

THUẬT TOÁN TÌM KIẾM

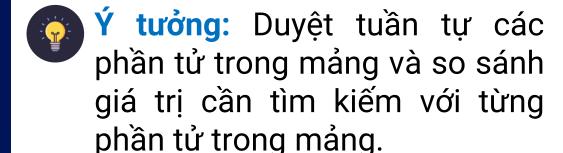


NỘI DUNG

- /01 Tìm kiếm tuyến tính (Linear search)
- /02 Tìm kiếm nhị phân (Binary search)
- /03 Vị trí đầu tiên trong mảng tăng dần
- /04 Vị trí cuối cùng trong mảng tăng dần
- /05 Vị trí đầu tiên lớn hơn hoặc bằng X trong mảng tăng dần
- /06 Vị trí cuối cùng lớn hơn hoặc bằng X trong mảng tăng dần



1.Tìm kiếm tuyến tính (Linear Search):

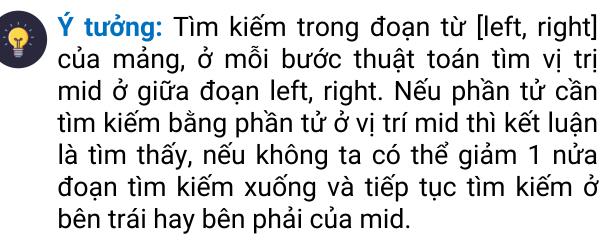


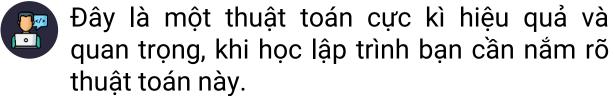


Các bài toán như tìm kiếm vị trí đầu tiên, cuối cùng, đếm số lần xuất hiện của phần tử trong mảng đều là biến đổi của thuật toán tìm kiếm tuyến tính.

```
code
int linearSearch(int a[], int n, int x){
  for(int i = 0; i < n; i++){
    if (x == a[i]){
      return 1;
    }
}
return 0;
}</pre>
```

2.Tìm kiếm nhị phân (Binary Search):





Diều kiện áp dụng: Mảng đã được sắp xếp.

Code int binarySearch(int a[], int n, int x){ int left = 0, right = n - 1; while(left <= right){</pre> int mid = (left + right) / 2; $if(a[mid] == x){$ return 1; else if(a[mid] < x){</pre> // Tìm kiếm ở bên phải left = mid + 1;else{ //Tìm kiếm ở bên trái right = mid - 1; Độ phức tạp: O(logN) return 0;



3. Vị trí đầu tiên trong mảng tăng dần:

Bài toán: Tìm vị trí đầu tiên của phần tử X trong mảng đã được sắp xếp

```
int firstPos(int a[], int n, int x){
    int res = -1, left = 0, right = n - 1;
    while(left <= right){</pre>
        int mid = (left + right) / 2;
        if(a[mid] == x){
            res = mid; // cập nhật
            //Tìm thêm đáp án tốt hơn
            right = mid - 1;
        else if(a[mid] < x){</pre>
            left = mid + 1;
        else{
            right = mid - 1;
                           Độ phức tạp: O(logN)
    return res;
```



4. Vị trí cuối cùng trong mảng tăng dần:

Bài toán: Tìm vị trí cuối cùng của phần tử X trong mảng đã được sắp xếp

```
int lastPos(int a[], int n, int x){
    int res = -1, left = 0, right = n - 1;
    while(left <= right){</pre>
        int mid = (left + right) / 2;
        if(a[mid] == x){
            res = mid; // cập nhật
            //Tìm thêm đáp án tốt hơn
            left = mid + 1;
        else if(a[mid] < x){</pre>
            left = mid + 1;
        else{
            right = mid - 1;
                           Độ phức tạp: O(logN)
    return res;
```



5. Vị trí đầu tiên lớn hơn hoặc bằng X trong mảng tăng dần:

Bài toán: Tìm vị trí đầu tiên của phần tử lớn hơn hoặc bằng X trong mảng đã được sắp xếp

```
int lower(int a[], int n, int x){
    int res = -1;
    int left = 0, right = n - 1;
    while(left <= right){</pre>
        int mid = (left + right) / 2;
        if(a[mid] >= x){
            res = mid; // cập nhật
            //Tìm thêm đáp án tốt hơn
            right = mid - 1;
        else{
            left = mid + 1;
                        Độ phức tạp: O(logN)
    return res;
```



6. Vị trí cuối cùng nhỏ hơn hoặc bằng X trong mảng tăng dần:

Bài toán: Tìm vị trí cuối cùng của phần tử nhỏ hơn hoặc bằng X trong mảng đã được sắp xếp

```
int upper(int a[], int n, int x){
    int res = -1;
    int left = 0, right = n - 1;
    while(left <= right){</pre>
        int mid = (left + right) / 2;
        if(a[mid] <= x){
            res = mid; // cập nhật
            //Tìm thêm đáp án tốt hơn
             left = mid + 1;
        else{
             right = mid - \overline{1};
                         Độ phức tạp: O(logN)
    return res;
```