列 G が与えられる。各グループの大きさがそれぞれ $G_0, G_1, \ldots, G_{M-1}$ となるように都市を M 個のグループに分割し、同じグループ内の都市が結になるように道路を建設したい。

回答の出力を行う前に、占いとして以下の形式のクエリを最大 Q 回まで行うことができる。

- 都市の集合 $C \subset \{0, 1, \cdots, N-1\}$ を選ぶ。ただし、その大きさ $l = |C|$ は、入力で指定される上限サイズ L に対して $2 \leq l \leq L$ を満たす必要があ
- クエリに対する応答として、都市集合 C 上での最小全域木を構成するための $l-1$ 本の道路それぞれが接続する都市の組が得られる。ここで、最小全域木以下のアルゴリズムによって構築される。
  - 辺の配列 E を、都市集合 C に含まれる任意の 2 都市間の辺で初期化する。
  - E を都市間の距離の昇順にソートする。都市間の距離は都市の真の座標に基づいて計算される。都市間の距離が等しい辺については、辺が結ぶ2都市 $u < v$ のペア (u, v) について昇順となるようにソートする。
  - E の先頭から順に辺を取り出し、その辺を追加したときに閉路ができないならば、その辺を最小全域木に追加する。
  - 最小全域木を構成する各辺について、辺が結ぶ2つの都市 $u < v$ のペア (u, v) について昇順にソートしたものをクエリに対する応答とする。

その後、以下を満たすような都市のグループ分けおよび道路の建設計画を出力せよ。

- N 個の都市を M 個のグループに分ける。各グループ i には G_i 個の都市を割り当てる必要がある。いずれのグループにも割り当てられない都市や、複数のグループに割り当てられるような都市があってはならない。
- 各グループ i について、そのグループに属する都市間を結ぶ道路を $G_i - 1$ 本選ぶ。このとき、各グループ内の任意の2都市間が道路によって相互に到達可能、すなわちグラフとして連結となる必要がある。

なお、それぞれの道路はその両端の 2 都市間のみを結ぶ。複数の道路が平面交差していても、道路から道路へ移ることはできない。

得点

出力した道路の建設計画における全ての道路の総長が絶対スコアとなる。 都市 i と都市 j を結ぶ道路の長さは、真の座標を用いて計算された都市間の距離 $\mathrm{dist}(i,j) = \left\lfloor \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \right\rfloor$ として計算される。 **絶対スコアは小さければ小さいほど良い。**

各テストケース毎に、 $\mathrm{round}(10^9 \times \frac{\text{全参加者中の最小絶対スコア}}{\text{自身の絶対スコア}})$ の**相対評価スコア**が得られ、その和が提出の得点となる。

最終順位はコンテスト終了後に実施されるより多くの入力に対するシステムテストにおける得点で決定される。 暫定テスト、システムテストともに、一部のテストケースで不正な出力や制限時間超過をした場合、そのテストケースの相対評価スコアは0点となり、そのテストケースにおいては「全参加者中の最小絶対スコア」の計算から除外される。 システムテストは CE **以外の結果を得た一番最後の提出**に対してのみ行われるため、最終的に提出する解答を間違えないよう注意せよ。

テストケース数

- 暫定テスト: 50個
- システムテスト: 3000個、コンテスト終了後に seeds.txt (https://img.atcoder.jp/ahc045/seeds.txt) (sha256=e4cb8e03f20f6a97dd082c75ad60cc31447bd24b5b3ebe5e64af4ac28712b618) を公開

相対評価システムについて

暫定テスト、システムテストともに、 CE 以外の結果を得た一番最後の提出のみが順位表に反映される。 相対評価スコアの計算に用いられる各テストケースの全参加者中の最小絶対スコアの算出においても、順位表に反映されている最終提出のみが用いられる。

順位表に表示されているスコアは相対評価スコアであり、新規提出があるたびに、相対評価スコアが再計算される。 一方、提出一覧から確認出来る各提出のスコアは各テストケース毎の絶対スコアをそのまま足し合わせたものであり、相対評価スコアは表示されない。 最新以外の提出の、現在の順位表における相対評価スコアを知るためには、再提出が必要である。 不正な出力や制限時間超過をした場合、提出一覧から確認出来るスコアは0となるが、順位表には正解したテストケースに対する相対スコアの和が表示されている。

実行時間について

実行時間には多少のブレが生じます。 また、システムテストでは同時に大量の実行を行うため、暫定テストに比べて数%程度実行時間が伸びる現象が確認されます。 そのため、実行時間制限ギリギリの提出がシステムテストで TLE となる可能性があります。 プログラム内で時間を計測して処理を打ち切るか、実行時間に余裕を持たせるようお願いします。

入出力

まず初めに、都市の個数 N 、回答における都市のグループの個数 M 、最大クエリ回数 Q 、クエリを行う都市の集合のサイズの上限 L 、都市の座標が含まれる長方形の幅や高さとして有り得る最大値 W 、各グループに割り当てる都市の個数を表す配列 G 、各都市が存在する可能性のある座標の長方形範囲 lx_i, rx_i, ly_i, ry_i が、以下の形式で標準入力から与えられる。

```
N M Q L W
G_0 G_1 ... G_{M-1}
lx_0 rx_0 ly_0 ry_0
⋮
lx_{N-1} rx_{N-1} ly_{N-1} ry_{N-1}
```

各値はそれぞれ以下の制約を満たす。

- $N = 800$
- $1 \leq M \leq 400$
- $Q = 400$
- $3 \leq L \leq 15$
- $500 \leq W \leq 2500$
- $1 \leq G_i \leq N$
- $\sum_{i=0}^{M-1} G_i = N$
- $0 \leq lx_i \leq rx_i \leq 10000$
- $0 \leq rx_i - lx_i \leq W$
- $0 \leq ly_i \leq ry_i \leq 10000$
- $0 \leq ry_i - ly_i \leq W$
- 入力は全て整数である。

上記の情報を読み込んだ後、以下のクエリを最大 Q 回まで繰り返す。

各クエリでは、都市の集合 $C \subset \{0, 1, \cdots, N-1\}$ を選ぶ。ただし、その大きさ $l = |C|$ は入力で指定される上限サイズ L に対して $2 \leq l \leq L$ を満たす必要がある。 これらの値を以下の形式で一行で標準出力に出力せよ。

```
? l C_0 C_1 ... C_{l-1}
```

出力の後には改行をし、更に標準出力を flush しなければならない。そうしない場合、TLEとなる可能性がある。

出力後に、都市集合 C 上での最小全域木を構成するための $l-1$ 本の道路それぞれが接続する都市の組 $a_i, b_i \in C$ が、以下の形式で標準入力から与えられる

$$
\begin{aligned}
&a_0\ \ b_0\\
&\vdots\\
&a_{l-2}\ \ b_{l-2}
\end{aligned}
$$

最大 Q 回までのクエリ処理が完了後、まず初めに記号 ! のみを一行で標準出力に出力せよ。

```
!
```

その後、各グループに対する回答を以下の形式で M グループ分繰り返し出力せよ。グループ k に対する回答では、そのグループに割り当てる都市の集合 C $\{0, 1, \cdots, N-1\}$ と、その都市間を結ぶ道路の組 $a_i, b_i \in C$ を以下の形式で出力せよ。

$$
\begin{aligned}
&C_0\ \ C_1\ \ \cdots\ \ C_{G_k-1}\\
&a_0\ \ b_0\\
&\vdots\\
&a_{G_k-2}\ \ b_{G_k-2}
\end{aligned}
$$

この時、出力は以下の制約を満たす必要がある。

- 各都市は必ず1つのグループに割り当てられる。どのグループにも割り当てられない都市や、複数のグループに割り当てられる都市は存在しない。
- 各グループ内の任意の2都市間は道路によって相互に到達可能、すなわちグラフとして連結である。

例

q	Output	Input

事前情報		`5 2 3 3 500`
		`3 2`
		`1375 1648 351 624`
		`1773 1900 3660 3787`
		`2922 3231 558 867`
		`5358 5640 8585 8867`
		`3218 3684 3330 3796`

0	`? 3 4 1 2`	`1 4` `2 4`
1	`? 3 1 3 4`	`1 4` `3 4`

回答	`!`
	`3 4 1`
	`3 4`
	`1 4`
	`2 0`
	`0 2`	

例を見る (https://img.atcoder.jp/ahc045/jOO09LxU.html?lang=ja&seed=0&output=sample)

入力生成方法

L 以上 U 以下の整数値を一様ランダムに生成する関数を $\mathrm{rand\_int}(L, U)$、 L 以上 U 未満の実数値を一様ランダムに生成する関数を $\mathrm{rand\_double}(L, U)$ す。

M, L, W の生成

- $M = \left\lfloor (\mathrm{rand\_double}(1.0, 20.0))^2 \right\rfloor$
- $L = \mathrm{rand\_int}(3, 15)$
- $W = \mathrm{rand\_int}(500, 2500)$

G の生成

- 1 以上 $N-1$ 以下の整数値を、重複しないようにして一様ランダムに $M-1$ 個生成し、昇順に並べたものを $A_1, A_2, \cdots, A_{M-1}$ とする。
- $A_0 = 0, A_M = N$ として、$G_i = A_{i+1} - A_i$ とする。

$x_i, y_i, lx_i, rx_i, ly_i, ry_i$ の生成

- $x_i = \mathrm{rand\_int}(0, 10000)$
- $y_i = \mathrm{rand\_int}(0, 10000)$
- $w_i = \mathrm{rand\_int}(0, W)$
- $dx_i = \mathrm{rand\_int}(0, w_i)$
- $rx_i = x_i + dx_i$
- $lx_i = rx_i - w_i$
- $dy_i = \mathrm{rand\_int}(0, w_i)$
- $ry_i = y_i + dy_i$
- $ly_i = ry_i - w_i$
- lx_i, rx_i, ly_i, ry_i が 0 未満の場合は 0 に、 10000 を超える場合は 10000 に置き換える。

ツール(入力ジェネレータ・ビジュアライザ)

- Web版 (https://img.atcoder.jp/ahc045/jOO09LxU.html?lang=ja): ローカル版より高性能でアニメーション表示が可能です。
- ローカル版 (https://img.atcoder.jp/ahc045/jOO09LxU_v2.zip): 使用するにはRust言語 (https://www.rust-lang.org/ja)のコンパイル環境をご用意下さい。
  - Windows用のコンパイル済みバイナリ (https://img.atcoder.jp/ahc045/jOO09LxU_windows_v2.zip): Rust言語の環境構築が面倒な方は代わりにこち利用下さい。

コンテスト期間中に、ビジュアライズ結果の共有や、解法・考察に関する言及は禁止されています。ご注意下さい。

ツールで用いられる入出力ファイルの仕様

ローカルテスタに与える入力ファイルは解答プログラムに与えられる事前情報の末尾に以下の形式の情報が追加されている。

```
x_0 y_0
⋮
x_{N-1} y_{N-1}
```

x_i, y_i は都市 i の真の座標であり、解答プログラムには与えられない。

サンプルプログラム

▶ Pythonによるサンプル実装

以下の処理を実装しています。

1. 都市を与えられた長方形範囲の中心座標 $\left(\frac{lx_i+rx_i}{2}, \frac{ly_i+ry_i}{2}\right)$ でソートし、その順番でグループに分割する。
2. 各グループについて、3都市クエリを2都市ずつスライドしながら行い、クエリの応答として得られた辺をそのまま回答に追加する。
3. 各グループについて最後の都市がクエリできなかった場合は、グループ内の直前の都市との辺を回答に追加する。