U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

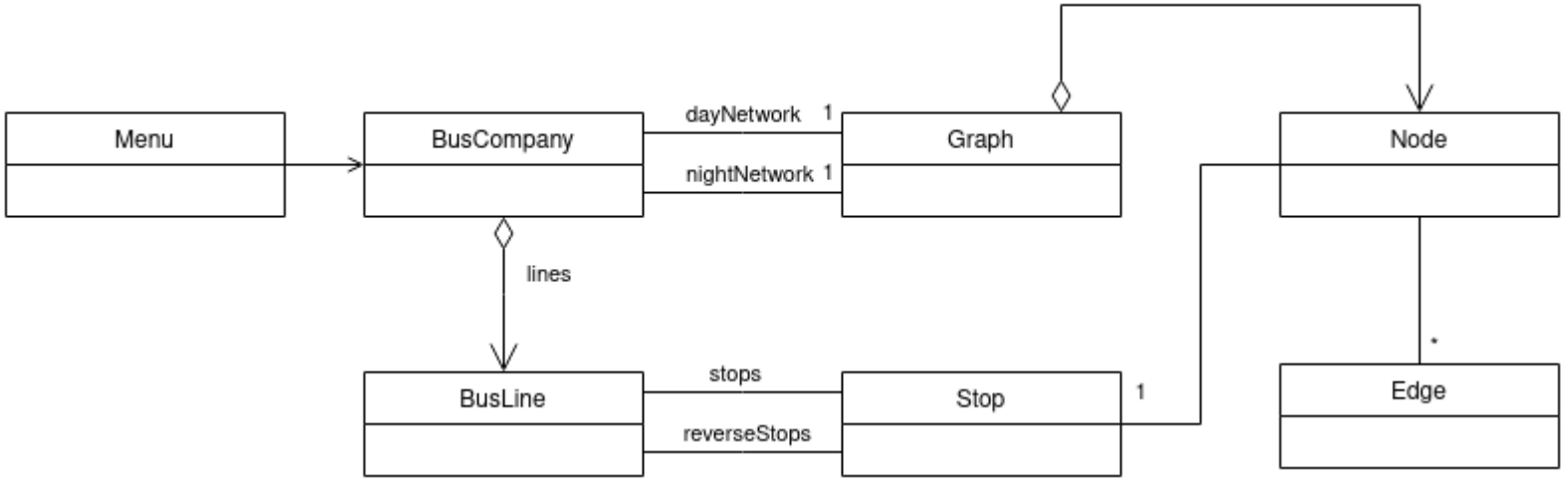
Algoritmos e Estruturas de Dados

Navegação nos transportes públicos do Porto

Trabalho realizado por:

Nuno Afonso Anjos Pereira - up202007865
Pedro Miguel Magalhães Nunes – up202004714
Vitor Manuel da Silva Cavaleiro – up202004724

Diagrama das classes



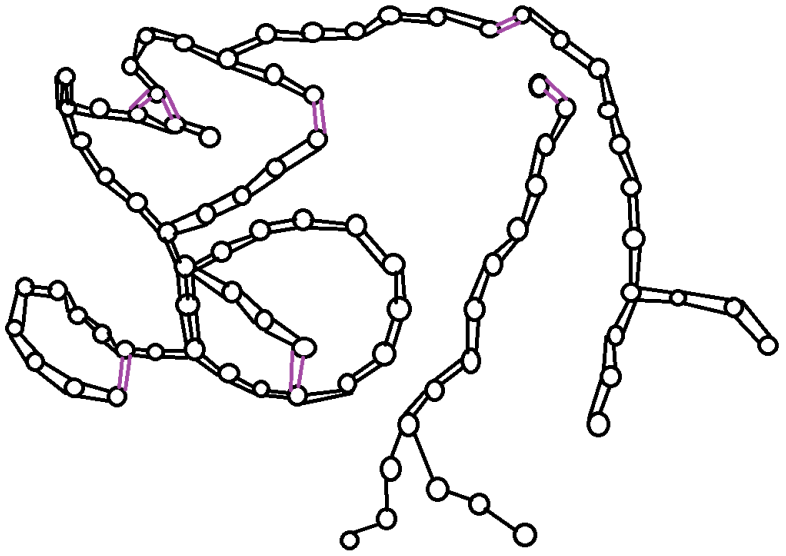
Leitura do dataset

- Os ficheiros .csv que compõem o dataset devem ser colocados num diretório “resources”;
- A leitura do dataset é realizada no construtor da classe BusCompany;
- A leitura dos dados é feita a partir da ferramenta utils::file::readFile que retorna um vetor de strings(cada elemento é uma linha do ficheiro);
- A classe Stop possui um método responsável por processar a string da paragem(parseLine), tal como a classe BusLine no caso da string da linha;
- A informação da paragem é carregada num objeto da classe Stop(informação usada para criar os nodes);
- Os dados de uma aresta são carregados num objeto da classe BusLine (informação usada para criar as edges).

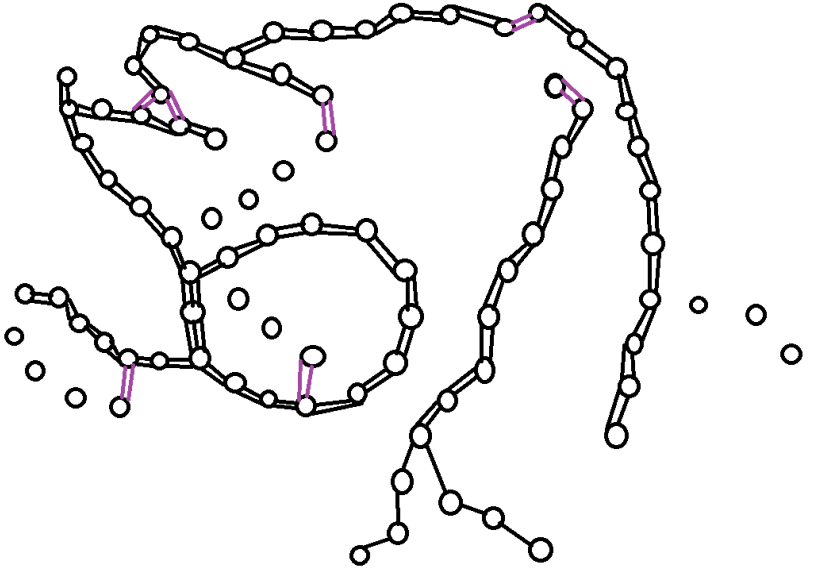
Representação do dataset

- O dataset é caracterizado por dois grafos direcionados, um para a rede noturna e outro para a rede diurna, na classe BusCompany;
- Os vértices dos grafos estão representados por um unordered_map, onde a chave corresponde ao código da paragem e o valor corresponde ao node(struct Node);
- Cada nó guarda um apontador para uma paragem, a lista de arestas, uma variável booleana visited e outras variáveis usadas nos algoritmos;
- As arestas dispõem de uma variável que guarda o destino(código da paragem), a distância e um código que identifica a linha de autocarro usada. Foram adicionadas algumas arestas que representam o caminho pedestre até uma paragem, dentro de uma distância pré-definida.

Grafo diurno



Grafo noturno



Funcionalidades e algoritmos

- O algoritmo de dijkstra é implementado com o recurso a red-black trees(set), se assumirmos que o número de arestas é superior ao número de vértices, a complexidade é de O(|E| log |V|);
- Nota: O algoritmo calcula as distâncias e caminhos para todos os nós, no entanto, de maneira a deixar o SSSP completo, não cessa o processo assim que chega ao destino.
- Os algoritmos BFS implementados são baseados no pseudocódigo apresentado na aula téorica, pelo que apresentam complexidade O(|E| + |V|);
- Na computação do “melhor caminho” admitimos que a origem e o destino são paragens;
- Também está implementado o método nearbyStops que retorna as paragens mais próximas de uma coordenada(pode ser fornecida pelo utilizador ou pode ser uma paragem). Apresenta complexidade O(|V|);

Funcionalidades e algoritmos

- “Melhor” caminho por distância percorrida:
- Método minDistance(devolve a menor distância), executa o algoritmo de dijkstra. Apresenta complexidade de O(|E| log |V|);
- Método minDistancePath(devolve o trajeto, com menor distância, ao longo dos vértices), executa o algoritmo de dijkstra e monta o caminho a partir do vértice final(uso de apontadores para o vértice anterior). Apresenta complexidade de O(|E| log |V|).

Funcionalidades e algoritmos

- “Melhor” caminho por número de paragens percorridas:
- Método minStops(devolve o menor número de paragens percorridas), aplica o algoritmo BFS. Apresenta complexidade de O(|E| + |V|);
- Método minStopsPath(devolve o trajeto, com menor número de paragens percorridas, ao longo dos vértices), aplica o algoritmo BFS e monta o caminho a partir do vértice final(uso de apontadores para o vértice anterior) . Apresenta complexidade de O(|E| + |V|).

Funcionalidades e algoritmos

- “Melhor” caminho por número de zonas percorridas:
- Método minZones(devolve o menor número de zonas percorridas), aplica o algoritmo de dijkstra. Apresenta complexidade de O(|E| log |V|);
- Método minZonesPath(devolve o trajeto, com menor número de zonas percorridas, ao longo dos vértices), aplica o algoritmo de dijkstra e monta o caminho a partir do vértice final(uso de apontadores para o vértice anterior) . Apresenta complexidade de O(|E| log |V|).

Funcionalidades e algoritmos

- No cálculo do “melhor” caminho, o processo de computação conta com a mudança de autocarro e também trajeto pedestre, de uma paragem a outra, se estas estiverem a uma distância pré-definida uma da outra (o cliente pode escolher);
- Está implementado uma função(travelPossibleTicket) que consegue determinar se o andante que o cliente possui, tem zonas suficientes para cumprir o trajeto que o cliente deseja;

Interface

- O menu do utilizador inclui do ponto de vista do cliente 10 funcionalidades:
- Listar as paragens;
- Listar as linhas de autocarro;
- Mostrar o caminho com menor distância percorrida(a origem e o destino podem ser paragens ou coordenadas);
- Mostrar o caminho com menos paragem percorridas(a origem e o destino podem ser paragens ou coordenadas);
- Mostrar o caminho com menos zonas percorridas(a origem e o destino podem ser paragens ou coordenadas);
- Confirmar se o título do cliente tem zonas suficientes para o caminho pretende percorrer;
- Alterar a distância que o cliente está disposto a andar.

Interface

- O menu de administrador inclui do ponto de vista de um funcionário 2 funcionalidades:
- Tornar ativa/inativa uma paragem;
- Tornar ativa/inativa uma linha;

Interface

- Exemplo de utilização:

```
        Welcome to STCP

 [1] Client
 [2] Admin
 [3] Exit

 >1
Hello client, what would you like to do?

 [1] List all stops
 [2] List lines (and check a line)
 [3] Find the shortest path for your travel
 [4] Find the shortest path for your travel (coordinates)
 [5] Find the path with less stops for your travel
 [6] Find the path with less stops for your travel (coordinates)
 [7] Find the cheapest path (less zones) for your travel
 [8] Find the cheapest path (less zones) for your travel (coordinates)
 [9] Can you travel with certain ticket?
 [10] Set your max walking distance
 [11] Back

 >
```

```
Do you plan on travelling during the day (D/d) or during the night (N/n)?d

 Please indicate the origin stop's code:HSJ3

 Please indicate the destiny stop's code:TRD6

 Trip's number of stops: 17
 Go from HSJ3 - HOSP. S. JO├âO (METRO) to IPO2 - INST. ONCOLOGIA on foot.
 Go from IPO2 - INST. ONCOLOGIA to ISEP4 - ISEP/AGRA following line 803
 Go from ISEP4 - ISEP/AGRA to CVLO2 - COVELO following line 603
 Go from CVLO2 - COVELO to BJ1 - BENTO J├ÚNIOR following line 204
 Go from BJ1 - BENTO J├ÚNIOR to MPL4 - MARQU├èS following line 603
 Go from MPL4 - MARQU├èS to BLRB3 - BOLH├âO following line 701
 Go from BLRB3 - BOLH├âO to BLRB1 - BOLH├âO on foot.
 Go from BLRB1 - BOLH├âO to TRD6 - TRINDADE following line 200

 [press ENTER to continue]
```

Destaques de funcionalidades

Cálculo em tempo de execução de edges que representam caminhos a pé, permite mudar dinamicamente o grafo de acordo com a vontade do utilizador.

Principais dificuldades

- Representação do dataset no grafo, especialmente na implementação das edges;

Valorizações extra

- Como proposto, implementámos viagens diurnas/noturnas;
- Implementamos também a possibilidade de fechar uma paragem/linha, bem como torná-la ativa de novo;