**Binary search**

تعد خوارزمية البحث الثنائي من أكثر خوارزميات البحث شيوعا وتشترط هذه الخوارزمية أن تكون عناصر المصفوفة مرتبة ترتيبا تصاعديا أو تنازليا

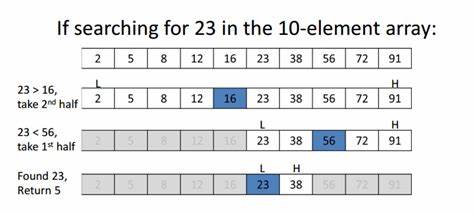
مبدأ العمل :

تبدأ الخوارزمية بمقارنة العنصر الموجود في منتصف المصفوفة مع العنصر الذي نريد البحث عنه اذا كان هو العنصر المطلوب تكون عملية البحث قد انتهت.

أما إذا كان العنصر المراد البحث عنه أكبر من العنصر الموجود في منتصف المصفوفة فإن العنصر الهدف يقع في النصف الأيمن من المصفوفة لذلك نتخلص من النصف الأيسر الذي لا يحتوي على الهدف الذي نبحث عنه وتتكرر عملية البحث في منتصف الجزء الأيمن الذي نتوقع أن تكون القيمة المطلوبة ضمنه

وأما إذا كان العنصر المراد البحث عنه أصغر من العنصر الموجود في منتصف المصفوفة فإن العنصر الهدف يقع في النصف الأيسر من المصفوفة لذلك نتخلص من النصف الأيمن الذي لا يحتوي على الهدف الذي نبحث عنه وتتكرر عملية البحث في منتصف الجزء الأيسر الذي نتوقع أن تكون القيمة المطلوبة ضمنه.

إذا كانت نتيجة البحث أن النصف المتبقي فارغ من العناصر فهذا يعني أن القيمة غير موجودة في المصفوفة



تعقيد الخوارزمية :

يكون التعقيد الزمني للخوارزمية في الحالة الأسوء o(logn)

أما في الحالة الأفضل عندما يكون العنصر الذي نبحث عنه في منتصف المصفوفة يكون التعقيد o(1)

تطبيق عملي :

لدينا مصفوفة حجمها N تتكون من أعداد صحيحة ويتم إعطاءك عنصر K نحتاج لتابع للبحث عن هذا العنصر في المصفوفة .

// C++ program to implement recursive Binary Search

#include <bits/stdc++.h>

using namespace std;

// A recursive binary search function. It returns

// location of x in given array arr[l..r] is present,

// otherwise -1

int binarySearch(int arr[], int l, int r, int x)

{

    if (r >= l) {

        int mid = l + (r - l) / 2;

        // If the element is present at the middle

        // itself

        if (arr[mid] == x)

            return mid;

        // If element is smaller than mid, then

        // it can only be present in left subarray

        if (arr[mid] > x)

            return binarySearch(arr, l, mid - 1, x);

        // Else the element can only be present

        // in right subarray

        return binarySearch(arr, mid + 1, r, x);

    }

    // We reach here when element is not

    // present in array

    return -1;

}

int main(void)

{

    int arr[] = { 2, 3, 4, 10, 40 };

    int x = 10;

    int n = sizeof(arr) / sizeof(arr[0]);

    int result = binarySearch(arr, 0, n - 1, x);

    (result == -1) ? cout << "Element is not present in array"

                   : cout << "Element is present at index " << result;

    return 0;

}