Pesquisa e Classificação de Dados
Lista 1 (Análise de Algoritmos)

Prof. Ricardo Oliveira

Esta lista **não** vale nota e **não** deve ser entregue, mas apenas utilizada como material de apoio para estudo. Naturalmente, você pode tirar eventuais dúvidas com o professor.

Exercícios marcados com (B) são básicos e essenciais para a matéria. Exercícios marcados com (C) são complementares, mas também importantes. Recomenda-se **fortemente** resolver todos os exercícios.

1. (B) Mostre que:
   (a) $n = O(n)$
   (b) $2n + \frac{n}{5} + 1 = O(n)$
   (c) $2n + \sqrt{n} = O(n)$
   (d) $n = O(n^2)$
   (e) $324 = O(1)$
   (f) $\frac{1}{n} = O(1)$
   (g) $2n + 2 = O(n^{123456789})$
2. (C) Prove que $n^2 \neq O(n)$.
3. (B) Prove que a relação definida pela notação O é transitiva, isto é, se $f(n) = O(g(n))$ e $g(n) = O(p(n))$, então $f(n) = O(p(n))$.
4. (C) Prove que se $f(n) = O(log_2 n)$, então $f(n) = O(log_b n)$ para qualquer base $b > 2$.
5. (B) Determine a complexidade de tempo de pior caso para os seguintes algoritmos:
   (a)
   ```
   leia(a,b)
   imprima( (a+b)/2 )
   ```
   (b)
   ```
   para i = 0 a n-1
       leia(a[i])
       leia(b[i])
   prod=0
   para i = 0 a n-1
       prod = prod + a[i]*b[i]
   imprima( prod )
   ```
   (c)
   ```
   para i = 0 a n-1
       para j = 0 a n-1
           soma = 0
           para k = 0 a n-1
               soma = soma + A[i][k]*B[k][j]
           C[i][j] = soma
   ```
6. (B) Mostre que:
   (a) $n + 4 = \Omega(n)$

(b) $3n^2 - 40n = \Omega(n^2)$
(c) $n^2 = \Omega(n \log n)$

7. (C) É verdade que $2^{n+1} = \Omega(2^n)$? É verdade que $3^n = \Omega(2^n)$?
8. (B) Mostre que:
   (a) $3n + 4 = \Theta(n)$
   (b) $n \log n + n^2 + 3 = \Theta(n^2)$
   (c) $2^n + n^2 = \Theta(2^n)$
9. Analise e apresente a complexidade de tempo de *pior caso* e de *melhor caso* para os seguintes algoritmos:
   (a) (B)
   ```
   maior = 0
   Para i = 1 a n
       Se v[i] > maior
           maior = v[i]
   ```
   (b) (B)
   ```
   i=0, j = n-1
   enquanto i <= n/2 e j >= n/2
       troque(v[i],v[j])           // Considere troque() com custo O(1)
       i = i + 1
       j = j - 1
   ```
   (c) (B)
   ```
   i = 2
   enquanto i*i <= n
       Se n mod i == 0
           retorne "composto"
       i = i + 1
   retorne "primo"
   ```
   (d) (C)
   ```
   // Calcula a menor potencia de 2 que eh >= n
   pot=1
   enquanto nao(pot >= n)
       pot = pot * 2
   ```
   (e) (B)
   ```
   x=0
   Para i = 1 a n
       x = x + vetorA[i]
   Para j = 1 a m
       x = x - vetorB[i]
   ```

(f) (B)

```
int pow(int base, int exp) {
    if (exp==0)
        return 1;
    res = pow(base, exp/2);          // '/2' eh divisao inteira
    res = res*res;
    if (exp é impar)
        res = res*base;
    return res;
}
```

(g) (C)

```
int tt(int n) {
    if (n==0) return 0;
    return tt(n-1) + 2*tt(n-1) + 4*tt(n-1);
}
```

10. O seguinte algoritmo calcula o $n-$ésimo número da série de Fibonacci.

```
int fib(int n) {
    if (n==0 or n==1) return n;
    return fib(n-1) + fib(n-2);
}
```

(a) (B) Mostre que a complexidade de tempo do algoritmo é $O(2^n)$.

(b) (C) Mostre que a complexidade de tempo do algoritmo é, de fato, $\Theta(\phi^n)$, onde $\phi = 1.6180339...$ é a proporção áurea[1].

11. (B) Considerando que o elemento procurado x tem a mesma probabilidade de ocorrer em cada uma das N posições de um vetor v, determine a complexidade de tempo de pior, melhor e médio caso da busca linear, dada abaixo. Lembre que o comando `return` encerra a execução do algoritmo.

```
int busca(int v[], int N, int x) {
    int i;
    for (i=0;i<N;i++) {
        if (v[i] == x)
            return 1;
    }
    return 0;
}
```

[1] *https://pt.wikipedia.org/wiki/Proporção_áurea*

12. (B) Determine agora a complexidade de tempo de pior, melhor e médio caso da seguinte modificação do algoritmo dado no exercício anterior:

```
int busca(int v[], int N, int x) {
    int i, encontrei=0;
    for (i=0;i<N;i++) {
        if (v[i] == x)
            encontrei = 1;
    }
    return encontrei;
}
```

13. (C) O algoritmo abaixo recebe um vetor `v` de tamanho N e determina o primeiro elemento maior que 50 do vetor, ou -1 se nenhum for encontrado. Como exemplo, para o vetor $[42, 10, 62, 22, 87]$, a saída é 62. Determine a complexidade de tempo de caso médio deste algoritmo. Cada posição do vetor contém um inteiro entre 1 e 100 (inclusive) com a mesma probabilidade (isto é, para toda posição i do vetor, tem-se $P(v[i] = 1) = P(v[i] = 2) = \ldots = P(v[i] = 100) = 1/100$).

```
Para cada elemento v[i] do vetor
|    Se v[i] > 50
|    |    imprima v[i] e encerre o algoritmo
imprima -1 e encerre o algoritmo
```