

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

1	Data Structures	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
2	Dynamic Programming	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
3	Geometry	9
3.1	Closest Pair / Geometry Helper	9
3.2	Convexhull	10
3.3	Geometry Formulae	10
4	Graph	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
5	Math	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
6	Strings	19
6.1	KMP Search	19
6.2	String Hashing	19
6.3	Trie	20
7	Misc	21
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[j] 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     // kq : 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 int get(int pos)
91 {
92     int minPos = 1e9;
93     for (; pos > 0; pos -= (pos & -pos))
94     {
95         minPos = min(minPos, bit[pos]);
96     }
97     return minPos;
98 }
99
100 void solve()
101 {
102     cin >> n >> k;
103
104     a.assign(n + 1, 0);
105     for (int i = 1; i <= n; i++)
106     {
107         cin >> a[i];
108         vals.push_back(a[i]);
109     }
110
111     sort(all(vals));
112     vals.erase(unique(all(vals), vals.end()));
113
114     for(int i = 1; i <= n; i++) {
115         a[i] = lower_bound(vals.begin(), vals.end(), a[i]) - vals.begin() + 1;
116     }
117
118     nn = vals.size();
119     bit.assign(nn + 1, 1e9);
120
121     update(a[1], 1);
122     int maxDist = 0;
123     for(int i = 2; i <= n; i++) {
124
125         int x = vals[a[i] - 1]; // tim gia tri
126         int target = x - k;
127
128         int p = upper_bound(all(vals), target) - vals.begin();
129
130         if(p > 0) {
131             int g = get(p);
132             maxDist = max(maxDist, i - g);
133         }
134
135         update(a[i], i);
136     }
137
138     cout << maxDist << endl;
139 }

```

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 nhât (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
   theo thu tu tang dan.
3 const int N = 1e5 + 5;
4 typedef vector<int> dta;
5 int n, q, l, r, k, a[N];
6 vector<int> v;
7 dta t[N * 4];
8
9 // merge(seg[id*2].begin(), seg[id*2].end(),
10 // seg[id*2+1].begin(), seg[id*2+1].end(),
11 // seg[id].begin()); : hoac dung ham merge
12 dta combine(dta u, dta v)
13 {
14     dta ans;
15     ans.reserve(u.size() + v.size());
16     int i = 0, j = 0;
17     while (i < u.size() && j < v.size())
18     {
19         if (u[i] < v[j])
20             ans.push_back(u[i++]);
21         else
22             ans.push_back(v[j++]);
23     }
24     while (i < u.size())
25         ans.push_back(u[i++]);
26     while (j < v.size())
27         ans.push_back(v[j++]);
28     return ans;
29 }
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
96 struct dta {
97     ll sum, prefix, suffix, best;
98 };
99 int n, a[N], q;
100 dta t[N * 4];
101 // sum : tong doan con(all)
102 // prefix : tong doan con lon nhat bat dau tai l
103 // suffix : tong doan con lon nhat ket thuc tai r
104 // best : tong doan con lon nhat nam trong doan l - r
105 dta combine(dta l, dta r) {
106     dta res;
107     res.sum = l.sum + r.sum; // tong ca doan
108     res.prefix = max(l.prefix, l.sum + r.prefix);
109     res.suffix = max(r.suffix, r.sum + l.suffix);
110     res.best = max(max(l.best, r.best), l.suffix + r.prefix);
111     return res;
112 }
113
114 dta make(int val) {
115     return {val, val, val, val};
116 }
117
118 void build(int v, int tl, int tr) {
119     if (tl == tr) {
120         t[v] = make(a[tl]);
121     } else {
122         int mid = (tl + tr) / 2;
123         build(v << 1, tl, mid);
124         build(v << 1 | 1, mid + 1, tr);
125         t[v] = combine(t[v << 1], t[v << 1 | 1]);
126     }
127 }
128
129 dta get(int v, int tl, int tr, int l, int r) {
130     if (r < tl || l > tr)

```

```

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
166 void update(int v, int tl, int tr, int l, int r, int x)
167 {
168     push(v, tl, tr);
169     if (tl > r || l > tr)
170         return;
171     if (l <= tl && tr <= r)
172     {
173         t[v].lazy += x;
174         push(v, tl, tr);
175     }
176     else
177     {
178         int mid = (tl + tr) >> 1;
179         update(v << 1, tl, mid, l, r, x);
180         update(v << 1 | 1, mid + 1, tr, l, r, x);
181         t[v].val = t[v << 1].val + t[v << 1 | 1].val;
182     }
183 }
184
185 ll get(int v, int tl, int tr, int l, int r)
186 {
187     push(v, tl, tr);
188     if (tl > r || l > tr)
189         return 0;
190     if (l <= tl && tr <= r)
191     {
192         return t[v].val;
193     }
194     else
195     {
196         int mid = (tl + tr) >> 1;
197         ll getl = get(v << 1, tl, mid, l, r);
198         ll getr = get(v << 1 | 1, mid + 1, tr, l, r);
199         return getl + getr;
200     }
201 }

```

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, block, q;
5
6 void build() {
7     block = sqrt(n);

```

```

8     for(int i = 0; i < n; i++) {
9         bucket[i / block] += a[i];
10    }
11
12    void update(int pos, int value) {
13        int b = pos / block;
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 / block;
21        int rb = r / block;
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 giua(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 = l; i < (lb + 1) * block; i++)
34                sum += a[i];
35
36            // tinh tong phan con lai ben phai
37            for(int i = rb * block; i <= r; i++)
38                sum += a[i];
39        }
40        return sum;
41    }
42
43    void solve() {
44        cin >> n >> q;
45
46        for(int i = 0; i < n; i++) {
47            cin >> a[i];
48        }
49
50        build();
51        while (q--) {
52            int t; cin >> t;
53            if (t == 2) {
54                int l, r; cin >> l >> r;
55                l--, r--;
56                cout << get(l, r) << "\n";
57            } else {
58                int pos, val; cin >> pos >> val;
59                pos--;
60                update(pos, val);
61            }
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 long long currSum = 0;
16 inline void add(int pos) { currSum += a[pos]; }
17 inline void remove(int pos) { currSum -= a[pos]; }
18 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 thieuphan tu, giam de cong them ben duoi
32      while (currR < qu.r) add(++currR); // bi thieuphan tu, tang len de cong them ben tren
33      while (currL < qu.l) remove(currL++); // bi duphan tu, tang len de tru di
34      while (currR > qu.r) remove(currR--); // bi duphan 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 };
18 void insert(int x) {
19     TrieNode* cur = root;
20     for (int i = MAXBIT; i >= 0; i--) {
21         int bit = (x >> i) & 1;
22         if (!cur->child[bit])
23             cur->child[bit] = new TrieNode();
24         cur = cur->child[bit];
25         cur->cnt++;
26     }
27 } // cac so khi xor voi X lon nhat la bn
28 int maxXOR(int x) {
29     TrieNode* cur = root;
30     int res = 0;
31     for (int i = MAXBIT; i >= 0; i--) {
32         int bit = (x >> i) & 1;
33         int need = 1 - bit;
34         if (cur->child[need]) {
35             res |= (1 << i);
36             cur = cur->child[need];
37         } else {
38             cur = cur->child[bit];
39         }
40     }
41     return res;
42 } // so lon thu k
43 int kthXOR(int x, int k) {
44     TrieNode* cur = root;
45     int res = 0;
46     for (int i = MAXBIT; i >= 0; i--) {
47         int bit = (x >> i) & 1;
48         int need = 1 - bit;
49         if (cur->child[need] && cur->child[need]->cnt >= k) {
50             res |= (1 << i);
51             cur = cur->child[need];
52         } else {
53             if (cur->child[need])
54                 k -= cur->child[need]->cnt;
55             cur = cur->child[bit];
56         }
57     }
58 }

```

```

57     }
58     return res;
59 }
60 };

```

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)+1][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) << "\n";
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 }

```

```

19 for(int i = 0; i < n; i++)
20     dp[(1 << i)][i] = 1;
21 for(int mask = 0; mask < (1 << n); mask++) {
22     if(__builtin_popcount(mask) == 1) continue;
23     for(int u = 0; u < n; u++) {
24         if(mask & (1 << u) && u != LOG[mask & -mask]) { // ko phải bit thấp nhất
25             for(int v = 0; v < n; v++) {
26                 if(G[u] & (1 << v) && mask & (1 << v)) {
27                     dp[mask][u] += dp[mask ^ (1 << u)][v];
28                 }
29             }
30         }
31     }
32 }
33 for(int mask = 0; mask < (1 << n); mask++) {
34     for(int u = 0; u < n; u++) {
35         if(__builtin_popcount(mask) < 3) continue;
36         if(!(mask & (1 << u))) continue;
37         int lb = LOG[mask & -mask];
38         if(G[lb] & (1 << u)) { // neu co duong di tu bit thấp nhất đến u
39             ans = ans + dp[mask][u];
40         }
41     }
42 }
43 ans >>= 1; // moi chu trình bi dem 2 lan
44 cout << ans << endl;
45 }
46
47 // xoa canh(hoac ko), sao cho moi dinh trong tplt deu co duong di den nhau
48 // in ra so luong tplt ít nhất sau khi xoa, neu ko xoa(nghĩa là do thì dùng voi yêu cầu) thì ban đầu
49 // = 1
49 vector<ll>dp(1 << 18, INT32_MAX);
50 vector<int>G(20, 0);
51 int n, m, x, y;
52 // dp[mask] = số clique ít nhất để phủ toàn bộ tập trong mask
53 void solve() {
54     cin >> n >> m;
55     for(int i = 0; i < m; i++) {
56         cin >> x >> y;
57         x--, y--;
58         G[x] |= (1 << y);
59         G[y] |= (1 << x);
60     }
61     for(int mask = 0; mask < (1 << n); mask++) {
62         bool connected = true;
63         for(int u = 0; u < n; u++) {
64             if(!(mask & (1 << u))) continue;
65             if((G[u] & (1 << u)) & mask) != mask) { //check connected all vertices
66                 connected = false;
67                 break;
68             }
69         }
70         if(connected == true) dp[mask] = 1; // if all vertices is connected, => 1 components
71     }
72     for(int mask = 0; mask < (1 << n); mask++) {
73         for(int sub = mask; sub > 0; sub = (sub - 1) & mask) { // sub : tap con
74             int rest = sub ^ mask; // rest : layphan bu trong mask (tức
75                 //nhưng phần tử của mask không có trong sub)
76             if(dp[rest] != INT32_MAX && dp[sub] != INT32_MAX) { // 7 => sub : 6 : 110, rest : 1 :
77                 //001
78                 dp[mask] = min(dp[mask], dp[sub] + dp[rest]);
79             }
80         }
81     }
82     cout << dp[(1 << n) - 1] << endl;
83 }

```

2.2 Digit DP

Listing 9: Digit DP

```

1 // đề yêu cầu gì, làm 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 ll &nemo = dp[pos][sum][fl];
12 if (nemo != -1)
13     return nemo;
14 nemo = 0;
15 int limit = (fl ? s[pos] - '0' : 9);
16 for (int i = 0; i <= limit; i++) {
17     if (!valid(pos + 1, i) || (pos == 0 && i == 0))
18         continue;
19     int nsum = (sum * 10 + i) % num;
20     int nfl = (fl && i == limit);
21     nemo = (nemo + f(s, N, pos + 1, nsum, 1, nfl)) % MOD;
22 }
23 return nemo;
24 }
25 ll get(const string &s, int sz) {
26     memset(dp, -1, sizeof(dp));
27     return f(s, sz, 0, 0, 0, 1);
28 }
29 // nên dùng hàm dec xu ly, có thể dùng cho trường hợp số >= 1e19 // f(b) - f(dec(a - 1))
30 string dec(string s) {
31     int i = s.size() - 1;
32     while (i >= 0 && s[i] == '0') s[i--] = '9';
33     if (i >= 0) s[i]--;
34     // if (s[0] == '0' && s.size() > 1) s.erase(0, 1);
35     return s;
36 }
37 void solve() {
38     cin >> num >> d;
39     cin >> a >> b;
40     a = dec(a);
41     ll res = (get(b, b.size()) - get(a, a.size()) + MOD) % MOD;
42     cout << res << '\n';
43 }

```

2.3 Divide and Conquer DP

Listing 10: Divide and Conquer DP

```

1 // cho n và g, và dãy số a, n ptu
2 // tìm cách chia, sao cho chia n ptu vào g nhóm sao cho chi phí ít nhất
3 // cost = slpt * pref[l -> r];
4 // dp[i][g] : phân hoạch i ptu đầu tiên vào g nhóm, với cost là nhỏ nhất
5 ll cost(int l, int r) {
6     if(l > r) return 0;
7     return (r - l + 1) * (pref[r] - pref[l - 1]);
8 }
9 void cal(int l, int r, int opl, int opr, int k) {
10     if(l > r) return;
11     int mid = (l + r) / 2;
12     ii best = {INF, -1};
13     for(int i = opl; i <= opr; i++) {
14         ll p = dp[i][k - 1] + cost(i + 1, mid);
15         // chia nhóm thành [l -> i][g - 1] và [i + 1 -> mid][g]
16         // chia 1 -> i vào g - 1 nhóm đầu tiên, còn i + 1 -> mid vào nhóm thứ g -> mọi
17         if(p < best.first) {
18             best.first = p;
19             best.second = i;
20         }
21     }
22     dp[mid][k] = best.first; // 1 -> mid, với số nhóm = k
23     int opt = best.second;
24     cal(l, mid - 1, opl, opt, k);
25     cal(mid + 1, r, opt, opr, k);
26 }
27
28 void solve() {
29     cin >> n >> g;
30     for(int i = 1; i <= n; i++)
31         cin >> a[i], pref[i] = pref[i - 1] + a[i];
32     for(int i = 1; i <= n; i++)
33         dp[i][1] = cost(1, i); // i ptu đầu tiên vào 1 nhóm
34     dp[0][0] = 0;
35     for(int d = 2; d <= g; d++)
36         cal(1, n, 1, n, d);
37     cout << dp[n][g] << endl;
38 }
39
40 // dp divide + Mo's algorithm
41 // problems : chia dãy n ptu thành k nhóm sao cho chi phí là ít nhất
42 // chi phí = số lượng cặp ptu giống nhau trong nhóm do
43 // {i < j && a[i] == a[j]}

```

```

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(1, 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        f[u] += f[v]; // f[v] : so diem cua cay con truoc do, ko kem u
11    }
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     // dao canh
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++) if(__builtin_popcount(mask) >= 2) {
13     int tmp = mask & -mask;
14     sum[mask] = sum[tmp] + sum[mask ^ tmp];
15 }

```

2.6 LIS DP

Listing 13: LIS DP

```

1 // Longest Increasing Subsequence - O(nlogn)
2 //
3 // dp[i] = max j<i { dp[j] | a[j] < a[i] } + 1
4 //
5 // also can use fenwick tree
6 // int dp[N], v[N], n, lis;
7 // or fenwick tree, maybe:D
8 memset(dp, 63, sizeof dp);
9 for (int i = 0; i < n; ++i) {
10     // increasing: lower_bound
11     // non-decreasing: upper_bound
12     int j = lower_bound(dp, dp + lis, v[i]) - dp;
13     dp[j] = min(dp[j], v[i]);
14     lis = max(lis, j + 1);
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 slpt 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 1.
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     ll ans = 0;
12     ll mod = 998244353;
13     for(int i = 1; i <= n; i++) {
14         vector<vector<vector<ll>>> dp(n + 1, vector<vector<ll>>>(i + 1, vector<ll>(i, 0)));
15
16         dp[0][0][0] = 1;

```

```

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

```

3 Geometry

3.1 Closest Pair / Geometry Helper

Listing 16: Closest Pair / Geometry Helper

```

1 struct Point {
2     double x, y;
3 };
4
5 double dist(Point a, Point b) {
6     return sqrt((a.x - b.x) * (a.x - b.x) +
7                 (a.y - b.y) * (a.y - b.y));
8 }
9
10 // Brute force cho n nho
11 double bruteForce(vector<Point>& P, int l, int r) {
12     double ans = 1e18;
13     for (int i = l; i <= r; i++) {
14         for (int j = i + 1; j <= r; j++) {
15             ans = min(ans, dist(P[i], P[j]));
16         }
17     }
18     return ans;
19 }
20
21 // ham chinh chia de tri
22 double closestUtil(vector<Point>& Px, vector<Point>& Py, int l, int r) {
23     int n = r - l + 1;
24     if (n <= 3) return bruteForce(Px, l, r);
25
26     int mid = (l + r) / 2;
27     double midx = Px[mid].x;
28
29     //chia py thanh trai/phai
30     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     double 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 //nguc 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 //crosss = xly2 - ylx2
26 //0 -> cung phuong or //
27 // + -> trai
28 // - -> phai
29 //dot = xlx2 + yly2
30 //0 -> 90 do
31 //> 0 -> goc < 90do
32 //< 0 -> goc > 90do

```

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
18 // Kiem tra 3 diem thang hang
19 bool areCollinear(Point a, Point b, Point c) {
20     return (b - a).cross(c - a) == 0;
21 }
22
23 // Dien tich da giac (cong thuc shoelace)
24 long long polygonArea(vector<Point>& points) {
25     long long area = 0;
26     int n = points.size();
27     for (int i = 0; i < n; i++) {
28         int j = (i + 1) % n;
29         area += points[i].x * points[j].y - points[j].x * points[i].y;
30     }
31     return abs(area) / 2;
32 }
33
34 // 1. QUAN HE VI TRI DIEM & DUONG THANG
35 // Huong cua diem C so voi duong AB:
36 // cross(AB, AC) = cross(B-A, C-A)
37 double orientation = cross(B-A, C-A);
38 if (orientation > 0) : C ben TRAI AB (nguc chieu kim dong ho)
39 if (orientation < 0) : C ben PHAI AB (cung chieu kim dong ho)
40 if (orientation == 0) : A, B, C thang hang
41
42 // Kiem tra diem M thuoc doan AB:
43 bool onSegment(Point A, Point B, Point M) {
44     return cross(B-A, M-A) == 0 &&
45         min(A.x, B.x) <= M.x && M.x <= max(A.x, B.x) &&
46         min(A.y, B.y) <= M.y && M.y <= max(A.y, B.y);
47 }
48
49 // 2. KHOANG CACH & HINH CHIEU
50 // Khoang cach tu diem M den duong thang AB:
51 double distToLine(Point A, Point B, Point M) {
52     return abs(cross(B-A, M-A)) / distance(A, B);
53 }
54
55 // Hinh chieu cua M len duong AB:
56 Point project(Point A, Point B, Point M) {
57     Point AB = B - A;
58     Point AM = M - A;
59     double t = dot(AB, AM) / dot(AB, AB); // t = (AB AM)/|AB|^2
60     return A + AB * t;
61 }
62
63 // Vi tri hinh chieu:
64 // t < 0: hinh chieu nam phia A (M', A, B)
65 // t > 1: hinh chieu nam phia B (A, B, M')
66 // 0 <= t <= 1: hinh chieu nam tren doan AB (A, M', B)
67
68 // 3. GIAO DIEM & GIAO DOAN
69 // Kiem tra 2 doan AB va CD cat nhau:
70 bool segmentsIntersect(Point A, Point B, Point C, Point D) {
71     double o1 = cross(B-A, C-A);
72     double o2 = cross(B-A, D-A);
73     double o3 = cross(D-C, A-C);
74     double o4 = cross(D-C, B-C);
75
76     return o1*o2 <= 0 && o3*o4 <= 0;
77 }
78

```

```

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 hybot(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 nhât tu s -> t
2 // neu gap '#' thi co the pha 1 hoac 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     {
14         for (int j = 0; j < m; j++)
15         {
16             cin >> A[i][j];
17         }
18     }
19     cin >> a >> b >> c >> d;
20     a--, b--, c--, d--;

```

```

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 {
26     return x < 0 || x >= n || y < 0 || y >= m;
27 };
28 while (!dq.empty())
29 {
30     auto [d, x, y] = dq.front();
31     dq.pop_front();
32     if (d != dis[x][y])
33         continue;
34     for (int k = 0; k < 4; k++)
35     {
36         bool wall = false;
37         for (int i = 1; i <= 2; i++)
38         {
39             int nx = x + dx[k] * i;
40             int ny = y + dy[k] * i;
41             if (Invalid(nx, ny))
42                 continue;
43             int cost = (i == 1 ? d : d + 1);
44             if (dis[nx][ny] <= cost)
45                 continue;
46             if (A[nx][ny] == '#')
47                 wall = true;
48             if (!wall)
49             {
50                 if (i == 1)
51                 {
52                     dis[nx][ny] = d;
53                     dq.push_front({cost, nx, ny}); // khong co trong so
54                     break;
55                 }
56                 else
57                 {
58                     dis[nx][ny] = d + 1;
59                     dq.push_back({d + 1, nx, ny}); // co trong so
60                 }
61             }
62         }
63     }
64     cout << (dis[c][d] == INF ? -1 : dis[c][d]) << endl;
65 }
66
67 // cho vi tri bat dau tai a[s][t] : duoc di len va xuong khong gioi han,
68 // nhung trai phai thi gioi han la x va y
69 // dem so luong o co the di qua voi so lan gioi han la x, y trai va phai
70 // voi bai toan, ta se day het so lan di qua trai truoc,
71 // roi sau do tinh gioi han duong di qua phai
72 int n, m, x, y, s, t, ans = 0;
73 vector<vector<int>> dis2(2002, vector<int>(2002, 1e9));
74 char a2[2002][2002];
75 const int dx2[4] = {-1, 0, 0, 1};
76 const int dy2[4] = {0, -1, 1, 0};
77
78 void solve2() {
79     cin >> n >> m >> s >> t >> x >> y;
80     for (int i = 0; i < n; i++) {
81         for (int j = 0; j < m; j++) {
82             cin >> a2[i][j];
83         }
84     }
85     s--, t--;
86     deque<array<int, 3>> dq;
87     dq.push_front({s, t, 0});
88     dis2[s][t] = 0;
89     auto Invalid = [&](int x, int y) -> bool {
90         return x < 0 || x >= n || y < 0 || y >= m;
91     };
92     while (!dq.empty()) {
93         auto [x, y, _] = dq.front();
94         dq.pop_front();
95         for (int k = 0; k < 4; k++) {
96             int nx = x + dx2[k];
97             int ny = y + dy2[k];
98             if (Invalid(nx, ny) || a2[nx][ny] == '*') continue;
99             int cost = dis2[x][y] + (k == 1 ? 1 : 0); // di qua trai => +1
100             if (cost < dis2[nx][ny]) {
101                 dis2[nx][ny] = cost;
102                 if (k == 1) // qua trai
103                     dq.push_back({nx, ny, 0});
104                 else // qua phai
105                     dq.push_front({nx, ny, 0});
106             }
107         }
108     }
109 }

```

```

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]; // so lan di trai it nhât
115             int r = (j - t) + 1; // so lan di phai it nhât
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     el;
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 vong toi 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

```

1 // join : gop, add tang diem, get : lay diem
2 // join : (a, b), add : tang moi diem gom ca con cua u, get lay diem dinh do
3 int findP(int u) {
4     if (u == par[u]) return u;
5     int p = par[u];
6     par[u] = findP(par[u]);
7     point[u] += point[p];
8     return par[u];
9 }
10 void join(int u, int v) {
11     u = findP(u);
12     v = findP(v);
13     if (u == v) return;
14
15     if (sz[u] < sz[v]) swap(u, v);
16     par[v] = u;
17     point[v] = pc[v] - pc[u];
18     sz[u] += sz[v];
19 }
20 void add(int u, int v) {
21     u = findP(u);
22     pc[u] += v;
23 }
24 long long get(int u) {
25     int r = findP(u);
26     return pc[r] + point[u];
27 }
28 void solve() {
29     init();
30     while (q--) {
31         cin >> t;
32         if (t == "join") {
33             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 }
43 }

```

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 void solve() {
29     int n, m;
30     cin >> n >> m;
31     vector<item> a(m);
32     int cnt = 0, res = 0;
33     DSU dsu(n);
34     dsu.make();
35     for(int i = 0; i < m; i++) {
36         cin >> a[i].x >> a[i].y >> a[i].w;
37     }
38     sort(all(a), [](item a, item b) {
39         return a.w < b.w;
40     });
41     for(int i = 0; i < m; i++) {
42         if(cnt == n - 1) { break; }
43         if(dsu.unionn(a[i].x, a[i].y)) {
44             res += a[i].w;
45             cnt++;
46         }
47     }
48     cout << res << endl;
49 }

```

4.6 LCA

Listing 24: LCA

```

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

```

```

5         par[v][0] = u;
6         high[v] = high[u] + 1;
7         dfs(v);
8     }
9 }
10
11 void build() {
12     for(int j=1; j<=log; j++) {
13         for(int i=1; i<=n; i++) { if(par[i][j-1])
14             par[i][j] = par[par[i][j-1]][j-1];
15         }
16     }
17 }
18
19 int jump(int u, int k) {
20     for(int i=LOG-1; i>=0; i--) {
21         if(MASK(i) <= k) {
22             u = par[u][i];
23             k -= MASK(i);
24         }
25     }
26     return u;
27 }
28
29 int lca(int u, int v) {
30     if(high[v] > high[u]) {
31         return lca(v, u);
32     }
33     //can bang do xau
34     for(int i=log; i>=0; i--) {
35         if(high[par[u][i]] >= high[v]) {
36             u = par[u][i];
37         }
38     }
39     if(u == v) {
40         return u;
41     }
42     for(int i=log; i>=0; i--) {
43         if(par[u][i] != par[v][i]) {
44             u = par[u][i];
45             v = par[v][i];
46         }
47     }
48     return par[u][0];
49 }
50
51 //lca O(1) truy van
52 void dfs(int u, int p) {
53     nodes[++cnt] = u;
54     pos[u] = cnt;
55     for(int v : adj[u]) {
56         if(v != p) {
57             high[v] = high[u] + 1;
58             dfs(v, u);
59             nodes[++cnt] = u;
60         }
61     }
62     fin[u] = cnt;
63 }
64
65 int minDepthHigh(int u, int v) {
66     return (high[u] < high[v] ? u : v);
67 }
68
69 void build() {
70     for(int i=1; i<=cnt; i++) {
71         minHigh[i][0] = nodes[i];
72     }
73
74     for(int j=1; MASK(j) <= cnt; j++) {
75         for(int i=1; i+MASK(j)-1<=cnt; i++) {
76             minHigh[i][j] = minDepthHigh(minHigh[i][j-1], minHigh[i+MASK(j-1)][j-1]);
77         }
78     }
79 }
80
81 int lca(int u, int v) {
82     int pu = pos[u];
83     int pv = pos[v];
84     if(pu > pv) swap(pu, pv);
85
86     int k = 31 - __builtin_clz(pv-pu+1);
87     return minDepthHigh(minHigh[pu][k], minHigh[pv-MASK(k)+1][k]);
88 }

```

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 duyét
7 //low :thu tu duyét nho nhát
8 //instack : gióng vis, kiém tra đã duyét chưa, nếu đã trong 1 tplt nào đó rồi thì bỏ qua, tránh vào
   truông hóp else low[u] = min(low[u], disc[v]);
9 void dfs(int u) {
10     low[u] = disc[u] = ++timer;
11     st.push(u);
12     for(int i = 0; i < adj[u].size(); i++) {
13         int v = adj[u][i];
14         if(instack[v]) continue;
15         if(disc[v] == -1) { // canh thuoc cay dfs, chưa duyét
16             dfs(v);
17             low[u] = min(low[u], low[v]);
18         } else { // canh không thuoc cay dfs, duyét rồi
19             low[u] = min(low[u], disc[v]);
20         }
21     }
22     if(low[u] == disc[u]) { // u là gúc của 1 tplt mạnh
23         vector<int>component;
24         while(1) {
25             int v = st.top(); st.pop();
26             component.push_back(v);
27             instack[v] = 1;
28             if(v == u) break;
29         }
30         SCC.push_back(component);
31     }
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         ed;
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(i, 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

```

1 // Euler directed (Hierholzer)
2 // Input: n, edges list (u->v), 1-indexed
3 vector<int> euler_directed(int n, const vector<pair<int,int>>& edges) {
4     int m = edges.size();
5     vector<vector<int>>> g(n+1);
6     vector<int> out(n+1,0), in(n+1,0);
7     for (int i = 0; i < m; ++i) {
8         int u = edges[i].first, v = edges[i].second;
9         g[u].push_back(i); // store edge index
10        out[u]++; in[v]++;
11    }
12    // Check degree conditions
13    int start = -1, cntStart=0, cntEnd=0;
14    for (int v=1; v<=n; ++v) {
15        if (out[v] - in[v] == 1) { cntStart++; start = v; }
16        else if (in[v] - out[v] == 1) cntEnd++;
17        else if (in[v] != out[v]) return vector<int>(); // impossible
18    }
19    if (!((cntStart==0 && cntEnd==0) || (cntStart==1 && cntEnd==1))) return vector<int>();
20    if (start == -1) {
21        // choose any vertex with out>0
22        for (int v=1; v<=n; ++v) if (out[v] > 0) { start = v; break; }
23        if (start == -1) return vector<int>(); // no edges
24    }
25    // Hierholzer with pointers on adjacency
26    vector<int> it(n+1, 0);
27    vector<int> st;
28    vector<int> path;
29    st.push_back(start);
30    while (!st.empty()) {
31        int v = st.back();
32        if (it[v] < (int)g[v].size()) {
33            int ei = g[v][it[v]++];
34            int to = edges[ei].second;
35            st.push_back(to);
36        } else {
37            path.push_back(v);
38            st.pop_back();
39        }
40    }
41    if ((int)path.size() != m+1) return vector<int>(); // not all edges used
42    reverse(path.begin(), path.end());
43    return path;
44 }
45
46 // Euler undirected (Hierholzer)
47 // Input: n = số đỉnh (1..n), edges = vector of (u,v) (1-indexed)
48 // Output: vector<int> path (list of vertices in order) or empty if none
49 vector<int> euler_undirected(int n, const vector<pair<int,int>>& edges) {
50     int m = edges.size();
51     vector<vector<pair<int,int>>>> g(n+1); // (neighbor, edge_id)
52     vector<int> deg(n+1,0);
53     vector<char> used(m,false);
54     for (int i = 0; i < m; ++i) {
55         int u = edges[i].first, v = edges[i].second;
56         g[u].push_back({v,i});
57         g[v].push_back({u,i});
58         deg[u]++; deg[v]++;
59     }
60    // find start: if exists vertex with odd degree -> start there, else any vertex with degree>0
61    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}
    depending)
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
    connected component
87 reverse(path.begin(), path.end());
88 return path;
89 }
90 // Example usage in main:
91 // int main() {
92 //     int n, m; cin >> n >> m;
93 //     vector<pair<int, int>> edges(m);
94 //     for (int i=0; i<m; i++) cin >> edges[i].first >> edges[i].second;
95 //     auto path = euler_undirected(n, edges);
96 //     if (path.empty()) { cout << "NO\n"; }
97 //     else {
98 //         for (int v: path) cout << v << ' ';
99 //         cout << '\n';
100 //     }
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))))
33                 continue;
34             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                 }
42                 dp[mask | (1 << (v - 1))][v] = min(dp[mask | (1 << (v - 1))][v], dp[mask][u] + a[u][
                    v]);
43             }
44         }
45     }
46
47     for (int i = 1; i <= n; i++)
48         ans = min(ans, dp[(1 << n) - 1][i] + a[i][0]);
49     cout << ans << endl;
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 child = (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 }
33 //ban dau to mau dinh i, trong dfs or bfs la 0 or 1 deu dc

```

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             return tr;
65         }
66         return 0;

```

```

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

```

4.13.2 Hopcroft–Karp Matching

Listing 32: Hopcroft–Karp Matching

```

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 nghĩa là có ít nhất 1 duong tang ton tai
40     return dist[0] != INF;
41 }
42
43
44 bool dfs(int u) {
45     if (u == 0) return true; // toi dinh ao = ket thuc 1 duong tang

```

```

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

```

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 }
31
32 void solve() {
33     cin >> n >> m >> k;
34     ans = 0; freq.clear();
35     for(int i = 1; i <= n; i++)
36         cin >> a[i], freq[a[i]]++;
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 voi a[l]
41     }
42     cout << ans << endl;
43 }
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(p) = 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 // Miller-Rabin cho so lon (optional)
12 bool millerTest(long long d, long long n) {
13     long long a = 2 + rand() % (n - 4);
14     long long x = power(a, d, n);
15     if (x == 1 || x == n - 1) return true;
16
17     while (d != n - 1) {
18         x = (x * x) % n;
19         d *= 2;
20         if (x == 1) return false;
21         if (x == n - 1) return true;
22     }
23     return false;
24 }
25
26 bool isPrimeMR(long long n, int k = 5) {
27     if (n <= 1 || n == 4) return false;
28     if (n <= 3) return true;
29
30     long long d = n - 1;
31     while (d % 2 == 0) d /= 2;
32
33     for (int i = 0; i < k; i++) {
34         if (!millerTest(d, n)) return false;
35     }
36     return true;
37 }
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 // ecuid + ecuid 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 ecuid(int a, int b) {
26 //     if(b == 0) {
27 //         x = 1; y = 0;
28 //         d = a; // ucln
29 //         return;
30 //     }
31 //     ecuid(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 tn ti : gcd ( , ) = 1 gcd(a,m)=1 ( t c l a v m nguyn t
43 // c ng nhau).
44 //Cl : Ecuid mo rong
45 // int x, y, d;

```

```

45 // void ecuid(int a, int b) {
46 //     if(b == 0) {
47 //         x = 1; y = 0;
48 //         d = a; // ucln
49 //         return;
50 //     }
51 //     ecuid(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 //     ecuid(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 < n; i++) {
7         while(j && t[i] != t[j]) j = lps[j-1];
8         if(t[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    }
19 }

```

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 = [&](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     // chuoai 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     // chuoai 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 }

```

```

84 }
85
86 sort(sub.begin(), sub.end());
87 sub.erase(unique(sub.begin(), sub.end()), sub.end());
88 int len = s.length(), ans = 0;
89 vector<int> dp(len + 1, 0);
90
91 for (int r = 1; r <= len; r++) {
92     for (int l = 1; l < sub.size(); l++) {
93         if (r - sub[l] + 1 < 0) break;
94         if (dict.count(h.getHash(r - sub[l] + 1, r))) {
95             dp[r] = max(dp[r], dp[r - sub[l]] + sub[l]);
96         }
97     }
98     ans = max(ans, dp[r]);
99 }
100
101 cout << ans << endl;
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]++; // increase count of words passing node
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] = false;
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]--; // decrease count
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
79 // delete word from trie
80 void deleteWord(const string &word) {
81     if (!search(word)) return;
82     process(0, word, 0);
83 }

```

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: layphan bu trong mask (tuc nhungphan 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
28 // 4 : cac ham thao tac bit
29 // __builtin_ctz(x): lay vi tri bit thap nhat
30 // __builtin_popcount(x): dem so luong bit trong mask
31
32 // tim vi tri cua bit cao nhat
33 // int pos = 31 - __builtin_clz(a); // vi tri MSB
34 // a ^= (1 << pos); // tat MSB
35
36 // a &= (a - 1): xoa bit 1 thap nhat
37 // x = a & -a: giu lai bit 1 thap nhat (co the hieu la tat moi bit tru bit thap nhat)
38
39 // 1 bai toan khac ve bit
40 void solve() {
41     int N, C;
42     cin >> N >> C;
43     vector<int> T(N), A(N);
44     for (int i = 0; i < N; i++) {
45         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 : Giong nhu binary search, nhưng 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 (a[mid2] == target) return mid2;
15 //         if (target < a[mid1]) r = mid1 - 1;
16 //         else if (target > a[mid2]) l = mid2 + 1;
17 //         else { // nam o giua
18 //             l = mid1 + 1;
19 //             r = mid2 - 1;
20 //         }
21 //     }
22 //     return -1;
23 // }
24
25 // Giong nhu do thi parabol
26 long long f(long long k, int n, int a, int b) {
27     long long cake = (n - k) * a;
28     long long cur = 2 * b - k;
29     long long cake_ = k * (cur + 1) / 2;
30     return cake + cake_;
31 }
32
33 void solve() {
34     cin >> n >> a >> b;
35     int l = 0, r = min(n, b);
36     while (r - l > 2) {
37         int mid1 = l + (r - l) / 3;
38         int mid2 = r - (r - l) / 3;
39         if (f(mid1, n, a, b) > f(mid2, n, a, b))
40             r = mid2;
41         else
42             l = mid1;
43     }
44     long long ans = 0;
45     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 0.000000001
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:
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() {

```

```

49        bitset<8> b("10100010");
50        cout << "Ban dau: " << b << "\n";
51        b.flip(0); // dao bit thu 0
52        cout << "Sau flip(0): " << b << "\n";
53        b.set(3);
54        cout << "Sau set(3): " << b << "\n";
55        cout << "So bit 1: " << b.count() << "\n";
56        cout << "Gia tri thap phan: " << b.to_ulong() << "\n";
57    }
58
59    void solve3() {
60        {
61            // bitset<10>a("100");
62            // bitset<10>b("101");
63            // if ((a & b).any()) cout << "Hai tap co giao\n";
64            bitset<8> mat[5];
65            mat[0][1] = 1;
66            mat[2][5] = 1;
67            for (int i = 0; i < 5; i++)
68                cout << mat[i] << "\n";
69
70            //-----
71            const int X = 3, Y = 3;
72            bitset<8> cube[X][Y]; // "bitset 3D" (X, Y, bit_index)
73            cube[0][0][2] = 1; // dat bit thu 2 o lop (0,0)
74            cube[1][2][5] = 1;
75            cube[2][1][7] = 1;
76            // In toan bo khoi
77            for (int i = 0; i < X; i++)
78            {
79                cout << "Lop " << i << ": \n";
80                for (int j = 0; j < Y; j++)
81                    cout << cube[i][j] << "\n";
82                cout << "\n";
83            }
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     cout << "Set: ";
139     for (int x : s) cout << x << " ";
140     cout << endl;
141 }
142 void multiset_tricks() {
143     multiset<int> ms = {3, 1, 3};
144     ms.insert(2);
145     ms.erase(ms.find(3)); // remove only one occurrence
146     cout << "Multiset: ";
147     for (int x : ms) cout << x << " ";
148     cout << endl;
149 }
150 // -----
151 // Map / Unordered map tricks
152 // -----
153 void map_tricks() {
154     map<string, int> mp;
155     mp["a"] = 5;
156     mp["b"] = 2;
157     // lower_bound / upper_bound in map
158     auto it = mp.lower_bound("b"); // first key >= "b"
159     if (it != mp.end()) cout << it->first << " -> " << it->second << endl;
160     mp.erase("b");
161     cout << "Map: ";
162     for (auto &pair : mp) {
163         cout << pair.first << "->" << pair.second << " ";
164     }
165     cout << endl;
166 }
167 void unordered_map_tricks() {
168     unordered_map<int, int> ump;
169     ump[1] = 10;
170     ump[2] = 20;
171     // default value trick
172     cout << "ump[3] = " << ump[3] << endl; // 0 if not exist
173     cout << "Unordered map: ";
174     for (auto &pair : ump) {
175         cout << pair.first << " " << pair.second << " ";
176     }
177     cout << endl;
178 }
179 // -----
180 // Stack / Queue / Priority Queue
181 // -----
182 void stack_queue_tricks() {
183     stack<int> st;
184     st.push(1);
185     st.push(2);
186     cout << "Stack top: " << st.top() << endl;
187     st.pop();
188     queue<int> q;
189     q.push(5);
190     q.push(6);
191     cout << "Queue front: " << q.front() << " back: " << q.back() << endl;
192     q.pop();
193     // max-heap (default)
194     priority_queue<int> pq;
195     pq.push(3);
196     pq.push(1);
197     pq.push(4);
198     cout << "Max-heap top: " << pq.top() << endl;
199     pq.pop();
200     // min-heap
201     priority_queue<int, vector<int>, greater<int>> pq_min;
202     pq_min.push(3);
203     pq_min.push(1);
204     pq_min.push(4);
205     cout << "Min-heap top: " << pq_min.top() << endl;
206     pq_min.pop();
207     // pair priority queue (min-heap based on first element)
208     priority_queue<pii, vector<pii>, greater<pii>> pqp;
209     pqp.push({2, 5});
210     pqp.push({1, 7});
211     cout << "Pair priority queue top: " << pqp.top().first << " " << pqp.top().second << endl;
212 }
213 // -----
214 // Bitset tricks
215 // -----
216 void bitset_tricks() {
217     bitset<10> b;
218     b.set(2);
219     b.set(5, 1);
220     b.reset(2);
221     b.flip(5);
222     cout << "Bitset: " << b << endl;
223     cout << "Count of 1s: " << b.count() << endl;
224 }
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 // demo bai toan vd su dung stl
241 void solve() {
242     cin >> n;
243     a.assign(n + 1, 0);
244     for (int i = 0; i < n; i++)
245         cin >> a[i];
246
247     for (int i = 0; i < n; i++) {
248         int start = a[i];
249         vector<int> sq = {start};
250         unordered_set<int> used;
251         used.insert(start);
252         while (sq.size() < n) {
253             int x = sq.back();
254             bool found = false;
255             if (x % 3 == 0 && !used.count(x / 3) && find(all(a), x / 3) != a.end()) {
256                 found = true;
257                 used.insert(x / 3);
258                 sq.push_back(x / 3);
259             } else if (!used.count(x * 2) && find(all(a), x * 2) != a.end()) {
260                 found = true;
261                 used.insert(x * 2);
262                 sq.push_back(x * 2);
263             }
264
265             if (found == false)
266                 break;
267         }
268         if (sq.size() == n)
269             for (int x : sq)
270                 cout << x << ' ';
271     }
272     el;
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) :
19     false; }
20 template <typename T, typename U> inline bool minimize(T &a, U b) { return a > b ? (a = b, true) :
21     false; }
22 const int MOD = 1e9 + 7;
23
24 #define DEBUG
25 #ifdef DEBUG
26     #define dbg(x) cerr << #x << " = " << (x) << "\n"
27 #else // elif cung dc: else if
28     #define dbg(x)
29 #endif
30
31 // Random integer in range [l, r]
32 mt19937 rd(chrono::steady_clock::now().time_since_epoch().count());
33 int Rand(const int &l, const int &r) {
34     assert(l <= r);
35     int sz = (r - l + 1);
36     return l + rd() % sz;
37 }
38
39 template <typename T> T mod(T a, T b) {
40     if (b < 0) b = -b;
41     return (a % b + b) % b;
42 }
43
44 void solve() {
45     // Code logic here
46 }
47
48 int main() {
49     faster();
50     // freopen("input.INP", "r", stdin);
51     // freopen("output.OUT", "w", stdout);
52     int t; t = 1;
53     // cin >> t;
54     while (t--) {
55         solve();
56     }
57 }

```

— **End** —

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