Contents

Ad hoc

Go up for ultras

```cpp
#include <cstdio>
#include <cmath>
#include <cstring>
#include <algorithm>

using namespace std;
int v[100005];
int seg[4000005];
int seg2[4000005];

int esquerda(int p){
    return 2 * p;
}

int direita(int p){
    return 2 * p + 1;
}

int query(int p, int i, int j, int L, int R){
    if(j < L || i > R) return 1000000000;
    if(i <= L && j >= R) return seg[p];
    int meio = (L + R) / 2;
    return min(query(esquerda(p), i, j, L, meio),
        query(direita(p), i, j, meio + 1, R));
}

int findL(int p, int i, int L, int R, int v){
    if(i <= L) return -1;
    if(v >= seg2[p]) return -1;
    if(L == R) return L;

    int meio = (L + R) / 2;
    int ret = -1;
    if(i > meio+1 && seg2[direita(p)] > v)
        ret = findL(direita(p), i, meio+1, R, v);
    if(ret != -1) return ret;
    return findL(esquerda(p), i, L, meio, v);
}

int findR(int p, int i, int L, int R, int v){
    if(i > R) return -1;
    if(v >= seg2[p]) return -1;
    if(L == R) return L;

    int meio = (L + R) / 2;
    int ret = -1;
```

```cpp
    if(i < meio && seg2[esquerda(p)] > v)
        ret = findR(esquerda(p), i, L, meio, v);
    if(ret != -1) return ret;
    return findR(direita(p), i, meio+1, R, v);
}
int constructMax(int L, int R, int p){
    if(L == R) return seg2[p] = v[L];
    int mid = (L + R) / 2;
    return seg2[p] = max(constructMax(L, mid, esquerda(p)),
        constructMax(mid+1, R, direita(p)));
}

int constructMin(int L, int R, int p){
    if(L == R) return seg[p] = v[L];
    int mid = (L + R) / 2;
    return seg[p] = min(constructMin(L, mid, esquerda(p)),
        constructMin(mid+1, R, direita(p)));
}

int main(){
    int n;

    while(scanf("%d", &n) == 1){
        memset(seg, 0, sizeof(seg));
        memset(seg2, 0, sizeof(seg2));

        for(int i = 0; i < n; i++)
            scanf("%d", v + i);

        constructMax(0, n-1, 1);
        constructMin(0, n-1, 1);

        bool first = true, ultra, left, right;
        for(int i = 0; i < n; i++){
            ultra = false;
            if((i == 0 || v[i] > v[i-1]) && (i == n-1 || v[i] > v[i+1])){
                int id = findL(1, i, 0, n-1, v[i]);
                int id2 = findR(1, i, 0, n-1, v[i]);
                if(id == -1 && id2 == -1){
                    if(v[i] >= 150000)
                        ultra = true;
                }
                else{
                    left = false;
                    if(id == -1) left = true;
                    else{
                        int d = query(1, id+1, i-1, 0, n-1);
                        if(v[i] - d >= 150000) left = true;
                    }
                    right = false;
                    if(id2 == -1) right = true;
                    else{
```

```
                int d = query(1, i+1, id2-1, 0, n-1);
                if(v[i] - d >= 150000) right = true;
            }
            if(left && right){
                ultra = true;
            }
        }
        if(ultra){
            if(!first) printf(" ");
            printf("%d", i+1);

            first = false;
        }
    }
}
printf("\n");
}

    return 0;
}
```

Estrutura de Dados

Consultas Horríveis

```c
#include <stdio.h>
#include <string.h>

long long st[800004];
long long troca[800004];

void update(int p, int ini, int fim, int b, int e, long long valor){
    st[p] += (troca[p] * (fim - ini + 1));
    troca[(p << 1)] += troca[p];
    troca[(p << 1) + 1] += troca[p];
    troca[p] = 0;

    if(b > fim || e < ini) return;

    st[p] += ((fim < e ? fim : e) - (ini > b ? ini : b) + 1) * valor;

    if(b <= ini && e >= fim){
        troca[(p << 1)] += valor;
        troca[(p << 1) + 1] += valor;
        return;
    }

    int meio = (ini + fim) >> 1;
```

```c
    if(b <= meio) update((p << 1), ini, meio, b, e, valor);
    if(e > meio) update((p << 1) + 1, meio+1, fim, b, e, valor);
}

long long query(int p, int ini, int fim, int b, int e){
    st[p] += (troca[p] * (fim - ini + 1));
    troca[(p << 1)] += troca[p];
    troca[(p << 1) + 1] += troca[p];

    troca[p] = 0;

    if(b > fim || e < ini) return 0;
    if(b <= ini && e >= fim) return st[p];

    int meio = (ini + fim) >> 1;

    return query((p << 1), ini, meio, b, e) + query((p << 1) + 1, meio+1, fim, b, e);
}

int main(){
    int t, n, c, i, p, q, v;

    scanf("%d", &t);
    while(t--){
        scanf(" %d %d", &n, &c);

        memset(st, 0, sizeof(st));
        memset(troca, 0, sizeof(troca));
        while(c--){
            scanf(" %d %d %d", &i, &p, &q);
            if(!i){
                scanf(" %d", &v);
                update(1, 0, n-1, p-1, q-1, v);
            }
            else printf("%lld\n", query(1, 0, n-1, p-1, q-1));
        }
    }

    return 0;
}
```

Paradigmas

Garota Hiperativa

```cpp
#include <cstdio>
#include <cstring>
#include <utility>
#include <algorithm>
```

3

```cpp
#define MOD (100000000)
using namespace std;

int n, m, dp[104][104];
pair <int, int> activity[105];

int f(int prev, int next){
    if(dp[prev][next] != -1) return dp[prev][next];

    if(activity[next].second == m) return 1;

    int aux = 0;
    for(int i = next+1; i < n; i++)
        if(activity[i].first > activity[next].first &&
            activity[i].first <= activity[next].second &&
            activity[i].second > activity[next].second &&
            (prev == next || activity[i].first > activity[prev].second))
            aux = (aux + f(next, i)) % MOD;

    return dp[prev][next] = aux;
}

int main(){

    while(scanf("%d %d", &m, &n) == 2 && n + m){
        for(int i = 0; i < n; i++)
            scanf("%d %d", &activity[i].first, &activity[i].second);

        memset(dp, -1, sizeof dp);
        sort(activity, activity+n);

        int ans = 0;
        for(int i = 0; i < n && activity[i].first == 0; i++)
            ans = (ans + f(i, i)) % MOD;

        printf("%d\n", ans);
    }

    return 0;
}
```

Hard Problem

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

#define oo 1000000000000000000

using namespace std;

string s[100005];
string r[100005];
```

```cpp
int n, c[100005];

long long dp[100005][2];
bool mark[100005][2];

long long f(int x, int rev){
    if(mark[x][rev]) return dp[x][rev];
    mark[x][rev] = 1;

    if(x == n) return 0;

    if(x == 0) return min(c[x] + f(x+1, 1), f(x+1, 0));

    bool canforward = false, canreverse = false;
    long long ans = oo;

    if(rev){
        if(s[x] >= r[x-1]) canforward = true;
        if(r[x] >= r[x-1]) canreverse = true;
    }
    else{
        if(s[x] >= s[x-1]) canforward = true;
        if(r[x] >= s[x-1]) canreverse = true;
    }

    if(canforward) ans = min(ans, f(x+1, 0));
    if(canreverse) ans = min(ans, c[x]+f(x+1, 1));

    return dp[x][rev] = ans;
}

int main(){

    scanf("%d", &n);

    for(int i = 0; i < n; i++)
        scanf("%d", c+i);

    for(int i = 0; i < n; i++){
        cin >> s[i];
        r[i] = s[i];
        reverse(r[i].begin(), r[i].end());
    }

    long long ret = f(0, 0);
    printf("%lld\n", ret == oo ? -1 : ret);

    return 0;
}
```

Matemática

Quantos zeros e quantos digitos?

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

using namespace std;

int main(){
    vector<pair<int,int>> v(1000);

    for(int i = 1; i < 1000; i++){
        pair<int, int> last = {1, 1};
        int tmp = i;
        for(int j = 2; j*j <= tmp; j++){
            if(tmp % j == 0){
                last = {j, 0};
                while(tmp % j == 0){
                    tmp /= j;
                    last.second++;
                }
            }
        }
        if(tmp > 1){
            last = {tmp, 1};
        }
        v[i] = last;
    }

    int n, b;

    while(scanf("%d_%d", &n, &b) == 2){
        double tmp1 = 0;
        for(int i = 1; i <= n; i++)
            tmp1 += log(i);
        int digits = 1 + (int)(tmp1 / log(b) + 1e-9);

        int tmp = v[b].first, count = 0;
        while(tmp <= n){
            count += n / tmp;
            tmp *= v[b].first;
        }

        int zeros = count / v[b].second;

        printf("%d_%d\n", zeros, digits);
    }

    return 0;
}
```

Grafos

Floid Fill

Lazy Painting

```cpp
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream>
#include <algorithm>
#define ff first
#define ss second

using namespace std;

vector<vector<bool> > mat;
vector<vector<int> > prox;

int n, m, h, w, q, r, c, total;

int dx[] = {-1, 0, 0, 1};
int dy[] = {0, 1, -1, 0};

void dfs(int x, int y){
    if(mat[x][y] == true) return;
    total--;
    mat[x][y] = true;

    int a, b;
    for(int i = 0; i < 4; i++){
        a = dx[i] + x;
        b = dy[i] + y;
        if(a < r || a >= r+h || b < c || b >= c+w) continue;
        dfs(a, b);
    }
}


int main(){

    scanf("%d_%d_%d_%d_%d", &n, &m, &h, &w, &q);
    mat.assign(n, vector<bool> (m, false));
    prox.assign(n, vector<int> (m, 1));

    total = n*m;
```

5

```
        for(int i = 0; i < q; i++){
                scanf("%d_%d", &r, &c); r--; c--;
                for(int j = r; j < r+h; ){
                        dfs(j, c);
                        int tmp = prox[j][c];
                        prox[j][c] = max(prox[j][c], r+h - j);
                        j += tmp;
                }
                printf("%d\n", total);
        }


        return 0;
}
```

Colonia LCA

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

#define mp make_pair
#define ff first
#define ss second

using namespace std;

typedef long long ll;

vector<pair<int,int>> G[100005];
int lvl[100005], p[100005], anc[100005][20], logg;
ll dist[100005];

void dfs(int v, int l, ll d){
    lvl[v] = l;
    dist[v] = d;
    for(pair<int,int> &x : G[v]){
        if(lvl[x.ff] == -1){
            p[x.ff] = v;
            dfs(x.ff, l+1,d+x.ss);
        }
    }
}

int LCA(int u, int v){
    if(lvl[u] < lvl[v]) swap(u,v);

    for(int i = logg; i >= 0; i--)
        if(lvl[u] - (1<<i) >= lvl[v])
            u = anc[u][i];

    if(u == v) return u;
```

```
    for(int i = logg; i >= 0; i--){
        if(anc[u][i] != -1 && anc[u][i] != anc[v][i]){
            u = anc[u][i];
            v = anc[v][i];
        }
    }

    return anc[u][0];
}

int main() {
    int a,b,n,q,i,j;

    while(scanf("%d", &n),n){
        for(int i = 1; i <= n-1; i++){
            scanf("%d%d", &a,&b);
            G[i].push_back(mp(a,b));
            G[a].push_back(mp(i,b));
            lvl[i] = -1;
        }
        dfs(0,0,0);
        for(logg = 0; 1<<logg <= n; logg++);
        for(i = 0; i < n; i++)
            for(j = 0; j < logg; j++)
                anc[i][j] = -1;
        for(i = 0; i < n; i++)
            anc[i][0] = p[i];
        for(j = 1; j <= logg; j++)
            for(i = 0; i < n; i++)
                if(anc[i][j-1] != -1)
                    anc[i][j] = anc[ anc[i][j-1] ][j-1];
        scanf("%d", &q);
        while(q--){
            scanf("%d%d", &a,&b);
            int x = LCA(a,b);
            ll ans = dist[a]+dist[b]-2*dist[x];
            if(q) printf("%lld_", ans);
            else printf("%lld", ans);
        }
        printf("\n");
        for(i = 0; i < n; i++) G[i].clear();
    }

    return 0;
}
```

Fluxo Máximo

The Cool Monkeys

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

using namespace std;

#define ff first
#define ss second
#define pb push_back
#define mp make_pair

const int oo = 1000000000;

typedef pair<int,int> ii;

vector<vector<int> > g;

int m, na, nb, t, source, target, ha[505], hb[505];
int mat[2020][2020], p[2020], vis[2020];

int back(int u, int minEdge){
        if(u == source) return minEdge;
        int f = back(p[u], min(minEdge, mat[ p[u] ][u]));
        mat[ p[u] ][u] -= f;
        mat[u][ p[u] ] += f;
        return f;
}

int maxflow(){
        int mf = 0, f = 1;
        while(f){
                queue<int> q;
                q.push(source);
                memset(vis, 0, sizeof vis);

                vis[source] = 1;
                p[source] = source;

                while(!q.empty()){
                        int u = q.front(); q.pop();

                        if(u == target) break;

                        for(int i = 0; i < g[u].size(); i++){
                                int v = g[u][i];
                                if(mat[u][v] > 0 && vis[v] != 1){
                                        vis[v] = 1;
                                        p[v] = u;
                                        q.push(v);
                                }
                        }
                }

                if(vis[target] != 1) break;
```

```
                f = back(target, oo);
                mf += f;

        }
        return mf;
}

int buildRun(int *ha, int na, int *hb, int nb){
        g.assign(2020, vector<int>());
        sort(ha, ha+na, greater<int>());
        sort(hb, hb+nb);

        memset(mat, 0, sizeof mat);

        int cnt = 0;
        for(int i = 0; i < na; i++){
                for(int j = 0; j < nb; j++){
                        int vin = i;
                        int uin = na+j;
                        int vout = na+nb+i;
                        int uout = na+nb+na+j;
                        if(abs(ha[i] - hb[j]) < t){
                                // vout -> uin
                                // uout -> vin
                                // printf("%d -> %d\n", i, j);
                                g[vout].pb(uin);
                                g[uin].pb(vout);
                                g[uout].pb(vin);
                                g[vin].pb(uout);
                                mat[vout][uin] = oo;
                                mat[uout][vin] = oo;
                        }
                }
        }

        for(int i = 0; i < na; i++){
                int vin = i;
                int vout = na+nb+i;
                g[vin].pb(vout);
                g[vout].pb(vin);
                mat[vin][vout] = 1;
        }


        for(int j = 0; j < nb; j++){
                int uin = na+j;
                int uout = na+nb+na+j;
                g[uin].pb(uout);
                g[uout].pb(uin);
                mat[uin][uout] = 1;
        }
```

```
    for(int i = 0; i < m; i++){
        int vin = i;
        int uout = na+nb+na+i;
        g[source].pb(vin);
        g[vin].pb(source);
        mat[source][vin] = 1;

        g[uout].pb(target);
        g[target].pb(uout);
        mat[uout][target] = 1;
    }

    return maxflow();
}

int main(){
    source = 2018;
    target = 2019;

    scanf("%d %d %d %d", &m, &na, &nb, &t);

    for(int i = 0; i < na; i++)
        scanf("%d", ha+i);
    for(int i = 0; i < nb; i++)
        scanf("%d", hb+i);

    if(buildRun(ha, na, hb, nb) == m || buildRun(hb, nb, ha, na) == m) printf("S\n");
    else printf("N\n");

    return 0;
}
```

Strings

YATG

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

#define ff first
#define ss second
#define mp make_pair
#define oo 1000000000

using namespace std;

int n, k, mat[300005][26], ans;

int dfs(int u){
    int ret = oo;
    ans++;
```

```
        for(int i = 0; i < 26; i++){
            if(mat[u][i]){
                ret = min(ret, dfs(mat[u][i]));
            }
        }
        if(ret == oo){
            ret = 0;
            ans++;
        }
        ret++;
        if(ret > k){
            ans++;
            ret = 1;
        }
        return ret;
}

int main(){

    scanf("%d %d", &n, &k);

    char s[100005];

    int ptr = 1;

    for(int i = 0; i < n; i++){
        scanf(" %s", s);
        int node = 0;
        for(int j = 0; s[j]; j++){
            int letra = s[j]-'a';
            if(!mat[node][letra])
                mat[node][letra] = ptr++;
            node = mat[node][letra];
        }
    }

    for(int i = 0; i < 26; i++)
        if(mat[0][i])
            dfs(mat[0][i]);

    ans += n;

    printf("%d\n", ans);

    return 0;
}
```

Vasiliy's Multiset

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

```
#define oo 1000000000000000000

using namespace std;

int mat[6400640][2];
int cnt[6400640][2];

int main(){
    int n, x, ptr, next = 1;
    char c;

    scanf("%d", &n);

    ptr = 0;
    for(int i = 30; i >= 0; i--){
        if(!mat[ptr][0]) mat[ptr][0] = next++;
        cnt[ptr][0]++;
        ptr = mat[ptr][0];
    }

    for(int i = 0; i < n; i++){
        scanf(" %c %d", &c, &x);

        if(c == '+'){
            ptr = 0;
            for(int i = 30; i >= 0; i--){
                if((1 << i) & x){
                    if(!mat[ptr][1]) mat[ptr][1] = next++;
                    cnt[ptr][1]++;
                    ptr = mat[ptr][1];
                }
                else{
                    if(!mat[ptr][0]) mat[ptr][0] = next++;
                    cnt[ptr][0]++;
                    ptr = mat[ptr][0];
                }
            }
        }
        else if(c == '-'){
            ptr = 0;
            for(int i = 30; i >= 0; i--){
                if((1 << i) & x){
                    cnt[ptr][1]--;
                    ptr = mat[ptr][1];
                }
                else{
                    cnt[ptr][0]--;
                    ptr = mat[ptr][0];
                }
            }
        }
```

```
        else{
            int ans = 0;
            ptr = 0;
            for(int i = 30; i >= 0; i--){
                if((1 << i) & x){
                    if(cnt[ptr][0] > 0){
                        ans += (1 << i);
                        ptr = mat[ptr][0];
                    }
                    else ptr = mat[ptr][1];
                }
                else{
                    if(cnt[ptr][1] > 0){
                        ans += (1 << i);
                        ptr = mat[ptr][1];
                    }
                    else ptr = mat[ptr][0];
                }
            }
            printf("%d\n", ans);
        }
    }

    return 0;
}
```

Trie ponteiro

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

using namespace std;

typedef long long ll;

struct node{
    bool is_end;
    int prefixes, words, maxsize;
    struct node* edge[26];
    //initialize()
    //addword(vertex, word)
    //countPrefixes(vertex, prefix)
    //countWords(vertex, word)

    node(){
        maxsize = 0;
        prefixes = 0;
        words = 0;
        is_end = false;
        for(int i = 0; i < 26; i++) edge[i] = NULL;
    }
```

```cpp
        void addWord(string word, int tam){
                maxsize = max(tam, maxsize);
                if(word.empty()){
                        prefixes++;
                        words++;
                }
                else{
                        prefixes++;
                        int k = word[0] - 'a';
                        if(edge[k] == NULL){
                                node *p1 = new node();
                                p1->addWord(word.substr(1,word.size()-1), tam );
                                edge[k] = p1;
                        }
                        else
                                edge[k]->addWord(word.substr(1,word.size()-1), tam );
                }
        }

        int countWords(string word){
                if(word.empty())
                        return words;
                int k = word[0] - 'a';
                if(edge[k] == NULL)
                        return 0;
                return edge[k]->countWords(word.substr(1,word.size()-1) );
        }

        int countPrefixes(string word){
                if(word.empty())
                        return prefixes;
                int k = word[0] - 'a';
                if(edge[k] == NULL)
                        return 0;
                return edge[k]->countPrefixes(word.substr(1,word.size()-1) );
        }

        int countSize(string word){
                if(word.empty())
                        return maxsize;
                int k = word[0] - 'a';
                if(edge[k] == NULL)
                        return 0;
                return edge[k]->countSize(word.substr(1,word.size()-1) );
        }
};

int main() {
        int n,m;
        string s;

        while(scanf("%d", &n) != EOF){
```

```cpp
                node *trie = new node();
                while(n--){
                        cin >> s;
                        trie->addWord(s, s.size());
                }
                scanf("%d", &m);
                while(m--){
                        cin >> s;
                        int x = trie->countPrefixes(s);
                        if(x == 0)
                                cout << "-1\n";
                        else
                                cout << x << " " << trie->countSize(s) << endl;
                }
        }

        return 0;
}
```

Geometria

Dividindo a coca

```cpp
#include <cstdio>
#include <cmath>
using namespace std;

int main(){
    int c, n, l, b, H, i;
    double B, h, sup, inf, mid, bb, v;

    scanf("%d", &c);

    while(c--){
        scanf("%d %d %d %lf %d", &n, &l, &b, &B, &H);

        sup = H;
        inf = 0;

        while(sup - inf > 1e-9){
            mid = (sup + inf) / 2;

            bb = b + (B - b)*mid/sup;

            v = (M_PI*mid*(bb*bb + bb*b+ b*b))/3;
            if(v*n > l){
                sup = mid;
                B = bb;
            }
```

```
        else inf = mid;
    }
    printf("%.2lf\n", sup);
}

return 0;
```

```
}
// r*r = B*B*H/h

// H/(R-r) = h/(R1-r)
// R1-r = (R-r)*h/H
// R1 = r + (R-r)*h/H
```