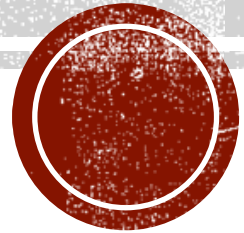
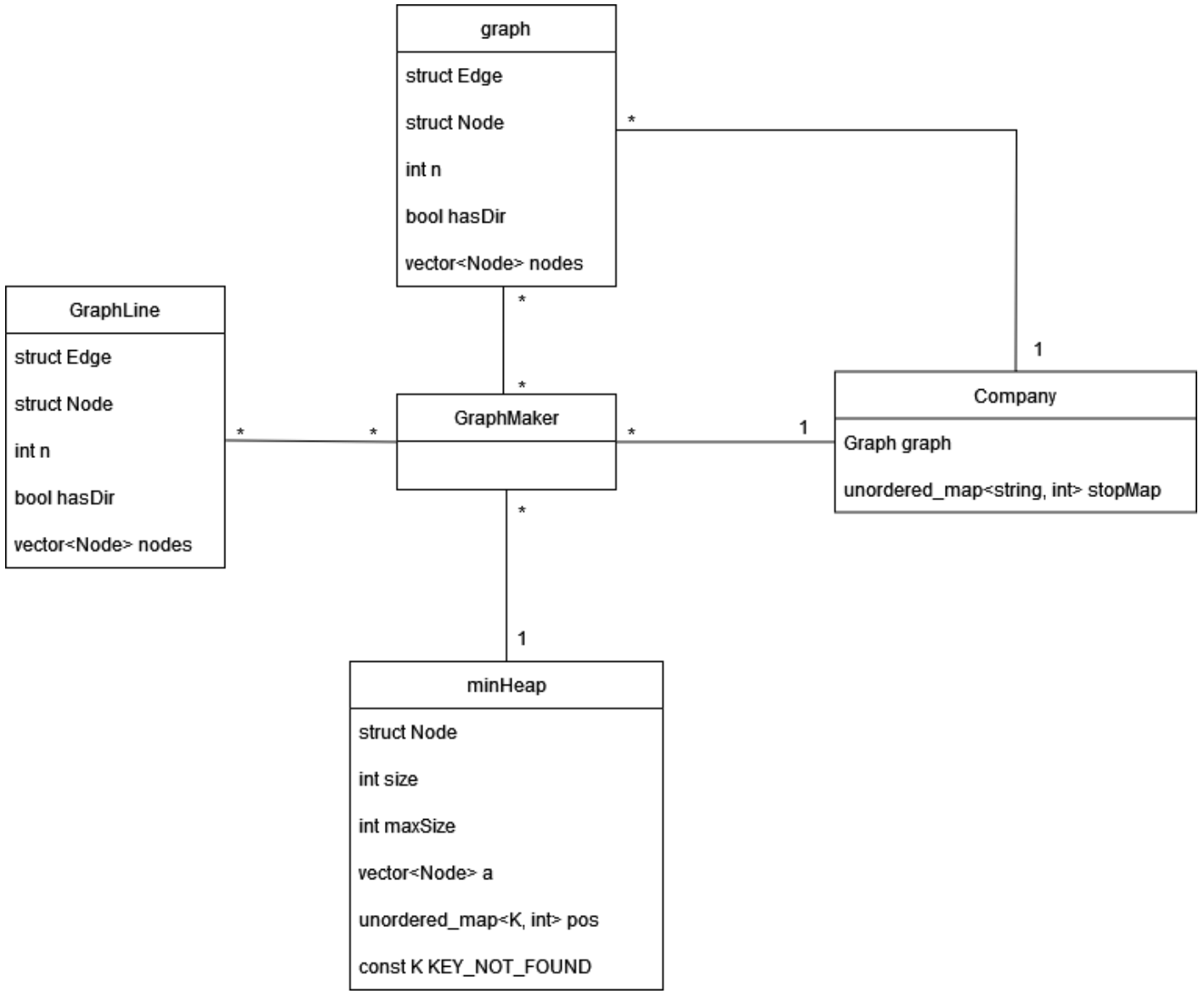
FORMAS EFICIENTES DE DESLOCAÇÃO NA REDE STCP



José Albano de Almeida Gaspar, up202008561
Maria Eduarda Pacheco Mendes Araújo, up202004473
Miguel Ângelo Silva Teixeira, up202005208

G 42

DIAGRAMA DE CLASSES

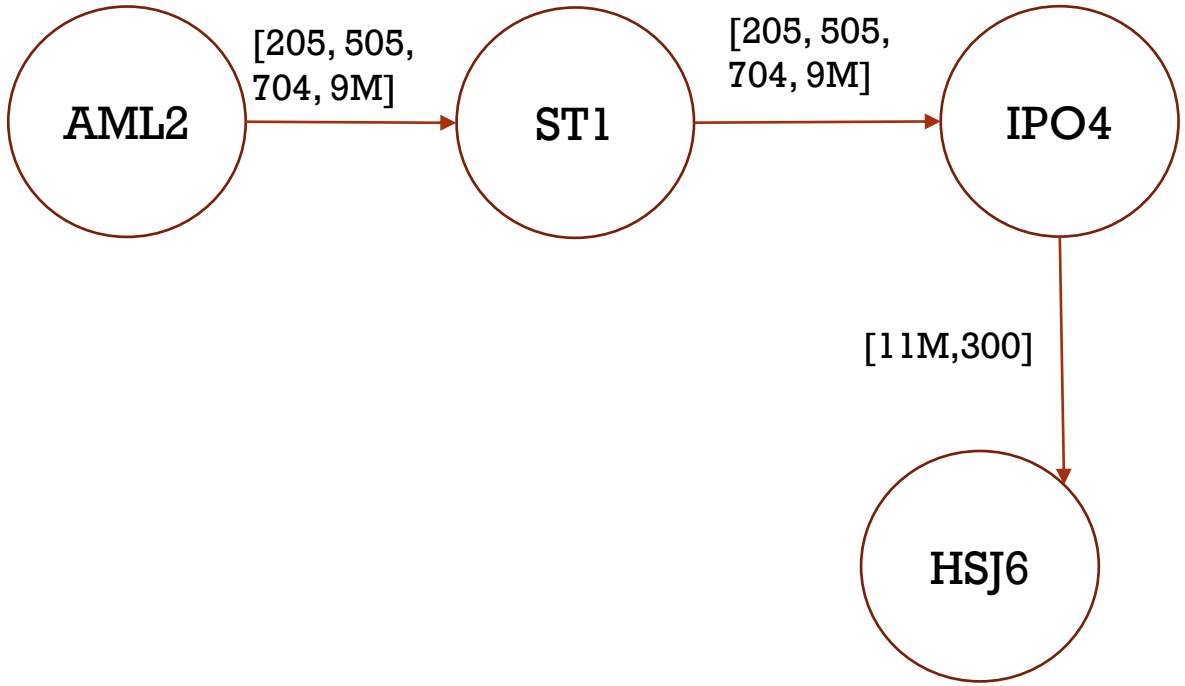


LEITURA DO DATASET

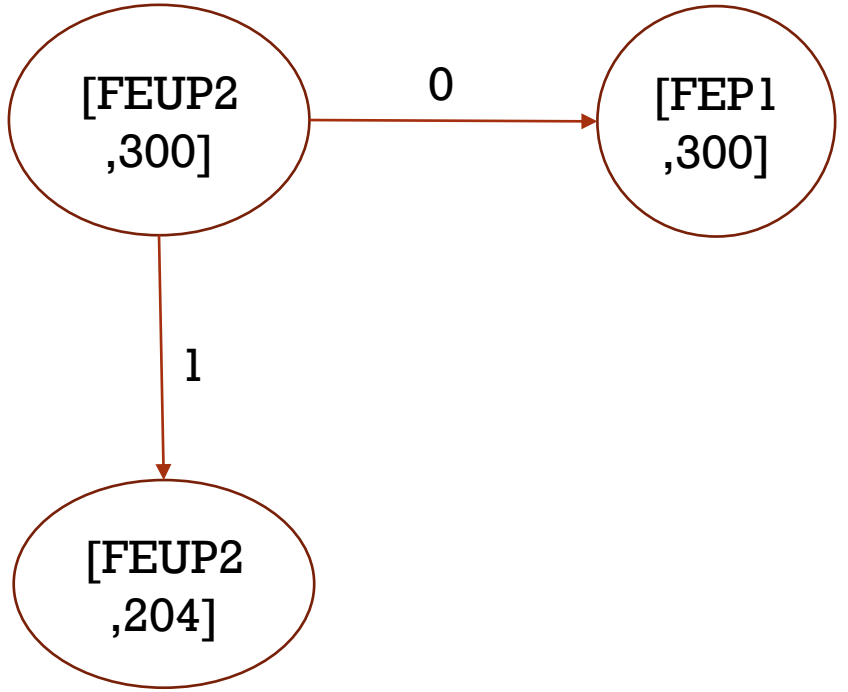
- General Graph
  - Leitura do ficheiro "stops.txt" para conseguir armazenar num vetor auxiliar toda a informação relativa às paragens que são os nós do grafo.
  - Leitura do ficheiro "lines.txt" para obtenção dos códigos das paragens.
  - A partir dos códigos anteriores foram lidos os restantes ficheiros, criando as arestas do grafo entre duas estações consecutivas.
  - Em casos de múltiplas linhas passarem pela mesma aresta, os seus códigos eram adicionados numa lista ao invés de ser criada uma nova aresta
- Line Graph
  - Leitura do ficheiro "lines.txt" para obter o número de nós do grafo (os nós seriam todos os pares (paragem, linha).
  - Leitura dos ficheiros das paragens por linha para adicionar entre esses nós consecutivos arestas de peso 0.
  - Adição das arestas de peso 1 para as ligações entre nós com a mesma paragem e linhas diferentes.

DESCRIÇÃO DOS GRAFOS UTILIZADOS

GENERAL GRAPH



LINE GRAPH



DESCRIÇÃO DOS GRAFOS UTILIZADOS

GENERAL GRAPH

```
class Graph {
    struct Edge {
        int dest;   // Destination
        double weight; // A double
        int changeZone;
        list<string> lineCode;
    };

    struct Node {
        list<Edge> adj; // The list
        int parent;
        double distance;
        bool visited;
        //proprieties of STCP stops
        string code;
        string name;
        string zone;
        double latitiude;
        double longitude;
    };
```

LINE GRAPH

```
class GraphLine {
    struct Edge {
        int dest;   // Destination
        int changeLine; //weight
    };

    struct Node {
        list<Edge> adj; // The list
        int parent;
        int distance;
        bool visited;
        string stop;
        string line;
    };
```

FEATURES

- Caminho mais curto entre duas paragens
  - Algoritmo
    - Algoritmo de Dijkstra
  - Complexidade temporal
    - O(|E|log |V|)
  - Grafos utilizados:
    - General Graph
- Caminho que passa por menos paragens
  - Algoritmo
    - BFS
  - Complexidade temporal
    - O(|E| + |V|)
  - Grafos utilizados:
    - General Graph

FEATURES

- Caminho que passa por menos zonas
  - Algoritmo
    - Algoritmo de Dijkstra
    - (o peso representa a mudança de zona)
  - Complexidade temporal
    - O(|E|log |V|)
  - Grafos utilizados:
    - General Graph

- Caminho mais próximo + possibilidade de ir a pé entre paragens próximas
  - Algoritmo
    - Adicionar "walking edges"
    - Aplicar Dijkstra sobre esse grafo
  - Complexidade temporal
    - O(|V|^2) → devido à inserção das edges extra
  - Grafos utilizados
    - General Graph modificado

FEATURES

- Caminho que passa por menos linhas
  - Algoritmo
    - Algoritmo de Dijkstra
    - (o peso simboliza as mudanças de linha)
  - Complexidade temporal
    - O(|E| log |V|)
  - Grafos utilizados
    - LineGraph

- Definição de local:
  - Estações específicas
    - Algoritmo
      - Pedir duas estações e prosseguir com os algoritmos descritos anteriormente
    - Complexidade temporal
      - Igual à do algoritmo usado
  - Locais aleatórios
    - Algoritmo
      - Pedir 2 pontos por latitude/longitude
      - Calcular os nodes a menos de x metros dos locais
      - Para cada par de potenciais nós iniciais e finais usar um dos algoritmos descritos e ver o melhor
    - Complexidade
      - O(|V|^2)

DESCRIÇÃO DA INTERFACE COM O UTILIZADOR

- Foi construído um menu simples e de rápida utilização para interação com o utilizador.
- Primeiramente, é mostrado ao utilizador um menu principal, onde é dada a opção de escolha de inserção de dados, ou pelo sistema de coordenadas ou por inserção direta através de estações.

```
STCP PROJECT
stations  - s ||  position - p:
Press q to exit
```

- De seguida, avançamos para os menus das ações disponíveis. Tanto o menu de ações do sistemas de coordenadas, quer o menu de estações têm as mesmas funções, diferindo apenas no tratamento dos dados de entrada.
- Estando disponíveis aqui diversas funcionalidades:

```
1 - Shortest path (by distance)
2 - Shortest path (by number of stops)
3 - Cheapest path (less zones)
4 - Most Convenient path (less line's exchange)
5 - Bus + Walking path
6 - Exit
```

EXEMPLOS DE OUTPUT

CAMINHO QUE PASSA POR MENOS LINHAS

```
Station code:FEUP2
Station code:OUTE4
 --(204)-> FEUP2 --(204)-> FEP1 --(204)-> MLAR2 --(204)-> OUTE4
```

RESTANTES FEATURES

```
Station code:AML2
Station code:FEUP2
 --> AML2 --> ST1 --> IPO4 --> HSJ6 --> ESED2 --> FEUP2
205 505 704 9M
205 505 704 9M
11M 300
11M 300
11M 204 300
```

BEST FEATURE
CAMINHO QUE PASSA POR MENOS LINHAS

- Criação de um novo grafo (classe GraphLine) que possui como nós todos os pares (paragem , linha).
- As ligações (arestas) deste grafo têm peso 0, se 2 nós tiverem paragens diferentes e a mesma linha, e peso 1, se tiverem paragens iguais e linhas diferentes, simbolizando assim estas arestas uma mudança de linha.
- Cálculo do caminho : Utilização do algoritmo de Dijkstra (o peso corresponde às mudanças de linha).

PRINCIPAIS DIFICULDADES ENCONTRADAS

- Criação de grafos a partir do dataset
- Eficiência de alguns métodos implementados
- Caminho que tem menos trocas de linhas

FUTURAS IMPLEMENTAÇÕES

- Viagens de noite / dia
- Horários
- Cálculo de MSTs