

# Team notebook

November 26, 2018

## Contents

1	BinaryLifting	1
2	Chancellor Programming Contest 2018	1
2.1	Problem B . . . . .	1
2.1.1	solution . . . . .	1
2.2	Problem F . . . . .	2
2.2.1	solution . . . . .	2
3	Diophantine	2
4	GCD	3
5	LCA	3
6	LISegmentTree	4
7	Max-flow-Ford-Fulkerson	5
8	MergeSort	6
9	geometry	6
10	phieuler	7
11	trie	7

## 1 BinaryLifting

```
int n, l;  
vector<vector<int>> adj;
```

```
int timer;  
vector<int> tin, tout;  
vector<vector<int>> up;  
  
void dfs(int v, int p)  
{  
    tin[v] = ++timer;  
    up[v][0] = p;  
    for (int i = 1; i <= l; ++i)  
        up[v][i] = up[up[v][i-1]][i-1];  
  
    for (int u : adj[v]) {  
        if (u != p)  
            dfs(u, v);  
    }  
  
    tout[v] = ++timer;  
}  
  
bool is_ancestor(int u, int v)  
{  
    return tin[u] <= tin[v] && tout[u] >= tout[v];  
}  
  
int lca(int u, int v)  
{  
    if (is_ancestor(u, v))  
        return u;  
    if (is_ancestor(v, u))  
        return v;  
    for (int i = l; i >= 0; --i) {  
        if (!is_ancestor(up[u][i], v))  
            u = up[u][i];  
    }
```

```

    }
    return up[u][0];
}

void preprocess(int root) {
    tin.resize(n);
    tout.resize(n);
    timer = 0;
    l = ceil(log2(n));
    up.assign(n, vector<int>(l + 1));
    dfs(root, root);
}

```

## 2 Chancellor Programming Contest 2018

### 2.1 Problem B

#### 2.1.1 solution

```

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

int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL);
    vector<char> v({'G','B','R'});
    long long n;
    while(cin >> n){
        n--;
        stack<char> pila;
        while(n>0){
            n--;
            pila.push(v[n%3]);
            n/=3;
        }
        while(!pila.empty()){
            cout << pila.top();
            pila.pop();
        }
        cout << "S\n";
    }
    return 0;
}

```

## 2.2 Problem F

### 2.2.1 solution

```

#include <bits/stdc++.h>
using namespace std;
const int tam = 100000000;

int GCD(int a, int b){
    return (b==0)? a : GCD(b, a%b);
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL);
    vector<bool> criba(tam, true);
    criba[0]=criba[1]=false;
    for(int i=4; i<tam; i+=2){
        criba[i] =false;
    }
    for(int i=3; i*i<=tam; i+=2){
        if(criba[i])
            for(int j=i*i; j<tam; j+=i){
                criba[j]=false;
            }
    }
    int t, a, b;
    cin >> t;
    while(t--){
        cin >> a >> b;
        if(GCD(a,b)!=1)
            cout << "unknown\n";
        else{
            int i=0, cont=0;
            while(cont<10){
                int aux = a*i+b;
                if(criba[aux]){
                    cont++;
                    cout << aux << ((cont==10)? '\n' : ' ');
                }
                i++;
            }
        }
    }
    return 0;
}

```

---

```
}

```

### 3 Diophantine

---

```
int gcd(int a, int b, int &x, int &y) {
    if (a == 0) {
        x = 0; y = 1;
        return b;
    }
    int x1, y1;
    int d = gcd(b%a, a, x1, y1);
    x = y1 - (b / a) * x1;
    y = x1;
    return d;
}

bool find_any_solution(int a, int b, int c, int &x0, int &y0, int &g) {
    g = gcd(abs(a), abs(b), x0, y0);
    if (c % g) {
        return false;
    }

    x0 *= c / g;
    y0 *= c / g;
    if (a < 0) x0 = -x0;
    if (b < 0) y0 = -y0;
    return true;
}

void shift_solution (int & x, int & y, int a, int b, int cnt) {
    x += cnt * b;
    y -= cnt * a;
}

int find_all_solutions (int a, int b, int c, int minx, int maxx, int
    miny, int maxy) {
    int x, y, g;
    if (! find_any_solution (a, b, c, x, y, g))
        return 0;
    a /= g; b /= g;

    int sign_a = a>0 ? +1 : -1;

```

```
int sign_b = b>0 ? +1 : -1;

shift_solution (x, y, a, b, (minx - x) / b);
if (x < minx)
    shift_solution (x, y, a, b, sign_b);
if (x > maxx)
    return 0;
int lx1 = x;

shift_solution (x, y, a, b, (maxx - x) / b);
if (x > maxx)
    shift_solution (x, y, a, b, -sign_b);
int rx1 = x;

shift_solution (x, y, a, b, - (miny - y) / a);
if (y < miny)
    shift_solution (x, y, a, b, -sign_a);
if (y > maxy)
    return 0;
int lx2 = x;

shift_solution (x, y, a, b, - (maxy - y) / a);
if (y > maxy)
    shift_solution (x, y, a, b, sign_a);
int rx2 = x;

if (lx2 > rx2)
    swap (lx2, rx2);
int lx = max (lx1, lx2);
int rx = min (rx1, rx2);

if (lx > rx) return 0;
return (rx - lx) / abs(b) + 1;
}

```

---

### 4 GCD

---

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

int GCD(int a, int b){
    return b? GCD(b, a%b) : a;
}

```

```

}

int main(){
    ios_base::sync_with_stdio(false);cin.tie(NULL);
    int a, b;
    cin >> a >> b;
    cout << GCD(a,b);
    return 0;
}

```

---

## 5 LCA

```

struct LCA {
    vector<int> height, euler, first, segtree;
    vector<bool> visited;
    int n;

    LCA(vector<vector<int>> &adj, int root = 0) {
        n = adj.size();
        height.resize(n);
        first.resize(n);
        euler.reserve(n * 2);
        visited.assign(n, false);
        dfs(adj, root);
        int m = euler.size();
        segtree.resize(m * 4);
        build(1, 0, m - 1);
    }

    void dfs(vector<vector<int>> &adj, int node, int h = 0) {
        visited[node] = true;
        height[node] = h;
        first[node] = euler.size();
        euler.push_back(node);
        for (auto to : adj[node]) {
            if (!visited[to]) {
                dfs(adj, to, h + 1);
                euler.push_back(node);
            }
        }
    }
}

```

```

void build(int node, int b, int e) {
    if (b == e) {
        segtree[node] = euler[b];
    } else {
        int mid = (b + e) / 2;
        build(node << 1, b, mid);
        build(node << 1 | 1, mid + 1, e);
        int l = segtree[node << 1], r = segtree[node << 1 | 1];
        segtree[node] = (height[l] < height[r]) ? l : r;
    }
}

int query(int node, int b, int e, int L, int R) {
    if (b > R || e < L)
        return -1;
    if (b >= L && e <= R)
        return segtree[node];
    int mid = (b + e) >> 1;

    int left = query(node << 1, b, mid, L, R);
    int right = query(node << 1 | 1, mid + 1, e, L, R);
    if (left == -1) return right;
    if (right == -1) return left;
    return height[left] < height[right] ? left : right;
}

int lca(int u, int v) {
    int left = first[u], right = first[v];
    if (left > right)
        swap(left, right);
    return query(1, 0, euler.size() - 1, left, right);
}
};

```

---

## 6 LISegmentTree

```

#include <bits/stdc++.h>
using namespace std;
typedef vector<int> vi;

class SegmentTree{
private:

```

```

vi A, st;
int n;
int right(int i){ return (i<<1)+1;}
int left(int i){ return (i<<1);}
void build(int p, int L, int R){
    if(L==R) st[p]=R;
    else{
        int mid = (L+R)/2;
        build(left(p), L, mid);
        build(right(p), mid+1, R);
        int x = st[left(p)], y = st[right(p)];
        st[p] = (A[x] >= A[y])? x : y;
    }
}

int rmq(int p, int i, int j, int L, int R){
    if(R<i || j<L) return -1;
    if(i<=L && R<=j) return st[p];
    int mid=(L+R)>>1;
    int x=rmq(left(p), i, j, L, mid),
        y = rmq(right(p), i, j, mid+1, R);
    if(x!=-1) return y;
    if(y!=-1) return x;
    return (A[x]>=A[y])? x : y;
}

void update(int p, int i, int value, int L, int R){
    if(i<L || R<i) return;
    if(i==L && R==i){ A[i]++;}
    else{
        int mid = (L+R)/2;
        update(left(p), i, value, L, mid);
        update(right(p), i, value, mid+1, R);
        int x = st[left(p)], y = st[right(p)];
        st[p] = (A[x] >= A[y])? x : y;
    }
}

public:
SegmentTree(int a){
    n=a;
    A.assign(n+1, 0);
    st.assign(4*(n+1), 0);
    build(1, 0, n);
}

```

```

int rmq(int i, int j){
    return A[rmq(1, i, j, 0, n)];
}

void update(int i, int value){
    update(1, i, value, 0, n);
}

};

int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL);
    int n, mx=0;
    cin >> n;
    vector<int> v(n);
    for(int i=0; i<n; i++){ cin >> v[i]; mx = max(mx, v[i]);}
    SegmentTree st(mx);
    vector<int> d(n,0);
    for(int i=0; i<n; i++){
        d[i]=st.rmq(0, v[i]-1);
    }
    cout << st.rmq(0, mx) << '\n';
    return 0;
}

```

## 7 Max-flow-Ford-Fulkerson

```

int n;
vector<vector<int>> capacity;
vector<vector<int>> adj;

int bfs(int s, int t, vector<int>& parent) {
    fill(parent.begin(), parent.end(), -1);
    parent[s] = -2;
    queue<pair<int, int>> q;
    q.push({s, INF});

    while (!q.empty()) {
        int cur = q.front().first;
        int flow = q.front().second;
        q.pop();
    }
}

```

```

    for (int next : adj[cur]) {
        if (parent[next] == -1 && capacity[cur][next]) {
            parent[next] = cur;
            int new_flow = min(flow, capacity[cur][next]);
            if (next == t)
                return new_flow;
            q.push({next, new_flow});
        }
    }
}

return 0;
}

int maxflow(int s, int t) {
    int flow = 0;
    vector<int> parent(n);
    int new_flow;

    while (new_flow = bfs(s, t, parent)) {
        flow += new_flow;
        int cur = t;
        while (cur != s) {
            int prev = parent[cur];
            capacity[prev][cur] -= new_flow;
            capacity[cur][prev] += new_flow;
            cur = prev;
        }
    }

    return flow;
}

```

---

## 8 MergeSort

```

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

struct mergeSort{
    vector<int> g;
    mergeSort(vector<int> &v){

```

```

        g = v;
        merge(0, (int)g.size()/2, (int)g.size()-1);
        for(int to : g) cout << to << ' ';
    }

    void merge(int p, int q, int r){
        int a=p, b=q+1;
        if(p==r) return;
        merge(p, (p+q)/2, q);
        merge(q+1, (q+1+r)/2, r);
        int i=0, n = r-p+1;
        vector<int> aux(n);
        while(i<n){
            if(a>q){aux[i]=g[b++]; i++; continue;}
            if(b>r){aux[i]=g[a++]; i++; continue;}
            if(g[a]<g[b])aux[i]=g[a++];
            else aux[i]=g[b++];
            i++;
        }
        for(int i=0; i<n; i++)g[i+p]=aux[i];
    }

};

int main(){
    vector<int> v;
    for(int i=100; i>=0; i--)v.push_back(i);
    mergeSort a(v);
    return 0;
}

```

---

## 9 geometry

```

struct point{
    long double x, y;
    point(){};
    point(long double x_, long double y_): x(x_), y(y_){}
    point operator + (const point & other) const { return point(x +
        other.x, y + other.y); }
    point operator - (const point & other) const { return point(x -
        other.x, y - other.y); }
}

```

```

    point operator * (long double & n) const { return point(x * n, y *
n); }
    point operator / (long double & n) const { return point(x / n, y /
n); }
};

void centerThreePoints(const point & a, const point & b, const point & c,
point & center, long double & r){
    long double A = a.x*a.x + a.y * a.y;
    long double B = b.x*b.x + b.y * b.y;
    long double C = c.x*c.x + c.y * c.y;
    long double D = a.x * (b.y - c.y) - b.x * (a.y - c.y) + c.x * (a.y
- b.y);
    center.x = (A * (b.y - c.y) - B * (a.y - c.y) + C * (a.y -
b.y))/(2.0*D);
    center.y = (a.x * (B - C) - b.x * (A - C) + c.x * (A -
B))/(2.0*D);
    r = sqrt((a.x - rx) * (a.x - rx) + (a.y - ry) * (a.y - ry));
}

point rotation(const point & p, const point & center, long double theta){
    point ans;
    ans.x = (p.x - center.x) * cos(theta) - (p.y - center.y) *
sin(theta) + center.x;
    ans.y = (p.y - center.y) * cos(theta) + (p.x - center.x) *
sin(theta) + center.y;
    return ans;
}

```

## 10 phieuler

```

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

int phiEuler(int n){
    int result=n;
    for(int i=2; i*i<=n; i++){
        if(n%i==0){
            while(n%i==0)n/=i;
            result-=result/i;
        }
    }
}

```

```

    }
    if(n>1)result-=result/n;
    return result;
}

int main(){
    ios_base::sync_with_stdio(false);cin.tie(NULL);
    int n;
    cin >> n;
    cout << phiEuler(n);
    return 0;
}

```

## 11 trie

```

struct trie{
    map<char,trie> m;
    int end, freq;
    trie(){end=0; freq=0;}
    void add(const string &s, int p=0){
        if((int)s.size()>p) m[s[p]].add(s, p+1);
        else{ end = 1; freq++;}
    }
    void dfs(int carry){
        int sum = ((int)m.size()>1 || end);
        for(auto &to : m)
            to.second.dfs(carry+sum);
        if(end){cont+=freq*carry;}
    }
};

```