SCC0601 – Introdução à Ciência da Computação II
Prof. Lucas Antiqueira

3ª lista de exercícios

1) Vamos supor que estamos comparando duas implementações diferentes que solucionam o mesmo problema. Para entradas de tamanho n, o primeiro algoritmo realiza $8n^2$ operações, enquanto que o segundo algoritmo realiza $64n\log n$ operações. Para quais valores de n o primeiro algoritmo é mais eficiente que o segundo?

2) Escreva as seguintes funções em notação assintótica *O* (Big-Oh):

a. $g(n) = n^3 - 1$
b. $g(n) = n^2 + 2\log n$
c. $g(n) = 3n^n + 5 \cdot 2^n$
d. $g(n) = (n-1)^n + n^{n-1}$
e. $g(n) = 302$

3) Mostre que $g(n) = n + \sqrt{n}$ é $O(n)$.

4) Mostre que $g(n) = n/1000$ não é $O(1)$.

5) Mostre que $g(n) = 0{,}5n^2 - 3n$ é $O(n^2)$.

6) Mostre que $g(n) = 0{,}5n^2$ não é $O(n)$.

7) É verdade que $2^{n+1} = O(2^n)$?

8) É verdade que $2^{2n} = O(2^n)$?

9) Ordene as seguintes funções por suas taxas de crescimento (comportamento assintótico):

a. $n\log n$
b. $n\log(\log n)$
c. $n(\log n)^2$
d. $n\log n^2$
e. $n^2 \log n$

10) Expresse a função $g(n) = n^3/1000 - 100n^2 - 100n + 3$ em termos da notação Θ (Big-Theta).

11) Para cada um dos seguintes trechos de código abaixo:

I. Calcule o número de vezes que a linha "soma = soma + 1" é executada como função da entrada n.
II. Implemente os códigos e execute-os para valores crescentes de n;
III. Compare a análise do item (I) com os tempos obtidos em (II).

(a)

```
soma = 0;
for (i=0; i<n; i++)
   soma = soma + 1;
```

(b)

```
soma = 0;
for (i=0; i<n; i++)
   for (j=0; j<n; j++)
      soma = soma + 1;
```

(c)

```
soma = 0;
for (i=0; i<n; i++)
   for (j=0; j<n*n; j++)
      soma = soma + 1;
```

(d)

```
soma = 0;
for (i=0; i<n; i++)
   for (j=0; j<i; j++)
      soma = soma + 1;
```

(e)

```
soma = 0;
for (i=0; i<n; i++)
   for (j=0; j<i*i; j++)
      for (k=0; k<j; k++)
         soma = soma + 1;
```

(f)

```
soma = 0;
for (i=0; i<n; i++)
   for (j=0; j<=i*i; j++)
      if (j%i == 0)
         for (k=0; k<j; k++)
            soma = soma + 1;
```

12) Por que, em geral, não há tanta preocupação com a complexidade de espaço?

13) Escreva um algoritmo para determinar se um inteiro positivo n é primo. Faça a análise de sua complexidade. Procure implementações alternativas e as compare com a sua, inclusive em termos de complexidade.

14) Discorra sobre a importância e utilidade do estudo de diferentes métodos de ordenação interna.

15) Explique porque é possível utilizar apenas as operações dominantes no cálculo da complexidade dos algoritmos de ordenação.

16) Quais as diferenças entre os algoritmos de ordenação por inserção e por seleção.

17) Por que o primeiro loop do algoritmo de ordenação por seleção (linha em destaque abaixo) só precisa ser executado n-1 vezes e não n vezes?

```
1.  void selection_sort(int v[], int n) {
2.        int i, j, aux;
3.        for (i = 0; i < n - 1; i++)
4.              for (j = i + 1; j < n; j++)
5.                    if (v[j] < v[i]) {
6.                          aux = v[j];
7.                          v[j] = v[i];
8.                          v[i] = aux;
9.                    }
10. }
```

18) Explique porque, no melhor caso, o algoritmo de ordenação por inserção tem complexidade de tempo $O(n)$ e não $O(n^2)$ como no pior caso. Faça menção ao código abaixo, se precisar.

```
1.  void insertion_sort(int v[], int n) {
2.        int i, j, x;
3.        for (i = 1; i < n; i++) {
4.              x = v[i];
5.              j = i - 1;
6.              while ((j >= 0) && (x < v[j])) {
7.                    v[j+1] = v[j];
8.                    j--;
9.              }
10.             v[j+1] = x;
11.       }
12. }
```

19) Escreva em C uma versão recursiva do Bubble Sort. A complexidade de tempo (número de comparações) do algoritmo muda? Na prática, a versão recursiva é mais rápida que a iterativa?

20) Mostre que a complexidade no melhor caso do Bubble Sort aprimorado é $O(n)$.

```
1.  void bubble_sort_improved(int v[], int n) {
2.        int i, j, aux, trocou = 1;
3.        for (i = n - 1; (i > 0 && trocou); i--) {
4.              trocou = 0;
5.              for (j = 0; j < i; j++)
6.                    if (v[j] > v[j+1]) {
7.                          aux = v[j];
8.                          v[j] = v[j+1];
9.                          v[j+1] = aux;
10.                         trocou = 1;
11.                   }
12.       }
13. }
```