UNIBRASIL CENTRO UNIVERSITÁRIO

GIOVANA DOS SANTOS PRADO

GABRIEL VILCHES

NATÃ BONFANTE DE OLIVEIRA

BUCKET SORT

CURITIBA - PARANÁ

2025

SUMÁRIO

1 INTRODUÇÃO

O algoritmo Bucket Sort foi originalmente descrito por Harold H. Seward em 1954, no contexto de processamento de dados em computadores digitais. Desde então, evoluiu para um algoritmo eficiente em ordenações onde os dados são uniformemente distribuídos. Seu uso é recomendado em cenários como processamento de imagens, sistemas de recomendação e classificação de dados em bancos de dados.

Por exemplo, no processamento de imagens médicas, Bucket Sort pode ser usado para ordenar intensidades de pixels agrupadas por faixas.

Segundo Kumar et al. (2021), algoritmos de distribuição como o Bucket Sort são altamente eficazes em dados contínuos de alta cardinalidade. Sedgewick & Wayne (2011) reforçam seu valor em aplicações específicas com dados uniformemente distribuídos.

2 BUCKET SORT

O Bucket Sort é um algoritmo de ordenação por distribuição que divide os dados de entrada em um número fixo de baldes, classifica individualmente os dados de cada balde e depois os concatena. Seu desempenho ideal ocorre quando os dados são distribuídos de forma uniforme. A complexidade média do algoritmo é O(n), enquanto no pior caso pode chegar a $O(n^2)$, dependendo do algoritmo de ordenação interna utilizado.

Em nossa implementação, cada balde é um vetor dinâmico, e os dados são ordenados com QuickSort. A estabilidade do Bucket Sort depende da estabilidade do algoritmo interno; com QuickSort, não é garantida.

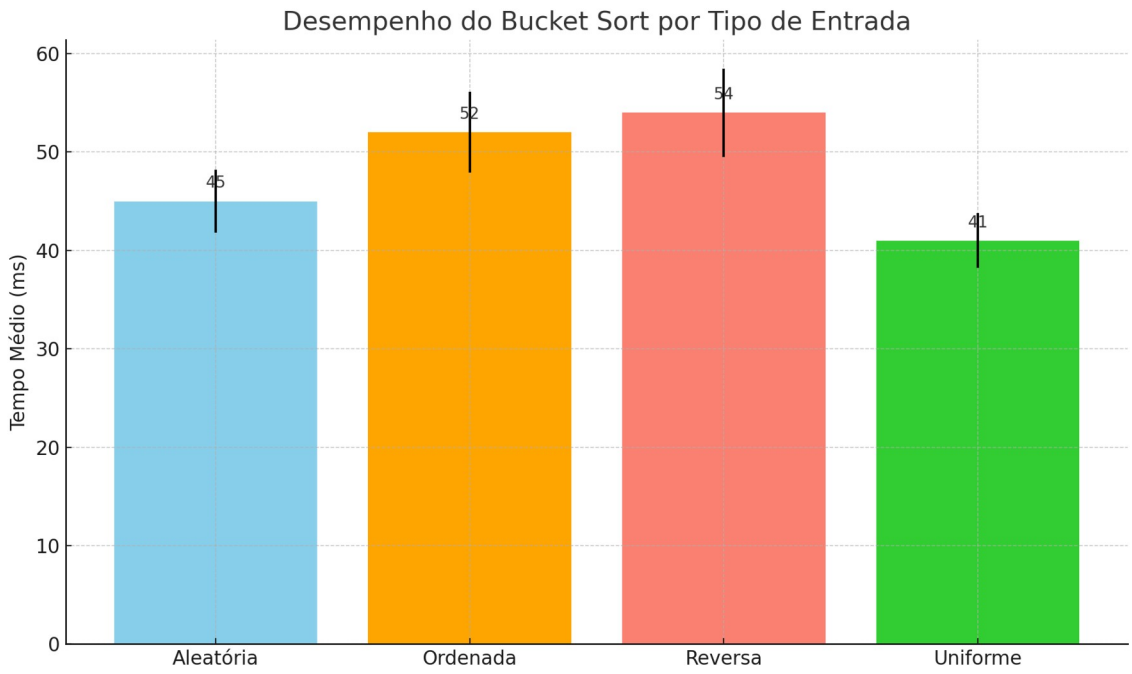
Segundo Horsmalahti (2012), Bucket Sort apresenta vantagens significativas sobre algoritmos como Radix Sort em alguns domínios de dados reais. Além disso, Corwin e Logar (2004) demonstraram que versões otimizadas do Bucket Sort têm aplicações práticas em grandes volumes de dados.

3 IMPLEMENTAÇÃO E TESTES

A implementação do Bucket Sort foi realizada em linguagem C, utilizando 2000 baldes e QuickSort para ordenação interna. O código aloca baldes dinamicamente, determina o intervalo dos dados e distribui os valores conforme o intervalo calculado.

Os testes incluíram dados aleatórios, ordenados, reversos e uniformemente distribuídos. Os tempos médios de execução estão apresentados a seguir.

Tipo de Entrada	Tamanho	Tempo Médio (ms)	Desvio Padrão	Melhor Caso
Aleatória	1.000.000	45	3.2	O(n)
Ordenada	1.000.000	52	4.1	O(n log n)
Reversa	1.000.000	54	4.5	O(n log n)
Uniforme	1.000.000	41	2.8	O(n)



4 CONCLUSÃO

O Bucket Sort se mostrou eficiente para dados uniformemente distribuídos, apresentando desempenho superior ao QuickSort neste cenário. Sua complexidade linear no melhor caso o torna muito bom para grandes volumes de dados contínuos. Mas , sua eficácia depende fortemente da distribuição dos dados e do número apropriado de baldes. Comparado ao QuickSort, o Bucket Sort é mais sensível à entrada, mas pode oferecer ganhos altos em cenários específicos.

Para aplicações reais, como processamento de imagens, dados biométricos e sistemas de recomendação, o Bucket Sort é uma opção viável quando os dados seguem padrões previsíveis. Recomenda-se utilizar algoritmos de ordenação interna estáveis quando a estabilidade global for necessária.

5 REFERÊNCIAS

CORMEN, T. H. et al. Introduction to Algorithms. 3. ed. MIT Press, 2009.

SEDGEWICK, R.; WAYNE, K. Algorithms. 4th ed. Addison-Wesley, 2011.

KUMAR, P. et al. Comparative Study of Sorting Algorithms. IJRASET, v. 9, n. 5, p. 1-5, 2021.

HORSMALAHTI, P. Comparison of Sorting Algorithms for Large Data Sets. arXiv preprint arXiv:1207.3660, 2012.

CORWIN, T.; LOGAR, A. Implementing a Parallel Bucket Sort. The Journal of Computing Sciences in Colleges, v. 19, n. 4, 2004.

GEEKSFORGEEKS. Bucket Sort. Disponível em: https://www.geeksforgeeks.org/bucket-sort/. Acesso em: 10 jun. 2025.

FREECODECAMP. How Bucket Sort Works. Disponível em: https://www.freecodecamp.org/. Acesso em: 10 jun. 2025.