

# THUẬT TOÁN TÌM KIẾM

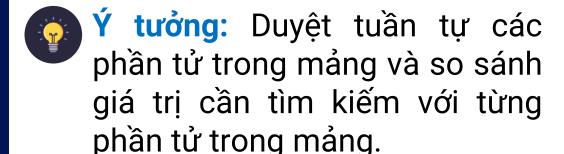


# **NỘI DUNG**

- /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
- /07 LOWER\_BOUND /08 UPPER\_BOUND



#### 1.Tìm kiếm tuyến tính (Linear Search):

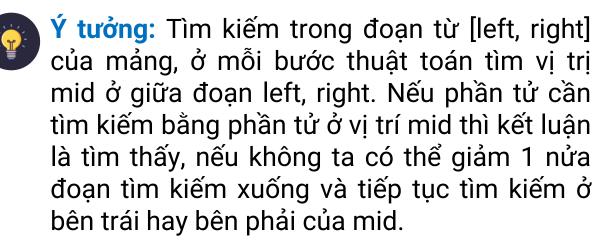


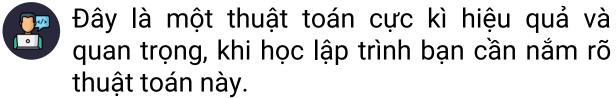


Các bài toán như tìm kiếm vị trí đầu tiên, cuối cùng, đếm số lần xuất hiện của phần tử trong mảng đều là biến đổi của thuật toán tìm kiếm tuyến tính.

```
Code
bool linearSearch(int a[], int n, int x){
  for(int i = 0; i < n; i++){
    if (x == a[i]){
      return true;
    }
}
return false;
}</pre>
Dô phức tạp: O(N)
```

#### 2.Tìm kiếm nhị phân (Binary Search):





Diều kiện áp dụng: Mảng đã được sắp xếp.

```
Code
bool 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 true;
        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 false;
```



#### 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;
```



#### 7. LOWER\_BOUND:

## CÚ PHÁP

lower\_bound(first\_iter, last\_iter, X);

## SỬ DỤNG LOWER\_BOUND



Lower\_bound là một hàm tương tự như mục 5 ở trên, nó có thể áp dụng cho mảng, vector, set, map.

Điều kiện áp dụng: Mảng, vector đã được sắp xếp tăng dần



Tương tự như hàm sort, lower\_bound cũng trả về iterator chứ không trả về giá trị.

Nó trả về vị trí đầu tiên của phần tử lớn hơn hoặc bằng X, nếu trong mảng, vector bạn tìm kiếm không có phần tử lớn hơn hoặc bằng X (tất cả đều nhỏ hơn X) thì lower\_bound trả về last\_iter

## Ví dụ lower\_bound với mảng:

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int a[7] = \{1, 2, 3, 3, 3, 4, 6\};
    auto it = lower_bound(a, a + 7, 3);
    cout << *it << endl;</pre>
    cout << (it - a) << endl;</pre>
            OUTPUT: 3
```

**Giải thích:** Phần tử đầu tiên lớn hơn hoặc bằng 3 trong mảng là 3 ở chỉ số 2

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int a[7] = {1, 2, 3, 3, 4, 6};
    auto it = lower_bound(a, a + 7, 8);
    if(it == a + 7){
        cout << "NOT FOUND";</pre>
    else{
        cout << *it << endl;</pre>
    cout << (it - a) << endl;</pre>
        OUTPUT: NOT FOUND
```

#### Ví du lower\_bound với vector:

```
#include <bits/stdc++.h>
using namespace std;
int main(){
   vector<int> a = {1, 2, 3, 3, 3, 4, 6};
   auto it = lower_bound(a.begin(), a.end(), 3);
   if(it == a.end()){
       cout << "NOT FOUND\n";</pre>
   else{
       cout << *it << endl;</pre>
   cout << (it - a.begin()) << endl;</pre>
                 OUTPUT: 3
```

```
#include <bits/stdc++.h>
using namespace std;
int main(){
   vector<int> a = {1, 2, 3, 3, 3, 4, 6};
   auto it = lower_bound(a.begin(), a.end(), 8);
   if(it == a.end()){
       cout << "NOT FOUND\n";</pre>
   else{
       cout << *it << endl;</pre>
   cout << (it - a.begin()) << endl;</pre>
           OUTPUT: NOT FOUND
```

#### Quiz 1

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    vector<int> a = {1, 1, 3, 3, 3, 4, 6};
    auto it = lower_bound(a.begin(), a.end(), 2);
    --it;
    cout << *it << endl;
}</pre>
```

```
#include <bits/stdc++.h>
using namespace std;

int main(){
   vector<int> a = {1, 1, 3, 3, 3, 4, 6};
   auto it = lower_bound(a.begin(), a.end(), 7);
   --it;
   cout << *it << endl;
}</pre>
```

#### Quiz 3

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    vector<int> a = {1, 1, 3, 3, 3, 4, 6};
    auto it = lower_bound(a.begin(), a.end(), 7);
    it -= 2;
    cout << *it << endl;
}</pre>
```

```
#include <bits/stdc++.h>
using namespace std;

int main(){
   vector<int> a = {1, 1, 3, 3, 3, 4, 6};
   auto it = lower_bound(a.begin(), a.end(), 0);
   it--;
   cout << *it << endl;
}</pre>
```



#### 8. UPPER\_BOUND:

## CÚ PHÁP

upper\_bound(first\_iter, last\_iter, X);

## SỬ DỤNG UPPER\_BOUND



Điều kiện áp dụng: Mảng, vector đã được sắp xếp tăng dần



Upper\_bound trả về vị trí đầu tiên của phần tử lớn X, nếu trong mảng, vector bạn tìm kiếm không có phần tử lớn hơn hoặc bằng X (tất cả đều nhỏ hơn hoặc bằng X) thì upper\_bound trả về last\_iter.

## Ví dụ upper\_bound với mảng:

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int a[7] = \{1, 2, 3, 3, 3, 4, 6\};
    auto it = upper_bound(a, a + 7, 3);
    cout << *it << endl;</pre>
    cout << (it - a) << endl;</pre>
          OUTPUT: 4
```

Giải thích: Phần tử đầu tiên lớn hơn 3 trong mảng là 4 ở chỉ số 5

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int a[7] = {1, 2, 3, 3, 4, 6};
    auto it = upper_bound(a, a + 7, 8);
    if(it == a + 7){
        cout << "NOT FOUND";</pre>
    else{
        cout << *it << endl;</pre>
    cout << (it - a) << endl;</pre>
        OUTPUT: NOT FOUND
```

## Ví dụ upper\_bound với vector:

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    vector<int> a = {1, 2, 3, 3, 4, 6};
    auto it = upper_bound(a.begin(), a.end(), 3);
    if(it == a.end()){
        cout << "NOT FOUND\n";</pre>
    else{
        cout << *it << endl;</pre>
    cout << (it - a.begin()) << endl;</pre>
              OUTPUT: 4
```

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    vector<int> a = {1, 2, 3, 3, 3, 4, 6};
    auto it = upper_bound(a.begin(), a.end(), 6);
    if(it == a.end()){
        cout << "NOT FOUND\n";</pre>
    else{
        cout << *it << endl;</pre>
    cout << (it - a.begin()) << endl;</pre>
            OUTPUT: NOT FOUND
```

#### Quiz 1

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    vector<int> a = {1, 1, 3, 3, 3, 4, 6};
    auto it = upper_bound(a.begin(), a.end(), 2);
    --it;
    cout << *it << endl;
}</pre>
```

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    vector<int> a = {1, 1, 3, 3, 3, 4, 6};
    auto it = upper_bound(a.begin(), a.end(), 6);
    --it;
    cout << *it << endl;
}</pre>
```

#### Quiz 3

```
#include <bits/stdc++.h>
using namespace std;
int main(){
   vector<int> a = {1, 1, 3, 3, 3, 4, 6};
   auto it = upper_bound(a.begin(), a.end(), 6);
   it -= 2;
   cout << *it << endl;</pre>
```

```
#include <bits/stdc++.h>
using namespace std;
int main(){
   vector\langle int \rangle a = {1, 1, 3, 3, 3, 4, 6};
   auto it = upper_bound(a.begin(), a.end(), 0);
   it--;
   cout << *it << endl;</pre>
```