Ordenação (Rápida)

Técnicas de Programação

Igor Montagner (igorsm1@insper.edu.br)

Ordenação

> Dado um array de números A, ordená-lo em ordem crescente

Igor Montagner (igorsm1@insper.edu.br)

Algoritmos de ordenação simples

Bubble Sort - trocar vizinhos até estar ordenado

Selection Sort - colocar o menor na posição `0`, o segundo menor na posição `1` e assim por diante

Igor Montagner (igorsm1@insper.edu.br)

Bubble Sort

```
BUBBLE_SORT(A)
    N ← TAMANHO(A)
    FEZ_TROCA ← VERDADEIRO

    ENQUANTO FEZ_TROCA FAÇA
        FEZ_TROCA ← FALSO
        PARA CADA I ← 0 ATÉ N-1 FAÇA
            SE A[I] > A[I+1] ENTÃO
                TROQUE OS VALORES DE A[I] E A[I+1]
                FEZ_TROCA ← VERDADEIRO
            FIM
        FIM
    FIM
```

Igor Montagner (igorsm1@insper.edu.br)

Bubble Sort

Características:

- melhor caso => loop de fora dá uma volta, de dentro faz `N` iterações
- pior caso => loop de fora dá `N` voltas, de dentro faz `N` iterações

Pior caso é N^2!

Igor Montagner (igorsm1@insper.edu.br)

Selection Sort

```
SELECTION_SORT(A)
    N ← TAMANHO(A)
    PARA I = 0 ATÉ N - 1 FAÇA
        MENOR_IDX = I
        PARA J = I+1 ATÉ N FAÇA
            SE A[J] < A[MENOR_IDX] FAÇA
                MENOR_IDX = J
            FIM
        FIM
        TROCA VALORES DE A[I] E A[MENOR_IDX]
    FIM
```

Igor Montagner (igorsm1@insper.edu.br)

Selection Sort

Características:

- sempre roda igual => loop externo roda `N-1` vezes, loop interno roda `N-I` vezes para o `i` atual

Tempo final também é N^2!

Igor Montagner (igorsm1@insper.edu.br)

Selection Sort vs Bubble Sort

- Se já estiver ordenado (ou quase isso), Bubble é melhor
- Selection faz menos escritas no vetor e isso é mais rápido em geral
- No pior caso ambos são iguais, mas o array que atinge o pior caso é diferente para ambos

Nova idéia: *Quick Sort*

1. escolha um elemento do vetor (pode ser o primeiro)
2. coloque ele no lugar certo
   - índice `i` tal que todo $A[j] \leq A[i], j < i$
   - índice `i` tal que todo $A[j] \geq A[i], j > i$
3. ordene a parte à esquerda de `i`
4. ordene a parte à direita de `i`

Igor Montagner (igorsm1@insper.edu.br)

Prática

Atividade Quick Sort (passos 1 e 2)

Igor Montagner (igorsm1@insper.edu.br)

Nova idéia: *Quick Sort*

1. escolha um elemento do vetor (pode ser o primeiro)
2. coloque ele no lugar certo
   - índice `i` tal que todo $A[j] \leq A[i], j < i$
   - índice `i` tal que todo $A[j] \geq A[i], j > i$
3. **ordene a parte à esquerda de `i`**
4. **ordene a parte à direita de `i`**

Ordenar o array inteiro depende de ordenar um sub-array

Igor Montagner (igorsm1@insper.edu.br)

Recursão

algoritmo que chama a si mesmo uma ou mais vezes

passando uma instância menor do problema

Igor Montagner (igorsm1@insper.edu.br)

Quicksort

```
QUICKSORT(A, INI, FIM)

IF INI <= FIM - 1 THEN
    RETURN
END

LP = PARTICIONA(A, INI, FIM)
QUICKSORT(A, INI, LP)
QUICKSORT(A, LP+1, FIM)
```