MC336 - PROJETO 5

Imprimir a menor distancia de um vértice a todos os outros vértices de um grafo.

Escreva um predicado em Prolog que recebe uma lista de arestas `LA`, da forma

```
[[a, b, 5.3], [b, c, 12] ...]
```

e um vértice `V`.

A lista de arestas acima indica que há um arco entre os vértices `a` e `b` e o custo deste arco é 5.3, e que há um arco entre `b` e `c` com custo 12. Um arco aparece apenas uma vez na lista de entrada, e portanto não haverá um arco entre `b` e `a` na lista.

O predicado `pred(LA,V,Dist)`, modo + + - , retorna em `Dist` uma lista das distancias entre o vértice `V` e os vértices do grafo, da forma

```
[ [a, 16.0], [b, 2.6], [c, inf], [d, 0], ...]
```

ordenados por ordem crescente do nome do vértice, um por linha. A dupla `[c, inf]` indica que não há um caminho entre o vértice inicial e o vértice `c` - e portanto, indica um custo ou distância infinitos. A dupla `[d, 0]` indica que `d` era o vértice inicial.

Use o algorítimo de Dijkstra para calcular as distancias (projeto do GPS em Python), mas em vez de parar no vértice destino (como voce fez no projeto de GPS), voce faz para todos os vértices do grafo. Mas note que o algoritmo pode parar antes de alcançar todos os vértices, já que o grafo pode não ser conectado.

Uma vez que o predicado `pred` esteja funcionando, inclua no seu arquivo o seguinte predicado:

```
main :- read(LA),
        read(V),
        ignore(pred(LA,V,Dist)),
        print(Dist),
        nl.
```

que lê os dados do stdin, roda seu programa, e imprime a resposta no stdout. O nome `main` é importante pois esse é o predicado que o Susy vai chamar para rodar o seu programa. Na verdade o Susy vai chamar os seu programa da seguinte forma:

```
swipl -q -f ex4.pl -t 'main' < arq1.in
```