**MÉTODOS ELEMENTARES DE ORDENAÇÃO – ANÁLISE EMPÍRICA**

**Bubble Sort - Elapsed sorting time in seconds**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **1K** | **10K** | **100K** |
| **Nearly Sorted** | 0.002015 | 0.193291 | 18.356081 |
| **Reverse Sorted** | 0.005176 | 0.452123 | 41.470691 |
| **Sorted** | 0.001949 | 0.187330 | 18.391421 |
| **Unif Rand** | 0.003790 | 0.241927 | 40.397652 |

O bubble sort lida melhor com vetores que estão parcialmente ou totalmente ordenados, obtendo a metade do tempo de execução em relação às entradas ordenadas de forma reversa e aleatórias.

**Shaker Sort - Elapsed sorting time in seconds**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **1K** | **10K** | **100K** |
| **Nearly Sorted** | 0.001775 | 0.299297 | 33.263597 |
| **Reverse Sorted** | 0.002283 | 0.239884 | 29.757094 |
| **Sorted** | 0.004273 | 0.327223 | 34.062456 |
| **Unif Rand** | 0.004265 | 0.365236 | 37.678179 |

O shaker sort teve desempenho pior do que o bubble sort exceto para casos maiores de entradas aleatórias e entradas ordenadas de forma reversa. O algoritmo parece insensível à ordenação dos dados, pois obteve tempos semelhantes para todos os casos e todos os N.

**Selection Sort - Elapsed sorting time in seconds**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **1K** | **10K** | **100K** |
| **Nearly Sorted** | 0.001183 | 0.103528 | 9.930046 |
| **Reverse Sorted** | 0.000959 | 0.092674 | 8.814414 |
| **Sorted** | 0.000975 | 0.103469 | 9.900164 |
| **Unif Rand** | 0.001098 | 0.108922 | 9.836782 |

O selection sort obteve tempos praticamente iguais para todos os tipos de entrada. Isso mostra que o selection sort é insensível aos dados de entrada e sua execução sempre se dará em N². Entretanto, esse método de ordenação possui o mínimo movimento de dados, havendo um número linear de trocas.

**Insertion Sort - Elapsed sorting time in seconds**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **1K** | **10K** | **100K** |
| **Nearly Sorted** | 0.001127 | 0.108124 | 10.350534 |
| **Reverse Sorted** | 0.001285 | 0.265107 | 30.894864 |
| **Sorted** | 0.001117 | 0.109155 | 10.324279 |
| **Unif Rand** | 0.002401 | 0.157433 | 21.377582 |

O insertion sort obteve um desempenho melhor quando as entradas já estavam ordenadas ou quase ordenadas. Se as entradas estão ordenadas, o algoritmo faz N - 1 comparações e 0 trocas, o que o faz ser mais rápido. Entretanto, quando as entradas estão ordenadas de forma reversa, o algoritmo faz 1/2N² comparações e 1/2N² trocas, o que o faz ser mais lento. Para as entradas quase ordenadas, vale a preposição de que o algoritmo roda em tempo linear, sendo o número de comparações igual ao número de inversões (par de chaves fora de ordem) + N - 1 e o número de trocas igual ao número de inversões. Isso justifica porque o tempo de execução para essas entradas foi semelhante às entradas ordenadas.