

Selection Sort

Ozéias Souza

UFRR

14 de junho de 2022

Algoritmo

```
void selection_sort(int num[], int tam) {
    int i, j, min, aux;
    for (i = 0; i < (tam-1); i++)
    {
        min = i;
        for (j = (i+1); j < tam; j++) {
            if(num[j] < num[min])
                min = j;
        }
        if (i != min) {
            aux = num[i];
            num[i] = num[min];
            num[min] = aux;
        }
    }
}
```

Breve descrição

O selection sort separa o vetor em uma parte que está ordenada e uma parte não-ordenada, fazendo um número menor de operações.

Função de Custo

```
void selection_sort(int num[], int tam) {  
    int i, j, min, aux;  
    for (i = 0; i < (tam-1); i++)  
    {  
        min = i;  
        for (j = (i+1); j < tam; j++) {  
            if(num[j] < num[min])  
                min = j;  
        }  
        if (i != min) {  
            aux = num[i];  
            num[i] = num[min];  
            num[min] = aux;  
        }  
    }  
}
```

Operações de comparação e atribuição possuem um número constante de instruções:
 $T(n) = 1$.

Função de Custo

```
void selection_sort(int num[], int tam) {  
    int i, j, min, aux;  
    for (i = 0; i < (tam-1); i++)  
    {  
        min = i;  
        for (j = (i+1); j < tam; j++) {  
            if(num[j] < num[min])  
                min = j;  
        }  
        if (i != min) {  
            aux = num[i];  
            num[i] = num[min];  
            num[min] = aux;  
        }  
    }  
}
```

Verificando internamente o **for** vemos que existe apenas uma comparação, ou seja, $T(n) = 1$. Logo,

Função de Custo

```
void selection_sort(int num[], int tam) {  
    int i, j, min, aux;  
    for (i = 0; i < (tam-1); i++)  
    {  
        min = i;  
        for (j = (i+1); j < tam; j++) {  
            if(num[j] < num[min])  
                min = j;  
        }  
        if (i != min) {  
            aux = num[i];  
            num[i] = num[min];  
            num[min] = aux;  
        }  
    }  
}
```

Verificando internamente o **for** vemos que existe apenas uma comparação, ou seja, complexidade $O(1)$. Logo,

$$T(n) = \sum_{j=i+1}^n 1 = n - (i + 1) + 1 = n - i.$$

Função de Custo

```
void selection_sort(int num[], int tam) {  
    int i, j, min, aux;  
    for (i = 0; i < (tam-1); i++)  
    {  
        min = i;  
        for (j = (i+1); j < tam; j++) {  
            if(num[j] < num[min])  
                min = j;  
        }  
        if (i != min) {  
            aux = num[i];  
            num[i] = num[min];  
            num[min] = aux;  
        }  
    }  
}
```

Observando o **for** mais externo percebemos que,

$$T(n) = \sum_{i=0}^{n-1} n - i.$$

Desenvolvendo o somatório obtemos:

Função de Custo

$$T(n) = \sum_{i=0}^{n-1} n - i = \sum_{i=0}^{n-1} n - \sum_{i=0}^{n-1} i =$$

$$((n-1) - 0 + 1)n - \frac{(n-1)((n-1) + 1)}{2} =$$

$$n^2 - \frac{(n-1)(n)}{2} = n^2 - \frac{n^2 - n}{2} = \frac{1}{2}(n^2 - n)$$











Complexidade

Obtemos a função de custo:







$$T(n) = \frac{1}{2}(n^2 - n)$$

Então podemos dizer que esse algoritmo tem uma complexidade $O(n^2)$, pois n^2 é o termo de maior grau da função.

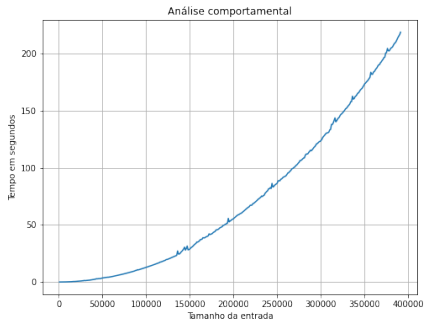
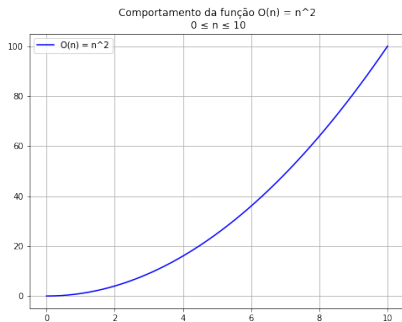
Experimentação

 n_1000.txt	4 KB
 n_2000.txt	9 KB
 n_3000.txt	14 KB
 n_4000.txt	19 KB
 n_5000.txt	24 KB
 n_6000.txt	29 KB
 n_7000.txt	34 KB
 n_8000.txt	38 KB
 n_9000.txt	43 KB
 n_10000.txt	48 KB

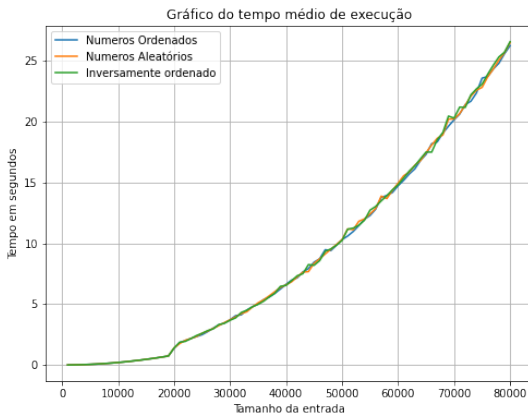
...

 n_994000.txt	6.687 KB
 n_995000.txt	6.694 KB
 n_996000.txt	6.701 KB
 n_997000.txt	6.708 KB
 n_998000.txt	6.714 KB
 n_999000.txt	6.720 KB

Foram gerados inúmeros casos de testes em arquivos de texto com números aleatórios. Foram executados 391 casos, ou seja, no máximo uma entrada com 391.000 elementos.



Foram executados 80 casos de testes(15x cada) utilizando listas com números ordenados, aleatórios e inversamente ordenados.



Selection Sort x Quick Sort

Tabela: Complexidades

Caso	Selection Sort	Quick
Melhor	$O(n^2)$	$O(n \log n)$
Médio	$O(n^2)$	$O(n \log n)$
Pior	$O(n^2)$	$O(n^2)$