Disciplina: Estrutura de Dados

Professora: Jaqueline Brigladori Pugliesi

Métodos de Ordenação ou Métodos de Classificação

Conceitos Básicos

- Ordenar: processo de rearranjar um conjunto de objetos em uma ordem ascendente ou descendente.
- A ordenação visa facilitar a recuperação posterior de itens do conjunto ordenado.
    - Catálogo telefônico.
- Notação utilizada nos algoritmos:
    - Os algoritmos trabalham sobre os registros de um arquivo.
    - Cada registro possui uma chave utilizada para controlar a ordenação.
    - Podem existir outros componentes em um registro.
- Os métodos de classificação interna são divididos em cinco grupos de acordo com a técnica empregada, são eles:
    1. Classificação por Inserção (inserção direta e shellsort)
    2. Classificação por Troca (bubblesort e quicksort)
    3. Classificação por Seleção (seleção direta e heapsort)
    4. Classificação por Distribuição (distribuição de chaves e radixsort)
    5. Classificação por Intercalação (mergesort)

Classificação por Troca

- Na classificação por troca, a classificação de um conjunto de registros é feita através de comparações entre os elementos e trocas sucessivas desses elementos entre posições no arquivo.

Método BubbleSort

- É o método mais simples, de entendimento e programação mais fáceis.
- É um dos mais conhecidos e utilizados métodos de ordenação de arquivos.
- Não apresenta um desempenho satisfatório se comparado com os demais.

- Baseia-se em trocas de valores entre posições consecutivas, levando os valores mais altos (ou mais baixos) para o final do arquivo.
- Exemplo:

18 15 20 12	compara os dois primeiros elementos – desordenado – troca
15 18 20 12	segundo com o terceiro – ordenados,
	então terceiro com o quarto – desordenado – troca
15 18 12 20	chegamos ao final – o 20 está na posição final!
	trocar o segundo com o terceiro elemento
15 12 18 20	chega ao fim e reinicia o processo
	compara os dois primeiros elementos – desordenado – troca
12 15 18 20	Arquivo ordenado!!!

Método Quicksort

- Proposto por Hoare em 1960 e publicado em 1962.
- É o algoritmo de ordenação interna mais rápido que se conhece para uma ampla variedade de situações.
- Provavelmente é o mais utilizado.
- A ideia básica é dividir o problema de ordenar um conjunto com n itens em dois problemas menores.
- Os problemas menores são ordenados independentemente.
- Os resultados são combinados para produzir a solução final.
- A parte mais delicada do método é o processo de partição.
- O vetor A[Esq..Dir] é rearranjado por meio da escolha arbitrária de um pivô x.
- O vetor A é particionado em duas partes:
  - a parte esquerda com chaves menores ou iguais a x.
  - a parte direita com chaves maiores ou iguais a x.
- Algoritmo para o particionamento:
  1. Escolha arbitrariamente um pivô x.
  2. Percorra o vetor a partir da esquerda até que $A[i] \geq x$.
  3. Percorra o vetor a partir da direita até que $A[j] \leq x$.
  4. Troque A[i] com A[j].
  5. Continue este processo até os apontadores i e j se cruzarem.

- Ao final, o vetor A[Esq::Dir] está particionado de tal forma que:
  - Os itens em A[Esq], A[Esq + 1], ..., A[j] são menores ou iguais a x.
  - Os itens em A[i], A[i + 1], ..., A[Dir] são maiores ou iguais a x.
- Ilustração do processo de partição:

  1 2 3 4 5 6

  O R **D** E N A

  A R **D** E N O

  A **D** R E N O
- O pivô x é escolhido como sendo A[(i + j) / 2].
- Como inicialmente i = 1 e j = 6, então x = A[3] = D.
- Ao final do processo de partição i e j se cruzam em i = 3 e j = 2.
- O anel interno do procedimento Particao é extremamente simples.
- Razão pela qual o algoritmo Quicksort é tão rápido.
- Exemplo do estado do vetor em cada chamada recursiva da função Ordena:

  Chaves iniciais: O R D E N A

  1 A D R E N O

  2 A D

  3 E R N O

  4 N R O

  5 O R

  A D E N O R
- Vantagens:
  - É extremamente eficiente para ordenar arquivos de dados.
  - Necessita de apenas uma pequena pilha como memória auxiliar.
  - Requer cerca de n log n comparações em média para ordenar n itens.
- Desvantagens:
  - Sua implementação é muito delicada e difícil: um pequeno engano pode levar a efeitos inesperados para algumas entradas de dados.
  - O método não é estável.

Classificação por Seleção

- É realizada através de sucessivas seleções do menor elemento.
- Durante o processo o menor valor encontrado é colocado na sua posição correta final e o processo é repetido para os elementos que ainda não foram selecionados.

Método Seleção Direta

- Um dos algoritmos mais simples de ordenação.
- Possui melhor desempenho que o método Bubblesort, porém, só deve ser utilizado em pequenos conjuntos de dados.
- Baseia-se na fixação da primeira posição e na busca do menor elemento quando então é feita a troca dos valores. No final, tem-se o menor valor (ou o maior, conforme a comparação) na primeira posição.
- Este primeiro passo garante que o menor elemento fique na primeira posição.
- Continuando a buscar os demais elementos, comparando-os com a segunda posição (já desconsiderando a primeira posição, que foi anteriormente ordenada em relação ao arquivo como um todo).
- Exemplo:

  10 20 12 5 8 15 fixa a 1ª posição, procura o menor elemento e troca pela 1ª posição

  5 20 12 10 8 15 fixa a 2ª posição e repete o processo

  5 8 12 10 20 15 3ª posição e repete o processo

  5 8 10 12 20 15

  5 8 10 12 20 15

  5 8 10 12 15 20 Arquivo ordenado!
- Vantagens:
  - Custo linear no tamanho da entrada para o número de movimentos de registros.
  - É o algoritmo a ser utilizado para arquivos com registros muito grandes.
  - É muito interessante para arquivos pequenos.
- Desvantagens:
  - O fato de o arquivo já estar ordenado não ajuda em nada, pois o custo continua quadrático.
  - O algoritmo não é estável.

Método Heapsort

- Possui o mesmo princípio de funcionamento da ordenação por seleção.
- Algoritmo:
    - Selecione o menor item do vetor.
    - Troque-o com o item da primeira posição do vetor.
    - Repita essas duas operações com os n–1 itens restantes, depois com os n–2 itens, até que reste apenas um elemento.
- O custo para encontrar o menor (ou o maior) item entre n itens é n–1 comparações.
- Isso pode ser reduzido utilizando uma fila de prioridades.

Filas de Prioridades

- É uma estrutura de dados onde a chave de cada item reflete sua habilidade relativa de abandonar o conjunto de itens rapidamente.
- Aplicações:
    - SOs usam filas de prioridades, nas quais as chaves representam o tempo em que eventos devem ocorrer.
    - Métodos numéricos iterativos são baseados na seleção repetida de um item com maior (menor) valor.
    - Sistemas de gerência de memória usam a técnica de substituir a página menos utilizada na memória principal por uma nova página.

Filas de Prioridades - Tipo Abstrato de Dados

- Operações:
    1. Constrói uma fila de prioridades a partir de um conjunto com n itens.
    2. Informa qual é o maior item do conjunto.
    3. Retira o item com maior chave.
    4. Insere um novo item.
    5. Aumenta o valor da chave do item i para um novo valor que é maior que o valor atual da chave.
    6. Substitui o maior item por um novo item, a não ser que o novo item seja maior.
    7. Altera a prioridade de um item.
    8. Remove um item qualquer.
    9. Ajunta duas filas de prioridades em uma única.

Filas de Prioridades - Algoritmos de Ordenação

- As operações das filas de prioridades podem ser utilizadas para implementar algoritmos de ordenação.
- Basta utilizar repetidamente a operação Insere para construir a fila de prioridades.
- Em seguida, utilizar repetidamente a operação Retira para receber os itens na ordem reversa.
- O uso de listas lineares não ordenadas corresponde ao método da seleção.
- O uso de listas lineares ordenadas corresponde ao método da inserção.
- O uso de heaps corresponde ao método Heapsort.

Heaps

- É uma sequência de itens com chaves c[1], c[2], ..., c[n], tal que:

  $c[i] \geq c[2i]$;

  $c[i] \geq c[2i + 1]$;

para todo i = 1; 2; ... ; n/2.

- A definição pode ser facilmente visualizada em uma árvore binária completa.
- O algoritmo não necessita de nenhuma memória auxiliar.
- Dado um vetor A[1], A[2], ..., A[n].
- Os itens A[n/2 + 1], A[n/2 + 2], ..., A[n] formam um heap:
  - Neste intervalo não existem dois índices i e j tais que j = 2i ou j = 2i + 1.

	1	2	3	4	5	6	7
Chaves iniciais:	O	R	D	E	N	A	S
Esq = 3	O	R	**S**	E	N	A	**D**
Esq = 2	O	R	S	E	N	A	D
Esq = 1	**S**	R	**O**	E	N	A	D

1	2	3	4	5	6	7
S	R	O	E	N	A	D
R	**N**	O	E	**D**	A	**S**
O	N	**A**	E	D	R	
N	**E**	A	**D**	O		
E	**D**	A	N			
D	**A**	E				
A	D					

- Os itens de A[4] a A[7] formam um heap.
- O heap é estendido para a esquerda (Esq = 3), englobando o item A[3] pai dos itens A[6] e A[7].

- A condição de heap é violada:
  - O heap é refeito trocando os itens D e S.
- O item R é incluindo no heap (Esq = 2), o que não viola a condição de heap.
- O item O é incluindo no heap (Esq = 1).
- A Condição de heap violada:
  - O heap é refeito trocando os itens O e S heap, encerrando o processo.
- Vantagens:
  - O comportamento do Heapsort é sempre O(n log n), qualquer que seja a entrada.
- Desvantagens:
  - O anel interno do algoritmo é bastante complexo se comparado com o do Quicksort.
  - O Heapsort não é estável.

Classificação por Inserção

Método Inserção Direta

- Método preferido dos jogadores de cartas.
- Algoritmo:
  - Em cada passo a partir de i=2 faça:
    - Selecione o i-ésimo item da sequência fonte.
    - Coloque-o no lugar apropriado na sequência destino de acordo com o critério de ordenação.
- O método é ilustrado abaixo:

	1	2	3	4	5	6
Chaves iniciais:	**O**	R	D	E	N	A
i = 2	**O**	**R**	D	E	N	A
i = 3	**D**	**O**	**R**	E	N	A
i = 4	**D**	**E**	**O**	**R**	N	A
i = 5	**D**	**E**	**N**	**O**	**R**	A
i = 6	**A**	**D**	**E**	**N**	**O**	**R**

- As chaves em negrito representam a sequência destino.

- O processo de ordenação pode ser terminado pelas condições:
    - Um item com chave menor que o item em consideração é encontrado.
    - O final da sequência destino é atingido à esquerda.
- Solução:
    - Utilizar um registro sentinela na posição zero do vetor.

Método Shellsort

- Proposto por Shell em 1959. É uma extensão do algoritmo de ordenação por inserção direta.
- Problema com o algoritmo de ordenação por inserção direta:
    - Troca itens adjacentes para determinar o ponto de inserção.
    - São efetuadas n - 1 comparações e movimentações quando o menor item está na posição mais à direita no vetor.
- O método de Shell contorna este problema permitindo trocas de registros distantes um do outro.
- Os itens separados de h posições são rearranjados.
- Todo h-ésimo item leva a uma sequência ordenada.
- Tal sequência é dita estar h-ordenada.
- Exemplo de utilização:

	1 2 3 4 5 6
Chaves iniciais:	O R D E N A
h = 4	N A D E O R
h = 2	D A N E O R
h = 1	A D E N O R

- Quando h = 1 Shellsort corresponde ao algoritmo de inserção direta.
- A implementação do Shellsort não utiliza registros sentinelas.
- Seriam necessários h registros sentinelas, uma para cada h-ordenação.
- Ordenação com sequência h = {1,3,4}

25 57 48 37 12 92 86 33

	1	2	3	4	5	6	7	8
1.	25	57	48	37	12	92	86	33

h = 4 - foram comparadas as posições 1 com 5 (trocados 25 e 12), 2 com 6 (sem trocas), 3 com 7 (sem trocas) e 4 com 8 (trocados 37 com 33).

2.	12	57	48	33	25	92	86	37

h = 3 - foram comparadas as posições 1 com 4, com 7 (sem trocas), 2 com 5 com 8 (2 trocas) e 3 com 6 (sem trocas).

3.	12	25	48	33	37	92	86	57

h = 1 - A partir deste ponto o procedimento se reduz ao da ordenação por inserção direta.

4.	12	25	33	37	48	57	86	92

Resultado final.