TRABALHO – ESTRUTURA DE DADOS

Aluna: Amanda Mendes Reis de Araújo

Curso: Ciência da Computação – 2 período

Algoritmos de ordenação

1)O que é?

Os algoritmos de ordenação se referem ao conjunto de instruções ou dados, colocados em uma dada sequência em uma certa ordem. São amplamente utilizados em aplicações que envolvem manipulação de dados, busca ou analise dos mesmos.

Tais estruturas e algoritmos são importantes devido à redução da de alguns problemas. Cada algoritmo de ordenação requer diferentes níveis de complexidade.

1) Tipos

Existem alguns diferentes tipos de algoritmos de ordenação, são eles:

- bubble sort.
- select sort.
- insert sort.
- merge sort.
- quick sort .
- heap sort.
- shell sort.

Neste trabalho serão apresentadas as explicações dos seguintes modelos: algoritmos de ordenação merge, quick sort, shell sort e o heap sort.

2) MERGE SORT

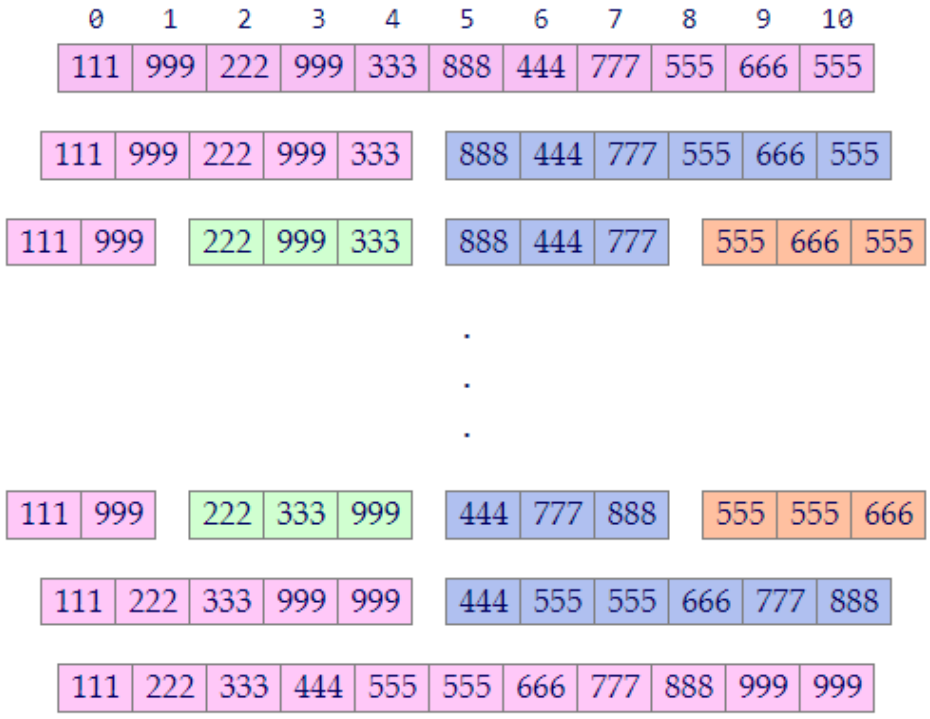
3.1 - O Merge Sort é um algoritmo de ordenação que é conhecido também pela frase “dividir e conquistar”. O mesmo envolve a quebra de uma estrutura (vetor) em subgrupos menores e a aplicação da ordenação nos elementos extraídos da estrutura original. Depois que esses subgrupos estão ordenados, eles são combinados (merge)

em um conjunto final que está ordenado. Quando utilizado em maiores quantidades de dados, se mostra mais eficiente que os demais algoritmos.

3.2

- Exemplo imagem:

Repare que o grupo maior é subdivido em grupos menores. A cada iteração, os elementos são reorganizados, resultando na combinação dos subconjuntos novamente. O final demonstra a frase “Dividir para conquistar”.



Fonte: https://www.ime.usp.br/~pf/algoritmos/aulas/mrgsrt.html

3) QUICK SORT

4.1 O Quick sort, diferente do Merge (exemplo 1) necessita de menos memória, ou seja, ocupa um espaço que não há necessidade extra.

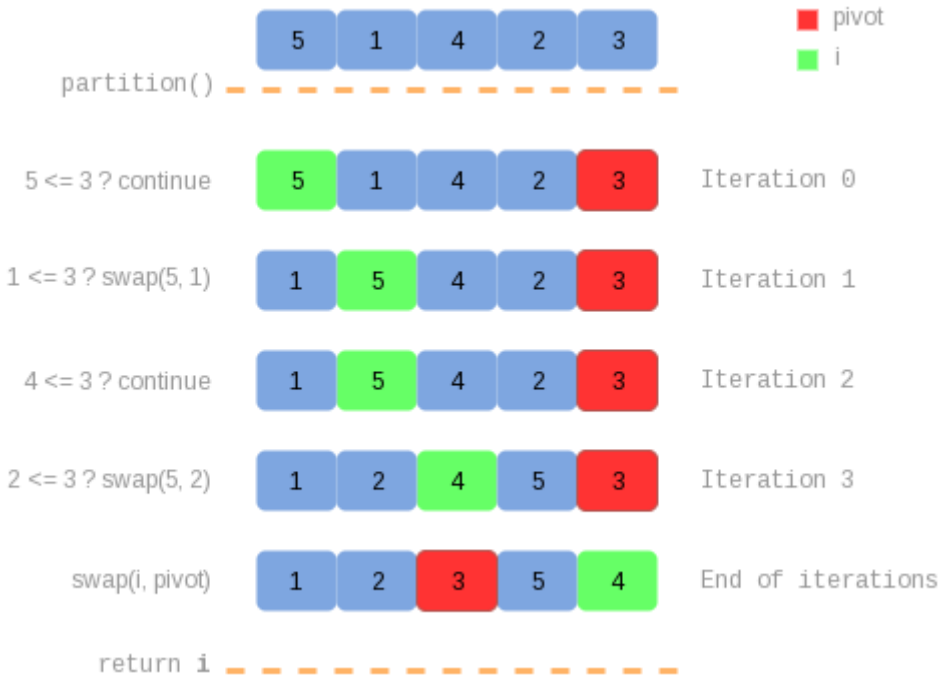
Neste exemplo, vamos dividir a explicação do Quick sort em etapas.

- **Escolha do Pivô:** Seleciona-se um elemento da lista, chamado de pivô. A escolha do pivô pode ser feita pelo primeiro elemento, último elemento, mediana, etc.
- **Particionar:** A lista é organizada de acordo com a posição que o pivô ocupe, sendo assim, a ordem é organizada deixando elementos menores a sua esquerda e maiores a sua direita.
- **Recursão:** O algoritmo é aplicado recursivamente as duas sublistas resultantes (esquerda e a direita do pivô).
- **Base da Recursão:** O processo continua até que as sublistas tenham um único elemento ou estejam vazias, momento em que estão naturalmente ordenadas.

4.2

- Exemplo imagem:

Repare que neste exemplo o 3 é escolhido como pivô na sequência dada. A medida que as iterações acontecem, as condições são analisadas. Caso o número seja maior que 3, ele é alocado a direita e o próximo numero é analisado. No final da sequência, o pivô recebe os números menos que ele a sua esquerda e os maiores a sua direita.



Fonte: https://blog.pantuza.com/artigos/o-algoritmo-de-ordenacao-quicksort

4) SHELL SORT

5.1 O **shell sort** também chamado de ordenação por incrementos diminutos, melhora a ordenação por inserção ao quebrar a lista original, a-dividindo em sublistas, as quais são ordenadas, inicialmente. Para cada sublista, é aplicado o Insertion Sort.O processo é repetido, diminuindo o valor do intervalo até que ele seja 1, momento em que o algoritmo se comporta como um Insertion Sort.

5.2 Vantagens

- Melhora a eficiência do Insertion Sort, especialmente para listas grandes.
- É um algoritmo in-place, o que significa que requer pouca memória adicional.

5.3 Exemplo

E	X	E	M	P	L	O
E	X	E	M	P	L	O
E	L	E	M	P	X	O
E	L	E	M	P	X	O

h = 4

E	L	E	M	P	X	O
E	L	E	M	P	X	O
E	L	E	M	P	X	O
E	L	E	M	P	X	O
E	L	E	M	P	X	O
E	L	E	M	O	X	P

h = 2

h = 1 (inserção)

E	L	E	M	O	X	P
E	L	E	M	O	X	P
E	L	E	M	O	X	P
E	E	L	M	O	X	P
E	E	L	M	O	X	P
E	L	E	M	O	X	P
E	L	E	M	O	X	P
E	L	E	M	O	P	X

Fonte: material pdf universidade de minas gerais (https://homepages.dcc.ufmg.br/~cunha/teaching/20121/aeds2/shellsort.pdf)

6) Heap sort

6.1 – O Heap sort utiliza uma estrutura de dados chamada heap binário. A medida que os elementos são inseridos na estrutura, o heap binário os ordena. Citaremos os dois tipos de heap, os máximos e mínimos. O heap máximo tem ordenação crescente (seu maior elemento fica na raiz, sendo o nó pai maior ou igual que o nó filho), ao contrario do mínimo que o seu elemento da raiz é o menor.

vantagens:

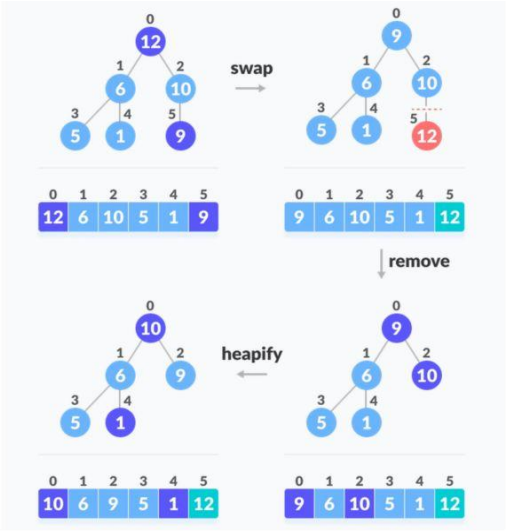
- **Complexidade de Tempo:** Garante **O(n log n)** no pior caso.
- **In-Place:** Necessita de apenas uma quantidade constante de espaço adicional.

desvantagens:

- **Não-Estável:** Em sua forma padrão, o Heap Sort não mantém a ordem relativa de elementos iguais.
- **Acesso à Memória:** Requer acesso aleatório à memória, o que pode não ser tão eficiente para certos tipos de dados ou estruturas de memória.

6.2

- Exemplo imagem:
- Nesta imagem veremos o exemplo da transformação de um heap. (Exemplo completo no link da referência).




```
heapsort(L, n) {
      # Constrói max heap
      for i = n//2-1 downto 0
          heapify(L, n, i)
      for i = n-1 downto 1 {
          # Troca
          swap(L[i], L[0])
          # Heapifica o elemento raiz
          heapify(L, i, 0)
      }
}
```

Fonte: https://www.linkedin.com/posts/alexandre-levada-45a90533_o-algoritmo-heapsort-utiliza-um-max-heap-activity-7163528598128201728-yvqf/?originalSubdomain=pt