Projeto 4

Data: 25/9 (ate meia noite)

Pode ser feito individualmente ou em grupos de até 2 pessoas.

- se for feito em duplas, escreva um comentário no topo do arquivo com o nome e RA dos membros do grupo
- se for feito em duplas, apenas um dos membros do grupo submete.

1 Uma versão simplificada do Dykstra

Para um gráfico não direcionado, e dado um vértice de origem e um de destino, usar o algoritmo de Dykstra para calcular a menor distancia entre a origem e o destino.

O gráfico será dado como uma lista de triplas `[("ab1","b67",10.4),("ab1","cc",11.2)...]` onde os 2 primeiros componentes da tupla são os nomes (um string) dos vértices, e o terceiro componente a distancia entre os 2 vértices. Note que se a distancia entre os vértices "ab1" e "b67" é 10.4 então a distancia entre "b67" e "ab1" também é de 10.4 mas a lista **não** vai conter uma entrada `( "b67",  "ab1", 10,4)`.

O problema é uma versão simplificada do Dykstra. Na versão "normal" do Dykstra queremos não só a menor distancia entre 2 vértices mas também o caminho com essa menor distancia. Mas para esse problema não precisa computar o caminho, apenas a menor distancia.

Você pode assumir que o grafo é conectado, ou seja existe um caminho entre quaisquer 2 nós do grafo.

Você não precisa usar estruturas de dados complexas como um "priority queue" que sao $O(1)$ para achar o minimo. Pode fazer uma busca linear para achar o mínimo e usar as funções já disponíveis no Haskell.

A função principal deve se chamar `proj4` e ela recebe 3 argumentos, o grafo no formato especificado, o nó origem e o nó destino.

Vc pode usar as bibliotecas padrão do haskell.

A pagina do Dykstra na wikipedia https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm tem uma animaçao do algoritmo para um grafo simples. Aquele grafo corresponde ao dado abaixo.

```
[ ("1", "2", 7),
  ("1", "3", 9),
  ("1", "6", 14),
  ("2", "3", 10),
  ("2", "4", 15),
  ("3", "4", 11),
  ("3", "6", 2),
  ("4", "5", 6),
  ("5", "6", 9)
  ]
```

2 grafo não necessariamente conectado

Esta parte do projeto vale apenas 1/4 da nota total do projeto.

Na parte anterior assuminos que o grafo era conectado. No miolo do Dykstra, ha o passo onde precisamos achar a aresta de menor tamanho que liga um vertice já visitado com um não visitado. Se o grafo é conectado havera sempre pelo menos 1 aresta entre os 2 conjuntos de nós. Se o grafo não é conectado, pode não haver nenhuma aresta ligando esses 2 conjuntos.

Agora o grafo não será necessariamente conectado e sua funcão deve retornar alguma indicacão que não existe um caminho que liga o vertice origem do vertice destino. Voce deve retornar um `Maybe distancia-minima`: um `Just x` indica que a distancia minima é `x` e o `Nothing` indica que não há um caminho.

Sem ter ainda implementado esse problema, eu acho que é suficiente no passo acima, vc pode retornar um Maybe aresta. Eu acho que se não há essa aresta isso vai acabar contaminando as computações subsequentes em `Nothing`. Infelizmente vc precisará mudar a sintaxe do programa, para usar o `do` e utilizar a monada de forma conveniente.