U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Conceção e Análise de Algoritmos

Bike-Sharing

2MIEIC06 – Grupo C

(7 de abril de 2018)

Bernardo Santos **up201504711@fe.up.pt**

João Gonçalves **up201604245@fe.up.pt**

Miguel Moura **up201609149@fe.up.pt**

Índice

Descrição do Tema – “Bike Sharing”

O projeto realizado no âmbito da unidade curricular de Conceção e Análise de Algoritmos tem como objetivo o processamento de um mapa real, obtido pelo parsing de uma porção de mapa obtida através do “OpenStreetmaps”, e respetivo cálculo do caminho da localização de um utente até um ponto de partilha de bicicletas.

A plataforma Eco Rides consiste na partilha de bicicletas através de pontos espalhados por vários locais, onde um número de bicicletas está disponível para aluguer.

Na escolha do caminho, o utilizador vai ter várias possibilidades de escolha, sendo elas: o caminho que leva ao ponto de partilha mais perto do biciclista; o caminho que leva ao ponto de partilha em que o preço do alugamento das bicicletas é mais barato e, por fim, o caminho mais rentável.

No contexto do problema, o “caminho mais rentável” vai assumir diferentes variáveis, tal como o custo das bicicletas nessa certa loja, a distância da pessoa até ao ponto de partilha e, por fim, a topografia(elevação) em que se encontra o ponto, visto que as viagens em bicicletas são extremamente afetadas pelos percursos predominantemente ascendentes.

Definição e Formalização do Problema

Dados de entrada

Construção de um grafo dirigido pesado: Gi = <Vi, Ei>, composto por:

- V - Vértices (representam pontos que ligam duas retas/pontos de partilha).
  - Adj ⊆ E - conjunto de arestas que partem do vértice.
- E - Arestas (representam a ligação entre pontos) com:
  - ID - idenficador único de uma aresta.
  - W – peso da aresta (distância entre 2 vértices).
  - Dest ∈ Vi - vértice de destino.
- I - Ponto Inicial.
- F - Ponto Final.

Dados de saída

Todos os vértices por onde o utilizador passou, de forma a poder otimizar o seu trajeto e o valor da distância.

- Trajeto = { Vi }, sendo o i de 1 até n.
- Valor da distância.

Restrições

As restrições para este grafo vão ser que:

- Os valores das distâncias têm de ser superiores a 0.
- No trajeto obtido, os vértices não podem estar mais do que uma vez.
- O valor de vértices e de arestas tem de ser sempre maior que 0.
- O vértice de destino tem de ser alcançável.

Função objetivo

A solução ótima deste problema passa por calcular o mínimo valor possível da distância, de forma a disponibilizar ao utilizador os melhores trajetos para o critério de seleção que este desejar (preço, distancia ou rentabilidade).

Min (Valor da distância):

- Valor da distância = $\sum_{i=1}^{n}$ (E i,j), sendo que i,j pertence ao trajeto.

Solução Implementada

O primeiro passo na realização do trabalho foi definir a área que iríamos utilizar, e para tal, decidimos usar um mapa real, através do "OpenStreetMaps", de onde obtivemos a maior parte dos ficheiros de texto que acabaríamos por usar. Depois disto, decidimos atribuir ficticiamente a localização dos pontos de partilha, bem como a informação dos clientes em ficheiros de texto.

Usamos o parser disponibilizado nas aulas teóricas a criação dos ficheiros de Nodes, Roads e Subroads.

Depois de termos os ficheiros todos definidos, criámos as funções destinadas à leitura de ficheiros de texto, bem como as classes Vertex, Edge e Graph e os seus respetivos métodos, como, por exemplo, o algoritmo de Dijkstra, que nos pareceu ser o algoritmo mais indicado a ser usado neste trabalho para calcular o caminho mais curto, uma vez que se trata de um grafo pesado. Para isto, usamos como base as soluções das fichas 5 e 6 das aulas práticas.

Rapidamente percebemos que iríamos precisar de criar uma classe para os nós de forma a guardar a localização dos vértices (latitude, longitude). Sendo que cada nó vai conter como dados, o seu Id, que não pode ser repetido, a sua latitude e a sua longitude.

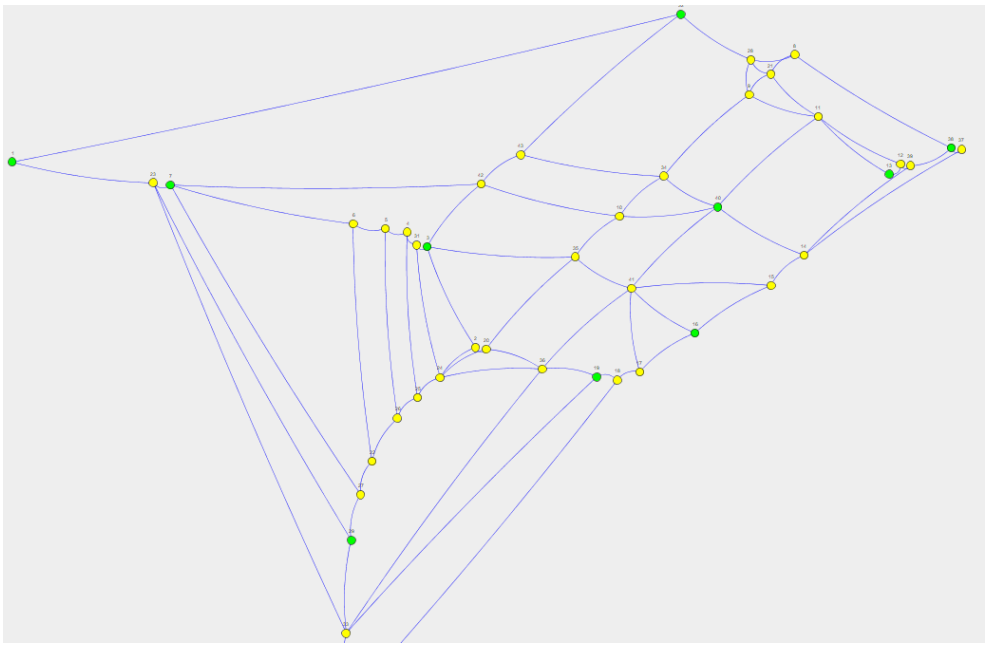
Cada ponto de partilha é um objeto que contem o nó a ele associado, sendo que vai usar a altitude para calcular o preço da bicicleta. A classe dos pontos de partilha é uma classe derivada da classe dos Nós, uma vez que um ponto de partilha é obviamente um nó, mas com mais alguns atributos.

As ruas, que representam um conjunto de arestas, contêm não só um ID e o nome da mesma, mas também a informação sobre a existência de dois sentidos (identificadas pelo 0) ou apenas um sentido (identificadas pelo 1).

Quanto às sub-estradas (vão representar as retas que a rua faz), vão ser guardadas num ficheiro de texto, o id da rua em que se localizam, de forma a saber se a rua é bidirecional ou não, e o id dos nós que estas ligam.

No mapa, os pontos de Partilha vão ser identificados pela cor Verde, as arestas pela cor Azul, e por fim, para identificar o caminho mais curto no mapa, definimos que os nós que constituem o caminho teriam a cor Vermelha.

Mapa total:



Depois, na classe dos utentes, vamos guardar a informação relativa às pessoas que reservaram bicicletas, mais especificamente, vamos guardar o seu id, o seu nome, a sua localização atual (x, y), se o utente está a utilizar uma bicicleta e, se sim, a bicicleta que este está a usar e, por fim, o seu histórico de reservas.

Casos de Utilização

Classes

Sistema → classe que contem a informação do programa em execução assim como os métodos para o iniciar.

Node → classe representativa de um local (nó), contendo um ID, a latitude e a longitude desse nó.

Street → classe representativa de uma estrada (várias arestas), contendo um ID, o nome da rua, um booleano indicando se é ou não de dois sentidos e um vector de nós que a estrada liga.

PontoPartilha → classe que vai guardar a altitude, a capacidade, o nome, o preço das bicicletas e as bicicletas existentes de um ponto de partilha

Bicicleta → classe que vai atribuir o nome a uma bicicleta, de forma a ser possível identificar as bicicletas.

Opção Inválida → classe que vai ser utilizada de forma a lançar exceções quando o utilizador utiliza uma opção errada no programa.

Abertura Falhada → classe que vai ser utilizada verificar se os ficheiros de texto foram abertos com sucesso.

Localização → classe que vai definir como vai ser guardada as coordenadas em que os utentes se localizam.

Localização → classe que vai guardar a estrutura de como a data vai ser guardada (dia/mês/ano).

Ficheiros

subroads.txt – ficheiro contendo a informação relativa a arestas como o seu ID e os nós que liga.

nodes.txt – ficheiro contendo o ID do nó, e as coordenadas x e y no mapa.

pontosPartilha.txt – ficheiro contendo o nome, o ID, a localização, a altitude, a capacidade e, por fim, o preço das bicicletas de cada ponto de Partilha.

roads.txt – ficheiro contendo a informação das ruas, como o seu ID, o nome da mesma, e se é bidirecional (0), ou unidirecional (1).

utentes.txt – ficheiro que guarda o nome, a localização atual, o numero de reservas totais, e o respetivo histórico (nome de bicicleta + data da reserva) de cada utente

Algoritmos

Djikstra

O algoritmo que usamos para o cálculo do trajeto mais curto foi o algoritmo de Dijkstra, visto tratar-se de um grafo pesado com valores positivos. Este algoritmo cria uma arvore de caminhos mais curtos desde um vértice inicial até todos os outros vértices do grafo.

Pseudocódigo do algoritmo:

Algoritmo de Dijkstra (adaptado)

```
DIJKSTRA(G, s):   // G=(V,E), s ∈ V
1.    for each v ∈ V do
2.       dist(v) ← ∞
3.       path(v) ← nil
4.    dist(s) ← 0
5.    Q ← ∅   // min-priority queue
6.    INSERT(Q, (s, 0)) // inserts s with key 0
7.    while Q ≠ ∅ do
8.       v ← EXTRACT-MIN(Q) // greedy
9.       for each w ∈ Adj(v) do
10.         if dist(w) > dist(v) + weight(v,w) then
11.            dist(w) ← dist(v)+ weight(v,w)
12.            path(w) ← v
13.            if w ∉ Q then  // old dist(w) was ∞
14.               INSERT(Q, (w, dist(w)))
15.            else
16.               DECREASE-KEY(Q, (w, dist(w)))
```

Em termos de analise de complexidade espacial e temporal este algoritmo apresenta:

- Espacialmente: O (N^2) onde N representa o número de nós.
- Temporalmente: O ([M+N] log n) onde N representa o número de nós e M o de arestas.

Este algoritmo revela-se como o mais eficiente dos lecionados, impondo apenas a restrição dos valores das arestas serem positivos.

Programa

Ao iniciar o programa, este vai executar o checkin do sistema; vai apresentar a interface; vai realizar a execução das funções selecionadas, que são: o processamento da informação dos ficheiros de texto, respetiva criação do grafo e visualização do mesmo no GraphViewer e, no final, faz checkout do sistema.

Depois disto, o programa vai necessitar do input do utilizador, a quem lhe vão dar 4 opções diferentes de escolha: Registar, Entrar, Administrar e Sair.

As duas primeiras opções destinam-se a utentes, sendo que a primeira se destina à inscrição de novos utilizadores e a segunda leva a outro menu com várias funcionalidades. Já a terceira opção destina-se à gestão da plataforma por parte de administradores sendo, como tal, apenas acessível por pessoal autorizado e requerendo uma password ("1234"). A última opção simplesmente sai do programa.

```
APPLICATION LOADING

######  ####### #######      ##########  ##  #####    ######  #######
##      ##      ##   ##      ##      ##  ##  ##  ##   ##      ##
#####   ##      ##   ##      ##  ######  ##  ##   ##  #####   #######
##      ##      ##   ##      ##    ##    ##  ##  ##   ##          ##
######  ####### #######      ##     ###  ##  #####    ######  #######

1 - Registar
2 - Entrar
3 - Administrador
4 - Sair

Introduza uma opcao (1-4):
```

Se o input do utilizador não corresponder a nenhuma opção dada. Este vai lançar uma exceção e dar outra tentativa ao utilizador, sendo que se não estiver no menu inicial, só vai dispor de três tentativas, até voltar a esse menu inicial.

```
Introduza uma opcao (1-4): p
Opcao invalida(p) ! Tente novamente.

Introduza uma opcao (1-4): 7
Opcao invalida(7) ! Tente novamente.

Introduza uma opcao (1-4):
```

No que diz respeito ao menu de utilizadores, após o utilizador identificar o utente a quem pretende trabalhar, são apresentadas seis possibilidades (além da alternativa de sair):

```
Utentes:

   Nome          ID    GPS
-> Andre         1          (41.1737  ,-8.60358 )
-> Jose          2          (41.1732  ,-8.59805 )
-> Ana           3          (39.8532  ,-8.2315  )
-> Rui           4          (40.1235  ,-8.12395 )
-> Cesar         5          (41.8273  ,-7.12385 )
-> Joana         6          (41.173   ,-8.60026 )
-> Roberto       7          (41.1779  ,-8.59924 )
-> Bernardo      8          (41.1748  ,-8.5904  )
-> Pedro         9          (41.1752  ,-8.60007 )

Login [Utente ID]: 1
#######  ####### #######      ##########  ##  #####    ######  #######
##       ##      ##   ##      ##      ##  ##  ##  ##   ##      ##
#####    ##      ##   ##      ##  ######  ##  ##   ##  #####   #######
##       ##      ##   ##      ##    ##    ##  ##  ##   ##            ##
######   ####### #######      ##     ###  ##  #####    ######  #######

Bem-Vindo !

1  - Alugar bicicleta
2  - Devolver bicicleta
3  - Historico
4  - Atualizar localizacao
5  - Pontos de partilha mais proximos
6  - Informacoes sobre ECO_RIDES
7  - Logout


Introduza uma opcao (1-7):
```

Na primeira opção o sistema vai alugar a bicicleta no ponto de partilha mais perto da localização desse utente, se o utilizador permitir. Para a segunda hipótese, para poder devolver uma bicicleta, o utilizador necessita de primeiro a alugar e só posteriormente devolvê-la. Se o utilizador não atualizar a localização do cliente, este vai automaticamente devolver a bicicleta no ponto de partilha mais próximo disponível. Se a localização mudar, aquando for a devolver, o utilizador vai ter as opções de devolver no ponto de partilha mais perto de si, no ponto mais barato ou no ponto mais rentável, sendo este uma mistura da distância, preço e topografia.

A terceira opção apresenta o histórico de pagamentos do utente, fazendo uma listagem das utilizações de bicicletas que já foram pagas. Na quarta opção o utente consegue alterar a sua localização atual para uma nova localização. Na quinta opção, tal como o nome indica, apresenta ao utilizador uma listagem, por ordem crescente de distância, dos pontos de partilha mais próximos da sua localização, o que permite por sua vez orientar o utente nas suas decisões. A sexta opção é meramente informativa, mostrando no ecrã a informação relativa aos pontos de partilha e aos utentes. Na última opção o sistema volta para o menu principal, se não houver uma bicicleta naquele momento que ainda não tenha sido devolvida.

No menu de administração, o utilizador vai dispor de duas opções (além da possibilidade de sair),

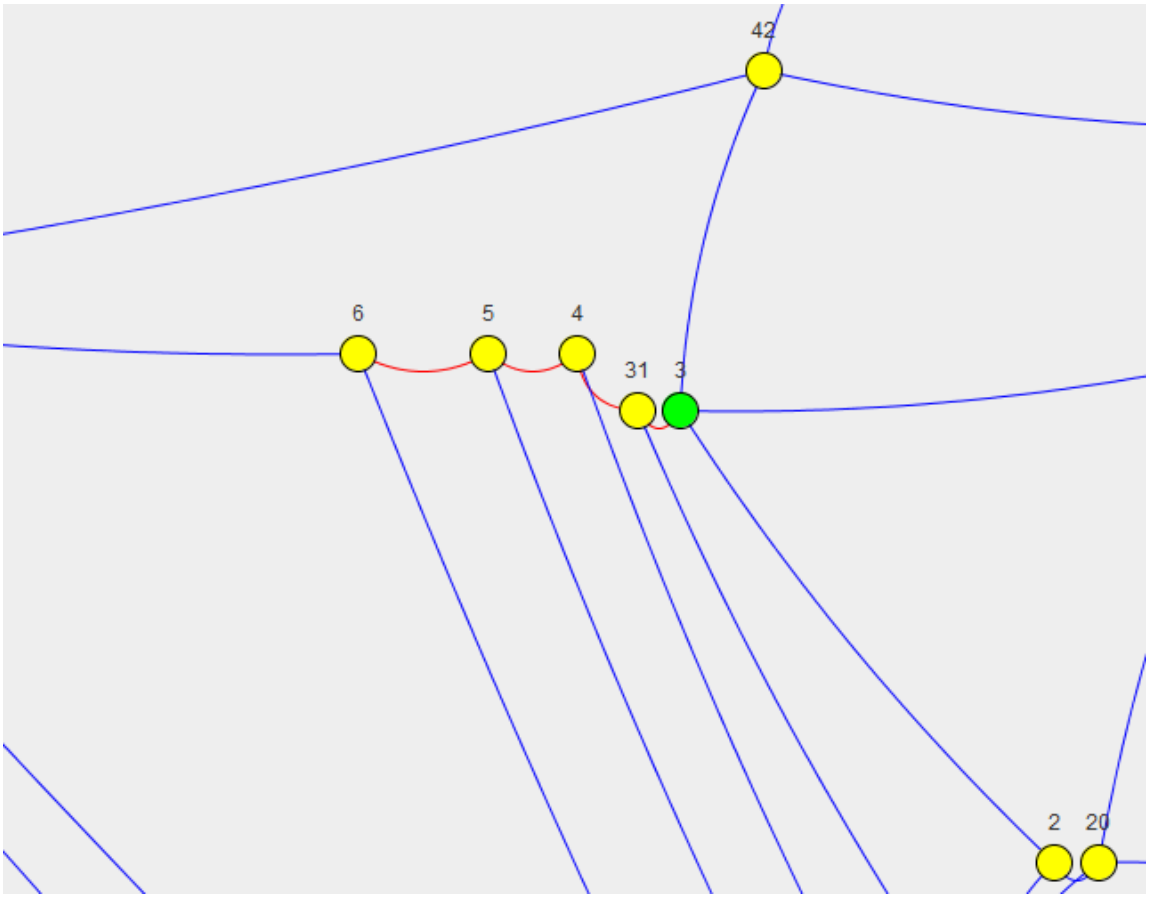
```
Administracao

1 - Remover utente
2 - Informacoes sobre ECO_RIDES
3 - Sair

Introduza uma opcao (1-3):
```

sendo que a primeira vai remover o utente que o utilizador identificar pelo Id, enquanto que segunda opção é meramente informática e vai apresentar o mesmo que a opção 6 do menu dos utentes.

De seguida apresentamos o resultado, no Graph Viwer, do caminho mais curto que o cliente percorre até chegar ao ponto de partilha que se encontra mais perto deste:



Dificuldades

Durante a realização deste projeto as maiores dificuldades foram: compreender como funcionava o GraphViewer, visto que as coordenadas dadas pelo open street maps eram pouco distantes; o cálculo das distâncias entre diferentes localizações, porque às vezes dava-nos infinito em algumas distâncias.

Distribuição

Todos os membros do grupo colaboraram quer na estratificação do problema quer na compreensão do enunciado. Relativamente à implementação da solução, a cada um elemento foi atribuído partes do código diferentes, porém, tentamos sempre trabalhar juntos de forma a entreajudar-nos uns aos outros.

Estivemos juntos inúmeras vezes na faculdade a trabalhar em grupo e todos os elementos se apresentaram sempre.

A distribuição foi acordada entre todos e, portanto, não houve nenhum problema a nível de distribuição de tarefas.

Conclusão

Trabalhar em grupo é a melhor forma de nos preparar para o futuro. É isto que inexoravelmente concluímos. São nos apresentados outros tipos de desafios, temos de trabalhar em equipa, temos prazos a cumprir, temos de ler e perceber código que não é nosso, o que nos alerta e nos prepara para escrever um código mais limpo e bem comentado de forma a que quem o leia a seguir o compreenda como nós o compreendemos.

Relativamente aos específicos deste trabalho, foi a primeira vez que fomos confrontados com algoritmos "pesados". É importante para compreender que a grande escala, algoritmos de "brute-force" não serão uma opção viável. Temos de ter em atenção a otimização e a complexidade e isso ficou bem patente com este trabalho.

Para além desta noção da importância da algoritmia tivemos também um primeiro contacto com o que é trabalhar com mapas reais, com aplicações reais na vida prática.

A utilização e manipulação de grafos também se mostrou bastante elucidativa e mostrou-nos a importância de trabalhar com este tipo de estruturas no futuro. Penso que no fim do trabalho todos os membros se mostram confortáveis para trabalhar com grafos.

Em suma, este trabalho foi extremamente enriquecedor, a nível coletivo, na medida em que nos prepara para o que é o nosso futuro. Trabalhar em grupo. Resolver questões inesperadas. Prazos a cumprir. A importância dos algoritmos. A atenção ao detalhe. A escrita de melhor código e ultimamente o crescimento pessoal e profissional.

"Se quer ir rápido, vá sozinho. Se quer ir longe, vá em grupo."