Estrutura de Dados II

Métodos de Ordenação
Parte I

Profª Márcio Bueno
ed2tarde@marciobueno.com / ed2noite@marciobueno.com

Material baseado nos materiais da
Profª Ana Eliza e Prof. Robson Lins

Objetivo

- Rearranjar um conjunto de itens em uma ordem ascendente ou descendente.
  - ✓ Facilita a recuperação posterior de itens do conjunto ordenado.
- Considerar métodos de ordenação de arquivos de itens contendo chaves.
- As chaves, que são apenas parte dos itens, são usadas para controlar a ordenação.

```
Tipo item = registro
     chave : tipochave
     {outros componentes}
fim
```

- A escolha do tipo de chave é arbitrária
  - ✓ Qualquer tipo sobre o qual exista uma regra de ordenação bem definida (alfabética ou numérica).

Aplicações de Ordenação

- Muitas aplicações podem se beneficiar da ordenação dos itens de dados:
  - ✓ **Busca** – Busca binária permite que você teste se um item está em um arquivo em um tempo *O (log n)* se as chaves do arquivo estiverem ordenadas. Busca é uma das mais importantes aplicações de ordenação.
  - ✓ **Par mais Próximo** – Dado um conjunto de ***n*** números, como encontrar o par de números que tem a menor diferença entre eles? Depois dos números estarem ordenados, o par mais próximo estará disposto próximo um do outro de forma ordenada. Assim, uma busca completará o trabalho.

Aplicações de Ordenação

- ✓ **Elementos Duplicados** – Existem elementos duplicados em um conjunto de ***n*** itens? O algoritmo mais eficiente consiste em ordená-los e então fazer uma busca para verificar os pares adjacentes. Este é um caso especial do Par mais Próximo, onde se procura por um par separado por um espaço de zero.
- ✓ **Freqüência de Distribuição** – Dado um conjunto de ***n*** itens, qual o elemento que ocorre mais vezes? Se os itens estiverem ordenados, pode-se percorrer da esquerda para a direita e contá-los, uma vez que todos os elementos idênticos ficarão juntos durante a ordenação.
- ✓ **Seleção** – Qual o maior elemento no conjunto? Se as chaves estão ordenadas em um vetor, então a maior chave pode ser encontrada simplesmente por buscar a k-ésima posição no vetor.

Classificação

- São classificados em 2 grupos:
  - ✓ Ordenação Interna
    - Quando o arquivo a ser ordenado cabe totalmente na memória principal
    - A quantidade de itens a ser ordenada cabe em um vetor
    - Qualquer item pode ser acessado imediatamente
  - ✓ Ordenação Externa
    - Quando o arquivo a ser ordenado está armazenado em memória secundária
    - A quantidade de itens a ser ordenada não cabe em um vetor
    - Itens são acessados seqüencialmente ou em blocos.

Escolha

- Aspecto predominante: tempo gasto para ordenar um arquivo.
- As medidas utilizadas para avaliar o desempenho de um algoritmo de ordenação são:
  - ✓ Número de comparações entre chaves
  - ✓ Número de movimentações (ou trocas) dos objetos
- Quantidade extra de memória auxiliar utilizada pelo algoritmo
  - ✓ Métodos que utilizam vetores e que executam a permutação dos itens no próprio vetor são os preferidos
  - ✓ Métodos que utilizam listas encadeadas necessitam n palavras extras de memória para os apontadores e são utilizados apenas em casos especiais.
  - ✓ Métodos que utilizam uma quantidade extra de memória para armazenar uma outra cópia dos itens são menos importantes

Ordenação Interna

- Métodos de Ordenação Interna:
  - ✓ Elementares
    - Arquivos pequenos – requer $O(n^2)$ comparações
  - ✓ Sofisticados
    - Arquivos grandes – requer $O(n \log n)$ comparações

- Métodos Elementares
  - ✓ Produzem programas pequenos, fáceis de entender
  - ✓ Terminologia e mecanismos básicos para estudar e desenvolver métodos mais sofisticados
  - ✓ Mais eficientes que alguns.

Estabilidade

- Um método de ordenação é estável se ele preserva a ordem relativa dos itens com chaves duplicadas.
- Um método de ordenação não é estável se ele não preserva a ordem relativa dos itens com chaves duplicadas.
  - ✓ A ordenação dos registros a seguir pode ser apropriada em qualquer uma das chaves.
    - Suponha que as chaves estejam inicialmente ordenadas pela primeira chave (esquerda).
    - Um algoritmo de ordenação não estável não preserva a ordem dos registros com chaves duplicadas (centro).
    - Um algoritmo estável preserva a ordem (direita).

Estabilidade

Adão	1
Beto	2
Bruno	4
João	2
José	4
Sonia	1
Tomaz	4
Vanda	2

Arquivo Original

Adão	1
Sonia	1
Vanda	2
João	2
Beto	2
Tomaz	4
Bruno	4
José	4

Não Estável

Adão	1
Sonia	1
Beto	2
João	2
Vanda	2
Bruno	4
José	4
Tomaz	4

Estável

Estabilidade

- Muitos (não todos) algoritmos simples preservam a estabilidade, enquanto muitos sofisticados (não todos) não preservam.
- Se estabilidade é vital, podemos forçar adicionando um índice a chave ou simplesmente aumentando o tamanho da chave.

Classificação em Memória Primária

- **Métodos Elementares**
  - ✓Classificação por Inserção
    - Método da Inserção Direta
    - Método dos Incrementos Decrescentes - Shellsort
  - ✓Classificação por Trocas
    - Método da Bolha – Bubblesort
  - ✓Classificação por Seleção
    - Método da Seleção Direta
  - ✓Classificação por Intercalação
    - Método da Intercalação Simples - MergeSort

Classificação em Memória Primária

- **Métodos Eficientes (Sofisticados)**
  - ✓Classificação por Troca
    - Método de Partição e Troca - Quicksort
  - ✓Classificação por Seleção
    - Método de Seleção em Árvore - Heapsort

Classificação por Inserção

- **Definição**
  - ✓Este método consiste em realizar a ordenação pela inserção de cada um dos elementos em sua posição correta, levando em consideração os elementos já ordenados.

Classificação por Inserção

- **Método da Inserção Direta**
  - ✓Funcionamento - Ordenação crescente:
    - O algoritmo consiste em n – 1 passos. Onde n é o tamanho do vetor de dados a ser ordenado.
    - Para os passos p = 1 até n – 1, inserção assegura que os elementos da posição 0 até p já estão ordenados.
    - No passo p, nós movemos o elemento na posição p para a esquerda, até que seu lugar correto seja encontrado entre os p primeiros elementos.
    - O elemento na posição p é salvo em tmp, e todos os elementos maiores são movidos uma casa à direita e, então, tmp é colocado na posição correta.

Classificação por Inserção

- **Método da Inserção Direta**
  - ✓Exemplo: Ordenação após cada passo

Original	34 8 64 51 32 21	Posições Movidas
Depois p = 1	8 34 64 51 32 21	1
Depois p = 2	8 34 64 51 32 21	0
Depois p = 3	8 34 51 64 32 21	1
Depois p = 4	8 32 34 51 64 21	3
Depois p = 5	8 21 32 34 51 64	4

Classificação por Inserção

- **Método da Inserção Direta**
  - ✓ Algoritmo:

```
void insertionSort( int vet[ ], int n )
{
      int j, p, tmp;
      for( p = 1; p < n; p++ )
      {
          tmp = vet[ p ];
          for( j = p; j > 0 && tmp < vet[ j - 1 ]; j-- )
              vet[ j ] = vet[ j - 1 ];
          vet[ j ] = tmp;
      }
 }
```

Classificação por Inserção

- **Análise do Método da Inserção Direta**
  - ✓ O número mínimo de comparações ocorre quando os itens estão originalmente em ordem
  - ✓ O número máximo de comparações ocorre quando os itens estão originalmente na ordem reversa
  - ✓ Para arquivos já ordenados o algoritmo descobre a um custo *O(n)* que cada item já está no seu lugar
    - Deve ser utilizado quando o arquivo está "quase" ordenado
    - Quando se deseja adicionar uns poucos itens a arquivos já ordenados

Classificação por Inserção

- **Análise do Método da Inserção Direta**
  - ✓Melhor caso (o vetor já está ordenado):
    - Nenhum movimento substancial é realizado, somente a variável tmp é inicializada e o valor armazenado nela é movido de volta para a mesma posição;
    - É necessário pelo menos uma comparação para cada posição p num total de n - 1 comparações que são O(n)
    - 2(n -1) movimentos desnecessários (tmp) são realizados, também é O(n)

Classificação por Inserção

- **Análise do Método da Inserção Direta**
  - ✓Pior caso (o vetor está em ordem inversa):
    - Cada elemento a ser inserido será menor que todos os elementos já ordenados
      - vet[p] é menor que vet[0], ..., vet[p-1]
    - Portanto, todos os elementos deverão ser deslocados uma posição à direita;
    - Para cada interação p do *for* mais externo existem p comparações e o número total de comparações para todos os passos é:

$$\sum_{i=1}^{n-1} i = 1 + 2 + 3 + \ldots + (n-1)$$
$$= n(n-1)/2 = O(n^2)$$

Classificação por Inserção

- **Análise do Método da Inserção Direta**
  - ✓Pior caso (o vetor está em ordem inversa):
    - Número de vezes em que a atribuição no *for* mais interno é executada pode ser calculado usando a mesma fórmula:

      $\Sigma_{i=1}^{n-1} i = 1 + 2 + 3 + \ldots + (n-1)$
      $= n(n-1)/2 = O(n^2)$

    - O número de vezes em que tmp é carregada e descarregada é somado àquele dando um total de movimentos de:
    - $n(n-1)/2 + 2(n-1) = (n^2 + 3n - 4)/2 = O(n^2)$

Classificação por Inserção

- **Análise do Método da Inserção Direta**
  - ✓Desempenho Médio do Método:
    - Dados em ordem aleatória
    - O desempenho médio corresponderá à média aritmética do desempenho nos casos extremos:

      $((n - 1) + ((n^2 - n) / 2)) / 2 = (n^2 + n - 2) / 4 = \mathbf{O(n^2)}$
    - A mesma idéia para o cálculo médio de movimentos: $2(n - 1) + (n^2 + 3n - 4)/2) / 2 = (n^2 + 5n - 6) / 4 = \mathbf{O(n^2)}$
    - O desempenho médio do método é da ordem de n^2, ou seja, é proporcional ao quadrado do número de elementos do vetor.
    - Este método não é indicado para vetores com muitos elementos.

Classificação por Seleção

- **Definição**
  - ✓ Este processo de classificação consiste em uma seleção sucessiva do menor ou do maior valor contido no vetor, dependendo se a classificação dos elementos será em ordem crescente ou decrescente.
  - ✓ A cada passo, o elemento de menor (ou maior) valor é selecionado e colocado em sua posição correta dentro do vetor classificado. Esse processo é repetido para o segmento do vetor que contém os elementos ainda não selecionados.

Classificação por Seleção

- **Método da Seleção Direta**
  - ✓ O vetor é dividido em dois segmentos: o primeiro contendo os valores já classificados e o segundo contendo os elementos ainda não selecionados.
  - ✓ Inicialmente, o primeiro segmento está vazio e o segundo contém todos os elementos do vetor.

Classificação por Seleção

- **Método da Seleção Direta - Algoritmo**

  1°) É feita uma varredura no segmento que contém os elementos ainda não selecionados, identificando o elemento de menor (ou maior) valor;

  2°) É realizada a troca do elemento identificado na etapa anterior com o primeiro elemento do segmento;

  3°) O tamanho do segmento que contém os elementos ainda não selecionados é atualizado, ou seja, subtrai-se um de seu tamanho;

  4°) O processo é repetido até que o segmento fique com apenas um elemento, que é o maior (ou menor) valor do vetor.

Classificação por Seleção

- **Método da Seleção Direta – Exemplo 1**

Original	34	8	64	51	32	21
Depois i = 0	8	34	64	51	32	21
Depois i = 1	8	21	64	51	32	34
Depois i = 2	8	21	32	51	64	34
Depois i = 3	8	21	32	34	64	51
Depois i = 4	8	21	32	34	51	64

Classificação por Seleção

- **Método da Seleção Direta – Exemplo 2**

Chaves Iniciais

i = 0
i = 1
i = 2
i = 3
i = 4

0	1	2	3	4	5
O	R	D	E	N	A
A	R	D	E	N	O
A	D	R	E	N	O
A	D	E	R	N	O
A	D	E	N	R	O
A	D	E	N	O	R

Classificação por Seleção

- **Método da Seleção Direta – Exemplo 3**

Vetor Inicial	(21 27 **12** 20 37 19 17 15)	TAM = 8
i = 0	(12 27 21 20 37 19 17 **15**)	TAM = 7
i = 1	(12 15 21 20 37 19 **17** 27)	TAM = 6
i =2	(12 15 17 20 37 **19** 21 27)	TAM = 5
i = 3	(12 15 17 19 37 **20** 21 27)	TAM = 4
i = 4	(12 15 17 19 20 37 **21** 27)	TAM = 3
i = 5	(12 15 17 19 20 21 37 **27**)	TAM = 2
i = 6	(12 15 17 19 20 21 27 37)	TAM = 1

TAM = tamanho do vetor desordenado

Classificação por Seleção

- **Método da Seleção Direta - Algoritmo**

```
void selectionSort ( int vet[ ], int n )
{
   int i, j, min, tmp;
   for ( i = 0;  i < n - 1;  i++ )
   {
      for (j = i + 1,min = i;  j < n;  j++ ){
          if (vet[ j ] < vet[ min ])
              min = j;
      }
      tmp  =  vet[ i ];
      vet[ i ]  =  vet[ min ];
      vet[ min ]  =  tmp;
   }
}
```

Classificação por Seleção

- **Análise do Método da Seleção Direta**
  - ✓ A análise é simplificada pela presença de dois laços *for* com os limites inferior e superior.
  - ✓ O laço mais externo executa n - 1 vezes, e, para cada i entre 0 e (n - 2) o laço mais interno interage j = (n – 1) – i vezes.
  - ✓ Como as comparações de chaves são feitas no laço mais interno, o número total de comparações é dado por:

$$\sum_{i=0}^{n-2} (n - 1 - i) = (n - 1) + (n - 2) + (n - 3) + \ldots + 2 + 1$$

$$= n(n - 1)/2 = O(n^2)$$

Classificação por Seleção

- **Análise do Método da Seleção Direta**
  - ✓ Adequado para ser utilizado em arquivos com registros muito grandes
  - ✓ Número de movimentos dos registros é pequeno
  - ✓ Para tais aplicações, o custo de movimentar supera o custo de comparações, e nenhum outro algoritmo ordena um arquivo com tão poucos movimentos quanto este.
  - ✓ O fato do arquivo está ordenado não diminui o número de movimentos.

Classificação por Trocas

- **Definição**
  - ✓Este processo de classificação consiste na comparação de pares de chaves de ordenação, trocando os elementos correspondentes caso estejam fora de ordem.

Classificação por Trocas

- **Método da Bolha - Bubblesort**
  - ✓Neste método, o princípio geral da classificação por trocas é aplicado a todos os ***pares consecutivos*** de chaves não ordenados. Quando não restarem mais pares não ordenados, o vetor estará classificado.
  - ✓Pode ser melhor entendido se o vetor for visto como uma coluna vertical cujos menores elementos estão no topo e os maiores, na base.

Classificação por Trocas

- **Método da Bolha - Algoritmo**
  1º) Em cada passo, cada elemento é comparado com seu próximo;
  2º) Se o elemento estiver fora de ordem a troca é realizada;
  3º) Realizam-se tantos passos quanto forem necessários até que não ocorram mais trocas.

Classificação por Trocas

- **Método da Bolha – Ordenação Crescente**
  – Exemplo:

Vetor inicial (28 26 30 24 25)

Primeira Varredura (i = 0):

(28 26 30 24 25) compara (25,24): não troca.

(28 26 30 24 25) compara (24,30): troca.

(28 26 24 30 25) compara (24,26): troca.

(28 24 26 30 25) compara (24,28): troca.

(**24** 28 26 30 25) fim da primeira varredura.

Classificação por Trocas

- **Método da Bolha - Comentários**
  - ✓ O processo de comparação dos **n - 1** pares de chaves é denominado ***varredura***.
  - ✓ Cada varredura sempre irá posicionar a chave de **menor** valor em sua posição correta definitiva (no início do vetor).
  - ✓ Isso significa que a cada nova varredura podemos desconsiderar a primeira posição do vetor, que fica reduzido de um elemento.

Classificação por Trocas

- **Método da Bolha – Ordenação Crescente - Exemplo (cont.)**

Vetor inicial (**24** | 28 26 30 25)

- Segunda Varredura (i = 1):

(**24** | 28 26 30 25) compara (25,30): troca.
(**24** | 28 26 25 30) compara (25,26): troca.
(**24** | 28 25 26 30) compara (25,28): troca.
(**24** | **25** 28 26 30) fim da segunda varredura.

Classificação por Trocas

- **Método da Bolha – Ordenação Crescente - Exemplo (cont.)**

Vetor inicial (**24 25** | 28 26 30)

- Terceira Varredura (i = 2):

(**24 25** | 28 26 30) compara (30,26): não troca.
(**24 25** | 28 26 30) compara (26,28): troca.
(24 25 **26** | 28 30) fim da terceira varredura.

Classificação por Trocas

- **Método da Bolha – Ordenação Crescente - Exemplo (cont.)**

Vetor inicial (**24 25 26** | 28 30)

- Quarta Varredura (i = 3):

(**24 25 26** | 28 30) compara (30,28): não troca.

(**24 25 26 28** | 30) fim da quarta varredura.

Classificação por Trocas

- **Método da Bolha - Algoritmo**

```
void bubblesort ( int vet[], int n)
{
  int i, j, tmp;
  for (i = 0; i < n - 1; i++)
     for (j = n - 1; j > i; j--)
          if( vet[j] < vet[j - 1] ){
              tmp = vet[j - 1];
              vet[j - 1] = vet[j];
              vet[j] = tmp;
          }
}
```

Classificação por Trocas

- **Análise do Método da Bolha**
  - ✓ O número de comparações é o mesmo em cada caso (melhor, pior e médio) e igual ao número total de comparações do laço for mais interno:

$$\Sigma_{i=0}^{n-2} (n - 1 - i) = n(n-1)/2 = O(n^2)$$

Classificação por Trocas

- **Análise do Método da Bolha**
  - ✓Desvantagens:
    - Borbulha itens etapa por etapa para cima, em direção ao topo da matriz. Procura dois elementos adjacentes e os troca, caso estejam fora de lugar
    - Se um elemento deve ser movido da base para o topo, ele troca de lugar com cada elemento do vetor. Ele não os pula!
    - Elementos que já estão em suas posições finais são movidos de lugar, para depois voltarem
      - Exemplo: 5 2 3 8 1

Classificação por Trocas

- **Análise do Método da Bolha**
  - ✓Exercício:
    - Faça um algoritmo que se o vetor já está ordenado, ele pare de fazer comparações desnecessárias.
    - Em seguida analise o mesmo para o melhor, pior e médio caso.

Exercícios

1) Implemente uma nova versão dos algoritmos dos métodos *insertionSort*, *selectionSort* e *bubbleSort* para permitir ordenação decrescente.
   - ✓ Obs.: Utilize a nova versão do *bubbleSort* implementada no exercício anterior.

2) Faça um programa para verificar a estabilidade dos três métodos apresentados.