U.PORTO

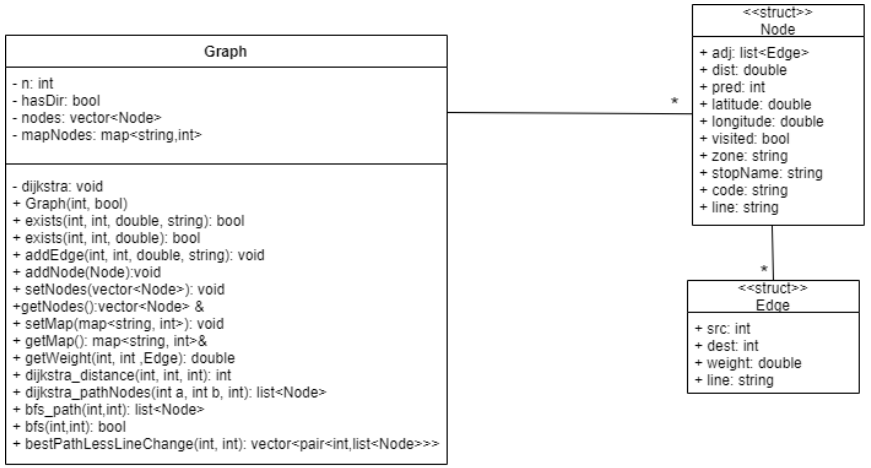
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

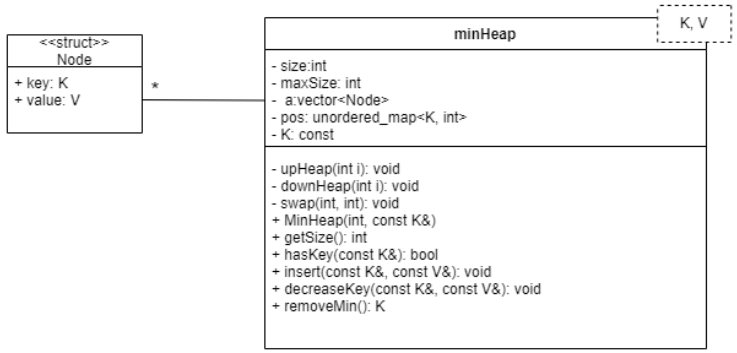
NAVEGAÇÃO NOS TRANSPORTES PÚBLICOS DO PORTO

PROJETO 2 - AED

Hugo Almeida up202006814@edu.fe.up.pt
Sara Reis up202005388@edu.fe.up.pt
Teresa Ferreira up202004852@edu.fe.up.pt
G74, Turma 7

DIAGRAMA DE CLASSES





GRAFOS UTILIZADOS

Umas das funcionalidades implementadas pelo nosso grupo foi a possibilidade do utilizador escolher se pretende pesquisar rotas por apenas linhas diurnas, noturnas ou ambas.

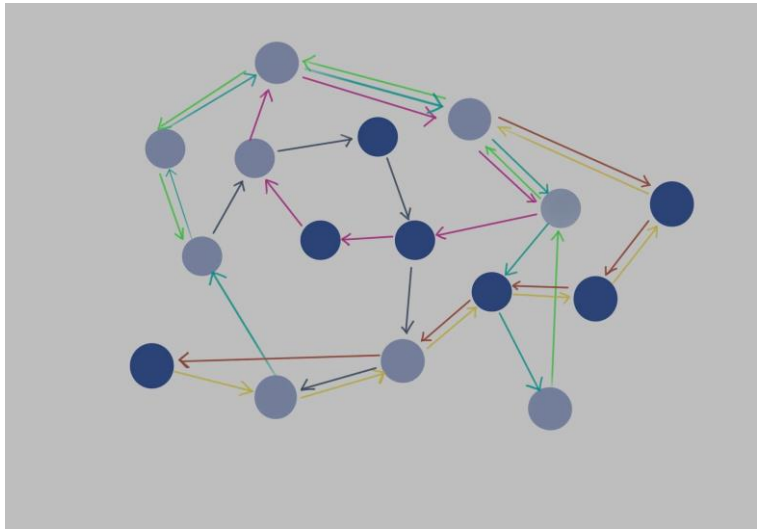
Assim, achamos pertinente a criação de três grafos, representativos de cada uma destas três opções:

- **graph** – grafo que contém todas as estações
- **graphN** – grafo que contém as estações das linhas noturnas
- **graphD** – grafo que contém as estações das linhas diurnas

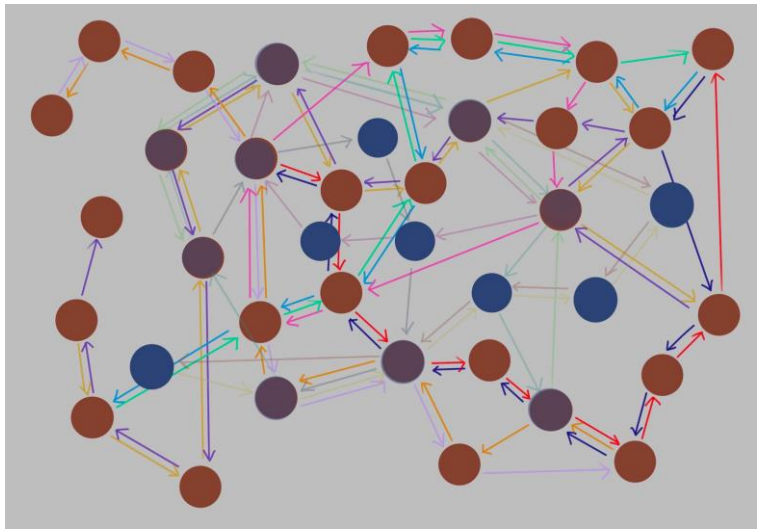
Estes três grafos foram criados para evitar a pesquisa por todas as estações existentes e ter que verificar se a linha presente da edge correspondia àquelas que o utilizador pretendia visualizar.

Note-se que criando os grafos logo de início, não precisamos de fazer leituras repetidas aos ficheiros.

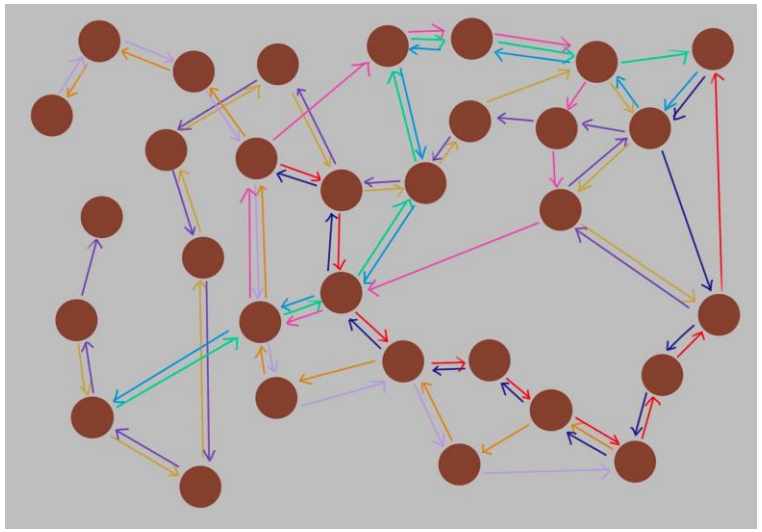
GRAFOS UTILIZADOS



graphN



graph



graphD

NOTA1: Cores diferentes implicam linhas e sentidos diferentes
NOTA2: Os grafos são meramente ilustrativos, visto que não têm as dimensões reais

LEITURA DO DATASET

Para a leitura do dataset, começamos por ler o ficheiro 'stops.csv', guardando a informação de cada estação como um nó do grafo que inclui todas as estações.

De seguida, para criar os edges entre estes nós, lemos o ficheiro 'lines.csv'. Assim, para cada linha lida, abrimos os ficheiros com os dois sentidos correspondentes a essa linha e formamos as edges com a informação acerca das ligações existentes entre as paragens dessa linha. De facto, duas estações seguidas no ficheiro implicam que haja uma edge entre elas. Visto que cada linha tem dois sentidos, decidimos colocar as edges direcionadas. Note-se que nós utilizamos a distância entre elas como valor de defeito para o peso.

Ao mesmo tempo que criamos estas ligações, aproveitamos para preencher os grafos noturno e diurno. Caso o nome da linha possuísse o caracter 'M', adicionávamos os nós ao graphN. Caso contrário, adicionávamos ao graphD.

FUNCIONALIDADES IMPLEMENTADAS

Distância que pretende andar a pé

O utilizador escolhe no início a distância máxima que aceita andar a pé.
Esta distância vai ser utilizada para unir estações de diferentes linhas. Assim, **caso duas estações estejam a menos da distância selecionada, cria-se uma edge entre os dois nós.**
Desta forma, quando o utilizador for verificar o caminho entre duas estações, aparecerão opções onde ele terá que andar entre estações, caso tal seja a opção mais favorável.

Origem/Destino

O programa implementado permite ao utilizador inserir a sua origem e o seu destino tanto em forma de **coordenadas** como com o **código** de uma paragem.

Se a primeira for escolhida, serão mostradas as **cinco paragens mais próximas** das coordenadas inseridas, das quais o utilizador poderá selecionar uma paragem.

FUNCIONALIDADES IMPLEMENTADAS

Algoritmo Dijkstra - O(|E| log |V|), onde E=Edges, V=Nodes

Este método é utilizado para calcular o melhor caminho num grafo pesado. No nosso caso, usámo-lo em 3 situações:

- Melhor preço (passa por menos zonas)
- Menor distância
- Menor mudanças de linhas

Numa primeira fase, colocámos as variáveis *dist* a infinito e *visited* a false e inserimos todos os valores na minHeap com peso infinito.
De seguida, colocamos tanto a distância como o peso do nó inicial a zero e definimos que o seu *pred* seria ele próprio.
Até à minheap ficar vazia, retiramos o nó com menor peso, marcamo-lo como visitado e para cada uma das suas edges calculamos o seu peso (consoante o método escolhido). Caso o peso que estava na minheap para aquele nó fosse maior, trocámo-lo.
Assim, para qualquer nó, foi possível reconstruír o caminho com menor peso.

FUNCIONALIDADES IMPLEMENTADAS

Melhor preço

O utilizador pode escolher ver o caminho mais barato, ou seja, o caminho que passa por menos zonas.

Para determinar o peso utilizado entre os diferentes nós no algoritmo de Dijkstra, considerou-se o seguinte critério:
Se a estação que está a ser analisada localiza-se numa zona diferente da anterior, então o peso da edge corresponde 1. Caso a zona seja igual, o peso é 0.

Dijsktra

NOTA: o grafo utilizado depende da escolha do utilizador. Caso este prentenda ver o caminho com todas as linhas, utilizamos o graph, caso pretenda ver à noite usamos o graphN e caso pretenda ver de dia usamos o graphD.

FUNCIONALIDADES IMPLEMENTADAS

Menos mudanças de linhas

O utilizador pode escolher ver o caminho com menos mudanças de linhas.

Para determinar o peso utilizado entre os diferentes nós no algoritmo de Dijkstra, considerou-se o seguinte critério:
Se a estação que está a ser analizada tem uma linha diferente da anterior, então o peso da edge corresponde 1. Caso a linha seja igual, o peso é 0.

Dijsktra - O(|E| log |V|), onde E=Edges, V=Nodes
BestPathLessLineChange – guarda num set o nome das diferentes linhas; faz o disjkstra para cada uma destas linhas, tendo uma em conta como inicial; compara os pesos de cada caminho obtido; faz apenas display dos que têm menor peso **- O(|L| |E| log |V|)**, onde L=lines, E=Edges, V=Nodes

NOTA: o grafo utilizado depende da escolha do utilizador. Caso este prentenda ver o caminho com todas as linhas, utilizamos o graph, caso pretenda ver à noite usamos o graphN e caso pretenda ver de dia usamos o graphD.

NOTA_2: ir a pé até uma estação vizinha conta como uma mudança de linha.

FUNCIONALIDADES IMPLEMENTADAS

Menor distância

O utilizador pode escolher ver o caminho com menor distância.

Para determinar o peso utilizado entre os diferentes nós no algoritmo de Dijkstra, considerou-se o seguinte critério:
O peso corresponde à distância entre os dois nós que estão a ser ligados pela edge.

Dijsktra

NOTA: o grafo utilizado depende da escolha do utilizador. Caso este prentenda ver o caminho com todas as linhas, utilizamos o graph, caso pretenda ver à noite usamos o graphN e caso pretenda ver de dia usamos o graphD.

FUNCIONALIDADES IMPLEMENTADAS

Menos estações

O utilizador pode escolher ver o caminho que passa por menos estações.

Para calcular este caminho utilizamos BFS, visto que cada nó corresponde a uma estação.

BFS

NOTA: o grafo utilizado depende da escolha do utilizador. Caso este prentenda ver o caminho com todas as linhas, utilizamos o graph, caso pretenda ver à noite usamos o graphN e caso pretenda ver de dia usamos o graphD.

BFS - O(|V| + |E|), , onde E=Edges, V=Nodes**:**
Começamos por criar uma fila.
Colocamos todos os nós com *visited* igual a false e com *pred* igual a -1.
De seguida, colocamos o nó de origem como visitado e adicionamos a sua posição à fila.
Depois e enquanto a fila não estiver vazia, retiramos o primeiro elemento e para cada um dos seus edges verificamos se o nó encontrado já foi visitado. Caso ainda não tenha sido, adicionamos a sua posição à fila, colocamo-lo a visitado e colocamos o *pred* igual à posição do nó de onde parte a edge. Caso se tenha encontrado o destino final antes da fila ter ficado vazia, retorna-se true, pois encontrou-se o caminho mais pequeno.
Para recuperar o percurso, começa-se no nó final e verifica-se o *pred* e vai-se percorrendo o caminho até que se encontre um *pred* a -1 que corresponde ao nó da origem.

INTERFACE COM O UTILIZADOR

```
Search for:
[0] All lines (day and night)
[1] Day lines
[2] Night lines
0
What's your max walking distance (km) :
```

```
-------------MENU-------------
[1] Choose stops by their code
[2] Choose stops by coordinates
[0] Exit
```

```
---------Stops Codes---------
Source stop code ( 0 to go back):HSJ12
Destination stop code ( 0 to go back):SAL4
```

```
--------SRC COORDINATES-------
Source Latitude:41.17
Source Longitude:-8.6

Closest bus stops:
[1] SAL2 - SALGUEIROS 0.0618787Km
[2] SAL1 - SALGUEIROS 0.0955281Km
[3] AUL1 - AUGUSTO LESSA 0.205683Km
[4] AUL2 - AUGUSTO LESSA 0.206182Km
[5] OUTE3 - OUTEIRO 0.235674Km
[0] Go back

Choose one of the bus stops:2

-------DEST COORDINATES------
Destination Latitude:41.14
Destination Longitude:-8.66

Closest bus stops:
[1] DL1 - D. LEONOR 0.781965Km
[2] DL2 - D. LEONOR 0.808673Km
[3] DL6 - D.LEONOR 0.813135Km
[4] DL5 - D.LEONOR 0.823895Km
[5] PIC1 - PIC├âO 0.904913Km
[0] Go back

Choose one of the bus stops:1
```

INTERFACE COM O UTILIZADOR - CONTINUAÇÃO

```
-----------METHODS-----------
[1] Best price
[2] Less changes between lines
[3] Less bus stops
[4] Smallest distance
[0] Exit
```

```
------OPTION 1-------
You change lines 1 time(s)

Take the 301 - CIRCULAR HOSPITAL S.JO├âO-S├Ü DA BANDEIR line
HSJ12(PRT3)->IPO5(PRT3)->ST4(PRT3)->IB2(PRT3)->ISEP2(PRT3)->AVTE2(PRT3)->TLHR3(PRT3)->MIS4(PRT2)->BRA5(PRT2)->STLZ2(PRT2
)->PPR2(PRT2)->CI22(PRT1)->HPRL2(PRT1)->CVI2(PRT1)->QTAM2(PRT1)->CZV2(PRT1)->CMIC2(PRT1)->ICDF1(PRT1)->CDF2(PRT1)->TRH3(
PRT1)->MATR2(PRT1)->MGB2(PRT1)->HSA(PRT1)->CORD7(PRT1)->PRFL1(PRT1)->1AL6(PRT1)->SCAT1(PRT1)->PDR(PRT1)->C24A2(PRT1)->BF
M8(PRT1)->BL1(PRT1)->ESR1(PRT1)->SCRP5(PRT1)->MAV1(PRT1)->IANT4(PRT1)->DC2(PRT1)->SVT1(PRT1)->VDAM1(PRT1)->SAL3(PRT1)

Walk from SAL3(PRT1) to SAL4(PRT1)
The total price will be 1.6 euros
```

[2] Less changes between lines

```
Total distance traveled is 1 km

Walk from HSJ12(PRT3) to HSJ6(PRT3)

Take the 11M - HOSP. S. JO├âO - COIMBR├òES line
HSJ6(PRT3)->ESED2(PRT3)->FEUP2(PRT3)->FEP1(PRT3)

Take the 204 - HOSPITAL DE S.JO├âO - FOZ line
FEP1(PRT3)->MLAR2(PRT3)->OUTE4(PRT1)

Take the 300 - CIRCULAR HOSPITAL DE S.JO├âO-ALIADOS line
OUTE4(PRT1)->SAL4(PRT1)
The total price will be 1.25 euros
```

[4] Smallest Distance

DESTAQUE DE FUNCIONALIDADE

Neste projeto, em vez de tentarmos adicionar uma funcionalidade extra achamos crucial valorizar mais certos detalhes. Note-se:

- Em vez de criarmos várias funções Dijkstra, criamos um método (graph.getWeight()) que seleciona o peso da edge tendo em conta o método escolhido para melhor caminho. Assim, evitamos a repetição de código e tornámo-lo mais extensível, respeitando o Open–closed principle.
- Utilizamos três grafos diferentes que nos permitem fazer uma pesquisa dos caminhos mais seletiva tendo em conta o horário pretendido, sem a necessidade de impormos restrições, tendo apenas de percorrer paragens ativas no horário selecionado.
- O utilizador pode, no menu, voltar atrás em qualquer momento.
- O utilizador pode no ínício escolher uma distância máxima que pretende andar – não nos limitamos a fixar um valor para todos, visto que é algo que depende muito de pessoa para pessoa.

DIFICULDADES E ESFORÇO

Gostaríamos de ter tido mais algum tempo para conseguirmos implementar mais funcionalidades e para pensarmos noutras formas para construírmos os grafos.

Hugo Almeida 33%

Sara Reis 33%

Teresa Ferreira 33%

TAREFAS DE VALORIZAÇÃO

Linhas diurnas e noturnas

Neste trabalho, permitimos que o utilizador selecione viagens diurnas, noturnas (linhas com sufixo M) ou veja caminhos com todas as estações.

Tal como já foi referido, após esta decisão ser tomada, utiliza-se o grafo que mais se adequa. Caso queira ver viagens noturnas utiliza-se o *graphN*, caso queira ver viagens diurnas utiliza-se o *graphD*, caso queira ver com todas as estações utiliza-se o *graph*.