Logística de Viagens

Desenho de Algoritmos

Grupo 37

Guilherme Moreira - up202007036 (2LEIC11)
Isabel Amaral - up202006677 (2LEIC12)
Milena Gouveia - up202008862 (2LEIC12)

Descrição do problema

O objetivo deste trabalho é a criação de uma plataforma eletrónica personalizada às necessidades de uma agência de viagens, procurando otimizar vários cenários com que a empresa se depara no seu dia a dia.

Essencialmente, pretende-se dar resposta a esses cenários segundo o tipo de viagem:

- Viagens sem separação de grupo.
- Viagens com separação de grupo.

Cenário 1.1 - Caminho de capacidade máxima

Objetivo: Encontrar o caminho, independentemente do tamanho do seu percurso, que consiga transportar o maior grupo de pessoas possível, sem que estas se possam separar.

Algoritmo utilizado: Adaptação do algoritmo de Dijkstra

Formalização do problema:

Dados de entrada:

- G, grafo correspondente à rede de transportes
- src, inteiro representativo do ponto de origem do percurso
- dest, inteiro representativo do ponto de chegada do percurso

Output: P = {v1, v2, ..., vn} - percurso que o grupo deve seguir desde a origem ao destino de forma a poder ter a maior dimensão possível sem se separar

C - capacidade do percurso

Cenário 1.1 - Caminho de capacidade máxima

Descrição da solução: Adaptação do algoritmo de Dijkstra

- Inicialização da capacidade de todos os nós a INT_MIN e do nó de origem a INT_MAX
- Adiciona-se o nó de origem a um vetor
- Enquanto o vetor não estiver vazio:
  - Escolhe-se o nó com maior capacidade, sendo este eliminado do vetor
  - Para cada nó adjacente ao nó selecionado anteriormente, calcula-se o mínimo entre entre a capacidade do nó selecionado e a capacidade da aresta que os liga e, se este valor for maior que a capacidade do nó adjacente, esta é atualizada

Cenário 1.1 - Caminho de capacidade máxima

Complexidade Temporal: O(E * V), V é o número de paragens de autocarro, E é o número de autocarros

- Inicialização da capacidade de cada nó a INT_MIN - O(V)
- Em cada iteração procura-se o nó não visitado de capacidade máxima - O(V)
- Para cada nó, percorre-se todas as suas arestas de modo a encontrar o caminho de capacidade máxima - O(E)

Complexidade Espacial: O(V), V é o número de paragens de autocarro

- Cria-se um vetor auxiliar onde se introduz todos os nós do grafo em questão com a sua respetiva capacidade

Cenário 1.2 - Caminho de capacidade máxima vs. caminho com menor nº de transbordos

Objetivo: Encontrar o caminho que minimize o número de transbordos de forma a comparar este resultado com o obtido na alínea anterior

Algoritmo utilizado: BFS

Formalização do problema:

Dados de entrada:

- G, grafo correspondente a rede de transportes
- src, inteiro representativo do ponto de origem do percurso
- dest, inteiro representativo do ponto de chegada do percurso

Output: P = {v1, v2, ..., vn} - percurso que o grupo deve seguir desde a origem até ao destino de forma a realizar menor numero de transbordos

t, número de transbordos

Cenário 1.2 - Caminho de capacidade máxima vs. caminho com menor nº de transbordos

Descrição da solução:

- Declara-se todos os nós como não visitados
- É utilizada um *queue* de nós visitados, inicialmente apenas com o nó de origem
- Enquanto a *queue* não estiver vazia ou o nó de destino não tiver sido atingido:
    - Remover um nó da *queue* e percorrer todos os seus nós adjacentes que ainda não tenham sido visitados, acrescentando-os também à *queue.*

Cenário 1.2 - Caminho de capacidade máxima vs. caminho com menor nº de transbordos

Complexidade Temporal: O(V + E), V é o número de paragens de autocarro, E é o número de autocarros

- Todos os nós são inicializados como não visitados - O(V)
- Para cada nó, percorre-se todas as suas arestas de modo a encontrar o caminho mais curto até ao destino - O(V + E)

Complexidade Espacial: O(V), V é o número de paragens de autocarros

- Cria-se uma queue auxiliar onde se introduz todos os nós do grafo em questão

Cenário 2.1 - Percurso para grupos que se podem separar

Objetivo: Encontrar um ou mais caminhos para um grupo, dada a sua dimensão, sendo que os seus elementos podem se separar.

Algoritmo utilizado: Edmonds–Karp e BFS

Formalização do problema:

Dados de entrada:

- G, grafo correspondente a rede de transportes
- src, inteiro representativo do ponto de origem do percurso
- dest, inteiro representativo do ponto de chegada do percurso
- dim, dimensão do grupo a transportar

Output: P = {{n1, n2, ..., n}, {n1, n2, ..., n}}, percurso que cada subgrupo deve seguir desde a origem ao destino.

C = {c1, c2,....,cn}, capacidade de cada caminho encontrado, sendo também correspondente a dimensão de cada subgrupo

Cenário 2.1 - Percurso para grupos que se podem separar

Descrição da solução: Aplicação do algoritmo de Edmonds–Karp BFS

- Determinação do fluxo máximo e do grafo residual (Edmonds-Karp)
- Atualização do fluxo de cada aresta do grafo original
- Determinação de todos caminhos possíveis desde a origem ao destino, sendo apenas consideradas as arestas com um fluxo positivo
- Determinação da capacidade máxima de cada caminho
- Ordenação decrescente dos caminhos segundo a capacidade máxima
- Enquanto existir lugares disponíveis e passageiros por transportar:
  - Para cada caminho com capacidade máxima cMax, alocar um subgrupo com dimensão menor ou igual a cMax
  - Atualizar os lugares disponíveis e a o número de pessoas por transportar

Cenário 2.1 - Percurso para grupos que se podem separar

Complexidade Temporal: O(V ^V), V é número de nós.

- Determinar o fluxo máximo - enquanto houver caminho entre a origem e o destino (bfs), atualizar as capacidades no grafo residual e o fluxo no original (Edmonds-Karp) - O(V*E^2)
- Encontrar todos os caminhos desde a origem ao destino, sendo que no pior caso todos os nós podem estar interligados - O(V^V)
- Para cada caminho encontrado, alocar um subgrupo: O(P), P é o número de caminhos possíveis.

Complexidade Espacial: O(V ^ V), V é o número de paragens

- Criação do grafo residual e adição de ramos com direções contrárias durante a atualização das capacidades do mesmo: - O(V+2*E) ≈ O(V+E)
- Guardar em um vetor todos os caminhos possíveis, no pior caso pode conter todas as combinações dos nós: -O (V^V)

Cenário 2.2 - Correção de percurso

Objetivo: Corrigir o encaminhamento encontrado no cenário 2.1 de modo a aumentar a dimensão do grupo em X unidades dadas.

Algoritmo utilizado: Edmonds–Karp

Formalização do problema:

Dados de entrada:

- G, grafo correspondente a rede de transportes
- src, inteiro representativo do ponto de origem do percurso
- dest, inteiro representativo do ponto de chegada do percurso
- dim, dimensão do grupo a transportar
- extra, número de elementos adicionais a transportar

Output: P1 = {{n1, n2, ..., n}, {n1, n2, ..., n}}, percurso determinado no cenário 2.1
P2 = {{n1, n2, ..., n}, {n1, n2, ..., n}}, percurso que cada subgrupo do grupo adicional deve seguir

Cenário 2.2 - Correção de percurso

Descrição da solução: Aplicação do algoritmo de Edmonds–Karp

- Determinação do encaminhamento para o grupo inicial (sem os elementos extra)
- Se restarem lugares:
  - Se os caminhos utilizados anteriormente não estiverem lotados, parte dos elementos serão alocados para o mesmo
  - O resto do grupo será alocado para os restantes percursos usando a mesma lógica do cenário anterior e sendo também divido em subgrupos se necessário

Cenário 2.2 - Correção de percurso

Complexidade Temporal: O(V ^V), V é número de nós.

- Determinar o fluxo máximo - enquanto houver caminho entre a origem e o destino (bfs), atualizar as capacidades no grafo residual e os fluxos no original (Edmonds-Karp) - O (V*E^2)
- Encontrar todos os caminhos desde a origem ao destino, que no pior caso todos os nós podem estar interligados - O(V^V)
- Para cada caminho encontrado, alocar um subgrupo do grupo inicial: O(P), P é o número de caminhos possíveis.
- Se restam lugares (caminhos), alocar subgrupos do grupo extra- O(P'),P' é o número de caminhos que sobraram após encontrar solução do cenário 2.1.

Complexidade Espacial: O(V ^ V), V é o número de paragens

- Criação do grafo residual e adição de ramos com direções contrárias durante a atualização das capacidades do mesmo: - O(V+2*E) ≈ O(V+E)
- Guardar em um vetor todos os caminhos possíveis, no pior caso pode conter todas as combinações dos nós: -O (V^V)

Cenário 2.3 - Dimensão máxima para grupos que se podem separar

Objetivo: Saber qual é a máxima dimensão possível assumindo que o grupo se pode separar

Algoritmo utilizado: Edmonds–Karp

Formalização do problema:

Dados de entrada:

- G, grafo representativo da rede de transportes a considerar
- src, inteiro representativo do ponto de origem do percurso
- dest, inteiro representativo do ponto de chegada do percurso

Output: f, fluxo máximo da rede de transportes a considerar. Este valor corresponde ao maior número de pessoas que é possível transportar de cada vez nos vários caminhos desde a origem até ao destino

Cenário 2.3 - Dimensão máxima para grupos que se podem separar

Descrição da solução: Aplicação direta do algoritmo de Edmonds-Karp

- Cálculo do grafo das capacidades residuais
- A cada iteração, enquanto for possível encontrar no grafo residual um caminho da origem até ao destino:
  - Calcular o caminho mais curto da origem ao destino - bfs
  - Calcular a capacidade residual mínima desse caminho
  - Em cada aresta do caminho a ser considerado incrementar o fluxo um nº de unidades igual ao da capacidade residual mínima nesse caminho
  - Atualizar o grafo residual perante a variação do fluxo
- O fluxo máximo é a soma dos fluxos de cada aresta que sai da origem

Cenário 2.3 - Dimensão máxima para grupos que se podem separar

Complexidade Temporal: O(E^2 * V), V é o número de paragens de autocarro, E é o número de autocarros

- inicialização da rede residual a partir do grafo original - O(E+V)
- encontrar o menor caminho da origem até ao destino (bfs) - O(E+V)
- aplicação do bfs e atualização do grafo residual enquanto for possível encontrar um caminho da origem até ao destino (edmonds-karp) - O(E^2 * V)

Complexidade Espacial: O(V+E), V é o número de paragens de autocarro

- construção do grafo residual a partir do grafo inicial, todos os nós e arestas são copiados, são também acrescentadas arestas na direção contrária das já existentes - O(V+2*E) ≈ O(V+E)

Cenário 2.4 - Ponto de encontro para grupos que se podem separar

Objetivo: Saber em que ponto/paragem de autocarro (nó) todos os grupos se podem reunir novamente. No máximo isto acontece na paragem de destino, contudo, pode ser mais cedo

Algoritmo utilizado: Greedy (tenta que o grupo se reúna o mais cedo possível)

Formalização do problema:

Dados de entrada:

- G, grafo representativo da rede de transportes a considerar
- src, inteiro representativo do ponto de origem do percurso
- dest, inteiro representativo do ponto de chegada do percurso
- P = {{v1, v2, ..., vn}, {v1, v2, ..., vn}} - caminhos a ser utilizados pelos vários grupos

Output: n, nó correspondente à paragem em que todos os grupos se reúnem

Cenário 2.4 - Ponto de encontro para grupos que se podem separar

Descrição da solução:

- Considera-se que o ponto em que o grupo se pode reunir é o destino - earliest
- A cada iteração:
  - Para cada grupo vê-se qual é o nó anterior a earliest
  - Se o nó for o mesmo para todos os grupos, passa-se a considerar que se podem reunir nesse ponto (atualiza-se o valor de earliest)
  - Caso contrário, considera-se que o ponto em que todos os grupos se podem reunir o mais cedo possível já foi encontrado (earliest)

Cenário 2.4 – Ponto de encontro para grupos que se podem separar

Complexidade Temporal: O(V), V é o número de paragens de autocarro

- No pior caso, os grupos reúnem-se na origem (os caminhos são os mesmos) e o caminho passa por todos os nós do grafo

Complexidade Espacial: O(1)

- Todas as variáveis utilizadas têm espaço constante

Cenário 2.5 - Tempo de espera para grupos que se podem separar

Objetivo: Saber qual o tempo de espera no nó em que os grupos se reúnem. Todos os grupos deverão esperar pelo último grupo a chegar

Algoritmo utilizado: Critical Path Method

Formalização do problema:

Dados de entrada:

- G, grafo representativo da rede de transportes a considerar
- src, inteiro representativo do ponto de origem do percurso
- dest, inteiro representativo do ponto de chegada do percurso
- r, inteiro representativo do ponto de encontro de todos os grupos
- P = {{v1, v2, ..., vn}, {v1, v2, ..., vn}} - caminhos a ser utilizados pelos vários grupos
- Tv = {tv1, tv2, ..., tvn} - duração de cada viagem de uma paragem até outra

Output: Te = {te1, te2, ..., ten} - tempo de espera de cada grupo, o último grupo a chegar terá te = 0

Cenário 2.5 - Tempo de espera para grupos que se podem separar

Descrição da solução:

- Calcula-se o tempo de viagem de todos os grupos desde o nó de origem até ao nó de encontro através da soma dos tempos de viagem de cada aresta
- Encontra-se o grupo com maior tempo de viagem
- Para os restantes grupos calcula-se o tempo de espera subtraindo o tempo de viagem de cada grupo ao tempo de viagem do último grupo ao tempo de viagem do último grupo a chegar

Cenário 2.5 - Tempo de espera para grupos que se podem separar

Complexidade Temporal: O(P * E), P é o número de grupos a viajar separadamente (número de caminhos), E é o número de autocarros

- Para cada caminho - O(P)
  - É calculado o tempo de viagem, para isto é necessário encontrar quais das arestas do grafo pertencem a este caminho - O(E) (no pior caso - se for preciso avaliar todas as arestas)

Complexidade Espacial: O(P), P é o número de grupos a viajar separadamente (número de caminhos)
Todas as variáveis ocupam espaço constante com excessão da seguinte variável auxiliar:

- lista de tempos de viagem - tamanho p

Esforço e principais dificuldades

As principais dificuldades encontradas foram, de modo geral, no planeamento do algoritmo ideal para cada cenário e na escolha das estruturas de dados a usar para retornar o resultado de alguns destes.

No geral, o trabalho foi dividido pelos elementos do grupo de forma mais ou menos equilibrada. Numa fase inicial, um elemento ficou responsável pela leitura dos datasets e posteriormente, cada elemento ficou encarregue da implementação de dois a três cenários. Contudo, o planeamento foi feito em conjunto e, durante todo o processo, todos os elementos do grupo se mostraram disponíveis para ajudar e esclarecer as dúvidas que foram surgindo ao longo do trabalho.