EST - UEA
ESCOLA SUPERIOR DE TECNOLOGIA
UNIVERSIDADE DO ESTADO DO AMAZONAS

Bucket Sort

Manaus - AM
2019

Adria Albuquerque Castro
Felipe Fonseca Bremgartner
Raul Henrique Moraes de Oliveira

Bucket Sort

Trabalho apresentado no curso de Engenharia da Computação na Universidade do Estado do Amazonas; para a disciplina de Algoritmo de Estrutura de Dados 2, professor: Sergio Cleger Tamayo.

Manaus - AM
201

Introdução

O presente trabalho tem como principal objetivo apresentar a ferramenta de ordenação denominada Bucket Sort. Conjuntamente com esse conceito, apresenta-se ademais informações computacionais sobre este método, a exemplo de sua complexidade computacional, estabilidade e tipo de ordenação; bem como, suas vantagens e desvantagens, além de uma aplicação prática do Bucket Sort.

BUCKET SORT

Bucket Sort: onde usar ou como usar?

O bucket sort é um método de ordenação que busca facilitar a ordenação de elementos de um vetor grande, distribuindo em grupos menores, denominados buckets (baldes).

Bucket Sort:Tipo de Ordenação

Um tipo de ordenação pode classificar as ferramentas de ordenação em dois grupos, sendo eles: completo e parcial.

Um algoritmo ser classificado como tipo de ordenação completa, significa que eu só vou poder saber parte de sua ordenação apenas no final de sua operação. Quanto que no tipo de ordenação parcial, é possível saber os menores valores (ou os maiores), por exemplo, logo após o início das primeiras comparações.

No caso do Bucket Sort, o tipo de ordenação é completa, dependendo do algoritmo de ordenação secundário utilizado.

Bucket Sort: Forma de Ordenação

A forma como o Bucket Sort ordena seus elementos, é pelo método de distribuição. Ele distribui os elementos em baldes, e após isso é necessário a utilização de um outro método para que este venha ordenar os elementos inseridos em cada balde.

Bucket Sort: Complexidade

A complexidade no Bucket Sort varia de acordo com a quantidade de baldes, e de elementos do conjunto a ser ordenados.

Para o melhor caso, a complexidade do Bucket Sort seria de $2n$, sendo n, o numero de elementos da lista. Nesta circunstância, todos os elementos cairiam em baldes diferentes.

Para o pior caso, todos os elementos cairiam em um único balde, e neste caso, a complexidade do algoritmo seria equivalente a complexidade do algoritmo secundário utilizado para ordenar aquele determinado balde.

Em um caso médio, a complexidade do Bucket Sort é de $O(n + k)$. Donde, n é o numero de elementos e k é o número de baldes.

Bucket Sort: Estabilidade

Um método de ordenação é considerado estável se a ordem relativa dos itens iguais não se altera durante a ordenação. O Bucket Sort pode sim ser considerado um algoritmo estável, pois ele não realiza troca em valores iguais.

Bucket Sort: Aplicações

Bucket Sort: Vantagens e Desvantagens

Vantagens

- Complexidade O(n + k) *[n = número de elementos, k = alcance dos elementos]*
- Algoritmo Estável
- Otimizado, pois utiliza sempre melhores casos para o *Sort* secundário.
- Não necessita utilizar comparações no melhor caso.

Desvantagens

- Não pode fazer ordenação parcial
- Implementação trabalhosa, pois exige a implementação de outro algorítmo para a segunda parte do Sort.

Implementação

Para a implementação do Bucket Sort segue-se os seguintes, passos:

1. Inicialize um vetor de "baldes", inicialmente vazios.
2. Vá para o vetor original, incluindo cada elemento em um balde.
3. Ordene todos os baldes não vazios.
4. Coloque os elementos dos baldes que não estão vazios no vetor original.

O código, cujo qual apresentamos:

```
void bucket_sort(int v[],int tam){

    int i,j,k,l;
    int maior = 0, proximo = 0;
    double maximo = 1;
    for(i=0;i<tam; i++){  //achar o maior valor, para ver o intervalo dos valores
        proximo = v[i];
        if(maior<proximo){
            maior=proximo;
        }
    }
    j=-1;
    for(i=1; i<1000000; i *= 10){  //achar o maior intervalo
        if((maior/i) != 0 ){
            j++;
        }
        if((maior/i) == 0 ){
            k=j;
            break;
        }
    }
    int bancas[10][tam];
    int aux[10];
    int num;
    int w;
    maximo=1;
    for(l=0;l<=k;l++){
        for(i=0;i<10;i++){          //inicializar matriz dos baldes
            for(j=0; j<tam;j++){
```

```
    int w;
    maximo=1;
    for(l=0;l<=k;l++){
        for(i=0;i<10;i++){           //inicializar matriz dos baldes
            for(j=0; j<tam;j++){
                bancas[i][j]=-1;
            }
        }
        for(i=0;i<10;i++){ //inicializar vetor auxiliar
            aux[i]=0;
        }
        j=0;
        for(i=0; i<tam; i++){  //colocar nos baldes
            num =(v[i]/maximo);
            num = num%10;
            bancas[num][aux[num]]=v[i];
            aux[num]++;
        }
        w=0;
        for(i=0;i<10;i++){         //juntar em 1 vetor só
            for(j=0;j<tam;j++){
                if(bancas[i][j] != -1){
                    v[w] = bancas[i][j];
                    w++;
                }
            }
        }
    maximo= maximo*10;

    }
}
```

Referências Bibliográficas

https://en.wikipedia.org/wiki/Bucket_sort
https://www.geeksforgeeks.org/bucket-sort-2/
https://www.ime.usp.br/~cris/aulas/13_1_338/slides/aula11.pdf
https://pt.wikipedia.org/wiki/Bucket_sort
http://wurthmann.blogspot.com/2015/06/bucket-sort.html