### ƯỚC CHUNG LỚN NHẤT

Thí sinh có thể dễ dàng đạt 50% số điểm của bài này khi sử dụng thuật toán vét cạn với độ phức tạp .

Đoạn chương trình tham khảo

typedef long long ll;

ll a[maxN+1];

int n, k;

ll GCD(ll a, ll b)

{

while (b > 0)

{

ll c = a % b;

a = b;

b = c;

}

return a;

}

void Solve()

{

ll ans = 1;

for (int i = 1; i + k - 1 <= n; ++i)

{

ll v = 0;

for (int j = 1; j <= k; ++j)

v = GCD(v, a[i+j-1]);

ans = max(ans, v);

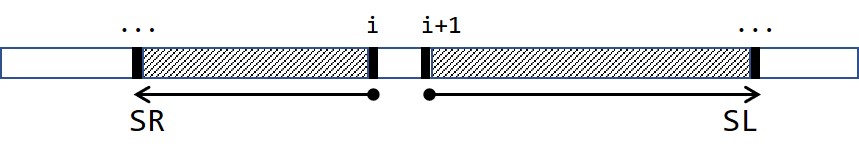
}

cout<<ans;

}

Để cải tiến thuật toán, thí sinh cần nắm được cách tìm ước chung lớn nhất của một dãy các phần tử.

Nhận xét: Trong cách làm trên, khi tính GCD của đoạn tiếp theo thì các đoạn con bên trong bị tính lại nhiều lần. Để khắc phục, ta tổ chức stack SL lưu ước chung lớn nhất của các đoạn con từ trái qua và stack SR lưu ước chung lớn nhất của các đoạn con từ phải qua:



Việc thêm phần tử vào 2 stack phải luôn đảm bảo tổng số phần tử trong 2 stack đúng bằng . Khi đó . Để tính GCD cho đoạn tiếp theo , ta chỉ cần hủy phần tử đầu ra khỏi stack SR và thêm phần tử mới vào đầu stack SL.

Đoạn chương trình cải tiến tham khảo:

typedef pair<ll, ll> pii;

stack<pii> SL, SR;

void Init()

{

SL.push(pii(0, 0));

SR.push(pii(0, 0));

for (int i = 1; i <= k; ++i)

{

ll u = SL.top().second;

u = GCD(u, a[i]);

SL.push(pii(a[i], u));

}

}

void Solve()

{

ll ans = SL.top().second;

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

{

ll u = SL.top().second;

u = GCD(u, a[i]);

SL.push(pii(a[i], u));

if (SR.size() == 1)

{

while (SL.size() > 1)

{

ll u = SL.top().first;

SL.pop();

ll v = GCD(u, SR.top().second);

SR.push(pii(u, v));

}

}

SR.pop();

ans = max(ans, GCD(SL.top().second, SR.top().second));

}

cout<<ans;

}

Trong cách cài đặt trên, mỗi đoạn con trong đoạn gồm phần tử được xét lại tối đa 2 lần. Do đó thuật toán có độ phức tạp .