TRƯỜNG THPT CHUYÊN NGUYỄN TRÃI

HƯỚNG DẪN CHẤM
KỲ THI OLYMPIC CNT LẦN THỨ II NĂM 2026
MÔN THI: TIN HỌC
(Hướng dẫn chấm có 13 trang)

TỔNG QUAN ĐỀ THI

Bài	Tên bài	Tên file chương trình	Tên file input	Tên file output	Điểm
1	Truy vấn số đẹp	BAI1.*	BAI1.INP	BAI1.OUT	2
2	Thi đấu	BAI2.*	BAI2.INP	BAI2.OUT	2
3	Tính chất cân bằng	BAI3.*	BAI3.INP	BAI3.OUT	2
4	Tấm gỗ	BAI4.*	BAI4.INP	BAI4.OUT	2
5	Trạm nghỉ	BAI5.*	BAI5.INP	BAI5.OUT	2

Dấu * là py hoặc cpp tùy theo ngôn ngữ lập trình dùng là Python hoặc C++

Bài 1. Truy vấn số đẹp

Cho hàm T(x) là tổng các ước dương của số nguyên dương x. Một số nguyên dương x được gọi là số đẹp nếu T(x)>2x.

Ví dụ: Số 12 đẹp vì có tổng ước dương là T(12)=1+2+3+4+6+12=28 so sánh 28 >2.12=24

Yêu cầu: Cho 2 số nguyên L và R (0<L<R). Hãy xác định trong đoạn [L, R] có bao nhiêu số đẹp.

Dữ liệu: Vào từ file BAI1.INP gồm:

- Hai số nguyên dương L,R ($1 \le L \le R \le 1000$)

Kết quả: Ghi ra file BAI1.OUT một số nguyên là kết quả tìm được.

Ví dụ:

BAI1.INP	BAI1.OUT	Giải thích
1 20	3	Có 3 số đẹp là 12, 18, 20

Thuật toán:

- Xây dựng 1 hàm tính tổng các ước của x là T(x)

- Sau đó duyệt các giá trị x từ L đến R, Với mỗi giá trị x gọi hàm T(x) là tổng các ước của x sau đó so sánh với 2*X nếu thỏa mãn T(x)>2x thì đếm số X thỏa mãn yêu cầu đề bài.

Code chương trình :

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
int l,r;
long long ans;
long long T(int x){
   long long s=x;
      for (int i=1; i<x; i++)
         if(x%i==0) s+=i;
   return s;
}
```

```
int main()
{
    freopen("BAI1.INP", "r", stdin);
   freopen("BAI1.OUT", "w", stdout);
   ios::sync_with_stdio(false);
   cin.tie(NULL); cout.tie(NULL);
   cin>>l>>r;
   int ans=0;
   for (int x=l; x<=r; x++)
      if (T(x)>2*x) ans++;
   cout<<ans;
}
```

Bài 2. Thi đấu

Trung tâm thể thao thành phố tổ chức một giải đấu đối kháng giao lưu giữa hai đội tuyển trẻ. Để các trận đấu diễn ra cân bằng và hấp dẫn, ban tổ chức cần ghép các vận động viên của hai đội thành từng cặp thi đấu.

- Đội A có N vận động viên, vận động viên thứ i có trình độ a_i.
- Đội B có M vận động viên, vận động viên thứ j có trình độ bj.

Một trận đấu chỉ được chấp nhận nếu hai vận động viên được ghép cặp có trình độ chênh lệch nhau không quá K. Mỗi vận động viên chỉ được tham gia nhiều nhất một trận đấu.

Yêu cầu: Hãy lập trình xác định số lượng cặp thi đấu tối đa có thể được hình thành sao cho mọi cặp đều thỏa mãn điều kiện chênh lệch trình độ không quá K.

Dữ liệu: Vào từ file BAI2.INP gồm:

- Dòng đầu tiên chứa ba số nguyên dương N, M, K ($1\le N, M\le 10^3$, $0\le K\le 100$) lần lượt là số lượng các vận động viên trong đội A, B và độ chênh lệch.
- Dòng thứ hai chứa dãy số $a_1,a_2,\ldots,a_N$ ($1\le a_i\le 10^6$), trong đó a_i là trình độ vận động viên thứ i.
- Dòng thứ ba chứa dãy số b1,b2,…,b_M ($1\le b_j\le 10^6$) trong đó b_j là trình độ vận động viên thứ j.

Kết quả: Ghi ra file BAI2.OUT

Một số duy nhất là số lượng cặp đôi tối đa có thể được hình thành.

Ví dụ:

BAI2.INP	BAI2.OUT	Giải thích
4 5 1 1 4 6 2 5 1 5 7 9	3	Số cặp đôi có thể hình thành tối đa là 3 cặp đôi : (1,1); (4,5); (6,5)
4 4 3 4 2 3 6 8 9 8 10	1	Số cặp tìm được là 1 cặp đôi (6,8) hoặc (6,9)

Thuật toán:

Đây là bài toán ghép cặp tối đa với đk |ai-bj|≤k mỗi người đội A và một người đội B chỉ được dùng trong nhiều nhất 1 cặp

Để tạo được nhiều cặp nhất ta nên:

- Sắp xếp trình độ của đội A tăng dần
- Sắp xếp trình độ của đội B tăng dần

Dùng 2 con trỏ để ghép tham lam

Gọi : i là vị trí đang xét trong dãy đội A, j là vị trí đang xét trong dãy đội B. Cách xử lí

- Nếu |ai-bj| ≤k → ghép được 1 cặp tăng cả i và j khi đó ans++
- Nếu |ai-bj|>k → không ghép được cặp, có 2 khả năng xảy ra:
  + Nếu $b_j - a_i > k$ → bạn i trong đội A này quá yếu ko thể ghép với bạn j của đội B hiện tại, tìm các bạn có trình độ gần bj hơn tức là bạn đứng sau bạn thứ i của đội A →tăng i
  + Nếu $a_i - b_j > k$ → bạn thứ j của đội B này quá yếu, ko thể ghép với bạn i của đội A hiện tại, tìm các bạn có trình độ gần ai hơn tức là bạn đứng sau bạn thứ j của đội B → tăng j

Code chương trình :

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    ios::sync_with_stdio(false);
   cin.tie(nullptr);
   freopen("BAI2.INP", "r", stdin);
   freopen("BAI2.OUT", "w", stdout);
   int n, m,k;
   cin >> n>>m>>k;
   vector<int> a(n);
   for (int i = 0; i < n; i++) cin >> a[i];
   vector<int> b(m);
   for (int i = 0; i < m; i++) cin >> b[i];
   sort(a.begin(), a.end());
   sort(b.begin(), b.end());
   int l = 0, r = 0, ans = 0;
   while (l < n && r < m){
      if (abs(a[l] - b[r]) <= k){
         ans++;
         l++;
         r++;
      }
      else if ( b[r]- a[l]>k) {
         l++;
      }
      else if (a[l]-b[r]>k{
         r++;
      }
   }
   cout << ans;
}
```

Bài 3. Tính chất cân bằng

Trong một lớp học có n học sinh tham gia một bài kiểm tra. Mỗi học sinh được chấm một điểm số và được ghi lại theo danh sách. Gọi điểm của học sinh thứ i là a_i (với mọi i≠j thì $a_i \neq a_j$).

Ba học sinh i, j, k được gọi là có tính chất cân bằng nếu thỏa mãn: $a_i+a_j=2\times a_k$ (i≠j, j≠k, i≠k)

Nhiệm vụ của bạn là đếm số bộ ba học sinh thỏa mãn tính chất trên.

Yêu cầu: Cho danh sách điểm của n học sinh. Hãy xác định có bao nhiêu bộ ba học sinh có tính chất cân bằng.

Dữ liệu: vào từ file BAI3.INP gồm:

- Dòng 1: chứa số nguyên dương n, là số điểm phiếu ($2 \le n \le 10^4$)
- Dòng 2: chứa n số nguyên, số thư i là a_i-điểm của phiếu số i ($|a_i| \le 10^6$)

Kết quả Ghi ra file BAI3.OUT: Một số nguyên duy nhất là số bộ ba thỏa mãn.

Ví dụ:

BAI3.INP	BAI3.OUT	Giải thích
5 2 3 1 5 6	2	Bộ 3 số (3, 1, 2) và bộ 3 số (1, 5, 3) đều thỏa mãn tính chất trên.

Ràng buộc:

- Subtask 1 (50% số điểm): $2 \le n \le 200$;
- Subtask 2 (30% số điểm): $200 < n \le 2000$;
- Subtask 3 (20% số điểm): $2000 < n \le 10^4$.

Thuật toán:

- Sub1: Cách làm đơn giản là ba vòng for duyệt với a[i]+a[j]==2*a[k] thì đếm. với cách này độ phức tạp là $O(n^3)$

```
void sub1(){
  int ans = 0;
   for (int i = 1; i <= n; i++) {
      for (int j = 1; j <= n; j++) {
         if (i == j) continue;
         for (int k = 1; k <= n; k++) {
            if (k == i || k == j) continue;
            if (a[i] + a[j] == 2 * a[k]) {
               ans++;
            }
         }
      }
   }
   cout << ans / 2; // chia 2 vì đếm trùng (i,j) và (j,i)
}
```

Sub2,3: Với mỗi bộ ba học sinh phân biệt i, j, k cần kiểm tra: $a_i+a_j=2a_k$ tức là a_k là trung bình cộng của a_i và a_j. Do các điểm đôi một khác nhau, nên với mỗi cặp (i,j) nhiều nhất chỉ có một k tương ứng.

Ý tưởng

ta có thể duyệt mọi cặp học sinh i,j:

- Tính $s=a_i+a_j$
- Nếu s chẵn thì cần tìm xem có học sinh nào có điểm bằng s/2 hay không
- Nếu có thì ta được một bộ ba cân bằng

Lưu ý đếm trùng

Khi duyệt mọi cặp i<j, mỗi bộ ba hợp lệ chỉ được tính đúng **1 lần**:

- a_k là trung bình cộng của hai số còn lại
- nên với một bộ ba hợp lệ, chỉ có đúng một cặp (i,j) sinh ra k

Vì vậy chỉ cần đếm theo các cặp i<j.

Cách làm

- Dùng map để đánh dấu các giá trị điểm có xuất hiện hay không.
- Duyệt hai vòng lặp i<j.
- Nếu (ai+aj) chẵn và tồn tại giá trị (ai+aj)/2 trong dãy thì tăng đáp án.

```
void sub23()
{
   long long ans = 0;
   for (int i = 1; i <= n; i++) {
      for (int j = i + 1; j <= n; j++) {
         long long sum = 1LL*a[i] + a[j];
         if (sum % 2 == 0) {
            long long mid = sum / 2;
            if (mp.count(mid)) ans++;
         }
      }
   }
   cout << ans;
}
```

Code chương trình :

```
#include <bits/stdc++.h>
using namespace std;
int n,a[10005];
map<long long, int> mp;

void sub1(){
  int ans = 0;
   for (int i = 1; i <= n; i++) {
      for (int j = 1; j <= n; j++) {
         if (i == j) continue;
         for (int k = 1; k <= n; k++) {
            if (k == i || k == j) continue;
            if (a[i] + a[j] == 2 * a[k]) {
               ans++;
            }
         }
      }
   }
   cout << ans / 2; // chia 2 vì đếm trùng (i,j) và (j,i)
}
void sub23()
{
   long long ans = 0;
   for (int i = 1; i <= n; i++) {
      for (int j = i + 1; j <= n; j++) {
```

```
            long long sum = 1LL*a[i] + a[j];
            if (sum % 2 == 0) {
                long long mid = sum / 2;
                if (mp.count(mid)) ans++;
            }
        }
    }

    cout << ans;
}
int main() {
        freopen("BAI3.INP", "r", stdin);
        freopen("BAI3.OUT", "w", stdout);
     ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cin >> n;
    for (int i = 1; i <= n; i++) {
            cin >> a[i];
            mp[a[i]]=1;
    }
    if (n<=200)sub1();     else sub23();
    return 0;
}
```

Bài 4. Tấm gỗ

Tại một xưởng chế biến gỗ, người ta có N tấm gỗ được xếp thành một hàng theo thứ tự từ 1 đến N. Sau khi gia công, mỗi tấm gỗ có một chiều cao xác định, không nhất thiết bằng nhau.

Để đảm bảo tính thẩm mỹ, người thợ muốn giữ lại một số tấm gỗ sao cho khi nhìn từ đầu hàng đến cuối hàng, chiều cao các tấm gỗ còn lại thỏa mãn một trong hai dạng sau:

- Tăng nghiêm ngặt: mỗi tấm đứng sau có chiều cao lớn hơn tấm đứng trước;
- Giảm nghiêm ngặt: mỗi tấm đứng sau có chiều cao nhỏ hơn tấm đứng trước.

Người thợ được phép loại bỏ tùy ý một số tấm gỗ khỏi hàng.

Yêu cầu: Hãy xác định số lượng tấm gỗ ít nhất cần loại bỏ để dãy còn lại thỏa mãn một trong hai điều kiện trên.

Dữ liệu Vào từ file BAI4.INP:

- Dòng 1: chứa số nguyên dương N ($1 \le N \le 10^5$);
- Dòng 2: chứa N số nguyên a1, a2, ..., aN ($1 \le ai \le 10^9$), trong đó ai là chiều cao của tấm gỗ thứ i.

Kết quả Ghi ra file BAI4.OUT một số nguyên duy nhất là số tấm gỗ cần loại bỏ ít nhất.

Ví dụ:

BAI4.INP	BAI4.OUT	Giải thích
5 1 4 2 3 9	1	Giữ lại các tấm gỗ có chiều cao 1 2 3 9 còn bỏ đi tấm có chiều cao là 4
7 8 1 7 3 5 2 1	2	Giữ lại các tấm gỗ có chiều cao 8 7 5 2 1 còn bỏ đi tấm có chiều cao là 1, 3 Hoặc giữ lại 8 7 3 2 1 bỏ đi 1, 5

Ràng buộc

- Subtask 1 (40% số điểm): $N \leq 25$;
- Subtask 2 (30% số điểm): $N \leq 2000$;
- Subtask 3 (30% số điểm): $N \leq 10^5$

Sub1: có n<=25.

Sinh ra các dãy nhị phân x1x2x3…xn mỗi dãy nhị phân biểu diễn trạng thái các tấm gỗ có bị loại bỏ hay không. Trong đó xi=0 nếu tấm gỗ thứ i giữ lại, xi=1 nếu tấm gỗ thứ i được loại bỏ.

Với mỗi trạng thái của xâu x, loại bỏ những tấm gỗ ở vị trí i tương ứng với x[i]=1 rồi kiểm tra lại dãy có thỏa mãn điều kiện đề bài hay không để cập nhật kết quả bài toán

Sub2: có $N \leq 2000$;

Bài toán thực chất là dãy con tăng dài nhất, với dãy con tăng dài nhất chính là số lượng tấm gỗ giữ lại nhiều nhất và các tấm gỗ không thuộc dãy con tăng dài nhất chính là số lượng ít nhất các tấm gỗ sẽ bị loại bỏ.

Gọi f[i] , g[i]lần lượt là số lượng tấm gỗ được giữ lại nhiều nhất khi xét đến tấm gỗ thứ i theo chiều tăng , hoặc giảm

f[i]=max(f[i],f[j]+1) nếu a[j]<a[i] với i=1.. n; j=1 → i-1

g[i]=max(g[i],g[j]+1) nếu a[j]>a[i] với i=1 → n; j=1 → i-1

ans= max(f[i],g[i]) với i=1..n

Kết quả bài toán: n-ans

Code chương trình:

```
#include <bits/stdc++.h>
#define maxn 100005
using namespace std;
int n;
long long a[maxn];
int f[maxn], g[maxn];
int main()
{
   ios::sync_with_stdio(false);
   cin.tie(nullptr);
   freopen("BAI4..INP", "r", stdin);
   freopen("BAI4.OUT", "w", stdout);
   cin >> n;
   for (int i = 1; i <= n; i++) cin >> a[i];
   for (int i = 1; i <= n; i++) {
      f[i] = 1;
      for (int j = 1; j < i; j++) {
         if (a[j] < a[i])
            f[i] = max(f[i], f[j] + 1);
      }
   }
   for (int i = 1; i <= n; i++) {
      g[i] = 1;
      for (int j = 1; j < i; j++) {
         if (a[j] > a[i])
```

```
            g[i] = max(g[i], g[j] + 1);
        }
    }
    int lis = 0, lds = 0;
    for (int i = 1; i <= n; i++) {
        lis = max(lis, f[i]);
        lds = max(lds, g[i]);
    }
    cout << n - max(lis, lds);
}
```

Sub3: do $n \leq 10^5$ nên việc tính 2 vòng lặp for không hiệu quả:

Bài toán dãy con tăng dài nhất được cải tiến bằng dãy con tăng bản khó:

Ý tưởng

Ta duy trì mảng b[k], b[k]= giá trị nhỏ nhất có thể của phần tử cuối cùng của một dãy tăng độ dài kkk

Tính chất:

- b[1]<b[2]<⋯<b[L]
- Mảng b luôn tăng dần

Cách cập nhật

Duyệt từng phần tử a[i]

- Tìm vị trí: t=lower_bound(b+1,b+L+1,a[i]) -b
- Gán: b[t]=a[i]
- Cập nhật: L=max(L,t)

Nên với dãy giảm nghiêm ngặt ta sử dụng –a[i] để tính

```
            int giam = 1;
        for (int i=1;i<=n;i++) {
            int k = lower_bound(dg+1, dg+n+1, -a[i]) - dg;
            dg[k] = -a[i];
            giam = max(giam, k);
           }
```

Code chương trình :

```
#include<bits/stdc++.h>
using namespace std;
int n;int a[100005],dp[100005],dg[100005];
int main(){
      freopen("BAI4.INP","r",stdin);
      freopen("BAI4.OUT","w",stdout);
     ios::sync_with_stdio(0);    cin.tie(0);    cout.tie(0);
      cin>>n;
   for(int i=1;i<=n;i++) cin>>a[i];
   for (int i=1; i<=n; i++) dp[i]=INT_MAX,dg[i]=INT_MAX;
   int tang = 1;
   for (int i=1;i<=n;i++) {
      int k = lower_bound(dp+1, dp+n+1, a[i]) - dp;
      dp[k] = a[i];
```

```
        tang = max(tang, k);
    }
    int giam = 1;
    for (int i=1;i<=n;i++) {
        int k = lower_bound(dg+1, dg+n+1, -a[i]) - dg;
        dg[k] = -a[i];
        giam = max(giam, k);
    }
    cout<<n-max(tang,giam);
}
```

Bài 5. Trạm nghỉ

Trên một tuyến đường cao tốc có n vị trí có thể xây dựng trạm nghỉ, vị trí thứ i nằm tại tọa độ a_i trên trục đường thẳng. Do hạn chế về kinh phí, ban quản lý quyết định loại bỏ đúng **k** vị trí và chỉ giữ lại các vị trí còn lại để xây trạm nghỉ.

Yêu cầu đặt ra là sau khi loại bỏ k vị trí, khoảng cách nhỏ nhất giữa hai trạm nghỉ bất kỳ còn lại phải lớn nhất có thể**.**

Dữ liệu: Vào từ file BAI5.INP gồm:

- Dòng đầu chứa hai số nguyên dương n, k ($k \le n-2$).
- Dòng thứ hai chứa n số nguyên $a_1, a_2, \ldots, a_n$ ($|ai| \le 10^9$).

Kết quả: Ghi ra file BAI5.OUT

- In ra một số nguyên: giá trị lớn nhất của khoảng cách nhỏ nhất giữa hai trạm nghỉ bất kỳ còn lại.

Ví dụ:

BAI5.INP	BAI5.OUT	Giải thích
5 1 4 1 2 3 9	1	Test 1: Xóa 1 phần tử bất kỳ, thì dãy còn lại luôn tồn tại 2 số tự nhiên liên tiếp nhau, nên độ chênh lệch lớn nhất là 1.
5 2 10 -5 3 -2 1	7	Test 2: Trong các cách xóa 2 phần tử bất kỳ, cách xóa chỉ còn 3 phần tử (10, -5, 3) sẽ có độ chênh lệch nhỏ nhất là 7. Cách xóa này là cách xóa có độ chênh lệch nhỏ nhất giữa các phần tử là lớn nhất.

Ràng buộc:

- Subtask 1 (20% số điểm): $n \le 20, k=1$;
- Subtask 2 (30% số điểm): $20 < n \le 100$;
- Subtask 3 (25% số điểm): $100 < n \le 2000$;
- Subtask 4 (25% số điểm): $2000 < n \le 10^5$.

Thuật toán

Sub1: k = 1

Ta phải xóa đúng 1 phần tử**,** sau đó:

- Tính chênh lệch nhỏ nhất giữa mọi cặp phần tử còn lại
- Lấy giá trị lớn nhất có thể

Vì $n \le 20$, ta có thể:

1. Thử xóa từng phần tử i (tổng cộng n cách)
2. Với mỗi cách:
    - Lấy dãy còn lại (n-1 phần tử)

- Tính chênh lệch nhỏ nhất giữa mọi cặp

3. Lấy giá trị lớn nhất trong các phương án

Sau khi xóa 1 phần tử:

- Sắp xếp dãy còn lại
- Chỉ cần kiểm tra các cặp kề nhau
- Lấy min(a[i+1] - a[i])

Vì trong dãy đã sort, khoảng cách nhỏ nhất luôn nằm giữa hai phần tử kề nhau.

Nhận xét: Xóa k phần tử ⇔ giữ lại m=n−k phần tử (và m≥2 vì k≤n−2).

Ta muốn: max(min|bi−bj|) i≠j

với b là dãy còn lại.

Nếu ta sắp xếp dãy tăng dần: a1≤a2≤⋯≤an.

Khi đó, với một tập các phần tử đã chọn theo thứ tự, chênh lệch nhỏ nhất giữa mọi cặp chính là chênh lệch nhỏ nhất giữa các phần tử kề nhau trong tập đã chọn.

Vì khoảng cách giữa hai phần tử xa hơn luôn ≥ min của các khoảng kề.

⇒ Bài toán: Chọn m phần tử trong dãy đã sort sao cho khoảng cách tối thiểu giữa hai phần tử được chọn liên tiếp là lớn nhất. dùng backtracking để thử tất cả các khả năng, các phần tử không bị xóa sẽ lưu vào mảng b, sau đó tìm khoảng cách mina là khoảng cách nhỏ nhất của các phần tử liên tiếp trong b, trong các khoảng cách chọn max(mina)

```
long long ans = -1e18;
void xuli(){
   long long mn = 1e18;
   for(int i=1;i<b.size();i++){
      mn = min(mn, b[i] - b[i-1]);
   }
   ans = max(ans,mn);
}
void  backtrack(int start){
   if(b.size() == n-k){
      xuli();
      return;
   }
   for(int i = start; i <= n; i++){
      b.push_back(a[i]);
      backtrack(i+1);
      b.pop_back();
   }
}
void sub1()
{
   backtrack(1);
   cout << ans;
}
int main(){
   freopen("BAI5.INP", "r", stdin);
   freopen("BAI5.OUT", "w", stdout);
```

```
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cin >> n >> k;
    for(int i=1;i<=n;i++)    cin >> a[i];
    sort(a+1,a+n+1);
    if (k==1) sub1();
  }
```

Sub 2, 3: n≤2000

Cần xóa k phần tử tức là giữ lại đúng m=n-k phần tử

sau khi sắp xếp dãy tăng dần, giả sử ta chọn ra: $b_1<b_2<...<b_m$

thì giá trị cần tối ưu là: $\min(b_2-b_1, b_3-b_2, \ldots, b_m-b_{m-1})$

tức là : Trong dãy được giữ lại ta nhìn các khoảng cách giữa hai phần tử liên tiếp, lấy khoảng cách nhỏ nhất, rồi cố làm cho giá trị nhỏ nhất đó lớn nhất có thể.

B1: Sắp xếp dữ liệu : vì chỉ quan tâm đến chênh lệch giữa các giá trị, ta sort dãy $a_1 \le a_2 \le \ldots \le a_n$

Sau đó chỉ việc chọn 1 dãy con tăng theo chỉ số.

B2: Quy hoạch động trạng thái

+ Gọi dp[i][t] là giá trị lớn nhất của chênh lệch nhỏ nhất khi ta chọn đúng t phần tử và phần tử cuối cùng được chọn là ai.
+ Trường hợp cơ sở: Nếu mới chọn đúng 1 phần tử thì chưa có cặp nào cả nên chưa có khoảng cách nhỏ nhất thực sự : khởi tạo dp[i][1]=INF;
+ Khi xét phần tử trước đó là a[j] với j<i
+ Nếu trước đó ta đã chọn t-1 phần tử và kết thúc ở a[j], rồi thêm a[i] vào thì:
  - Khoảng cách mới tạo ra là a[i]-a[j]
  - Chênh lệch nhỏ nhất của cả dãy mới sẽ là: min(dp[j][t-1],a[i]-a[j])

  Do đó dp[i][t]=max{min(dp[j][t-1],a[i]-a[j]) } với 1≤j≤i
+ Kết quả cần tìm là max{dp[i][m]} với i=1..n; m=n-k

```
#include <bits/stdc++.h>
#define ll long long
#define maxn 2005
using namespace std;
const ll INF = (ll)4e18;
int n, m,k;
ll a[maxn];
ll dp[maxn][maxn];
void sub23(){
    for (int i = 1; i <= n; i++) dp[i][1] = INF;
  for (int t = 2; t <= m; t++) {
    for (int i = 1; i <= n; i++) {
      dp[i][t] = -1;
      for (int j = 1; j < i; j++) {
        if (dp[j][t - 1] == -1) continue;
        ll val = min(dp[j][t - 1], a[i] - a[j]);
        dp[i][t] = max(dp[i][t], val);
      }
    }
  }
```

```
    ll ans = 0;
    for (int i = 1; i <= n; i++) {
        ans = max(ans, dp[i][m]);
    }
    cout << ans << '\n';
}
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
   freopen("BAI5.INP", "r", stdin);
   freopen("BAI5.OUT", "w", stdout);
    cin >> n >> k;
    for (int i = 1; i <= n; i++) cin >> a[i];
    sort(a+1, a+n+1);
    m = n - k; // số phần tử cần giữ lại
    if (n<=2000) sub23();
    return 0;
}
```

Sub 4: $n \leq 10^5$

Xóa đúng k phần tử, sau khi xóa còn lại m=n-k phần tử. Cần làm sao để min(ai-aj) đạt lớn nhất.

Nhận xét: sắp xếp dãy tăng dần

Nếu ta chọn một số phần tử trong dãy đã sắp xếp thì khoảng cách nhỏ nhất giữa mọi cặp phần tử được chọn sẽ chính là khoảng cách nhỏ nhất giữa hai phần tử kề nhau trong tập được chọn. nghĩa là cần đảm bảo ai_{t+1}-ai_t>=d

Thuật toán:

+ B1: sắp xếp dãy số tăng dần
+ B2: Chặt nhị phân d trong đoạn [0, a[n]-a[1]]
+ B3: Với mỗi d kiểm tra tham lam xem có chọn được ít nhất n-k phần tử không

Code full điểm

```
#include <bits/stdc++.h>
using namespace std;
int n, k;
long long a[100005];
bool check(long long d) {
    int dem = 1;
    long long last = a[1
    for (int i = 2; i <= n; i++) {
        if (a[i] - last >= d) {
            dem++;
            last = a[i];
        }
    }
    if (dem >= n - k) return true; else return false
}
int main() {
```

```
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cin >> n >> k;
    for (int i = 1; i <= n; i++) cin >> a[i];
    sort(a + 1, a + n + 1);
    long long left = 0;
   long long right = a[n] - a[1];
   long long ans = 0;
    while (left <= right) {
        long long mid = (left + right) / 2;
        if (check(mid)) {
            ans = mid;
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    cout << ans;
    return 0;
}
```