Smart Delivery

EMPRESA DE DISTRIBUIÇÃO DE MERCADORIAS

2MIEIC07

Gustavo Nunes Ribeiro de Magalhães – 201705072 – up201705072@fe.up.pt
Luís Rafael Fernandes Mendes Afonso – 201406189 - up201406189@icbas.up.pt
Miguel Maio Romariz – 201708809 - up201708809@fe.up.pt

Conceção e Análise de Algoritmos

Abril 2019

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Índice

Introdução - Descrição do Tema

O código a desenvolver na segunda parte prática deste trabalho no âmbito da unidade curricular de conceção e análise de algoritmos, tem como objetivo a organização, manipulação, alocação e pesquisa de dados referentes a uma empresa distribuidora de mercadorias hipotética por recurso a grafos.

Ignorando detalhes técnicos, é importante considerar primeiro o funcionamento real deste tipo de empresas. Ignorando a situação económica e envergadura espacial da empresa, consideraremos os detalhes mais importantes de cada.

Primeiramente a empresa terá uma frota de camiões ou mais, caso agrupadas por zonas ou tipo de mercadoria, onde ao fim de um dia de entregas deverão todos estes veículos regressar a garagem da empresa ou a sítios específicos declarados pela própria empresa.

Por conseguinte o foco principal deste trabalho é a melhor gestão possível de rotas à empresa por métodos computacionais e automatizados. Todos os itens a distribuir deverão ser organizados antecipadamente, a saber-se quais poderão ser entregues, mas essencialmente para que sejam agrupados por zonas de entregas para que os camiões consigam fazer os trajetos mais otimizados possíveis. Neste aspeto será também importante considerar pesos e volume tanto de objetos como de camiões, para que seja feita a dada acomodação do objeto.

Num plano já mais avançado criar-se-ão também soluções a quando de problemas externos, como interrupção de vias.

Este problema pode e será, portanto, decomposto em três iterações:

- Iteração 1: considerando que a empresa apenas possui um camião, simplificando e muito o problema, tem como por objetivo também, um primeiro teste aos algoritmos principais de escrita de rotas. Onde todos os itens terão de ser entregues no menor trajeto possível.
- Iteração 2: nesta, a empresa terá já uma frota, e dado já haver garantia de que o melhor percurso é selecionado, terá de ser implementada uma seleção prévia de itens a distribuir por cada camião para minimizar o percurso. Para isso serão agrupados por zonas de proximidade.
- Iteração 3: contém apenas uma ligeira diferença em relação à anterior, sendo a capacidade de cada camião, que apenas criará uma exceção no modo de como os itens são acomodados em cada camião.

De atentar que não será considerado o nível de combustível de cada veículo.

Formalização do problema

Considerando uma abordagem *top-down* na explanação da implementação do projeto e própria constituição do relatório, achamos importante a criação de uma GUI *text-based,* não só para fácil manipulação de dados de entrada como saída, mas também, para que seja possível a materialização dos dados tornando-se algo mais apelativo ao utilizador.

Para isso, no início do programa serão feitas perguntas ao utilizador, para que este insira ficheiros em formato de texto (*.txt) com o grafo do mapa a ler, lista de itens e ficheiro com dados da empresa.

- “Insert the map that you which to read: “;
- “Insert the trucks to make the transportation: “;
- “Insert the item to deliver: “;

Todos os ficheiros serão lidos criando respetivas estruturas de dados a serem utilizadas. Caso algum erro aconteça na leitura, será gerada uma mensagem de erro respetiva terminando o programa. Aquando da leitura do ficheiro para o grafo será gerada uma janela com o grafo desenhado.

Após todos os dados terem sido devidamente processados será escrita na consola a GUI propriamente dita, da seguinte maneira:

```
================================================================
MAP: “file_name” | TRUCKS: “file_name” | ITEMS: “file_name”
================================================================
0 – Process Route
1 – Block street
2 – Create/Add extracting point
3 – Reset
4 – EXIT
================================================================
```

O programa será uma aplicação do tipo consola, sendo que todas as opções serão realizadas por seleção do número respetivo, sempre que uma acaba o seu processamento será novamente apresentada a *interface.*

A opção 4 permite o fecho do programa e a 3 um *reset* aos dados para que retornem ao estado inicial aquando do seu carregamento por ficheiro, isto para anular decisões tomadas na segunda opção, que terá de ser escolhida pelo menos uma vez caso haja incompatibilidade de dados entre empresa e mapas. Após seleção da opção 1 e 2 será mostrado:

- “Insert point name: ”;
- “Insert latitude and longitude: “;

Onde, com recurso ás coordenadas geográficas, será guardado,à latitude irá guardar num vértice do grafo, informação para um ponto de recolha no caso da opção 2, ou uma rua impedida desativando uma aresta do grafo no caso de 1. Caso o valor inserido tenha correspondência, caso contrário será gerado um erro, alertando o utilizador.

Por último com a opção 0 pretende-se fazer todo o processo de rotas, havendo a criação de um ficheiro de texto, guardando a informação do caminho resultante, mas também abrindo uma janela auxiliar com o mapa do grafo com o(s) trajetos(s) coloridos.

Perspetiva de solução

Na primeira fase do problema haverá apenas o foco de minimizar a rota traçada por um camião, sendo que para isso todos os métodos e algoritmos a serem descritos a seguir, terão obrigatoriamente de ser comuns a todas as iterações descritas na descrição do problema.

Será utilizada a implementação de grafos fornecida nas aulas práticas desta unidade curricular, podendo haver mudanças e adições de alguns atributos, como exemplo variáveis auxiliares a algoritmos ou nomes para melhor identificar a abstração como street em vez de aresta.

O primeiro passará por definir classes para que se organize da melhor maneira possível o código, mas principalmente para efetuar uma abordagem orientada a objetos. Criar-se-ão as seguintes classes (nomes poderão sofrer alterações aquando da implementação):

- **Interface/Menu:** tratará de toda a interface gráfica a escrever na consola, possuirá todas a funções disponíveis de controlo de menus, e terá acesso direto à classe empresa, para que seja feita a comunicação dos dados ao utilizador, não sendo uma classe com grande importância;
- **Company:** define empresa, será a classe mais importante onde todos os dados serão guardados, sendo estes o grafo do mapa, vetor de camiões e <estrutura de dados apropriada> de itens.
- **Item:** representarão objetos possuindo atributos, peso, volume e morada, sendo o último o mais importante pois será o utilizado para a escrita de rotas no grafo do mapa.
- **Truck:** tal como item servirá para representar um camião onde só consideramos relevante o volume de carga, peso e vetor de itens a transportar.

Atributos como nomes e descrição não consideramos relevantes, dado ser algo importante na descrição de cada um destes objetos, não é no sumo do projeto em si.

Para que seja feita a melhor conexão e interlocução entre as classes serão definidos headers:

- **Utils:** irá definir funções que leem os ficheiros retornando as estruturas de dados respetivas, por exemplo no caso do mapa servirá para a conversão do grafo. Escreverá também grafos e ficheiros com grafos para serem lidos pela classe viewer.
- **Algorithm:** servirão para definir funções com algoritmos de ordenação, caminho mais curto, pesquisa, etc.

Não se achou também pertinente a necessidade de herança de classes dado este problema não ter realmente grande implicação de objetos.

Todos os ficheiros gerados e lidos terão de respeitar uma listagem própria aos objetos que definem, e serão manipulados através de streams.

- Itens:

  ```
  x;y;weight;volume;

  x;y;adress;weight;volume;

  …
  ```

- Trucks:

  ```
  max_weight;max_volume;

  max_weight;max_volume;

  …
  ```

- Map:

  ```
  Nodes.txt                                   Edges.txt

  id;x;y                                    id;src;dest

  id;x;y                                   id;src;dest

  …                                                 …
  ```

Para que desta maneira sejam facilmente lidos os ficheiros através de um algoritmo do tipo:

```
While(getline(file_name,line)){
	New Object;
	While(line not empty){
		Read line until ‘;’
		Change object
		Cut line piece until ‘;’
	}
	Allocate in respective data structure;
}
```

A escrita será mais simples pois será feita através da concatenação de strings para streams escrevendo no ficheiro respetivo.

A visualização do grafo será feita, não só na geração do ficheiro de texto, mas com o auxílio da API de visualização de grafos fornecida também nesta unidade curricular. Onde a sua visualização será feita através da leitura de um ficheiro usando o mesmo método acima, mas com ciclos auxiliares responsáveis por criar nós e arestas na API. A janela de visualização terá um tamanho generalizado ao longo de todo o programa assim como cores, as únicas cores que poderão mudar são as de certas arestas aquando da geração do trajeto mínimo.

Todos os ficheiros gerados, com as sucessivas escritas e alterações feitas no decorrer do programa, terão um id numérico declarado como variável global tendo assim no fim o utilizador decidir qual o trajeto que quer (*_N.txt).

Quanto ao foco principal da implementação dos grafos, um primeiro passo, passará por definir qual a área na qual o serviço se encontrará disponível e criar um grafo que representará todos os locais de interesse a considerar para possíveis entregas e todas as ruas que os interligam.

Os nós deste grafo terão como atributos o nome do local, um par de coordenadas (x,y) e uma lista de arestas às quais o nó está ligado. Estas coordenadas serão uma aproximação às coordenadas geográficas do local.

As ruas são compostas por um conjunto de arestas, e contêm um id da aresta, o nome da rua e a indicação se a rua é de um ou dois sentidos.

Na descrição do trabalho prático é sugerido o auxílio de mapas do www.openstreetmap.org será esta a nossa principal fonte de mapas, no entanto, dada a elevada precisão desta ferramenta todos os ficheiros por ela gerados, contém excesso de informação sendo demasiado vastos, num problema em que só se pretendem rotas. Como serão implementados algoritmos que funcionam em qualquer tipo de grafo, a dimensão não o mais relevante, considerando que também caso se usassem grafos com 10000 nós e arestas provavelmente o processamento poderia ter tempos inesperados. Posto isto os grafos a utilizar serão acompanhados de imagens de mapa que serão convertidos por nós, dado a maneira como o openstreet gera os ficheiros, iremos claro também atentar ao facto de se usar pelo menos um grafo mais vasto e robusto. Serão convertidos da seguinte maneira:

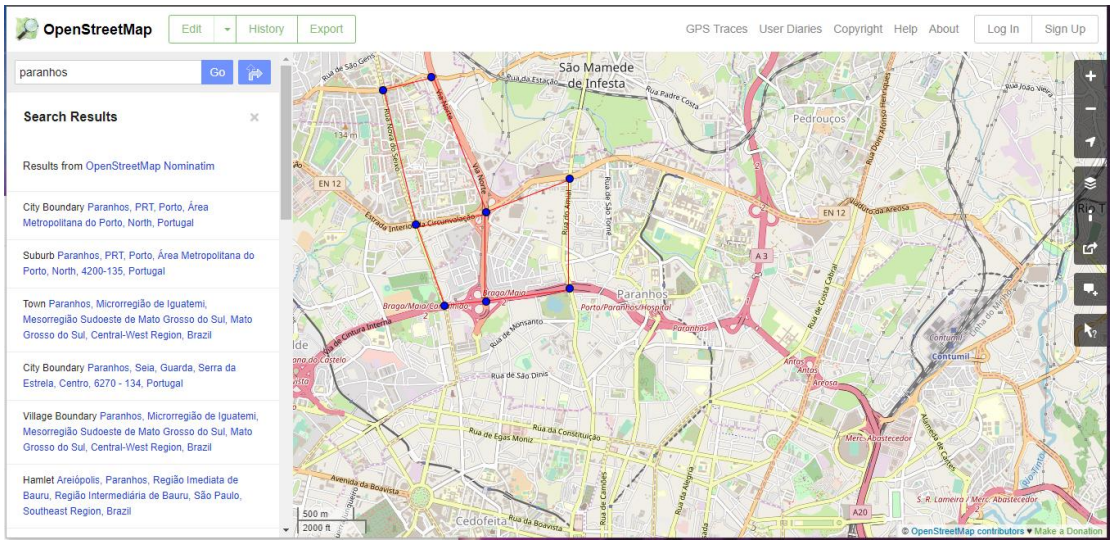

Figura 1 - exemplo da criação dos grafos

Após feito o tracejado como na figura serão criados ficheiros edges.txt e nodes.txt com toda a informação necessária a criação do grafo a usar.

Sendo o foco principal deste trabalho a otimização das rotas de transportes num mapa gerado em formato de grafo, a melhor solução para este problema passará essencialmente

pela melhor aplicação possível de um, ou mais algoritmos de caminho mais curto, dependerá agora a escolha feita dada a envergadura do problema. Como se planeia utilizar um mapa real, será expectável que o grafo utilizado seja dirigido, orientação das ruas, e pesado, suas distâncias, sendo assim será espectável o uso do algoritmo de Dijkstra para a geração de todas as rotas. Tal é sugerido nos slides desta UC.

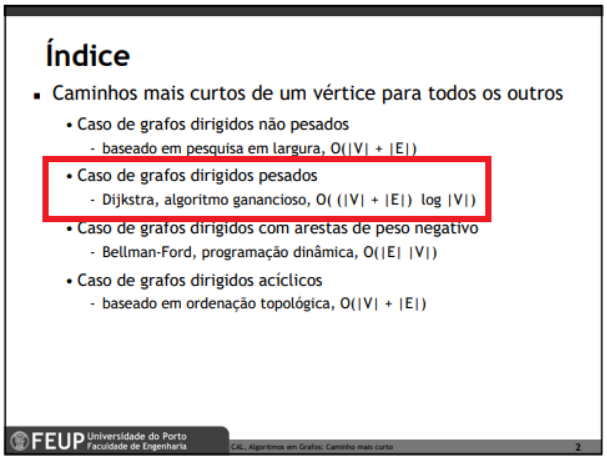

Figura 2 - Algoritmos de caminha mais curto

A implementação do algoritmo de Dijkstra está também presente nos mesmos slides, será a implementação que iremos usar, adaptando-a ao contexto do problema, portando iremos implementar assim:

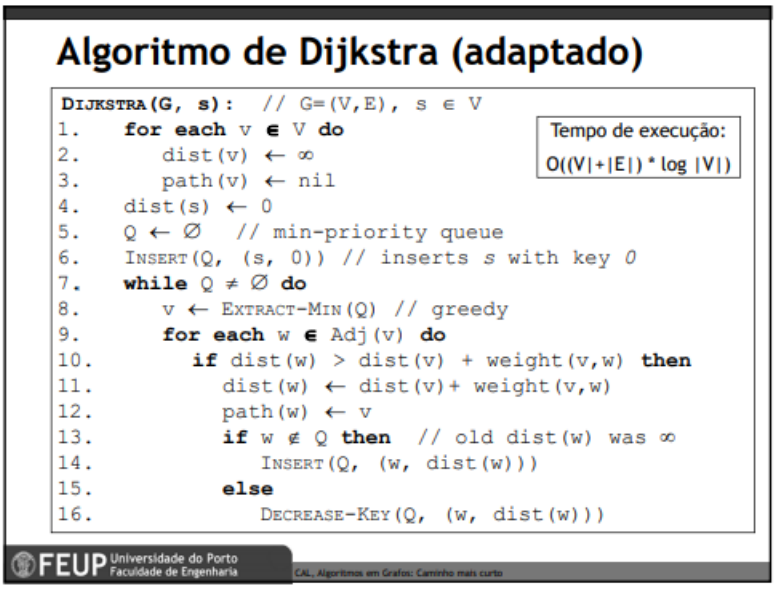

Figura 3 - Pseudo código algoritmo de Dijkstra

Portanto com todos estes métodos já seriamos capazes de implementar a iteração 1. Para que se implementar as outras teremos de focar em métodos capazes de agrupar os itens em vários camiões de modo a que estes complementem em simultâneo uma distribuição eficiente.

Na segunda e terceira iterações o problema principal reside na melhor organização dos itens. Como cada item tem a ele associado um vértice do grafo, acaba por ser um "problema do caixeiro viajante" (https://pt.wikipedia.org/wiki/Problema_do_caixeiro-viajante)(TSP), onde se pretende saber qual a melhor rota a visitar os produtos e a voltar à origem, no trajeto mais eficaz.

Para isso, camiões podem ser agrupados em vetores, pois podem seguir qualquer rota dado o ponto de partida ser sempre o mesmo, podendo o ponto de chegada variar.

Os itens serão também alocados num vetor, e serão ordenados através de um algoritmo do tipo "vizinho mais próximo" .

Adaptando o algoritmo, o vetor seria assim ordenado:

```
For each v ∈V do (
      if(v == start_item){
            mainVertex = v
            discovered(v) ← true(v)
            vector.push(start_item)
      }else{ discovered(v) ← false(v)}
While(at least one vertex isn't visited){
      lighterEdge=0;
      For each mainVertex.edges do{
            //search for the lighter edge
            If(edges.weight < lighterEdge){
```

```
                lighterEdge=edge.weight
            }
        }

        For each edge vertex do{
            If (vertex not visited) {
                mainVertex=Vertex
                mainVertex.discovered=true
                vector.push(mainVertex)
            }
        }
}
```

Pseudocódigo adaptado de (https://pt.wikipedia.org/wiki/Algoritmo_do_vizinho_mais_pr%C3%B3ximo).

Posteriormente os camiões serão preenchidos através de um ciclo, de modo a não excederem a capacidade do veículo, isto será feito até esgotarem o número de camiões ou itens. E finalmente após cada camião ser preenchido, ser lhe ia aplicada a rota através do algoritmo de Dijkstra (anteriormente definido) aos seus itens.

Nestas últimas iterações seriam implementadas restrições no código das classes para restringir cargas.

Casos de Utilização

Classes:

- Company: Contem os métodos para correr e manter todo o programa.
- Node: Local passível de receber encomendas, caracterizado por um ID, um nome, e um par de coordenadas (x,y).
- Street: Representação de uma rua, composto por uma ou múltiplas arestas, um ID, o nome da rua, e uma variável booleana que indica se a rua é de sentido duplo ou único.
- Truck: Camião de entregas contendo um ID, um peso máximo de carga, e um volume máximo.
- Item: Os itens do depósito são identificados com o nome do destinatário, valor do conteúdo de cada item, volume do item, destino, número de fatura.

Ficheiros:

- Nodes.txt - Ficheiro contento nome do local, ID e par de coordenadas(X,Y)
- Edges.txt - ID e nós que liga
- Fleet.txt - Informação relativa a todas as carrinhas que compõem à frota da empresa.
- Items.txt - Informação relativa a todas as características dos itens a serem entregues.

Conclusão

Em suma, pretendemos cumprir com todos os métodos descritos no relatório, e desejamos que a nossa proposta de solução funcione como esperado. Não consideramos o tema de trabalho algo de muita complexidade, e como referido ao longo do relatório muitas situações reais foram ocultas, como datas, combustíveis entre outros, dado acharmos que o foco principal da unidade curricular ser a escolha e implantação de algoritmos. Daí tentaremos simplificar a interface com foco à melhor e mais rápida produção de resultados possível, não só para efeitos de avaliação como de utilização

O presente relatório é na sua totalidade, fruto de um esforço conjunto de todo o grupo, pelo que a sua elaboração não foi dividida em partes. Destaca-se, no entanto uma prestação mais acentuada do Gustavo.