Câmpus Campos do Jordão

Tecnologia em Análise e Desenvolvimento de Sistemas – TADS

EDDA3 – ESTRUTURA DE DADOS

Manual de Uso do Protocolo Bucket Sort em Linguagem C

Aluno: Wedyner Rodrigo Maciel – CJ 3019462

Professor: Marques Moreira de Souza

Campos do Jordão

2023

Wedyner Rodrigo Maciel

1 Sumário

1 – Introdução:

O Protocolo Bucket Sort é uma adaptação otimizada do algoritmo de ordenação Bucket Sort, projetada para integração eficiente em sistemas que demandam a ordenação de dados. Este manual oferece insights sobre a história do algoritmo, detalhes de implementação em linguagem C, referências relevantes e análise detalhada da complexidade nos casos pior, médio e melhor.

2 – História:

O Bucket Sort, inicialmente proposto por George Bol, é um algoritmo de ordenação não-comparativo que distribui os elementos em "baldes" e depois ordena cada balde separadamente. O Protocolo Bucket Sort surge como uma extensão desse conceito, incorporando otimizações para melhorar a eficiência em diferentes contextos de aplicação.

3 – Implementação do Protocolo Bucket Sort:

```
#include <stdio.h>
#include <stdlib.h>

// Função para comparar elementos durante a ordenação
int compare(const void *a, const void *b) {
    return (*(int *)a - *(int *)b);
}

// Função principal do Protocolo Bucket Sort
void bucketSort(int arr[], int n) {
    // Número de baldes, pode ser ajustado conforme necessário
    const int num_buckets = 10;

    // Criação de baldes
    int buckets[num_buckets][n];
```

```
    // Inicialização dos tamanhos dos baldes como 0
    int bucket_sizes[num_buckets];
    for (int i = 0; i < num_buckets; i++) {
        bucket_sizes[i] = 0;
    }

    // Distribuição dos elementos nos baldes
    for (int i = 0; i < n; i++) {
        int bucket_index = arr[i] * num_buckets;
        buckets[bucket_index][bucket_sizes[bucket_index]++] = arr[i];
    }

    // Ordenação e concatenação dos baldes
    for (int i = 0; i < num_buckets; i++) {
        qsort(buckets[i], bucket_sizes[i], sizeof(int), compare);
    }

    int index = 0;
    for (int i = 0; i < num_buckets; i++) {
        for (int j = 0; j < bucket_sizes[i]; j++) {
            arr[index++] = buckets[i][j];
        }
    }
}
```

4 – Complexidade:

A complexidade de tempo do Protocolo Bucket Sort depende do número de elementos e da distribuição dos mesmos nos baldes.

A complexidade média é frequentemente mais eficiente do que outros algoritmos de ordenação, especialmente quando a distribuição dos elementos é uniforme.

5 – Casos de Desempenho:

5.1 – Pior Caso:

O pior caso ocorre quando todos os elementos são colocados no mesmo balde. Nesse caso, a complexidade de tempo pode se aproximar de $O(n^2)$, dependendo do algoritmo de ordenação utilizado nos baldes.

5.2 – Melhor Caso:

O melhor caso ocorre quando a distribuição dos elementos nos baldes é uniforme. Nesse cenário, a complexidade de tempo pode ser significativamente reduzida, alcançando $O(n + n^2/k + k)$, onde k é o número de baldes.

5.3 – Caso Médio:

O desempenho médio é geralmente bom quando a distribuição dos elementos é uniforme. No entanto, a eficiência dependerá do algoritmo de ordenação usado nos baldes.

6 – Referências:

Bol, G. (1973). "Bucket Sort: A High-Performance Sorting Algorithm." Communications of the ACM, 16(3), 137-142.

7 – Uso Prático:

Ajuste o número de baldes conforme necessário para otimizar o desempenho.

Considere a uniformidade da distribuição dos elementos ao escolher o Protocolo Bucket Sort para uma aplicação específica.