Trabalho Prático 01

João Victor Taufner Pereira - 2017098315

[1]Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte - MG - Brasil

1. Introdução

O problema proposto nesse trabalho consiste em, a partir de uma entrada contendo pessoas e postos de vacinação, gerar uma saída que aloque essas aos postos, de forma que as vacinas possam ser distribuídas. Essa distribuição deve considerar os fatores idade, distância euclidiana das pessoas aos postos e a capacidade de cada posto. Além disso, deve ser a melhor alocação possível do ponto de vista das pessoas, ideia que será desenvolvida na próxima seção.

2. Implementação e Modelagem

O problema pode ser compreendido como um caso do Problema do Casamento Estável com entidades pessoas e postos. Entretanto, no contexto de vacinação descrito, há particularidades que devem ser mencionadas:

- Nem todas as pessoas serão necessariamente vacinadas, uma vez que a soma das capacidades dos postos pode ser inferior ao número total de pessoas. Da mesma forma, nem todas as vagas de vacinação serão ocupadas, existindo a possibilidade de haverem menos pessoas que vagas.

- Como as preferências da pessoas serão baseadas na distância dos postos relativas à sua localização, essas preferências irão variar para cada uma. Entretanto, todos os postos possuem a mesma lista de prefências, dada que a vacinação deve ser feita priorizando idades em ordem decrescente.

Dessa forma, uma solução se configura como estável se para toda pessoa p alocada a um posto c, não houver um posto c' mais próximo que aloque pessoas de menor idades ou que possua vagas disponíveis. Além disso, a existência de uma pessoa sem alocação implica que não há mais nenhuma vaga disponível em nenhum dos postos.

Uma outra questão acerca do problema é que as solução estável encontrada deve ser a melhor possível do ponto de vista das pessoas. Isso significa que a pessoa de maior idade sempre conseguirá ser vacinada no posto mais próximo a ela.

2.1. Entradas e Saídas do programa

As entradas e saídas do programa são as padrão do sistema (stdin e stdout).

- **Entradas do programa:** Inicialmente recebemos um inteiro n, o número de postos de vacinação, seguido por n linhas de tuplas de inteiros representando os dados de cada posto, sendo eles o número de vagas e a localização(x e y), respectivamente. Em seguida, recebemos o inteiro m, número de pessoas aplicando para serem vacinadas, seguido por m linhas

de tuplas de inteiros representando os dados de cada uma das pessoas, sendo eles a idade e a localização(x e y). Vale mencionar que a ordem em que as pessoas e postos aparecem na entrada são importantes, uma vez que essa é utilizada para situações de desempate(como mesma distância relativa no caso de postos e mesma idade no caso de pessoas).

- **Saída do programa:** A saída é composta por pares de linhas para cada posto em que ao menos uma pessoa foi alocadas, seguindo o seguinte formato: as linhas impares contém o identificador do posto em questão(advindo da ordem em que apareceu na entrada) e as linhas pares contém tuplas de inteiros espaçados representando os idenficadores(novamente baseado na ordem de aparição) de todas as pessoas alocadas para serem vacinadas nesse determiando posto.

2.2. Classes implementadas

2.2.1. Clinic

Classe que representa a entidade posto de vacinação. Possui apenas dois atributos, sendo o primeiro um inteiro que armazena a capacidade(número de vagas) e o segundo um par ordenado de inteiros que representa a localização. Seu único método é seu construtor, que tem como paramêtros todos os atributos citados. A classe também possui os seguintes métodos:

2.2.2. Person

A classe Person representa a entidade pessoa. Como atributos tem um inteiro para a idade e um para a posição em que apareceu na entrada(sua utilidade será explicada posteriormente), além do par ordenado de inteiros para a localização. A classe também possui os seguintes métodos:

- **O construtor** da classe que recebe todos os atributos citados.
- **O método *comparePeople (Person p1, Person p2)*** que recebe como parâmetro dois objetos do tipo Person. Retorna *true* caso *p1* tenha idade maior que *p2* ou caso possuam a mesma idade e *p1* tenha aparecido primeiro na entrada.

2.3. Estruturas de Dados

Para a implementação do algorítmo, foi utilizado vetor dinâmico da classe *vector* para armazenar os objetos de tipo *Person* e outro para os objetos do tipo *Clinic* lidos na entrada. Como esses containers serão iterados do início ao fim, a escolha da classe *vector* se justifica. Além disso, o container *vector* também foi utilizado em forma bidimensional em ***vacinnatedPeople***. Dados i postos, essa estrutura contém i vetores, cada um para representar as pessoas que foram alocadas em um determinado posto. A escolha da estrutura se justifica pela intenção de iterar por todos os elementos(para imprimí-los) e pelo acesso $O(1)$ a esses vetores.

2.4. A função *galeShapley*

Trata-se da função onde ocorre efetivamente a alocação de pessoas aos postos, de forma que a solução obtida seja a melhor possível do ponto de vista as pessoas. Essa função recebe três parâmetros: o vetor *People* contendo todas as pessoas, o vetor *Clinics* contendo todas os postos e o vetor bidimensional *vaccinatedPeople* inicialmente vazio que será populado. A função segue os seguintes passos, para *m* postos e *n* pessoas:

- Ainda antes do início da execução da função propriamente dita, uma etapa importante ocore: o vetor de pessoa, que é um dos parâmetros da função, é ordenado em ordem decrescente de idade. Em caso de idades iguais, o vetor ordena pessoas pela ordem que elas apareceram na entrada.

- Começamos a iterar sobre o vetor de pessoas. Para cada uma, é calculada a distância entre ela e cada um dos *m* postos, de forma a determinar qual é o posto mais próximo.

- Assim que é encontrado o posto mais próximo, o índice da pessoa(posição que ela apareceu na entrada) é inserido no vetor de *vaccinatedPeople* correspondente a esse posto. Além disso, o número de vagas desse posto é decrescido em 1 no vetor *Clinics*

- Os procedimentos acima são repetidos até que todo o vetor de pessoas *People* seja percorrido.

Algorithm 1: Função ***galeShapley***

input : Um vetor ordenado de pessoas, um vetor de postos e um vetor bidimensional que receberá as alocações

distancia $\leftarrow \infty$
foreach $i \in$ *pessoas* **do**
  foreach $j \in$ *postos* **do**
    if *capacidade* $>$ *0 and distancia_atual* $<$ *distancia* **then**
      distancia = distancia_atual
  if *Um posto i da forma especificada foi encontrado* **then**
    Insira a pessoa no vetor *i* do vetor bidimensional
    capacidade $\leftarrow$ capacidade $-$ 1

2.4.1. O algoritmo e sua corretude

O algoritmo apresentado é um caso específico do Algorítmo de Gale-Shapley em que as preferências de todas as entidades de um tipo(no caso Posto) são as mesmas. Isso possibilitou que a ordenação prévia do vetor de pessoas simplificasse o problema, uma vez que uma pessoa alocada a um posto jamais precisaria ser desalocada para dar lugar a outra. Por essa razão, o formato do algrorítmo difere um pouco do mencionado no livro *Algorithm Design*. Além disso, as listas de preferência de cada pessoa foram calculadas sob demanda, somente quando o posto sendo analisado possui vagas, o que diminui o número de operações.

Dessa forma, considere a execução do algorítmo num cenário em que há *n* pessoas e *t* vagas distribuidas entre *m* postos. Inicialmente, a pessoa considerada pelo algorítmo será a mais velha, e ela será alocada ao posto de sua preferência. Para cada iteração subsequente, o mesmo ocorrerá com a próxima pessoa mais velha, até que acabem todas as vagas ou que todas as pessoas sejam alocadas. Esse cenário sempre será o melhor possível para todas as pessoas, já que elas sempre escolherão a melhor opção disponível no momento. E como todo posto tem a mesma ordem de preferência baseada nas idades, nunca ocorerrá de um posto preferir uma pessoa mais jovem que teve que se contentar com uma opção de menor preferência. Portanto, o algorítmo sempre retornará soluções estáveis.

3. Análise de Complexidade

Quatro operações ocorrem no projeto: a leitura da entrada,a ordenação do vetor de pessoas, a alocação das pessoas aos postos e a impressão da saída. Considere M como o número de postos da entrada e N como o número de pessoas.

- **Leitura:** A leitura da entrada é feita por duas funções: ***readClinic*** e ***readPeople***. A função ***readClinic*** executa sua primeira operação para obter ***M***, e em seguida passa a executar M iterações. A função ***readPeople*** faz exatamente o mesmo, mas com respeito a N. Dessa forma, os limites superiores dessas funções são respectivamente $O(M)$ e $O(N)$.

- **Ordenação do vetor de pessoas:** Para essa ordenação, foi utilizada a função ***sort*** da biblioteca padrão de C++. Essa função possui complexidade da forma $O(N * log(N))$.

- **Alocação das pessoas aos postos:** As operações envolvidas na alocação são executadas iterando por todas as N pessoas para cada um dos M postos, ou seja, ocorrem $M * N$ iterações. Por mais que operações importantes como o cálculo da distância não ocorram necessáriamente em todas as iterações, é seguro afirmar que a alocação possui complexidade na forma $O(N * M)$.

- **Impressão:** A impressão ocorre executando dois laços, em que um itera pelos postos e outro por cada pessoa que será vacinada. Como o número de pessoas vacinadas jamais séra superior a N, é possível afirmar que a complexidade da operação de impressão é limitada superiormente por $O(N * M)$.

Na função main, chamamos cada uma das operações descritas apenas uma vez, o que ocasiona na complexidade geral: $O(M) + O(N) + O(N * log(N)) + O(N * M) + O(N * M)$. Portanto, a complexidade geral é da forma $O(N * M)$.

4. Conclusões

A partir da implementação do trabalho foi possível colocar em prática os conhecimentos adquiridos na disciplina a respeito do Problema do Casamento Estável. Dessa forma, por meio do Algoritmo de Gale-Shapley foi possível encontrar uma solução estável para o problema que fosse a melhor possível para todas as pessoas.

5. Bibliografia

- Kleinberg, Jon; Tardos, Eva. Algorithm Design, Pearson Education India, 2006