FUNDAÇÃO GETÚLIO VARGAS

ESCOLA DE MATEMÁTICA APLICADA

MESTRADO 2015.1

ESTRUTURA DE DADOS E SEUS ALGORITMOS

Prof Alexandre Rademaker

Resolução dos Exercícios Selecionandos do Capítulo 1

GRUPO:

KIZZY TERRA

GUSTAVO AVILA

CAROLINE FIALHO

RIO DE JANEIRO

MARÇO DE 2015

1 Exercício 1

Falso ou verdadeiro? "Em toda instância do problema de pareamento estável, existe um pareamento estével contendo um par (m, w) tal que m está ranqueado primeiro na lista de preferências de w e w está ranqueado primeiro na lista de preferências de m."

A afirmação é falsa. Contraexemplo:

Considere $w_1, w_2, w_3, ..., w_n$ um conjunto de mulheres e $m_1, m_2, m_3, ..., m_n$ um conjunto de homens. Suponha a seguinte instância:

1) Para todo homem m_i tal que$1 \leq i \leq n$, a primeira preferência da lista é a mulher w_i.

2) Para toda mulher $w_i$tal que$1 \leq i \leq n-1$, a primeira preferência da lista é o homem m_{i+1}e a primeira preferência da lista da mulher w_né o homem m_1.
Um pareamento instável para esta instância poderia ser:

$$(w_1, m_2),\ (w_2, m_3),\ (w_3, m_4),\ (w_4, m_5),\ ...\ ,\ (w_{n-1}, m_n),\ (w_n, m_1)$$

Este pareamento seria tal que todas as mulheres estão com sua primeira preferência da lista e nenhum homem está com a primeira preferência da lista e ainda assim não existiriam pares de instabilidades, pois não haveria um homem e uma mulher que trocariam seus pares para ficar juntos.

2 Exercício 3

1) Iremos construir um algortimo que retorna um par de agendas (S, T) válido, dados P e Q dois conjuntos de n programas de TV com suas respectivas avaliações.

Uma solução para o problema consiste no par (S, T) tal que S consiste na lista de todos os n programas de A ordenados e da mesma forma, T consiste na lista de todos os n programas de B ordenados.
O esquema abaixo, facilira o entendimento dessa propriedade se S e T:

$S(A)$	$T(B)$
s_1	t_1
s_2	t_2
s_3	t_3
s_4	t_4
...	...
s_n	t_n

tal que $s_1 \leq s_2 \leq s_3 \leq ... \leq s_n$e $t_1 \leq t_2 \leq t_3 \leq ... \leq t_n$.
A idéia do algoritmo seria construir uma agenda a partir de S e T, escolhendo o programa com maior avaliação entre s_i e $t_i$para ocupar o horário i.

2)Se S e T acima forem agendas de A e B, respectivamente ordenadas segundo a avaliação dos seus programas de TV então S e T sempre será estável.

Prova por absurdo:

Suponha que S e T com a propriedade explicada acima não formam um par estávelm isso implica que deve haver:
caso 1: S' tal que S' vence mais horários na agenda final do que S em relação a T ou
caso 2: T' tal que T' vence mais horários na agenda final do que T em relação a S.

Analisando o caso 1:

Para que exista S' melhor do que S em relação a T, deve ser possível rearranjar os elementos de S de forma a vencer mais horários da agenda. Para que seja possível vencer mais horários então deve existir pares tal que $s_i < t_i$. Para que S' exista deve ser possíver trocar estes s_i por algum s_k tal que $s_k > t_i$ e $s_i > t_k$. Entretanto, podemos observar que isso não é possível analisando S e T do item 1, pois se existir $s_k$maior do que s_i, $s_k$será maior do que t_i, mas $s_i$será menor do que t_k, uma vez que $s_i < t_i$< t_k.
Logo, não é possível rearranjar S e vencer mais horários na agenda neste caso 1.

Analisando o caso 2:

Analogamente ao caso 1, não é possível rearranjar T de forma a construir um T' tal que T' vence mais horários em relação a S.

3) Algoritmo para calcular par (S, T)

S' ← mergesort (S(A))
T' ← mergesort (T(B))
retornar (S', T')

4) Algoritmo de ordenação

Iremos utilizar o algoritmo de ordenação merge-sort recursivo (a implementação deste algoritmo está na solução dos exercícios do cap. 2, aqui iremos apresentar apenas o pseudo-código):

```
mergesort(L[1,...n]):
Se n > 1:
    retornar merge(mergesort(a[1 ... n/2]), mergesort(a[n/2+1, ... , n]))
Se não:
    retornar L

merge(x[1... k], y[1 ... l])
Se k = 0: retornar y[1 ... l]
Se l = 0: retornar x[1... k]
Se x[1] ≤y[1]:
    retornar x[1]°merge(x[2 ... k], y[1 ... l])
Se não:
    retornar x[1]°merge(x[1... k], y[2 ... l])
```

3 Exercício 4

a) Algoritmo

Inicialmente todos os estudantes estão livres e todas as vagas também.
Enquanto houver algum estudante *e* que está sem vaga e ainda não aplicou para todos os hospitais:

Escolha um estudante e :
O estudante aplica para o primeiro hospital h , segundo sua lista de preferência, para o qual ainda não aplicou:
Se o hospital h possuir vagas:
O estudante e consegue uma vaga
Se naõ:
Escolhe-se entre os estudantes que estão ocupando as vagas um estudante e' que possui a preferência mais baixa na lista do hospital h:
Se o estudante e tiver maior preferência do que e' :
O estudante e fica com a vaga do estudante e'
Se não:
O estudante e continua sem vaga

b) Queremos mostrar que para toda saída S do algoritmo acima composta pelo pareamento entre os estudantes e os hospitais, S é um pareamento estável:

Prova por absurdo:
Suponha que S possui uma instabilidade (e', h) , ela pode ser de dois tipos segundo o enunciado do problema:

Tipo 1:
e tem uma vaga de h e
e' está sem nenhuma vaga e $\Rightarrow (e', h)$ é um par de instabilidade
h prefere e' a e.

Se e'ficou sem nenhuma vaga então, pelo algoritmo acima, e' aplicou para todos os hospitais e foi negado por todos, em outras palavras e' não possuía preferência maior do que os estudantes que já ocupavam as vagas dos hospitais. Portanto, não poderia existir hospital h que prefere e' a e e (e', h) não pode ser uma instabilidade do tipo 1.

Tipo 2:
e tem uma vaga de h e
e' tem uma vaga de h'e $\Rightarrow (e', h)$ é um par de instabilidade
h prefere e' a e e
e' prefere h a h'.

Supondo que e'prefere h a h'e e' terminou com uma vaga de h', pelo algoritmom acima:

1) e' aplicou para h antes de aplicar para h' e h preferiu outro estudante e''a e', rejeitando a vaga para e' . Logo, h não prefere e' e (e', h) não pode ser uma instabilidade do tipo 2.

2) Da mesma forma, supondo que h prefere e' a e e sabendo que e' terminou com h', então e' não chegou a aplicar para h e portanto e' prefere h' a h e (e', h) não pode ser uma instabilidade do tipo 2.