Trabalho Prático 1

Raissa Miranda Maciel
17 de Maio de 2022
Matrícula: 2020006965

1. Introdução

Nos dias de hoje, muitas situações exigem a formação de pares de acordo com preferências dos elementos a serem alocados. Essa documentação lida com o problema de alocar visitantes a bicicletas levando em consideração as bicicletas preferidas das pessoas e a distância até elas. A solução utiliza uma busca em largura para calcular as distâncias e o algoritmo Gale-Shapley para realizar os emparelhamentos estáveis. As seções estão divididas de forma a abordar a modelagem computacional do problema, as estruturas, os algoritmos e a análise de complexidade de tempo.

2. Modelagem Computacional do Problema

A primeira parte desse problema consiste em determinar uma lista de preferências das pessoas e das bicicletas. A preferência das pessoas é informada em forma de “notas” dadas a cada uma das bicicletas disponíveis no mapa em questão. Por outro lado, a preferência das bicicletas é relacionada a ordem crescente de distância até cada pessoa. Dessa forma, para construir essa série, é realizada uma *breadth-first search* no grafo obtido a partir do mapa lido da entrada,

O mapa é constituído de pessoas, bicicletas, obstáculos e caminhos livres. Para isso, ele é representado na forma de um grafo de n linhas e m colunas, tendo arestas que ligam qualquer vértice que não seja um obstáculo. Por exemplo, a figura 1 mostra as pessoas nos índices de cor rosa, as bicicletas nos roxos, os caminhos livres nos amarelos e os obstáculos são os cinzas. Esse grafo é lido como um vetor de n * m vértices e, a partir dele são calculadas as distâncias a partir dos seus índices vizinhos. Com a *bfs* para esse exemplo, teríamos calculado a distância de 3 arestas da bicicleta 0 até a pessoa a e a distância de 6 arestas da bicicleta 1 até a pessoa b. Assim, a lista de preferências da bicicleta 0 tem primeiro o visitante a seguido do visitante b. O casamento estável, cujo algoritmo é mostrado nas próximas seções, favorece a preferência das pessoas, uma vez que elas estão procurando pelas bicicletas, e, para esse exemplo, seriam formados os pares (a,1) e (b,0). Essas alocações formam um emparelhamento estável porque não possui um par (pessoa, bicicleta) que se prefira mutuamente e não estão juntos.

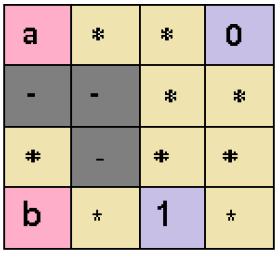
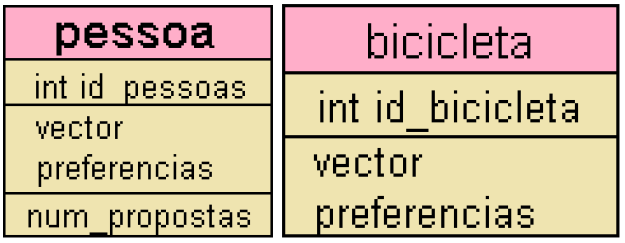


Figura 1

3. Estruturas de dados

Para a solução desse problema foram criadas as seguintes classes:



Essas duas classes terão os ids das pessoas e das bicicletas e o seu respectivo vetor de preferência em estruturas <vector>. Assim, todas as informações necessárias para a realização do emparelhamento entre elas está contido nesses objetos. Essa estrutura é também utilizada como auxiliar para outras partes do programa.

A *bfs* necessita de uma estrutura fila do tipo <queue> para calcular as distâncias entre as bicicletas e os visitantes.

Além disso, durante a execução do algoritmo Gale-Shapley, as pessoas são organizadas em uma estrutura de pilha do tipo <stack>.

4. Algoritmos

Os algoritmos implementados para a solução desse problema são mostrados a seguir:

● Breadth-First Search:

A busca em largura em grafos é utilizada para percorrer uma travessia no grafo a partir de um vértice fonte. Ele foi utilizado para calcular a distância entre bicicletas e visitantes e montar o vetor de preferências de cada bicicleta com uma *bfs* a partir de cada vértice que possui uma.

Sucintamente, a função *bfs* recebe um vértice de bicicleta e uma lista de adjacências que possui os vizinhos de cada vértice já previamente calculada. Uma fila armazenará, inicialmente, o vértice fonte e, na medida em que os vizinhos desse vértice forem visitados, eles são colocados na fila. A distância do vértice fonte até cada vértice é armazenado em um vetor, seguindo o fato de que, a distância dele até ele mesmo é 0 e, cada vizinho possui 1 aresta de distância a mais do vértice que o colocou na fila. Dessa forma, a chamada dessa função para cada vértice que possui uma bicicleta permitirá que as bicicletas tenham a distância até cada pessoa em um

vetor, que posteriormente terá o identificador dessas pessoas ordenados de acordo com a ordem crescente das distâncias aqui calculadas.

- **Gale-Shapley:**

O algoritmo Gale-Shapley é utilizado para encontrar um casamento perfeito entre duas entidades. Nesse contexto, são encontrados pares estáveis entre visitantes e bicicletas que seguem a lista de preferência de cada um deles e prioriza as preferências dos visitantes.
Sucintamente, a função *gale_shapley* recebe um vetor de pessoas e um de bicicletas, que possuem os elementos que serão realizados os emparelhamentos. Todas as pessoas são colocadas em uma pilha e, sua lista de preferência de bicicletas é analisada. Se a bicicleta não estiver alocada com nenhum visitante, esse par é formado. Se a bicicleta já estiver alocada com alguém, mas preferir o visitante que está propondo atualmente, o par dessa bicicleta é atualizado e a pessoa rejeitada é devolvida para a pilha. Por outro lado, se a bicicleta preferir a organização que já está feita, a pessoa irá propor para a próxima bicicleta do seu vetor de preferências.

5. Análise de complexidade de tempo assintótica

Essa seção faz referência à análise temporal dos principais algoritmos implementados.

- **Breadth-First Search:**

Esse algoritmo, representado pela função *bfs,* possui várias funções de atribuição, inicialização, retorno, além de métodos relacionados à fila em tempo O(1). Seja n a quantidade de vértices presentes no grafo (resultado da multiplicação do número de linhas e colunas do mapa). Essa busca em largura adiciona na fila os vizinhos dos vértices que já estão nela e, por isso, sua pior situação temporal seria se o vértice em questão conseguisse alcançar todos os vértices do grafo. Nesse caso, cada vértice seria visitado uma única vez (uma vez visitado, o vértice não é adicionado mais na fila) e a complexidade temporal do algoritmo é O(n).

- **Gale-Shapley**

Esse algoritmo, representado pela função *gale_shapley*, possui várias funções de atribuição, inicialização, retorno, além de métodos relacionados à pilha em tempo O(1). Seja n a quantidade de pessoas, ou seja, também a quantidade de bicicleta e de pares que serão formados, uma vez que o casamento é estável, a função começa adicionando todas as n pessoas na pilha, o que é feito em tempo O(n). Em seguida, as propostas são realizadas em tempo O(1) e, como as bicicletas podem trocar seus parceiros durante o algoritmo, o máximo de propostas feitas de acordo com a lista de preferência das pessoas é n * n = n^2. Entretanto, para decidir se a bicicleta prefere a pessoa que já está alocada ou a pessoa que está recebendo a proposta no momento, é chamada a função *prefere_a_pessoa,* que acessa a lista de preferências da bicicleta e procura qual dessas pessoas aparece primeiro em tempo O(n). Dessa forma, a complexidade temporal do algoritmo é O(n^3).

- **Função principal:**

Inicialmente, é criada a lista de adjacência e, para isso, são lidos todos os vértices presentes no arquivo de entrada em tempo O(n), sendo n o número de vértices e cada índice dessa lista é preenchido com os seus vizinhos em tempo O(1). Posteriormente, o cálculo das distâncias é feito com a chamada da função *bfs* para cada vértice que possui uma bicicleta. Assim, sendo m o número de bicicletas/alocações, esse laço possui complexidade $O(m*n^2)$. A ordenação da lista de preferências das bicicletas e das pessoas é feita com os métodos *ordena_distancias* e *ordena_pesos* que possuem complexidade $O(m^2)$. Como essa ordenação é feita para cada pessoa/bicicleta, a complexidade dessa parte é $O(m^3)$. Por fim, a função *gale_shapley* com complexidade calculada anteriormente $O(n^3)$ é chamada. Contabilizando todas essas etapas, o programa possui complexidade $O(m*n^2 +)$

6. Observações críticas

Minha interpretação sobre os pesos que as pessoas forneceram para as bicicletas não influencia na troca da alocação, em outras palavras, o valor não importaria, levando em consideração apenas qual bicicleta é preferida. Essa suposição alterou o resultado dos casamentos, uma vez que a lista de preferências da pessoa em relação às bicicletas foi criada com a interpretação explicada acima.

7. Instruções para execução:

Para realizar a instalação do programa, é necessário descompactar o arquivo 2020006965_RaissaMirandaMaciel.zip. Uma vez descompactado, basta acessar o diretório em que o programa foi armazenado:

> *cd <diretoriodestino>*

Agora, é necessário realizar a compilação e execução do programa informando um arquivo *.txt* de entrada:

> *g++ tp1.cpp -o tp01*
> *./tp01 < <arquivoteste.txt>*

Será mostrado na linha de comando os pares entre bicicletas e pessoas gerados.