Lời giải đề thi HSG 9, TP.HCM năm 2025

14/03/2025

Lưu ý:

- Đề bài đã được mình tóm tắt rất ngắn gọn, dù không chép y nguyên đề bài gốc nhưng tất cả các yêu cầu của đề bài thì mình giữ nguyên!

- Lời giải này là lời giải không chính thức, được nghĩ ra bởi mình ngay trong phòng thi, và là do mình tự viết nên là **KHÔNG ĐẢM BẢO** chính xác 100% và có thể **KHÔNG** AC được bài! Các bạn chỉ nên tham khảo thôi nha :)

- **Nếu bạn không muốn tự gây áp lực lên bản thân sau kì thi thì mình khuyên bạn tắt file này đi và hãy dành cho mình một khoảng thời gian yên tĩnh, thả lỏng tâm hồn và tối được ngủ ngon. Chứ nếu đọc mà phát hiện ra bạn sai nhảm chỗ nào đó thì coi như 1 tuần sau mất ngủ đó :))**

Bài 1: SAPXEP.cpp

Tóm tắt: cho một dãy a gồm n phần tử, ta thực hiện sắp xếp nổi bọt (bubble sort) lên dãy đó. Hãy đếm xem có bao nhiêu cặp phần tử bị đổi chỗ trong quá trình sắp xếp.

Giới hạn: $1 \leq n \leq 2.10^5, 1 \leq a_i \leq 10^9$

- **Vét cạn:**
  Để vét cạn thì tương đối dễ, ta chỉ cần thực hiện bubble sort xong đếm xem có bao nhiêu cặp vị trí i, j thỏa $a_i > a_j$ là được.
  Code:

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

// An optimized version of Bubble Sort
int bubbleSort(vector<int>& arr) {
    int n = arr.size();
    bool swapped;

    long long ans=0;
    for (int i = 0; i < n - 1; i++) {
        swapped = false;
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                swap(arr[j], arr[j + 1]);
                swapped = true;

                ++ans;
            }
        }

        // If no two elements were swapped, then break
        if (!swapped)
            break;
    }

    return ans;
}

int main() {
    ios_base::sync_with_stdio(0);cin.tie(0);
    int n;
    cin>>n;

    vector<int> arr(n);
    for(int i=0;i<n;++i){
        cin>>arr[i];
    }

    cout<<bubbleSort(arr)<<'\n';
    return 0;
}
```

(Lúc viết lời giải thì do mình khá lười nên để cho nhanh thì mình đã lấy code từ https://www.geeksforgeeks.org/bubble-sort-algorithm/ rồi chỉnh sửa lại để giải bài này)

- **Tối ưu:**
  Lúc ở trong phòng thi thì mình không nghĩ ra được lời giải nào cả nên đành phải dùng đến CTDL nâng cao là Fenwick tree. Sau khi code xong mình thấy nó đúng nên nộp luôn bằng cách đó (mình không có thời gian để nghĩ cách khác). Nên là ai có ý tưởng nào đơn giản hơn thì hãy comment xuống bên dưới nhé ;)

Nhận thấy rằng khi sắp xếp xong thì các phần tử được sắp xếp tăng dần, nghĩa là nếu ta đang ở vị trí i thì trong các phần tử $a_{i+1}, a_{i+2}, \dots, a_n$, số lần hoán đổi mà có phần tử a_i sẽ chính là số phần tử trong đó mà bé hơn a_i. Nghĩa là với mỗi a_i mình sẽ đếm xem có bao nhiêu vị trí j thỏa mãn $a_i > a_j$ và $i < j \leq n$. Trong lúc thi thì mình nghĩ ngược lại, giả sử phần tử ta đang xét hiện tại là a_j, thì ta sẽ đếm xem có bao nhiêu i thỏa mãn $a_j < a_i$ và $1 \leq i < j$. Để làm điều đó thì ta sẽ dùng mảng đánh dấu: gọi $mark_k$ là số lượng phần tử trong mảng a có giá trị bằng k. Lúc này số phần tử bé hơn a_i sẽ là $mark_1 + mark_2 + \cdots + mark_{a_i-1}$. Sau đó ta sẽ cập nhật lại giá trị của $mark_{a_i} = mark_{a_i} + 1$. Dễ thấy đây chính là bài toán ứng dụng cơ bản của Fenwick tree. Tuy nhiên do a_i có thể lên tới 10^9 mà n chỉ lên tới 2.10^5 nên ta cần phải nén mảng lại để giữ nguyên độ lớn của các phần tử.

Code:

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;

const int MAXN=2e5;
int a[MAXN+5],n;

int temp[MAXN+5];
void compress(){
    for(int i=1;i<=n;++i){
        temp[i]=a[i];
    }

    sort(temp+1,temp+n+1);
    for(int i=1;i<=n;++i){
        a[i]=lower_bound(temp+1,temp+n+1,a[i])-temp;
    }
}

int BIT[MAXN+5];
void update(int idx,int value){
    while(idx<=n){
        BIT[idx]+=value;
        idx+=idx&(-idx);
    }
}

int get(int idx){
    int res=0;
    while(idx>0){
        res+=BIT[idx];
        idx-=idx&(-idx);
    }
    return res;
}

void solve(){
    int ans=0;
    for(int i=1;i<=n;++i){
        ans+=get(n)-get(a[i]);
        update(a[i],1);
    }

    cout<<ans<<'\n';
}

signed main(){
    ios_base::sync_with_stdio(0);cin.tie(0);
    cin>>n;
    for(int i=1;i<=n;++i){
        cin>>a[i];
    }

    compress();
    solve();
    return 0;
}
```

Bài 2: KHUVUC.cpp

Tóm tắt: cho một dãy a có n phần tử. Hãy thay đổi giá trị của 1 phần tử bất kì sao cho ƯCLN của dãy sau khi thay đổi là lớn nhất.

Giới hạn: $1 \leq n \leq 10^5, 1 \leq a_i \leq 10^9$

- **Vét cạn:**
  Nhận thấy rằng việc thay đổi phần tử a_i tương đương với việc loại bỏ phần tử đó ra khỏi dãy, sau đó thêm một phần tử khác vào. Mà phần tử khác đó khi thêm vào thì để ƯCLN còn lại lớn nhất thì nó phải là bội của ƯCLN của $n-1$ phần tử còn lại sau khi xóa phần tử cũ đi. Vậy ta đưa về bài toán: loại bỏ 1 phần tử sao cho ƯCLN của $n-1$ phần tử còn lại của dãy là lớn nhất. Để vét cạn thì cũng khá đơn giản, đó chính là tại mỗi vị trí ta thử loại bỏ phần tử đó đi và tính ƯCLN của dãy.
  Code:

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;

const int MAXN=1e5;
vector<int>a;
int n;

int gcd(vector<int>arr,int pop_idx){
    arr.erase(begin(arr)+pop_idx);
    if(arr.empty())return 0;

    int res=arr[0];
    for(int element:arr){
        res=__gcd(res,element);
    }
    return res;
}

void solve(){
    int ans=0;
    for(int i=0;i<n;++i){
        ans=max(ans,gcd(a,i));
    }
    cout<<ans<<'\n';
}

signed main(){
    ios_base::sync_with_stdio(0);cin.tie(0);
    cin>>n;
    a.resize(n);
    for(int i=0;i<n;++i){
        cin>>a[i];
    }

    solve();
    return 0;
}
```

- **Tối ưu:**
  Gọi $left_i$ là $\gcd(a_1, a_2, \dots, a_i)$, và $right_i$ là $\gcd(a_i, a_{i+1}, \dots, a_n)$
  Dễ thấy rằng sau khi loại bỏ phần tử tại vị trí i thì ƯCLN của các phần tử còn lại là $\gcd(left_{i-1}, right_{i+1})$. Vậy ta chỉ cần 1 vòng lặp duyệt qua dãy và cập nhật giá trị lớn nhất là được.
  Code:

```cpp
#include<bits/stdc++.h>
#define left sussy
#define right baka
#define int long long
using namespace std;

const int MAXN=1e5;
int a[MAXN+5],n,left[MAXN+5],right[MAXN+5];

void compute(){
    left[1]=a[1];
    for(int i=2;i<=n;++i){
        left[i]=__gcd(left[i-1],a[i]);
    }

    right[n]=a[n];
    for(int i=n-1;i>=1;--i){
        right[i]=__gcd(right[i+1],a[i]);
    }
}

void solve(){
    compute();
    int ans=0;
    for(int i=1;i<=n;++i){
        ans=max(ans,__gcd(left[i-1],right[i+1]));
    }
    cout<<ans<<'\n';
}

signed main(){
    ios_base::sync_with_stdio(0);cin.tie(0);
    cin>>n;
    for(int i=1;i<=n;++i){
        cin>>a[i];
    }

    solve();
    return 0;
}
```

(bạn nào thấy mình define left và right thành sussy baka mà không hiểu tại sao thì đơn giản là do C++ có hàm std::left() và std::right() nên nếu mình gọi left[i-1] và right[i+1] thì chương trình sẽ không hiểu ý mình gọi hàm std hay là mảng nên sẽ báo lỗi, mà mình thấy đổi lại thì nó khó hiểu hơn nên là mình define luôn cho lẹ :D)

Bài 3: GIAIDAU.cpp

Tóm tắt: cho một dãy a có n phần tử và q truy vấn. Mỗi truy vấn cho 2 số u, v $(u < v)$. Với mỗi truy vấn hãy tìm cách chia đoạn con gồm các phần tử $a_u, a_{u+1}, \dots a_v$ thành 2 phần liên tiếp sao cho chênh lệch giữa tổng 2 phần là nhỏ nhất. Nghĩa là chọn vị trí i $(u \leq i < v)$ sao cho $|(a_u + a_{u+1} + \cdots + a_i) - (a_{i+1} + a_{i+2} + \cdots + a_v)|$ là nhỏ nhất.

Giới hạn: $1 \leq n, q \leq 10^5, 1 \leq a_i \leq 10^9$

- **Vét cạn:**
  Để vét cạn thì rất đơn giản, ta chạy i từ u tới $v-1$ và lấy min của
  $$|(a_u + a_{u+1} + \cdots + a_i) - (a_{i+1} + a_{i+2} + \cdots + a_v)|$$
  là được. Để tính nhanh tổng 1 đoạn thì ta sử dụng prefix sum.
  Code:

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;

const int MAXN=1e5;
int a[MAXN+5],n,q,prefix[MAXN+5];

void solve(){
    int u,v;
    cin>>u>>v;

    int ans=1e18;
    for(int i=u;i<v;++i){
        ans=min(ans,llabs((prefix[v]-prefix[i])-(prefix[i]-prefix[u-1])));
    }
    cout<<ans<<'\n';
}

signed main(){
    ios_base::sync_with_stdio(0);cin.tie(0);
    cin>>n>>q;
    for(int i=1;i<=n;++i){
        cin>>a[i];
        prefix[i]=prefix[i-1]+a[i];
    }

    while(q--){
        solve();
    }
    return 0;
}
```

- **Tối ưu:**

  Ta có: $a_u + a_{u+1} + \cdots + a_i = prefix_i - prefix_{u-1}$

  và $a_{i+1} + a_{i+2} + \cdots + a_v = prefix_v - prefix_i$

  Viết lại biểu thức, ta cần tìm giá trị nhỏ nhất của $|M|$ với:

$$M = (prefix_v - prefix_i) - (prefix_i - prefix_{u-1})$$
$$= prefix_{u-1} + prefix_v - 2.prefix_i$$

  TH1:

$$M \geq 0$$

  Dễ thấy để $|M|$ bé nhất, mà $prefix_{u-1} + prefix_v$ là cố định nên là $2.prefix_i$ phải lớn nhất!

  ⇨ $prefix_{u-1} + prefix_v \geq 2.prefix_i$

  ⇨ $prefix_i \leq \frac{prefix_{u-1}+prefix_v}{2}$

  Vậy M bé nhất khi $prefix_i$ lớn dần và càng tiến gần tới $\frac{prefix_{u-1}+prefix_v}{2}$.

  ⇨ Để tìm được i thì ta thực hiện tìm kiếm nhị phân vị trí i lớn nhất thỏa mãn

$$prefix_i \leq \frac{prefix_{u-1} + prefix_v}{2}$$

  Do $a_i \geq 1$ nên $prefix_{i-1} < prefix_i$, nghĩa là dãy $prefix$ đã được sắp xếp tăng dần sẵn nên ta có thể thực hiện tìm kiếm nhị phân.

  TH2:

$$M < 0$$

  Dễ thấy lúc này $|M| = 2.prefix_i - (prefix_{u-1} + prefix_v)$, mà $prefix_{u-1} + prefix_v$ là cố định nên để $|M|$ bé nhất thì là $2.prefix_i$ phải bé nhất!

  ⇨ $prefix_{u-1} + prefix_v < 2.prefix_i$

  ⇨ $prefix_i > \frac{prefix_{u-1}+prefix_v}{2}$

  Vậy M bé nhất khi $prefix_i$ nhỏ dần và càng tiến gần tới $\frac{prefix_{u-1}+prefix_v}{2}$.

  ⇨ Để tìm được i thì ta thực hiện tìm kiếm nhị phân vị trí i nhỏ nhất thỏa mãn

$$prefix_i > \frac{prefix_{u-1} + prefix_v}{2}$$

  Để tránh sử dụng 2 lần tìm kiếm nhị phân thì nhận thấy rằng giá trị ở 2 TH để có chung vế phải là $\frac{prefix_{u-1}+prefix_v}{2}$, chỉ khác mỗi dấu là $\leq$ và $>$. Nhận thấy rằng 2 dấu này ngược nhau nên nếu tìm được i bé nhất thỏa:

$$prefix_i > \frac{prefix_{u-1} + prefix_v}{2}$$

  thì ta phải tìm giá trị của i' lớn nhất thỏa:

$$prefix_{i'} \leq \frac{prefix_{u-1} + prefix_v}{2}$$

  ⇨ $i' = i - 1$ (Do i là giá trị nhỏ nhất thỏa $>$ thì $i - 1$ sẽ là giá trị lớn nhất thỏa dấu $\leq$)

Sau đó ta lấy giá trị nhỏ nhất của 2 trường hợp là được.

Code:

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;

const int MAXN=1e5;
int a[MAXN+5],n,q,prefix[MAXN+5];

void solve(){
    int u,v;
    cin>>u>>v;

    int mid=(prefix[u-1]+prefix[v])/2;
    int idx=upper_bound(prefix+u,prefix+v+1,mid)-prefix;

    int ans1=llabs(prefix[u-1]+prefix[v]-2*prefix[idx]),
        ans2=llabs(prefix[u-1]+prefix[v]-2*prefix[idx-1]);
    cout<<min(ans1,ans2)<<'\n';
}

signed main(){
    ios_base::sync_with_stdio(0);cin.tie(0);
    cin>>n>>q;
    for(int i=1;i<=n;++i){
        cin>>a[i];
        prefix[i]=prefix[i-1]+a[i];
    }

    while(q--){
        solve();
    }
    return 0;
}
```

Lời kết: nếu bạn đã đọc đến đây thì mình xin cảm ơn vì đã dành thời gian ra đọc lời giải của mình :). Lời giải này là tất cả những gì mình đã nghĩ ra được trong phòng thi và về nhà chỉ viết lại và chia sẻ ra thôi! Do vậy có thể không đúng, nên là mình nghĩ rằng nó chỉ để tham khảo, đồng thời mình cũng muốn chia sẻ cách của mình với mọi người khác. Nếu lời giải có sai sót gì thì hãy bình luận vào bên dưới. Và mình xin chúc tất cả các bạn trong kì thi đạt được kết quả mình mong muốn :D

bye bye!