

## 二分查找

### 二分查找模板

二分查找模板，适用于有序数组，返回目标值的索引，如果不存在则返回-1

### 二分查找模板

二分查找模板，返回目标值的索引，如果不存在则返回-1

- 二分查找1

```
while(head < tail)
{
    mid = (head + tail) / 2;
    if(arr[mid] < target) head = mid + 1;
    else tail = mid;
}
return head;
```

二分查找模板

- 二分查找1

```
while(head < tail)
{
    mid = (head + tail + 1) / 2; //防止溢出
    if(arr[mid] < target) head = mid;
    else tail = mid - 1;
}
return head;
```

## 二分查找

二分查找模板，返回目标值的索引，如果不存在则返回-1

- 二分查找模板x二分查找模板x二分查找模板x二分查找模板
- 二分查找模板x二分查找模板x二分查找模板x二分查找模板

## 二分查找模板

### 二分查找

- 二分查找O(1)
- 二分查找模板x二分查找模板x二分查找模板x二分查找模板
- 二分查找模板x二分查找模板
- 二分查找模板x二分查找模板
  - 二分查找模板x二分查找模板

- 時間複雜度
- 空間複雜度
- 代碼實現
- 測試用例

測試用例

- 測試用例
- 測試用例1
- 測試用例2
- 測試用例3
- 測試用例4

測試用例

測試用例

 alt text

- unordered\_map
- unordered\_map

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        unordered_map<int,int> h;
        vector<int> ret(2);
        for(int i = 0;i < nums.size();i++)
        {
            if(h.find(target - nums[i]) != h.end())
            {
                ret[0] = h[target - nums[i]];
                ret[1] = i;
                break;
            }
            h[nums[i]] = i;
        }
        return ret;
    }
};
```

- 測試用例
- 測試用例
- 測試用例


```
class Solution {
public:
    int find(vector<int> &nums,vector<int> &ind,int target,int i)
    {
        int head = i,tail = nums.size() - 1,mid;
```

```

        while(head <= tail)
        {
            mid = (tail + head) / 2;
            if(nums[ind[mid]] == target)
                return mid;
            if(nums[ind[mid]] < target)
                head = mid + 1;
            else
                tail = mid - 1;
        }
        return 0;
    }
    vector<int> twoSum(vector<int>& nums, int target) {
        vector<int> ret(2);
        vector<int> ind(nums.size());
        for(int i = 0; i < nums.size(); i++) ind[i] = i;
        sort(ind.begin(), ind.end(), [&](int i, int j) -> bool
        {
            return nums[i] < nums[j];
        });
        for(int i = 0; i < nums.size(); i++)
        {
            if(find(nums, ind, target - nums[ind[i]], i + 1))
            {
                ret[0] = ind[i];
                ret[1] = ind[find(nums, ind, target - nums[ind[i]], i + 1)];
                break;
            }
        }
        return ret;
    }
};

```

□□□□□□

 alt text


- □□□□□□□□□□□□□□□□

```

class Solution {
public:
    int searchInsert(vector<int>& nums, int target) {
        int head = 0, tail = nums.size() - 1, mid;
        while(head <= tail)
        {
            mid = (head + tail) / 2;
            if(nums[mid] == target) return mid;
            if(nums[mid] < target) head = mid + 1;
            else tail = mid - 1;
        }
        return head ; //□□□□□□□□target□□□□□□
    }
};

```

□□□□□□

 alt text


- 
- 

```
class Solution {
public:
    bool containsDuplicate(vector<int>& nums) {
        int head ,tail ,mid;
        vector<int> ind(nums.size());
        for(int i = 0;i < nums.size();i++) ind[i] = i;
        sort(ind.begin(),ind.end(),[&](int i ,int j)->bool{
            return nums[i] <nums[j];
        });
        for(int i = 0;i < nums.size();i++)
        {
            head = i + 1,tail = nums.size() - 1;
            while(head <= tail)
            {
                mid = (head + tail) / 2;
                if(nums[ind[mid]] == nums[ind[i]]) return true;
                if(nums[ind[mid]] < nums[ind[i]]) head = mid + 1;
                else tail = mid - 1;
            }
        }
        return false;
    }
};
```

- `0000`
- `0000000000000000true000000`

```
class Solution {
public:
    bool containsDuplicate(vector<int>& nums) {
        unordered_set<int> s;
        for(auto x:nums)
        {
            if(s.find(x) != s.end()) return true;
            s.insert(x);
        }
        return false;
    }
};
```

Intersection

 alt text

- Intersection
- `set` Intersection
- Intersection

```
class Solution {
public:
    vector<int> intersection(vector<int>& nums1, vector<int>& nums2) {
        vector<int> ret;
        unordered_set<int> h;
        for(auto x: nums1) h.insert(x);
        for(auto x: nums2)
        {
            if(h.find(x) == h.end()) continue;
            ret.push_back(x);
            h.erase(x);
        }
        return ret;
    }
};
```

Longest Substring

 alt text

Longest Substring

- `ans`
- `ans` `i` `j` (substring)
- `map` `i`
- Longest Substring


```
class Solution {
public:
    int lengthOfLongestSubstring(string s) {
        unordered_map<char,int> h;
        int ans = 0;
        for(int i = 0; i < s.size(); i++)
        {
            for(int j = i; j <= s.size(); j++)
            {
                if(j == s.size() || h.find(s[j]) != h.end())
                {
                    ans = max(ans, j - i);
                }
            }
        }
    }
};
```

```

        if(j == s.size()) return ans;
        i = max(i,h[s[j]] + 1);
    }
    h[s[j]] = j;
}
}
return ans;
}
};

```

□□□□+□□□□

- □□□□□□□□□□□□□□□□1  alt text
- □□ans□□□□□□□□□□□□k□□□□□□□□□□

```

class Solution {
public:
    bool check(string &s,int l)
    {
        int ans[256] = {0},k = 0;
        for(int i = 0;s[i]; i++)
        {
            ans[s[i]] += 1;
            if(ans[s[i]] == 1) k += 1;
            if(i >= l)
            {
                ans[s[i-l]] -= 1;
                if(ans[s[i - l]] == 0) k -=1 ;
            }
            if(k == l) return true;
        }
        return false;
    }

    int lengthOfLongestSubstring(string s) {
        int head = 0,tail = s.size(),mid;
        while(head < tail)
        {
            mid = (head + tail + 1) / 2;
            if(check(s,mid)) head = mid;
            else tail = mid - 1;
        }
        return head;
    }
};

```


□□□□□□□□□□□□

 alt text

- 0000000000000000k000
- 0000000000000000k / 20000000000k / 2000
- 0000000000000000k / 2000
- 000000

```
#include <inttypes>
class Solution {
public:
    int findk(vector<int>& n1,int ind1,vector<int>& n2,int ind2,int k)
    {
        int n = n1.size(),m = n2.size();
        if(k == 1)
        {
            int a = (ind1 == n)?INT32_MAX:n1[ind1];
            int b = (ind2 == m)?INT32_MAX:n2[ind2];
            return min(a,b);
        }
        if(n == ind1) return n2[k - 1];
        if(m == ind2) return n1[k - 1];
        int cnt1 = min(k / 2,n - ind1);
        int cnt2 = min(k - cnt1,m - ind2);
        cnt1 = k - cnt2;
        if(n1[cnt1 + ind1 - 1] <= n2[cnt2 + ind2 - 1])
            return findk(n1,ind1 + cnt1,n2,ind2,k - cnt1);
        else
            return findk(n1,ind1,n2,ind2 + cnt2,k - cnt2);
    }
    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
        int n = nums1.size(),m = nums2.size();
        if((n + m) % 2 == 1) return findk(nums1,0,nums2,0,(n + m) / 2 + 1);
        double a = findk(nums1,0,nums2,0,(n + m) / 2);
        double b = findk(nums1,0,nums2,0,(n + m) / 2 + 1);
        return (a + b) / 2.0;
    }
};
```

0000

 alt text

0000000000000000  alt text

- 00000000x00000000y0000000000!0000000000000000C

```
#include <bits/stdc++.h> //hzoj244 0000
using namespace std;
struct point
```

```

{
    int x,y;
}arr[505];

int temp[505];

bool cmp(const point &a,const point &b)
{
    if(a.x != b.x) return a.x < b.x;
    return a.y < b.y;
}

int check_y(point *arr,int n,int c,int b,int e,int l)
{
    int cnt = 0;
    for(int i = b;i <= e;i++) temp[cnt++] = arr[i].y;
    sort(temp,temp + cnt);
    for(int i = c - 1;i < cnt;i++)
    {
        if(temp[i] - temp[i - c + 1] < l) return 1;
    }
    return 0;
}

int check(point *arr,int n,int c,int l)//l□□□□□□
{
    int j = 0;
    for(int i = 0;i < n;i++)
    {
        while(arr[i].x - arr[j].x >= l) j += 1;
        if(i - j + 1 < c) continue;
        if(check_y(arr,n,c,j,i,l))
            return 1;
    }
    return 0;
}

int bs(int l,int r,point *arr,int n,int c)
{
    int mid = 0;
    while(l < r)
    {
        mid = (l + r) / 2;
        if(check(arr,n,c,mid)) r = mid;
        else l = mid + 1;
    }
    return l;
}

```



```

int main()
{
    int c,n;
    cin >> c >> n;
    for(int i = 0; i < n;i++)
        cin >> arr[i].x >> arr[i].y;
    sort(arr,arr + n,cmp);
    cout << bs(0,10000,arr,n,c) << endl;
    return 0;
}

```

**check\_y**□□□□□□□□

```

int check_y(point *arr,int n,int c,int b,int e,int l)
{
    int j = 0,cnt = 0;
    for(int i = b;i <= e;i++) temp[cnt++] = arr[i].y;
    sort(temp,temp + e - b + 1);
    for(int i = 0;i <= e - b;i++)
    {
        while(temp[i] - temp[j] >= l) j += 1;
        if(i - j + 1 >= c) return 1;
    }
    return 0;
}

```