Curso: Sistemas de Informação / Engenharia de Software

Disciplina: Estrutura de Dados II

Professora: Jaqueline Brigladori Pugliesi

Alunos: Ramon Luiz Souza - 21079

Eduardo de Paula Xavier - 22570

1. **Tempo de execução**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bubblesort** | | | | |
|  | 100 | 1000 | 10.000 | 100.000 |
| Aleatória | 0.00002s | 0.0089s | 0.325s | 27.09s |
| Crescente | 0.000004s | 0.0034s | 0.0028s | 0.00017s |
| Decrescente | 0.00003s | 0.0014s | 0.178s | 14.81s |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Seleção direta** | | | | |
|  | 100 | 1000 | 10.000 | 100.000 |
| Aleatória | 0.00004s | 0.0034s | 0.218s | 19.96s |
| Crescente | 0.00002s | 0.0052s | 0.224s | 20.05s |
| Decrescente | 0.00002s | 0.0026s | 0.238s | 25.40s |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Inserção direta** | | | | |
|  | 100 | 1000 | 10.000 | 100.000 |
| Aleatória | 0.00002s | 0.006s | 0.104s | 10.41s |
| Crescente | 0.000005s | 0.003s | 0.0035s | 0.00059s |
| Decrescente | 0s | 0.000005s | 0.00004s | 0.00048s |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Quicksort** | | | | |
|  | 100 | 1000 | 10.000 | 100.000 |
| Aleatória | 0.0002s | 0.0051s | 0.032s | 0.087s |
| Crescente | 0.0001s | 0.0019 | 0.0095s | 0.053s |
| Decrescente | 0.00004s | 0.0010 | 0.0092s | 0.047s |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Shellsort** | | | | |
|  | 100 | 1000 | 10.000 | 100.000 |
| Aleatória | 0.00002s | 0.0070s | 0.010s | 0.029s |
| Crescente | 0.000005s | 0.0044s | 0.0056s | 0.0086s |
| Decrescente | 0.00001s | 0.000054s | 0.0014s | 0.0058s |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Heapsort** | | | | |
|  | 100 | 1000 | 10.000 | 100.000 |
| Aleatória | 0.00001s | 0.0076s | 0.013s | 0.020s |
| Crescente | 0.00002s | 0.0055s | 0.0048s | 0.017s |
| Decrescente | 0.00001s | 0.00010s | 0.0010s | 0.012s |

1. **Comparação dos métodos**

**Vantagens:**

Bubblesort: O algoritmo é muito simples e facilita muito o aprendizado, os elementos são trocados de lugar sem utilizar armazenamento temporário o que faz o requerimento de espaço ser mínimo.

Seleção direta: Não apresenta melhorias significativas em execução quando o vetor já está organizado, não é estável e rápido para vetores pequenos

Inserção Direta: Faz menos comparações quando o vetor já está ordenado, quando o vetor está completamente em ordem reversa o algoritmo age de forma natural para a sua ordenação, é um algoritmo estável

Quick Sort: Rápida execução em vetores grandes, necessita de apenas uma pequena pilha como memória auxiliar, requer O(n log n) comparações em média (caso médio) para ordenar n itens.

Heapsort: O comportamento do Heapsort é sempre O(n log n), qualquer que seja a entrada. Não necessita de nenhuma memória adicional. É bom para arquivos com grandes registros.

Shellsort: É uma ótima opção para arquivos de tamanho moderado. Sua implementação é simples e requer uma quantidade de código pequena

**Desvantagens:**

Bubblesort: É o método mais simples, porém é o menos eficiente para listas grandes, a maior desvantagem que esse método possui é a quantidade de tempo que leva para classificar. O tempo médio aumenta quase exponencialmente à medida que o número de elementos da tabela aumenta.

Seleção direta: Ele é um algoritmo simples, não necessita de um vetor auxiliar, o que ocupa menos espaço de memória.

Inserção direta: Contém um alto custo de movimentação de elementos no vetor.

Quicksort: Método não é estável, tem um pior caso O(n2) comparações e difícil implementação.

Heapsort: Este método não é recomendado para vetores de entrada com poucos elementos, devido a complexidade do heap. O Heapsort não é estável. O anel interno do algoritmo é bastante complexo se comparado com o do Quicksort.

Shellsort: O tempo de execução do algoritmo é sensível à

ordem inicial do arquivo, o método não é estável.

1. **Pior e melhor caso**

Bubblesort:

Melhor caso: Quando o vetor é pequeno e já se encontra ordenado (0.000004s), nenhuma troca ocorre na primeira varredura.

Pior caso: Quando o vetor é grande e se encontra na ordem aleatória

(27.09s).

Seleção direta:

Melhor caso: Quando o vetor é pequeno e já se encontra ordenado ou decrescente (0.00002s).

Pior caso: Quando o vetor é grande e se encontra na ordem inversa a desejada (25.40s).

Inserção direta:

Melhor caso: Quando o vetor é pequeno e se encontra ordenado de forma inversa (0s).

Pior caso: Quando o vetor é grande se encontra em ordem aleatória (10.41s).

Quick sort:

Melhor caso: Acontece em vetor pequeno ordenado de forma decrescente (0.00004s).

Pior caso: Quando o vetor é grande e aleatório (0.087s).

Shellsort:

Melhor caso: Ocorre em vetor pequeno e já ordenado (0.000005s)

Pior caso: Ocorre em vetor grande e aleatório (0.029s).

Heapsort:

Melhor caso: Em algoritmos aleatórios de tamanho 100 ele faz (0.00001s) e decrescentes(0.00001s) ele faz o mesmo tempo se tratando de um número pequeno.

Pior caso: Em algoritmos aleatórios de tamanho 100.000 ele demora mais para ordenar (0.020s).

Geral:

Melhor caso: Inserção direta com vetor pequeno de forma decrescente (0s).

Pior caso: Bubblesort vetor grande aleatório(27.09s).

1. **Complexidade**

Bubble Sort: Complexidade simples com a ideia de percorrer o [vetor](https://pt.wikipedia.org/wiki/Vector) diversas vezes, e a cada passagem fazer flutuar para o topo o maior elemento da sequência.

No melhor caso, o algoritmo executa *n* operações relevantes, onde *n* representa o número de elementos do vetor. No pior caso, são feitas *n²* operações.

Complexidade pior caso *O(n²)*.

Complexidade melhor caso *O(n)*.

Seleção direta: É um [algoritmo de ordenação](https://pt.wikipedia.org/wiki/Algoritmo_de_ordena%C3%A7%C3%A3o) baseado em se passar sempre o menor valor do vetor para a primeira posição (ou o maior dependendo da ordem requerida), depois o de segundo menor valor para a segunda posição, e assim é feito sucessivamente com os *n-1* elementos restantes, até os últimos dois elementos.

Complexidade pior caso *O(n²).*

Complexidade melhor caso *O(n²).*

Inserção direta: É o algoritmo de ordenação que, dado uma estrutura (array, lista) constrói uma matriz final com um elemento de cada vez, uma inserção por vez. Assim como algoritmos de ordenação quadrática, é bastante eficiente para problemas com pequenas entradas, sendo o mais eficiente entre os algoritmos desta ordem de classificação.

complexidade pior caso *O(n²)*.

complexidade melhor caso *O(n)*.

Quick sort: É um método de ordenação muito rápido e eficiente. O quicksort adota a estratégia de [divisão e conquista](https://pt.wikipedia.org/wiki/Divis%C3%A3o_e_conquista). A estratégia consiste em rearranjar as chaves de modo que as chaves "menores" precedem as chaves "maiores". Em seguida o quicksort ordena as duas sublistas de chaves menores e maiores recursivamente até que a lista completa se encontre ordenada,

Complexidade pior caso *O(n²).*

Complexidade melhor caso *O(n log n).*

Shellsort: Shell sort é o mais eficiente [algoritmo](https://pt.wikipedia.org/wiki/Algoritmo) de [classificação](https://pt.wikipedia.org/wiki/Ordena%C3%A7%C3%A3o_(computa%C3%A7%C3%A3o)) dentre os de [complexidade](https://pt.wikipedia.org/wiki/An%C3%A1lise_de_algoritmos) quadrática. Basicamente o algoritmo passa várias vezes pela lista dividindo o grupo maior em menores. Nos grupos menores é aplicado o método da [ordenação por inserção](https://pt.wikipedia.org/wiki/Insertion_sort).

Complexidade no pior caso *O(n log2 n).*

Complexidade no melhor caso *O(n log2 n).*

Heapsort: O heapsort utiliza uma estrutura de dados chamada [heap](https://pt.wikipedia.org/wiki/Heap), para ordenar os elementos à medida que os insere na estrutura. Assim, ao final das inserções, os elementos podem ser sucessivamente removidos da raiz da heap, na ordem desejada, lembrando-se sempre de manter a propriedade de max-heap.

Complexidade no pior caso: *O(n log n).*

Complexidade no melhor caso: *O(n log n).*

1. **Referências**

<https://pt.wikipedia.org/wiki/Bubble_sort>

<https://pt.wikipedia.org/wiki/Selection_sort>

<https://pt.wikipedia.org/wiki/Insertion_sort>

<https://pt.wikipedia.org/wiki/Quicksort>

<https://pt.wikipedia.org/wiki/Shell_sort>

<https://pt.wikipedia.org/wiki/Heapsort>

Acesso em: 15/09/2019