Algortimos de Busquedas

Por Ariel Parra.

Busqueda Lineal

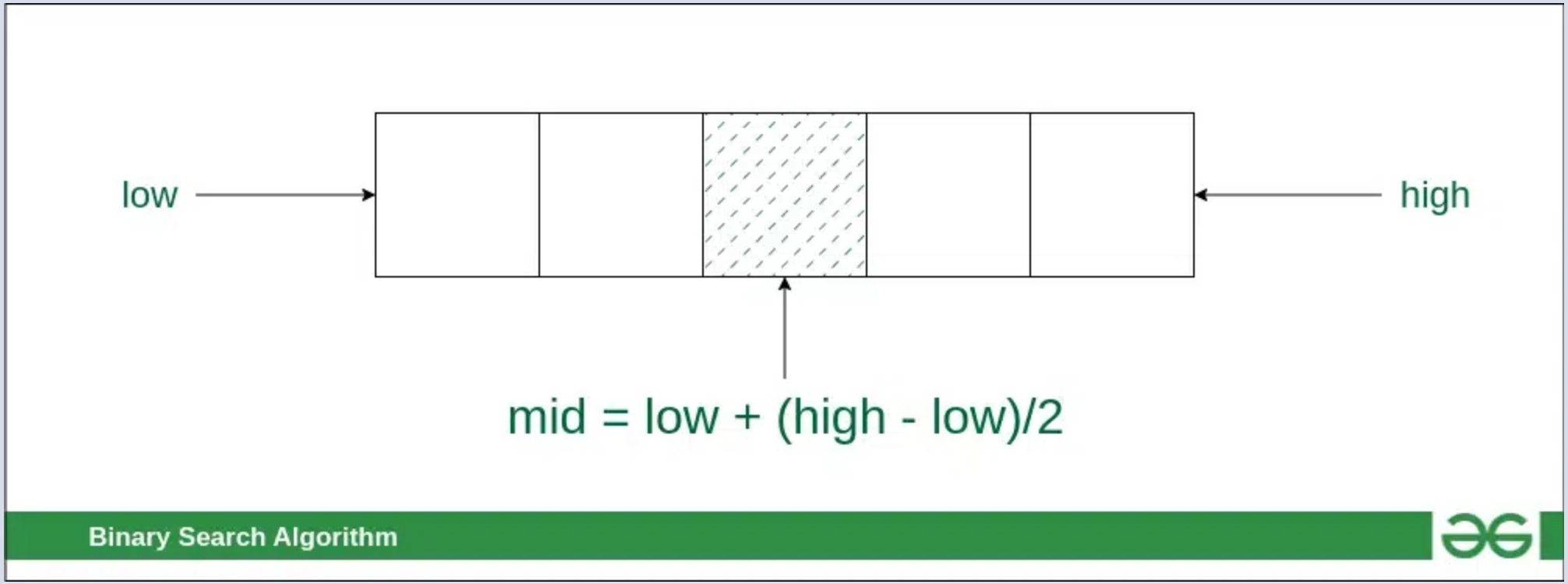
Supongamos que el artículo está en un array en orden aleatorio y tenemos que encontrar un elemento. Entonces la única manera de buscar un elemento objetivo es, para empezar, la primera posición y compararlo con el objetivo. Si el artículo está en el mismo, devolveremos la posición del artículo actual. De lo contrario, nos moveremos a la siguiente posición. Si llegamos a la última posición de un array y todavía no podemos encontrar el objetivo, regresamos -1 de que no se encontro.

```
int search(int array[], int n, int x){
        for (int i = 0; i < n; i++)
                if (array[i] == x)
                        return i;
        return -1;
}
```

```
int search(const std::vector<int>& vec, int x) {
    auto it = std::find(vec.begin(), vec.end(), x);
    if (it != vec.end()) {
        return std::distance(vec.begin(), it);
    }
    return -1;
}
```

Busqueda binaria

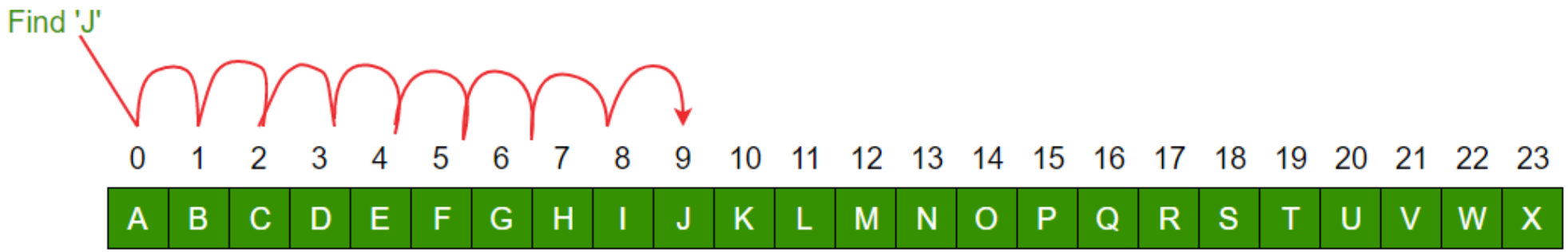
La búsqueda binaria se define como un algoritmo de búsqueda usado en una matriz ordenada dividiendo repetidamente el intervalo de búsqueda en la mitad. La idea de búsqueda binaria es utilizar la información que el array está ordenados.

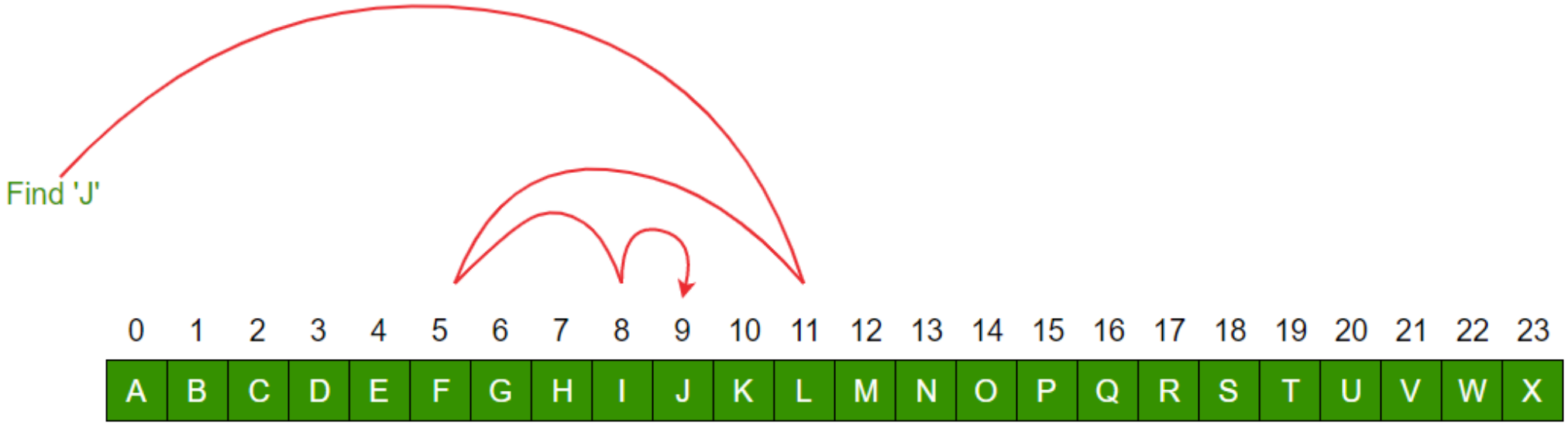


El algoritmo:

- Divide el espacio de búsqueda en dos mitades encontrando el índice medio “mid”.
- Encontrar el índice medio "mid" en el algoritmo de búsqueda binaria
- Encontrar el índice medio “mid” en el algoritmo de búsqueda binaria
- Compare el elemento medio del espacio de búsqueda con la clave.
- Si la clave se encuentra en el elemento medio, el proceso se termina.
- Si la clave no se encuentra en el elemento medio, elija la mitad se utilizará como el siguiente espacio de búsqueda.
- Si la clave es más pequeña que el elemento medio, entonces el lado izquierdo se utiliza para la siguiente búsqueda.
- Si la clave es más grande que el elemento medio, entonces el lado derecho se utiliza para la siguiente búsqueda.
- Este proceso continúa hasta que se encuentre la clave o se agote el espacio total de búsqueda.

Búsqueda lineal	Búsqueda binaria
Los datos de entrada de búsqueda lineal no deben estar en orden	En la búsqueda binaria los datos de entrada deben estar en orden ordenado.
La complejidad del tiempo de búsqueda lineal O(n).	La complejidad del tiempo de búsqueda binaria O(log n).
La matriz multidimensional se puede utilizar.	Sólo se utiliza una matriz dimensional única.
Es menos complejo.	Es más complejo.
Es muy lento proceso.	Es muy rápido.


Find 'J'
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
A B C D E F G H I J K L M N O P Q R S T U V W X


Find 'J'
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
A B C D E F G H I J K L M N O P Q R S T U V W X

Implementacion Iterativa

```
int binarySearch(int array[], int x, int low, int high){
        while (low <= high) {
                int mid = low + (high - low) / 2;
                if (array[mid] == x)
                        return mid;
                if (array[mid] < x)
                        low = mid + 1;
                else
                        high = mid - 1;
        }
        return -1;
}
```

```
int binarySearch(std::vector<int>& vec,int x){
    int low = 0;
    int high = vec.size() - 1;

    while (low <= high) {
        int mid = low + (high - low) / 2;
        if (vec[mid] == x)
            return mid;
        if (vec[mid] < x)
            low = mid + 1;
        else
            high = mid - 1;
    }
return -1;
}
```

Implementacion Recursiva

```
int binarySearch(int array[],int x,int low,int high){
    if (high >= low) {
        int mid = low + (high - low) / 2;
        if (array[mid] == x)
            return mid;
        if (array[mid] > x)
            return binarySearch(array, x, low, mid - 1);
        return binarySearch(array, x, mid + 1, high);
    }
return -1;
}
```

```
int binarySearch(std::vector<int>& vec,int x,int low,int high){
    if (high >= low) {
        int mid = low + (high - low) / 2;
        if (vec[mid] == x)
            return mid;
        if (vec[mid] > x)
            return binarySearch(vec, x, low, mid - 1);
        return binarySearch(vec, x, mid + 1, high);
    }
return -1;
}
```

Aplicaciones de la búsqueda binaria

La búsqueda binaria se puede utilizar como un bloque de construcción para algoritmos más complejos utilizados en el aprendizaje automático, como algoritmos para entrenar redes neuronales o encontrar los hiperparametros óptimos para un modelo.

Se puede utilizar para buscar en gráficos informáticos como algoritmos para rastreo de rayos o cartografía de textura.

Se puede utilizar para buscar una base de datos.

Problema

448D: Multiplication Table

Referencias

https://www.geeksforgeeks.org/searching-algorithms/
https://www.geeksforgeeks.org/binary-search/
https://en.wikipedia.org/wiki/Binary_search_algorithm
https://www.geeksforgeeks.org/linear-search-vs-binary-search/
https://codeforces.com/blog/entry/96699