**CHUYÊN ĐỀ MẢNG 1 CHIỀU**

Sắp xếp và tìm kiếm đóng vai trò rất quan trọng cuộc sống nói chung và tin học nói riêng. Đơn giản với một cuốn từ điển, nếu các từ không được sắp xếp thì việc tra một từ gặp khó khăn đến mức như thế nào? Việc tìm kiếm sẽ mất thời gian ra sao nếu không có một thuật toán tìm kiếm tối ưu khi gặp một bộ dữ liệu lớn?

Chuyên đề này sẽ giới thiệu một số thuật toán sắp xếp và tìm kiếm cơ bản và các bài tập áp dụng các thuật toán đó nhằm bước đầu tiếp cận và làm quen với các dạng bài tập về tìm kiếm và sắp xếp.

Đối tượng phạm vi nghiên cứu:Các tài liệu ôn thi học sinh giỏi, phần chất lỏng, các phép toán sơ cấp và cao cấp được áp dụng vào vật lí.

**I. Sắp xếp:**

## 1. Bài toán tổng quát:

Bài toán sắp xếp được phát biểu như sau: Cho mảng các đối tượng, cần sắp xếp lại các thành phần (phần tử) của mảng để nhận được mảng mới với các thành phần có các giá trị không tăng hoặc không giảm.

## 2. Thuật toán sắp xếp thông dụng:

### 2.1. Thuật toán sắp xếp chọn (Selection Sort)

-Ý tưởng: Lần lượt chọn phần tử nhỏ nhất trong toàn mảng đổi chỗ cho phần tử thứ nhất, chọn phần tử nhỏ nhất trong các phần tử còn lại (từ vị trí 2 đến cuối ) đổi chỗ cho phần tử thứ hai, … tiếp tục như vậy cho đến khi đổi chỗ xong cho phần tử thứ n-1

-Ví dụ: Cho mảng A[1..8] có 8 phần tử chưa sắp là:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 4 | 12 | 6 | 3 | 2 | 10 | 7 |

Phần tử nhỏ nhất trong các phần tử từ 1 đến 8 là A[6]=2:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 4 | 12 | 6 | 3 | 2 | 10 | 7 |

Đổi chỗ A[6] với A[1] được:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 4 | 12 | 6 | 3 | 8 | 10 | 7 |

Phần tử nhỏ nhất trong các phần tử từ 2 đến 8 là A[5]=3:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 4 | 12 | 6 | 3 | 8 | 10 | 7 |

Đổi chỗ A[5] với A[2] được:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 12 | 6 | 4 | 8 | 10 | 7 |

Phần tử nhỏ nhất trong các phần tử từ 3 đến 8 là A[5]=4:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 12 | 6 | 4 | 8 | 10 | 7 |

Đổi chỗ A[5] với A[3] được:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 4 | 6 | 12 | 8 | 10 | 7 |

Phần tử nhỏ nhất trong các phần tử từ 4 đến 8 là A[4]=6:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 4 | 6 | 12 | 8 | 10 | 7 |

Phần tử nhỏ nhất trong các phần tử từ 5 đến 8 là A[8]=7:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 4 | 6 | 12 | 8 | 10 | 7 |

Đổi chỗ A[5] với A[8] được:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 4 | 6 | 7 | 8 | 10 | 12 |

-Thuật toán được viết bằng C++ như sau:

void selection\_sort(float a[],int n)  
{  
    float tg;  
    for (int i=0;i<n;i++)  
        for(int j=0;j<i;j++)  
           {  
                if(a[i]<a[j])  
                {  
 swap(a[i],a[j]);  
           }  
}

-Đánh giá độ phức tạp:

### 2.2. Thuật toán sắp xếp nổi bọt (Bubble Sort)

-Ý tưởng: Tìm và đổi chỗ các cặp phần tử kề nhau sai thứ tự (phần tử đứng trước có khóa lớn hơn khóa của phần tử đứng sau) cho đến khi không tồn tại cặp nào sai thứ tự (dãy được sắp xếp).

-Các bước để sắp xếp một mảng tăng dần:

B1: So sánh phần tử thứ 0 và phần tử thứ 1. Nếu phần tử thứ 0 lớn hơn thì tráo đổi chúng cho nhau. Làm tương tự với phần tử thứ 1 và phần tử thứ 2 cho đến cuối dãy. Khi đó phần tử cuối cùng của mảng được sắp xếp.

B2: Lặp lại bước 1 cho đến khi mảng được sắp xếp

-Chúng ta xét qua ví dụ: Cho mảng A gồm n=6 phần tử là các số nguyên, cần sắp A không giảm.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | 3 | 5 | 8 | 6 | 2 |

So sánh A[1]=4 và A[2]=3 phải đổi chỗ cho nhau, được:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 4 | 5 | 8 | 6 | 2 |

Tiếp theo so sánh A[2]=4 và A[3]=5, thấy không cần tráo đổi

Tiếp theo so sánh A[3]=5 và A[4]=8, thấy không cần tráo đổi.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | 3 | 5 | 8 | 6 | 2 |

Tiếp theo so sánh A[4]=8 và A[5]=6 phải tráo đổi cho nhau được:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | 3 | 5 | 6 | 8 | 2 |

Tiếp theo so sánh A[5]=8 và A[6]=2 phải tráo đổi cho nhau được:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | 3 | 5 | 6 | 2 | 8 |

Vậy đã chuyển được số lớn nhất về cuối cùng của mảng là vị trí n .

Tiếp theo, tương tự tìm số lớn nhất trong các số từ 1 đến n-1 chuyển vào vị trí n-1, rồi tìm số lớn nhất trong các số từ 1 đến n-2 chuyển vào vị trí n-2, …Quá trình tiếp tục như vậy cho đến khi sắp xong vị trí 2.

-Thuật toán được viết bằng C++ như sau:

void bubbleSort(int arr[], int n)

{

   int i, j;

   for (i = 0; i < n-1; i++)

       for (j = 0; j < n-i-1; j++)

           if (a[j] > a[j+1])

              swap(a[j], a[j+1]);

}

-Đánh giá độ phức tạp:

### 2.3. Thuật toán sắp xếp nhanh (Quick Sort)

-Ý tưởng thuật toán:

Để sắp xếp dãy coi như là sắp xếp đoạn từ chỉ số 1 đến chỉ số . Để sắp xếp một đoạn trong dãy, nếu đoạn chỉ có một phần tử thì dãy đã được sắp xếp, ngược lại ta chọn một phần tử x trong đoạn đó làm “chốt”, mọi phần tử có khóa nhỏ hơn khóa của “chốt” được xếp vào vị trí đứng trước chốt, mọi phần tử có khóa lớn hơn khóa của “chốt” được xếp vào vị rí đứng sau chốt. Sau phép hoán chuyển như vậy thì đoạn đang xét được chia làm hai đoạn mà mọi phần tử trong đoạn đầu đều có khóa khóa của “chốt” và mọi phần tử trong đoạn sau đều có khóa khóa của “chốt”. Tiếp tục sắp xếp kiểu như vậy với 2 đoạn con, ta sẽ được đoạn đã cho được sắp xếp theo chiều tăng dần của khóa.

-Thuật toán được viết bằng ngôn ngữ C++ như sau:

void quickSort(int A[], int l, int r)

{

if(l>=r) return;

int i=l;

int j=r;

int x=A[(l+r)/2];

while(i<=j)

{

while(A[i]<x) i++;

while(A[j]>x) j--;

if(i<=j)

{

int temp=A[i];

A[i]=A[j];

A[j]=temp;

i++;

j--;

}

}

quickSort(A,l,j);

quickSort(A,i,r);

}

Ngoài ra, trong C++ ta có thể sử dụng trực tiếp hàm **qsort** có sẵn trong thư viện.

-Đánh giá độ phức tạp: trong trường hợp trung bình và với trường hợp xấu nhất.

*Chú ý:* Với chương trình ít gọi tới thủ tục sắp xếp và chỉ trên tập dữ liệu nhỏ, ta hoàn toàn có thể sử dụng một thuật toán sắp xếp đơn giản để dễ cài đặt. Tuy nhiên với các tập dữ liệu lớn, ta nên sử dụng thuật toán sắp xếp nhanh có độ phức tạp cỡ .

**2.4 Thuật toán sắp xếp dùng sort có sẵn có độ phức tạp O(nlogn)**

# **II. Tìm kiếm**

## 1. Tìm kiếm tuyến tính

-Ý tưởng: Tìm kiếm lần lượt từng phần tử trong mảng. Thích hợp trong các trường hợp với danh sách dữ liệu đủ nhỏ hay một danh sách chưa được sắp xếp thứ tự đơn giản.

-Các bước: xét dãy số cần tìm có n phần tử: a[0], a[1], a[2], ... , a[n-1]. Giá trị cần tìm là x.  
  + Bắt đầu từ khoá đầu tiên, lần lượt so sánh khoá x với khoá tương ứng trong dãy.  
  + Quá trình tìm kiếm kết thúc khi tìm được khoá thoả mãn hoặc đi đến hết dãy hoặc gặp điều kiện dừng vòng lặp.

-Thuật toán được viết bằng ngôn ngữ C++ như sau:

int  SequentialSearch (int a[], int n, int x)

{

For (int i = 0; i < n; i++)

if (a[i] == x) return 1;

return 0;

}

-Đánh giá độ phức tạp:

## 2. Tìm kiếm nhị phân

-Ý tưởng: Trong mỗi bước, so sánh phần tử cần tìm với phần tử nằm ở chính giữa danh sách. Nếu hai phần tử bằng nhau thì phép tìm kiếm thành công và thuật toán kết thúc. Nếu chúng không bằng nhau thì tùy vào phần tử nào lớn hơn, thuật toán lặp lại bước so sánh trên với nửa đầu hoặc nửa sau của danh sách.

-Các bước thực hiện: Cho N số trên mảng một chiều A[0..N-1] đã sắp tăng, ban đầu khởi trị đầu trái của khoảng chứa k là *l*=0; và đầu phải của khoảng chứa k là *r*=N-1. Trong khi *l<r* thực hiện vòng lặp: chọn m là điểm giữa của *l* và *r*, nếu A[m]<k thì giá trị *k* chỉ có thể trong đoạn [m+1,r] do đó gán lại giá trị *l=m+1*; nếu k≤A[m] thì giá trị *k* chỉ có thể nằm trong đoạn [*l, m*] do đó gán lại *r=m*;

Kết thúc vòng lặp nếu a[r]=k thì xuất ra vị trí r, ngược lại thì xuất ra thông báo “-1” (nghĩa là vô nghiệm).

- Thuật toán được viết bằng ngôn ngữ C++ như sau:

int binarySearch(int arr[], int l, int r, int x)

{

   if (r >= l)

   {

        int mid = l + (r - l)/2;

        if (arr[mid] == x)

            return mid;

        if (arr[mid] > x)

            return binarySearch(arr, l, mid-1, x);

        return binarySearch(arr, mid+1, r, x);

   }

   return -1;

}

*Đặc biệt.* Trong lớp vector, C++ xây dựng sẵn hàm tìm kiếm nhị phân (binary\_search(p1, p2, x));

Nếu x là một trong các phần tử của vector kể từ vị trí lặp *p1* đến vị trí lặp *p2* thì hàm này trả về giá trị true, ngược lại trả về giá trị false.

-Đánh giá độ phức tạp:

- Tìm kiếm bằng dùng hàm có sẵn :

lower\_bound(a+1,a+n+1,x)-a;

**upper\_bound(a+1,a+n+1,x)-a-1**

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  int n,x,a[1003];  int main() {  cin>>n>>x;  for(int i=1;i<=n;i++)  cin>>a[i];  int pos = upper\_bound(a+1, a+n+1, x)-a;  cout << "Con trỏ trả về: " << pos << endl;  // cout << "Giá trị tại đó: " << \*pos << endl;  } |

**Giống nhau giữa lower\_bound và upper\_bound**

✅ Cả hai đều trả về một **con trỏ**.  
✅ Để lấy chỉ số, ta cần **trừ a** (pointer - array\_base).  
✅ Thường hoạt động chính xác nếu **mảng đã sắp xếp**.

**2.5: Mảng cộng dồn**

Bài toán tìm tổng dãy con liên tiếp là một bài toán cơ bản có rất nhiều biến thể trong các bài toán tin mức độ trung bình đến tương đối khó, phù hợp với đối tượng học sinh tiền đội tuyển lớp 10 chuyên. Đây là chủ đề vận dụng kiến thức nâng cao sau khi học mảng một chiều, mảng hai chiều, quy hoạch động cơ bản.

Phát biểu bài toán cơ sở: cho mảng A gồm N phần tử. Gọi mảng F là mảng cộng dồn – là một mảng có kích thước như A và giá trị được tính theo công thức:

Để thực hiện việc tính mảng F, ta sử dụng đoạn code đơn giản sau:

**f[0]=0;**

**for (int i=1; i<=n; i++)**

**f[i]=f[i-1]+a[i];**

Bài toán: Cho mảng A kích thước N phần tử. Ban đầu A[i]=0; Thực hiện M truy vấn: Tăng giá trị đoạn từ A[a] đến A[b] một giá trị Val; Tìm giá trị lớn nhất trong mảng sau M truy vấn?

**Input:**

* Dòng đầu ghi số nguyên N (): số lượng phần tử; M số lượng truy vấn, Val: giá trị tăng
* M dòng tiếp theo thể hiện M truy vấn: mỗi dòng gồm hai số i,j cách nhau bởi dấu cách

**Output:**

* Ghi một số nguyên duy nhất là giá trị lớn nhất mảng sau khi thực hiện lần lượt M truy vấn

**Ví dụ:**

|  |  |
| --- | --- |
| Input | Output |
| 5 3 100  2 4  1 3  1 2 | 300 |

Giải thích:

* Sau lượt 1: A={0;100;100;100;0}
* Sau lượt 2: A={100;200;200;100;0}
* Sau lượt 3: A={200;300;200;100;0}
* Vậy giá trị lớn nhất là 300.

**Hướng dẫn:**

Sol1 (Trâu): mỗi cặp [a,b] ta tăng Val cho các phần tử từ A[a] đến A[b];

Sol2 (dùng mảng cộng dồn F):

* Mỗi cặp [a,b] ta cộng vào A[a]+=Val; và A[b+1]-=Val

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  int main() {  int N, M, Val;  cin >> N >> M >> Val;  vector<long long> diff(N + 2, 0); // Mảng chênh lệch (N+2 để tránh out of bound)  // Nhập M truy vấn và cập nhật mảng chênh lệch  for (int q = 1; q <= M; q++) {  int i, j;  cin >> i >> j;  diff[i] += Val;  diff[j + 1] -= Val; // Giảm tại j+1 để giữ đúng biên  }  // Khôi phục mảng A từ mảng chênh lệch  long long max\_value = 0, current = 0;  for (int i = 1; i <= N; i++) {  current += diff[i];  max\_value = max(max\_value, current);  }  cout << max\_value << endl;  return 0;  } |

**2.6: Kĩ thuật đếm & set &map…**

**Bài toán:**

**Phát biểu bài toán:**

Cho một mảng gồm n số nguyên. Hãy đếm số lượng các phần tử khác nhau trong mảng.

**Dữ liệu vào:**

* Một số nguyên n (số lượng phần tử trong mảng).
* Một mảng gồm n số nguyên.

**Dữ liệu ra:**

* Một số nguyên là số lượng phần tử khác nhau trong mảng.

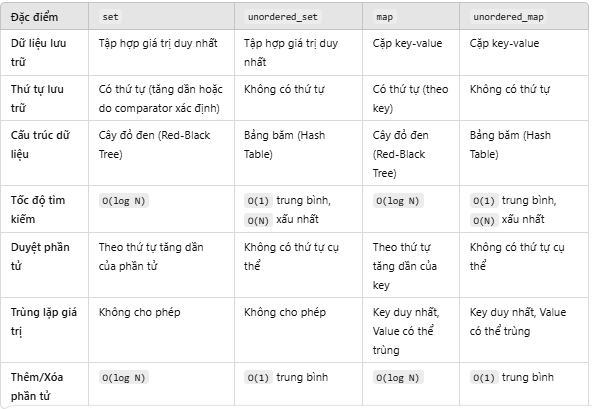
|  |  |
| --- | --- |
| **input** | **ouput** |
| 5  1 1 2 3 3 | 3 |

Bài toán đặt ra:

- Nếu phạm vi giá trị của mảng nhỏ và biết trước (ví dụ: [−106,106].

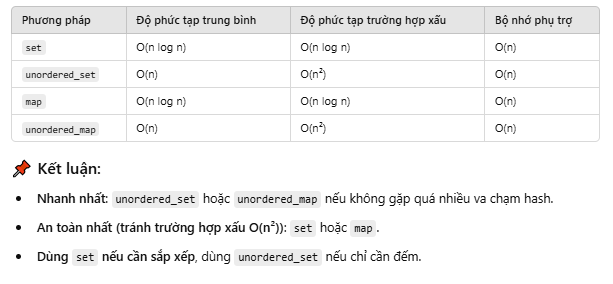
🡪Dùng mảng đếm

|  |
| --- |
| #include <bits/stdc++.h>  #define int long long  using namespace std;  int n,m,k;  int d[1000003], a[1000003];  int kq=0;  main()  {  cin>>n;  for(int i=1;i<=n;i++)  {  cin>>a[i];  d[a[i]]++;  }  for(int i=1;i<=n;i++)  {  if(d[a[i]]>0) kq++;  d[a[i]]=0;  }  cout<<kq;  } |

****

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  int main() {  // Khai báo set - Duy trì thứ tự tăng dần  set<int> s1 = {5, 2, 8, 1, 3,3};  cout << "Set: ";  for (int x : s1) {  cout << x << " "; // Output: 1 2 3 5 8  }  cout << "\n";  // Khai báo unordered\_set - Không có thứ tự cố định  unordered\_set<int> s2 = {5, 2, 8, 1, 3,3};  cout << "unordered\_set: ";  for (int x : s2) {  cout << x << " "; // Output có thể thay đổi mỗi lần chạy  }  cout << "\n";  // Khai báo map - Duy trì thứ tự key tăng dần  map<int, string> mp = {{2, "Hai"}, {1, "Mot"}, {3, "Ba"}};  cout << "Map:\n";  for (auto p : mp) {  cout << p.first << " -> " << p.second << "\n"; // Output: 1 -> Mot, 2 -> Hai, 3 -> Ba  }  // Khai báo unordered\_map - Không có thứ tự cố định  unordered\_map<int, string> mp2 = {{2, "Hai"}, {1, "Mot"}, {3, "Ba"}};  cout << "Unordered Map:\n";  for (auto p : mp2) {  cout << p.first << " -> " << p.second << "\n"; // Output có thể thay đổi mỗi lần chạy  }  } |

|  |  |
| --- | --- |
| #include <bits/stdc++.h>  using namespace std;  int main() {  int n;  cin >> n;  set<int> s;  for (int i = 0; i < n; i++) {  int x;  cin >> x;  s.insert(x); // Chèn vào set  }  cout << s.size() << endl;  for (int x : s) { cout << x << " "; }  } | #include <bits/stdc++.h>  using namespace std;  int main() {  int n;  cin >> n;  unordered\_set<int> s;  for (int i = 0; i < n; i++) {  int x;  cin >> x;  s.insert(x); // Chèn vào set  }  cout << s.size() << endl;  for (int x : s) { cout << x << " "; }  } |
| #include <iostream>  #include <map>  using namespace std;  int main() {  int n;  cin >> n;  map<int, int> mp;  for (int i = 0; i < n; i++) {  int x;  cin >> x;  mp[x]++; // Đếm số lần xuất hiện của từng số  }  cout << mp.size() << endl;  for (auto it : mp)  { cout << it.first << " "; }  } | #include <iostream>  #include <unordered\_map>  using namespace std;  int main() {  int n;  cin >> n;  unordered\_map<int, int> freq;  for (int i = 0; i < n; i++) {  int x;  cin >> x;  freq[x]++; // Đếm số lần xuất hiện của từng số  }  cout << freq.size() << endl;  for (auto it : freq)  { cout << it.first << " "; }  } |



**2.6 I. Giới thiệu kĩ thuật 2 con trỏ.**

Hai con trỏ thực sự là một kĩ thuật dễ dàng và hiệu quả dùng trong tìm kiếm các cặp thỏa mãn điều kiện nào đó trong mảng đã được sắp xếp.

Ví dụ: Cho một mảng đã sắp xếp A (được sắp xếp theo thứ tự tăng dần), có N số nguyên, hãy tìm xem tồn tại bất kì một cặp (A[i], A[j]) mà có tổng bằng X.

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  int a[10003],n,x;  bool findd(int A[], int N, int X) {  int left = 1, right = N;  while (left < right) {  int sum = A[left] + A[right];  if (sum == X) {  cout << A[left] << " " << A[right]<<endl;  return true;  }  if (sum < X) left++;  else right--;  }  cout << "Không tìm thấy cặp nào.\n";  return false;  }  main() {  cin>>n>>x;  for(int i=1;i<=n;i++)  cin>>a[i];  sort(a+1,a+n+1);  finddum(a,n, x);  } |

Code in ra tất cả các cặp:

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std  int a[10003], n, x;  void findd(int A[], int N, int X) {  int left = 1, right = N;  bool found = false; // Biến kiểm tra xem có cặp nào hợp lệ không  while (left < right) {  int sum = A[left] + A[right];  if (sum == X) {  cout << A[left] << " " << A[right] << endl;  found = true;  left++; // Tiếp tục tìm tiếp các cặp khác  right--;  }  else if (sum < X) {  left++;  }  else {  right--;  }  }  if (!found) {  cout << "Not found.\n";  }  }  int main() {  cin >> n >> x;  for (int i = 1; i <= n; i++)  cin >> a[i];  sort(a + 1, a + n + 1);  findd(a, n, x);  } |

Code in ra cặp cuối cùng nếu có nhiều cặp:

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  int a[10003], n, x;  void findd(int A[], int N, int X) {  int left = 1, right = N;  int lastLeft = -1, lastRight = -1; // Lưu cặp hợp lệ cuối cùng  while (left < right) {  int sum = A[left] + A[right];  if (sum == X) {  lastLeft = A[left];  lastRight = A[right];  left++; // Tiếp tục tìm để lấy cặp cuối cùng  right--;  }  else if (sum < X) {  left++;  }  else {  right--;  }  }  if (lastLeft != -1 && lastRight != -1) {  cout << lastLeft << " " << lastRight << endl;  } else {  cout << "NOT FOUND.\n";  }  }  int main() {  cin >> n >> x;  for (int i = 1; i <= n; i++)  cin >> a[i];  sort(a + 1, a + n + 1); // Sắp xếp mảng trước khi dùng hai con  findd(a, n, x);  } |

# **III. Một số bài tập áp dụng:**

## Bài 1: Xếp mã số học sinh

**Đề bài:**

Trong thời gian tiến hành Đại hội thể dục thể thao, mỗi học sinh được nhận một số hiệu là một số tự nhiên. Cần sắp danh sách các học sinh tham gia Đại hội theo điểm thi đấu của họ.

Tệp dữ liệu vào là **XEPMASO.INP** Dòng đầu tiên là số - số học sinh N≤105. Trên mỗi dòng tiếp theo cho mã số và điểm thi đấu của mỗi học sinh. Tất cả các số không vượt quá .

Tệp kết quả ra là **XEPMASO.OUT** đưa ra danh sách giảm dần theo thứ tự điểm thi đấu. Nếu hai học sinh cùng số điểm thì sắp tăng theo mã số.

Trong tệp input cũng như tệp output, thông tin về mỗi học sinh ghi trên một dòng, mã số ghi trước, điểm thi đấu ghi sau.

Ví dụ

|  |  |
| --- | --- |
| XEPMASO.INP | XEPMASO.OUT |
| 3  101 80  305 90  200 14 | 305 90  101 80  200 14 |
| 3  20 80  30 90  25 90 | 25 90  30 90  20 80 |

**Ràng buôc:**

* 50% số test tương ứng với 50% số điểm N≤103
* 50% số test tương ứng với 50% số điểm N≤105

**CODE THAM KHẢO:**

1. **Thuật toán trâu O(N2)**

|  |
| --- |
| #include<bits/stdc++.h>  using namespace std;  int n, id[100000], v[100000];  main()  {  // freopen("XEPMASO.INP","r",stdin);  //freopen("XEPMASO.OUT","w",stdout);  cin >> n;  for(int i = 0; i < n; i++)  cin >> id[i] >> v[i];  for(int i = 0; i < n-1; i++)  for(int j = i+1; j < n; j++) {  if(v[j] > v[i]) {  swap(v[j], v[i]);  swap(id[j], id[i]);  }  if(v[i] == v[j] && id[i] > id[j])  swap(id[i], id[j]);  }  for(int i = 0; i < n; i++)  cout << id[i] << ' ' << v[i] << "\n";  **}** |

**Code: Sử dụng quicksort**

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  int a[100005], b[100005], n;  void quickSort(int a[], int b[], int l, int r) {  int i = l, j = r;  int pivot = b[(l + r) / 2]; // Chọn phần tử giữa làm pivot  while (i <= j) {  while (b[i] > pivot) i++;  while (b[j] < pivot) j--;  if (i <= j) {  swap(b[i], b[j]);  swap(a[i], a[j]);  i++;  j--;  }  }  if (l < j) quickSort(a, b, l, j);  if (i < r) quickSort(a, b, i, r);  }  int main() {  cin >> n;    for (int i = 1; i <= n; i++)  cin >> a[i] >> b[i];    quickSort(a, b, 1, n);  int j = 0; // Khởi tạo j  for (int i = 1; i < n; i++) {  if (b[i] == b[i + 1])  j++;  else {  sort(a + i - j, a + i + 1); // Sắp xếp lại chỉ số nếu b[i] giống nhau  j = 0;  }  }  // Xử lý trường hợp cuối cùng nếu có nhiều số bằng nhau ở cuối mảng  if (j > 0)  sort(a + n - j, a + n + 1);  for (int i = 1; i <= n; i++)  cout << a[i] << " " << b[i] << endl;  return 0;  } |

**Tóm lại, thuật toán của bạn chạy trong O(n log n) trong trường hợp trung bình và tốt nhất, nhưng có thể lên O(n²) trong trường hợp xấu nhất.**

|  |
| --- |
| **Thuật toán: Sử dụng Pair // O(n log n)**  #include <bits/stdc++.h>  #define p(a) pair<long long , long long> a  using namespace std;  bool cmp(p(a) , p(b)){  if(a.second == b.second) return a.first < b.first;  return a.second > b.second;  }  pair<long long , long long> a[100005];  long long n;  signed main(){  ios\_base::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  //freopen("XEPMASO.INP" , "r" , stdin);  // freopen("XEPMASO.OUT" , "w" , stdout);  cin >> n;  for(int i = 1;i <= n;i++)  cin >> a[i].first >> a[i].second;  sort(a + 1,a + n + 1, cmp);  for(int i = 1;i <= n;i++)  cout << a[i].first << " " << a[i].second << '\n';  } |

**Câu 1a (1.5 điểm).**

Trong năm học vừa qua, để có những gói quà gửi cho các bạn học sinh bị Covid-19. Các nhà trường đã kêu gọi sự chung tay, giúp sức của các bạn học sinh. Sau một thời gian, nhà trường đã thu được một số món quà nhất định. Để đóng gói n (n 100) gói quà có kích thước a1, a2,….,an (|ai| 1000) vào các hộp có kích thước m (m 2000). Bình là một bạn học sinh rất sôi nổi khi tham gia vào các hoạt động này.

***Yêu cầu:*** Em hãy giúp bạn Bình lựa chọn 2 gói quà đầu tiên trong n gói quà, có thể đóng vừa hộp có kích thước m.

***Dữ liệu vào:*** Lấy dữ liệu từ tệp **Cau1a.inp**

- Dòng đầu chứa 2 số nguyên n, m (0<n 106, m 2000).

- Dòng thứ 2 chứa n số nguyên a1, a2,….,an (|ai| 1000).

***Dữ liệu ra:*** Kết quả được ghi ra tệp **Cau1a.out**

- Ghi ra vị trí đầu tiên mà 2 số có tổng là m trong dãy a.

- Nếu không tồn tại số nào, ghi ra -1.

***Ví dụ:***

|  |  |
| --- | --- |
| **Cau1a.inp** | **Cau1a.out** |
| 6 10  3 4 1 7 5 2 | 1 4 |
| 5 5  1 2 3 4 5 | 1 4 |
| 5 5  1 1 4 4 5 | 1 3 |

**Sub 1: n<=103 Sub 2: n<=106**

|  |
| --- |
| #include<bits/stdc++.h>  #define int long long  using namespace std;  int a[1000003];  int n,m;  int d[1000003];  unordered\_map<int, int> s;  pair <int, int> mp;  int sl=-1;  vector<pair<int, int>> pairs;  int t,tt;  main()  {  cin>>n>>m;  for(int i=1; i<=n; i++)  {  cin>>a[i];  }  for(int i=1; i<=n; i++)  {  if (s[m-a[i]]!=0) {  t=s[m-a[i]];  tt=i;  pairs.push\_back(t,tt);  }  //if(s[m-a[i]]==0)  s[a[i]] = i;  }  if (!pairs.empty()) {  sort(pairs.begin(), pairs.end()); // Sắp xếp để ưu tiên cặp có index nhỏ nhất  cout << pairs[0].first << " " << pairs[0].second << endl;  } else {  cout << "-1" << endl;  } |

## Bài 2: Sắp người xuống bè

**Đề bài:**

Các nhà thám hiểm được tập hợp trong cuộc thám hiểm đến miền Bắc cực. Họ có một chiếc bè lớn gồm N×M chiếc bè nhỏ gắn với nhau. Mỗi chiếc bè nhỏ có một sức chứa riêng, và mỗi nhà thám hiểm cũng có trọng lượng của mình. Mỗi chiếc bè nhỏ không thể chở hơn một nhà thám hiểm. Nếu sức chứa của chiếc bè nhỏ hơn trọng lượng nhà thám hiểm chọn nó thì nhà thám hiểm đó có thể bị chết đuối khi bước xuống chiếc bè nhỏ này. Người lãnh đạo cuộc thám hiểm nghĩ cách xếp bè. Hãy giúp đỡ ông ta xác định số nhiều nhất các nhà thám hiểm có thể đi.

Tệp dữ liệu vào là **SAPBE.INP**. Dòng đầu tiên là các số N và M (1≤N, M≤40). Trong N dòng tiếp theo, mỗi dòng M số là sức chứa M bè nhỏ. Dòng thứ N+2 là số K (1≤K≤2000) là số lượng các nhà thám hiểm. Trong dòng thứ N+3 chứa K số, số thứ *i* trong chúng là trọng lượng nhà thám hiểm thứ *i*. Tất cả trọng lượng các nhà thám hiểm và sức chứa các bè không vượt quá 109.

Tệp kết quả ra là **SAPBE.OUT** đưa ra một số là số nhiều nhất các nhà thám hiểm tham gia cuộc thám hiểm này.

Ví dụ

|  |  |
| --- | --- |
| SAPBE.INP | SAPBE.OUT |
| 3 2  5 10  7 5  5 5  6  9 5 3 5 12 10 | 4 |

**Giải:**

**Test:** <https://drive.google.com/open?id=17J0L8XehKWweCk_nPiUGY8otLfYZi_u8>

|  |
| --- |
| #include<bits/stdc++.h>  #define int long long  using namespace std;  int a[1000007],b[1000007],m,n,ii,res;  main()  {  freopen("sapbe.inp","r",stdin); freopen("sapbe.out","w",stdout);  cin>>m>>n;  for(int i=1;i<=m\*n;i++) cin>>a[i];  sort(a+1,a+1+m\*n);  m\*=n;  cin>>n;  for(int i=1;i<=n;i++) cin>>b[i];  sort(b+1,b+1+n);  ii=0;  for(int i=1;i<=n&&ii<=m;i++)  {  do  ii++;  while(b[i]>a[ii]);  cout<<i<<' '<<ii<<'\n';  res++;  }  cout<<res-1;  } |

Cải tiến:

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  int n,m,k,a[100003],b[100003];  int d=0;  main()  {  cin>>n>>m;  for(int i=1;i<=n\*m;i++)  {  cin>>a[i];  }  cin>>k;  for(int i=1;i<=k;i++)  {  cin>>b[i];  }  sort(a+1,a+m\*n+1);  sort(b+1,b+k+1);  int j=1;  int vt=0;  for(int i=1;i<=n\*m;i++)  {  vt=lower\_bound(a+vt+1,a+n\*m+1,b[j])-a;  {  if(vt<=n\*m && j<=k)  {  //cout<<vt<<" ";  d++;  j++;  }  else  break;  }  }  cout<<d;  } |

**Bài 3 : dayso.cpp**

Dãy Cho hai dãy số nguyên 𝑥1, 𝑥2, … , 𝑥𝑚 và 𝑦1, 𝑦2, … , 𝑦𝑛 mà A tạo ra, cho 𝑠1, 𝑠2, … , 𝑠𝑘 là 𝑘 câu hỏi của 𝐵. Với câu hỏi 𝑠𝑖 (𝑖 = 1,2, … , 𝑘) đưa ra giá trị chênh lệch nhỏ nhất của 𝑠𝑖 với tổng hai số tìm được.

**Dữ liệu:**

- Dòng đầu chứa ba số nguyên dương 𝑚, 𝑛, 𝑘;

- Dòng thứ hai chứa 𝑚 số nguyên 𝑥1, 𝑥2, … , 𝑥𝑚 (|𝑥𝑖| ≤ 109);

- Dòng thứ ba chứa 𝑛 số nguyên 𝑦1, 𝑦2, … , 𝑦𝑛 (|𝑦𝑖| ≤ 109);

- Dòng thứ tư chứa 𝑘 số nguyên 𝑠1, 𝑠2, … , 𝑠𝑘 (|𝑠𝑖| ≤ 109).

**Kết quả:**

Gồm 𝑘 dòng, dòng thứ 𝑖 ghi giá trị chênh lệch nhỏ nhất của 𝑠𝑖 với tổng hai số tìm được.

**Ví dụ:**

|  |  |
| --- | --- |
| input | output |
| 3 4 3 1 3 2 -1 5 3 1 1 2 9 | 0  0 1 |

**Ràng buộc:**

* Có 40% số test ứng với 40% số điểm của bài có 𝑚, 𝑛 ≤ 1000; 𝑘 ≤ 10;
* Có 40% số test khác ứng với 40% số điểm của bài có 𝑚, 𝑛 ≤ 105; 𝑘 ≤ 10;
* Có 20% số test còn lại với 20% số điểm còn lại của bài có 𝑚, 𝑛 ≤ 105; 𝑘 ≤ 500.

|  |
| --- |
| #include <bits/stdc++.h>  #define int long long  using namespace std;  const int N = 1e5 + 5;  int m , n , k;  int x[N] , y[N];  void sub1(){  while(k--){  int l;  cin >> l;  int res = 2e9;  for(int i = 1;i <= m;i++)  for(int j = 1;j <= n;j++)  res = min(res , abs(l - (x[i] + y[j])));  cout << res << '\n';  }  }  void sub2(){  while(k--){  int l;  cin >> l;  int res = 2e9;  for(int i = 1;i <= m;i++){  int j = lower\_bound(y + 1 , y + n + 1 , l - x[i]) - y;  if(j == 1)  res = min(res , abs(l - (x[i] + y[1])));  else if(j == n + 1)  res = min(res , abs(l - (x[i] + y[n])));  else res = min({res , abs(l - (x[i] + y[j - 1])) , abs(l - (x[i] + y[j])) });  }  cout << res << '\n';  }  }  void sub3(){  while(k--){  int l;  cin >> l;  int res = 2e9;  int i = 1, j = n;  y[0] = INT\_MAX;  while(i <= m){  while(j > 0 && x[i] + y[j - 1] >= l)  j--;  if(j < 1)  j++;  res = min({res , abs(l - x[i] - y[j]) , abs(l - x[i] - y[j - 1])});  i++;  }  cout << res << '\n';  }  }  signed main(){  // freopen("test.inp" , "r" , stdin);  // freopen("test.out" , "w" , stdout);  ios\_base::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  cin >> m >> n >> k;  for(int i = 1;i <= m;i++)  cin >> x[i];  for(int i = 1;i <= n;i++)  cin >> y[i];  sort(x + 1 , x + m + 1);  sort(y + 1 , y + n + 1);  if(m <= 1e3 && n <= 1e3 && k <= 10)  sub1();  else if(m <= 1e5 && n <= 1e5 && k <= 10)  sub2();  else if(m <= 1e5 && n <= 1e5 && k <= 500)  sub3();  } |

sub3();

|  |
| --- |
| cin>>m>>n>>k;  for (int i=1; i<=m; i++)  cin>>x[i];  for (int i=1; i<=n; i++)  cin>>y[i];  sort(x+1,x+m+1);  sort(y+1,y+n+1);  while (k--) {  cin>>s;  int l=1,r=n,diff=INT\_MAX;  while (l<=m && r>=1) {  if (diff==0)  break;  if (abs(x[l]+y[r]-s)<diff)  diff=abs(x[l]+y[r]-s);  if (x[l]+y[r]>s)  r--;  else  l++;  }  printf("%d\n",diff);  }  } |

## Bài 3: Khiêu vũ

**Đề bài:**

Một làng quê có chàng trai đánh số từ 1 tới và cô gái đánh số từ 1 tới . Chàng trai thứ có chiều cao (), cô gái thứ có chiều cao ().

Trong một buổi khiêu vũ, người ta muốn chọn ra một số cặp nhảy. Mỗi cặp nhảy gồm đúng 1 chàng trai và 1 cô gái và trong cặp đó, chàng trai phải cao hơn cô gái. Mỗi chàng trai, cô gái trong làng không được tham gia quá 1 cặp nhảy.

**Yêu cầu:** Tìm một số nhiều nhất các cặp nhảy thỏa mãn yêu cầu trên.

**Dữ liệu:** Vào từ file văn bản DANCE.INP

* Dòng 1 chứa hai số nguyên dương
* Dòng 2 chứa số nguyên dương ()
* Dòng 3 chứa số nguyên dương ()

*Các số trên một dòng được ghi cách nhau ít nhất một dấu cách*

**Kết quả:** Ghi ra file văn bản DANCE.OUT một số nguyên duy nhất là số cặp nhảy theo phương án tìm được.

**Ví dụ**

|  |  |
| --- | --- |
| DANCE.INP | DANCE.OUT |
| 3 2  1 2 3  2 3 | 1 |
| 4 5  1 2 3 4  2 2 1 4 5 | 3 |

**Chú ý:** Ít nhất 50% số điểm ứng với các test có .

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  long long n,m,a[100005],b[100005],res;  int main()  {  freopen("dance.inp","r",stdin);  freopen("dance.out","w",stdout);  cin>>m>>n;  for(int i=1; i<=m; i++)  cin>>a[i];  for(int i=1; i<=n; i++)  cin>>b[i];  sort(a+1,a+m+1,greater<long long>());  sort(b+1,b+n+1,greater<long long>());  int i=1,j=1;  while(i<=m&&j<=n)  {  if(a[i]>b[j])  {  res++;  i++;  j++;  }  else j++;  }  cout<<res;  } |

**Giải:**

Ta nhận thấy, sử dụng thuật toán sắp xếp để sắp 2 mảng tưởng ứng với chàng trai và cô gái. Sau khi sắp xong, dựa theo điều kiện đầu bài để tìm các cặp phù hợp, khi tìm được thì tăng biến đếm. Với giới hạn bài này nếu sử dụng thuật toán sắp xếp thường chỉ được 50% số điểm. Ta sử dụng thuật toán Quick Sort để đảm bảo về mặt thời gian.

Chương trình tham khảo:

**Test:** <https://drive.google.com/open?id=1UCoV3A817hS-qWmq8hmupeA22iadRJfS>

## Bài 4: Hình vuông

**Đề bài:**

Trên mặt phẳng với hệ tọa độ Descartes vuông góc cho điểm. Hãy tìm hình vuông nhỏ nhất có cạnh song song với một trong hai trục tọa độ chứa tất cả điểm đã cho (điểm nằm trên cạnh hình vuông cũng bị tính là chứa trong hình vuông)

**Dữ liệu:** Vào từ file văn bản **SQUARE.INP**:

* Dòng 1 chứa số nguyên dương
* dòng tiếp theo, dòng thứ chứa hai số nguyên là tọa độ của một điểm ,

**Kết quả:** Ghi ra file văn bản **SQUARE.OUT** một số nguyên duy nhất là diện tích hình vuông tìm được

0

1

2

3

4

5

6

7

1

2

3

4

5

6

7

**Ví dụ**

|  |  |
| --- | --- |
| SQUARE.INP | SQUARE.OUT |
| 3  3 4  5 7  4 3 | 16 |

**Giải:**

Ta nhận thấy để tìm được hình vuông nhỏ nhất chứa tất cả các điểm, tức là tìm hiệu độ dài lớn nhất giữa các giá trị lớn nhất và nhỏ nhất tương ứng với hai trục x,y. Vì vậy ta có thể sắp tăng 2 mảng, sau đó trừ phần tử cuối và phần tử đầu rồi so sánh, lấy giá trị lớn nhất bình phương chính là diện tích hình vuông ta cần tìm.

Chương trình tham khảo:

#include <bits/stdc++.h>

using namespace std;

void quickSort(int A[], int l, int r)

{

if(l>=r) return;

int i=l;

int j=r;

int x=A[(l+r)/2];

while(i<=j)

{

while(A[i]<x) i++;

while(A[j]>x) j--;

if(i<=j)

{

int temp=A[i];

A[i]=A[j];

A[j]=temp;

i++;

j--;

}

}

quickSort(A,l,j);

quickSort(A,i,r);

}

int main()

{

freopen("square.inp","r",stdin);

freopen("square.out","w",stdout);

long long n;

cin >> n;

long long a[n], b[n];

for (long long i = 0; i < n; i++) cin >> a[i] >> b[i];

//quickSort(a,0,n);

//quickSort(b,0,n);

sort(a,a+n);

sort(b,b+n);

long long c,d,maxx;

c = a[n-1]-a[0];

d = b[n-1]-b[0];

if (c>=d) maxx = c;

else maxx = d;

cout << maxx \* maxx;

return 0;

}

**Test:** <https://drive.google.com/open?id=1aoyAWU7TBT6_01rPherGdRuCvxsyS47o>

## Bài 5: Running

**Đề bài:**

Sau nhiều ngày học căng thẳng, thầy giáo quyết định cho bạn học sinh lớp Tin thi chạy với khoảng cách là 100m. Vì phải bấm giờ nên thầy đã nhờ một bạn ghi lại điểm thi chạy của bạn học sinh. Thầy có xem qua bảng ghi điểm và nhận thấy Minh là người về nhì. Thật không may, do sơ suất nên thầy đã làm đổ cốc trà đá lên tờ giấy ghi điểm làm nhòe hết phần kết quả. Thầy chỉ còn danh sách gồm bạn học sinh và số điểm của các bạn. Hãy giúp thầy tìm ra điểm Minh và hai bạn có thời gian chạy cao nhất để trao thưởng.

Input

* Dòng đầu tiên chứa số nguyên dương là số bạn học sinh tham gia chạy.
* Dòng thứ hai chứa số nguyên lần lượt là điểm thi chạy của bạn (thứ tự theo số hiệu của các bạn từ đến ).

Output

* Dòng đầu tiên xuất ra điểm của Minh, nếu không thể xác nhận được thì xuất ra "-1".
* Dòng thứ hai ghi số hiệu của hai bạn có điểm cao nhất

|  |  |
| --- | --- |
| Running.inp | Running.out |
| 6  1 5 3 3 4 4 | 4  2 5 |
| 5  1 1 1 1 1 | -1  1 2 |
| 10  10 3 3 3 4 10 3 4 10 3 | 4  1 6 |

**Giải:**

Ta nhận thấy đây là bài toán tìm giá trị lớn thứ hai trong mảng một chiều. Khởi trị hai biến: *nhat* và *nhi* bằng giá trị âm có giá trị tuyệt đối khá lớn. Duyệt mảng A chứa các điểm thi từ A[0] đến A[N-1]:

if a[i]>nhat then begin

if (nhat>nhi) then nhi := nhat;

nhat := a[i];

end else

if (a[i]<nhat) and (a[i]>nhi) then nhi := a[i];

Sau khi thực hiện câu a) tìm được giá trị lớn thứ nhất là *nhat* và giá trị lớn thứ nhì là *nhi*. Nếu có hai phần tử bằng nhau thì duyệt mảng A đưa ra vị trí của hai phần tử này. Nếu chỉ có một phần tử bằng nhau thì đưa ra vị trí của phần tử này rồi duyệt lại mảng A, đưa thêm vị trí của một phần tử bằng *nhi*.

Chương trình tham khảo:

#include <fstream>

#include <vector>

using namespace std;

int vc = -2000000000;

int n, nhat, nhi, x;

vector<int> a;

ifstream f;

ofstream g;

int main() {

*// Đọc dữ liệu từ tệp input*

f.open("running.inp");

f >> n;

for (int i=0;i!=n;i++) {

f>>x;

a.push\_back(x);

}

f.close();

*// Tìm giá trị lớn nhất, lớn nhì*

nhat = vc; nhi = vc;

for (int i=0; i!=n; i++) {

if (a[i]>nhat) {

if(nhat>nhi) nhi=nhat;

nhat=a[i];

} else

if((a[i]<nhat)&&(a[i]>nhi))

nhi=a[i];

}

*// Ghi kết quả vào output*

g.open("running.out");

if (nhi!=vc)

g<<nhi<<endl;

else

g<<"vo nghiem"<<endl;

*/\* ghi chỉ số 2 phần tử trong dãy A có giá trị cao nhất \*/*

int count=0;

for (int i=0;i!=n;i++) {

if (a[i]==nhat) {

count++;

g<<i+1 <<" ";

if (count==2) break;

}

}

if (count==1)

for (int i=0;i!=n;i++) {

if (a[i]==nhi) {

count++; g<<i+1;

if (count==2) break;

}

}

g.close();

return 0;

}

**Test:** <https://drive.google.com/open?id=1WS_v3HwZ9F59P5CvnhzSo_0qO3NpMXAy>

## Bài 6: NKSGAME – VOI08 Trò chơi với dãy số

**Đề bài:**

Hai bạn học sinh trong lúc nhàn rỗi nghĩ ra trò chơi sau đây. Mỗi bạn chọn trước một dãy số gồm n số nguyên. Giả sử dãy số mà bạn thứ nhất chọn là: b1, b2, ..., bn còn dãy số mà bạn thứ hai chọn là c1, c2, ..., cn.

Mỗi lượt chơi mỗi bạn đưa ra một số hạng trong dãy số của mình. Nếu bạn thứ nhất đưa ra số hạng , còn bạn thứ hai đưa ra số hạng thì giá của lượt chơi đó sẽ là

**Ví dụ:** Giả sử dãy số bạn thứ nhất chọn là 1, -2; còn dãy số mà bạn thứ hai chọn là 2, 3. Khi đó các khả năng có thể của một lượt chơi là (1, 2), (1, 3), (-2, 2), (-2, 3). Như vậy, giá nhỏ nhất của một lượt chơi trong số các lượt chơi có thể là 0 tương ứng với giá của lượt chơi (-2, 2).

**Yêu cầu:** Hãy xác định giá nhỏ nhất của một lượt chơi trong số các lượt chơi có thể.

**Dữ liệu**

* Dòng đầu tiên chứa số nguyên dương
* Dòng thứ hai chứa dãy số nguyên
* Dòng thứ hai chứa dãy số nguyên , , ...,  (|| ≤ , i=1, 2, ..., n)

Hai số liên tiếp trên một dòng được ghi cách nhau bởi dấu cách.

**Kết quả:** Ghi ra giá nhỏ nhất tìm được.

Ràng buộc: 60% số tests ứng với 60% số điểm của bài có 1 ≤ n ≤ 1000.

**Ví dụ:**

|  |  |
| --- | --- |
| NKSGAME.inp | NKSGAME.out |
| 2  1 -2  2 3 | 0 |

**Giải:**

Nhận thấy bài toán ta cần tìm là với mỗi cần tìm một giá trị sao cho nhỏ nhất. Như vậy ta cần sắp lại mảng , với mỗi giá trị ta tìm kiếm nhị phân thỏa mãn gần nhất.

Chương trình tham khảo:

#include <bits/stdc++.h>

using namespace std;

const int maxn = 100009;

const int o = 2000000009;

int i, j, n, ans, tmp, b[maxn], a[maxn];

bool cmp (int x, int y) {

return(x < y);

}

int main() {

cin >> n;

for (i = 1; i <= n; i++) cin >> a[i];

for (i = 1; i <= n; i++) cin >> b[i];

sort (b+1, b+n+1, cmp);

ans = o;

for (i = 1; i <= n; i++) {

tmp = lower\_bound(b+1,b+n+1,0-a[i]) - b;

if (tmp > n) tmp = n;

ans = min(ans, abs(a[i] + b[tmp]));

if (tmp == 1) continue;

ans = min(ans, abs(a[i] + b[tmp-1]));

}

cout << ans;

return 0;

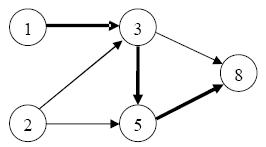
}

**Test:** <https://drive.google.com/open?id=1up87esaQnCYUn9_lgt5_Usrazs0_vxex>

## Bài 7: NKJUMP - VOI08 Lò cò

Nhảy lò cò là trò chơi dân gian của Việt Nam. Người trên hành tinh X cũng rất thích trò chơi này và họ đã cải biên trò chơi này như sau: Trên mặt phẳng vẽ n vòng tròn được đánh số từ 1 đến n. Tại vòng tròn i người ta điền số nguyên dương ai. Hai số trên hai vòng tròn tùy ý không nhất thiết phải khác nhau. Tiếp đến người ta vẽ các mũi tên, mỗi mũi tên hướng từ một vòng tròn đến một vòng tròn khác. Quy tắc vẽ mũi tên là: Nếu có ba số ai, aj, ak thỏa mãn ak = ai + aj thì vẽ mũi tên hướng từ vòng tròn i đến vòng tròn k và mũi tên hướng từ vòng tròn j đến vòng tròn k. Người chơi chỉ được di chuyển từ một vòng tròn đến một vòng tròn khác nếu có mũi tên xuất phát từ một trong số các vòng tròn, di chyển theo cách mũi tên đã vẽ để đi đến các vòng tròn khác. Người thắng cuộc sẽ là người tìm được cách di chuyển qua nhiều vòng tròn nhất.

**Ví dụ:** Với 5 vòng tròn và các số trong vòng tròn là 1, 2, 8, 3, 5, trò chơi được trình bày trong hình dưới đây:



Khi đó có thể di chuyển được nhiều nhất qua 4 vòng tròn (tương ứng với đường di chuyển được tô đậm trên hình vẽ).

**Yêu cầu**

Hãy xác định xem trong trò chơi mô tả ở trên, nhiều nhất có thể di chuyển được qua bao nhiêu vòng tròn.

**Dữ liệu**

* Dòng đầu chứa số nguyên ;
* Dòng thứ hai chứa dãy số nguyên dương

Hai số liên tiếp trên một dòng được ghi cách nhau bởi dấu cách.

**Kết quả**

Ghi ra số lượng vòng tròn trên đường di chuyển tìm được.

**Ràng buộc**

* 60% số tests ứng với 60% số điểm của bài có .

**Ví dụ:**

|  |  |
| --- | --- |
| NKJUMP.inp | NKJUMP.out |
| 5  1 2 8 3 5 | 4 |

**Giải:**

Ta thấy rằng từ một vòng chỉ có thể nhảy được đến vòng lớn hơn. Sử dụng quy hoạch động ta có thể giải bài toán.

* Bước đầu sắp xếp lại mảng
* Gọi là số bước nhảy lớn nhất để nhảy tới ô thứ i.
* với và tồn tại k sao cho
* Kiểm tra bằng tìm kiếm nhị phân

Chương trình tham khảo:

#include <bits/stdc++.h>

using namespace std;

const int maxn = 1009;

int i, j, n, a[maxn], b[maxn], f[maxn];

int main() {

freopen("NKJUMP.inp","r",stdin);

freopen("NKJUMP.out","w",stdout);

cin >> n;

for (int i = 1; i <= n; i++) cin >> a[i];

sort(a+1,a+n+1);

int tmp, ans = 0;

for (int i = 1; i <= n; i++) f[i] = 1;

for (int i = 3; i <= n; i++) {

for (int j = 1; j < i; j++) {

bool tmp = binary\_search(a+1,a+j,a[i] - a[j]);

if (tmp == 1) {

f[i] = max(f[i], f[j] + 1);

}

}

ans = max(ans, f[i]);

}

cout << ans << endl;

return 0;

}

**Test:** <https://drive.google.com/open?id=1u2p8gydkidpsjJ8PM1xENw0JRKHP_Kpw>

## Bài 8: Chênh lệch

**Đề bài:**

Cho một dãy số gồm số nguyên dương . Với một cặp số bạn hãy tìm độ chênh lệch bé nhất khi chia đoạn con thành hai phần.

Bạn hãy xem ví dụ:

- dãy A: 3 1 4 2 5

- Với cặp (2,5) bạn cần tìm chênh lệch nhỏ nhất khi chia đoạn này ra hai phần. Ví dụ có các cách chia sau: (1) và (4,2,5); (1,4) và (2,5); (1,4,2) và (5); (1,4,2,5) và (); Khi đó chênh lệch bé nhất là 2 – Tương ứng với cặp (1,4) (2,5).

**Input:**

  -  Dòng đầu tiên là hai số nguyên dương là số lượng truy vấn.

  -  Dòng tiếp theo là số nguyên dương trong dãy

  -  dòng tiếp theo mỗi dòng là 2 số nguyên dương . Mỗi cặp số là một truy vấn cần bạn trả lời.

**Output:**

  -  Gồm Q dòng, dòng thứ là giá trị chênh lệch nhỏ nhất tìm được ứng với truy vấn .

**Ví dụ:**

|  |  |
| --- | --- |
| chenhlech.inp | chenhlech.out |
| 5 1 3 1 4 2 5 2 5 | 2 |

**Giải:**

Ta sử dụng tìm kiếm nhị phân để tìm giá trị nhỏ nhất trong đoạn cần tìm. Mỗi lần tìm được một giá trị, ta lưu vào một mảng riêng . Kết thúc in các giá trị đã lưu trong mảng .

Chương trình tham khảo:

#include <bits/stdc++.h>

using namespace std;

long long n,m,res,x,y,x1,x2,mid,i,k,o,l,r;

long long a[1000006],b[1000006],c[1000005];

int main()

{

freopen("chenhlech.inp","r",stdin);

freopen("chenhlech.out","w",stdout);

cin>>n>>m;

for(i=1;i<=n;i++)

{

cin>>a[i];

b[i]=b[i-1]+a[i];

}

while(m>0)

{

m--;o++;res=1e9;

cin>>x>>y;

l=x;r=y;

while(l<=r)

{

mid=(l+r)/2;

x1=b[mid]-b[x-1];

x2=b[y]-b[mid];

res=min(res,abs(x1-x2));

if(x1==x2)

{

c[o]=res;

break;

}

if(x1<x2)l=mid+1;

else r=mid-1;

}

if(l>r)c[o]=res;

}

for(i=1;i<=o;i++)

cout<<c[i]<<endl;

return 0;

}

**Test:** <https://drive.google.com/open?id=1XcRa3AJulDC_BcKio7MLjkUhL01J2gii>

## Bài 9: Mobi – Đặt trạm phát sóng (OLP 30/4)

**Đề bài:**

Nhà cung cấp dịch vụ viễn thông Mobi đã khảo sát số lượng người sẽ dùng dịch vụ trên một con đường thẳng mới được xây dựng và đánh dấu lại những vị trí trên con đường này. Đầu con đường được đánh tọa độ bắt đầu từ 0. Tại vị trí có tọa độ **X (X nguyên dương)** có số lượng người sẽ sử dụng dịch vụ là **Y**. Trước mắt, nhà cung cấp dịch vụ cần đặt **một** trạm phát sóng có bán kính phủ sóng là **K** đơn vị chiều dài để phủ sóng cho một số người sử dụng dịch vụ trên con đường này.

**Yêu cầu**: Bạn hãy xác định vị trí đặt trạm phát sóng (tọa độ nguyên dương) sao cho trạm có thể phục vụ được số lượng người sử dụng nhiều nhất có thể.

**Dữ liệu**:

- Dòng đầu tiên ghi hai số nguyên **N** và **K** (0 < **N** ≤ 106, 0 < **K** ≤ 2\*106), trong đó **N** là số điểm dân cư đã được đánh dấu, **K** là bán kính phủ sóng của trạm.

- Trong N dòng tiếp theo, dòng thứ i (i=1..N) ghi hai số nguyên **X**và **Y** cho biết tại vị trí **X** có số lượng người dùng là **Y** (0 ≤ **X** ≤ 106, 0 ≤ **Y** ≤104). Các số trên cùng dòng viết cách nhau ít nhất một dấu cách.

**Kết quả**: một số nguyên cho biết số người dùng nhiều nhất sẽ được phục vụ.

**Ví dụ:**

|  |  |
| --- | --- |
| mobi.inp | mobi.out |
| 4 3 7 4 15 10 2 2 1 5 | 11 |

**Giải:**

Ta đưa các cặp x,y vào vector. Sắp xếp lại vector và chặt nhị phân để tìm ra các giá trị thỏa mãn khoảng cách giữa trạm và điểm dân cư là k. Với các giá trị thỏa mãn yêu cầu đầu bài ta cộng số người dân ở mỗi điểm vào tổng. Cập nhật mỗi lần tìm ra điểm mới và so sánh với tổng cũ. Tổng cuối cùng là số người nhiều nhất được phục vụ.

Chương trình tham khảo:

#include<bits/stdc++.h>

using namespace std;

pair<int,int>a[1000000];

long long n,i,j,k,maxx=0,e,tong=0,t[1000005];

long long bs(long long b,long long l,long long r)

{

long long kq=0;

while(l<=r)

{

long long mid=(l+r)/2;

if(a[mid].first==b)return mid;

else

{

if(a[mid].first>b)r=mid-1;

else{kq=mid;l=mid+1;}

}

}

return kq;

}

int main()

{

ios::sync\_with\_stdio(false);cin.tie();

freopen("mobi.inp","r",stdin);

freopen("mobi.out","w",stdout);

cin>>n>>k;

k\*=2;

for(i=1;i<=n;i++)

cin>>a[i].first>>a[i].second;

sort(a+1,a+n+1);

for(i=1;i<=n;i++)

t[i]=t[i-1]+a[i].second;

for(i=1;i<=n;i++)

{

j=bs(a[i].first+k,1,n);

maxx=max(maxx,t[j]-t[i-1]);

if(j==n)break;

}

cout<<maxx;

return 0;

}

**Test:** <https://drive.google.com/open?id=1YuvXK6TzQTAgFD1JPfdj7Jzqcgoavbd3>

## Một số bài tập tự giải:

Bài 1. Cho dãy số N số nguyên a1≤ a2≤ … ≤ aN, hãy đưa ra thuật toán O(N log N) để tìm 2 chỉ số i, j mà i<j và ai+aj=0

Bài 2. Cho dãy số gồm N số nguyên dương (N ≤ 100 000) số a1, a2, …,aN . Hãy tìm dãy con liên tiếp dài nhất có tổng bằng K.

**Bài 3.** Tìm dãy con liên tiếp dài nhất có tổng bằng 0.

Chuyên đề này đưa ra các bài toán cơ bản nhất nhằm giúp học sinh nắm bắt các thuật toán và bước đầu giải các bài tập dạng sắp xếp, tìm kiếm. Rất mong sự góp ý của các thầy cô.

**IV. Tài liệu tham khảo:**

Hồ Sĩ Đàm và cộng sự, 2009. Tài liệu chuyên tin quyển 1

Lê Minh Hoàng, 2002. Giải thuật và lập trình

<http://vn.spoj.com>

http://vnoi.info

http://tnucoder.net