Section 2

2.2-1

Express the function $n^3/1000 - 100n^2 - 100n + 3$ in terms of Θ-notation.

Expresse a função $n^3/1000 - 100n^2 - 100n + 3$ em termos da notação Θ.

Resposta:

Como o maior termo da equação é n^3, a resposta é: $\Theta(n^3)$.

2.2-2

Consider sorting n numbers stored in array A by first finding the smallest element of A and exchanging it with the element in A[1]. Then find the second smallest element of A, and exchange it with A[2]. Continue in this manner for the first n - 1 elements of A. Write pseudocode for this algorithm, which is known as selection sort. What loop invariant does this algorithm maintain? Why does it need to run for only the first n - 1 elements, rather than for all n elements? Give the best-case and worst-case running times of selection sort in Θ- notation.

Resposta:

```
for (i = 1 to length[A])
   do min = i
      for (j = i + i to length[A])
         do if A[j] < A[min]
            then min = j
      tmp = A[i]
      A[i] = A[min]
      A[min] = tmp
```

Loop invariant: at the start of each iteration of the outer for loop, the subarray A[1..j-1] consists of the j-1 smallest elements in the array A[1..n] and this subarray is in sorted order.

After the first n-1 elements, the subarray A[1..n-1] contains the smallest n-1 elements, sorted, and therefore element A[n] must be the largest element

O tempo de execução de cada linha, respectivamente:

i. $c1 * n$
ii. $c2 * (n-1)$
iii. $c3\sum_{i=1}^{n-1} (n-i+1) = c3\frac{n^2+n-2}{2}$
iv. $c4\sum_{i=1}^{n-1} (n-i) = c4\frac{n^2+n-4}{2}$
v. $c5\sum_{i=1}^{n-1} t_i = v$
vi. $c6 * (n-1)$
vii. $c7 * (n-1)$

A soma dos termos da:

(\frac{c3+c4}{2})n²+(c1+c2+\frac{c3+c4}{2}+c6+c7)n-(c2+c3+2*c4+c6+c7) + c5*v

onde *v* depende da ordem dos elementos da array. No melhor caso(array ordenada), a linha 5 não é executada nenhuma vez (*v* = 0), enquanto no pior caso(array ordenada do maior pro menor), a linha 5 é executada o mesmo números de vezes da linha 4 (v = c5\frac{n²+n-4}{2}), nos dois casos o tempo de execução é Θ(n²).

2.2-3

Consider linear search again (see Exercise 2.1-3). How many elements of the input sequence need to be checked on the average, assuming that the element being searched for is equally likely to be any element in the array? How about in the worst case? What are the average-case and worst-case running times of linear search in Θ-notation? Justify your answers.

Resposta:

Assuming equal probability of occurrence 1/n, average number of elements which need to be checked is 1/n * (1 + 2 + ... +n) = (n+1)/2. Running time is Θ(n)

O pseudocodigo é o seguinte:

```
for j = 1 to length[A]
  do if v == A[j]
    return j
return nil
```

i. c1 * i
ii. c2 * i_2
iii. c3 (retorno)

onde i é o menor valor tal que v = A[i], ou i = n+1, caso não exista nenhum A[i] = v; e i_2 = i, ou i_2 = n, caso i=n+1.

No caso médio, i=i_2=n/2, logo o tempo de execução é: (\frac{c1 + c2}{2})n + c3, que é Θ(n).

No pior caso (o elemento não está na lista) i=n+1, e i_2=n, logo o tempo de execução é: (c1+c2)n + (c1+c3), que também é Θ(n).

2.2-4

How can we modify almost any algorithm to have a good best-case running time?

Resposta:

Modify the algorithm so it checks whether the input satisfies some special case condition. If it does, output a pre-computed answer.