

# VanHien University ACM-ICPC Team Notebook

**School :** VanHien University  
**Team :** VHU.dattebayo  
**Members :** Pham Duc Long  
 Nguyen Hoang Quoc Anh  
 Tran Phuc Nam  
**Coach :** Pham Thanh Dat  
**Competition :** ICPC Vietnam National 2025

---

## Contents

<b>1 Data Structures</b>	3
1.1 Fenwick Tree . . . . .	3
1.2 2D Fenwick Tree . . . . .	3
1.3 Segment Tree . . . . .	4
1.4 Sqrt Decomposition . . . . .	5
1.5 Mo's Algorithm . . . . .	5
1.6 Trie XOR . . . . .	6
1.7 Bua Thang . . . . .	6
<b>2 Dynamic Programming</b>	6
2.1 Bitmask DP . . . . .	6
2.2 Digit DP . . . . .	7
2.3 Divide and Conquer DP . . . . .	7
2.4 Tree DP / Reroot . . . . .	8
2.5 SOS DP . . . . .	8
2.6 LIS DP . . . . .	8
2.7 Fibonacci Matrix . . . . .	9
2.8 Another DP . . . . .	9
<b>3 Geometry</b>	9
3.1 Closest Pair / Geometry Helper . . . . .	9
3.2 Convexhull . . . . .	10
3.3 Geometry Formulae . . . . .	10
<b>4 Graph</b>	11
4.1 0-1 BFS . . . . .	11
4.2 Bellman–Ford . . . . .	12
4.3 Dijkstra . . . . .	12
4.4 DSU . . . . .	12
4.5 Kruskal MST . . . . .	13
4.6 LCA . . . . .	13
4.7 Tarjan SCC . . . . .	14
4.8 Topological Sort . . . . .	14
4.9 Euler Path . . . . .	14
4.10 Floyd–Warshall . . . . .	15
4.11 Articulation Points & Bridges . . . . .	15
4.12 Two-Sided Graph . . . . .	16
4.13 Flows . . . . .	16
<b>5 Math</b>	17
5.1 Combinatorics . . . . .	17
5.2 Divisor / Factors . . . . .	18

---

5.3	Euler's Totient . . . . .	18
5.4	Sieve . . . . .	18
5.5	Prime Numbers . . . . .	18
5.6	Math Formulae . . . . .	19
<b>6</b>	<b>Strings</b>	<b>19</b>
6.1	KMP Search . . . . .	19
6.2	String Hashing . . . . .	19
6.3	Trie . . . . .	20
<b>7</b>	<b>Misc</b>	<b>21</b>
7.1	Bitset tricks . . . . .	21
7.2	Ternary Search . . . . .	21
7.3	STL Reference . . . . .	22
7.4	Python Utilities . . . . .	23
7.5	Small templates . . . . .	24

# 1 Data Structures

## 1.1 Fenwick Tree

Listing 1: Fenwick Tree

```

1 // Problem : Day nghich the do dai k
2 //template <typename T>
3 struct Fenwick {
4     int n;
5     vector<ll>bit;
6
7     Fenwick(int _n) : n(_n), bit(_n + 1, 0) { }
8
9     inline static int lsone(int x) {return x & -x; }
10
11    void add(int pos, ll value) {
12        for(; pos <= n; pos += lsone(pos)) {
13            addMod(bit[pos], value);
14        }
15    }
16
17    ll sum(int pos) {
18        ll sum = 0;
19        for(;pos > 0; pos -= lsone(pos)) {
20            addMod(sum, bit[pos]);
21        }
22        return sum;
23    }
24
25    ll get(int l, int r) {
26        return sum(r) - sum(l - 1);
27    }
28};
29
30    ll res = 0;
31    int n, k;
32    vector<int>a, r;
33    vector<vector<ll>>f(_N, vector<ll>(_N, 0));
34
35    void solve() {
36        cin >> n >> k;
37        a.assign(n + 1, 0);
38        r.assign(n + 1, 0);
39
40        for(int i = 1; i <= n; i++) {
41            cin >> a[i];
42        }
43
44        // nen
45        vector<int>v = a;
46        sort(v.begin() + 1, v.begin() + n + 1);
47        v.erase(unique(v.begin() + 1, v.begin() + n + 1), v.end());
48        int R = v.size();
49        auto findRank = [&](int x) -> int {
50            return lower_bound(v.begin() + 1, v.begin() + n + 1, x) - v.begin();
51        };
52
53        for(int i = 1; i <= n; i++) {
54            f[1][i] = 1; // so day nghich the do dai la 1 ket thuc tai i : => la chinh no
55            r[i] = findRank(a[i]);
56        }
57
58        // f[i][j] : so day nghich the do dai i ket thuc tai j
59        for(int i = 2; i <= k; i++) {
60            Fenwick fw(R);
61            for(int j = 1; j <= n; j++) {
62                // lay tong cac F[i-1][j1] voi j1 < j va r[j1] > r[j]
63                // => lay so co gia tri lon hon a[j1] da xuat hien truoc do
64                if(r[j] + 1 <= R) f[i][j] = fw.get(r[j] + 1, R);
65                fw.add(r[j], f[i - 1][j]); // update
66            }
67
68        // kg : f[k][j] so day nghich the do dai k ket thuc tai j
69        for(int j = 1; j <= n; j++) {
70            addMod(res, f[k][j]);
71        }
72
73        cout << res << endl;
74    }
75
76}

```

```

77 // Problem : Buoc nhay xa nhat : 1 <= i < j <= N; Aj - Ai >= P; j - i lon nhat. Khi do j - i duoc
78 // goi la do dai buoc nhay xa nhat cua day.
79 int n, k, nn = 0;
80 vector<int>a;
81 vector<int>bit;
82 vector<int>vals;
83 void update(int pos, int id)
84 {
85     for (; pos <= nn; pos += (pos & -pos))
86     {
87         bit[pos] = min(bit[pos], id);
88     }
89 }
90
91 int get(int pos)
92 {
93     int minPos = 1e9;
94     for (; pos > 0; pos -= (pos & -pos))
95     {
96         minPos = min(minPos, bit[pos]);
97     }
98     return minPos;
99 }
100
101 void solve()
102 {
103     cin >> n >> k;
104
105     a.assign(n + 1, 0);
106     for (int i = 1; i <= n; i++)
107     {
108         cin >> a[i];
109         vals.push_back(a[i]);
110     }
111
112     sort(all(vals));
113     vals.erase(unique(all(vals)), vals.end());
114
115     for(int i = 1; i <= n; i++) {
116         a[i] = lower_bound(vals.begin(), vals.end(), a[i]) - vals.begin() + 1;
117     }
118
119     nn = vals.size();
120     bit.assign(nn + 1, 1e9);
121
122     update(a[1], 1);
123     int maxDist = 0;
124     for(int i = 2; i <= n; i++) {
125
126         int x = vals[a[i] - 1]; // tim gia tri
127         int target = x - k;
128
129         int p = upper_bound(all(vals), target) - vals.begin();
130
131         if(p > 0) {
132             int g = get(p);
133             maxDist = max(maxDist, i - g);
134         }
135
136         update(a[i], i);
137
138     }
139
140     cout << maxDist << endl;
141 }

```

## 1.2 2D Fenwick Tree

Listing 2: 2D Fenwick Tree

```

1 // Fenwick tree 2D
2
3 struct Fenwick2D {
4     int n, m;
5     vector<vector<long long>> bit;
6
7     Fenwick2D(int _n, int _m) {
8         n = _n;
9         m = _m;
10        bit.assign(n + 1, vector<long long>(m + 1, 0));
11    }
12

```

```

13 // Cong them val vao o (x, y)
14 void update(int x, int y, long long val) {
15     for (int i = x; i <= n; i += i & -i) {
16         for (int j = y; j <= m; j += j & -j) {
17             bit[i][j] += val;
18         }
19     }
20 }
21
22 // Lay tong tu (1,1) den (x,y)
23 long long query(int x, int y) {
24     long long res = 0;
25     for (int i = x; i > 0; i -= i & -i) {
26         for (int j = y; j > 0; j -= j & -j) {
27             res += bit[i][j];
28         }
29     }
30     return res;
31 }
32
33 // Lay tong trong hinh chu nhat (x1,y1) -> (x2,y2)
34 long long rangeQuery(int x1, int y1, int x2, int y2) {
35     return query(x2, y2)
36         - query(x1 - 1, y2)
37         - query(x2, y1 - 1)
38         + query(x1 - 1, y1 - 1);
39 }
40 }
41
42 // update :
43 fenw.update(x, y, val);
44
45 // get :
46 cout << fenw.rangeQuery(x1, y1, x2, y2) << "\n";

```

### 1.3 Segment Tree

Listing 3: Segment Tree

```

1 // Cho Q truy van, moi truy van co dang: L R K
2 // Yeu cau: moi truy van xuat ra phan tu lon thu K sau khi sap xep cac phan tu AL, AL+1, ..., AR
3 // theo thu tu tang dan.
4 const int N = 1e5 + 5;
5 typedef vector<int> dta;
6 int n, q, l, r, k, a[N];
7 vector<int> v;
8 dta t[N * 4];
9
10 // merge(seg[id*2].begin(), seg[id*2].end(),
11 //        seg[id*2+1].begin(), seg[id*2+1].end(),
12 //        seg[id].begin()); : hoac dung ham merge
13 dta combine(dta u, dta v)
14 {
15     dta ans;
16     ans.reserve(u.size() + v.size());
17     int i = 0, j = 0;
18     while (i < u.size() && j < v.size())
19     {
20         if (u[i] < v[j])
21             ans.push_back(u[i++]);
22         else
23             ans.push_back(v[j++]);
24     }
25     while (i < u.size())
26         ans.push_back(u[i++]);
27     while (j < v.size())
28         ans.push_back(v[j++]);
29     return ans;
30 }
31 void build(int v, int tl, int tr)
32 {
33     if (tl == tr)
34     {
35         t[v].push_back(a[tl]);
36     }
37     else
38     {
39         int mid = (tl + tr) / 2;
40         build(v << 1, tl, mid);
41         build(v << 1 | 1, mid + 1, tr);
42         t[v] = combine(t[v << 1], t[v << 1 | 1]); // merge sort
43     }
44 }
45
46 int get(int v, int tl, int tr, int l, int r, int x)
47 {
48     if (r < tl || l > tr)
49         return 0;
50     if (tl >= l && tr <= r)
51         return upper_bound(t[v].begin(), t[v].end(), x) - t[v].begin();
52     int mid = (tl + tr) / 2;
53     return get(v << 1, tl, mid, l, r, x) + get(v << 1 | 1, mid + 1, tr, l, r, x); // slpt <= x
54 }
55
56 void solve()
57 {
58     cin >> n;
59     v.resize(n);
60     for (int i = 1; i <= n; i++)
61         cin >> a[i], v[i - 1] = a[i];
62
63     sort(v.begin(), v.end());
64     v.erase(unique(v.begin(), v.end()), v.end());
65     for (int i = 1; i <= n; i++)
66         a[i] = lower_bound(v.begin(), v.end(), a[i]) - v.begin() + 1;
67
68     build(1, 1, n);
69
70     cin >> q;
71     while (q--)
72     {
73         cin >> l >> r >> k;
74         int low = 1, high = v.size(), ans = -1;
75         // tim phan tu lon thu K -> tim phan tu nho thu (R-L+1-K+1)
76         // nhung o day ta giu nguyen: ta binary search de tim phan tu nho thu K
77         // binary search tren gia tri X de tim gia tri sao cho co dung K phan tu <= x.
78         while (low <= high)
79         {
80             int mid = (low + high) / 2;
81             int cnt = get(1, 1, n, l, r, mid);
82             if (cnt >= k)
83             {
84                 ans = mid;
85                 high = mid - 1;
86             }
87             else
88                 low = mid + 1;
89         }
90
91         // tra ve gia tri that (da nen)
92         cout << v[ans - 1] << "\n";
93     }
94
95     struct dta {
96         ll sum, prefix, suffix, best;
97     };
98     int n, a[N], q;
99     dta t[N * 4];
100    // sum : tong doan con(all)
101    // prefix : tong doan con lon nhat bat dau tai l
102    // suffix : tong doan con lon nhat ket thuc tai r
103    // best : tong doan con lon nhat nam trong doan l - r
104    dta combine(dta l, dta r) {
105        dta res;
106        res.sum = l.sum + r.sum; // tong ca doan
107        res.prefix = max(l.prefix, l.sum + r.prefix);
108        res.suffix = max(r.suffix, r.sum + l.suffix);
109        res.best = max(max(l.best, r.best), l.suffix + r.prefix);
110        return res;
111    }
112
113    dta make(int val) {
114        return {val, val, val, val};
115    }
116
117    void build(int v, int tl, int tr) {
118        if (tl == tr)
119        {
120            t[v] = make(a[tl]);
121        }
122        else
123        {
124            int mid = (tl + tr) / 2;
125            build(v << 1, tl, mid);
126            build(v << 1 | 1, mid + 1, tr);
127            t[v] = combine(t[v << 1], t[v << 1 | 1]);
128        }
129    }
130
131    dta get(int v, int tl, int tr, int l, int r) {
132        if (r < tl || l > tr)
133

```

```

131     return make(-1e9);
132     if (tl >= l && tr <= r)
133         return t[v];
134     int mid = (tl + tr) / 2;
135     return combine(get(v << 1, tl, mid, l, r), get(v << 1 | 1, mid + 1, tr, l, r));
136 }
137
138 void solve() {
139     cin >> n;
140     for(int i = 1; i <= n; i++)
141         cin >> a[i];
142
143     build(1, 1, n);
144     cin >> q;
145     while(q--) {
146         int l, r;
147         cin >> l >> r;
148         cout << get(1, 1, n, l, r).best << endl;
149     }
150 }
151
152 // lazy update
153 void push(int v, int tl, int tr)
154 {
155     if (t[v].lazy == 0)
156         return;
157     t[v].val += (tr - tl + 1) * t[v].lazy;
158     if (tl != tr)
159     {
160         t[v << 1].lazy += t[v].lazy;
161         t[v << 1 | 1].lazy += t[v].lazy;
162     }
163     t[v].lazy = 0;
164 }
165 void update(int v, int tl, int tr, int l, int r, int x)
166 {
167     push(v, tl, tr);
168     if (tl > r || l > tr)
169         return;
170     if (l <= tl && tr <= r)
171     {
172         t[v].lazy += x;
173         push(v, tl, tr);
174     }
175     else
176     {
177         int mid = (tl + tr) >> 1;
178         update(v << 1, tl, mid, l, r, x);
179         update(v << 1 | 1, mid + 1, tr, l, r, x);
180         t[v].val = t[v << 1].val + t[v << 1 | 1].val;
181     }
182 }
183 ll get(int v, int tl, int tr, int l, int r)
184 {
185     push(v, tl, tr);
186     if (tl > r || l > tr)
187         return 0;
188     if (l <= tl && tr <= r)
189     {
190         return t[v].val;
191     }
192     else
193     {
194         int mid = (tl + tr) >> 1;
195         ll getl = get(v << 1, tl, mid, l, r);
196         ll getr = get(v << 1 | 1, mid + 1, tr, l, r);
197         return getl + getr;
198     }
199 }

```

## 1.4 Sqrt Decomposition

Listing 4: Sqrt Decomposition

```

1 // sqrt decomposition
2 const int N = 2e5 + 5;
3 ll a[N], bucket[450]; // sqrt(1e5)
4 int n, blockSize;
5 void build() {
6     blockSize = sqrt(n);
7

```

```

8     for(int i = 0; i < n; i++) {
9         bucket[i / blockSize] += a[i];
10    }
11
12    void update(int pos, int value) {
13        int b = pos / blockSize;
14        bucket[b] += value - a[pos];
15        a[pos] = value;
16    }
17
18    ll get(int l, int r) {
19        ll sum = 0;
20        int lb = l / blockSize;
21        int rb = r / blockSize;
22
23        if(lb == rb) {
24            for(int i = l; i <= r; i++) {
25                sum = sum + a[i];
26            }
27        } else {
28            // tinh tong cac block o giao(full)
29            for(int i = lb + 1; i < rb; i++)
30                sum += bucket[i];
31
32            // tinh tong phan con lai ben trai
33            for(int i = lb * blockSize; i <= r; i++)
34                sum += a[i];
35
36            // tinh tong phan con lai ben phai
37            for(int i = rb * blockSize; i <= r; i++)
38                sum += a[i];
39
40        }
41        return sum;
42    }
43
44    void solve() {
45        cin >> n >> q;
46
47        for(int i = 0; i < n; i++) {
48            cin >> a[i];
49        }
50
51        build();
52        while (q--) {
53            int t; cin >> t;
54            if (t == 2) {
55                int l, r; cin >> l >> r;
56                l--, r--;
57                cout << get(l, r) << "\n";
58            } else {
59                int pos, val; cin >> pos >> val;
60                pos--;
61                update(pos, val);
62            }
63        }
64    }
65 }

```

## 1.5 Mo's Algorithm

Listing 5: Mo's Algorithm

```

1 // Mo algorithm
2 const int N = 1e5+5;
3 int a[N];
4 int n, q, blockSize;
5 struct Query {
6     int l, r, idx;
7     bool operator < (const Query &other) const {
8         if (l / blockSize != other.l / blockSize)
9             return l < other.l;
10        return r < other.r;
11    }
12 };
13 vector<Query> queries;
14 long long ans[N];
15 inline void add(int pos) { currSum += a[pos]; }
16 inline void remove(int pos) { currSum -= a[pos]; }
17 void solve() {

```

```

19   cin >> n >> q;
20   for (int i = 0; i < n; i++) cin >> a[i];
21   blockSize = sqrt(n);
22   for (int i = 0; i < q; i++) {
23     int l, r;
24     cin >> l >> r;
25     l--; r--;
26     queries.push_back({l, r, i});
27   }
28   sort(queries.begin(), queries.end());
29   int currL = 0, currR = -1; // input st == 1, currL = 1, currR = 0
30   for (auto qu : queries) {
31     while (currL > qu.l) add(--currL); // bi thieu phan tu, giam de cong them ben duoi
32     while (currR < qu.r) add(++currR); // bi thieu phan tu, tang len de cong them ben tren
33     while (currL < qu.l) remove(currL++); // bi du phan tu, tang len de tru di
34     while (currR > qu.r) remove(currR--); // bi du phan tu, giam xuong tru di
35     ans[qu.idx] = currSum;
36   }
37   for (int i = 0; i < q; i++) cout << ans[i] << "\n";
38 }

```

## 1.6 Trie XOR

Listing 6: Trie XOR

```

1 // Trie XOR - Find max XOR pair
2 // Time: O(n*log(max_val))
3 const int MAXBIT = 30;
4 struct TrieNode {
5   TrieNode* child[2];
6   int cnt;
7   TrieNode() {
8     child[0] = child[1] = NULL;
9     cnt = 0;
10  }
11 };
12 struct TrieXOR {
13   TrieNode* root;
14   TrieXOR() {
15     root = new TrieNode();
16   }
17   void insert(int x) {
18     TrieNode* cur = root;
19     for(int i = MAXBIT; i >= 0; i--) {
20       int bit = (x >> i) & 1;
21       if(!cur->child[bit])
22         cur->child[bit] = new TrieNode();
23       cur = cur->child[bit];
24       cur->cnt++;
25     }
26   } // cac so khi xor voi X lon nhat la bn
27   int maxXOR(int x) {
28     TrieNode* cur = root;
29     int res = 0;
30     for(int i = MAXBIT; i >= 0; i--) {
31       int bit = (x >> i) & 1;
32       int need = 1 - bit;
33       if(cur->child[need]) {
34         res |= (1 << i);
35         cur = cur->child[need];
36       } else {
37         cur = cur->child[bit];
38       }
39     }
40     return res;
41   } // so lon thu k
42   int kthXOR(int x, int k) {
43     TrieNode* cur = root;
44     int res = 0;
45     for(int i = MAXBIT; i >= 0; i--) {
46       int bit = (x >> i) & 1;
47       int need = 1 - bit;
48       if(cur->child[need] && cur->child[need]->cnt >= k) {
49         res |= (1 << i);
50         cur = cur->child[need];
51       } else {
52         if(cur->child[need])
53           k = cur->child[need]->cnt;
54         cur = cur->child[bit];
55       }
56     }

```

```

57   }
58   return res;
59 }

```

## 1.7 Bua Thang

Listing 7: Bua Thang

```

1 const int MAXN = 1e6 + 5;
2 const int LOG = 20;
3
4 int n, a[10];
5 int minV[MAXN][LOG];
6
7 void build() {
8   for(int i=1;i<=n;i++) minV[i][0] = a[i];
9
10  for(int j=1;MASK(j) <=n;j++) {
11    for(int i=1;i+MASK(j)-1<=n;i++) {
12      minV[i][j] = min(minV[i][j-1],minV[i+MASK(j-1)][j-1]);
13    }
14  }
15 }
16
17 int getMin(int l,int r){
18   int k = 31 - __builtin_clz(r-l+1);
19   return min(minV[l][k],minV[r-MASK(k)][k]);
20 }
21
22 int main() {
23 #ifdef MY
24 #endif
25   ios_base::sync_with_stdio(false);
26   cin.tie(nullptr);
27   cin >> n;
28   for(int i=1;i<=n;i++) cin >> a[i];
29   build();
30   int q;
31   cin >> q;
32   while(q--){
33     int l,r;
34     cin >> l >> r;
35     cout << getMin(l,r) << endl;
36   }
37 }
38

```

## 2 Dynamic Programming

### 2.1 Bitmask DP

Listing 8: Bitmask DP

```

1 //dem so luong chu trinh, do dai chu trinh co the >= 3
2 int n, m, x, y, ans = 0;
3 vector<int> G(20, 0);
4 int dp[{1 << 20}][20];
5 int LOG[{1 << 20}]; // bit thap nhat cua dinh i
6 // so duong di tu dinh s, ket thuc tai u va di qua moi dinh trong tap mask
7 // voi moi mask, de xac dinh chu trinh ta se lay bit thap nhat cua tap mask de lam dinh bat dau
8 // va ta se dem so luong duong di giua cac dinh ko phai bit thap nhat, roi sau do moi xu ly sau
9 void solve() {
10   cin >> n >> m;
11   for(int i = 0; i < m; i++) {
12     cin >> x >> y;
13     x--, y--;
14     G[x] |= (1 << y);
15     G[y] |= (1 << x);
16   }
17   for(int i = 2; i <= 1e6; i++)
18     LOG[i] = LOG[i / 2] + 1; // bit thap nhat cua moi tap mask

```

```

19   for(int i = 0; i < n; i++) {
20     dp[(1 << i)][i] = 1;
21   }
22   for(int mask = 0; mask < (1 << n); mask++) {
23     if(__builtin_popcount(mask) == 1) continue;
24     for(int u = 0; u < n; u++) {
25       if(mask & (1 << u) && u != LOG[mask & -mask]) { // ko phai bit thap nhat
26         for(int v = 0; v < n; v++) {
27           if(G[u] & (1 << v) && mask & (1 << v)) {
28             dp[mask][u] += dp[mask ^ (1 << u)][v];
29           }
30         }
31       }
32     }
33   }
34   for(int mask = 0; mask < (1 << n); mask++) {
35     for(int u = 0; u < n; u++) {
36       if(__builtin_popcount(mask) < 3) continue;
37       if(!(mask & (1 << u))) continue;
38       int lb = LOG[mask & -mask];
39       if(G[lb] & (1 << u)) { // neu co duong di tu bit thap nhat den u
40         ans = ans + dp[mask][u];
41       }
42     }
43   }
44   ans >= 1; // moi chu trinh bi dem 2 lan
45   cout << ans << endl;
46 }
47 // xoa canh(hoac ko), sao cho moi dinh trong tplt deu co duong di den nhau
48 // in ra so luong tplt it nhat sau khi xoa, neu ko xoa(nghia la do thi dung voi yeu cau) thi ban dau
49 // = 1
50 vector<ll>dp(1 << 18, INT32_MAX);
51 vector<int>G(20, 0);
52 int n, m, x, y;
53 // dp[mask] = so clique it nhat de phu toan bo tap trong mask
54 void solve() {
55   cin >> n >> m;
56   for(int i = 0; i < m; i++) {
57     cin >> x >> y;
58     x--;
59     y--;
60     G[x] |= (1 << y);
61     G[y] |= (1 << x);
62   }
63   for(int mask = 0; mask < (1 << n); mask++) {
64     bool connected = true;
65     for(int u = 0; u < n; u++) {
66       if (!(mask & (1 << u))) continue;
67       if((G[u] |= (1 << u)) & mask) != mask) { //check connected all vertices
68         connected = false;
69         break;
70       }
71     }
72     if(connected == true) dp[mask] = 1; // if all vertices is connected, => 1 components
73   }
74   for(int mask = 0; mask < (1 << n); mask++) {
75     for(int sub = mask; sub > 0; sub = (sub - 1) & mask) { // sub : tap con
76       int rest = sub ^ mask; // rest : lay phan bu trong mask (tuc
77       // nhung phan tu cua mask khong co trong sub)
78       if(dp[rest] != INT32_MAX && dp[sub] != INT32_MAX) { // 7 => sub : 6 : 110, rest : 1 : 001
79         dp[mask] = min(dp[mask], dp[sub] + dp[rest]);
80       }
81     }
82   }
83   cout << dp[(1 << n) - 1] << endl;
84 }

```

## 2.2 Digit DP

Listing 9: Digit DP

```

1 // de yeu cau gi, lam do
2 ll dp[2004][2004][2];
3 string a, b;
4 int num, d;
5 bool valid(int pos, int i) {
6   return (pos % 2 == 0 ? i == d : i != d);
7 }
8 ll f(const string &s, int N, int pos, int sum, int t, int fl) {
9   if (pos == N)
10     return sum == 0 && t == 1;
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
837
838
839
839
840
841
842
843
844
844
845
846
846
847
847
848
849
849
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
15
```

```

44 int n, k, ans = 0;
45 int L = 1, R = 0;
46 ll dp[N][21];
47 vector<int> a(N);
48 vector<int> cnt(N, 0);
49
50 // ----- Sliding window Mo's Algorithm -----
51 void add(int id) { ans += cnt[a[id]]++; }
52 void del(int id) { ans -= --cnt[a[id]]; }
53
54 ll move(int l, int r) {
55     while(L > l) add(--L); // thieu
56     while(R < r) add(++R); // thieu
57     while(L < l) del(L++); // du
58     while(R > r) del(R--); // du
59     return ans;
60 }
61
62 // ----- dp divide -----
63 void cal(int l, int r, int opl, int opr, int g) {
64     if(l > r) return;
65
66     int mid = (l + r) / 2;
67     ii best = {INF, -1};
68
69     for(int i = opl; i <= min(mid, opr); i++) {
70         ll cost = dp[i][g - 1] + move(i + 1, mid);
71
72         if(cost < best.first) {
73             best.first = cost;
74             best.second = i;
75         }
76     }
77
78     dp[mid][g] = best.first;
79     int opt = best.second;
80
81     cal(l, mid - 1, opl, opt, g);
82     cal(mid + 1, r, opt, opr, g);
83 }
84
85 void solve() {
86     cin >> n >> k;
87     memset(dp, 0x3f, sizeof(dp));
88     for(int i = 1; i <= n; i++)
89         cin >> a[i];
90
91     dp[0][0] = 0;
92     for(int i = 1; i <= n; i++) {
93         ans = ans + cnt[a[i]]++;
94         dp[i][1] = ans;
95     }
96
97     // reset
98     fill(cnt.begin(), cnt.end(), 0);
99     ans = 0, L = 1, R = 0;
100    for(int d = 2; d <= k; d++) {
101        cal(l, n, 1, n, d);
102    }
103    cout << dp[n][k];
104 }

```

## 2.4 Tree DP / Reroot

Listing 11: Tree DP / Reroot

```

1 // bai to mau : voi moi lan to 1 mau tren cay thanh mau den se duoc so diem la
2 // so luong dinh trang dang lien thong voi dinh hien tai + 1(dinh hien tai)
3 // tim cach to sao cho diem la lon nhat
4 void dfs(int u, int p) {
5     sz[u] = 1;
6     for(auto v : G[u]) {
7         if(v == p) { continue; }
8         dfs(v, u);
9         sz[u] += sz[v];
10    }
11    f[u] += f[v]; // f[v] : so diem cua cay con truoc do, ko kem u
12    f[u] += sz[u]; // += so luong con cua u, tuong duong so diem truoc khi xoa u
13 }
14 // cap nhat nguoc lai tu dinh cha ve dinh con
15 void cal(int u, int p) {

```

```

16     for(int v : G[u]) {
17         if(v == p) { continue; }
18         f[v] = f[v] + (f[u] - f[v] - sz[v]) + (n - sz[v]);
19         // (dp[u] - dp[v] - sz[v]): tong diem tu phan con lai cua cay (ngoai tru subtree v).
20         // n - sz[v] : khi chon v, ta con nhan duoc diem con lai = so node ngoai subtree v.
21         cal(v, u);
22     }
23 }
24 void solve() {
25     dfs(1, 1);
26     cal(1, 1);
27     ans = *max_element(f + 1, f + n + 1);
28     cout << ans << endl;
29 }
30
31 // cho 1 cay co huong, tim 1 thanh pho dat lam thu do sao cho no co the di toi moi dinh khac, va co
32 // the xoay voi so canh it nhat
33 // neu co nhieu thanh pho, in ra so lan xoay it nhat + cac thanh pho do
34 void dfs(int u, int p) {
35     for(auto& [v, w] : G[u]) {
36         if(v == p) continue;
37         dfs(v, u);
38         dp[u] += dp[v] + w;
39     }
40 }
41 void cal(int u, int p) {
42     for(auto& [v, w] : G[u]) {
43         if(v == p) continue;
44         if(w == 0)
45             dp[v] = dp[u] + 1;
46         else
47             dp[v] = dp[u] - 1;
48         cal(v, u);
49     }
50 }
51 void solve() {
52     cin >> n;
53     for(int i = 0; i < n - 1; i++) {
54         cin >> x >> y;
55         G[x].push_back({y, 0}); // inv direc
56         G[y].push_back({x, 1}); // direc
57     }
58     dfs(1, 1);
59     cal(1, 1);
60     cap = *min_element(dp.begin() + 1, dp.begin() + n + 1);
61     cout << cap << endl;
62     for(int i = 1; i <= n; i++) {
63         if(dp[i] == cap)
64             cout << i << ' ';
65     }
66     cout << "\n";
67 }

```

## 2.5 SOS DP

Listing 12: SOS DP

```

1 //: DP SOS O(N * 2^N)
2 // A[i] = initial values
3 // Calculate F[i] = Sum of A[j] for j subset of i
4 for(int i = 0; i < (1 << N); i++)
5     F[i] = A[i];
6 for(int i = 0; i < N; i++)
7     for(int j = 0; j < (1 << N); j++)
8         if(j & (1 << i))
9             F[j] += F[j ^ (1 << i)];
10
11 //tinh tong cac tap con khac sos
12 for(int mask = 0; mask < (MASK(n)); mask++)
13     if(__builtin_popcount(mask) >= 2) {
14         int tmp = mask & -mask;
15         sum[mask] = sum[tmp] + sum[mask ^ tmp];
16     }

```

## 2.6 LIS DP

Listing 13: LIS DP

```

1 // Longest Increasing Subsequence - O(nlogn)
2 // dp[i] = max j < i { dp(j) | a[j] < a[i] } + 1
3 // also can use fenwick tree
4 // int dp[N], v[N], n, lis;
5 // or fenwick tree, maybe:D
6 memset(dp, 63, sizeof dp);
7 for (int i = 0; i < n; ++i) {
8     // increasing: lower_bound
9     // non-decreasing: upper_bound
10    int j = lower_bound(dp, dp + lis, v[i]) - dp;
11    dp[j] = min(dp[j], v[i]);
12    lis = max(lis, j + 1);
13 }
14
15

```

## 2.7 Fibonacci Matrix

Listing 14: Fibonacci Matrix

```

1 // fibo-matrix
2 struct Matrix {
3     ll a, b, c, d;
4     Matrix () : a(1), b(1), c(1), d(0) {}
5     Matrix (ll _a, ll _b, ll _c, ll _d) : a(_a), b(_b), c(_c), d(_d) {}
6     Matrix operator * (const Matrix &oth) {
7         ll x = ((a * oth.a) % MOD + (b * oth.c) % MOD) % MOD;
8         ll y = ((a * oth.b) % MOD + (b * oth.d) % MOD) % MOD;
9         ll z = ((c * oth.a) % MOD + (d * oth.c) % MOD) % MOD;
10        ll t = ((c * oth.b) % MOD + (d * oth.d) % MOD) % MOD;
11        return Matrix(x, y, z, t);
12    }
13 };
14 Matrix binPow(Matrix base, ll n) {
15     if(n <= 1) return base;
16     Matrix tmp = binPow(base, n / 2);
17     tmp = tmp * tmp;
18     if(n & 1) tmp = tmp * base;
19     return tmp;
20 }
21 void solve() {
22     ll n; cin >> n;
23     if(n <= 1) {
24         cout << n << endl;
25     } else {
26         Matrix ans = binPow(Matrix(), n - 1);
27         cout << ans.a % MOD << endl;
28     }
29 }

```

## 2.8 Another DP

Listing 15: Another DP

```

1 // dem so luong day chia het cho splt trong day == 0
2 // vd : 2 2 6 => [2 2] / 2 = 2, [6] / 1 = 6 => 2 day
3 int n, ans = 0;
4 vector<int>a(105);
5 // dp[j][k][l]: xet j phan tu dau tien, da chon k phan tu, tong chia i du l.
6 // dp[n][i][0]: la so tap con co i phan tu va tong chia het cho i.
7 void solve() {
8     cin >> n;
9     for(int i = 0; i < n; i++) {
10         cin >> a[i];
11     }
12     ll ans = 0;
13     ll mod = 998244353;
14     for(int i = 1; i <= n; i++) {
15         vector<vector<vector<ll>>> dp(n + 1, vector<vector<ll>>(i + 1, vector<ll>(i, 0)));
16         dp[0][0][0] = 1;
17     }
18 }

```

```

17     for(int j = 0; j < n; j++) {
18         for(int k = 0; k <= i; k++) {
19             for(int l = 0; l < i; l++) {
20                 // ko chon a[i]
21                 dp[j + 1][k][l] += dp[j][k][l];
22                 dp[j + 1][k][l] %= mod;
23                 // chon a[i]
24                 if(k == i)
25                     continue;
26                 dp[j + 1][k + 1][(l + a[j]) % i] += dp[j][k][l];
27                 dp[j + 1][k + 1][(l + a[j]) % i] %= mod;
28             }
29         }
30     }
31     ans = (ans + dp[n][i][0] % mod) % mod;
32 }
33 cout << ans << endl;
34
35 // dp interval
36 // tren chuo
37 void solve() {
38     cin >> s;
39     n = s.length();
40     mem(dp, 0);
41     for(int i = 0; i < n; i++) {
42         dp[i][i] = 0; // chuo 1 ki tu : l == r : ...
43         dp[i + 1][i] = 1; // chuo rong l > r : ...
44     }
45     for(int d = 1; d < n; d += 2) {
46         for(int l = 0, r = d; r < n; l++, r++) {
47             if(s[l] != '(' && s[l] != '?') continue; // ko ghep dc
48             if(s[r] == '?' || s[r] == ')') dp[l][r] += dp[l + 1][r - 1]; // ghep dc
49             for(int k = l + 1; k < r; k += 2) {
50                 if(s[k] != '1' && s[k] != '?') continue;
51                 dp[l][r] = (dp[l][r] + (dp[l + 1][k - 1] * dp[k + 1][r]) % MOD) % MOD;
52             }
53         }
54     }
55     cout << dp[0][n - 1] << endl;
56 }
57

```

## 3 Geometry

### 3.1 Closest Pair / Geometry Helper

Listing 16: Closest Pair / Geometry Helper

```

struct Point {
    double x, y;
};

double dist(Point a, Point b) {
    return sqrt((a.x - b.x) * (a.x - b.x) +
                (a.y - b.y) * (a.y - b.y));
}

// Brute force cho n nho
double bruteForce(vector<Point>& P, int l, int r) {
    double ans = 1e18;
    for (int i = l; i <= r; i++) {
        for (int j = i + 1; j <= r; j++) {
            ans = min(ans, dist(P[i], P[j]));
        }
    }
    return ans;
}

// ham chinh chia de tri
double closestUtil(vector<Point>& Px, vector<Point>& Py, int l, int r) {
    int n = r - l + 1;
    if (n <= 3) return bruteForce(Px, l, r);

    int mid = (l + r) / 2;
    double midx = Px[mid].x;

    //chia py thanh trai/phai
    vector<Point> Pyl, Pyr;

```

```

31     for (auto& p : Py) {
32         if (p.x <= midx) Pyl.push_back(p);
33         else Pyr.push_back(p);
34     }
35
36     double dl = closestUtil(Px, Pyl, l, mid);
37     double dr = closestUtil(Px, Pyr, mid + 1, r);
38     d = min(dl, dr);
39
40     // tao strip
41     vector<Point> strip;
42     for (auto& p : Py) {
43         if (fabs(p.x - midx) < d) strip.push_back(p);
44     }
45
46     // Kiem tra trong strip =>toi da 7 diem
47     for (int i = 0; i < (int)strip.size(); i++) {
48         for (int j = i + 1; j < (int)strip.size() && j <= i + 7; j++) {
49             d = min(d, dist(strip[i], strip[j]));
50         }
51     }
52
53     return d;
54 }
55
56 double closestPair(vector<Point>& P) {
57     int n = P.size();
58     vector<Point> Px = P, Py = P;
59     //sort theo toa do x
60     sort(Px.begin(), Px.end(), [](Point a, Point b) { return a.x < b.x; });
61     //sort theo toa do y
62     sort(Py.begin(), Py.end(), [](Point a, Point b) { return a.y < b.y; });
63     return closestUtil(Px, Py, 0, n - 1);
64 }
65
66 int main() {
67     int n;
68     cin >> n;
69     vector<Point> P(n);
70     for (int i = 0; i < n; i++) cin >> P[i].x >> P[i].y;
71     //fixed va setprecision(6)
72     cout << fixed << setprecision(6) << closestPair(P) << "\n";
73     return 0;
74 }

```

## 3.2 Convexhull

Listing 17: Convexhull

```

1 vector<Point> makeHull(vector<Point>& pts) {
2     int n = pts.size();
3     vector<int> s(n * 2);
4     vector<bool> onHull(n, false);
5     int k = 0;
6     for (int i = 0; i < n; i++) {
7         while (k >= 2 && CCW(pts[s[k-2]], pts[s[k-1]], pts[i]) > 0) k--;
8         s[k++] = i;
9     }
10    int t = k;
11    for (int i = n - 1; i >= 0; i--) {
12        while (k > t && CCW(pts[s[k-2]], pts[s[k-1]], pts[i]) > 0) k--;
13        s[k++] = i;
14    }
15    for (int i = 0; i < k - 1; i++) onHull[s[i]] = true;
16    vector<Point> next;
17    for (int i = 0; i < n; i++) if (!onHull[i]) next.push_back(pts[i]);
18    return next;
19 }
20 //nguoc chieu kim dong ho
21 sort(a.begin(), a.end(), [&](Point A, Point B) {
22     return atan2(A.y - O.y, A.x - O.x) < atan2(B.y - O.y, B.x - O.x);
23 });
24 //cos * 180/pi = goc
25 //cross = x1y2 - y1x2
26 =0 -> cung phuong or //
27 + -> trai
28 - -> phai
29 //dot = x1x2 + y1y2
30 = 0 -> 90 do
31 > 0 -> goc < 90 do
32 < 0 -> goc > 90 do

```

## 3.3 Geometry Formulae

Listing 18: Geometry Formulae

```

1 struct Point {
2     long long x, y;
3     Point(long long x = 0, long long y = 0) : x(x), y(y) {}
4     Point operator-(const Point& other) const {
5         return Point(x - other.x, y - other.y);
6     }
7     long long cross(const Point& other) const {
8         return x * other.y - y * other.x;
9     }
10    long long dot(const Point& other) const {
11        return x * other.x + y * other.y;
12    }
13    long long distSq() const {
14        return x * x + y * y;
15    }
16
17 // Kiem tra 3 diem thang hang
18 bool areCollinear(Point a, Point b, Point c) {
19     return (b - a).cross(c - a) == 0;
20 }
21
22 // Dien tich da giac (cong thuc shoelace)
23 long long polygonArea(vector<Point>& points) {
24     long long area = 0;
25     int n = points.size();
26     for (int i = 0; i < n; i++) {
27         int j = (i + 1) % n;
28         area += points[i].x * points[j].y - points[j].x * points[i].y;
29     }
30     return abs(area) / 2;
31 }
32
33 // 1. QUAN HE VI TRI DIEM & DUONG THANG
34 // Huong cua diem C so voi duong AB:
35 // cross(AB, AC) = cross(B-A, C-A)
36 double orientation = cross(B-A, C-A);
37 if (orientation > 0) : C ben TRAI AB (nguoc chieu kim dong ho)
38 if (orientation < 0) : C ben PHAI AB (cung chieu kim dong ho)
39 if (orientation == 0) : A, B, C thang hang
40
41 // Kiem tra diem M thuoc doan AB:
42 bool onSegment(Point A, Point B, Point M) {
43     return cross(B-A, M-A) == 0 &&
44         min(A.x, B.x) <= M.x && M.x <= max(A.x, B.x) &&
45         min(A.y, B.y) <= M.y && M.y <= max(A.y, B.y);
46 }
47
48 // 2. KHOANG CACH & HINH CHIEU
49 // Khoang cach tu diem M den duong thang AB:
50 double distToLine(Point A, Point B, Point M) {
51     return abs(cross(B-A, M-A)) / distance(A, B);
52 }
53
54 // Hinh chieu cua M len duong AB:
55 Point project(Point A, Point B, Point M) {
56     Point AB = B - A;
57     Point AM = M - A;
58     double t = dot(AB, AM) / dot(AB, AB); // t = (AB . AM) / |AB|
59     return A + AB * t;
60 }
61
62 // Vi tri hinh chieu:
63 // t < 0: hinh chieu nam phia A (M', A, B)
64 // t > 1: hinh chieu nam phia B (A, B, M')
65 // 0 < t < 1: hinh chieu nam tren doan AB (A, M', B)
66
67 // 3. GIAO DIEM & GIAO DOAN
68 // Kiem tra 2 doan AB va CD cat nhau:
69 bool segmentsIntersect(Point A, Point B, Point C, Point D) {
70     double o1 = cross(B-A, C-A);
71     double o2 = cross(B-A, D-A);
72     double o3 = cross(D-C, A-C);
73     double o4 = cross(D-C, B-C);
74
75     return o1 * o2 <= 0 && o3 * o4 <= 0;
76 }
77

```

```

79 // Tim giao diem 2 doan:
80 Point findIntersection(Point A, Point B, Point C, Point D) {
81     Point AB = B - A, CD = D - C, AC = C - A;
82     double t = cross(AC, CD) / cross(AB, CD);
83     return A + AB * t;
84 }
85
86 // 4. DIEN TICH & CHU VI
87 // Dien tich da giac (bat ky):
88 double polygonArea(vector<Point> &p) {
89     double area = 0;
90     int n = p.size();
91     for (int i = 0; i < n; i++) {
92         int j = (i + 1) % n;
93         area += (p[i].x - p[j].x) * (p[i].y + p[j].y);
94     }
95     return abs(area) / 2.0;
96 }
97
98 // Dau cua area:
99 // area > 0: da giac nguoc chieu kim dong ho (CCW)
100 // area < 0: da giac cung chieu kim dong ho (CW)
101
102 // Sap xep dinh da giac loi theo CCW:
103 Point center; // Tam trung binh
104 center.x = accumulate(p.begin(), p.end(), 0.0,
105 [](double s, Point pt) { return s + pt.x; }) / p.size();
106 center.y = accumulate(p.begin(), p.end(), 0.0,
107 [](double s, Point pt) { return s + pt.y; }) / p.size();
108
109 sort(p.begin(), p.end(), [&](Point a, Point b) {
110     return atan2(a.y - center.y, a.x - center.x) <
111             atan2(b.y - center.y, b.x - center.x);
112 });
113
114 // 5. CONG THUC BO SUNG
115 // Hypotenuse (canh huyen):
116 double hypot(double a, double b) {
117     return sqrt(a*a + b*b);
118 }
119
120 // a1x + b1y + c1 = 0
121 // a2x + b2y + c2 = 0
122 double D = a1*b2 - a2*b1;
123 double Dx = -c1*b2 + c2*b1;
124 double Dy = -a1*c2 + a2*c1;
125
126 // Giai he phuong trinh 2 an:
127 if (D != 0): x = Dx/D, y = Dy/D // 1 nghiem
128 if (D == 0 && Dx == 0 && Dy == 0): vo so nghiem
129 if (D == 0 && (Dx != 0 || Dy != 0)): vo nghiem

```

## 4 Graph

### 4.1 0-1 BFS

Listing 19: 0-1 BFS

```

1 // tim duong di ngan nhat tu s -> t
2 // neu gap '#' thi co the phai hoc 2 '#' voi chi phi la 1 va di qua no
3 const int dx[4] = {-1, 0, 0, 1};
4 const int dy[4] = {0, -1, 1, 0};
5 char A[1001][1001];
6 vector<vector<int>> dis(1001, vector<int>(1001, 1e9));
7 int n, m, a, b, c, d;
8 void solve()
9 {
10     cin >> n >> m;
11
12     for (int i = 0; i < n; i++) {
13         for (int j = 0; j < m; j++) {
14             cin >> A[i][j];
15         }
16     }
17     cin >> a >> b >> c >> d;
18     a--, b--, c--, d--;
19
20
21     deque<array<int, 3>> dq;
22     dq.push_front({0, a, b});
23     dis[a][b] = 0;
24     auto Invalid = [&](int x, int y) -> bool {
25         return x < 0 || x >= n || y < 0 || y >= m;
26     };
27     while (!dq.empty()) {
28         auto [d, x, y] = dq.front();
29         dq.pop_front();
30         if (d != dis[x][y])
31             continue;
32         for (int k = 0; k < 4; k++) {
33             bool wall = false;
34             for (int i = 1; i <= 2; i++) {
35                 int nx = x + dx[k] * i;
36                 int ny = y + dy[k] * i;
37                 if (Invalid(nx, ny))
38                     continue;
39                 int cost = (i == 1 ? d : d + 1);
40                 if (dis[nx][ny] <= cost)
41                     continue;
42                 if (A[nx][ny] == '#')
43                     wall = true;
44                 if (!wall) {
45                     if (i == 1) {
46                         dis[nx][ny] = d;
47                         dq.push_front({cost, nx, ny}); // khong co trong so
48                         break;
49                     }
50                 }
51             }
52             else {
53                 dis[nx][ny] = d + 1;
54                 dq.push_back({d + 1, nx, ny}); // co trong so
55             }
56         }
57     }
58     cout << (dis[c][d] == INF ? -1 : dis[c][d]) << endl;
59
60
61
62
63
64
65
66
67
68
69 // cho vi tri bat dau tai a[s][t] : duoc di len va xuong khong gioi han,
70 // nhung trai phai thi gioi han la x va y
71 // dem so luong o co the da quia voi so lan gioi han la x, y trai va phai
72 // voi bai toan, ta se day het so lan di qua trai truoc,
73 // roi sau do tinh gioi han duong di qua phai
74 int n, m, x, y, s, t, ans = 0;
75 vector<vector<int>> dis2(2002, vector<int>(2002, 1e9));
76 char a2[2002][2002];
77 const int dx2[4] = {-1, 0, 0, 1};
78 const int dy2[4] = {0, -1, 1, 0};
79
80 void solve2() {
81     cin >> n >> m >> s >> t >> x >> y;
82     for (int i = 0; i < n; i++) {
83         for (int j = 0; j < m; j++) {
84             cin >> a2[i][j];
85         }
86     }
87     s--, t--;
88     deque<array<int, 3>> dq;
89     dq.push_front({s, t, 0});
90     dis2[s][t] = 0;
91     auto Invalid = [&](int x, int y) -> bool {
92         return x < 0 || x >= n || y < 0 || y >= m;
93     };
94     while (!dq.empty()) {
95         auto [x, y, _] = dq.front();
96         dq.pop_front();
97         for (int k = 0; k < 4; k++) {
98             int nx = x + dx2[k];
99             int ny = y + dy2[k];
100            if (Invalid(nx, ny) || a2[nx][ny] == '*') continue;
101            int cost = dis2[x][y] + (k == 1 ? 1 : 0); // di qua trai => +
102            if (cost < dis2[nx][ny]) {
103                dis2[nx][ny] = cost;
104                if (k == 1) // qua trai
105                    dq.push_back({nx, ny, 0});
106                else // qua phai
107                    dq.push_front({nx, ny, 0});
108            }
109        }
110    }

```

```

110 }
111 for (int i = 0; i < n; i++) {
112     for (int j = 0; j < m; j++) {
113         if (dis2[i][j] == 1e9) continue;
114         int l = dis2[i][j];
115         int r = (j - t) + l; // so lan di trai it nhat
116         if (l <= x && r <= y)
117             ans++;
118     }
119 }
120 cout << ans << endl;
121 }
```

## 4.2 Bellman–Ford

Listing 20: Bellman–Ford

```

1 struct edge {
2     int u, v, w;
3 };
4 vector<ll> f(N + 5, oo), trace(N + 5, -1), cycle(N + 5, 0);
5 vector<edge> edges;
6 int n, m, x, y, w;
7 void findcycle(int start) {
8     vector<int> cur;
9     if(start == -1) {
10         cout << "NO\n";
11         return;
12     }
13     for (int i = 1; i <= n; i++) {
14         start = trace[start]; // dua ve chu trinh am, vi chua chac no nam trong chu trinh
15     }
16     cout << "YES\n";
17     cur.push_back(start);
18     int v = trace[start];
19     while (v != start) {
20         cur.push_back(v);
21         v = trace[v];
22     }
23     cur.push_back(start);
24     reverse(all(cur));
25     for(int v : cur) {
26         cout << v << ' ';
27     }
28     else;
29 }
30 void bellmanford(int st) {
31     f[st] = 0;
32     for(int i = 1; i <= n - 1; i++) {
33         for(auto e : edges) {
34             if(f[e.u] + e.w < f[e.v]) {
35                 f[e.v] = f[e.u] + e.w;
36                 trace[e.v] = e.u;
37             }
38         }
39     }
40     // sau n - 1 vongtoi uu, neu ma con co the toi uu nua => cycle am
41     // vong thu n, phat hien chu trinh am
42     int start = -1;
43     for(auto e : edges) {
44         if(f[e.u] != oo && f[e.u] + e.w < f[e.v]) {
45             cycle[e.u] = cycle[e.v] = 1;
46             trace[e.v] = e.u;
47             start = e.v;
48         }
49     }
50     findcycle(start);
51 }
52 void solve() {
53     cin >> n >> m;
54     for(int i = 0; i < m; i++) {
55         cin >> x >> y >> w;
56         edges.push_back(edge{x, y, w});
57     }
58     bellmanford(1);
59 }
```

## 4.3 Dijkstra

Listing 21: Dijkstra

```

1 vector<pair<int, int>> adj[N];
2 bool vis[N] = {};
3 int parent[N] = {};
4 const ll INF = 1e18;
5 void Dijkstra() {
6     priority_queue<pair<ll, int>, vector<pair<ll, int>>, greater<pair<ll, int>> Q;
7     vector<ll> step(n + 1, INF);
8     step[s] = 0; parent[s] = -1;
9     Q.push({0, s});
10    while(!Q.empty()) {
11        auto [du, u] = Q.top(); Q.pop();
12        if(vis[u]) continue;
13        vis[u] = true;
14        for(auto [w, v] : adj[u]) {
15            if(step[v] > step[u] + w) {
16                step[v] = step[u] + w;
17                parent[v] = u;
18                Q.push({step[v], v});
19            }
20        }
21    }
22    if(step[t] == INF) {
23        cout << -1 << "\n";
24        return;
25    }
26    cout << step[t] << "\n";
27    vector<int> path;
28    int cur = t;
29    while(cur != -1) {
30        path.pb(cur);
31        cur = parent[cur];
32    }
33    reverse(all(path));
34    for(int v : path) cout << v << ' ';
35 }
```

## 4.4 DSU

Listing 22: DSU

```

// join : gop, add tang diem, get : lay diem
// join : (a, b), add : tang moi diem gom ca con cua u, get lay diem dinh do
int findP(int u) {
    if (u == par[u]) return u;
    int p = par[u];
    par[u] = findP(par[u]);
    point[u] += point[p];
    return par[u];
}
void join(int u, int v) {
    u = findP(u);
    v = findP(v);
    if (u == v) return;
    if (sz[u] < sz[v]) swap(u, v);
    par[v] = u;
    point[v] = pc[v] - pc[u];
    sz[u] += sz[v];
}
void add(int u, int v) {
    u = findP(u);
    pc[u] += v;
}
long long get(int u) {
    int r = findP(u);
    return pc[r] + point[u];
}
void solve() {
    init();
    while (q--) {
        cin >> t;
        if (t == "join") {
            int x, y; cin >> x >> y;
        }
    }
}
```

```
34     join(x, y);
35 } else if (t == "add") {
36     int x, v; cin >> x >> v;
37     add(x, v);
38 } else if (t == "get") {
39     int x; cin >> x;
40     cout << get(x) << "\n";
41 }
42 }
```

## 4.5 Kruskal MST

Listing 23: Kruskal MST

```

1 struct item {
2     int x, y, w;
3     item() { }
4     item(int _x, int _y, int _w) : x(_x), y(_y), w(_w) { }
5 };
6
7 struct DSU {
8     int n;
9     vector<int> parent, size;
10    DSU(int _n) : n(_n), parent(_n + 1, 0), size(_n + 1, 0) { }
11    void make() {
12        iota(parent.begin() + 1, parent.end(), 1);
13    }
14    int findd(int u) {
15        return u == parent[u] ? u : parent[u] = findd(parent[u]);
16    }
17    bool unionn(int a, int b) {
18        a = findd(a);
19        b = findd(b);
20        if(a == b) { return false; }
21        if(size[a] < size[b]) { swap(a, b); }
22        parent[b] = a;
23        size[a] += size[b];
24        return true;
25    }
26
27 };
28
29 void solve() {
30     int n, m;
31     cin >> n >> m;
32     vector<item>a(m);
33     int cnt = 0, res = 0;
34     DSU dsu(n);
35     dsu.make();
36     for(int i = 0; i < m; i++) {
37         cin >> a[i].x >> a[i].y >> a[i].w;
38     }
39     sort(all(a), [] (item a, item b) {
40         return a.w < b.w;
41     });
42     for(int i = 0; i < m; i++) {
43         if(cnt == n - 1) { break; }
44         if(dsu.unionn(a[i].x, a[i].y)) {
45             res += a[i].w;
46             cnt++;
47         }
48     }
49     cout << res << endl;
50 }

```

## 4.6 LCA

Listing 24: LCA

```
//lca O(log(n))
void dfs(int u) {
    for(int v : adj[u]) {
        if(v != par[u][0]) {
```

```

        par[v][0] = u;
        high[v] = high[u] + 1;
        dfs(v);
    }
}
void build() {
    for(int j=1;j<=log;j++) {
        for(int i=1;i<=n;i++) (if(par[i][j-1])
            par[i][j] = par[par[i][j-1]][j-1];
    }
}
int jump(int u,int k) {
    for(int i=LOG-1;i>=0;i--) {
        if(MASK(1)<=k) {
            u = par[u][i];
            k -= MASK(i);
        }
    }
    return u;
}
int lca(int u,int v) {
    if(high[v] > high[u]) {
        return lca(v,u);
    }
    //can bang do xau
    for(int i=log;i>=0;i--) {
        if(high[par[u][i]] >= high[v]) {
            u = par[u][i];
        }
    }
    if(u == v) {
        return u;
    }
    for(int i=log;i>=0;i--) {
        if(par[u][i] != par[v][i]) {
            u = par[u][i];
            v = par[v][i];
        }
    }
    return par[u][0];
}

//lca O(1) truy van
void dfs(int u,int p) {
    nodes[++cnt] = u;
    pos[u] = cnt;
    for(int v : adj[u]){
        if(v != p){
            high[v] = high[u] + 1;
            dfs(v,u);
            nodes[++cnt] = u;
        }
    }
    fin[u] = cnt;
}

int minDepthHigh(int u,int v) {
    return (high[u] < high[v] ? u : v);
}

void build() {
    for(int i=1;i<=cnt;i++) {
        minHigh[i][0] = nodes[i];
    }

    for(int j=1;MASK(j) <=cnt;j++) {
        for(int i=1;i+MASK(j)-1<=cnt;i++) {
            minHigh[i][j] = minDepthHigh(minHigh[i][j-1],minHigh[i+MASK(j-1)][j-1]);
        }
    }
}

int lca(int u,int v) {
    int pu = pos[u];
    int pv = pos[v];
    if(pu > pv) swap(pu,pv);

    int k = 31 - __builtin_clz(pv-pu+1);
    return minDepthHigh(minHigh[pu][k],minHigh[pv-MASK(k)+1][k]);
}

```

## 4.7 Tarjan SCC

Listing 25: Tarjan SCC

```

1 int n, m, timer = 0, scc;
2 unordered_map<int, vector<int>>adj;
3 vector<vector<int>>SCC;
4 vector<int>disc, low, instack;
5 stack<int>st;
6 //disc : thu tu duyet
7 //low : thu tu duyet nho nhat
8 //instack : giong vis, kiem tra da duyet chua, neu da trong 1 tplt nao do roi thi bo qua, tranh vao
9 // truong hop else low[u] = min(low[u], disc[v]);
10 void dfs(int u) {
11     low[u] = disc[u] = ++timer;
12     st.push(u);
13     for(int i = 0; i < adj[u].size(); i++) {
14         int v = adj[u][i];
15         if(instack[v]) continue;
16         if(disc[v] == -1) { // canh thuoc cay dfs, chua duyet
17             dfs(v);
18             low[u] = min(low[u], low[v]);
19         } else { // canh khong thuoc cay dfs, duyet roi
20             low[u] = min(low[u], disc[v]);
21         }
22     }
23     if(low[u] == disc[u]) { // u la goc cua 1 tplt manh
24         vector<int>component;
25         while(1) {
26             int v = st.top(); st.pop();
27             component.push_back(v);
28             instack[v] = 1;
29             if(v == u) break;
30         }
31         SCC.push_back(component);
32     }
33 }
34 void solve() {
35     cin >> n >> m;
36     disc.assign(n + 1, -1);
37     low.assign(n + 1, -1);
38     instack.resize(n + 1);
39     for(int i = 0; i < m; i++) {
40         int x, y;
41         cin >> x >> y; //x++, y++;
42         adj[x].push_back(y);
43     }
44     for(int i = 1; i <= n; i++) {
45         if(disc[i] != -1) continue;
46         dfs(i);
47     }
48     for(auto it : SCC) {
49         for(auto v : it) {
50             cout << v << ' ';
51         }
52     }
53 }
54 cout << SCC.size();
55 }
```

## 4.8 Topological Sort

Listing 26: Topological Sort

```

1 void kahn(){
2     queue<int>q;
3     FOR(i, 1, n){
4         if(degree[i] == 0) q.push(i);
5     }
6     while(!q.empty()){
7         int u = q.front(); q.pop();
8         topo.push_back(u);
9         for(auto v : adj[u]){
10             degree[v]--;
11             if(degree[v] == 0) q.push(v);
12         }
13     }
14 }
```

```

13 }
14 if(topo.size() != n) cout << "Cycle!!!\n";
15 else for(auto x : topo) cout << x << ' ';
16 ed;
17 }
18 void solve(){
19     cin >> n >> m;
20     FOB(1, 0, m){
21         int x, y;
22         cin >> x >> y;
23         adj[x].pb(y);
24         degree[y]++;
25     }
26     kahn();
27 }
```

## 4.9 Euler Path

Listing 27: Euler Path

```

// Euler directed (Hierholzer)
// Input: n, edges list (u->v), 1-indexed
vector<int> euler_directed(int n, const vector<pair<int,int>>& edges) {
    int m = edges.size();
    vector<vector<int>> g(n+1);
    vector<int> out(n+1, 0), in(n+1, 0);
    for (int i = 0; i < m; ++i) {
        int u = edges[i].first, v = edges[i].second;
        g[u].push_back(i); // store edge index
        out[u]++;
        in[v]++;
    }
    // Check degree conditions
    int start = -1, cntStart=0, cntEnd=0;
    for (int v=1; v<=n; ++v) {
        if (out[v] - in[v] == 1) { cntStart++; start = v; }
        else if (in[v] - out[v] == 1) cntEnd++;
        else if (in[v] != out[v]) return vector<int>(); // impossible
    }
    if (!(cntStart==0 && cntEnd==0) || (cntStart==1 && cntEnd==1)) return vector<int>();
    if (start == -1) {
        // choose any vertex with out>0
        for (int v=1; v<=n; ++v) if (out[v] > 0) { start = v; break; }
        if (start == -1) return vector<int>(); // no edges
    }
    // Hierholzer with pointers on adjacency
    vector<int> it(n+1, 0);
    vector<int> st;
    vector<int> path;
    st.push_back(start);
    while (!st.empty()) {
        int v = st.back();
        if (it[v] < (int)g[v].size()) {
            int ei = g[v][it[v]++];
            int to = edges[ei].second;
            st.push_back(to);
        } else {
            path.push_back(v);
            st.pop_back();
        }
    }
    if ((int)path.size() != m+1) return vector<int>(); // not all edges used
    reverse(path.begin(), path.end());
    return path;
}

// Euler undirected (Hierholzer)
// Input: n = so dinh (1..n), edges = vector of (u,v) (1-indexed)
// Output: vector<int> path (list of vertices in order) or empty if none
vector<int> euler_undirected(int n, const vector<pair<int,int>>& edges) {
    int m = edges.size();
    vector<vector<pair<int,int>>> g(n+1); // (neighbor, edge_id)
    vector<int> deg(n+1, 0);
    vector<char> used(m, false);
    for (int i = 0; i < m; ++i) {
        int u = edges[i].first, v = edges[i].second;
        g[u].push_back((v,i));
        g[v].push_back((u,i));
        deg[u]++;
        deg[v]++;
    }
    // find start: if exists vertex with odd degree -> start there, else any vertex with degree>0
    int odd = 0, start = -1;
```

```

62     for (int v = 1; v <= n; ++v) {
63         if (deg[v] % 2 == 1) odd++;
64         if (deg[v] > 0 && start == -1) start = v;
65     }
66     if (start == -1) return vector<int>(); // no edges -> no meaningful path (could also return {1}
67     if (!(odd == 0 || odd == 2)) return vector<int>(); // no eulerian path/circuit
68     // Hierholzer
69     vector<int> it(n+1, 0);
70     vector<int> st;
71     vector<int> path;
72     st.push_back(start);
73     while (!st.empty()) {
74         int v = st.back();
75         while (it[v] < (int)g[v].size() && used[g[v][it[v]].second]) it[v]++;
76         if (it[v] == (int)g[v].size()) {
77             path.push_back(v);
78             st.pop_back();
79         } else {
80             auto [to, id] = g[v][it[v]];
81             used[id] = true;
82             st.push_back(to);
83         }
84     }
85     // path currently is vertices in reverse order; length should be m+1
86     if ((int)path.size() != m+1) return vector<int>(); // not all edges used -> no eulerian trail in
87     connected component
88     reverse(path.begin(), path.end());
89     return path;
90 }
91 // Example usage in main:
92 // int main() {
93 //     int n, m; cin >> n >> m;
94 //     vector<pair<int,int>> edges(m);
95 //     for (int i=0;i<m;i++) cin >> edges[i].first >> edges[i].second;
96 //     auto path = euler_undirected(n, edges);
97 //     if (path.empty()) { cout << "NO\n"; }
98 //     else {
99 //         for (int v: path) cout << v << ' ';
100 //         cout << '\n';
101 //     }

```

## 4.10 Floyd–Warshall

Listing 28: Floyd–Warshall

```

1 // floyd + dp bitmask
2 int n, ans = oo;
3 int dp[(1 << 21)][21];
4 int a[21][21];
5
6 void process() {
7
8     for(int i = 0; i <= n; i++)
9         for(int j = 0; j <= n; j++)
10            cin >> a[i][j];
11
12     for(int k = 0; k <= n; k++)
13         for(int i = 0; i <= n; i++)
14             for(int j = 0; j <= n; j++)
15                 a[i][j] = min(a[i][j], a[i][k] + a[k][j]);
16 }
17
18 void solve() {
19     cin >> n;
20     n = n * 2;
21     process();
22
23     for(int mask = 0; mask < (1 << n); mask++)
24         for(int i = 0; i <= n; i++)
25             dp[mask][i] = oo;
26
27     for(int i = 0; i < n / 2; i++) // 0 -> i
28         dp[(1 << i)][i + 1] = a[0][i + 1]; // (2 ^ 0) = a[1] thay vi = a[0]
29
30     for(int mask = 1; mask < (1 << n); mask++) {
31         for(int u = 1; u <= n; u++) { // i
32             if(!(mask & (1 << (u - 1)))) continue;
33             for(int v = 1; v <= n; v++) { // i + n

```

```

35             if((mask & (1 << (v - 1))))
36                 continue;
37             if(v > n / 2) {
38                 if(!(mask & (1 << (v - n / 2 - 1)))) {
39                     continue;
40                 }
41                 dp[mask | (1 << (v - 1))][v] = min(dp[mask | (1 << (v - 1))][v], dp[mask][u] + a[u][v]);
42             }
43         }
44     }
45
46     for(int i = 1; i <= n; i++)
47         ans = min(ans, dp[(1 << n) - 1][i] + a[i][0]);
48     cout << ans << endl;
49 }
50

```

## 4.11 Articulation Points & Bridges

Listing 29: Articulation Points & Bridges

```

1 int n, m;
2 vector<int> adj[MAXN];
3 pair<int, int> edges[MAXN];
4 int high[MAXN], low[MAXN], num[MAXN], fin[MAXN];
5 bool Bridge[MAXN];
6 int par[MAXN][LOG + 1];
7 int dsu[MAXN];
8 int cnt = 0;
9 // ===== DSU =====
10 int find_set(int u) {
11     if (dsu[u] < 0) return u;
12     return dsu[u] = find_set(dsu[u]);
13 }
14 bool unite(int u, int v) {
15     int x = find_set(u);
16     int y = find_set(v);
17     if (x == y) return false;
18
19     if (dsu[x] > dsu[y]) swap(x, y);
20     dsu[x] += dsu[y];
21     dsu[y] = x;
22     return true;
23 }
24 // ===== INPUT =====
25 void nhap() {
26     cin >> n >> m;
27     for (int i = 1; i <= m; i++) {
28         int u, v;
29         cin >> u >> v;
30         edges[i] = {u, v};
31         adj[u].push_back(i);
32         adj[v].push_back(i);
33     }
34 }
35 // ===== DFS =====
36 void dfs(int u) {
37     num[u] = low[u] = ++cnt;
38     for (int id : adj[u]) {
39         int v = edges[id].fi + edges[id].se - u;
40         if (!num[v]) {
41             par[v][0] = u;
42             high[v] = high[u] + 1;
43             dfs(v);
44             low[u] = min(low[u], low[v]);
45             if (low[v] > num[u]) {
46                 Bridge[id] = true;
47             }
48         } else if (v != par[u][0]) {
49             low[u] = min(low[u], num[v]);
50         }
51     }
52     fin[u] = cnt;
53 }
54 // ===== PREPARE =====
55 void prepare() {
56     fill(dsu, dsu + MAXN, -1);
57     for (int i = 1; i <= m; i++) {
58         unite(edges[i].fi, edges[i].se);
59     }

```

```

60     for (int i = 1; i <= n; i++) {
61         if (!num[i]) dfs(i);
62     }
63     for (int j = 1; j <= LOG; j++) {
64         for (int i = 1; i <= n; i++) {
65             if (par[i][j - 1]) {
66                 par[i][j] = par[par[i][j - 1]][j - 1];
67             }
68         }
69     }
70 }
71 // ===== SUBTREE CHECK =====
72 bool isSub(int u, int a) {
73     return (num[u] <= num[a]) && (num[a] <= fin[u]);
74 }
75 // ===== JUMP UP =====
76 int jump(int u, int k) {
77     for (int j = LOG; j >= 0; j--) {
78         if (MASK(j) <= k && par[u][j]) {
79             u = par[u][j];
80             k -= MASK(j);
81         }
82     }
83     return u;
84 }
85 // ===== SOLVE QUERY =====
86 bool solve(int u, int v, int a, int b) {
87     if (find_set(a) != find_set(b)) return false;
88     if (u > v) swap(u, v);
89     int id = lower_bound(edges + 1, edges + m + 1, make_pair(u, v)) - edges;
90     if (!Bridge[id]) return true;
91     int childA = (par[v][0] == u ? v : u);
92     bool inA = isSub(child, a);
93     bool inB = isSub(child, b);
94     return (inA == inB);
95 }
96 bool checkCut(int a, int b, int u) {
97     if (findSet(a) != findSet(b)) return false;
98     if (!isCut[u]) return true;
99     int childA = -1, childB = -1;
100    if (a != u && inSubtree(u, a)) {
101        childA = jump(a, high[a] - high[u] - 1);
102    }
103    if (b != u && inSubtree(u, b)) {
104        childB = jump(b, high[b] - high[u] - 1);
105    }
106    if (childA > 0 && low[childA] < num[u]) childA = -1;
107    if (childB > 0 && low[childB] < num[u]) childB = -1;
108    return (childA == childB);
109 }

```

## 4.12 Two-Sided Graph

Listing 30: Two-Sided Graph

```

1  bool dfs(int u, int col) {
2      color[u] = col;
3      for(auto v : adj[u]) {
4          if(color[v] == -1) {
5              if(!dfs(v, 1 - color[u])) return false;
6          } else {
7              if(color[v] == color[u]) return false;
8          }
9      }
10     return true;
11 }
12 void __init__() {
13     cin >> n >> m;
14     for(int i = 0; i < m; i++) {
15         int x, y;
16         cin >> x >> y;
17         adj[x].pb(y);
18         adj[y].pb(x);
19     }
20     color.assign(n + 1, -1);
21 }
22 void solve() {
23     __init__();
24     for(int i = 0; i < n; i++) {
25         if(color[i] == -1) {
26             if(!bfs(i)) // dfs(i, 0 or 1)

```

```

27             cout << "NO\n"; exit(0);
28         }
29     }
30 }
31 cout << "YES\n";
32 //ban dau to mau dinh i, trong dfs or bfs la 0 or 1 deu dc
33

```

## 4.13 Flows

### 4.13.1 Dinic's Max Flow

Listing 31: Dinic's Max Flow

```

1 #include<bits/stdc++.h>
2 #define ll long long
3 #define en "\n"
4
5 using namespace std;
6
7 struct edges{
8     int v,cap;
9     ll rev;
10 };
11
12 struct dinic{
13     int n;
14     vector<vector<edges>>adj;
15     vector<int>ptr;
16     vector<int>level;
17     dinic(int _n) : n(_n){
18         adj.assign(n+1,{});
19         level.assign(n+1,0);
20         ptr.assign(n+1,0);
21     }
22
23     void add(int u,int v,int cap){
24         edges a = {v,cap,(int)adj[v].size()};
25         edges b = {u,0,(int)adj[u].size()};
26         adj[u].push_back(a);
27         adj[v].push_back(b);
28     }
29
30     bool bfs(int s,int t){
31         fill(level.begin(),level.end(),-1);
32         queue<int>q;
33         q.push(s);
34         level[s] = 0;
35
36         while(!q.empty()){
37             int u = q.front();
38             q.pop();
39             for(auto &e : adj[u]){
40                 if(e.cap > 0 && level[e.v] == -1){
41                     level[e.v] = level[u] + 1;
42                     q.push(e.v);
43                 }
44             }
45         }
46         return level[t] != -1;
47     }
48
49     bool dfs(int u,int t,int pushed){
50         if(pushed == 0) return 0;
51         if(u == t) return pushed;
52
53         for(int &i = ptr[u];i<(int)adj[u].size();i++){
54             edges &e = adj[u][i];
55             if(level[e.v] != level[u] + 1 || e.cap <=0){
56                 continue;
57             }
58             ll tr = dfs(e.v,t,min(pushed,e.cap));
59             if(tr == 0){
60                 continue;
61             }
62             e.cap -= tr;
63             adj[e.v][e.rev].cap += tr;
64         }
65         return 0;
66     }

```

```

67 }
68
69 int maxflow(int s,int t){
70     int ans = 0;
71
72     while(bfs(s,t)){
73         fill(ptr.begin(),ptr.end(),0);
74         while(ll pushed = dfs(s,t,1e9)){
75             ans += pushed;
76         }
77     }
78     return ans;
79 }
80
81 int main(){
82     ios_base::sync_with_stdio(false);
83     cin.tie(nullptr);
84     int n,m,s,t;
85     cin >> n >> m >> s >> t;
86     dinic d(n);
87
88     for(int i=0;i<m;i++){
89         int u,v;
90         ll cap;
91         cin >> u >> v >> cap;
92         d.add(u,v,cap);
93     }
94
95     cout << d.maxflow(s,t) << en;
96 }
97

```

#### 4.13.2 Hopcroft–Karp Matching

Listing 32: Hopcroft–Karp Matching

```

46
47     for (int v : adj[u]) {
48         int nxt = pairV[v];
49         if (dist[nxt] == dist[u] + 1 && dfs(nxt)) {
50             pairU[u] = v;
51             pairV[v] = u;
52             return true;
53         }
54     }
55
56     dist[u] = INF; // mark dead-end
57     return false;
58 }
59
60 int hopcroftKarp() {
61     memset(pairU, 0, sizeof(pairU));
62     memset(pairV, 0, sizeof(pairV));
63
64     int matching = 0;
65     while (bfs()) {
66         for (int u = 1; u <= n; u++)
67             if (pairU[u] == 0 && dfs(u))
68                 matching++;
69     }
70     return matching;
71 }
72
73 int main() {
74     cin >> n >> m;
75     int edges; cin >> edges;
76     for (int i = 0; i < edges; i++) {
77         int u, v;
78         cin >> u >> v;
79         adj[u].push_back(v);
80     }
81
82     int ans = hopcroftKarp();
83     cout << "Maximum Matching = " << ans << endl;
84
85     for (int u = 1; u <= n; u++)
86         if (pairU[u])
87             cout << u << " - " << pairU[u] << endl;
88 }

```

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAXN = 1005; // so dinh ben trai (U)
5 const int MAXM = 1005; // so dinh ben phai (V)
6 const int INF = 1e9;
7
8 int n, m; // so dinh U, V
9 vector<int> adj[MAXN]; // danh sach ke tu U -> V
10 int pairU[MAXN], pairV[MAXM]; // cap ghep hien tai
11 int dist[MAXN]; // khoang cach BFS
12
13 bool bfs() {
14     queue<int> q;
15     for (int u = 1; u <= n; u++) {
16         if (pairU[u] == 0) { // dinh tu do ben U
17             dist[u] = 0;
18             q.push(u);
19         } else {
20             dist[u] = INF;
21         }
22     }
23     dist[0] = INF; // "dinh ao" dai dien cho cac dinh tu do ben V
24
25     while (!q.empty()) {
26         int u = q.front(); q.pop();
27
28         if (dist[u] < dist[0]) { // chi mo rong neu con hy vong tim duong
29             for (int v : adj[u]) {
30                 int nxt = pairV[v]; // dinh U dang ghep voi v (co the = 0)
31                 if (dist[nxt] == INF) {
32                     dist[nxt] = dist[u] + 1;
33                     q.push(nxt);
34                 }
35             }
36         }
37     }
38
39     // neu dist[0] != INF nghia la co it nhat 1 duong tang ton tai
40     return dist[0] != INF;
41 }
42
43 bool dfs(int u) {
44     if (u == 0) return true; // toi dinh ao = ket thuc 1 duong tang

```

## 5 Math

### 5.1 Combinatorics

Listing 33: Combinatorics

```

1 // combinatorics + modulo inverse
2 // problem : dem so luong day dai m, va max(subm) - min(subm) <= k
3 int n, m, k;
4 ll ans = 0;
5 vector<int> a(N), fact(N, 0), inv(N, 0);
6 unordered_map<int, int> freq;
7 ll powMod(ll a, ll b, ll m) {
8     ll res = 1;
9     while(b) {
10         if(b & 1) res = (res * a) % m;
11         a = (a * a) % m;
12         b = b >> 1;
13     }
14     return res;
15 }
16 ll inverse(ll a, ll m) {
17     return powMod(a, m - 2, m);
18 }
19 ll C(int n, int k) {
20     if(k > n)
21         return 0;
22     return fact[n] * (inv[k] % MOD * inv[n - k] % MOD) % MOD;
23 }
24 void init() {
25     fact[0] = inv[0] = 1;
26     for(int i = 1; i < N; i++) {
27         fact[i] = (fact[i - 1] * i) % MOD;
28         inv[i] = inverse(fact[i], MOD);
29     }
30 }

```

```

29     }
30 }
31 void solve() {
32     cin >> n >> m >> k;
33     ans = 0; freq.clear();
34     for(int i = 1; i <= n; i++) {
35         cin >> a[i], freq[a[i]]++;
36     }
37     sort(a.begin() + 1, a.begin() + 1 + n);
38     for(int l = 1; l <= n; l++) {
39         auto r = upper_bound(a.begin() + 1, a.begin() + 1 + n, a[l] + k) - a.begin(); // 2 pointer or bs
40         ans = (ans + C(r - l - 1, m - 1)) % MOD; // so cach chon m - 1 ptu tu day dai r - l + 1, de ghep
41         voi a[i]
42     }
43     cout << ans << endl;
44 }
45 // combinatorics + for
46 long long res = 1;
47 for (int i = 1; i <= k; i++) {
48     res = res * (n - i + 1); CnK, AnK
49     res /= i; // CnK
50 }

```

## 5.2 Divisor / Factors

Listing 34: Divisor / Factors

```

1 // tong uoc so
2 long long sumOfDivisors(int n) {
3     long long sum = 1;
4     for (int p : primes) {
5         if (p * p > n) break;
6         if (n % p == 0) {
7             int cnt = 0;
8             long long power = 1;
9             while (n % p == 0) {
10                 cnt++;
11                 power *= p;
12                 n /= p;
13             }
14             sum *= (power * p - 1) / (p - 1);
15         }
16     }
17     if (n > 1) sum *= (1 + n);
18     return sum;
19 }
20 // dem uoc so
21 int countDivisors(int n) {
22     int cnt = 1;
23     for (int p : primes) {
24         if (p * p > n) break;
25         if (n % p == 0) {
26             int power = 0;
27             while (n % p == 0) {
28                 power++;
29                 n /= p;
30             }
31             cnt *= (power + 1);
32         }
33     }
34     if (n > 1) cnt *= 2;
35     return cnt;
36 }

```

## 5.3 Euler's Totient

Listing 35: Euler's Totient

```

1 const int MAXN = 1000001;
2 long long p[MAXN];
3 void fill() {
4     for(int i = 1; i < MAXN; i++) p[i] = i;

```

```

5     for(int i = 2; i < MAXN; i++) {
6         if(p[i] == i) { // i la snt
7             p[i] = i - 1; // phi(i) = p-1
8             for(int j = 2 * i; j < MAXN; j += i) {
9                 p[j] = p[j] - (p[j] / i);
10            }
11        }
12    }
13 }
14 long long phi(long long n) {
15     long long result = n;
16     for (long long p = 2; p * p <= n; p++) {
17         if (n % p == 0) {
18             while (n % p == 0) n /= p;
19             result -= result / p;
20         }
21     }
22     if (n > 1) result -= result / n;
23     return result;
24 }

```

## 5.4 Sieve

Listing 36: Sieve

```

1 // sang Eratosthenes
2 const int MAXN = 1e6 + 5;
3 bool isPrime[MAXN];
4 vector<int> primes;
5 void sieve() {
6     fill(isPrime, isPrime + MAXN, true);
7     isPrime[0] = isPrime[1] = false;
8     for (int i = 2; i * i < MAXN; i++) {
9         if (isPrime[i]) {
10             for (int j = i * i; j < MAXN; j += i) {
11                 isPrime[j] = false;
12             }
13         }
14     }
15     for (int i = 2; i < MAXN; i++) {
16         if (isPrime[i]) primes.push_back(i);
17     }
18 }
19
20 // phan tich thua so nt
21 vector<pair<int, int>> factorize(int n) {
22     vector<pair<int, int>> factors;
23     for (int p : primes) {
24         if (p * p > n) break;
25         if (n % p == 0) {
26             int cnt = 0;
27             while (n % p == 0) {
28                 cnt++;
29                 n /= p;
30             }
31             factors.push_back({p, cnt});
32         }
33     }
34     if (n > 1) factors.push_back({n, 1});
35     return factors;
36 }

```

## 5.5 Prime Numbers

Listing 37: Prime Numbers

```

1 bool isPrime(long long n) {
2     if (n < 2) return false;
3     if (n == 2 || n == 3) return true;
4     if (n % 2 == 0 || n % 3 == 0) return false;
5
6     for (long long i = 5; i * i <= n; i += 6) {
7         if (n % i == 0 || n % (i + 2) == 0) return false;

```

```

8     }
9     return true;
10    }
11
12 // Miller-Rabin cho so lon (optional)
13 bool millerTest(long long d, long long n) {
14     long long a = 2 + rand() % (n - 4);
15     long long x = power(a, d, n);
16     if (x == 1 || x == n - 1) return true;
17
18     while (d != n - 1) {
19         x = (x * x) % n;
20         d *= 2;
21         if (x == 1) return false;
22         if (x == n - 1) return true;
23     }
24     return false;
25 }
26
27 bool isPrimeMR(long long n, int k = 5) {
28     if (n <= 1 || n == 4) return false;
29     if (n <= 3) return true;
30
31     long long d = n - 1;
32     while (d % 2 == 0) d /= 2;
33
34     for (int i = 0; i < k; i++) {
35         if (!millerTest(d, n)) return false;
36     }
37     return true;
38 }

```

## 5.6 Math Formulae

Listing 38: Math Formulae

```

1 // tong 1 + 2 + ... + n
2 long long sumtoN(long long n) {
3     return n * (n + 1) / 2;
4 }
5 // tong binh phuong 1^2 + 2^2 + ... + n^2
6 long long sumSquares(long long n) {
7     return n * (n + 1) * (2 * n + 1) / 6;
8 }
9 // tong lap phuong 1^3 + 2^3 + ... + n^3
10 long long sumCubes(long long n) {
11     long long s = sumtoN(n);
12     return s * s;
13 }
14
15 // euclid + euclid extend
16 int gcd(int a, int b) {
17     return b == 0 ? a : gcd(b, a % b);
18 }
19
20 int lcm(int a, int b) {
21     return a / gcd(a, b) * b;
22 }
23
24 // int x, y, d;
25 // void euclid(int a, int b) {
26 //     if(b == 0) {
27 //         x = 1; y = 0;
28 //         d = a; // ucln
29 //         return;
30 //     }
31 //     euclid(b, a % b);
32 //     int tmp = x;
33 //     x = y;
34 //     y = tmp - a / b * y;
35 // }
36
37 // inverse modulo
38 // ax + my = 1
39 // ax % m + my % m = 1 % m
40 // ax % m = 1
41
42 // iu kin t n t i : gcd ( , ) = 1 gcd(a,m)=1 ( t c l a v m nguy n t
43 // C1 : Euclid mo rong
44 // int x, y, d;

```

```

45 // void ecuclid(int a, int b) {
46 //     if(b == 0) {
47 //         x = 1; y = 0;
48 //         d = a; // ucln
49 //         return;
50 //     }
51 //     ecuclid(b, a % b);
52 //     int tmp = x;
53 //     x = y;
54 //     y = tmp - a / b * y;
55 // }
56
57 // void inverse(int a, int m) {
58 //     ecuclid(a, m);
59 //     if(d != 1) {
60 //         cout << "-1" << endl;
61 //     } else {
62 //         cout << (d % MOD + MOD) % MOD;
63 //     }
64 // }
65
66 // C2 : fermat nho(phi ham euler)
67 // voi m la 1 so nguyen to
68 ll powMod(ll a, ll b, ll m) {
69     ll res = 1;
70     while(b) {
71         if(b & 1) res = (res * a) % m;
72         a = (a * a) % m;
73         b = b >> 1;
74     }
75     return res;
76 }
77
78 ll inverse(ll a, ll m) {
79     return powMod(a, m - 2, m);
80 }
81
82 // a * x % m = 1
83 // 5 * 2 % 9 = 1
84 void solve() {
85     ll a, m;
86     cin >> a >> m;
87     cout << inverse(a, m);
88 }

```

## 6 Strings

### 6.1 KMP Search

Listing 39: KMP Search

```

1 // KMP - O(n+m)
2 int lps[100005];
3 void kmp(string s, string t) {
4     int n = s.size(), m = t.size();
5     // Build LPS
6     for(int i = 1, j = 0; i < m; i++) {
7         while(j && s[i] != t[j]) j = lps[j-1];
8         if(s[i] == t[j]) lps[i] = ++j;
9     }
10    // Search
11    for(int i = 0, j = 0; i < n; i++) {
12        while(j && s[i] != t[j]) j = lps[j-1];
13        if(s[i] == t[j]) j++;
14        if(j == m) {
15            // Found at position i-m+1
16            j = lps[j-1];
17        }
18    }
}

```

### 6.2 String Hashing

Listing 40: String Hashing

```

1 // Dem so luong xau con cua S la palindrome.
2 const int BASE = 31;
3 struct Hash {
4     vector<ll> hash, pw;
5     int n;
6
7     inline int gc(char c) { return c - 'a' + 1; }
8
9     Hash(string s) {
10        s = '@' + s;
11        n = s.size() - 1;
12        hash.assign(n + 1, 0);
13        pw.assign(n + 1, 1);
14        for(int i = 1; i <= n; i++) {
15            hash[i] = (hash[i - 1] * BASE + gc(s[i])) % MOD;
16            pw[i] = (pw[i - 1] * BASE) % MOD;
17        }
18    }
19
20    ll getHash(int l, int r) {
21        return (hash[r] - hash[l - 1] * pw[r - l + 1] % MOD + MOD) % MOD;
22    }
23};
24
25 void solve() {
26     string s; cin >> s;
27     int n = s.size();
28
29     string t = s;
30     reverse(t.begin(), t.end());
31
32     Hash h(s), hr(t);
33
34     auto valid = [&s](int l, int r) -> bool {
35         int L = n - r + 1, R = n - l + 1;
36         return h.getHash(l, r) == hr.getHash(L, R);
37     };
38
39     ll ans = 0;
40
41     // chuo i le, nam giua i - 1 va i + 1
42     for(int i = 1; i <= n; i++) {
43         int lo = 0, hi = min(i - 1, n - i), res = 0;
44         while (lo <= hi) {
45             int mid = (lo + hi) / 2;
46             if (valid(i - mid, i + mid)) {
47                 res = mid;
48                 lo = mid + 1;
49             } else hi = mid - 1;
50         }
51         ans += res + 1; // aba : b, aba
52     }
53
54     // chuo i chan, nam giua i va i + 1
55     for(int i = 1; i < n; i++) {
56         int lo = 0, hi = min(i, n - i), res = 0;
57         while (lo <= hi) {
58             int mid = (lo + hi) / 2;
59             if (valid(i - mid + 1, i + mid)) {
60                 res = mid; // ban kinh lon nhat ma tai do doan s[i - res ... i + res] van la
61                 // palindrome, va tam la i
62                 lo = mid + 1;
63             } else hi = mid - 1;
64         }
65         ans += res; // aa : aa
66     }
67
68     cout << ans << "\n";
69 }
70
71 // Tim xau con dai nhat cua S ma co the duoc ghep boi cac xau T1, T2, ...Tn, Mot xau co the duoc su
72 // dung nhieu lan.
73 string s, t;
74 int n;
75 unordered_set<ll>dict;
76 vector<int>sub = {0};
77 void solve() {
78     cin >> n >> s;
79
80     Hash h(s);
81     for(int i = 0; i < n; i++) {
82         cin >> t;
83         Hash su(t);
84         dict.insert(su.getHash(1, t.length()));
85         sub.push_back(t.size());
86     }
87 }
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
}

```

```

84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102

```

### 6.3 Trie

Listing 41: Trie

```

1 // trie
2 const int MAXN = 1000005; // max number of nodes
3 int child[MAXN][26]; // child[u][c] = next node from u by character c
4 bool isEnd[MAXN]; // mark end of a word
5 int cnt[MAXN]; // count words passing through node
6 int nodeCount = 1; // 0 is root
7
8 // insert word
9 void insert(const string &word) {
10    int u = 0;
11    for (char c : word) {
12        int pos = (c - 'a');
13        if (!child[u][pos])
14            child[u][pos] = nodeCount++;
15        u = child[u][pos];
16        cnt[u]++;
17    }
18    isEnd[u] = true;
19}
20
21 // check if word exists
22 bool search(const string &word) {
23    int u = 0;
24    for (char c : word) {
25        int pos = (c - 'a');
26        if (!child[u][pos])
27            return false;
28        u = child[u][pos];
29    }
30    return isEnd[u]; // check if it ends a word
31}
32
33 // check if prefix exists
34 bool prefix(const string &word) {
35    int u = 0;
36    for (char c : word) {
37        int pos = (c - 'a');
38        if (!child[u][pos])
39            return false;
40        u = child[u][pos];
41    }
42    return true;
43}
44
45 // count words with prefix s
46 int countPrefix(const string &s) {
47    int u = 0;
48    for (char c : s) {
49        int pos = (c - 'a');
50        if (!child[u][pos])
51            return 0;
52        u = child[u][pos];
53    }
54    return cnt[u];
55}
56
57 // internal recursive helper to delete a word

```

```

58 bool process(int u, const string &s, int i) {
59     if (i == (int)s.size()) {
60         if (!isEnd[u]) return false; // word does not exist
61         isEnd[u] = true;
62     } else {
63         int pos = s[i] - 'a';
64         int v = child[u][pos];
65         if (v == 0) // not the target word
66             return false;
67         bool childDeleted = process(v, s, i + 1);
68         if (childDeleted)
69             child[u][pos] = 0;
70     }
71     if (u != 0) {
72         cnt[u]--;
73         // can delete node if no word passes and not end
74         return cnt[u] == 0 && isEnd[u] == 0;
75     }
76     return false;
77 }
78 // delete word from trie
79 void deleteWord(const string &word) {
80     if (!search(word)) return;
81     process(0, word, 0);
82 }

```

## 7 Misc

### 7.1 Bitset tricks

Listing 42: Bitset tricks

```

1 // thao tac bit
2
3 // 1 : lay cac bit trong tap mask
4 int m = mask;
5 while (m)
6 {
7     int bit = m & -m; // lay bit thap nhat bat
8     int pos = __builtin_ctz(m);
9     cout << pos << " ";
10    cout << bitset<10>(pos) << endl; // in bitset
11    m -= bit; // tat bit do
12 }
13
14 // 2 : kiem tra x la tap con y
15 if ((x & y) == x) { // x la tap con cua y }
16
17 // 3 : duyet qua moi subset cua mask
18 for (int mask = 0; mask < (1 << n); mask++) {
19     for (int sub = mask; sub > 0; sub = (sub - 1) & mask) { // sub: tap con
20         int rest = sub ^ mask; // rest: lay phan bu trong mask (tuc nhung phan tu cua mask khong co
21         // trong sub)
22         if (dp[rest] != INT32_MAX && dp[sub] != INT32_MAX) { // 7 => sub: 6 (110), rest: 1 (001)
23             dp[mask] = min(dp[mask], dp[sub] + dp[rest]);
24     }
25 }
26
27 // 4 : cac ham thao tac bit
28 // __builtin_ctz(x): lay vi tri bit thap nhat
29 // __builtin_popcount(x): dem so luong bit trong mask
30
31 // tim vi tri cua bit cao nhat
32 // int pos = 31 - __builtin_clz(a); // vi tri MSB
33 // a ^= (1 << pos); // tat MSB
34
35 // a &= (a - 1); xoa bit 1 thap nhat
36 // x = a & ~a: giu lai bit 1 thap nhat (co the hieu la tat moi bit tru bit thap nhat)
37
38 // 1 bai toan khac ve bit
39 void solve() {
40     int N, C;
41     cin >> N >> C;
42     vector<int> T(N), A(N);
43     for (int i = 0; i < N; i++) {
44         cin >> T[i] >> A[i];

```

```

45     }
46     vector<int> ans(N);
47     vector<int> f0(30, 0), f1(30, 1);
48     for (int i = 0; i < N; i++) {
49         int x = 0;
50         for (int k = 0; k < 30; k++) {
51             int a = (A[i] >> k) & 1;
52             if (T[i] == 1) { // AND
53                 f0[k] = f0[k] & a;
54                 f1[k] = f1[k] & a;
55             } else if (T[i] == 2) { // OR
56                 f0[k] = f0[k] | a;
57                 f1[k] = f1[k] | a;
58             } else { // XOR
59                 f0[k] = f0[k] ^ a;
60                 f1[k] = f1[k] ^ a;
61             }
62             int bit = (C >> k) & 1;
63             if (bit == 0) {
64                 x |= (f0[k] << k);
65             } else {
66                 x |= (f1[k] << k);
67             }
68         }
69         C = x;
70         ans[i] = C;
71     }
72     for (int i = 0; i < N; i++) {
73         cout << ans[i] << endl;
74     }
75 }

```

### 7.2 Ternary Search

Listing 43: Ternary Search

```

1 // ternary search : Giang nhu binary search, nhung thay vi chia doi, ta chia doan [l, r] thanh 3
2 // phan:
3 mid1 = l + (r - l) / 3;
4 mid2 = r - (r - l) / 3;
5 // Neu f(mid1) < f(mid2) => dinh nam ben phai, loai bo doan [l, mid1].
6 // Nguoc lai => dinh nam ben trai, loai bo doan [mid2, r].
7
8 // int TernarySearch(int a[], int n, int target) {
9 //     int l = 0, r = n - 1;
10 //     while (l <= r) {
11 //         int mid1 = l + (r - l) / 3;
12 //         int mid2 = r - (r - l) / 3;
13 //         if (a[mid1] == target) return mid1;
14 //         if (target < a[mid1]) r = mid1 - 1;
15 //         else if (target > a[mid2]) l = mid2 + 1;
16 //         else { // nam o giao
17 //             l = mid1 + 1;
18 //             r = mid2 - 1;
19 //         }
20 //     }
21 //     return -1;
22 // }
23
24 // Giang nhu do thi parabol
25 long long f(long long k, int n, int a, int b) {
26     long long cake = (n - k) * a;
27     long long cur = 2 * b - k;
28     long long cake_ = k * (cur + 1) / 2;
29     return cake + cake_;
30 }
31
32 void solve() {
33     cin >> n >> a >> b;
34     int l = 0, r = min(n, b);
35     while (r - l > 2) {
36         int mid1 = l + (r - l) / 3;
37         int mid2 = r - (r - l) / 3;
38         if (f(mid1, n, a, b) > f(mid2, n, a, b))
39             r = mid2;
40         else
41             l = mid1;
42     }
43     long long ans = 0;
44     for (int i = l; i <= r; i++) // ket qua nam trong doan l -> r

```

```

45     ans = max(ans, f(i, n, a, b));
46     cout << ans << endl;
47 }
48
49 double f(double x) {
50     double mi = 1e18 + 1, mx = 0;
51     FOR(i, 1, n) {
52         mi = min(mi, x * a[i] + b[i]);
53         mx = max(mx, x * a[i] + b[i]);
54     }
55     return mx - mi;
56 }
57
58 const double epsilon = 1e-9; // sai so 0.00000001
59 void find() {
60     double l = 0, r = k;
61     while (r - l > epsilon) {
62         double mx1 = l + (r - l) / 3.0;
63         double mx2 = r - (r - l) / 3.0;
64         if (f(mx1) > f(mx2)) l = mx1;
65         else r = mx2;
66     }
67     cout << fixed << setprecision(6) << f(l) << endl;
68 }
69
70 FOR(i, 1, n) cin >> a[i] >> b[i];
71 // print 6 chu so => +3 => 1e-9
72 // print 9 chu so => +3 => 1e-12

```

## 7.3 STL Reference

Listing 44: STL Reference

```

1 // custom priority_queue
2 struct item {
3     int a, b, c;
4     item() {}
5     item(int _a, int _b, int _c) : a(_a), b(_b), c(_c) {}
6     bool operator > (const item &other) const { // priority nguoc lai voi bthuong
7         if (a != other.a) return a > other.a;
8         if (b != other.b) return b > other.b;
9         return c > other.c;
10    }
11 }
12 // bool operator < (const item &a, const item &b) {      }
13 priority_queue<item, vector<item>, greater<item>>pq;
14 struct point {
15     int x, y;
16     point() {}
17     point(int _x, int _y) : x(_x), y(_y) {}
18     point operator + (const point &a) {
19         return point(x * a.x, y * a.y);
20     }
21 };
22 int operator + (ii a, ii b) { return (a.fi + a.se) * (b.fi + b.se); }
23 void solve() {
24     ii a = {1, 2}, ii b = {2, 3};
25     cout << a + b << endl;
26     point p1(1, 2), point p2(2, 3);
27     point p3 = p1 + p2;
28 }
29
30 // tim va thay the
31 size_t pos = s.find(oldStr);
32 if (pos != string::npos) {
33     s.replace(pos, oldStr.length(), newStr);
34 }
35 // bitset : ko biet dung nen ghi hoi nhieu phan nay:D
36 void solve1()
37 {
38     bitset<8> b(13); // 00001101
39     cout << b.count() << "\n"; // 3 bit 1
40     cout << b.any() << "\n"; // true
41     cout << b.to_ulong() << "\n"; // 13
42     cout << b.to_string() << "\n"; // "00001101"
43 }
44
45 // if ((A & B) == A) cout << "A la subset cua B";
46 // if ((A & B).any()) cout << "Co phan giao";
47 void solve2()
48 {

```

```

49
50     bitset<8> b("10100010");
51     cout << "Ban dau: " << b << "\n";
52     b.flip(0); // dao bit thu 0
53     cout << "Sau flip(0): " << b << "\n";
54     b.set(3);
55     cout << "Sau set(3): " << b << "\n";
56     cout << "So bit 1: " << b.count() << "\n";
57     cout << "Gia tri thap phan: " << b.to_ulong() << "\n";
58 }
59
60 void solve3()
61 {
62     // bitset<10>a("100");
63     // bitset<10>b("101");
64     // if ((a & b).any()) cout << "Hai tap co giao\n";
65     bitset<8> mat[5];
66     mat[0][1] = 1;
67     mat[2][5] = 1;
68     for (int i = 0; i < 5; i++)
69         cout << mat[i] << "\n";
70
71     //-----
72     const int X = 3, Y = 3;
73     bitset<8> cube[X][Y]; // "bitset 3D" (X, Y, bit_index)
74     cube[0][0][2] = 1; // dat bit thu 2 o lon (0,0)
75     cube[1][2][5] = 1;
76     cube[2][1][7] = 1;
77     // In toan bo khai
78     for (int i = 0; i < X; i++)
79     {
80         cout << "Lop " << i << ":\n";
81         for (int j = 0; j < Y; j++)
82             cout << cube[i][j] << "\n";
83         cout << "\n";
84     }
85
86 // STL :
87 // -----
88 // Vector tricks
89 // -----
90 void vector_tricks() {
91     vector<int> v = {5, 2, 9, 1};
92     v.push_back(7);
93     v.pop_back();
94     sort(v.begin(), v.end());
95     reverse(v.begin(), v.end());
96     auto it = lower_bound(v.begin(), v.end(), 3); // >=3
97     if (it != v.end()) cout << "lower_bound = " << *it << endl;
98     v.erase(unique(v.begin(), v.end()), v.end()); // remove duplicates
99     cout << "Vector: ";
100    for (int x : v) cout << x << " ";
101    cout << endl;
102 }
103
104 // Deque tricks
105 // -----
106 void deque_tricks() {
107     deque<int> dq;
108     dq.push_back(1);
109     dq.push_front(2);
110     cout << "Deque front: " << dq.front() << " back: " << dq.back() << endl;
111     dq.pop_back();
112     dq.pop_front();
113     dq.emplace_back(5);
114     dq.emplace_front(6);
115     // sliding window maximum (example)
116     vector<int> a = {1, 3, 2, 5, 4};
117     int k = 3;
118     deque<int> q;
119     cout << "Sliding window maximum: ";
120     for (int i = 0; i < a.size(); i++) {
121         while (!q.empty() && a[q.back()] <= a[i]) q.pop_back();
122         q.push_back(i);
123         if (q.front() == i - k) q.pop_front();
124         if (i >= k - 1) cout << a[q.front()] << " ";
125     }
126     cout << endl;
127 }
128
129 // Set / Multiset tricks
130 // -----
131 void set_tricks() {
132     set<int> s = {3, 1, 4};
133     s.insert(2);
134     s.erase(1);
135     cout << (s.count(3) ? "3 exists\n" : "3 not exists\n");
136     auto it = s.lower_bound(2); // >=2
137     if (it != s.end()) cout << "lower_bound = " << *it << endl;

```

```

138
139     cout << "Set: ";
140     for (int x : s) cout << x << " ";
141     cout << endl;
142 }
143 void multiset_tricks() {
144     multiset<int> ms = {3, 1, 3};
145     ms.insert(2);
146     ms.erase(ms.find(3)); // remove only one occurrence
147     cout << "Multiset: ";
148     for (int x : ms) cout << x << " ";
149     cout << endl;
150 }
151 // -----
152 // Map / Unordered map tricks
153 // -----
154 void map_tricks() {
155     map<string, int> mp;
156     mp["a"] = 5;
157     mp["b"] = 2;
158     // lower_bound / upper_bound in map
159     auto it = mp.lower_bound("b"); // first key >= "b"
160     if (it != mp.end()) cout << it->first << " -> " << it->second << endl;
161     mp.erase("b");
162     cout << "Map: ";
163     for (auto &pair : mp) {
164         cout << pair.first << " -> " << pair.second << " ";
165     }
166     cout << endl;
167 }
168 void unordered_map_tricks() {
169     unordered_map<int, int> ump;
170     ump[1] = 10;
171     ump[2] = 20;
172     // default value trick
173     cout << "ump[3] = " << ump[3] << endl; // 0 if not exist
174     cout << "Unordered map: ";
175     for (auto &pair : ump) {
176         cout << pair.first << " " << pair.second << " ";
177     }
178     cout << endl;
179 }
180 // -----
181 // Stack / Queue / Priority Queue
182 // -----
183 void stack_queue_tricks() {
184     stack<int> st;
185     st.push(1);
186     st.push(2);
187     cout << "Stack top: " << st.top() << endl;
188     st.pop();
189     queue<int> q;
190     q.push(5);
191     q.push(6);
192     cout << "Queue front: " << q.front() << " back: " << q.back() << endl;
193     q.pop();
194     // max-heap (default)
195     priority_queue<int> pq;
196     pq.push(3);
197     pq.push(1);
198     pq.push(4);
199     cout << "Max-heap top: " << pq.top() << endl;
200     pq.pop();
201     // min-heap
202     priority_queue<int, vector<int>, greater<int>> pq_min;
203     pq_min.push(3);
204     pq_min.push(1);
205     pq_min.push(4);
206     cout << "Min-heap top: " << pq_min.top() << endl;
207     pq_min.pop();
208     // pair priority queue (min-heap based on first element)
209     priority_queue<pii>, vector<pii>, greater<pii>> pqp;
210     pqp.push({2, 5});
211     pqp.push({1, 7});
212     cout << "Pair priority queue top: " << pqp.top().first << " " << pqp.top().second << endl;
213 }
214 // -----
215 // Bitset tricks
216 // -----
217 void bitset_tricks() {
218     bitset<10> b;
219     b.set(2);
220     b.set(5, 1);
221     b.reset(2);
222     b.flip(5);
223     cout << "Bitset: " << b << endl;
224     cout << "Count of 1s: " << b.count() << endl;
225 }
226 // -----
227 // General STL tricks
228 // -----
229 void general_tricks() {
230     vector<int> v(5);
231     iota(v.begin(), v.end(), 1); // fill 1..5
232     reverse(v.begin(), v.end());
233     random_device rd;
234     mt19937 g(rd());
235     shuffle(v.begin(), v.end(), g);
236     cout << "Shuffled: ";
237     for (int x : v) cout << x << " ";
238     cout << endl;
239 }
240
241 // demo bai toan vd su dung stl
242 void solve() {
243     cin >> n;
244     a.assign(n + 1, 0);
245     for(int i = 0; i < n; i++)
246         cin >> a[i];
247
248     for(int i = 0; i < n; i++) {
249         int start = a[i];
250         vector<int> sq = {start};
251         unordered_set<int> used;
252         used.insert(start);
253         while(sq.size() < n) {
254             int x = sq.back();
255             bool found = false;
256             if(x % 3 == 0 && !used.count(x / 3) && find(all(a), x / 3) != a.end()) {
257                 found = true;
258                 used.insert(x / 3);
259                 sq.push_back(x / 3);
260             } else if(!used.count(x * 2) && find(all(a), x * 2) != a.end()) {
261                 found = true;
262                 used.insert(x * 2);
263                 sq.push_back(x * 2);
264             }
265             if(found == false)
266                 break;
267         }
268         if(sq.size() == n)
269             for(int x : sq)
270                 cout << x << ' ';
271     }
272 }
273
274 }

```

## 7.4 Python Utilities

Listing 45: Python Utilities

```

1 # ===== ICPC Python Template ===== #
2 import sys
3 import math
4 import bisect
5 from collections import *
6 from itertools import *
7 from heapq import *
8 from functools import *
9 input = sys.stdin.readline # faster input
10
11 # ----- BASIC UTILITIES -----
12 INF = 10**18
13 MOD = 10**9 + 7
14 YES = lambda: print("YES")
15 NO = lambda: print("NO")
16
17 # ----- INPUT -----
18 n = int(input())
19 arr = list(map(int, input().split()))
20 a, b, c = map(int, input().split())
21 s = input().strip()
22
23 # read multiple lines quickly
24 data = [list(map(int, input().split())) for _ in range(n)]
25
26 # ----- OUTPUT -----
27 # print(*arr)
28 # print("Case #{}: {}".format(t, ans))

```

```

29
30 # ----- MATH -----
31 def ceil_div(a, b): return -(a // b)
32 def gcd(a, b): return math.gcd(a, b)
33 def lcm(a, b): return a * b // gcd(a, b)
34 def modinv(a, mod=MOD): return pow(a, mod - 2, mod)
35
36 # ----- LIST / SET / MAP -----
37 # list comprehension
38 # arr = [x*x for x in range(10)]
39 # set comprehension
40 # s = {x % 3 for x in arr}
41 # dict comprehension
42 # mp = {x: x*x for x in arr}
43
44 # set usage
45 # s = set()
46 # s.add(x)
47 # if x in s: ...
48 # s.remove(x)
49
50 # dict (map)
51 # mp = defaultdict(int)
52 # mp['abc'] += 1
53 # for k, v in mp.items(): ...
54
55 # Counter
56 # cnt = Counter(arr)
57 # cnt.most_common(3)
58
59 # ----- HEAP -----
60 # min heap
61 # heap = []
62 # heappush(heap, x)
63 # x = heappop(heap)
64 # max heap (negate values)
65 # heappush(heap, -x)
66 # x = -heappop(heap)
67
68 # ----- SORT / BISECT -----
69 # arr.sort(key=lambda x: (x[0], -x[1]))
70 # idx = bisect_left(arr, val)
71 # bisect_right(arr, val)
72
73 # ----- GRAPH -----
74 # G = [[] for _ in range(n+1)]
75 # for _ in range(m):
76 #     u, v = map(int, input().split())
77 #     G[u].append(v)
78 #     G[v].append(u)
79
80 # BFS
81 # from collections import deque
82 # q = deque([start])
83 # dist = [-1]*(n+1)
84 # dist[start] = 0
85 # while q:
86 #     u = q.popleft()
87 #     for v in G[u]:
88 #         if dist[v] == -1:
89 #             dist[v] = dist[u] + 1
90 #             q.append(v)
91
92 # DFS (recursive)
93 # sys.setrecursionlimit(10**7)
94 # def dfs(u, p):
95 #     for v in G[u]:
96 #         if v != p:
97 #             dfs(v, u)
98
99 # ----- TRICKS -----
100 # multiple test cases
101 # for _ in range(int(input())):
102 #     solve()
103
104 # Large integers handled natively in Python!
105 # Use pow(a, b, mod) for fast modular exponentiation
106 # Use itertools.permutations / combinations for brute force enumeration
107
108 # ----- DEMO problem by [LongPham] -----
109 rows, cols = 12, 12
110
111 # a = [[0] * col for _ in range(row)]
112 point = [[0] * cols for _ in range(rows)]
113
114 def invalid(x, y, n, m): return x < 0 or x >= n or y < 0 or y >= m
115 def dfs(i, j, n, m, cnt, a):
116
117     if invalid(i, j, n, m):
118         print(cnt)
119         return

```

```

120
121     if point[i][j] != 0 :
122         print("%d %d" % (point[i][j], cnt - point[i][j]))
123         return
124     point[i][j] = cnt
125
126     if a[i][j] == 'S' : dfs(i + 1, j, n, m, cnt + 1, a)
127     if a[i][j] == 'N' : dfs(i - 1, j, n, m, cnt + 1, a)
128     if a[i][j] == 'W' : dfs(i, j - 1, n, m, cnt + 1, a)
129     if a[i][j] == 'E' : dfs(i, j + 1, n, m, cnt + 1, a)
130
131 def solve():
132     n, m, st = map(int, input().split())
133     a = [list(input().strip()) for _ in range(n)]
134     dfs(0, st - 1, n, m, 0, a)
135
136 if __name__ == '__main__':
137     solve()

```

## 7.5 Small templates

Listing 46: Small templates

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define MASK(x) ((1) << (x)) //2^x
5 #define BIT(mask,i) (((mask) >> (i))&1) //kiem tra bit thu i cua mask
6 typedef long long ll;
7 typedef unsigned long long ull;
8 typedef unsigned long long int ulli;
9 #define fi first
10 #define se second
11 #define el cout << "\n"
12 #define FOR(i, l, r) for (int i = l; i <= r; i++)
13 #define FOD(i, r, l) for (int i = r; i >= l; i--)
14 #define FOB(i, l, r) for (int i = l; i < r; i++)
15 #define mem(a, b) memset(a, b, sizeof(a))
16 #define all(a) a.begin(), a.end()
17 #define faster() ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL)
18 template <typename T, typename U> inline bool maximize(T &a, U b) { return a < b ? (a = b, true) : false; }
19 template <typename T, typename U> inline bool minimize(T &a, U b) { return a > b ? (a = b, true) : false; }
20 const int MOD = 1e9 + 7;
21
22 #define DEBUG
23 #ifdef DEBUG
24     #define dbg(x) cerr << #x << " = " << (x) << "\n"
25 #else // elif cung dc: else if
26     #define dbg(x)
27 #endif
28
29 // Random integer in range [l, r]
30 mt19937 rd(chrono::steady_clock::now().time_since_epoch().count());
31 Rand(const int &l, const int &r) {
32     assert(l <= r);
33     int sz = (r - l + 1);
34     return l + rd() % sz;
35 }
36
37 template <typename T> T mod(T a, T b) {
38     if (b < 0) b = -b;
39     return (a % b + b) % b;
40 }
41
42 void solve() {
43     // Code logic here
44 }
45
46 int main() {
47     faster();
48     // freopen("input.INP", "r", stdin);
49     // freopen("output.OUT", "w", stdout);
50     int t; t = 1;
51     // cin >> t;
52     while (t--) {
53         solve();
54     }
55     return (0 ^ 0);
56 }

```

---

— End —

*"There is nothing shameful about learning, what's shameful is not knowing."*

---