

二分法

二分法の概要

二分法は、二つの要素を比較して、二つの要素の間に含まれる要素を特定するアルゴリズムです。二つの要素は、二つの要素の間に含まれる要素を特定するアルゴリズムです。

二分法の概要

二分法の概要  alt text

- 二分法1

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

 alt text

- 二分法1

```
• while(head < tail)
{
    mid = (head + tail + 1) / 2; // 1要素を考慮
    if(arr[mid] < target) head = mid;
    else tail = mid - 1;
}
return head;
```

二分法

二分法の概要  alt text

- 二分法の概要 x 二分法の概要 x 二分法の概要 x 二分法の概要
- 二分法の概要 x 二分法の概要 x 二分法の概要 x 二分法の概要

二分法の概要

二分法

- 二分法の概要 O(1)
- 二分法の概要 x 二分法の概要 x 二分法の概要 x 二分法の概要
- 二分法の概要 x 二分法の概要 x 二分法の概要 x 二分法の概要
- 二分法の概要 x 二分法の概要 x 二分法の概要 x 二分法の概要
 - 二分法の概要 x 二分法の概要 x 二分法の概要 x 二分法の概要

- 亂序
- 帶有重複元素的有序
- 帶有重複元素的無序
- 空

解法

- 暴力
- 帶有重複元素的有序
1. 双指针
- 帶有重複元素的無序
- 帶有重複元素的無序
2. 哈希表

帶有重複元素的無序

解法



- `unordered_map`
- 帶有重複元素的無序

```
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;  
    }  
};
```

- 哈希(帶重複)
- 帶重複
- 帶有重複元素的無序

```
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;
}
};


```

二分查找



- 二分查找

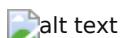
```

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 ; //二分查找target插入位置
    }
};

```

```
    }  
};
```

• 亂數



- 亂數
- 亂數

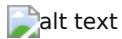
```
class Solution {  
public:  
    bool containsDuplicate(vector<int>& nums) {  
        int head ,tail ,mid;  
        vector<int> ind(nums.size());  
        for(int i = 0;i < nums.size();i++) ind[i] = i;  
        sort(ind.begin(),ind.end(),[&](int i ,int j)->bool{  
            return nums[i] 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

- 亂數

- 亂數

```
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;  
    }  
};
```

二重集合



- 二重vector
- 二重set
- 二重map

```
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;  
    }  
};
```

二重map



二重map

- ans
- ans[i][j] ans[i][j] (二重vector)
- map[i] map[i] (二重map)
- 二重map

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

```
        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;
    }
};

```

□□□□□□□□□□



- ဗိသုကာပညာနှင့်အမြတ်ဆင့် k ရှိခဲ့
 - ဗိသုကာပညာနှင့်အမြတ်ဆင့် k / 2 ရှိခဲ့ကြည့်ရန် / 2 ရှိခဲ့
 - ဗိသုကာပညာနှင့်အမြတ်ဆင့် k / 2 ရှိခဲ့
 - ဗိသုကာပညာ

```

#include <cinttypes>
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;
    }
};

```

1



   alt text

- $\text{C}_x \text{H}_y \text{O}_z$ | $\text{C}_x \text{H}_{y+z}$

```
#include <bits/stdc++.h> //hzoj244 二叉树  
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区间
{
    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;
}
```