Universidade Federal do Ceará – UFC
Centro de Ciências – CC
Mestrado e Doutorado em Ciências da Computação - MDCC
Estruturas de Dados Avançadas

Exercício: Complexidade de Algoritmos
Objetivos: Exercitar os conceitos de Complexidade de Algoritmos.
Data da Entrega: 04/04/2017

OBS 1: Exercício Indivudual.
OBS 2: A entrega desta lista deverá ser executada via SIGAA.

NOME: **Renan Pereira de Figueiredo** **MATRÍCULA:** **396921**

Questão 1

As funções *f(n)* mostradas abaixo fornecem o tempo de processamento *T(n)* de um algoritmo resolvendo um problema de tamanho *n.* Complete a tabela abaixo colocando, para cada algoritmo, sua complexidade (*O maiúsculo*) e a ordem do mais eficiente para o menos eficiente. Em caso de empate repita a ordem (por exemplo: 1º ,2º, 2º, ...).

f (n)	$O(\ldots)$	*ordem*
$5 + 0.001n^3 + 0.025n$	$O(n^3)$	**9**
$500n + 100n^{1.5} + 50n \log_{10} n$	$O(n^{1.5})$	**5**
$0.3n + 5n^{1.5} + 2.5 \cdot n^{1.75}$	**O($n^{1.75}$)**	**6**
$n^2 \log_2 n + n(\log_2 n)^2$	**O(n^2 log n)**	**8**
$n \log_3 n + n \log_2 n$	**O(n log n)**	**2**
$3 \log_8 n + \log_2 \log_2 \log_2 n$	**O(log n)**	**1**
$100n + 0.01n^2$	**O(n^2)**	**7**
$0.01n + 100n^2$	**O(n^2)**	**7**
$2n + n^{0.5} + 0.5n^{1.25}$	**O($n^{1.25}$)**	**4**
$0.01n \log_2 n + n(\log_2 n)^2$	**O(n(logn)^2)**	**3**
$100n \log_3 n + n^3 + 100n$	**O(n^3)**	**9**
$0.003 \log_4 n + \log_2 \log_2 n$	**O(log n)**	**1**

Questão 2

Os algoritmos abaixo são usados para resolver problemas de tamanho n. Descreva e informe para cada algoritmo sua complexidade no pior caso (O maiúsculo/Ômicron). Tente entender o problema antes de apresentar uma resposta.

a)

```
for ( i=1; i < n; i *= 2 ) {
      for ( j = n; j > 0; j /= 2 ) {
            for ( k = j; k < n; k += 2 ) {
                  sum += (-j * k) << i/2;
            }
      }
}
```

No laço *for* mais externo temos que o número de passo é reduzido à metade em cada interação pois a variável i sempre dobra de valor (i*=2), logo a complexidade seria O(log n). No laço do meio temos também a mesma situação, pois a variável j reduz seu valor a metade em cada interação (j/=2), assim a complexidade também é O(log n). No último laço o tempo de execução seria de n/2 considerando que a variável k é sempre incrementado de 2 a cada passo, mas como n tendendo a infinito temos a complexidade O(n). Como os laços estão aninhado temos que a complexidade do algoritmo igual ao produto da complexidade de cada laço, ou seja:

$O(\log n \times \log n \times n) = O(n(\log n)^2)$.

b)

```
Leia(n);
x ← 0
Para i ← 1 até n faça
      Para j ← i+1 até n faça
            Para k ← 1 até j-i faça
                  x ← x + 1
```

No primeiro laço, temos uma complexidade de O(n). Na segunda interação, mesmo j começando recebendo i +1, para n tendendo ao infinito temos um custo de O(n) também. No último laço, analisando o pior caso, quando o j = n e i = 1, temos que o custo também O(n). Logo o custo do algoritmo é $O(n^3)$.

Questão 3

Suponha um algoritmo A e um algoritmo B com funções de complexidade de tempo $a(n) = n^2 - n + 549$ e $b(n) = 49n + 49$, respectivamente. Determine quais são os valores de n pertencentes ao conjunto dos números naturais para os quais A leva menos tempo para executar do que B.

$a(n) < b(n) \Rightarrow n^2 - n + 549 < 49n + 49 \Rightarrow n^2 - 50n + 500 < 0$

$\Delta = (-50)^2 - 4*1*500 = 2500 - 2000 = 500$

$$\boldsymbol{n' = \frac{-(-50)+\sqrt{500}}{2*1} = n' = \frac{50+22{,}36}{2} = n' = 36{,}18}$$

$$\boldsymbol{n' = \frac{-(-50)-\sqrt{500}}{2*1} = n' = \frac{50-22{,}36}{2} = n' = 13{,}82}$$

Logo, considerando somente os números naturais, a(n) < b(n) para 13 < n < 37.

<u>Questão 4</u>

O Casamento de Padrões é um problema clássico em ciência da computação e é aplicado em áreas diversas como pesquisa genética, editoração de textos, buscas na internet, etc. Basicamente, ele consiste em encontrar as ocorrências de um padrão P de tamanho m em um texto T de tamanho n. Por exemplo, no texto T = "PROVA DE AEDSII" o padrão P = "OVA" é encontrado na posição 3 enquanto o padrão P ="OVO" não é encontrado. O algoritmo mais simples para o casamento de padrões é o algoritmo da "Força Bruta", mostrado abaixo. Analise esse algoritmo e responda: Qual é a função de complexidade do número de comparações de caracteres efetuadas no melhor caso e no pior caso. Dê exemplos de entradas que levam a esses dois casos. Explique sua resposta!

```
#define MaxTexto 100
#define MaxPadrao 10

/* Pesquisa o padrao P[1..m] no texto  T[1..n] */
void ForcaBruta(   char T[MaxTexto], int n,
                   char T[MaxPadrao], int m)
{
    int i,j,k;
    for( i = 0 ; i < n - m + 1 ; i++ )
    {
         k = i;
         j = 0;
         while ( ( j <= m ) && ( T[k] == P[j] ) )
         {
              j = j + 1;
              k = k + 1;
         }
         if (j > m)
         {
              printf("Casamento na posicao %d",i);
              break;
         }
    }
}
```

Temos que o melhor caso seria quando o padrão P é encontrado logo na primeira posição. Assim, o primeiro laço "*for*" só executará um única vez, tendo assim a complexidade $\Omega(1)$. O segundo laço "while", como faz uma comparação para checar se o padrão foi encontrado, terá que executar m vezes, sendo $\Omega(m)$. Logo temos a complexidade do algoritmo no melhor caso de $\Omega(1*m) = \Omega(m)$. Ex: T = "123456789" e P = "123".

No pior caso temos o texto sendo uma sequência de caracteres repetidos e o padrão como sendo a mesma sequência de caracteres com apenas o último caractere diferente, não sendo possível encontrar o padrão no texto. Assim, o primeiro laço "*for*" terá que executar n-m+1 vezes, com n tendendo ao infinito temos uma complexidade de O(n). O segundo laço "*while*", será necessário executar m vezes, logo temos um custo O(m). Dessa foram, o custo total do algoritmo no pior caso é de O(nm). Ex: T = "11111111111111" e P = "112".

Questão 5

Considere que você tenha um problema para resolver e duas opções de algoritmos. O primeiro algoritmo é quadrático tanto no pior caso quanto no melhor caso. Já o segundo algoritmo, é linear no melhor caso e cúbico no pior caso. Considerando que o melhor caso ocorre 90% das vezes que você executa o programa enquanto o pior caso ocorre apenas 10% das vezes, qual algoritmo você escolheria? Justifique a sua resposta em função do tamanho da entrada.

Para entradas relativamente pequenas, em que é viável executar um algoritmo quadrático, eu escolheria o primeiro algoritmo. Dessa forma o algoritmo pode ser executado em tempo viável, para todas as entradas, diferente do segundo algoritmo, cuja a possibilidade de o custo ser cúbico pode inviabilizar sua utilização. Contudo, se as entradas forem muito grandes, a ponto de não ser viável a execução de um algoritmo quadrático, escolheria o segundo algoritmo, pois a grande possibilidade de um custo linear pode tornar viável sua execução.

Questão 6

Perdido em uma terra muito distante, você se encontra em frente a um muro de comprimento infinito para os dois lados (esquerda e direita). Em meio a uma escuridão total, você carrega um lampião que lhe possibilita ver apenas a porção do muro que se encontra exatamente à sua frente (o campo de visão que o lampião lhe proporciona equivale exatamente ao tamanho de um passo seu). Existe uma porta no muro que você deseja atravessar. Supondo que a mesma esteja a n passos de sua posição inicial (não se sabe se à direita ou à esquerda), elabore um algoritmo para caminhar ao longo do muro que encontre a porta em O(n) passos. Considere que n é um valor desconhecido (informação pertencente à instância). Considere que a ação composta por dar um passo e verificar a posição do muro correspondente custa O(1).

```
pos = 0;
while(encontrarPorta == False){
        pos += 1;
        ir para a posicao pos + 1.
        if(encontrarPorta != True){
                ir para a posicao -(pos + 1).
        }
}
```