Documentação Trabalho Prático I

Algoritmos I

Nome: Matheus Schimieguel Silva
Matrícula:2017014413

1 Introdução

1.1 Identificação do problema

O problema descrito trata de um exemplo de casamentos estáveis em que os estudantes escolhem as a universidades em que desejam estudar com uma ordem de preferência e essas por sua vez possuem preferência de alunos .Cada universidade possui uma nota de corte ’nc’ e um número de vagas ‘v’ .Os alunos só são admitidos na universidade se possuem uma nota ‘n’ maior ou igual a ‘nc’ da universidade inscrita e se ele está entre os ‘v’ alunos com maiores notas.

Nas instruções do problema pede-se que haja maior satisfação possível dos estudantes ,portanto eles devem ser o foco da resolução e nesse sentido eles devem ser ter o poder da escolha priorizado em detrimento das universidades.

As soluções retornadas pelo algoritmo deve priorizar a escolha dos estudantes de maior a nota a fim de não retorna uma resposta com casamentos instáveis.

1.2 Proposta de Solução do problema

A solução desse problema se dá pela adaptação do algoritmo de Gale-Shapley.Em que os estudantes se inscrevem nas universidades que desejam estudar, de modo que assim os estudantes sejam favorecidos e tenham uma maior satisfação .A Fim de se evitar casamentos instáveis os estudantes de maior nota fazem sua inscrição nas universidades preferidas começando pelas universidades com maior preferência .Assim cada universidade terá sua lista de candidatos inscritos ordenada pela nota dos alunos. Basta portanto cada universidade desclassificar os estudantes inscritos que possuem nota menor que a nota de corte ,os alunos que já foram aprovados e os alunos que não estejam entre os ‘v’ alunos inscritos com maior nota.

Os alunos restantes na lista de inscrição de cada universidade são os alunos aprovados e nas respectivas Universidade.Os alunos que não estejam nas lista de alguma universidade são aqueles que não serão alocados.

2. Estrutura de Dados e Algoritmos Utilizados

A Estrutura de dados utilizada para representar os alunos e as Universidades foram foram structs do tipo Aluno e Universidade.Cada estrutura do tipo Aluno possui três números inteiros que representam o seu identificador,sua nota,e o número de universidades que se inscreveu além de um vetor de inteiros que armazenam os identificadores das universidades em que se aplicou.

2.1 Estrutura de Dados

Cada estrutura do tipo universidade possui quatro números inteiros que representam o seu identificador,a nota de corte,o número de vagas, número de alunos inscritos,o número de alunos aprovados além de um vetor de inteiros com o identificador dos alunos inscritos que ao final do algoritmo guardará os identificadores dos alunos aprovados.

2.2 Algoritmo

O primeiro passo é instanciar os alunos e as universidades em respectivamente um vetor de estruturas do tipo Aluno e um outro vetor vetor de estruturas do tipo Universidade.

Em seguida deve-se ordenar os alunos em função da nota .Em caso de alunos com mesma nota um aluno com id menor deve ter prioridade.

Para ordenar o vetor Alunos foi usado o algoritmo de ordenação mergesort.
A partir do vetor ordenado de Alunos o aluno de maior nota tenta se inscrever na nas universidades que estejam em sua lista em ordem decrescente de prioridade.Caso a nota do aluno seja maior ou igual nota de corte e a quantidade de alunos aprovados nessa universidade seja menor que o número de vagas ,o aluno é aprovado nessa Universidade e não é preciso testar para as universidades restantes.Caso contrário o aluno não é aprovado .Caso o aluno não seja aprovado em nenhuma universidade então ele ficará desalocado .Passa-se para o próximo Aluno com maior nota repetindo os procedimentos descritos para o aluno anterior até chegar no aluno de menor nota.

Ao final desse procedimento cada universidade terá uma lista de aprovados que serão os grupos com alocação.Tais alocações será a que maximiza a satifaçao dos candidatos e que não possui instabilidades.

3.Análise De Complexidade

seja ‘n’ o número de alunos e ‘m’ o de universidades

3.1 Tempo

3.1.1 instanciação das Estruturas de dados

seja f(n,m) a quantidade de instruções para inicializar os Alunos
seja g(m,n) a quantidade de instruções para inicializar as Universidades
Seja A,B,C,D, constantes;

os pior caso tanto para inicializar os alunos quanto às universidade é quando todos os alunos se inscrevem em todas as universidades .

Nesse caso ‘f’ e ‘g’ são dados por :

$f(n,m) = Cn + Dnm$ $g(m,n) = Am+Bmn$

f(n,m) = O (n+nm) g(m,n) = O(m + mn)

3.1.2 Mergesort

A ordenação do vetor de alunos utiliza um algoritmo mergesort que é um tipo de ordenação por divisão e conquista.

O algoritmo mergesort faz duas chamadas recursivas para vetores com metade do tamanho do vetor da chamada atual,além de chamar a função merge.

Por se tratar de um algoritmo recursivo para calcular o custo de operações será necessário equações de recorrências.
Analisando a função merge pode-se ver que em relação ao tempo ela é O(n).
Assim a equação de recorrência do mergesort é :
T(n) = 2T(n/2) + O(n)

Com a =2 e b = 2 pelo segundo caso do teorema mestre temos que
h(n) = T(n) = Θ((n ^logb a)* log n) = O(nlog)
h(n) = O(nlogn)

3.1.2 Inscrição Aprovação dos Alunos

O pior caso é quando todos os estudantes se inscrevem em todas as universidades e o algoritmo verifica se o candidato foi aprovado em todas as universidades .

Seja i(n,m)a quantidade de instruções para encontrar os aprovados seja K uma constante
i(n,m) =Kmn
i(n,m) = O (mn)

3.1.3 Limite Assintótico Total Do Algoritmo Em Relação Ao Tempo

Seja l(n,m) a função que define o custo em processo do algoritmo
i(n,m) = O(f(n,m)+g(m,n)+ h(n) + i(n,m))
i(n,m) = O(n+nm + m + mn + nlogn + mn)
i(n,m) = O(n+m+mn+nlogn)

3.2 Espaço

3.2.1 Alunos

Seja p(n,m) o espaço ocupado pelos alunos proporcional à ‘n’ e ‘m’.
p(n,m) = O(n + nm)

3.2.2 Universidades

Seja q(n,m) o espaço ocupado pelos Universidades proporcional à ‘n’ e ‘m’.
q(n,m) = O(m + nm)

3.2.3 Mergesort

Seja r(n) a quantidade de espaço usado pelo algoritmo em função de n.
Seja T(n) = T(n -1) + c a equação de recorrência para r(n).
quando T(1) = d.
logo
T(n) = O(n)
r(n) = O(n)

3.2.4 Limite Assintótico Total Do Algoritmo Em Relação ao espaço

t(m,n) = O(n + mn + m + nm + n)
t(m,n) = O(n + m + mn)