Bucket Sort

O **Bucket Sort** é um algoritmo de ordenação que distribui os elementos em vários "baldes" e depois ordena esses baldes individualmente. O **Bucket Sort** é eficaz quando a entrada está uniformemente distribuída em um intervalo específico. Vamos primeiro entender o processo básico com números de ponto flutuante no intervalo [0,1)[0, 1)[0,1) e depois adaptá-lo para números inteiros em intervalos maiores.

Como o Bucket Sort funciona:

1. **Divisão em Baldes**: O intervalo de valores é dividido em baldes. Cada balde conterá uma faixa específica de valores.
2. **Distribuição**: Os elementos são distribuídos entre os baldes com base em seu valor. Por exemplo, valores entre 0 e 0.1 vão para o balde 1, valores entre 0.1 e 0.2 vão para o balde 2, e assim por diante.
3. **Ordenação**: Cada balde é ordenado individualmente (geralmente usando um algoritmo de ordenação eficiente como o **Insertion Sort** ou **Quick Sort**).
4. **Concatação**: Após a ordenação de cada balde, os baldes são concatenados de volta para formar a lista ordenada.

Adaptação para Ordenar Números Inteiros em Intervalos Maiores

Para adaptar o **Bucket Sort** para ordenar números inteiros positivos em intervalos maiores, podemos usar uma abordagem semelhante. Em vez de dividir os números no intervalo [0,1)[0, 1)[0,1), podemos dividir os números inteiros em intervalos maiores, como [0,k)[0, k)[0,k), onde kkk é o intervalo desejado (por exemplo, se quisermos ordenar números no intervalo [0, 100), usaremos 100 como kkk).

Como o Bucket Sort Funciona:

- **Baldes**: O intervalo dos números é dividido em kkk baldes (quanto maior o valor de kkk, mais distribuídos os elementos estarão).
- **Distribuição**: Cada número é atribuído a um balde com base no seu valor.
- **Ordenação**: Cada balde é ordenado (geralmente usando um algoritmo simples como **Insertion Sort** ou **Quick Sort**).
- **Combinação**: Os baldes ordenados são concatenados para formar a lista final ordenada.

Complexidade e Eficiência:

- **Tempo de execução**: O **Bucket Sort** tem uma complexidade de tempo de O(n+k⋅m)O(n + k \cdot m)O(n+k⋅m), onde nnn é o número de elementos e kkk é o

número de baldes. A complexidade depende de como os elementos são distribuídos e da eficiência do algoritmo de ordenação usado nos baldes.
- **Eficiência**: O **Bucket Sort** é muito eficiente quando os elementos estão distribuídos uniformemente. Para listas desbalanceadas, seu desempenho pode não ser tão bom quanto outros algoritmos, como **Quick Sort** ou **Merge Sort**.

Essa implementação do **Bucket Sort** funciona bem tanto para números flutuantes no intervalo [0,1)[0, 1)[0,1) quanto para números inteiros em intervalos maiores.