Estruturas de Dados 2

Prof. Silvia Brandão

2024.1

Ordenação

Conceitos básicos:

- Ordenação
  - Ato de colocar um conjunto de dados em uma determinada ordem predefinida
- Fora de ordem: 5, 2, 1, 3, 4
- Ordenado: 1, 2, 3, 4, 5 **OU** 5, 4, 3, 2, 1
- Algoritmo de ordenação
  - Coloca um conjunto de elementos em uma certa ordem

O problema da ordenação

- é um dos problemas com maior número de aplicações diretas ou indiretas (como parte da solução para um problema maior).

  - Exemplos de aplicações diretas:
    - Criação de rankings.
    - Definição de preferências em atendimentos por prioridade.

  - Exemplos de aplicações indiretas:
    - Otimização de sistemas de busca.
    - Manutenção de estruturas de bancos de dados.

Conceitos básicos

- A ordenação é baseada em uma chave qualquer.
  - Essa chave de ordenação é o campo do item utilizado para comparação. É por meio dela que sabemos se um determinado elemento está a frente ou não de outros no conjunto.

- Deve existir uma regra de ordenação bem-definida.
  - Alguns tipos de ordenação:
    - numérica: 1, 2, 3, 4, 5
    - lexicográfica (ordem alfabética): Ana, André, Bianca, Ricardo
  - Independentemente do tipo, a ordenação pode ser:
    - crescente: 1, 2, 3, 4, 5; Ana, André, Bianca, Ricardo
    - decrescente: 5, 4, 3, 2, 1; Ricardo, Bianca, André, Ana

Conceitos básicos

- Os algoritmos de ordenação podem ser classificados como de:
  - Ordenação interna:
    - O conjunto de dados a ser ordenado cabe todo na memória principal (RAM)
    - Qualquer elemento pode ser imediatamente acessado
  - Ordenação externa:
    - O conjunto de dados a ser ordenado não cabe na memória principal
    - Os dados estão armazenados em memória secundário (por exemplo, um arquivo)
    - Os elementos são acessados sequencialmente ou em grandes blocos

Conceitos básicos

- **Ordenação Interna**
  - Ordenação por Seleção
  - Ordenação por Inserção
  - Shellsort
  - Quicksort
  - Heapsort
  - Ordenação Parcial
    - Seleção Parcial
    - Inserção Parcial
    - Heapsort Parcial
    - Quicksort Parcial

- **Ordenação Externa**
  - Intercalação Balanceada de Vários Caminhos
  - Implementação por meio de Seleção por Substituição
  - Considerações Práticas
  - Intercalação Polifásica
  - Quicksort Externo

Ordenação interna

- Classificação dos métodos de ordenação interna:
  - Métodos simples:
    - Adequados para pequenos arquivos.
    - Requerem $O(n^2)$ comparações.
    - Produzem programas pequenos.
  - Métodos eficientes:
    - Adequados para arquivos maiores.
    - Requerem $O(n \log n)$ comparações.
    - Usam menos comparações.
    - As comparações são mais complexas nos detalhes.
    - Métodos simples são mais eficientes para pequenos arquivos.

Conceitos básicos

- A ordenação pode ser estável ou não.
  - Um algoritmo de ordenação é considerado estável se a ordem dos elementos com chaves iguais não muda durante a ordenação.
  - O algoritmo preserva a ordem relativa original dos valores.
    - Exemplo de dados não ordenados: 5a, 2, 5b, 3, 4, 1
    - Exemplo de dados ordenados:
      - 1, 2, 3, 4, 5a, 5b: ordenação estável
      - 1, 2, 3, 4, 5b, 5a: ordenação não-estável

- A maioria dos métodos de ordenação é baseada em **comparações** das chaves.
- Existem métodos de ordenação que utilizam o princípio da **distribuição.**

Exemplo de ordenação por distribuição

- Considere o problema de ordenar um baralho com 52 cartas na ordem:

$$A < 2 < 3 < \cdot\ \cdot\ \cdot < 10 < J < Q < K$$

e

$$\clubsuit < \diamondsuit < \heartsuit < \spadesuit$$

- Algoritmo:
  1. Distribuir as cartas abertas em treze montes: ases, dois, três, . . ., reis.
  2. Colete os montes na ordem especificada.
  3. Distribua novamente as cartas abertas em quatro montes: paus, ouros, copas e espadas.
  4. Colete os montes na ordem especificada.

Métodos de ordenação

- Bolha, Inserção e Seleção
  - métodos, considerados básicos, de fácil implementação, mas auxiliam o entendimento de algoritmos complexos;
  - utilizam como uma de suas operações básicas a comparação de elementos da lista.
- Shellsort, Mergesort, Quicksort e Heapsort
  - métodos, considerados sofisticados, que apresentam melhor desempenho;
  - utilizam estratégias mais sofisticadas como, por exemplo, divisão e conquista e o uso de estruturas de dados conhecidas como heaps binários;
  - também são baseados na operação de comparação.
- Countingsort, Bucketsort e Radixsort
  - esses métodos têm em comum o fato de não utilizarem a operação de comparação;
  - sob certas hipóteses, eles ordenam em tempo linearmente proporcional ao tamanho da lista.

Método Bubble Sort (bolha)

- É um dos algoritmos de ordenação mais conhecido que existe.
- Remete a ideia de bolhas flutuando em um tanque de água em direção ao topo até encontrarem o seu próprio nível (ordenação crescente).

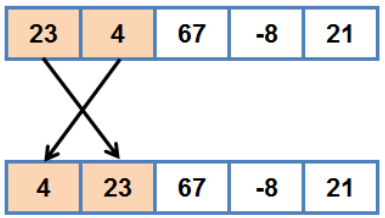
- Funcionamento
  - Compara pares de valores adjacentes e os troca de lugar se estiverem na ordem errada.
    - Trabalha de forma a movimentar, uma posição por vez, o maior valor existente na porção não ordenada de um array para a sua respectiva posição no array ordenado.
  - Esse processo se repete até que mais nenhuma troca seja necessária; isto é, elementos já ordenados.

Bubble Sort – algoritmo

```
//ENTRADA: um vetor numero com n posições
//SAÍDA: o vetor numero em ordem crescente

PARA i = 1 até n - 1 {
  PARA j = 0 até n - 1 - i {
    SE numero[j] > numero[j+1] {
      aux = numero[j] //swap (troca)
      numero[j] = numero[j+1]
      numero[j+1] = aux
    }
  }
}
```

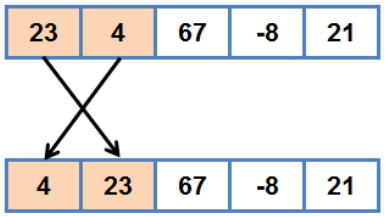
Troca dois valores consecutivos no vetor



Bubble Sort – algoritmo com flag

```
//ENTRADA: um vetor numero com n posições
//SAÍDA: o vetor numero em ordem crescente
troca = 1; //flag
ENQUANTO (troca == 1) {
    troca = 0;
    PARA (i = 0 até n – 1 - i) {
        SE (numero[i] > numero[i + 1]) {
            troca = 1;
            aux = numero[i];
            numero[i] = numero[i + 1];
            numero[i + 1] = aux;
        }
    }
}
```

Troca dois valores consecutivos no vetor



O comportamento do Bolha com Flag é sensível ao tipo do vetor fornecido como entrada.

Método Bubble Sort (bolha)

- Vantagens
  - Simples e de fácil entendimento e implementação
  - Está entre os métodos de ordenação mais difundidos existentes
- Desvantagens: Não é um algoritmo eficiente
  - Sua eficiência diminui drasticamente a medida que o número de elementos no array aumenta
  - É estudado apenas para fins de desenvolvimento de raciocínio

- **Complexidade:** Considerando um array com N elementos, o tempo de execução é:
  - $O(n)$, melhor caso: os elementos já estão ordenados.
  - $O(n^2)$, pior caso: os elementos estão ordenados na ordem inversa.
  - $O(n^2)$, caso médio.

Recursos

- Sorting - https://www.cs.usfca.edu/~galles/visualization/Algorithms.html

- Prof. Leonardo Cabral -Implementações http://www.larback.com.br/bolha.php

Algoritmo	Tempo		
	Melhor	Médio	Pior
Bubble sort	$O(n)$	$O(n^2)$	$O(n^2)$
Insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Merge sort	$O(n \log_2 n)$	$O(n \log_2 n)$	$O(n \log_2 n)$
Quick sort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$

Prática

- Implemente o método Bubble Sort e determine o tempo (custo) de ordenação considerando o melhor, o pior e o caso médio. Para isso, analise:
  - um array com 10,100,1000 e10000 números randômicos. Trace a curva de desempenho do método pra n x tempo de execução;
  - uma lista dinâmica com 1000 números randômicos;
  - um arquivo em disco com 1000 linhas de inteiros.

Próxima aula

- Métodos Simples – Selection Sort (com análise de custo)