Trabalho de Análise de Algoritmos.

Nome: Beatriz Borges, Beatriz Saito, Maria Angelica, Taynan Mancilla.

Selection Sort

A ordenação de elementos é uma operação fundamental na ciência da computação, desempenhando um papel crucial em diversas aplicações. Entre os vários algoritmos de classificação, o Selection Sort é um dos métodos mais simples e intuitivos. O Selection Sort é um algoritmo de ordenação por comparação que se destaca por sua facilidade de compreensão e implementação. Neste relatório, exploraremos a classificação por Selection Sort em detalhes, examinando sua lógica de funcionamento, complexidade assintótica e desempenho em comparação com outros algoritmos de classificação, como o Bubble Sort, o Insertion Sort, o Shell Sort e o QuickSort.

O **Selection Sort** (Ordenação por Seleção) é um algoritmo de ordenação que funciona selecionando o menor elemento de um arranjo e colocando-o na primeira posição, depois selecionando o segundo menor elemento e colocando-o na segunda posição, e assim sucessivamente até que o arranjo esteja ordenado. O Selection Sort se destaca em relação aos outros algoritmos de ordenação em dois pontos: (i) por realizar um número mínimo de trocas, e (ii) não necessitar de uma estrutura de dados auxiliar para funcionar. Por esses motivos, o Selection Sort é indicado para a ordenação de arranjos quando memória extra disponível para uso é escassa, ou então o custo para trocar-se elementos de posição no arranjo é elevado.

Ao comparar o Selection Sort com o Bubble Sort, notamos que ambos compartilham uma complexidade assintótica de O(n^2) em sua versão iterativa, onde 'n' representa o tamanho do conjunto a ser ordenado. No entanto, nas versões recursivas, o Selection Sort supera o Bubble Sort em termos de tempo de execução. O Selection Sort Recursivo levou 15.730043 segundos, enquanto o Bubble Sort Recursivo levou 16.229225 segundos. Essa diferença de desempenho pode ser atribuída à natureza das iterações, uma vez que o Bubble Sort realiza trocas repetidas, enquanto o Selection Sort faz uma única troca por passagem. A relação de recorrência em ambas as versões recursivas é semelhante, com chamadas recursivas em subconjuntos menores, mas o Bubble Sort sofre um pouco mais em relação ao Selection Sort, como refletido nos tempos de execução.

A análise dos algoritmos Selection Sort e Insertion Sort também revelou diferenças notáveis em seus tempos de execução. O Selection Sort, em sua versão iterativa, apresentou um tempo de execução de 16.186826 segundos, enquanto a versão recursiva teve um desempenho ligeiramente melhor, com 15.730043 segundos. Em contraste, o Insertion Sort demonstrou um desempenho significativamente mais eficiente, com tempos de execução muito menores, 9.561691 segundos na versão iterativa e 9.551868 segundos na versão recursiva. Ambos os algoritmos possuem complexidades de O(n^2) no pior caso, mas o Insertion Sort se destaca em cenários de arrays parcialmente ordenados, saindo mais cedo das iterações internas. A relação de recorrência da versão recursiva em ambos os algoritmos é semelhante, com T(n) = T(n-1) + O(n), mas o Insertion Sort é mais eficiente em situações em que o array de entrada já está parcialmente ordenado.

A comparação entre o Selection Sort e o Shell Sort revelou diferenças substanciais em termos de desempenho. O Selection Sort, em ambas as versões, apresentou tempos de execução significativamente mais longos em comparação com o Shell Sort. O tempo de execução do Selection Sort na versão iterativa foi de 16.186826 segundos, enquanto na versão recursiva foi de 15.730043 segundos. Em contraste, o Shell Sort demonstrou um desempenho excepcional, com

tempos de execução extremamente curtos, 0.040578 segundos na versão iterativa e 0.040778 segundos na versão recursiva. O Selection Sort possui complexidade O(n^2) no pior caso, devido ao processo de comparação e troca de elementos em todas as iterações. Em contrapartida, o Shell Sort é conhecido por ter uma complexidade média de O(n log^2 n) na versão iterativa. A análise exata da complexidade do Shell Sort depende do intervalo escolhido para as lacunas (gaps), mas, em geral, é mais eficiente do que o Selection Sort.

Na comparação final, o algoritmo de ordenação QuickSort superou o Selection Sort de maneira notável. Tanto na versão iterativa quanto na recursiva, o QuickSort apresentou tempos de execução significativamente mais curtos, 0.025358 segundos e 0.025036 segundos, respectivamente, em comparação com as versões correspondentes do Selection Sort, que levaram cerca de 16 segundos cada. Essa diferença acentuada pode ser atribuída à complexidade assintótica mais favorável do QuickSort, que é, em média, O(n log n), enquanto o Selection Sort tem complexidade O(n^2). A relação de recorrência da versão recursiva do QuickSort é geralmente expressa como T(n) = 2T(n/2) + O(n), refletindo seu desempenho superior em comparação com o Selection Sort.

Quando se trata de comparações, o MergeSort é outra adição importante a esta análise. O Selection Sort, tanto em sua versão iterativa quanto na recursiva, apresentou tempos de execução significativamente mais longos, levando cerca de 16 segundos cada. Isso se deve à natureza quadrática do Selection Sort, que não se comporta bem com conjuntos de dados grandes. Por outro lado, o MergeSort, nas versões iterativa e recursiva, demonstrou ser muito mais rápido, levando apenas cerca de 0,03 segundos. O desempenho superior do MergeSort se deve à sua análise assintótica de O(n log n), tornando-o altamente eficiente, especialmente para conjuntos de dados maiores.

Em resumo, a análise comparativa revelou que o Selection Sort, embora seja simples de entender e implementar, é menos eficiente em relação ao Insertion Sort, Shell Sort, QuickSort e MergeSort. A escolha do algoritmo de ordenação deve levar em consideração as características dos dados e as necessidades de desempenho específicas do contexto, garantindo que os dados usados na análise sejam mantidos para futuras referências.