Desenho de Algoritmos Trabalho_2

Agência de Viagens

André Costa	up201905916
Óscar Esteves	up201906834
Pedro Nunes	up201905396

Descrição dos Cenários

- **Cenário I:**

  Viagens para grupos que não se querem separar ao longo da viagem.

- **Cenário II:**

  Viagens para grupos que não se importam de serem separados ao longo da viagem.

Formalizações

Dados de entrada

- E_v − *set de todos os veículos (arestas/edges)*
- V_l − *set de todas as localidades (vértices)*
- v_i e v_j − *localidades de, respetivamente, origem e destino desejadas*
- c − *capacidade (dimensão) do grupo desejada*

Dados de saída

P − *set de caminho(s) que façam o trajeto desde o local de origem até ao local de destino, segundo os requisitos de cada problema*

Definições

- MAX_p − *capacidade(dimensão) máxima dos caminho p*
- p^k − *caminho p do veículo k*
- MAX_l − *capacidade máxima de uma localidade l*
- $cost_p$ − *custo do caminho p*
- $d(p)$ − *duração total do caminho p*
- $w(i, j)$ − *custo de passar fluxo pela aresta (i, j)*

Formalizações

Restrições

- *Qualquer variável pertence aos números inteiros positivos* Z_0^{+}
- $MAX_p >= c$ $-$ *capacidade do(s) caminho(s) p é necessariamente maior ou igual que a dimensão do(s) grupo(s)*
- $\sum_{i \in V_l} \sum_{k \in E_v} \sum_{v \in V_l} x_{kiiu} = 0, \forall_p \in P$ $-$ *Quando um veículo é utilizado para levar um grupo, não pode ser utilizado novamente*
- $\forall l \in V_l, MAX_L = \infty$ $-$ *Cada vértice (localidade) tem capacidade infinita*
- $\forall_{i,j \mid i \neq j}, w_{i,j} = 1$ $-$ *O custo de qualquer aresta (transbordo) é 1*

Objective Functions

- $T_{1.1} = max \sum_{p \in P} \sum_{i=1}^{|p|-1} c(p_i, p_{i+1})$
- $T_{1.2} = min \sum_{ij} c(i, j) \, x \, w(i, j) \; (1)$
- $T_{2.1} = ma \sum_{p \in P} \sum_{i=1}^{|p|-1} c(p_i, p_{i+1})$
- $T_{2.3} = max \, \forall p_{ij} \, c(p_i, p_j)$
- $T_{2.4} = min \sum_{i} \in P, d(P)$

Cenário I

Questão -> 1

- ## Descrição:

Nesta questão é pretendido determinar a dimensão máxima possível de um grupo, sabendo que este tem sair do local de origem e chegar ao destino pretendido, mostrando ainda o caminho percorrido.

- ## Solução:

A nossa abordagem a este exercício foi com base nos slides no algoritmo de uma **adaptação do Algoritmo de Dijkstra**. Para este Algoritmo, foi necessário 2 tipos de estrutura de dados, um grafo e uma fila de prioridade.

Cenário I

Questão -> 1

- **Descrição:**

Nesta questão é pretendido determinar a dimensão máxima possível de um grupo, sabendo que este tem sair do local de origem e chegar ao destino pretendido, mostrando ainda o caminho percorrido.

- **Solução:**

A nossa abordagem a este exercício foi com base nos slides no algoritmo de uma **adaptação do Algoritmo de Dijkstra**. Para este Algoritmo, foi necessário 2 tipos de estrutura de dados, um grafo e uma fila de prioridade.

Cenário I

Questão -> 1

(...)

A heap (fila de prioridade) foi usada para ir guardando o vértice/nó (local) com mais capacidade até ao momento. Era necessário também ir definindo os pais dos véstices com mais caoacidade para mostrar o caminho percorrido.

- **Complexidade Temporal:**

  2 * O(N) + O(T)

- **Complexidade Espacial:**

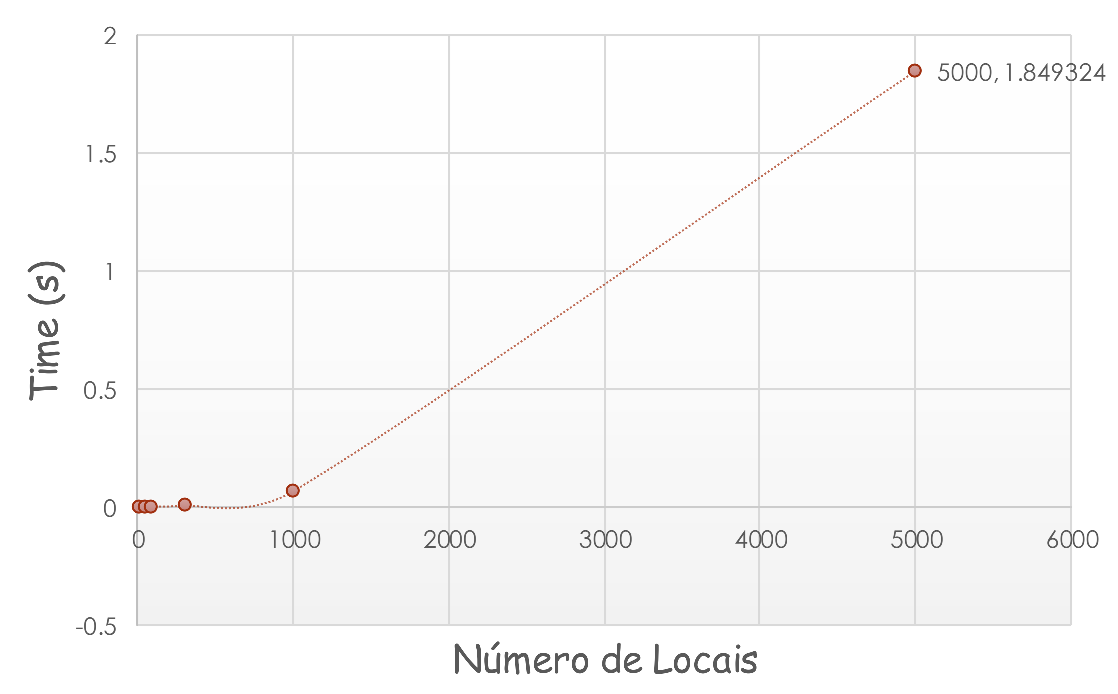
  2 * N + T

N -> nº de Vértices
T-> nº de Edges

Cenário I

Questão -> 1

- **Análise Empírica do Algoritmo:**



Cenário I

Questão -> 2

- **Descrição:**

O objetivo desta questão é muito parecido com a anterior, mas com um novo requisito. Para além do que já pede na questão anterior, pede para determinar a dimensão máxima com o menor número de transbordos. O objetivo nesta função, não é priorizar nenhum deles, mas sim priorizar os 2. Fazer os dois algoritmos e compará-los.

- **Solução:**

- O primeiro algoritmo consiste numa adaptação, tal como no primeiro exercício, do algoritmo de Dijkstra. O segundo algoritmo consiste num MaxFlowMinCost que obtém o caminho com menor transbordo e, dentro dos caminhos com menor transbordo, com maior capacidade (consegue transportar o maior número de pessoas).

Cenário II

Questões:

-> 1

-> 2

-> 3

- **Descrição:**
  - **Q -> 1:**

    Dada a dimensão do grupo, tinhamos de determinar um encaminhamento.

  - **Q -> 2:**

    Dado um encaminhamento, corrigi-lo para que a dimensão do grupo possa aumentar um determinado valor à nossa escolha, dando uma msg de erro caso ultrapasse o limite.

  - **Q -> 3:**

    Determinar a dimensão máxima possível para um grupo ir da origem até ao destino, desta vez, podendo se separar e mostrar o encaminhamento.

Cenário II

Questões:

-> 1
-> 2
-> 3

- **Solução:**

A nossa abordagem aos três problemas foi bastante semelhante. Pegamos no Método de Ford-Fulkerson e fomos adaptando consoante o requerido.

Começamos por criar uma rede residual que ia sendo alterada consoante os caminhos ainda disponíveis do local de origem para o destino. Estes caminhos iam ficando cheios, calculando o menor fluxo possível de um caminho e preenchendo as arestas.

A paragem do ciclo é que varia nos 3 algoritmos.

Cenário II

Questões:

-> 1

-> 2

-> 3

- **Solução (cont):**
  - **Q -> 1:**

    Condição de paragem: Enquanto a dimensão não fosse nula. Esta ia sendo decrementada com o fluxo mínimo que consegui chegar ao destino.

  - **Q -> 2:**

    Condição de paragem: Enquanto a diferença entre o novo fluxo e o antigo não fosse igual ao incremento. O novo fluxo começava igual ao fluxo anterior e ia sendo incrementado.

  - **Q -> 3:**

    Condição de paragem: Enquanto o ultimo vértice fosse visitado, ou seja, enquanto houvesse caminho até ao destino. A capacidade máxima vai ser igual à soma dos fluxos que saem da origem.

Cenário II

Questões:
-> 1
-> 2
-> 3

- **Complexidade Temporal:**

  n*((O(N) +2*(O(N + T))) + O(N + T))

- **Complexidade Espacial:**

  2 * (N + T) + “paths”

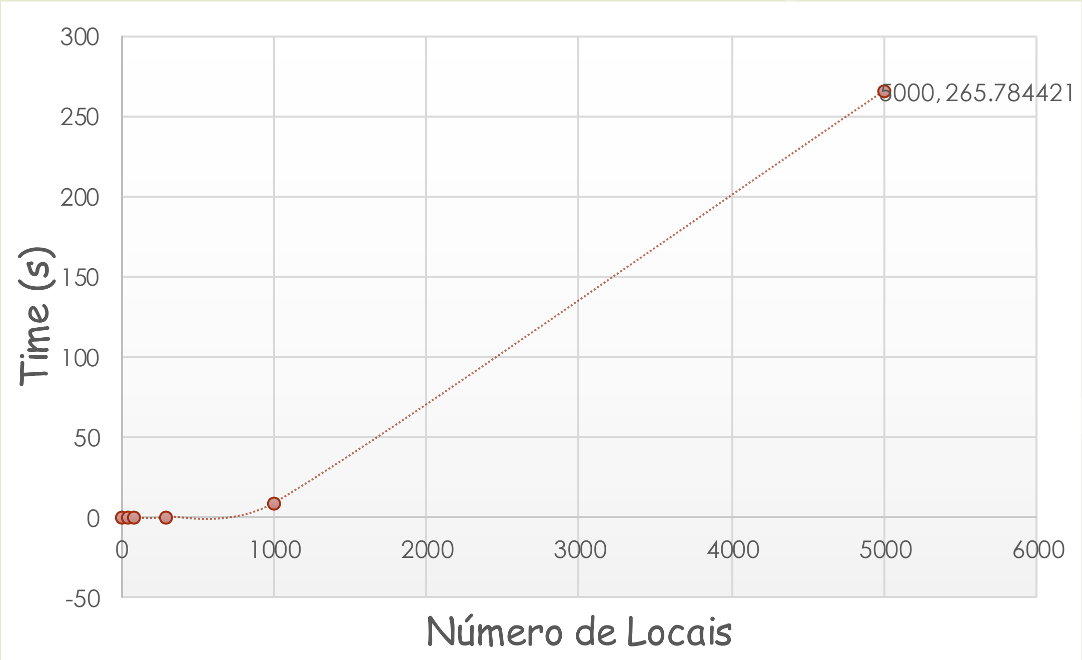
N -> nº de Vértices
T -> nº de Edges
n -> nº de vezes que corre o ciclo

paths -> caminhos disponíveis até ao destino que guarda alguns vértices, dependendo do caminho

Cenário II

Questões:
-> 1
-> 2
-> 3

Análise Empírica do Algoritmo:



Cenário II

Questão -> 2.4

- **Descrição:**

Nesta função, partindo de um encaminhamento, é necessário determinar quando o grupo se irá reunir novamente no destino, ou seja, ver quem foi pelo fluxo com a duração maior e determiná-la.

- **Solução:**

Tendo o grafo com um encaminhamento, seria necessário correr os fluxos todos e calcular a duração total da origem ao final, guardando a duração máxima.

Cenário II

Questão -> 2.4

- **Complexidade Temporal:**

  n * V * T

  n -> nº de paths, nº de vezes que vai fazer o while
  N -> nº de vértices
  T -> nº de Edges

- **Complexidade Espacial:**

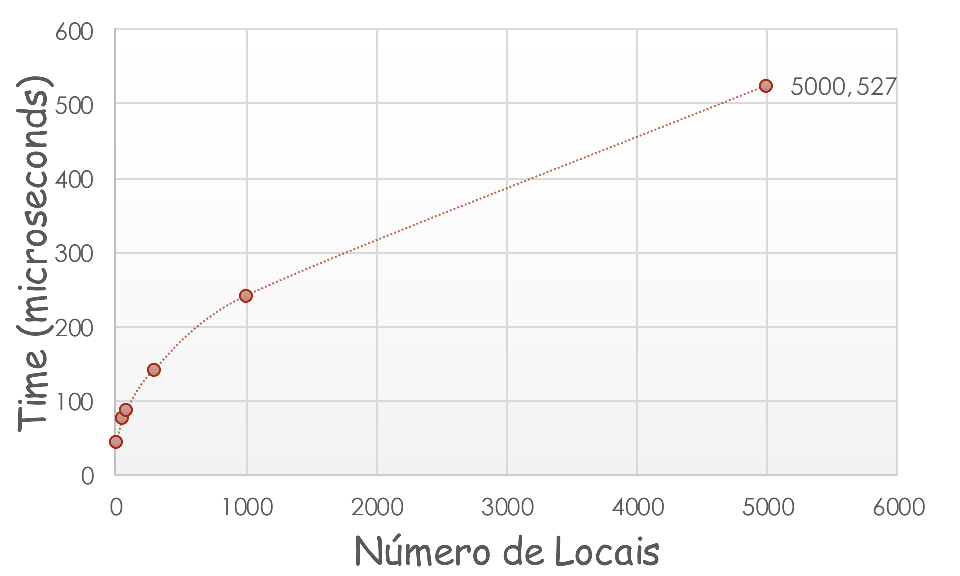
  N * T

  N -> nº de vértices
  T -> nº de Edges

Cenário II

Questão -> 2.4

- **Análise Empírica do Algoritmo:** (depende do nº de paths)



Cenário II

Questão -> 2.5

- **Descrição:**
- Nesta função, partindo de um encaminhamento,
é necessário determinar o tempo máximo de espera e os nós em que esse tempo ocorre
- **Solução:**
- Procedemos a calcular o earliest start de todos os nós, percorremos os mesmos outra vez, calculamos o tempo de espera e no final vemos quem teve o maior.

Cenário II

Questão -> 2.5

- **Complexidade Temporal:**

  6N*3E

N -> numero de nos

E -> numero de edges

- **Complexidade Espacial:**

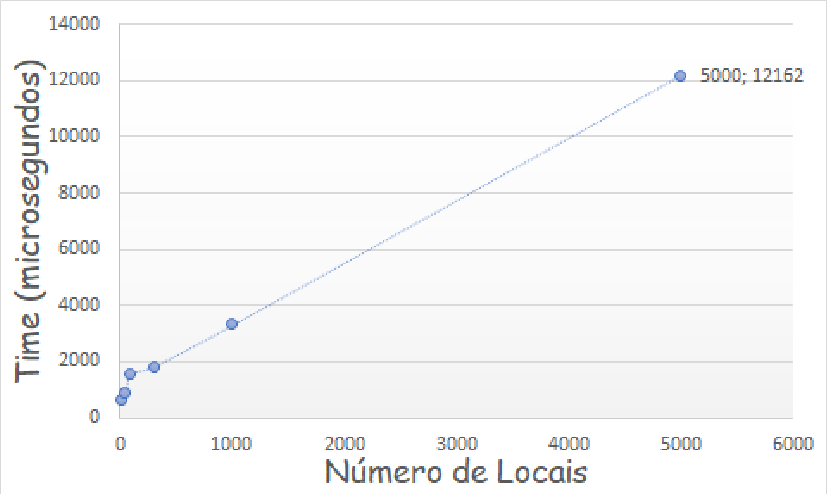
  2N

N -> numero de nós

Cenário II

Questão -> 2.5

- **Análise Empírica do Algoritmo:**



Autoavaliação do grupo

- André Costa: %
- Óscar Esteves: %
- Pedro Nunes: %

A discutir na aula