Algoritmo Jump Search

O algoritmo Jump Search é um método de busca em arrays ou listas ordenadas que combina as vantagens da busca linear e da busca binária. Ele funciona pulando uma quantidade fixa de elementos a cada iteração, em vez de dividir a lista pela metade como na busca binária. Isso torna o Jump Search mais eficiente do que a busca linear em termos de tempo de execução, mas menos eficiente do que a busca binária.

Durante a execução do algoritmo, a cada iteração, um salto é realizado em uma quantidade fixa de elementos até que se encontre um elemento que seja maior ou igual ao elemento a ser encontrado. Em seguida, a busca linear é realizada dentro de um subintervalo limitado pelo salto anterior e o salto atual para encontrar o elemento exato.

Embora o Jump Search não seja o algoritmo mais rápido para busca em arrays ou listas ordenadas, ele ainda é uma ótima alternativa para algumas situações em que a busca binária não é viável, como em arrays com baixa densidade de elementos.

Em conclusão, o algoritmo Jump Search é uma técnica eficiente de busca em arrays ou listas ordenadas que combina as vantagens da busca linear e da busca binária. Sua implementação pode ser útil em certas situações em que a busca binária não é a melhor opção.

Exemplo animado: no gif a seguir o algoritmo esta buscando o número 34 num vetor com 9 inteiros.

Referências:

- GeeksforGeeks. Jump Search. Disponível em: https://www.geeksforgeeks.org/jump-search/. Acesso em: 29 mar. 2023.
- Khan Academy. Jump search. Disponível em: https://www.khanacademy.org/computing/computer-science/algorithms/binary-search/a/jump-search. Acesso em: 29 mar. 2023.

Complexidade:

```c
Aluno *JumpSearch(Aluno *alunos, int qnta)
{
    char nome[100]; //c1, 1 vez
    int salto = sqrt(qnta); //c2, 1 vez
    int i = 0, j = 0; //c3, 1 vez
    while (i < qnta && strcmp(alunos[i].nome, nome) < 0)//c4, √qnta vezes
    {
        i += salto;//c5, √qnta vezes
    }
    if (i >= qnta || strcmp(alunos[i].nome, nome) > 0) //c6, 1 vez
    {
        i -= salto; //c7, 1 vez
        for (j = i; j < i + salto && j < qnta; j++)//c8, √qnta vezes
        {
            if (strcmp(alunos[j].nome, nome) == 0)//c9, √qnta vezes
            {
                return &alunos[j]; //c10, 1 vez
            }
        }
        return NULL;
    }
    else if (strcmp(alunos[i].nome, nome) == 0)//c11, 1 vez
    {
        return &alunos[i];
    }
    return NULL;
}

/*
quantidade de alunos = n

T(n) = (c1 + c2 + c3 + c6 + c7 + c10 + c11) + (c4 + c5 + c8 + c9)√n
a = (c4 + c5 + c8 + c9)        b = (c1 + c2 + c3 + c6 + c7 + c10 + c11)
T(n) = a√n + b

Ou seja

O(√n)
*/
```