MC346 - PROJETO PYTHON
Caixeiro Viajante

Escreva um programa em Python que resolve o problema do caixeiro viajante, ou seja, dado as coordenadas de n cidades, e a cidade de origem, determine o caminho que sai da cidade de origem, visita as outras $n-1$ cidades (sem repetição) e retorna à origem que possui a menor distancia percorrida total.

Para a solução deste problema voce terá que gerar todas as permutações das $n-1$ cidades que não a origem, e verificar a distancia percorrida total para cada uma destas alternativas.

Os dados de entrada estão no seguinte formato:

```
cidade1  coordx  coordy
cidade2  coordx  coordy
...
cidaden  coordx  coordy
origem
```

onde um ou mais brancos separam os dados de cada linha. Cada cidade tem duas coordenadas (x e y) e a distancia entre cidades deve ser calculado como a distancia euclidiana entre pontos em 2 dimensões. A ultima entrada é a cidade origem e destino final do caixeiro viajante.

Imprima o caminho, iniciando e terminando com a origem, e com uma cidade por linha.

```
origem
cidadea
cidadeb
...
origem
```

Na linha seguinte imprima distancia total percorrida do menor caminho arredondado para uma casa decimal após o ponto.

Note que há sempre 2 caminhos com menor custo. Se $[a, b, c, d, a]$ é o caminho de menor distancia percorrida, então o reverso $[a, d, c, b, a]$ também o é. Destas duas alternativas o seu programa deve imprimir aquele caminho onde o nome da 2a cidade (o b no primeiro caso) é lexicograficamente menor que o nome da penúltima cidade (o d).

O susy rodara seu programa como:

```
python3 ex1.py < arqx.in
```

Há duas maneiras de pensar na geração de todos os arranjos de uma lista. A vista em aula é:

- para todos os elementos da lista, compute os arranjos das listas restantes (sem o elemento) e inclua o elemento na frente de todas as listas
- para a lista [1,2,3], e para o elemento 1, calcula os arranjos da lista que sobra ([2,3]). esses possíveis arranjos sao [[2,3], [3,2]], e insere o 1 na frente de todos [[1,2,3],1[3,2].
- da mesma forma para o elemento 2. Os arranjos da lista restante ([1,3]) sao [[1,3],[3,1]] e insere-se o 2 na frente de todas as listas que resulta em [[2,1,3],[2,3,1]]
- e assim por diante

A segunda forma é:

- separa o primeiro elemento da lista, compute os arranjos do restante da lista e insira o primeiro elemento em todas as posições possíveis dos arranjos.
- para a lista [1,2,3], separe o 1, e compute todos os arranjos da lista restante [2,3]. que resulta em [[2,3],[3,2]]

- para cada uma das listas (por exemplo [2,3]) insira o 1 em todas as posições possíveis, resultando em [[1,2,3],[2,1,3],[2,3,1]]
- e da mesma forma para as outras listas (nesse caso [3,2])