# OLP khối không chuyên 2021

## Bài 1: Mã hóa

Đề bài: Olympic Sinh Viên 2021 - Không chuyên - Mã hóa

Sol: Bài này thì dễ quá nên không giải thích nữa

```
#include <bits/stdc++.h>

using namespace std;

int main(){
    ios :: sync_with_stdio(false);
        cin.tie(0); cout.tie(0);
    long long r,l,a,k;
    cin>>l>>r>>a>>k;
    k/=(__gcd(k,a));
    long long n = r/k - l/k;
    if(l%k==0)
        n++;
    cout<<n;
}
```

## Bài 2: Khoảng cách

Đề bài: Olympic Sinh Viên 2021 - Không chuyên - Khoảng cách

Sol:

Đầu tiên cần tính khoảng cách nhỏ nhất của mỗi mảng con 1 phần của A đến B. Với mỗi $a_i \in A$ thì ta chặt nhị phân để tìm ra giá trị trong B mà gần A nhất, lúc này ta sẽ tìm được khoảng cách từ mỗi phần tử trong A đến B.

Sau đó với mỗi đoạn [L:R] thì khoảng cách nhỏ nhất sẽ là min của khoảng cách nhỏ nhất của từng $a_i \in A, i = L, .., R$ .Quá trình này có thể được tăng tốc bằng cách sử dụng Segment tree như code dưới đây.

Vậy

```
#include <bits/stdc++.h>

using namespace std;

template<class T> struct Segment {
    const T ID = 1e18;
    int n;
    vector<T> seg;
    void init(int _n){
        n = _n; seg.assign(2*n,ID);
    }
    T comb(T a, T b){
        return min(a,b);
    }
    void pull(int p){
        seg[p] = comb(seg[2*p],seg[2*p+1]);
    }
    void upd(int p, T val) {
        seg[p += n] = val;
        for (p /= 2; p; p /= 2)
            pull(p);
    }
    T query(int l, int r) {
        T ra = ID, rb = ID;
        for (l += n, r += n+1; l < r; l /= 2, r /= 2) {
            if (l&1) ra = comb(ra,seg[l++]);
            if (r&1) rb = comb(seg[--r],rb);
        }
        return comb(ra,rb);
    }
};

int main(){
    ios :: sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    int n,m,k;
```

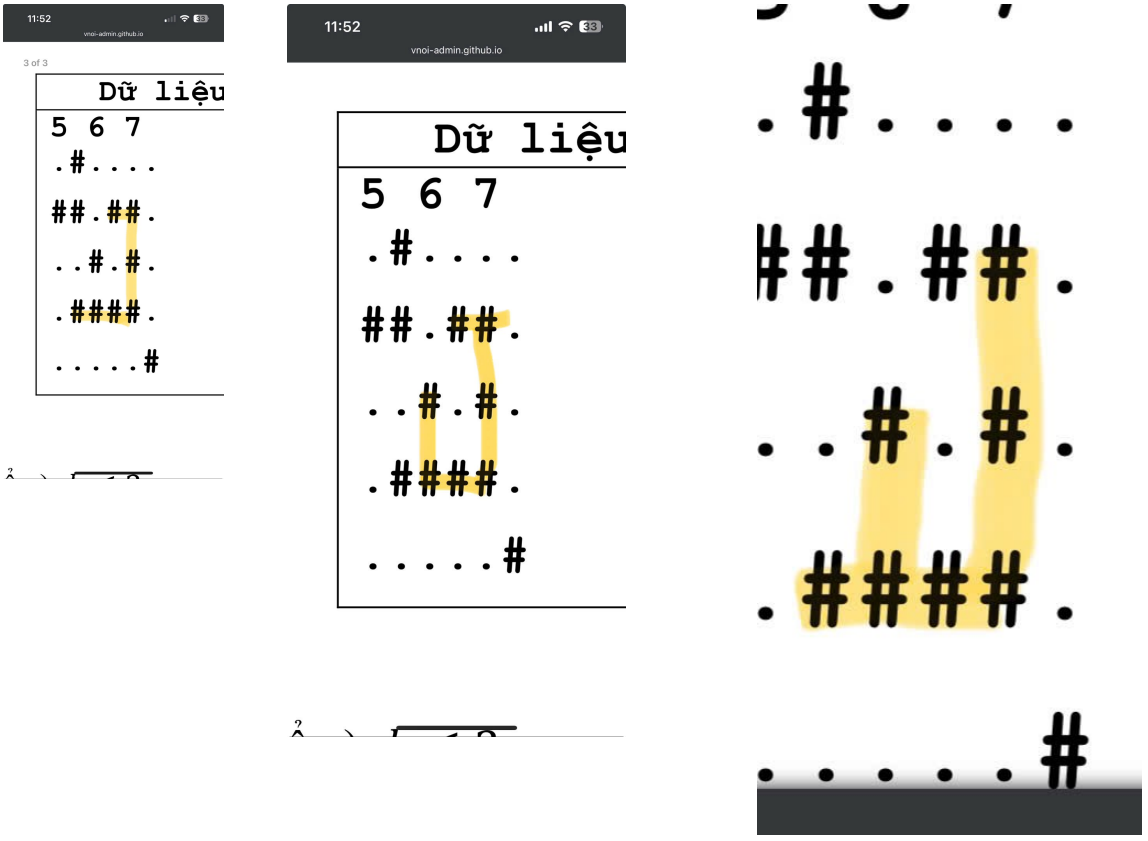
```
    cin>>m>>n>>k;
    vector<long long>a(m),b(n);
    for(int i=0;i<m;i++){
        cin>>a[i];
    }
    for(int i=0;i<n;i++){
        cin>>b[i];
    }
    Segment<long long>seg;
    seg.init(m);
    sort(b.begin(),b.end());
    for(int i=0;i<m;i++){
        auto j = upper_bound(b.begin(),b.end(),a[i]);
        long long x=min(abs(*j-a[i]), abs(*(j-1)-a[i]));
        seg.upd(i+1, x);
    }
    while(k--){
        int l,r;
        cin>>l>>r;
        cout<<seg.query(l,r)<<"\n";

    }
}
```

# Bài 3: Khóa bảng

Đề bài: Olympic Sinh Viên 2021 - Không chuyên - Khóa bảng

Bài này thì yêu cầu có thể giải thích là tìm tập các tập k điểm sao cho hai điểm bất kì trong 1 tập luôn tồn tại 1 đường đi từ đỉnh này đến đỉnh kia và ngược lại.

Đối với test mẫu, chúng ta cố 3 đoạn được tô sau đây:




Đối cách cách giải tối ưu, thì đầu tiên cần phải sinh ra toàn bộ các khối có đúng k điểm. Sau đó việc cần làm là so khớp các khối này với từng vị trí trong ma trận đầu vào.

Code thì bài này mình chưa AC nên chưa có code đâu :D tuy nhiên sol trên là sol chuẩn đấy.

# Bài 4: Trò chơi

Đề bài: Olympic Sinh Viên 2021 - Không chuyên - Trò chơi

Để thỏa mãn đề bài, giả sử biểu diễn giá trị b ở từng vị trí lên một biểu đồ đường thì hình hợp bởi trục x, trục y và đường nối các điểm giá trị b sẽ tạo thành 1 hình thoi. Nếu duyệt trâu bằng cách xét từng vị trí đỉnh của hình thoi nằm ở vị trí thứ i thì sẽ ăn được 32/52 test và TLE các test còn lại.

Để ăn được hết test thì cần sử dụng QHĐ kết hợp với CTDL Stack như sau:

- cần tìm dãy b tăng dần/giảm dần và thỏa mãn b[i] <= a[i]
- Nếu xét tăng dần thì ta gọi $dp[i]$ là tổng b lớn nhất khi xét i vị trí đầu tiên, ta có:

$dp[i] = a[i] * (i - j) + dp[j]$ với j là vị trí gần i nhất sao cho $a[j] <= a[i]$

- Quá trình này cần xét theo hai chiều sử dụng 2 mảng dp khác nhau, kết quả cuối cùng sẽ là max của tổng giá trị $dp1[i] + dp2[i] - a[i]$
- Giá trị j có thể được tăng tốc sử dụng stack như sau:

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int n;
    cin>>n;
    long long a[n+2];
    for(int i=1;i<=n;i++){
        cin>>a[i];
    }
    a[0]=0;
    a[n+1]=0;
    stack<long long>x;
    x.push(0ll);
    vector<long long>dp1(n+2,0);
    vector<long long>dp2(n+2,0);
    for(int i=1;i<=n;i++){
        while(a[i]<a[x.top()]&&x.top()!=0){
            x.pop();
        }
        dp1[i]+=dp1[x.top()]+a[i]*(i-x.top());
        x.push(i);
    }
    while (!x.empty()) {
        x.pop();
    }
    x.push(n+1);
    long long ans=0;
    for(int i=n;i>0;i--){
        while(a[i]<a[x.top()]&&x.top()!=n+1){
            x.pop();
        }
        dp2[i]+=dp2[x.top()]+a[i]*(x.top()-i);
```

```
            ans=max(ans, dp1[i]+dp2[i]-a[i]);
            x.push(i);
        }
        cout<<ans;
}
```