Universidade de São Paulo - Instituto de Ciências Matemáticas e de Computação
SCC0220 - Laboratório de Introdução à Ciência da Computação II
Aluno: Gabriel Toschi de Oliveira (9763039)

Atividade 1

Contagem de operações e de tempo de execução

Introdução

Esta atividade consiste em exercitar os processos de contagem formal de operações e medição de tempo de execução de algoritmos. Dois algoritmos foram feitos como objeto de estudo: um algoritmo para inverter a ordem de um vetor e o algoritmo de ordenação bubble sort.

Para cada um deles, será apresentado a resolução formal da equação de contagem de operações e os dados médios de tempo de execução para vetores de 10, 100, 1.000, 5.000 e 10.000 posições. Como as ordens de grandeza dos tempos médios são bem diferentes, não foi possível construir um gráfico único com os dois dados.

Para as análises formais, serão usados padrões para representar certos tipos de instrução, de forma a aproximá-las: **A** para operações aritméticas e atribuições, **C** para comparações, **M** para acessos à memória e **F** para chamadas e retornos de funções. Durante as simplificações, o símbolo **X** será usado para uma instrução qualquer. O símbolo **n** será usado para representar o tamanho do vetor.

Função de swap (troca)

Ambos os algoritmos utilizam uma função genérica para a troca de valores entre duas posições de um vetor. Para evitar repetições neste relatório, segue a análise formal de contagem de operações para esta função.

```
void swap(int *a, int *b) {     // F
 int aux = *a;                  // M + A
 *a = *b;                       // 2M + A
 *b = aux;                      // M + A
}                               // F
```

Equação de contagem de operações

F + M + A + 2M + A + M + A + F

= **2F + 4M + 3A**

Simplificação da equação (F = M = A = X)

2F + 4M + 3A

= 2X + 4X + 3X

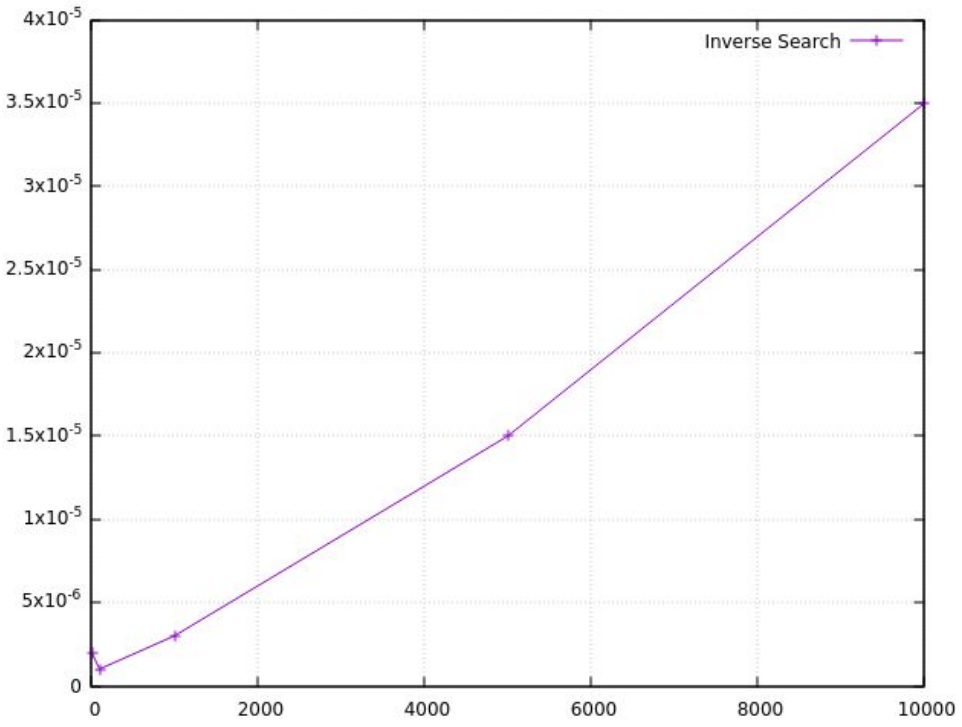
= **9X**

Complexidade de tempo

Constante, O(1). Por causa desta complexidade, será utilizado o símbolo **SWAP = 9X** nas próximas análises para simplificação do processo.

Algoritmo 1: inversão de vetor

Entrada	Tempo (s)
10	0.000002
100	0.000001
1.000	0.000003
5.000	0.000015
10.000	0.000035




```c
void reverseArray(int *array, int size) {       // F
 for (int i = 0; i < size / 2; i++) {           // 2A + C [1 it], n/2 * (3A + C) [2 it+]
   swap(&array[i], &array[size - 1 - i]);       // SWAP + M + 2A + M = 8X + 2A + 2M
 }
}                                               // F
```

Pior caso do algoritmo

Percorrer metade do vetor para invertê-lo. (também é melhor caso)

Equação de contagem de operações (SWAP = 9X)

F + 2A + C + n/2 * (3A + C) + n/2 * (SWAP + 2M + 2A) + F

= **(2)F + (2 + 5/2n)A + (1 + n/2)C + (n)M + (9/2n)X**

Simplificação da equação (F = M = A = C = X)

(2)F + (2 + 5/2n)A + (1 + n/2)C + (n)M + (9/2n)X

= (2)X + (2 + 5/2n)X + (1 + n/2)X + (n)X + (9/2n)X

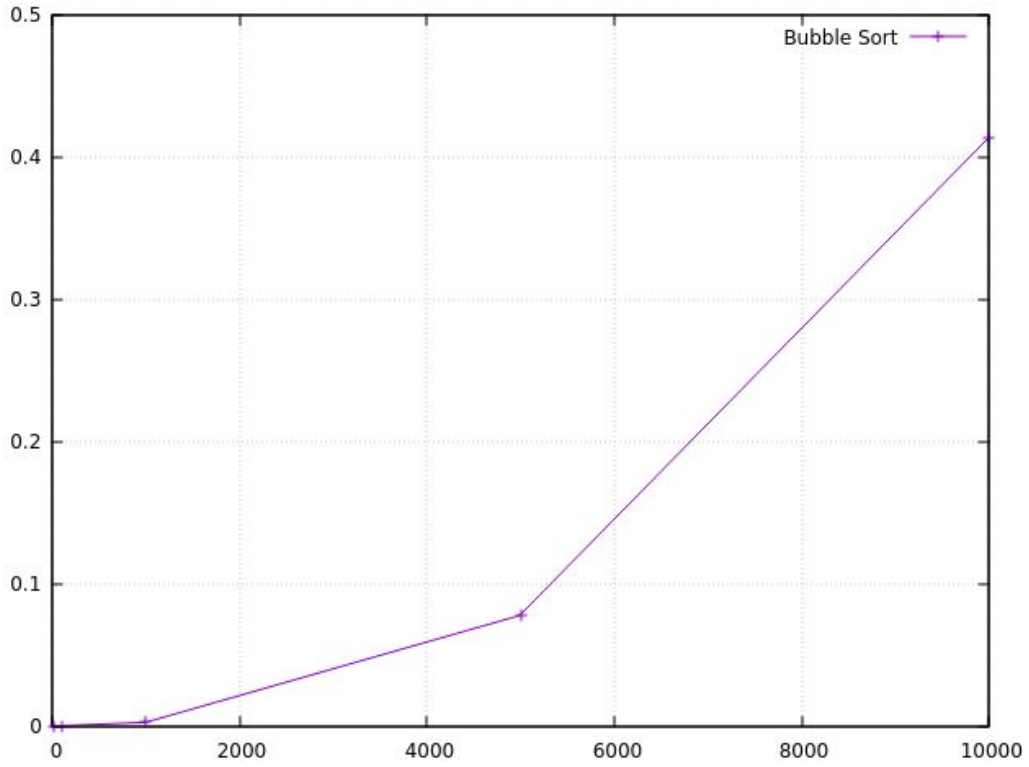
= **(5 + 15/2n)X**

Complexidade de tempo

Linear, O(n).

Algoritmo 2: bubble sort

Entrada	Tempo (s)
10	0.000003
100	0.000031
1.000	0.002426
5.000	0.077680
10.000	0.413284




```
void bubbleSort(int *array, int size) {          // F
 for (int i = 0; i < size - 1; i++) {            // 2A + C [1 it], n-1 * (2A + C) [2 it+]
   for (int j = 0; j < size - i - 1; j++) {      // 3A + C [1 it], (n²-n)/2 * (3A + C) [2 it+]
     if (array[j] > array[j+1]) {                // (n²-n)/2 * (2M + A + C)
       swap(&array[j], &array[j+1]);             // (n²-n)/2 * (SWAP + 2M + A)
     }
   }
 }
}                                                // F
```

Pior caso do algoritmo

Vetor é uma lista em ordem decrescente.

Equação de contagem de operações (SWAP = 9X)

F + 2A + C + n-1(2A + C) + 3A + C + (n²-n)/2 * (3A + C + 2M + A + C + SWAP + 2M + A) + F

= (2)F + (5/2n² - n/2 + 3)A + (n² + 1)C + (2n² - 2n)M + (9/2n² - 9/2n)X

Simplificação da equação (F = M = A = C = X)

(2)F + (5/2n² - n/2 + 3)A + (n² + 1)C + (2n² - 2n)M + (9/2n² - 9/2n)X

= (2)X + (5/2n² - n/2 + 3)X + (n² + 1)X + (2n² - 2n)X + (9/2n² - 9/2n)X

= (10n² - 7n + 6)X

Complexidade de tempo

Polinomial quadrático, O(n²).