Relatório - Projeto de Concepção e Análise de Algoritmos

Meat Wagons (Tema 8) - Parte 1

Turma 3 - Grupo 1

André Gomes - up201806224@fe.up.pt

Gonçalo de Batalhão Alves - up201806451@fe.up.pt

Pedro Jorge Fonseca Seixas - up201806227@fe.up.pt

Descrição do Tema

Uma estabelecimento prisional necessita de gerir o transporte dos seus prisioneiros entre diferentes penitenciárias, tribunais, estabelecimentos prisionais ou esquadras policiais, tendo ao seu dispor diferentes veiculos e sendo preciso ter em conta prisioneiros com o mesmo destino, propriedades do destino para escolher o veiculo e planeamento de rotas.

O objetivo é desenvolver um programa que calcule para o dia as rotas a tomar para cada veículo da frota tendo em conta os destinos do dia para a lista de prisioneiros.

Nesta fase inicial o projeto encontra-se dividido em 3 fases.

Fase 1 : Um veiculo para todos os prisioneiros

Tendo uma lista dos prisioneiros, e sabendo para cada um o seu destino, o objetivo desta primeira parte é ter um autocarro com capacidade ilimitada que passe por todos os POI designados para cada prisioneiro, tendo em conta o caminho mais curto que passe em todos os POI e que depois retorne para a origem. Certas vias podem ser inacessíveis por razões diversas (obras, cortes de estrada, largura de rua não ser suficiente). Assim, durante o processamento do grafo será necessário desprezar certas arestas.

Fase 2 : Diferentes Veiculos, capacidade ilimitada

Nesta fase cada Vértice do Grafo terá a informação extra sobre a sua Densidade Populacional (Cidade, Periferia ou Campo). Os prisioneiros serão previamente divididos pelos Veiculos tendo em conta o seu destino e cada veiculo estará especializado para uma Densidade Populacional. Começaremos por ter uma camioneta (Campo e Periferia) e um carro (Cidade), ambos com capacidade ilimitada, para testar a divisão dos prisioneiros e a formulação de rotas.

Fase 3 : Mais veiculos, capacidade Limitada

Na fase final teremos uma frota de veiculos com capacidade limitada e um conjunto de prisioneiros com o seu destino. A leitura de dados inicial será para o dia, ou seja, ao executar o programa fica em memório os prisioneiros com o seu destino e a frota disponivel. Caso toda a frota esteja ocupada e ainda sobrarem prisioneiros, tendo em conta o seu destino, será retornado para a origem os veiculos necessários para refazer rotas e os transportar.

Possiveis Problemas a Encontrar

- Leitura dos mapas proveninentes do Open Street Maps
- Peso das arestas do grafo
- Conversão dos dados de modo a possibilitar a utilização do GraphViewer
- Cruzamento de ruas com o mesmo nome
- Acentuação nos nomes das ruas
- Nos ficheiros de mapas não são disponibilizadas tags referentes a tribunais ou estabelecimentos prisionais

- distribuição de tag densidade populacional pelos vertices do grafo

Formalização do Problema

Dados de Entrada

Pi - Lista de prisioneiros com destinos para o dia, sendo P(i) o i-nésimo elemento, cada um é caracterizado por: - ID - número identificador de prisioneiro - Destino - número identificador de destino

O número identificador de destino terá duas partes, a primeira parte de 1 digito é correspondente ao tipo de destino e a segunda parte de 3 digitos é correspondente ao destino concreto de um certo tipo. Como exemplo: Um tribunal poderá ter um Destino de 1001 enquanto que um estabelecimento prisional poderá ter um Destino de 2001.

Fi - Lista de veiculos da frota, sendo F(i) o i-nésimo elemento, cada um é caracterizado por: - ID - número identificador do veículo (tal como em Destino, será um numero que terá implicito o tipo de veiculo) - cap - número de assentos destinados a prisioneiros

Gi = (Vi, Ei) - Grafo Dirigido Pesado (Dirigido -> sentido da rua|Pesado -> Distância entre vértices) - V - vértices representativos de pontos da cidade com: - type - Penitenciária, tribunal, esquadra policial ou estabelecimento prisional (cada um terá um ID igual a Destino)(Caso não seja nenhum desses, ID = 0) - dens - Informação sobre densidade populacional (Cidade, Periferia ou Campo) - Adj ⊆ E - conjunto de arestas que partem do vérce - E - arestas representativas das vias de comunicação - w - peso da aresta a (representa a distância entre os dois vérces que a delimitam) - ID - identificador único da aresta - dest ∈ Vi - vértice de destino

S ∈ Vi - vértice inicial (Estabelecimento prisional)

T ⊆ Vi - vértices finais (Destinos)

Dados de Saída

Gf = (Vf,Ef) grafo dirigido pesado, tendo Vf e Ef os mesmos atributos que Vi e Ei.

Ff- Lista ordenada de todos os veiculos usados, sendo Ff(i) o seu i-ésimo elemento. Cada um tem os seguintes valores: - cap - número de assentos utilizados - P = {e ∈ Ei | 1 ≤ j ≤ |P|} - sequência ordenada (com repetidos) de arestas a visitar, sendo ej o seu j-ésimo elemento.

Restrições

Sobre os Dados de entrada

- ∀ i ∈ [1 ; |Cf|], cap(Cf[i]) >= 0, dado que uma capacidade representa os assentos usadas, caso um veiculo não seja usado, cap = 0;
- ∀ v ∈ Vi, type ≥ 0;
- ∀ e ∈ Ei, w > 0, dado que o peso de uma aresta representa uma distância entre pontos de um mapa.
- ∀ e ∈ Ei, e deve ser utilizável pelo elemento da frota. Senão, não é incluída no grafo Gi.

Sobre os Dados de Saída

No grafo Gf: - ∀ vf ∈ Vf, ∃ vi ∈ Vi tal que vi e vf têm os mesmos valores para todos os atributos - ∀ ef ∈ Ef, ∃ ei ∈ Ei tal que ei e ef têm os mesmos valores para todos os atributos.

Função Objetivo

Diminuir a distância total percorrida pela frota que será a soma dos valores das arestas percorridas por cada veículo.

Perspetiva de solução

// Ver Issues referentes a esta parte

Fase 1

Esta primeira fase terá varios passos referentes á preparação do ambiente de trabalho, começando pela:

1. Preparação dos ficheiros de entrada

Serão utilizados os ficheiros de nodes e edges fornecidos pelos professores. A informação lida dos ficheiros será guardada num grafo G. Será criada uma tag para cada tipo de edifício de interesse (prisões, esquadras e tribunais) de forma a facilitar a identificação dos pontos de interesse.

2. Analise da Conectividade do Grafo

Para verificar as ruas que estão inacessiveis

3. Criação de POI's

Após a leitura dos ficheiros com os nodes e edges, serão lidos os ficheiros das tags de forma a identificar os pontos de interesse, alterando, para esse node, a sua variável type, inicializada a 0, para o seu valor correspondente ao tipo de ponto de interesse.

Assim que a preparação estiver pronta é possivel seguir para a implementação de código. Nesta fase será necessário que o programa consiga criar 2 rotas, uma de ida e outra de volta.

Para a Rota de ida será usado o algoritmo de Dijkstra:

DIJKSTRA(G, s): // G=(V,E), s in V	BIDIRECTIONAL DIJKSTRA(G, s):
1. for each v in V do 2. dist(v) <- INF 3. path(v) <- nil 4. dist(s) <- 0 5. Q <- 0 // min-priority queue 6. INSERT(Q, (s, 0)) // inserts s with key 0 7. while Q != 0 do 8. v <- EXTRACT-MIN(Q) 9. for each w in Adj(v) do 10. if dist(w) > dist(v) + weight(v,w) then 11. dist(w) <- dist(v)+ weight(v,w) 12. path(w) <- v 13. if w not in Q then 14. INSERT(Q, (w, dist(w))) 15. else 16. DECREASE-KEY(Q, (w, dist(w)))	A COMPLETAR

da seguinte forma: Começando no estabelecimento prisional onde se encontram os prisioneiros é usado o algoritmo até encontrar um vértice que é uma paragem de um dos prisioneiros. Neste ponto é usado outra vez o algoritmo de Dijkstra mas com o vértice encontrado a ser usado como vértice de inicio para encontrar a próxima paragem. Assim que todos os prisioneiros estiverem distribuidos será necessário encontrar o caminho de volta. Para isso é aplicado o algoritmo de Dijkstra Bi-Direcional para encontrar o caminho mais curto entre o Vértice final do passo anterior e o estabelecimento prisional inicial.

Fase 2

Na segunda fase teremos em conta o valor de densidade populacional (DP) dos vértices e 2 veículos, cada um especializado para os seus valores de DP. Iso leva-nos a 2 formas de encontrar o caminho mais curto, tendo em conta uma divisão prévia dos prisioneiros:

Hipótese 1 - Não tendo em conta o caminho

Como primeira hipótese considera-se apenas a DP dos destinos de cada um dos prisioneiros. Tendo isso em conta é feita uma divisão em dois grupos baseado no DP do destino de cada prisioneiro. Um grupo será levado por um carro (com capacidade infinita) para destinos com DP de *cidade* enquanto que o outro grupo será levado por um autocarro (também com capacidade infinita) para destinos com DP de *periferia* ou *campo*. Feita a divisão o problema simplifica-se a aplicar o método da fase 1 para cada veículo.

Hipótese 2 - Tendo em conta o caminho

Nesta Hipótese, que à primeira vista parece mais precisa, será feito no inicio o cálculo de uma rota como na fase 1 para todos os prisioneiros. A partir do processamento da rota, cada prisioneiro ficará com o valor de cada DP dos vértices pela qual passou. Tendo em conta o DP máximo de cada prisioneiro fazem-se as divisões em 2 grupos e segue-se como na hipótese anterior para a divisão nos veiculos e calculo de rotas. A contagem dos DP para cada prisioneiro terá em conta apenas a rota inicial e não as rotas criadas pelos veiculos a qual ficaram designados, isto poderá não trazer os melhores resultados quanto á divisão dos prisioneiros entre veiculos mas é uma melhoria face à hipótese anterior.

Fase 3

A diferença principal desta fase para a anterior é o limite em cada veiculo, com isso em conta, para esta fase adotamos os mesmos passos da hipótese 2 da fase 2 até à divisão dos prisioneiros pelos veiculos, mas neste caso teremos que fazer um cálculo do resto de prisioneiros caso não haja transporte para todos. Caso haja transporte para todos o problema torna-se igual á fase 2. Em caso contrário faz-se um cálculo dos veiculos que terão de retornar ao estabelecimento prisional inicial tendo em conta a sua capacidade, o número de prisioneiros e o grupo à qual os prisioneiros restantes estão designados. Assim que este cálculo for realizado repete-se este método a partir do ponto em que se calcula o resto dos prisioneiros.

Casos de utilização

Not sure o que pôr aqui. Pode ser usado para calcular as rotas para um dia.

Funcionalidade a implementar

Also not sure o que pôr aqui. Cálculo de rotas, Divisão de prisioneiros, Adição de tags a vértices do grafo, Adição de pesos a arestas do grafo. Criação de POI's personalizados ao nosso tema.

Conclusão

Temos um plano traçado e bem dividido, esperamos que na altura da implementação consigamos tratar de forma eficiente de todas as partes, principalmente do uso da parte gráfica de criação de grafos, visto que será a primeira vez neste curso que faremos uso de algo semelhante.