Algoritmos de Ordenação

SCC0201 - Introdução à Ciência de Computação II

Clausius G. Reis Leandro A. Amaral Tiago S. Nazaré
Vanessa Q. Marinho

Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo

22 de junho de 2015

Agenda

Agenda

Introdução

É um dos algoritmos mais simples de ordenação.

Ideia Básica:

- Usa a estratégia de "comparação e troca"
- É constituído por várias fases, as quais contemplam inúmeras iterações
- A cada iteração, dois valores são comparados e se necessário, são trocados
- Vídeo: https://www.youtube.com/watch?v=P00xJgWzz2c

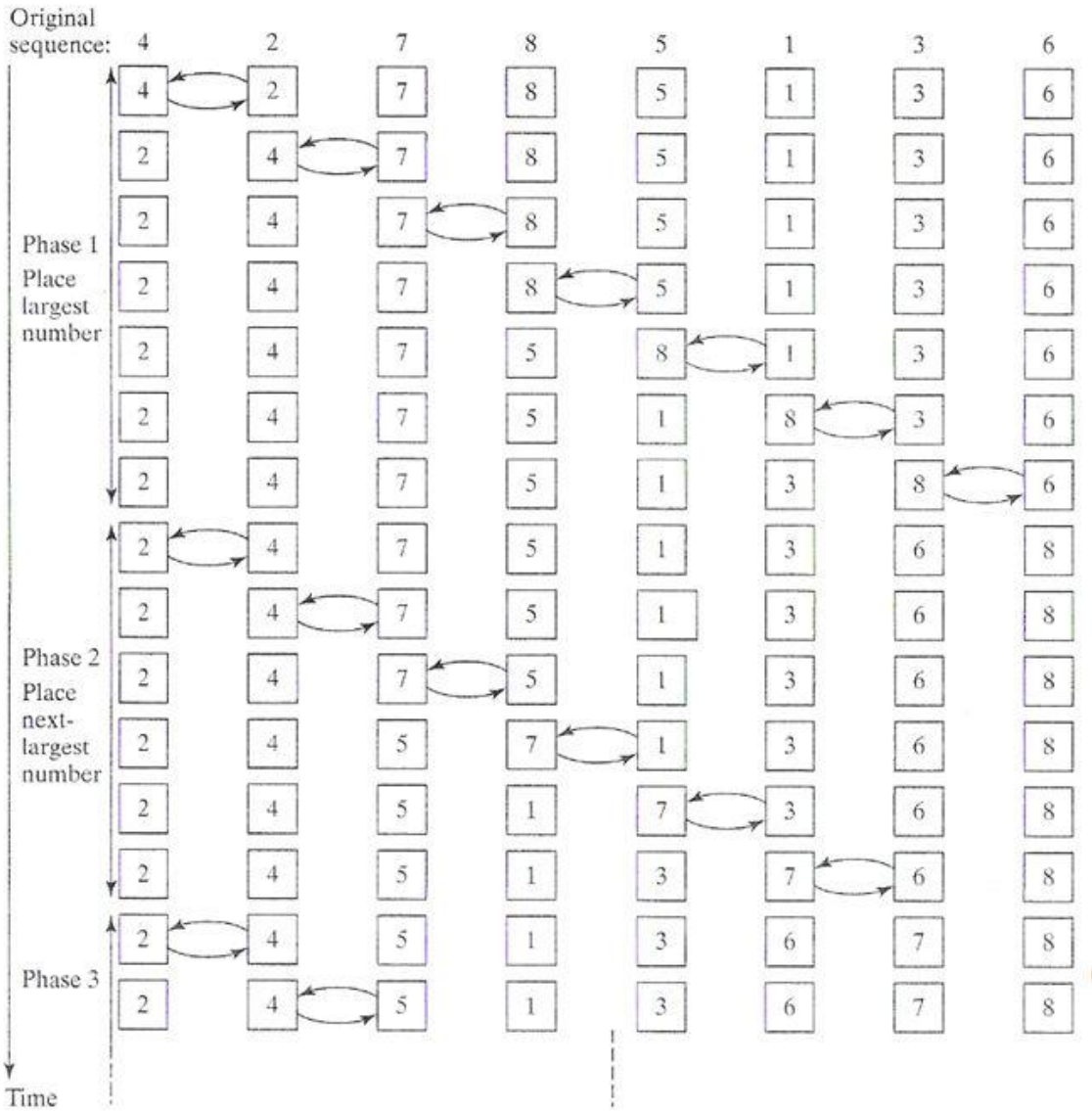
Agenda

Exemplo - Bubble Sort sequencial

A título de exemplo, considere um vetor, constituído por x_0, x_1, x_2, ..., x_n, não ordenado.

- Considere uma ordenação crescente
- O elemento de maior valor vai para o final do vetor
- A cada fase que é realizada, o número de iterações diminui em uma unidade
- Só existe troca na iteração caso o elemento da esquerda seja superior ao da direita
- Na pior das hipóteses existirão n-1 fases

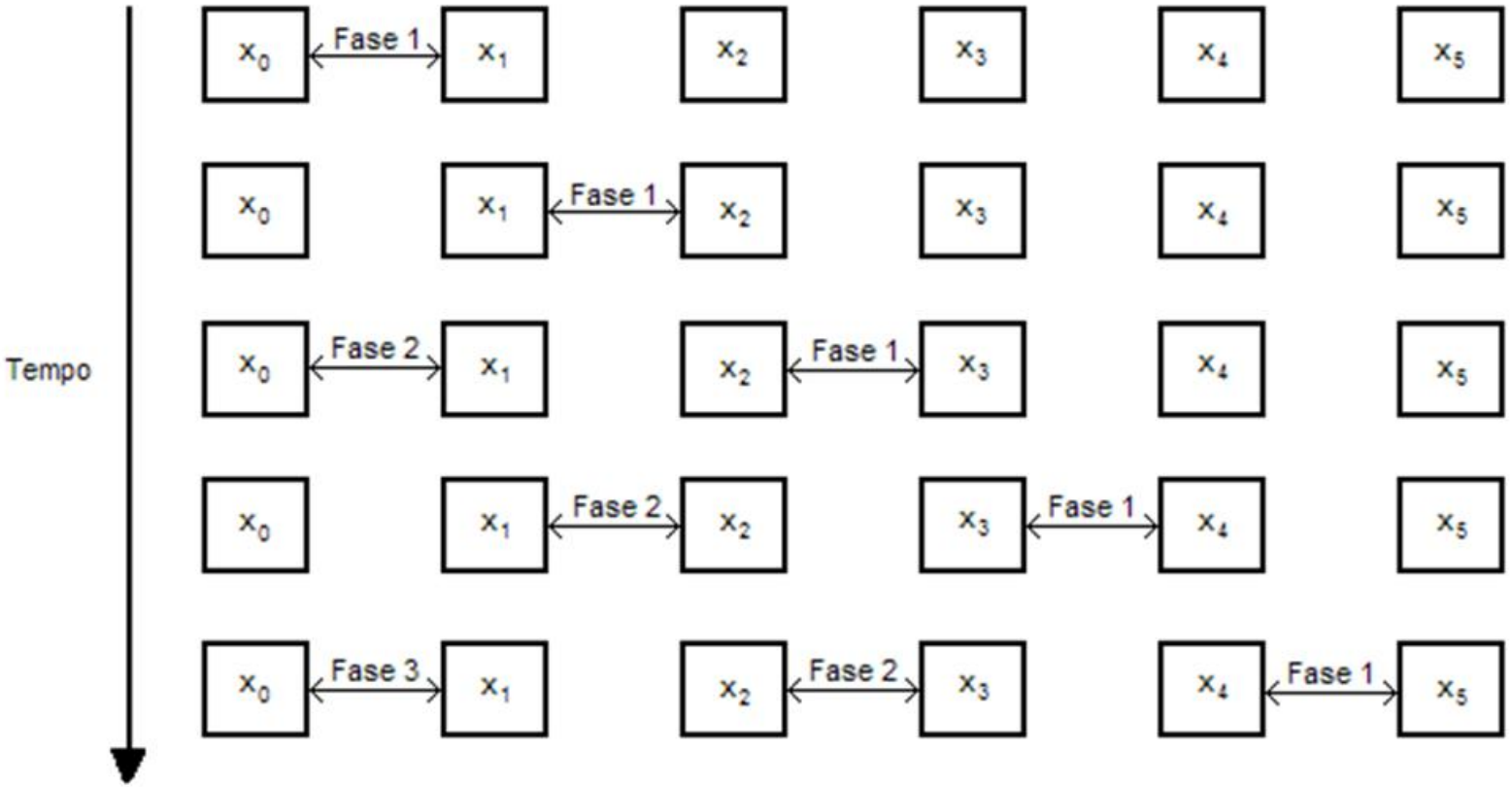
Exemplo - Bubble Sort sequencial



Exemplo - Bubble Sort com pipelining

- O pipelining é possivel pois as iterações de cada fase vão percorrendo o vetor, não necessitando posteriormente de valores anteriores
- Ou seja, quando a primeira fase já se encontra na comparação de x_2 e x_3, a segunda fase já pode usar os valores x_0 e x_1 para comparação

Exemplo - Bubble Sort com pipelining



Agenda

Bubble Sort - Código

```
void bubble_sort(int n, int *list) {
  long c, d, t;

  for (c = 0 ; c < ( n - 1 ); c++)
  {
    for (d = 0 ; d < n - c - 1; d++)
    {
      if (list[d] > list[d+1])
      {
        /* Swapping */

        t          = list[d];
        list[d]    = list[d+1];
        list[d+1] = t;
      }
    }
  }
}
```

Agenda

Análise de Complexidade

As operações de comparações e de troca de posição de elementos são executadas no pior caso, o algoritmo n-1 troca para o primeiro passo, depois n-2 trocas para o segundo elemento e assim sucessivamente. Trocas = n-1+n-2+n-3...+2+1 aproximadamente n^2 trocas. No melhor caso, nenhuma troca será realizada, pois em ambos os casos o algoritmo faz da ordem n comparações.

Complexidade no tempo: comportamento do algoritmo, em função do tamanho de entrada.
Complexidade no espaço: consumo de memória do algoritmo, em função do tamanho da entrada.

Análise de Complexidade

O tempo gasto na execução do algoritmo varia em ordem quadrática em relação ao número de elementos a serem ordenados.

T = $\mathcal{O}(n^2)$ - Notação "Big O"

Atividades mais custosas:

- Comparações
- Troca de posição (swap)

Melhor caso: vetor ordenado
Pior caso: vetor invertido

Exercício

Implementar (em C) o algoritmo de Bubble Sort de maneira recursiva.

Resposta

```
void bubble_sort_rec(int n, int *vet) {
        int i;
        int aux, flag = 0;

        for (i = 0; i < n-1; i++) {
                if (vet[i] > vet[i+1]) {
                        aux = vet[i];
                        vet[i] = vet[i + 1];
                        vet[i + 1] = aux;
                        flag = 1;
                }
        }

        if (flag != 0)
                bubble_sort_rec(n-1, vet);
}
void main() {
    bubble_sort_rec(4, vet);
}
```