

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):



Ý tưởng: Duyệt tuần tự các phần tử trong mảng và so sánh giá trị cần tìm kiếm với từng phần tử trong mảng.



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;  
}
```

Độ phức tạp: $O(N)$ ●



2. Tìm kiếm nhị phân (Binary Search):



Ý tưởng: Tìm kiếm trong đoạn từ [left, right] của mảng, ở mỗi bước thuật toán tìm vị trí mid ở giữa đoạn left, right. Nếu phần tử cần tìm kiếm bằng phần tử ở vị trí mid thì kết luận là tìm thấy, nếu không ta có thể giảm 1 nửa đoạn tìm kiếm xuống và tiếp tục tìm kiếm ở bên trái hay bên phải của mid.



Đây là một thuật toán cực kì hiệu quả và quan trọng, khi học lập trình bạn cần nắm rõ thuật toán này.



Điều 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){  
        int mid = (left + right) / 2;  
        if(a[mid] == x){  
            return 1;  
        }  
        else if(a[mid] < x){  
            // Tìm kiếm ở bên phải  
            left = mid + 1;  
        }  
        else{  
            //Tìm kiếm ở bên trái  
            right = mid - 1;  
        }  
    }  
    return 0;  
}
```

Độ phức tạp: $O(\log N)$



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){
        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){
            left = mid + 1;
        }
        else{
            right = mid - 1;
        }
    }
    return res;
}
```

Độ phức tạp: $O(\log N)$ ●

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){
        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){
            left = mid + 1;
        }
        else{
            right = mid - 1;
        }
    }
    return res;
}
```

Độ phức tạp: $O(\log N)$ ●

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){
        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;
        }
    }
    return res;
}
```

Độ phức tạp: $O(\log N)$ ●

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){
        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 - 1;
        }
    }
    return res;
}
```

Độ phức tạp: $O(\log N)$ ●