Олимпиада СПбГУ по информатике 2018/19 учебного года

Литвинский Семен Алексеевич

A	B	C	D	E	F	Sum
100	100	100	60	12	7	379

Task A (100)

```
#include <iostream>
#include <vector>

using namespace std;
/*
int mRand() {
    return (rand() << 15) + rand();
}

struct node {
    int x, y, sz = 0;
    node* l = nullptr, r = nullptr;
    node(int x):x(x), y(mRand()), sz(1){}
};

typedef *node Pn

int sz(Pn rt){return rt ? rt->sz : 0;}

void upd(Pn& rt){if(rt)rt->sz = sz(rt->l) + sz(rt->r);}

void mer(Pn& rt, Pn l, Pn r){
    if(!l || !r) return void(rt = l ? l : r);
    if(l->y > r->y){mer(l->r, l->r, r); rt = l; upd(rt);}
    else{mer(r->l, l, r->l); rt = r; upd(rt);}
}

void spl(Pn rt, Pn& l, Pn& r){
    if(!rt) return void(l = r = nullptr);
}
*/
int main()
{
    //srand(3032019);
    //cin.tie(0);
    //cout.tie(0);
    long long a, b;
    cin >> a >> b;
    while(a < b) {
        a *= 2;
    }
    if(b == a) {
        cout << "Yes\n";
    } else {
        cout << "No\n";
    }
    return 0;
}
```

Task B (100)

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>

using namespace std;
/*
int mRand() {
    return (rand() << 15) + rand();
}

struct node {
    int x, y, sz = 0;
    node* l = nullptr, r = nullptr;
    node(int x):x(x), y(mRand()), sz(1){}
};

typedef *node Pn

int sz(Pn rt){return rt ? rt->sz : 0;}

void upd(Pn& rt){if(rt)rt->sz = sz(rt->l) + sz(rt->r);}

void mer(Pn& rt, Pn l, Pn r){
    if(!l || !r) return void(rt = l ? l : r);
    if(l->y > r->y){mer(l->r, l->r, r); rt = l; upd(rt);}
    else{mer(r->l, l, r->l); rt = r; upd(rt);}
}

void spl(Pn rt, Pn& l, Pn& r){
    if(!rt) return void(l = r = nullptr);
}
*/
int main()
{
    //srand(3032019);
    //cin.tie(0);
    //cout.tie(0);
    string s;
    int n;
    cin >> n;
    cin >> s;
    if(n == 1) {
        cout << "No\n";
        return 0;
    }
    bool is = false;
    for(int i = 0; i < n - 1; i++) {
        if(s[i] == 'o' && s[i + 1] == 'r') is = 1;
    }
    for(int i = 0; i < n - 2; i++) {
        if(s[i] == 'o' && s[i + 2] == 'r') is = 1;
    }
    reverse(s.begin(), s.end());
    for(int i = 0; i < n - 1; i++) {
        if(s[i] == 'o' && s[i + 1] == 'r') is = 1;
    }
    if(is) {
        cout << "Yes\n";
    } else {
        cout << "No\n";
    }
    return 0;
}
```

Task C (100)

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>

using namespace std;
/*
int mRand() {
    return (rand() << 15) + rand();
}

struct node {
    int x, y, sz = 0;
    node* l = nullptr, r = nullptr;
    node(int x):x(x), y(mRand()), sz(1){}
};

typedef *node Pn

int sz(Pn rt){return rt ? rt->sz : 0;}

void upd(Pn& rt){if(rt)rt->sz = sz(rt->l) + sz(rt->r);}

void mer(Pn& rt, Pn l, Pn r){
    if(!l || !r) return void(rt = l ? l : r);
    if(l->y > r->y){mer(l->r, l->r, r); rt = l; upd(rt);}
    else{mer(r->l, l, r->l); rt = r; upd(rt);}
}

void spl(Pn rt, Pn& l, Pn& r){
    if(!rt) return void(l = r = nullptr);
}
*/
vector<int> G[100010];
pair<int, int> Es[100010];
pair<int, int> dp[100010];
int ans[100010];
int n;

int getDp(int i, int v){
    return Es[i].first == v ? dp[i].second : dp[i].first;
}

int get(int i, int v){
    return Es[i].first == v ? Es[i].second : Es[i].first;
}

int dfs(int v, int e) {
    int cnt = 1;
    for(auto i : G[v]) {
        if(i != e) {
            cnt += dfs(get(i, v), i);
        }
    }
    if(e != -1 && v == Es[e].first) {
        dp[e].first = cnt;
        dp[e].second = n - cnt;
    } else {
        dp[e].first = n - cnt;
        dp[e].second = cnt;
    }
    return cnt;
}
void getAns(int v) {
    ans[v] = 0;
    for(auto i : G[v]) {
        ans[v] = max(ans[v], getDp(i, v));
    }
}
```

```
int main()
{
    //srand(3032019);
    //cin.tie(0);
    //cout.tie(0);
    cin >> n;
    for(int i = 0; i < n - 1; i++) {
        cin >> Es[i].first >> Es[i].second;
        Es[i].first--;
        Es[i].second--;
        G[Es[i].first].push_back(i);
        G[Es[i].second ].push_back(i);
    }
    dfs(0, -1);
    for(int i = 0; i < n; i++) {
        getAns(i);
    }
    for(int i = 0; i < n; i++) {
        cout << ans[i] + 1 << '␣';
    }
    cout << '\n';
    return 0;
}
```

Task D (60)

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>

using namespace std;
/*
int mRand() {
    return (rand() << 15) + rand();
}

struct node {
    int x, y, sz = 0;
    node* l = nullptr, r = nullptr;
    node(int x):x(x), y(mRand()), sz(1){}
};

typedef *node Pn

int sz(Pn rt){return rt ? rt->sz : 0;}

void upd(Pn& rt){if(rt)rt->sz = sz(rt->l) + sz(rt->r);}

void mer(Pn& rt, Pn l, Pn r){
    if(!l || !r) return void(rt = l ? l : r);
    if(l->y > r->y){mer(l->r, l->r, r); rt = l; upd(rt);}
    else{mer(r->l, l, r->l); rt = r; upd(rt);}
}

void spl(Pn rt, Pn& l, Pn& r){
    if(!rt) return void(l = r = nullptr);
}
*/
vector<int> G[100010];
pair<int, int> Es[100010];
pair<int, int> dp[100010];
int ans[100010];
int n;

int getDp(int i, int v){
    return Es[i].first == v ? dp[i].second : dp[i].first;
}

int get(int i, int v){
    return Es[i].first == v ? Es[i].second : Es[i].first;
}

int dfs(int v, int e) {
    int cnt = 1;
    for(auto i : G[v]) {
        if(i != e) {
            cnt += dfs(get(i, v), i);
        }
    }
    if(e != -1 && v == Es[e].first) {
        dp[e].first = cnt;
        dp[e].second = n - cnt;
    } else {
        dp[e].first = n - cnt;
        dp[e].second = cnt;
    }
    return cnt;
}
void getAns(int v) {
    ans[v] = 0;
    for(auto i : G[v]) {
        ans[v] = max(ans[v], getDp(i, v));
    }
}
```

```
int main()
{
    string s, ans1, ans2, ans3, q1, q2, q3, res = "?????????";
    cin >> s;
    bool is = s[0] == 's';
    int T, n, p;
    cin >> T >> n >> p;
    for(int t = 0; t < T; t++) {
        if(is) {
            cin >> s;
            cout << "a" << s[0] << s[1] << s[2] << s[3] << s[4] << s[5] << '␣';
            cout << "b" << s[3] << s[4] << s[5] << s[6] << s[7] << s[8] << '␣';
            cout << "c" << s[0] << s[1] << s[2] << s[6] << s[7] << s[8] << '␣';
            if(n == 5) {
                cout << "a" << s[0] << s[1] << s[2] << s[3] << s[4] << s[5] << '␣';
                cout << "b" << s[3] << s[4] << s[5] << s[6] << s[7] << s[8] << '␣';
            }
            cout << '\n';
        } else {
            cin >> q1 >> q2;
            if(n == 5) {
                cin >> q3;
            }
            if(q1[0] == 'a'){res[0] = q1[1]; res[1] = q1[2]; res[2] = q1[3]; res[3] = q1[4], res
                [4] = q1[5]; res[5] = q1[6];}
            if(q1[0] == 'b'){res[3] = q1[1]; res[4] = q1[2]; res[5] = q1[3]; res[6] = q1[4], res
                [7] = q1[5]; res[8] = q1[6];}
            if(q1[0] == 'c'){res[0] = q1[1]; res[1] = q1[2]; res[2] = q1[3]; res[6] = q1[4], res
                [7] = q1[5]; res[8] = q1[6];}
            if(q2[0] == 'a'){res[0] = q2[1]; res[1] = q2[2]; res[2] = q2[3]; res[3] = q2[4], res
                [4] = q2[5]; res[5] = q2[6];}
            if(q2[0] == 'b'){res[3] = q2[1]; res[4] = q2[2]; res[5] = q2[3]; res[6] = q2[4], res
                [7] = q2[5]; res[8] = q2[6];}
            if(q2[0] == 'c'){res[0] = q2[1]; res[1] = q2[2]; res[2] = q2[3]; res[6] = q2[4], res
                [7] = q2[5]; res[8] = q2[6];}
            if(n == 5) {
            if(q3[0] == 'a'){res[0] = q3[1]; res[1] = q3[2]; res[2] = q3[3]; res[3] = q3[4], res
                [4] = q3[5]; res[5] = q3[6];}
            if(q3[0] == 'b'){res[3] = q3[1]; res[4] = q3[2]; res[5] = q3[3]; res[6] = q3[4], res
                [7] = q3[5]; res[8] = q3[6];}
            if(q3[0] == 'c'){res[0] = q3[1]; res[1] = q3[2]; res[2] = q3[3]; res[6] = q3[4], res
                [7] = q3[5]; res[8] = q3[6];}
            }
            cout << res << '\n';
        }
    }
    return 0;
}
```

Task E (12)

```cpp
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <cmath>

using namespace std;
/*
int mRand() {
    return (rand() << 15) + rand();
}

struct node {
    int x, y, sz = 0;
    node* l = nullptr, r = nullptr;
    node(int x):x(x), y(mRand()), sz(1){}
};

typedef *node Pn

int sz(Pn rt){return rt ? rt->sz : 0;}

void upd(Pn& rt){if(rt)rt->sz = sz(rt->l) + sz(rt->r);}

void mer(Pn& rt, Pn l, Pn r){
    if(!l || !r) return void(rt = l ? l : r);
    if(l->y > r->y){mer(l->r, l->r, r); rt = l; upd(rt);}
    else{mer(r->l, l, r->l); rt = r; upd(rt);}
}

void spl(Pn rt, Pn& l, Pn& r){
    if(!rt) return void(l = r = nullptr);
}
*/

struct P {
    long long x, y;
};

long long dist(P& a, P&b) {
    return (a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y);
}


int main()
{
    //srand(3032019);
    //cin.tie(0);
    //cout.tie(0);

    int n;
    cin >> n;
    vector<P> arr(n);
    for(int i = 0; i < n; i++) {
        cin >> arr[i].x >> arr[i].y;
    }
    P p, q, vec;
    cin >> p.x >> p.y >> q.x >> q.y;
    vec.x = 1000000000;
    vec.y = 0;
    int ii = 0;
    bool is = false;
    for(int i = 1; i < n; i++) {
        if(dist(vec, arr[i]) < dist(vec, arr[ii])) {
            ii = i;
        }
    }
    for(int i = 0; i < n; i++) {
        if(i != ii && dist(vec, arr[i]) == dist(vec, arr[ii])){
            is = true;
        }
    }
    if(is) {
        cout << -1;
    } else {
        cout << ii + 1;
    }
    return 0;
}
```

Task F (7)

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>

using namespace std;

int a[30];
int b[30];

int main()
{
    int n, k;
    cin >> n >> k;
    int ans = 0;
    for(int i = 0; i < n; i++) {
        cin >> a[i]  >> b[i];
    }
    for(int mask = 0; mask < (1<<n); mask++) {
        int blue = 0, red = 0, res = 0;
        for(int i = 0; i < n; i++) {
            if((1<<i) & mask) {
                blue += k;
            } else {
                red += k;
            }
            res += min(red, a[i]) + min(blue, b[i]);
            red -= min(red, a[i]);
            blue -= min(blue, b[i]);
        }
        ans = max(ans, res);
    }
    cout << ans;
    return 0;
}
```