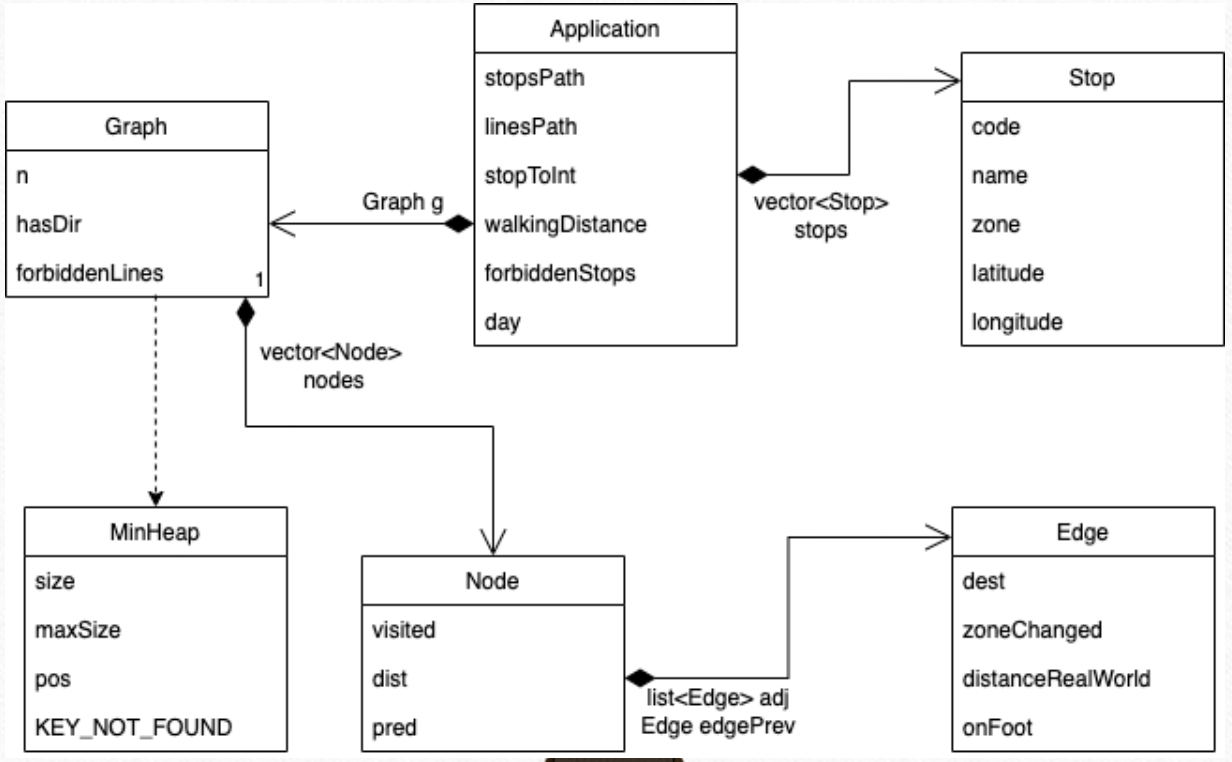
STCP Routing System

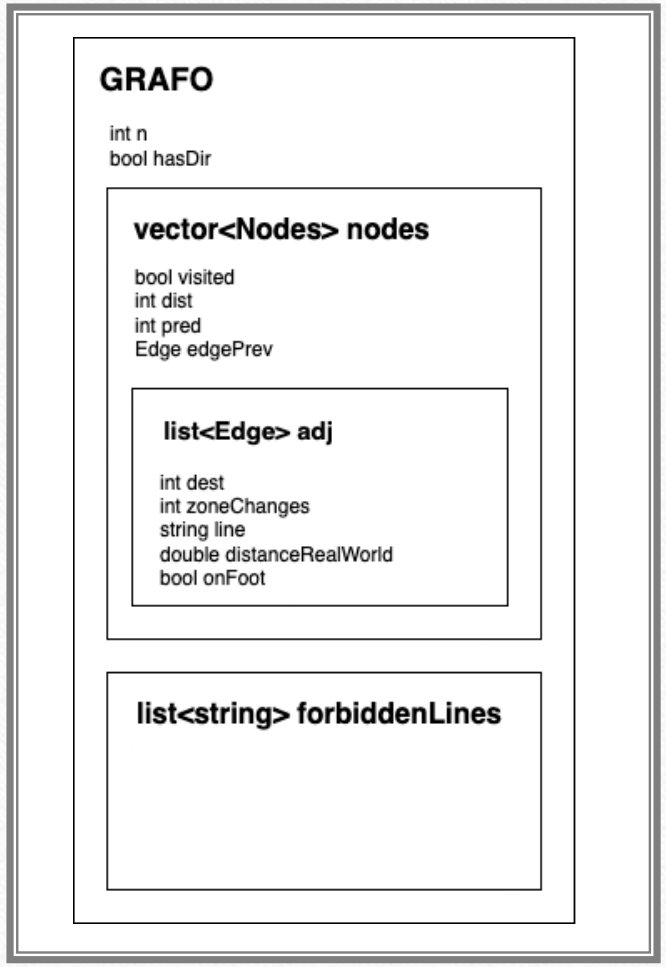
André Sousa – up202005277
João Félix – up202008867
Pedro Fonseca – up202008307
Grupo 65

Diagrama de Classes



Leitura do Dataset

- A função *readStops()* lê a informação das paragens de autocarro contidas no ficheiro stops.csv e guarda-os num vetor denominado "stops";
- A função *readEdges()* lê as linhas do ficheiro lines.csv e depois corre o método *addEdges()*.
- No *addEdges()*, são lidas as paragens em cada linha através dos ficheiros line_xxx_0x.csv (em ambas as direções) e acrescenta as *Edges* entre estas e guarda-as em cada nó, numa lista de *Nodes*.
- De acordo com o input do utilizador, o programa seleciona apenas as linhas noturnas ou diurnas. Para além disso, também ignora as paragens e linhas que o utilizador pretende não usar
- Por fim, são criadas arestas novas entre paragens cuja distância é igual ou inferior à distância que o utilizador está disposto a andar a pé



Grafos usados para representar o dataset

- É usada a classe Graph fornecida
- **Atributos:**
  - n – tamanho do grafo
  - hasDir – booleano de direção
  - nodes – vetor que guarda Node structs
  - forbiddenLines: lista com os códigos das linhas que devem ser ignoradas
  - **Node Struct:**
    - visited – booleano que define se já foi visitado
    - dist – double, estimativa da distância
    - pred – nó anterior, se existir
    - adj – list das arestas que "saem" do nó

Edge Struct:

- dest – node destino
- zoneChanges – indicativo de mudança de zona
- line – nome da linha
- distanceRealWorld – distância entre paragens
- onFoot – true se caminhar a pé

Funcionalidades implementadas e algoritmos associados

Origem/Destino:

- O nosso programa permite os seguintes inputs:
  - Código da paragem de partida para código da paragem de chegada;
  - Coordenada de partida para coordenada de chegada;
  - Código da paragem de partida para coordenada de chegada;
  - Coordenada de partida para código da paragem de chegada;

Funcionalidades implementadas e algoritmos associados

Conceito de melhor caminho:

- O nosso programa implementa os seguintes percursos de viagem:
  - Mais barato, ou seja, percurso que passa por menos zonas; ($O(|V|^2 * |E| * \log|V|)$);
  - Percurso mais rápido, ou seja, percorre menos distância; ($O(|V|^2 * |E| * \log|V|)$);
  - Percurso que implica menos mudanças de autocarro (de linha); ($O(|V|^2 * |E| * |V| * \log|V|)$);
  - Percurso com menos paragens. ($O(|V| + |E|)$).

Funcionalidades implementadas e algoritmos associados

Mudança de autocarro:

- O nosso algoritmo permite:
  - Mudança de autocarro na mesma paragem;
  - Andar a pé entre qualquer paragem, desde que, esteja dentro da distância máxima permitida a pé.

Extras:

- O nosso algoritmo permite ainda:
  - Escolher percursos noturnos ou diurnos
  - Escolher não passar por certas paragens e não usar certas linhas

Interface com o utilizador

- Em primeiro lugar, é exibida uma mensagem introdutória na consola onde o utilizador escolhe se quer viagens noturnas ou diurnas, as paragens e linhas que quer evitar e escolhe se quer calcular a MST do grafo, a partir de uma certa paragem. De seguida, escolhe o seu local de partida e chegada.
- Para escolher os locais o utilizador pode fazê-lo usando coordenadas-paragem, coordenadas-coordenadas, paragem-coordenadas ou paragem-paragem.
- Quando são utilizadas coordenadas, o programa procura todas as paragens que são alcançáveis a pé e procura a melhor opção. Quando não existirem paragens nesse raio, então é usada a paragem mais próxima.
- De seguida são apresentadas 4 opções, previamente explicadas, para realizar a viagem.

Destaque de funcionalidade

Uma característica do nosso projeto, que achamos que vale a pena destacar, é o facto de termos toda a informação fornecida contida em 1 único grafo. Esta estratégia permitiu-nos simplificar os algoritmos utilizados, visto que para calcular as diferentes versões de "melhor percurso" só é preciso adicionar um atributo novo à aresta do grafo (na maior parte dos casos) e alterar ligeiramente a implementação do algoritmo de Dijkstra, de forma a usar o atributo recém inserido.

Para além disso, estando a informação toda no mesmo sítio, o acesso a ela é extremamente simples e intuitivo.

Por fim, ao usar vários grafos podíamos cometer o erro de começarmos a repetir informação desnecessariamente e a interação entre os mesmos seria bem mais complexa do que a complexidade obtida utilizando apenas 1 grafo.

Principais dificuldades encontradas

Criação de um interface que seja amigo do utilizador e fácil de usar

Esforço de cada membro:

André Sousa: 33%

João Félix: 33%

Pedro Fonseca: 33%