

Período: 2005.2

2 de setembro de 2005

Horário: 2as-feiras e 4as-feiras de 9 às 11 horas

ANÁLISE DE ALGORITMOS (INF 1721)

1º Trabalho de Implementação (T1)

Data de Entrega: 5 de Outubro

Descrição

Este trabalho prático consiste em desenvolver códigos para diferentes buscas em matrizes de inteiros ordenadas nas linhas e nas colunas (ver descrição abaixo).

Considere uma matriz quadrada M contendo n números inteiros (para efeito de cálculo de complexidade teórica, você pode assumir que $n = (2^k)^2$ para algum k inteiro). Esta matriz é dada ordenada tanto nas linhas como nas colunas (veja o exemplo abaixo, para $n = 64$, k seria 3). Considere agora o seguinte problema: Dado um número inteiro i , deseja-se saber se este

$$\begin{bmatrix} 4 & 10 & 21 & 34 & 39 & 40 & 57 & 62 \\ 9 & 17 & 28 & 35 & 41 & 42 & 58 & 72 \\ 19 & 20 & 30 & 47 & 49 & 49 & 65 & 73 \\ 27 & 39 & 40 & 48 & 52 & 60 & 66 & 79 \\ 33 & 43 & 44 & 58 & 59 & 61 & 77 & 81 \\ 46 & 46 & 60 & 63 & 69 & 71 & 79 & 88 \\ 56 & 61 & 62 & 68 & 72 & 76 & 87 & 88 \\ 64 & 67 & 69 & 73 & 83 & 89 & 90 & 91 \end{bmatrix}$$

se encontra na matriz M . Analise a complexidade (de pior caso) dos algoritmos abaixo para resolver este problema. A complexidade deve ser em função de n .

1. Algoritmo I

Passo 0: Seja l uma linha de M . Faça $l = 0$.

Passo 1: Execute uma Busca Binária para encontrar i na linha l de M . Se i for encontrado, responda SIM e PARE. Senão, incremente l .

Passo 2 Se l é uma linha de M vá para o passo 1. Senão, responda NÃO e PARE.

2. Algoritmo II

Passo 0: Seja (p, q) a posição central de M , que divide M em M_1 , M_2 , M_3 e M_4 , onde M_1 é a submatriz superior esquerda, M_2 a superior direita, M_3 a inferior esquerda e M_4 a inferior direita, de tamanhos iguais. Se M_1 for vazia, Responda NÃO e PARE. Senão vá para o passo 1.

Passo 1 Se i for igual a $M(p, q)$, responda SIM e PARE. Senão vá para o passo 2.

Passo 2: Se i for maior que $M(p, q)$ execute Alg. II para as matrizes M_1 , M_2 , e M_3 . Senão execute Alg. II para M_2 , M_3 e M_4 .

3. Algoritmo III

Passo 0: Seja (p, p) uma posição da diagonal de M .

Passo 1: Busque sequencialmente na diagonal a posição p tal que $M(p, p) < i < M(p+1, p+1)$ utilize essa posição para obter quatro matrizes a partir de M : M_1 , M_2 , M_3 e M_4 (observe que o pior caso é o que tem estas 4 matrizes de tamanhos iguais). Se M_1 , M_2 , M_3 e M_4 forem vazias, Responda NÃO e PARE. Senão vá para o passo 2.

Passo 2 Se i for igual a $M(p, p)$ ou $M(p + 1, p + 1)$, responda SIM e PARE. Senão vá para o passo 3.

Passo 3: Execute Alg. III para as matrizes M_2 e M_3 .

4. Algoritmo IV

Passo 0: Seja (p, p) uma posição da diagonal de M .

Passo 1: Utilize *Busca Binária* na diagonal para encontrar a posição p tal que $M(p, p) < i < M(p + 1, p + 1)$ utilize essa posição para obter quatro matrizes a partir de M : M_1 , M_2 , M_3 e M_4 (observe que o pior caso é o que tem estas 4 matrizes de tamanhos iguais). Se M_1 , M_2 , M_3 e M_4 forem vazias, Responda NÃO e PARE. Senão vá para o passo 2.

Passo 2 Se i for igual a $M(p, p)$ ou $M(p + 1, p + 1)$, responda SIM e PARE. Senão vá para o passo 3.

Passo 3: Execute Alg. IV para as matrizes M_2 e M_3 .

Os algoritmos acima devem ser desenvolvidos e seu desempenho experimental deve ser analisado com respeito ao tempo de CPU. O desenvolvimento destes códigos e a análise devem seguir os seguintes roteiros:

- Descrever os algoritmos informalmente.
- Demonstrar o entendimento do algoritmo explicando, em detalhe, o resultado que o algoritmo deve obter e justificá-lo.
- Explicar a fundamentação do algoritmo e justificar a sua corretude. Apresentar e explicar a complexidade teórica esperada para cada algoritmo.

- Apresentar gráficos com os tempos de CPU para a execução de cada um dos casos teste (observe que tempos de CPU inferiores à 5s não são confiáveis, nesse caso repita a execução do teste até que o tempo ultrapasse 5s, divida então o tempo de CPU obtido pelo número de execuções que foram necessárias);
- Documente o arquivo contendo o código fonte de modo que cada passo do algoritmo esteja devidamente identificado e deixe claro como este passo é executado.

A corretude código será testada sobre instâncias de tamanho 64, 256, 1024, 2304, 4096, 9216, 12544 e 16384, que correspondem à matrizes quadradas de dimensão 8X8, 16X16, 32X32, 48X48, 64X64, 96X96, 112X112 e 128X128, respectivamente. O conteúdo das matrizes com n elementos serão os primeiros n inteiros ímpares. As buscas serão sempre por números pares, ou seja, será sempre um “pior caso”. O elemento a ser buscado em cada caso teste será $n/64$, $n/32$, $n/16$, $n/8$, $n/4$ e $n/2$. Ou seja, deverão ser testadas 6 buscas para cada valor de n (um total de 48 testes).

Incentiva-se mais testes com valores ainda maiores de n (256X256, 512X512, ...).

O trabalho entregue deve conter:

- Um documento contendo o roteiro de desenvolvimento dos algoritmos (e dos códigos), os itens pedidos acima, comentários e análises sobre a implementação e os testes realizados (papel).
- A impressão dos códigos fonte (papel).
- Um e-mail contendo os códigos fonte, os executáveis e o documento em versão eletrônica, correspondentes (no e-mail é obrigatório o uso do ASSUNTO (ou SUBJECT) AA052T1).
- O trabalho pode ser feito em grupo de até 2 alunos.

Observação:

No algoritmos II, III e IV, pode acontecer de a matriz entrada do algoritmo não seja quadrada. Isso NÃO é restritivo. Nesse caso, a posição central é dada pela média do índice inicial e final em cada dimensão. Considere também que a diagonal possui um elemento em cada uma das linhas ou colunas, aquele que acontecer em menor número. Os elementos da diagonal estarão espaçados homogeneamente na outra dimensão. Exemplo: Matriz retangular com posição inicial (1,1) e final (5,10). A diagonal será: (1,1), (2,4), (3,6), (4,8) e (5,10).

As implementações dos algoritmos TÊM que tratar esse caso.