Navegação nos Transportes Públicos do Porto

Algoritmos e Estruturas de Dados

Alunos - G82

João Teixeira - up202005437
Lucas Sousa - up202004682
Rui Soares - up202103631

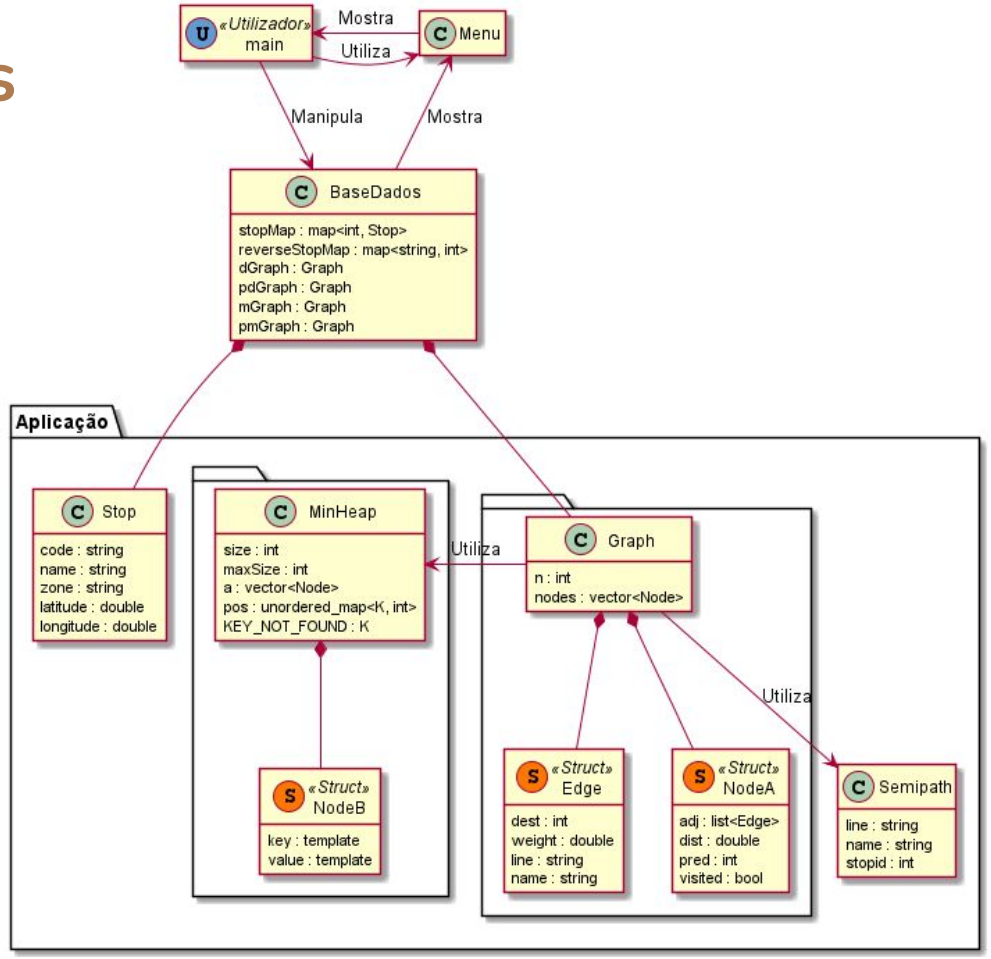
Orientadores

Ana Paula Cunha da Rocha
Pedro Manuel Pinto Ribeiro
Sofia Cardoso Martins

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Classes e Relações

- BaseDados
- Graph
- Menu
- MinHeap
- Semipath
- Stop



Leitura do *dataset*

- Para ler as stops, temos uma função na main que lê todas as linhas de stops.csv, com o getline, cria um objeto “Stop” (paragem) com todos os devidos atributos e adiciona-a a um mapa com um ID, de 1 ao número de paragens, correspondente.

  Ao mesmo tempo, é criado outro mapa, com chave Código e valor ID.

- Para ler as lines, temos, na classe BaseDados, duas funções:
  - A função loadAllLines, que lê todas as lines existentes em lines.csv e chama a função loadLine para cada linha;
  - A função loadLine, que vai ao ficheiro .csv de uma linha específica e, para cada par origem destino, adiciona uma aresta aos devidos grafos comas duas paragens, código e nome da linha.

Grafos

Todos os 4 grafos são dirigidos, têm peso e contém todas as paragens diurnas ou noturnas do dataset.

Os nós são os IDs atribuídos às paragens no início do programa.

As arestas são as linhas que ligam duas paragens. Guardam o nome e código da linha.

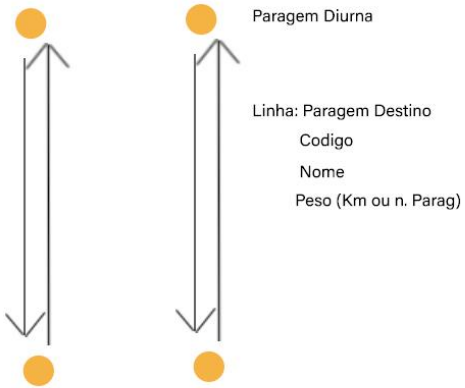
Os grafos estão classificados como Diurnos ou Noturnos e como tendo ou não caminhos a pé (<100m) entre paragens próximas.

	Dia	**Noite**
Pé	dGraph	mGraph
~ Pé	pdGraph	pmGraph

Grafos - Dia

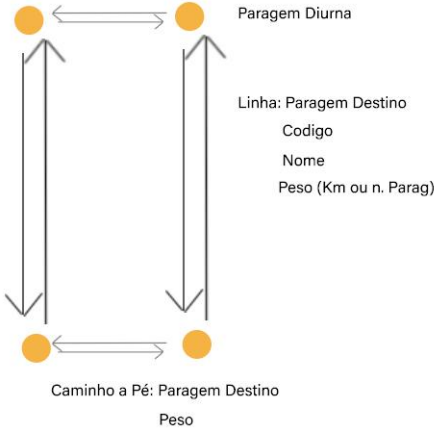
DIA e NORMAL (dGraph)

- Contém todas as paragens.
- Contém apenas linhas não marcadas como M.
- Não contém arestas a pé.



DIA a PÉ (pdGraph)

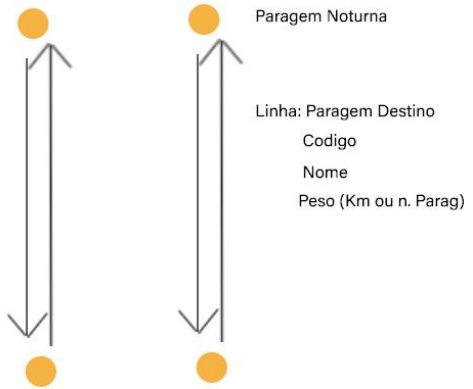
- Contém todas as paragens.
- Contém apenas linhas não marcadas como M.
- Contém arestas a pé entre paragens até 100 metros de distância.



Grafos - Noite

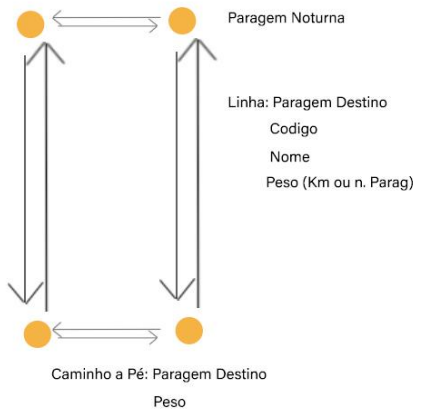
NOITE e NORMAL (mGraph)

- Contém todas as paragens.
- Contém apenas linhas marcadas como M.
- Não contém arestas a pé.



NOITE a PÉ (pmGraph)

- Contém todas as paragens.
- Contém apenas linhas marcadas como M.
- Contém arestas a pé entre paragens até 100 metros de distância.



Funcionalidades Implementadas

- Procura de Origem/Destino por código ou coordenadas.
- Paragem mais próxima ou lista de paragens num dado raio.
- Escolha de Caminho com menor Distância ou menos Paragens.
- Escolha de andar a pé entre paragens.
- Apresentação da Distância Total em Km ou do Número Total de Paragens a Percorrer.
- Apresentação do Caminho, mostrando, entre cada paragem, a linha a seguir ou andar a pé.

Funcionalidades - Origem/Destino

- O utilizador deve, no decorrer do programa, escolher a origem e o destino do seu trajeto.
- O utilizador, para escolher a origem e o destino, tem as seguintes opções:
  - Pode dar um código de uma paragem específica ($\log|V|$);
  - Pode dar coordenadas e escolher ou a paragem mais próxima desse ponto ou escolher de uma lista de paragens num raio escolhido por ele ($|V|\log|V|$).

De forma a obter as paragens mais próximas e dentro de um raio das coordenadas, foi utilizada a função de Haversine.

V = paragens no grafo

Funcionalidades - Caminhos

Para escolher o melhor caminho implementámos duas formas, ambas podem ser usadas nos 4 grafos:

- Menor número de paragens;
  - Utiliza Pesquisa em Largura (BFS) para obter o caminho e o número de paragens
  - Complexidade Temporal: O(|E|)
- Menor distância.
  - Utiliza o Algoritmo de Dijkstra para obter o caminho e a distância a percorrer.
  - Complexidade Temporal: O(|E|log|V|)

Existem também funções que retornam a distância e o caminho para cada opção, com a mesma complexidade temporal que o respetivo algoritmo.

E = arestas (paragem -> paragem) no grafo

V = paragens no grafo

```
void Graph::bfs(int v) {
    for (int v=1; v<=n; v++) {
        nodes[v].visited = false;
        nodes[v].dist = DBL_MAX;
        nodes[v].pred = -1;
    }
    queue<int> q; // queue of unvisited nodes
    q.push( x: v);
    nodes[v].dist = 0;
    nodes[v].pred = v;
    nodes[v]. visited = true;
    while (!q.empty()) { // while there are still unvisited nodes
        int u = q.front(); q.pop();
        //cout << u << " "; // show node order
        for (auto e :Edge  : nodes[u].adj) {
            int w = e.dest;
            if (!nodes[w].visited) {
                q.push( x: w);
                nodes[w].visited = true;
                nodes[w].dist = nodes[u].dist+1;
                nodes[w].pred = u;
            }
        }
    }
}
```

```
void Graph::dijkstra(int a) {
    for (int v=1; v<=n; v++){
        nodes[v].visited = false;
        nodes[v].dist = DBL_MAX;
    }
    nodes[a].dist = 0;
    nodes[a].pred = a;

    MinHeap<int, double> heap(n,  notFound: -1);
    for(int i = 1; i <= n; i++) heap.insert( key: i,  value: nodes[i].dist);

    while(heap.getSize() > 0){
        int min = heap.removeMin();
        nodes[min].visited = true;
        if(nodes[min].dist == DBL_MAX) continue;

        for(const Edge& edge: nodes[min].adj){
            int dest = edge.dest;
            double weight = edge.weight;
            if(!nodes[dest].visited && nodes[min].dist + weight < nodes[dest].dist){
                nodes[dest].dist = nodes[min].dist + weight;
                nodes[dest].pred = min;
                heap.decreaseKey( key: dest,  value: nodes[dest].dist);
            }
        }
    }
}
```

Funcionalidades - Mudança de Autocarro

Para as mudanças de autocarro, o utilizador tem duas opções:

- Apanhar autocarro na mesma paragem onde sai;
- Poder andar entre paragens para apanhar outros autocarros. As paragens que estão a uma distância de 100 metros (walking distance) são calculadas usando a fórmula de haversine e são guardadas num grafo (pdGraph para linhas diurnas e pmGraph para linhas noturnas) para cada uma das paragens existentes.

  Complexidade Temporal quando chamada no início do programa: O(|V|^2 log|V|^2 log|E|)

No resultado final é mostrado, entre cada paragem, a linha a seguir ou a necessidade de andar a pé.

E = arestas (paragem -> paragem) no grafo

V = paragens no grafo

Interface do Utilizador

Escolha de Horário (Diurno/Noturno)

```
--- Escolha o Tipo de Horario ---
1 - Dia
2 - Noite

0 - Sair
```

Modalidade de Escolha de Paragens

```
--- Como Pretende Escolher a sua Viagem? ---
1 - Nomes das Paragens
2 - Coordenadas

0 - Sair
```

Input de Paragens

```
--- Indique o Codigo da Paragem de Origem ---
Origem:
```

```
--- Indique as Coordenadas da Latitude da Origem ---
Latitude:
```

```
--- Como Pretende Escolher a Paragem?
1 - Mais Proxima
2 - Escolher de uma Lista num Certo Raio
```

Interface do Utilizador

Possibilidade de percorrer distâncias entre paragens a pé (num máximo de 100 metros)

```
 --- Pretende poder andar a pe entre paragens? (Max 100m)
1 - Sim
2 - Nao
```

Escolha de preferência de “melhor” caminho para o utilizador

```
--- Qual o tipo de caminho que deseja?
1 - Menos paragens
2 - Menor distancia
```

Exemplo de Output

```
Numero de Paragens: 16
ASP4 | ASPRELA | PRT3
   ||
   || 305 | 305 - CORDOARIA-HOSPITAL DE S.JO├âO
   ||
HSJ10 | HOSP. S. JO├âO (CIRCUNVALA├ç├âO) | PRT3
   ||
   || 305 | 305 - CORDOARIA-HOSPITAL DE S.JO├âO
   ||
IPO4 | IPO (CIRCUNVAL.) | PRT3
   ||
   || 300 | 300 - CIRCULAR HOSPITAL DE S.JO├âO-ALIADOS
```

Destaque de Funcionalidade

Durante este projeto, tornou-se evidente que funcionalidades têm uma implementação evidentemente superior em relação às demais. Consequentemente, esta foi uma das funcionalidades que mais se destacou, para nós:

Escolha e Desenho de Melhor Caminho

Dado a opção é possível obter o melhor caminho usando o algoritmo de Dijkstra ou BFS, independentemente do Grafo escolhido pelos parâmetros do utilizador.

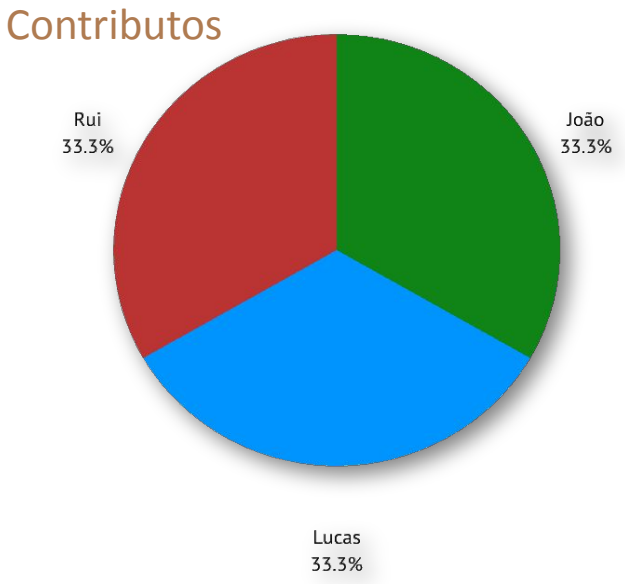
Como ambas as funções dijkstra_path(int a, int b) e bfs_path(int a, int b) criam uma lista de IDs de paragens, em ordem, é possível converter e retornar uma lista de Semipaths, onde é armazenado as informações da Paragem e da Linha a seguir, de forma a mais facilmente mostrar o resultado.

Complexidade Temporal: dijkstra_path(int a, int b): O(|E|log|V|)

bfs_path(int a, int b): O(|E|)

Dificuldades Encontradas e Contributos

- Falta de tempo
- Correr o debug, devido aos grafos serem de grande tamanho
- Primeira vez a usar grafos num contexto mais complexo



FIM


e é aqui que eu ponho os metros e comboios
SE OS TIVESSE
imgflip.com


Sempre foi
Espera, é tudo grafos?
imgflip.com