

## Contents

### 1 Basic

- 1.1 .vimrc . . . . .
- 1.2 Default Code . . . . .
- 1.3 Common Sense . . . . .
- 1.4 Useful STL . . . . .
- 1.5 Bi/Ternary Search . . . . .
- 1.6 TroubleShoot . . . . .

### 2 flow

- 2.1 MinCostFlow \* . . . . .
- 2.2 Dinic . . . . .
- 2.3 Kuhn Munkres 最大完美二分匹配 . . . . .
- 2.4 Directed MST \* . . . . .
- 2.5 SW min-cut (不限 S-T 的 min-cut) \* . . . . .
- 2.6 Bounded Max Flow . . . . .
- 2.7 Flow Method \* . . . . .

### 3 Math

- 3.1 Fast Pow & Inverse & Combination . . . . .
- 3.2 Ext GCD . . . . .
- 3.3 Sieve 質數篩 . . . . .
- 3.4 FFT \* . . . . .
- 3.5 NTT \* . . . . .
- 3.6 Linear Recurrence \* . . . . .
- 3.7 Miller Rabin . . . . .
- 3.8 Faulhaber ( $\sum_{i=1}^n i^p$ ) \* . . . . .
- 3.9 Chinese Remainder \* . . . . .
- 3.10 Pollard Rho \* . . . . .
- 3.11 Josephus Problem \* . . . . .
- 3.12 Gaussian Elimination \* . . . . .
- 3.13 Result \* . . . . .

### 4 Geometry

- 4.1 definition . . . . .
- 4.2 halfPlaneIntersection \* . . . . .
- 4.3 Convex Hull \* . . . . .
- 4.4 Convex Hull trick \* . . . . .
- 4.5 掃描的線 . . . . .
- 4.6 Polar sort . . . . .
- 4.7 Li Chao Segment Tree \* . . . . .
- 4.8 KD Tree \* . . . . .

### 5 Tree

- 5.1 LCA . . . . .

### 6 Graph

- 6.1 HeavyLightDecomposition \* . . . . .
- 6.2 Centroid Decomposition \* . . . . .
- 6.3 DominatorTree \* . . . . .
- 6.4 MaximumClique 最大團 \* . . . . .
- 6.5 MaximalClique 極大團 \* . . . . .
- 6.6 Minimum Steiner Tree . . . . .
- 6.7 BCC based on vertex \* . . . . .
- 6.8 Strongly Connected Component \* . . . . .
- 6.9 差分約束 \* . . . . .

### 7 String

- 7.1 PalTree \* . . . . .
- 7.2 SuffixArray \* . . . . .
- 7.3 MinRoation \* . . . . .
- 7.4 RollingHash . . . . .
- 7.5 KMP . . . . .
- 7.6 LCS & LIS . . . . .
- 7.7 Aho-Corasick \* . . . . .
- 7.8 Z Value \* . . . . .
- 7.9 manacher \* . . . . .

### 8 Data Structure

- 8.1 Treap . . . . .
- 8.2 BIT . . . . .
- 8.3 二維偏序 \* . . . . .
- 8.4 持久化 \* . . . . .
- 8.5 2D 線段樹 . . . . .
- 8.6 Disjoint Set . . . . .
- 8.7 Black Magic . . . . .

### 9 Others

- 9.1 SOS dp \* . . . . .
- 9.2 MO's Algorithm \* . . . . .

## 1 Basic

### 1.1 .vimrc

```

linenumber, relative-linenumber, mouse, cindent, expandtab,
shiftwidth, softtabstop, nowrap, ignorecase(when search), noVi-
compatible, backspace
nornu when enter insert mode

```

```

1 se nu rnu mouse=a cin et sw=2 sts=2 nowrap ic nocp bs=2
2 syn on

```

### 1.2 Default Code

所有模板的 define 都在這

```

1 #include<bits/stdc++.h>
2 using namespace std;
3
4 #ifndef LOCAL // ===== Local ===== g++ -DLOCAL ...
5 void dbg() { cerr << '\n'; }
6 template<class T, class ...U> void dbg(T a, U ...b) {
7     cerr << a << ' ', dbg(b...); }
8 template<class T> void org(T l, T r) {
9     while (l != r) cerr << *l++ << ' '; cerr << '\n'; }
10 #define DEBUG(args...) \
11     (dbg("#> (" + string(#args) + ") = (" + args, ")"))
12 #define ORANGE(args...) \
13     (cerr << "#> [" + string(#args) + "] = ", org(args))
14 #else // ===== OnlineJudge =====
15 #define DEBUG(...) ((void)0)
16 #define ORANGE(...) ((void)0)
17 #endif
18
19 #define ll long long
20 #define ld long double
21 #define INF 0x3f3f3f3f
22 #define LLINF 0x3f3f3f3f3f3f3f3f
23 #define NINF 0xc1c1c1c1
24 #define NLLINF 0xc1c1c1c1c1c1c1c1
25 #define X first
26 #define Y second
27 #define PB emplace_back
28 #define pll pair<ll, ll>
29 #define MEM(a,n) memset(a, n, sizeof(a))
30 #define ios ios::sync_with_stdio(0); cin.tie(0); cout.
31     tie(0);
32 const int MXN = + 5;
33
34 void sol(){}
35 int main(){
36     io int t=1;
37     cin >> t; // !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
38     while(t--){ sol(); } }

```

### 1.3 Common Sense

陣列過大時本機的指令：

windows: g++ -Wl,-stack,4000000 a.cpp

linux: ulimit -s unlimited

1e7 的 int 陣列 = 4e7 byte = 40 mb

STL 式模板函式名稱定義：

.init(n, ...) ⇒ 初始化並重置全部變數, 0-base

.addEdge(u, v, ...) ⇒ 加入一條邊, 有向圖為  $u \rightarrow v$ , 無向圖為  $u \leftrightarrow v$

.run() ⇒ 執行並回傳答案

.build() ⇒ 查詢前處理

.query(...) ⇒ 查詢並回傳答案

memset 設 -0x3f 的值是 -0x3e3e3e3f / 0xc1c1c1c1

### 1.4 Useful STL

```

1 // unique
2 sort(a.begin(), a.end());
3 a.resize(unique(a.begin(), a.end()) - a.begin());
4 // O(n) a[k] = kth small, a[i] < a[k] if i < k
5 nth_element(a.begin(), a.begin()+k, a.end());
6 // stable_sort(a.begin(), a.end())
7 // lower_bound: first element >= val
8 // upper_bound: first element > val
9 // set_union, set_intersection, set_difference,
10 // set_symmetric_difference
11 set_union(a.begin(), a.end(), b.begin(), b.end(),
12     inserter(c, c.begin()));
13 //next_permutation prev_permutation(sort/reverse first)
14 do{ for(auto i : a) cout << i << ' ';
15 } while(next_permutation(a.begin(), a.end()));

```

### 1.5 Bi/Ternary Search

```

1 while(l < r){ // first l of check(l) == true
2     ll m = (l + r) >> 1;
3     if(!check(m)) l = m + 1; else r = m; }
4 while(l < r){ // last l of check(l) == false
5     ll m = (l + r + 1) >> 1;
6     if(!check(m)) l = m; else r = m - 1; }
7 while(l < r){
8     ll ml = l + (r - l) / 3, mr = r - (r - l) / 3;
9     if(check(ml)>check(mr)) l = ml + 1; else r = mr - 1;}

```

## 1.6 Troubleshoot

提交前：

如果樣本不夠，寫幾個簡單的測資。  
複雜度會不會爛？生成最大的測資試試。  
記憶體使用是否正常？

會 overflow 嗎？  
確定提交正確的檔案。

WA：

記得輸出你的答案！也輸出 debug 看看。

測資之間是否重置了所有變數？

演算法可以處理整個輸入範圍嗎？

再讀一次題目。

您是否正確處理所有邊緣測資？

您是否正確理解了題目？

任何未初始化的變數？

有 overflow 嗎？

混淆 n, m, i, j 等等？

確定演算法有效嗎？

哪些特殊情況沒有想到？

確定 STL 函數按你的想法執行嗎？

寫一些 assert 看看是否有些東西不如預期？

寫一些測資來跑你的演算法。

產生一些簡單的測資跑演算法看看。

再次瀏覽此列表。

向隊友解釋你的演算法。

請隊友查看您的代碼。

去散步，例如去廁所。

你的輸出格式正確嗎？(包括空格)

重寫，或者讓隊友來做。

RE：

您是否在本機測試了所有極端情況？

任何未初始化的變數？

您是否在任何向量範圍之外閱讀或寫作？

任何可能失敗的 assert？

任何的除以 0？(例如 mod 0)

任何的無限遞迴？

無效的 pointer 或 iterator？

你是否使用了太多的記憶體？

TLE：

有無限迴圈嗎？

複雜度是多少？

是否正在複製大量不必要的數據？(改用參考)

有沒有開 io？

避免 vector/map。(使用 array/unordered\_map)

你的隊友對你的演算法有什麼看法？

MLE：

您的演算法應該需要的最大記憶體是多少？

測資之間是否重置了所有變數？

## 2 flow

### 2.1 MinCostFlow \*

```

1 struct zkwflow{
2     static const int MXN = 10000;
3     struct Edge{ int v, f, re; ll w;};
4     int n, s, t, ptr[MXN]; bool vis[MXN]; ll dis[MXN];
5     vector<Edge> E[MXN];
6     void init(int _n, int _s, int _t){
7         n=_n, s=_s, t=_t;
8         for(int i=0; i<n; i++) E[i].clear();
9     }
10    void addEdge(int u, int v, int f, ll w){
11        E[u].emplace_back(v, f, (int)E[v].size(), w);
12        E[v].emplace_back(u, 0, (int)E[u].size()-1, -w);
13    }
14    bool SPFA(){
15        fill_n(dis, n, LLMXN); memset(vis, 0, 4 * n);
16        queue<int> q; q.push(s); dis[s] = 0;
17        while (!q.empty()){
18            int u = q.front(); q.pop(); vis[u] = false;
19            for(auto &it : E[u]){
20                if(it.f > 0 && dis[it.v] > dis[u] + it.w){
21                    dis[it.v] = dis[u] + it.w;
22                    if(!vis[it.v]){
23                        vis[it.v] = 1; q.push(it.v);
24                    } } } }
25        return dis[t] != LLMXN;
26    }
27    int DFS(int u, int nf){
28        if(u == t) return nf;
29        int res = 0; vis[u] = 1;
30        for(int &i = ptr[u]; i < (int)E[u].size(); ++i){
31            auto &it = E[u][i];
32            if(it.f>0&&dis[it.v]==dis[u]+it.w&&!vis[it.v]){
33                int tf = DFS(it.v, min(nf, it.f));
34                res += tf, nf -= tf, it.f -= tf;
35                E[it.v][it.re].f += tf;
36                if(nf == 0){ vis[u] = false; break; }

```

```

37    }
38    }
39    return res;
40 }
41 pair<int, ll> flow(){
42     int flow = 0; ll cost = 0;
43     while (SPFA()){
44         memset(ptr, 0, 4 * n);
45         int f = DFS(s, INF);
46         flow += f; cost += dis[t] * f;
47     }
48     return{ flow, cost };
49 }
50 } flow;

```

### 2.2 Dinic

求最大流  $O(N^2 E)$ ，求二分最大匹配  $O(E\sqrt{N})$

dinic.init(n, st, en)  $\Rightarrow$  0-base

dinic.addEdge(u, v, f)  $\Rightarrow u \rightarrow v$ , flow f units

dinic.run()  $\Rightarrow$  return max flow from st to en

Dinic 玄學：若 TLE，可以先加“正向邊”且每次都 run()，再全加一次每次都 run()。

範例 code 待補

```

1 const int MXN = 10005;
2 struct Dinic{
3     struct Edge{ ll v, f, re; };
4     int n, s, t, lvl[MXN];
5     vector<Edge> e[MXN];
6     void init(int _n, int _s, int _t){
7         n = _n; s = _s; t = _t;
8         for(int i = 0; i < n; ++i) e[i].clear();
9     }
10    void addEdge(int u, int v, ll f = 1){
11        e[u].push_back({v, f, e[v].size()});
12        e[v].push_back({u, 0, e[u].size() - 1});
13    }
14    bool bfs(){
15        memset(lvl, -1, n * 4);
16        queue<int> q;
17        q.push(s);
18        lvl[s] = 0;
19        while(!q.empty()){
20            int u = q.front(); q.pop();
21            for(auto &i : e[u])
22                if(i.f > 0 && lvl[i.v] == -1)
23                    lvl[i.v] = lvl[u] + 1, q.push(i.v);
24        }
25        return lvl[t] != -1;
26    }
27    ll dfs(int u, ll nf){
28        if(u == t) return nf;
29        ll res = 0;
30        for(auto &i : e[u])
31            if(i.f > 0 && lvl[i.v] == lvl[u] + 1){
32                ll tmp = dfs(i.v, min(nf, i.f));
33                res += tmp, nf -= tmp, i.f -= tmp;
34                e[i.v][i.re].f += tmp;
35                if(nf == 0) return res;
36            }
37        if(!res) lvl[u] = -1;
38        return res;
39    }
40    ll run(ll res){
41        while(bfs()) res += dfs(s, LLINF);
42        return res;
43    }
44 }

```

### 2.3 Kuhn Munkres 最大完美二分匹配

二分完全圖最大權完美匹配  $O(n^3)$  (不太會跑滿)

轉換：

最大權匹配 (沒邊就補 0)

最小權完美匹配 (權重取負)

最大權重積 (ll 改 ld, memset 改 fill, w 取自然對數 log(w), 答案為 exp(ans))

二分圖判斷: DFS 建樹記深度 -> 有邊的兩點深度奇偶性相同 -> 奇環 -> 非二分圖

二分圖最小頂點覆蓋 = 最大匹配

| 最大匹配 | + | 最小邊覆蓋 | = |V|

| 最小點覆蓋 | + | 最大獨立集 | = |V|

| 最大匹配 | = | 最小點覆蓋 |

最大團 = 補圖的最大獨立集

```

1 const int MXN = 1005;
2 struct KM{ // 1-base
3     int n, mx[MXN], my[MXN], pa[MXN];
4     ll g[MXN][MXN], lx[MXN], ly[MXN], sy[MXN];
5     bool vx[MXN], vy[MXN];
6     void init(int _n){
7         n = _n;
8         MEM(g, 0);
9         void addEdge(int x, int y, ll w){ g[x][y] = w; }
10        void augment(int y){

```

```

11 for(int x, z; y; y = z)
12     x = pa[y], z = mx[x], my[y] = x, mx[x] = y; }
13 void bfs(int st){
14     for(int i = 1; i <= n; ++i)
15         sy[i] = LLINF, vx[i] = vy[i] = 0;
16     queue<int> q; q.push(st);
17     for(;;){
18         while(!q.empty()){
19             int x = q.front(); q.pop();
20             vx[x] = 1;
21             for(int y = 1; y <= n; ++y)
22                 if(!vy[y]){
23                     ll t = lx[x] + ly[y] - g[x][y];
24                     if(t == 0){
25                         pa[y] = x;
26                         if(!my[y]){ augment(y); return; }
27                         vy[y] = 1, q.push(my[y]); }
28                     else if(sy[y] > t) pa[y] = x, sy[y] = t; } }
29     ll cut = LLINF;
30     for(int y = 1; y <= n; ++y)
31         if(!vy[y] && cut > sy[y]) cut = sy[y];
32     for(int j = 1; j <= n; ++j){
33         if(vx[j]) lx[j] -= cut;
34         if(vy[j]) ly[j] += cut;
35         else sy[j] -= cut; }
36     for(int y = 1; y <= n; ++y)
37         if(!vy[y] && sy[y] == 0){
38             if(!my[y]){ augment(y); return; }
39             vy[y] = 1, q.push(my[y]); } } }
40 ll run(){
41     MEM(mx, 0), MEM(my, 0), MEM(ly, 0), MEM(lx, -0x3f);
42     for(int x=1; x <= n; ++x) for(int y=1; y <= n; ++y)
43         lx[x] = max(lx[x], g[x][y]);
44     for(int x = 1; x <= n; ++x) bfs(x);
45     ll ret = 0;
46     for(int y = 1; y <= n; ++y) ret += g[my[y]][y];
47     return ret; } };
```

## 2.4 Directed MST \*

```

1 struct DMST {
2     struct Edge{ int u, v, c;
3         Edge(int u, int v, int c):u(u),v(v),c(c){} };
4     int v, e, root;
5     Edge edges[MXN];
6     int newV(){ return ++v; }
7     void addEdge(int u, int v, int c)
8         { edges[++e] = Edge(u, v, c); }
9     bool con[MXN];
10    int mnInW[MXN], prv[MXN], cyc[MXN], vis[MXN];
11    int run(){
12        memset(con, 0, 4*(V+1));
13        int r1 = 0, r2 = 0;
14        while(1){
15            fill(mnInW, mnInW+V+1, INF);
16            fill(prv, prv+V+1, -1);
17            for(int i = 1; i <= e; ++i){
18                int u=edges[i].u, v=edges[i].v, c=edges[i].c;
19                if(u != v && v != root && c < mnInW[v])
20                    mnInW[v] = c, prv[v] = u; }
21            fill(vis, vis+V+1, -1);
22            fill(cyc, cyc+V+1, -1);
23            r1 = 0;
24            bool jf = 0;
25            for(int i = 1; i <= v; ++i){
26                if(con[i]) continue;
27                if(prv[i] == -1 && i != root) return -1;
28                if(prv[i] > 0) r1 += mnInW[i];
29                int s;
30                for(s = i; s != -1 && vis[s] == -1; s = prv[s])
31                    vis[s] = i;
32                if(s > 0 && vis[s] == i){
33                    jf = 1; int v = s;
34                    do{ cyc[v] = s, con[v] = 1;
35                        r2 += mnInW[v]; v = prv[v];
36                    }while(v != s);
37                    con[s] = 0;
38                } }
39            if(!jf) break;
40            for(int i = 1; i <= e; ++i){
41                int &u = edges[i].u;
42                int &v = edges[i].v;
```

```

43         if(cyc[v] > 0) edges[i].c -= mnInW[edges[i].v];
44         if(cyc[u] > 0) edges[i].u = cyc[edges[i].u];
45         if(cyc[v] > 0) edges[i].v = cyc[edges[i].v];
46         if(u == v) edges[i--] = edges[E--];
47     } }
48     return r1+r2;}};
```

## 2.5 SW min-cut (不限 S-T 的 min-cut) \*

```

1 struct SW{ // O(V^3)
2     int n,vst[MXN],del[MXN];
3     int edge[MXN][MXN],wei[MXN];
4     void init(int _n){
5         n = _n; memset(del, 0, sizeof(del));
6         memset(edge, 0, sizeof(edge));
7     }
8     void addEdge(int u, int v, int w){
9         edge[u][v] += w; edge[v][u] += w;
10    }
11    void search(int &s, int &t){
12        memset(vst, 0, sizeof(vst)); memset(wei, 0, sizeof(
13            wei));
14        s = t = -1;
15        while (true){
16            int mx=-1, cur=0;
17            for (int i=0; i<n; i++)
18                if (!del[i] && !vst[i] && mx<wei[i])
19                    cur = i, mx = wei[i];
20            if (mx == -1) break;
21            vst[cur] = 1;
22            s = t; t = cur;
23            for (int i=0; i<n; i++)
24                if (!vst[i] && !del[i]) wei[i] += edge[cur][i];
25        }
26    int solve(){
27        int res = 2147483647;
28        for (int i=0,x,y; i<n-1; i++){
29            search(x,y);
30            res = min(res,wei[y]);
31            del[y] = 1;
32            for (int j=0; j<n; j++)
33                edge[x][j] = (edge[j][x] += edge[y][j]);
34        }
35        return res;
36    } }graph;
```

## 2.6 Bounded Max Flow

```

1 // flow use ISAP
2 // Max flow with lower/upper bound on edges
3 // source = 1 , sink = n
4 int in[ N ] , out[ N ];
5 int l[ M ] , r[ M ] , a[ M ] , b[ M ]; //0-base, a下界, b
6 // 上界
7 int solve(){
8     flow.init( n ); //n為點的數量, m為邊的數量, 點是1-
9     base
10    for( int i = 0 ; i < m ; i ++ ){
11        in[ r[ i ] ] += a[ i ];
12        out[ l[ i ] ] += a[ i ];
13        flow.addEdge( l[ i ] , r[ i ] , b[ i ] - a[ i ] );
14        // flow from l[i] to r[i] must in [a[i], b[i]]
15    }
16    int nd = 0;
17    for( int i = 1 ; i <= n ; i ++ ){
18        if( in[ i ] < out[ i ] ){
19            flow.addEdge( i , flow.t , out[ i ] - in[ i ] );
20            nd += out[ i ] - in[ i ];
21        }
22        if( out[ i ] < in[ i ] )
23            flow.addEdge( flow.s , i , in[ i ] - out[ i ] );
24    }
25    // original sink to source
26    flow.addEdge( n , 1 , INF );
27    if( flow.maxflow() != nd )
28        return -1; // no solution
29    int ans = flow.G[ 1 ].back().c; // source to sink
30    flow.G[ 1 ].back().c = flow.G[ n ].back().c = 0;
31    // take out super source and super sink
32    for( size_t i = 0 ; i < flow.G[ flow.s ].size() ; i
33        ++ )
```

```

31 flow.G[ flow.s ][ i ].c = 0;
32 Edge &e = flow.G[ flow.s ][ i ];
33 flow.G[ e.v ][ e.r ].c = 0;
34 }
35 for( size_t i = 0 ; i < flow.G[ flow.t ].size() ; i
    ++ ){
36 flow.G[ flow.t ][ i ].c = 0;
37 Edge &e = flow.G[ flow.t ][ i ];
38 flow.G[ e.v ][ e.r ].c = 0;
39 }
40 flow.addEdge( flow.s , 1 , INF );
41 flow.addEdge( n , flow.t , INF );
42 flow.reset();
43 return ans + flow.maxflow();
44 }

```

## 2.7 Flow Method \*

Maximize  $c^T x$  subject to  $Ax \leq b, x \geq 0$ ;  
 with the corresponding symmetric dual problem,  
 Minimize  $b^T y$  subject to  $A^T y \geq c, y \geq 0$ .  
 Maximize  $c^T x$  subject to  $Ax \leq b$ ;  
 with the corresponding asymmetric dual problem,  
 Minimize  $b^T y$  subject to  $A^T y = c, y \geq 0$ .  
 Minimum vertex cover on bipartite graph =  
 Maximum matching on bipartite graph  
 Minimum edge cover on bipartite graph =  
 vertex number - Minimum vertex cover(Maximum matching)  
 Independent set on bipartite graph =  
 vertex number - Minimum vertex cover(Maximum matching)  
 找出最小點覆蓋，做完 dinic 之後，從源點 dfs 只走還有流量的  
 邊，紀錄每個點有沒有被走到，左邊沒被走到的點跟右邊被走  
 到的點就是答案  
 Maximum density subgraph  $(\sum W_e + \sum W_v)/|V|$   
 Binary search on answer:  
 For a fixed D, construct a Max flow model as follow:  
 Let S be Sum of all weight( or inf)  
 1. from source to each node with cap = S  
 2. For each  $(u,v,w)$  in E,  $(u \rightarrow v, \text{cap}=w)$ ,  $(v \rightarrow u, \text{cap}=w)$   
 3. For each node v, from v to sink with cap =  $S + 2 * D - \deg[v] - 2 * 9$   
 (W of v)  
 where  $\deg[v] = \sum \text{weight of edge associated with v}$   
 If maxflow <  $S * |V|$ , D is an answer.  
 Requiring subgraph: all vertex can be reached from source with  
 edge whose cap > 0.

### • Maximum/Minimum flow with lower bound / Circulation problem

1. Construct super source  $S$  and sink  $T$ .
2. For each edge  $(x,y,l,u)$ , connect  $x \rightarrow y$  with capacity  $u-l$ .
3. For each vertex  $v$ , denote by  $in(v)$  the difference between the sum of incoming lower bounds and the sum of outgoing lower bounds.
4. If  $in(v) > 0$ , connect  $S \rightarrow v$  with capacity  $in(v)$ , otherwise, connect  $v \rightarrow T$  with capacity  $-in(v)$ .
  - To maximize, connect  $t \rightarrow s$  with capacity  $\infty$  (skip this in circulation problem), and let  $f$  be the maximum flow from  $S$  to  $T$ . If  $f \neq \sum_{v \in V, in(v) > 0} in(v)$ , there's no solution. Otherwise, the maximum flow from  $s$  to  $t$  is the answer.
  - To minimize, let  $f$  be the maximum flow from  $S$  to  $T$ . Connect  $t \rightarrow s$  with capacity  $\infty$  and let the flow from  $S$  to  $T$  be  $f'$ . If  $f + f' \neq \sum_{v \in V, in(v) > 0} in(v)$ , there's no solution. Otherwise,  $f'$  is the answer.
5. The solution of each edge  $e$  is  $l_e + f_e$ , where  $f_e$  corresponds to the flow of edge  $e$  on the graph.

### • Construct minimum vertex cover from maximum matching $M$ on bipartite graph $(X,Y)$

1. Redirect every edge:  $y \rightarrow x$  if  $(x,y) \in M$ ,  $x \rightarrow y$  otherwise.
2. DFS from unmatched vertices in  $X$ .
3.  $x \in X$  is chosen iff  $x$  is unvisited.
4.  $y \in Y$  is chosen iff  $y$  is visited.

### • Maximum density induced subgraph

1. Binary search on answer, suppose we're checking answer  $T$
2. Construct a max flow model, let  $K$  be the sum of all weights
3. Connect source  $s \rightarrow v$ ,  $v \in G$  with capacity  $K$
4. For each edge  $(u,v,w)$  in  $G$ , connect  $u \rightarrow v$  and  $v \rightarrow u$  with capacity  $w$
5. For  $v \in G$ , connect it with sink  $v \rightarrow t$  with capacity  $K + 2T - (\sum_{e \in E(v)} w(e)) - 2w(v)$
6.  $T$  is a valid answer if the maximum flow  $f < K|V|$

### • Minimum weight edge cover

1. For each  $v \in V$  create a copy  $v'$ , and connect  $u' \rightarrow v'$  with weight  $w(u,v)$ .
2. Connect  $v \rightarrow v'$  with weight  $2\mu(v)$ , where  $\mu(v)$  is the cost of the cheapest edge incident to  $v$ .
3. Find the minimum weight perfect matching on  $G'$ .

### • Project selection problem

1. If  $p_v > 0$ , create edge  $(s,v)$  with capacity  $p_v$ ; otherwise, create edge  $(v,t)$  with capacity  $-p_v$ .
2. Create edge  $(u,v)$  with capacity  $w$  with  $w$  being the cost of choosing  $u$  without choosing  $v$ .
3. The mincut is equivalent to the maximum profit of a subset of projects.

### • 0/1 quadratic programming

$$\sum_x c_x x + \sum_y c_y \bar{y} + \sum_{xy} c_{xy} x \bar{y} + \sum_{xyx'y'} c_{xyx'y'} (x \bar{y} + x' \bar{y}')$$

can be minimized by the mincut of the following graph:

1. Create edge  $(x,t)$  with capacity  $c_x$  and create edge  $(s,y)$  with capacity  $c_y$ .
2. Create edge  $(x,y)$  with capacity  $c_{xy}$ .
3. Create edge  $(x,y)$  and edge  $(x',y')$  with capacity  $c_{xyx'y'}$ .

## 3 Math

### 3.1 Fast Pow & Inverse & Combination

$fpow(a,b,m) = a^b \pmod{m}$   
 $fa[i] = i! \pmod{MOD}$   
 $fi[i] = i!^{-1} \equiv 1 \pmod{MOD}$   
 $c(a,b) = \binom{a}{b} \pmod{MOD}$

```

1 ll fpow(ll a, ll b, ll m){
2   ll ret = 1;
3   a %= m;
4   while(b){
5     if(b&1) ret = ret * a % m;
6     a = a * a % m;
7     b >>= 1; }
8   return ret; }
9
10 ll fa[MXN], fi[MXN];
11 void init(){
12   fa[0] = 1;
13   for(ll i = 1; i < MXN; ++i)
14     fa[i] = fa[i-1] * i % MOD;
15   fi[MXN-1] = fpow(fa[MXN-1], MOD-2, MOD);
16   for(ll i = MXN-1; i > 0; --i)
17     fi[i] = fi[i+1] * i % MOD; }
18
19 ll c(ll a, ll b){
20   return fa[a] * fi[b] % MOD * fi[a-b] % MOD; }

```

### 3.2 Ext GCD

```

1 //a * p.first + b * p.second = gcd(a, b)
2 pair<ll, ll> extgcd(ll a, ll b) {
3   pair<ll, ll> res;
4   if (a < 0) {
5     res = extgcd(-a, b);
6     res.first *= -1;
7     return res;
8   }
9   if (b < 0) {
10    res = extgcd(a, -b);
11    res.second *= -1;
12    return res;
13   }
14   if (b == 0) return {1, 0};
15   res = extgcd(b, a % b);
16   return {res.second, res.first - res.second * (a / b)};
17 }

```

### 3.3 Sieve 質數篩

```

1 const int MXN = 2e9 + 5; // 2^27 約0.7s, 2^30 約6~7s
2 bool np[MXN]; // np[i] = 1 -> i is'n a prime
3 vector<int> plist; // prime list
4 void sieveBuild(int n){
5   MEM(np, 0);
6   for(int i = 2, sq = sqrt(n); i <= sq; ++i)
7     if(!np[i])
8       for(int j = i * i; j <= n; j += i) np[j] = 1;
9   for(int i = 2; i <= n; ++i) if(!np[i]) plist.PB(i); }

```



### 3.4 FFT \*

```

1 // const int MAXN = 262144;
2 // (must be 2^k)
3 // before any usage, run pre_fft() first
4 typedef long double ld;
5 typedef complex<ld> cplx; //real() ,imag()
6 const ld PI = acos(-1);
7 const cplx I(0, 1);
8 cplx omega[MAXN+1];
9 void pre_fft(){
10     for(int i=0; i<=MAXN; i++){
11         omega[i] = exp(i * 2 * PI / MAXN * I);
12     }
13     // n must be 2^k
14 void fft(int n, cplx a[], bool inv=false){
15     int basic = MAXN / n;
16     int theta = basic;
17     for (int m = n; m >= 2; m >= 1) {
18         int mh = m >> 1;
19         for (int i = 0; i < mh; i++) {
20             cplx w = omega[inv ? MAXN-(i*theta%MAXN)
21                           : i*theta%MAXN];
22             for (int j = i; j < n; j += m) {
23                 int k = j + mh;
24                 cplx x = a[j] - a[k];
25                 a[j] += a[k];
26                 a[k] = w * x;
27             }
28             theta = (theta * 2) % MAXN;
29         }
30         int i = 0;
31         for (int j = 1; j < n - 1; j++) {
32             for (int k = n >> 1; k > (i ^ k); k >= 1);
33             if (j < i) swap(a[i], a[j]);
34         }
35         if(inv) for (i = 0; i < n; i++) a[i] /= n;
36     }
37 cplx arr[MAXN+1];
38 inline void mul(int _n, ll a[], int _m, ll b[], ll ans[])
39 {
40     int n=_n, sum=_n+_m-1;
41     while(n<sum)
42         n<<=1;
43     for(int i=0; i<n; i++){
44         {
45             double x=(i<n?a[i]:0), y=(i<_m?b[i]:0);
46             arr[i]=complex<double>(x+y, x-y);
47         }
48         fft(n, arr);
49         for(int i=0; i<n; i++){
50             arr[i]=arr[i]*arr[i];
51             fft(n, arr, true);
52         }
53         for(int i=0; i<sum; i++){
54             ans[i]=(long long int)(arr[i].real()/4+0.5);
55         }
56     }
57 }

```

### 3.5 NTT \*

```

1 // Remember coefficient are mod P
2 /* p=a*2^n+1
3    n      2^n      p      a      root
4    16    65536    65537    1      3
5    20    1048576   7340033   7      3 */
6 // (must be 2^k)
7 template<LL P, LL root, int MAXN>
8 struct NTT{
9     static LL bigmod(LL a, LL b) {
10         LL res = 1;
11         for (LL bs = a; b; b >= 1, bs = (bs * bs) % P)
12             if(b&1) res=(res*bs)%P;
13         return res;
14     }
15     static LL inv(LL a, LL b) {
16         if(a==1) return 1;
17         return (((LL)(a-inv(b%a,a))*b+1)/a)%b;
18     }
19     LL omega[MAXN+1];
20     NTT() {
21         omega[0] = 1;
22         LL r = bigmod(root, (P-1)/MAXN);
23         for (int i=1; i<=MAXN; i++)

```

```

24         omega[i] = (omega[i-1]*r)%P;
25     }
26     // n must be 2^k
27 void tran(int n, LL a[], bool inv_ntt=false){
28     int basic = MAXN / n, theta = basic;
29     for (int m = n; m >= 2; m >= 1) {
30         int mh = m >> 1;
31         for (int i = 0; i < mh; i++) {
32             LL w = omega[i*theta%MAXN];
33             for (int j = i; j < n; j += m) {
34                 int k = j + mh;
35                 LL x = a[j] - a[k];
36                 if (x < 0) x += P;
37                 a[j] += a[k];
38                 if (a[j] > P) a[j] -= P;
39                 a[k] = (w * x) % P;
40             }
41         }
42         theta = (theta * 2) % MAXN;
43     }
44     int i = 0;
45     for (int j = 1; j < n - 1; j++) {
46         for (int k = n >> 1; k > (i ^ k); k >= 1);
47         if (j < i) swap(a[i], a[j]);
48     }
49     if (inv_ntt) {
50         LL ni = inv(n, P);
51         reverse(a+1, a+n);
52         for (i = 0; i < n; i++)
53             a[i] = (a[i] * ni) % P;
54     }
55 }
56 };
57 const LL P=2013265921, root=31;
58 const int MAXN=4194304;
59 NTT<P, root, MAXN> ntt;

```

### 3.6 Linear Recurrence \*

```

1 // Usage: linearRec({0, 1}, {1, 1}, k) //k'th fib
2 typedef vector<ll> Poly;
3 //S: 前i項的值, tr: 遞迴係數, k: 求第k項
4 ll linearRec(Poly& S, Poly& tr, ll k) {
5     int n = tr.size();
6     auto combine = [&](Poly& a, Poly& b) {
7         Poly res(n * 2 + 1);
8         rep(i, 0, n+1) rep(j, 0, n+1)
9             res[i+j]=(res[i+j] + a[i]*b[j])%mod;
10        for(int i = 2*n; i > n; --i) rep(j, 0, n)
11            res[i-1-j]=(res[i-1-j] + res[i]*tr[j])%mod;
12        res.resize(n + 1);
13        return res;
14    };
15    Poly pol(n + 1), e(pol);
16    pol[0] = e[1] = 1;
17    for (++k; k; k /= 2) {
18        if (k % 2) pol = combine(pol, e);
19        e = combine(e, e);
20    }
21    ll res = 0;
22    rep(i, 0, n) res=(res + pol[i+1]*S[i])%mod;
23    return res;
24 }

```

### 3.7 Miller Rabin

isprime(n) ⇒ 判斷 n 是否為質數  
記得填 magic number

```

1 // magic numbers when n <
2 // 4,759,123,141 : 2, 7, 61
3 // 1,122,004,669,633 : 2, 13, 23, 1662803
4 // 3,474,749,660,383 : 2, 3, 5, 7, 11, 13
5 // 2^64 : 2, 325, 9375, 28178, 450775,
6 // 9780504, 1795265022
7 // Make sure testing integer is in range [2, n-2] if
8 // you want to use magic.
9 vector<ll> magic = {};
10 bool witness(ll a, ll n, ll u, ll t){
11     if(!a) return 0;
12     ll x = fpow(a, u, n);
13     while(t--){
14         ll nx = x * x % n;

```

```

13 if(nx == 1 && x != 1 && x != n - 1) return 1;
14 x = nx; }
15 return x != 1; }
16 bool isprime(ll n) {
17 if(n < 2) return 0;
18 if(~n & 1) return n == 2;
19 ll u = n - 1, t = 0;
20 while(~u & 1) u >>= 1, t++;
21 for(auto i : magic){
22 ll a = i % n;
23 if(witness(a, n, u, t)) return 0; }
24 return 1; }

```

### 3.8 Faulhaber ( $\sum_{i=1}^n i^p$ ) \*

```

1 /* faulhaber' s formula -
2 * cal power sum formula of all p=1~k in O(k^2) */
3 #define MAXK 2500
4 const int mod = 1000000007;
5 int b[MAXK]; // bernoulli number
6 int inv[MAXK+1]; // inverse
7 int cm[MAXK+1][MAXK+1]; // combinactories
8 int co[MAXK][MAXK+2]; // coeeficient of x^j when p=i
9 inline int getinv(int x) {
10 int a=x, b=mod, a0=1, a1=0, b0=0, b1=1;
11 while(b) {
12 int q, t;
13 q=a/b; t=b; b=a-b*q; a=t;
14 t=b0; b0=a0-b0*q; a0=t;
15 t=b1; b1=a1-b1*q; a1=t;
16 }
17 return a0<0?a0+mod:a0;
18 }
19 inline void pre() {
20 /* combinational */
21 for(int i=0; i<=MAXK; i++) {
22 cm[i][0]=cm[i][i]=1;
23 for(int j=1; j<i; j++)
24 cm[i][j]=add(cm[i-1][j-1], cm[i-1][j]);
25 }
26 /* inverse */
27 for(int i=1; i<=MAXK; i++) inv[i]=getinv(i);
28 /* bernoulli */
29 b[0]=1; b[1]=getinv(2); // with b[1] = 1/2
30 for(int i=2; i<MAXK; i++) {
31 if(i&1) { b[i]=0; continue; }
32 b[i]=1;
33 for(int j=0; j<i; j++)
34 b[i]=sub(b[i], mul(cm[i][j], mul(b[j], inv[i-j+1])));
35 }
36 /* faulhaber */
37 // sigma_x=1~n {x^p} =
38 // 1/(p+1) * sigma_j=0~p {C(p+1, j)*Bj*n^(p-j+1)}
39 for(int i=1; i<MAXK; i++) {
40 co[i][0]=0;
41 for(int j=0; j<=i; j++)
42 co[i][i-j+1]=mul(inv[i+1], mul(cm[i+1][j], b[j]));
43 }
44 }
45 }
46 /* sample usage: return f(n,p) = sigma_x=1~n (x^p) */
47 inline int solve(int n, int p) {
48 int sol=0, m=n;
49 for(int i=1; i<=p+1; i++) {
50 sol=add(sol, mul(co[p][i], m));
51 m = mul(m, n);
52 }
53 return sol;
54 }

```

### 3.9 Chinese Remainder \*

```

1 LL x[N], m[N];
2 LL CRT(LL x1, LL m1, LL x2, LL m2) {
3 LL g = __gcd(m1, m2);
4 if((x2 - x1) % g) return -1; // no sol
5 m1 /= g; m2 /= g;
6 pair<LL, LL> p = gcd(m1, m2);
7 LL lcm = m1 * m2 * g;
8 LL res = p.first * (x2 - x1) * m1 + x1;

```

```

9 return (res % lcm + lcm) % lcm;
10 }
11 LL solve(int n) { // n>=2, be careful with no solution
12 LL res=CRT(x[0], m[0], x[1], m[1]), p=m[0]/__gcd(m[0], m
13 [1])*m[1];
14 for(int i=2; i<n; i++){
15 res=CRT(res, p, x[i], m[i]);
16 p=p/__gcd(p, m[i])*m[i];
17 }
18 return res;
19 }

```

### 3.10 Pollard Rho \*

```

1 // does not work when n is prime O(n^(1/4))
2 LL f(LL x, LL mod){ return add(mul(x,x,mod),1,mod); }
3 LL pollard_rho(LL n) {
4 if(!(n&1)) return 2;
5 while(true){
6 LL y=2, x=rand()%(n-1)+1, res=1;
7 for(int sz=2; res==1; sz*=2) {
8 for(int i=0; i<sz && res<=1; i++) {
9 x = f(x, n);
10 res = __gcd(abs(x-y), n);
11 }
12 y = x;
13 }
14 if (res!=0 && res!=n) return res;
15 } }

```

### 3.11 Josephus Problem \*

```

1 int josephus(int n, int m){ //n人 每m次
2 int ans = 0;
3 for (int i=1; i<=n; ++i)
4 ans = (ans + m) % i;
5 return ans;
6 }

```

### 3.12 Gaussian Elimination \*

```

1 const int GAUSS_MOD = 1000000007LL;
2 struct GAUSS{
3 int n;
4 vector<vector<int>> v;
5 int ppow(int a, int k){
6 if(k == 0) return 1;
7 if(k % 2 == 0) return ppow(a * a % GAUSS_MOD,
8 k >> 1);
9 if(k % 2 == 1) return ppow(a * a % GAUSS_MOD,
10 k >> 1) * a % GAUSS_MOD;
11 }
12 vector<int> solve(){
13 vector<int> ans(n);
14 REP(now, 0, n){
15 REP(i, now, n) if(v[now][now] == 0 && v[i
16 ][now] != 0)
17 swap(v[i], v[now]); // det = -det;
18 if(v[now][now] == 0) return ans;
19 int inv = ppow(v[now][now], GAUSS_MOD - 2)
20 ;
21 REP(i, 0, n) if(i != now){
22 int tmp = v[i][now] * inv % GAUSS_MOD;
23 REP(j, now, n + 1) (v[i][j] +=
24 GAUSS_MOD - tmp * v[now][j] %
25 GAUSS_MOD) %= GAUSS_MOD;
26 }
27 }
28 REP(i, 0, n) ans[i] = v[i][n + 1] * ppow(v[i
29 ][i], GAUSS_MOD - 2) % GAUSS_MOD;
30 return ans;
31 }
32 }
33 // gs.v.clear(), gs.v.resize(n, vector<int>(n + 1
34 , 0));
35 } gs;

```

### 3.13 Result \*

- Lucas' Theorem :  
For  $n, m \in \mathbb{Z}^+$  and prime  $P$ ,  $C(m, n) \bmod P = \prod_i (C(m_i, n_i))$  where  $m_i$  is the  $i$ -th digit of  $m$  in base  $P$ .
- Stirling approximation :  
$$n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{\frac{1}{12n}}$$

- Stirling Numbers(permutation  $|P| = n$  with  $k$  cycles):  
 $S(n, k) = \text{coefficient of } x^k \text{ in } \Pi_{i=0}^{n-1} (x+i)$
- Stirling Numbers(Partition  $n$  elements into  $k$  non-empty set):  
 $S(n, k) = \frac{1}{k!} \sum_{j=0}^k (-1)^{k-j} \binom{k}{j} j^n$
- Pick's Theorem :  $A = i + b/2 - 1$   
 其面積  $A$  和內部格點數目  $i$ 、邊上格點數目  $b$  的關係
- Catalan number :  $C_n = \binom{2n}{n} / (n+1)$   
 $C_{n+m} - C_{n+1}^m = (m+n)! \frac{2-m+1}{n+1}$  for  $n \geq m$   
 $C_n = \frac{1}{n+1} \binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$   
 $C_0 = 1$  and  $C_{n+1} = 2 \binom{2n+1}{n+2} C_n$   
 $C_0 = 1$  and  $C_{n+1} = \sum_{i=0}^n C_i C_{n-i}$  for  $n \geq 0$
- Euler Characteristic:  
 planar graph:  $V - E + F - C = 1$   
 convex polyhedron:  $V - E + F = 2$   
 $V, E, F, C$ : number of vertices, edges, faces(regions), and components
- Kirchhoff's theorem :  
 $A_{ii} = \deg(i), A_{ij} = (i, j) \in E ? -1 : 0$ , Deleting any one row, on column, and cal the  $\det(A)$
- Polya's theorem (c 為方法數, m 為總數):  
 $(\sum_{i=1}^m c^{gcd(i, m)}) / m$
- Burnside lemma:  
 $|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$
- 錯排公式: ( $n$  個人中, 每個人皆不再原來位置的組合數):  
 $dp[0] = 1; dp[1] = 0;$   
 $dp[i] = (i-1) * (dp[i-1] + dp[i-2]);$
- Bell 數 (有  $n$  個人, 把他們拆組的方法總數) :  
 $B_0 = 1$   
 $B_n = \sum_{k=0}^n s(n, k)$  (second - stirling)  
 $B_{n+1} = \sum_{k=0}^n \binom{n}{k} B_k$
- Wilson's theorem :  
 $(p-1)! \equiv -1 \pmod{p}$
- Fermat's little theorem :  
 $a^p \equiv a \pmod{p}$
- Euler's totient function:  
 $A^{B^C} \pmod{p} = \text{pow}(A, \text{pow}(B, C, p-1)) \pmod{p}$
- 歐拉函數降冪公式:  
 $A^B \pmod{C} = A^{B \pmod{\phi(C)} + \phi(C)} \pmod{C}$
- 6 的倍數:  
 $(a-1)^3 + (a+1)^3 + (-a)^3 + (-a)^3 = 6a$

## 4 Geometry

### 4.1 definition

```

1 const ld EPS = 1e-8;
2 const ld PI = acos(-1);
3 int dcmp(ld x){ // float x (<, ==, >) y -> (-1, 0, 1)
4   if(abs(x) < EPS) return 0;
5   else return x < 0 ? -1 : 1;
6 }
7 struct Pt{
8   ld x, y; // 改三維記得其他函式都要改
9   Pt(ld _x = 0, ld _y = 0): x(_x), y(_y){}
10  Pt operator+(const Pt &a) const{
11    return Pt(x + a.x, y + a.y); }
12  Pt operator-(const Pt &a) const{
13    return Pt(x - a.x, y - a.y); }
14  Pt operator*(const ld &a) const{
15    return Pt(x * a, y * a); }
16  Pt operator/(const ld &a) const{
17    return Pt(x / a, y / a); }
18  ld operator*(const Pt &a) const{ // dot product
19    return x * a.x + y * a.y; }
20  ld operator^(const Pt &a) const{ // cross product
21    return x * a.y - y * a.x; }
22  bool operator<(const Pt &a) const{
23    return x < a.x || (x == a.x && y < a.y); }
24  // return dcmp(x-a.x) < 0 ||
25  // (dcmp(x-a.x) == 0 && dcmp(y-a.y) < 0); }
26  bool operator==(const Pt &a) const{
27    return dcmp(x - a.x) == 0 && dcmp(y - a.y) == 0; }

```

```

28 int qua() { // 在哪個象限(軸上點歸類到逆時針的象限)
29   if(x > 0 && y >= 0) return 1;
30   if(x <= 0 && y > 0) return 2;
31   if(x < 0 && y <= 0) return 3;
32   if(x >= 0 && y < 0) return 4;
33 }
34 ld angle() const{ // -pi ~ pi
35   if(dcmp(x) == 0 && dcmp(y) == 0) return 0;
36   return atan2(y, x); }
37 }
38 };
39 ld norm2(const Pt &a){
40   return a * a; }
41 ld norm(const Pt &a){ // norm(a - b) = dis of a, b
42   return sqrt(norm2(a)); }
43 Pt perp(const Pt &a){ // 垂直向量(順時針旋轉90度)
44   return Pt(-a.y, a.x); }
45 Pt rotate(const Pt &a, ld ang){
46   return Pt(a.x * cos(ang) - a.y * sin(ang),
47            a.x * sin(ang) + a.y * cos(ang)); }
48 struct Line{
49   Pt s, e, v; // start, end, end - start
50   ld ang; // angle of v
51   Line(Pt _s = Pt(0, 0), Pt _e = Pt(0, 0)):
52     s(_s), e(_e) { v = e - s; ang = atan2(v.y, v.x); }
53   bool operator<(const Line &L) const{ // sort by angle
54     return ang < L.ang;
55   } };
56 struct Circle{
57   Pt o; ld r;
58   Circle(Pt _o = Pt(0, 0), ld _r = 0): o(_o), r(_r){}
59 };

```

### 4.2 halfPlaneIntersection \*

```

1 #define N 100010
2 #define EPS 1e-8
3 #define SIDE 10000000
4 struct PO{ double x, y; } p[ N ], o;
5 struct LI{
6   PO a, b;
7   double angle;
8   void in( double x1, double y1, double x2, double
9     y2 ){
10     a.x = x1; a.y = y1; b.x = x2; b.y = y2;
11 }
12 }li[ N ], deq[ N ];
13 int n, m, cnt;
14 inline int dc( double x ){
15   if ( x > EPS ) return 1;
16   else if ( x < -EPS ) return -1;
17   return 0;
18 }
19 inline PO operator-( PO a, PO b ){
20   PO c;
21   c.x = a.x - b.x; c.y = a.y - b.y;
22   return c;
23 }
24 inline double cross( PO a, PO b, PO c ){
25   return ( b.x - a.x ) * ( c.y - a.y ) - ( b.y - a.y )
26     * ( c.x - a.x );
27 }
28 inline bool cmp( const LI &a, const LI &b ){
29   if( dc( a.angle - b.angle ) == 0 ) return dc( cross(
30     a.a, a.b, b.a ) ) < 0;
31   return a.angle > b.angle;
32 }
33 inline PO getpoint( LI &a, LI &b ){
34   double k1 = cross( a.a, b.b, b.a );
35   double k2 = cross( a.b, b.a, b.b );
36   PO tmp = a.b - a.a, ans;
37   ans.x = a.a.x + tmp.x * k1 / ( k1 + k2 );
38   ans.y = a.a.y + tmp.y * k1 / ( k1 + k2 );
39   return ans;
40 }
41 inline void getcut(){
42   sort( li + 1, li + 1 + n, cmp ); m = 1;
43   for( int i = 2; i <= n; i++ )
44     if( dc( li[ i ].angle - li[ m ].angle ) != 0 )
45       li[ ++m ] = li[ i ];
46   deq[ 1 ] = li[ 1 ]; deq[ 2 ] = li[ 2 ];
47   int bot = 1, top = 2;

```

```

45 for( int i = 3 ; i <= m ; i ++ ){
46     while( bot < top && dc( cross( li[ i ].a , li[ i ].b ,
        getpoint( deq[ top ] , deq[ top - 1 ] ) ) < 0 ) top -- ;
47     while( bot < top && dc( cross( li[ i ].a , li[ i ].b ,
        getpoint( deq[ bot ] , deq[ bot + 1 ] ) ) < 0 ) bot ++ ;
48     deq[ ++ top ] = li[ i ] ;
49 }
50 while( bot < top && dc( cross( deq[ bot ].a , deq[ bot ].b ,
    getpoint( deq[ top ] , deq[ top - 1 ] ) ) < 0 ) top -- ;
51 while( bot < top && dc( cross( deq[ top ].a , deq[ top ].b ,
    getpoint( deq[ bot ] , deq[ bot + 1 ] ) ) < 0 ) bot ++ ;
52 cnt = 0 ;
53 if( bot == top ) return ;
54 for( int i = bot ; i < top ; i ++ ) p[ ++ cnt ] =
    getpoint( deq[ i ] , deq[ i + 1 ] ) ;
55 if( top - 1 > bot ) p[ ++ cnt ] = getpoint( deq[ bot ] ,
    deq[ top ] ) ;
56 }
57 double px[ N ] , py[ N ] ;
58 void read( int rm ) {
59     for( int i = 1 ; i <= n ; i ++ ) px[ i + n ] = px[ i ] ,
        py[ i + n ] = py[ i ] ;
60     for( int i = 1 ; i <= n ; i ++ ){
61         // half-plane from li[ i ].a -> li[ i ].b
62         li[ i ].a.x = px[ i + rm + 1 ] ; li[ i ].a.y = py[ i +
            rm + 1 ] ;
63         li[ i ].b.x = px[ i ] ; li[ i ].b.y = py[ i ] ;
64         li[ i ].angle = atan2( li[ i ].b.y - li[ i ].a.y ,
            li[ i ].b.x - li[ i ].a.x ) ;
65     }
66 }
67 inline double getarea( int rm ){
68     read( rm ) ; getcut() ;
69     double res = 0.0 ;
70     p[ cnt + 1 ] = p[ 1 ] ;
71     for( int i = 1 ; i <= cnt ; i ++ ) res += cross( o ,
        p[ i ] , p[ i + 1 ] ) ;
72     if( res < 0.0 ) res *= -1.0 ;
73     return res ;
74 }

```

### 4.3 Convex Hull \*

```

1 double cross( Pt o , Pt a , Pt b ){
2     return ( a-o ) ^ ( b-o ) ;
3 }
4 vector<Pt> convex_hull( vector<Pt> pt ){
5     sort( pt.begin() , pt.end() ) ;
6     int top=0 ;
7     vector<Pt> stk( 2*pt.size() ) ;
8     for( int i=0 ; i<(int)pt.size() ; i++){
9         while( top >= 2 && cross( stk[top-2] , stk[top-1] , pt[i] )
            <= 0 )
10             top-- ;
11         stk[top++] = pt[i] ;
12     }
13     for( int i=pt.size()-2 , t=top+1 ; i>=0 ; i-- ){
14         while( top >= t && cross( stk[top-2] , stk[top-1] , pt[i] )
            <= 0 )
15             top-- ;
16         stk[top++] = pt[i] ;
17     }
18     stk.resize( top-1 ) ;
19     return stk ;
20 }

```

### 4.4 Convex Hull trick \*

```

1 /* Given a convexhull, answer queries in O( lg N )
2 CH should not contain identical points, the area should
3 be > 0 , min pair(x, y) should be listed first */
4 double det( const Pt& p1 , const Pt& p2 )
5 { return p1.X * p2.Y - p1.Y * p2.X ; }
6 struct Conv{
7     int n ;
8     vector<Pt> a ;
9     vector<Pt> upper , lower ;
10    Conv( vector<Pt> _a ) : a( _a ){

```

```

        n = a.size() ;
        int ptr = 0 ;
        for( int i=1 ; i<n ; ++i ) if( a[ptr] < a[i] ) ptr = i ;
        for( int i=0 ; i<=ptr ; ++i ) lower.push_back( a[i] ) ;
        for( int i=ptr ; i<n ; ++i ) upper.push_back( a[i] ) ;
        upper.push_back( a[0] ) ;
    }
    int sign( LL x ){ // fixed when changed to double
        return x < 0 ? -1 : x > 0 ; }
    pair<LL, int> get_tang( vector<Pt> &conv , Pt vec ){
        int l = 0 , r = (int)conv.size() - 2 ;
        for( ; l + 1 < r ; ){
            int mid = ( l + r ) / 2 ;
            if( sign( det( conv[mid+1] - conv[mid] , vec ) ) > 0 ) r = mid ;
            else l = mid ;
        }
        return max( make_pair( det( vec , conv[r] ) , r ) ,
            make_pair( det( vec , conv[0] ) , 0 ) ) ;
    }
    void upd_tang( const Pt &p , int id , int &i0 , int &i1 ){
        if( det( a[i0] - p , a[id] - p ) > 0 ) i0 = id ;
        if( det( a[i1] - p , a[id] - p ) < 0 ) i1 = id ;
    }
    void bi_search( int l , int r , Pt p , int &i0 , int &i1 ){
        if( l == r ) return ;
        upd_tang( p , l % n , i0 , i1 ) ;
        int sl = sign( det( a[l % n] - p , a[(l + 1) % n] - p ) ) ;
        for( ; l + 1 < r ; ){
            int mid = ( l + r ) / 2 ;
            int smid = sign( det( a[mid % n] - p , a[(mid + 1) % n] - p ) ) ;
            if( smid == sl ) l = mid ;
            else r = mid ;
        }
        upd_tang( p , r % n , i0 , i1 ) ;
    }
    int bi_search( Pt u , Pt v , int l , int r ) {
        int sl = sign( det( v - u , a[l % n] - u ) ) ;
        for( ; l + 1 < r ; ){
            int mid = ( l + r ) / 2 ;
            int smid = sign( det( v - u , a[mid % n] - u ) ) ;
            if( smid == sl ) l = mid ;
            else r = mid ;
        }
        return l % n ;
    }
    // 1. whether a given point is inside the CH
    bool contain( Pt p ) {
        if( p.X < lower[0].X || p.X > lower.back().X )
            return 0 ;
        int id = lower_bound( lower.begin() , lower.end() , Pt( p.X ,
            -INF ) ) - lower.begin() ;
        if( lower[id].X == p.X ) {
            if( lower[id].Y > p.Y ) return 0 ;
        } else if( det( lower[id-1] - p , lower[id] - p ) < 0 ) return 0 ;
        id = lower_bound( upper.begin() , upper.end() , Pt( p.X ,
            INF ) , greater<Pt>() ) - upper.begin() ;
        if( upper[id].X == p.X ) {
            if( upper[id].Y < p.Y ) return 0 ;
        } else if( det( upper[id-1] - p , upper[id] - p ) < 0 ) return 0 ;
        return 1 ;
    }
    // 2. Find 2 tang pts on CH of a given outside point
    // return true with i0 , i1 as index of tangent points
    // return false if inside CH
    bool get_tang( Pt p , int &i0 , int &i1 ) {
        if( contain( p ) ) return false ;
        i0 = i1 = 0 ;
        int id = lower_bound( lower.begin() , lower.end() , p ) -
            lower.begin() ;
        bi_search( 0 , id , p , i0 , i1 ) ;
        bi_search( id , (int)lower.size() , p , i0 , i1 ) ;
        id = lower_bound( upper.begin() , upper.end() , p ,
            greater<Pt>() ) - upper.begin() ;
        bi_search( (int)lower.size() - 1 , (int)lower.size() -
            1 + id , p , i0 , i1 ) ;
        bi_search( (int)lower.size() - 1 + id , (int)lower.size() -
            1 + (int)upper.size() , p , i0 , i1 ) ;
        return true ;
    }
    // 3. Find tangent points of a given vector
    // ret the idx of vertex has max cross value with vec
    int get_tang( Pt vec ){

```



```

86 pair<LL, int> ret = get_tang(upper, vec);
87 ret.second = (ret.second+(int)lower.size()-1)%n;
88 ret = max(ret, get_tang(lower, vec));
89 return ret.second;
90 }
91 // 4. Find intersection point of a given line
92 // return 1 and intersection is on edge (i, next(i))
93 // return 0 if no strictly intersection
94 bool get_intersection(Pt u, Pt v, int &i0, int &i1){
95     int p0 = get_tang(u - v), p1 = get_tang(v - u);
96     if(sign(det(v-u,a[p0]-u))*sign(det(v-u,a[p1]-u))<0){
97         if (p0 > p1) swap(p0, p1);
98         i0 = bi_search(u, v, p0, p1);
99         i1 = bi_search(u, v, p1, p0 + n);
100         return 1;
101     }
102     return 0;
103 }

```

#### 4.5 掃描的線

```

1 ScanLine sl;
2 sl.add(兩點座標);
3 sl.run()
4
5 template <typename T>
6 struct SegmentTree{
7     struct Node{
8         T len = 0, tag = 0;
9         int nl, nr;
10        Node *l, *r;
11    } *root;
12    vector<T> vec;
13    int n;
14    SegmentTree(){
15        void init(vector<T> _vec){
16            vec = _vec;
17            n = vec.size() - 1;
18            root = build(0, n - 1);
19        }
20        Node* build(int l, int r){
21            Node *res = new Node();
22            res->nl = l, res->nr = r;
23            if(l == r){
24                res->l = res->r = nullptr;
25                return res;
26            }
27            int mid = (l + r) >> 1;
28            res->l = build(l, mid);
29            res->r = build(mid + 1, r);
30            return res;
31        }
32        void push(Node *cur){
33            int l = cur->nl, r = cur->nr;
34            if(cur->tag) cur->len = vec[r + 1] - vec[l];
35            else cur->len = l == r ? 0 : cur->l->len + cur->r->len;
36        }
37        void update(Node *cur, int ql, int qr, int x){
38            int l = cur->nl, r = cur->nr;
39            if(vec[r + 1] <= ql || qr <= vec[l]) return;
40            if(ql <= vec[l] && vec[r + 1] <= qr){
41                cur->tag += x;
42                push(cur);
43                return;
44            }
45            update(cur->l, ql, qr, x);
46            update(cur->r, ql, qr, x);
47            push(cur);
48        }
49        void update(int l, int r, int x){
50            update(root, l, r, x);
51        }
52    };
53 template <typename T>
54 struct ScanLine{
55     struct Line{
56         T l, r, h, flag;
57         bool operator<(const Line &rhs){
58             return h < rhs.h;
59         }
60     };

```

```

61 vector<T> vec; vector<Line> line; SegmentTree<T> seg;
62 int n, cnt = 0;
63 ScanLine<int> _n: n(_n << 1) {
64     line.resize(n), vec.resize(n);
65 }
66 void add(int x1, int y1, int x2, int y2){
67     line[cnt] = {x1, x2, y1, 1}, line[cnt + 1] = {x1,
68         x2, y2, -1};
69     vec[cnt] = x1, vec[cnt + 1] = x2;
70     cnt += 2;
71 }
72 T run(){
73     T res = 0;
74     sort(line.begin(), line.end());
75     sort(vec.begin(), vec.end());
76     vec.erase(unique(vec.begin(), vec.end()), vec.end());
77     seg.init(vec);
78     for(int i = 0; i < n - 1; ++i){
79         seg.update(line[i].l, line[i].r, line[i].flag);
80         res += seg.root->len * (line[i + 1].h - line[i].h);
81     }
82     return res;
83 }

```

#### 4.6 Polar sort

```

1 sort(pl.begin(), pl.end(), [&](Pt a, Pt b){
2     // a = a - o, b = b - o;
3     if(a.qua() == b.qua()) return (a ^ b) > 0;
4     return a.qua() < b.qua();
5 }); // degree 0 to 359
6 sort(pl.begin(), pl.end(), [&](Pt a, Pt b){
7     return (a - pt[i]).angle() < (b - pt[i]).angle();
8 }); // degree -180 to 180, slower

```

#### 4.7 Li Chao Segment Tree \*

```

1 struct LiChao_min{
2     struct line{
3         ll m, c;
4         line(ll _m=0, ll _c=0){ m=_m; c=_c; }
5         ll eval(ll x){ return m*x+c; } // overflow
6     };
7     struct node{
8         node *l, *r; line f;
9         node(line v){ f=v; l=r=NULL; }
10    };
11    typedef node* pnode;
12    pnode root; ll sz, ql, qr;
13    #define mid ((l+r)>>1)
14    void insert(line v, ll l, ll r, pnode &nd){
15        /* if(!(ql<=l&&r<=qr)){
16            if(!nd) nd=new node(line(0,INF));
17            if(ql<=mid) insert(v,l,mid,nd->l);
18            if(qr>mid) insert(v,mid+1,r,nd->r);
19            return;
20        } used for adding segment */
21        if(!nd){ nd=new node(v); return; }
22        ll trl=nd->f.eval(l), trr=nd->f.eval(r);
23        ll vl=v.eval(l), vr=v.eval(r);
24        if(trl<=vl&&trr<=vr) return;
25        if(trl>vl&&trr>vr) { nd->f=v; return; }
26        if(trl>vl) swap(nd->f,v);
27        if(nd->f.eval(mid)<v.eval(mid))
28            insert(v,mid+1,r,nd->r);
29        else swap(nd->f,v), insert(v,l,mid,nd->l);
30    }
31    ll query(ll x, ll l, ll r, pnode &nd){
32        if(!nd) return INF;
33        if(l==r) return nd->f.eval(x);
34        if(mid>=x)
35            return min(nd->f.eval(x), query(x,l,mid,nd->l));
36        return min(nd->f.eval(x), query(x,mid+1,r,nd->r));
37    }
38    /* -sz<=ll query_x<=sz */
39    void init(ll _sz){ sz=_sz+1; root=NULL; }
40    void add_line(ll m, ll c, ll l=-INF, ll r=INF){
41        line v(m,c); ql=l; qr=r; insert(v,-sz,sz,root);
42    }

```

```
43 ll query(ll x) { return query(x,-sz,sz,root); }
44 };
```

## 4.8 KD Tree \*

```
1 struct KDTree{ // O(sqrtN + K)
2     struct Nd{
3         LL x[MXK],mn[MXK],mx[MXK];
4         int id,f;
5         Nd *l,*r;
6     }tree[MXN],*root;
7     int n,k;
8     LL dis(LL a,LL b){return (a-b)*(a-b);}
9     LL dis(LL a[MXK],LL b[MXK]){
10         LL ret=0;
11         for(int i=0;i<k;i++) ret+=dis(a[i],b[i]);
12         return ret;
13     }
14     void init(vector<vector<LL>> &ip,int _n,int _k){
15         n=_n,k=_k;
16         for(int i=0;i<n;i++){
17             tree[i].id=i;
18             copy(ip[i].begin(),ip[i].end(),tree[i].x);
19         }
20         root=build(0,n-1,0);
21     }
22     Nd* build(int l,int r,int d){
23         if(l>r) return NULL;
24         if(d==k) d=0;
25         int m=(l+r)>>1;
26         nth_element(tree+l,tree+m,tree+r+1,[&](const Nd &a,
27             const Nd &b){return a.x[d]<b.x[d];});
28         tree[m].f=d;
29         copy(tree[m].x,tree[m].x+k,tree[m].mn);
30         copy(tree[m].x,tree[m].x+k,tree[m].mx);
31         tree[m].l=build(l,m-1,d+1);
32         if(tree[m].l){
33             for(int i=0;i<k;i++){
34                 tree[m].mn[i]=min(tree[m].mn[i],tree[m].l->mn[i]);
35                 tree[m].mx[i]=max(tree[m].mx[i],tree[m].l->mx[i]);
36             }
37         }
38         tree[m].r=build(m+1,r,d+1);
39         if(tree[m].r){
40             for(int i=0;i<k;i++){
41                 tree[m].mn[i]=min(tree[m].mn[i],tree[m].r->mn[i]);
42                 tree[m].mx[i]=max(tree[m].mx[i],tree[m].r->mx[i]);
43             }
44         }
45         return tree+m;
46     }
47     LL pt[MXK],md;
48     int mID;
49     bool touch(Nd *r){
50         LL d=0;
51         for(int i=0;i<k;i++){
52             if(pt[i]<=r->mn[i]) d+=dis(pt[i],r->mn[i]);
53             else if(pt[i]>=r->mx[i]) d+=dis(pt[i],r->mx[i]);
54         }
55         return d<md;
56     }
57     void nearest(Nd *r){
58         if(!r||!touch(r)) return;
59         LL td=dis(r->x,pt);
60         if(td<md) md=td,mID=r->id;
61         nearest(pt[r->f]<r->x[r->f]?r->l:r->r);
62         nearest(pt[r->f]>r->x[r->f]?r->r:r->l);
63     }
64     pair<LL,int> query(vector<LL> &_pt,LL _md=1LL<<57){
65         mID=-1,md=_md;
66         copy(_pt.begin(),_pt.end(),pt);
67         nearest(root);
68         return {md,mID};
69     } }tree;
```

## 5 Tree

### 5.1 LCA

求樹上兩點的最低共同祖先

lca.init(n)  $\Rightarrow$  0-base  
 lca.addEdge(u, v)  $\Rightarrow u \leftrightarrow v$   
 lca.build(root, root)  $\Rightarrow O(n \lg n)$   
 lca.qlca(u, v)  $\Rightarrow O(\lg n)$  u, v 的 LCA  
 lca.qdis(u, v)  $\Rightarrow O(\lg n)$  u, v 的距離 (可用倍增法帶權)  
 lca.anc[u][i]  $\Rightarrow$  u 的第  $2^i$  個祖先

```
1 const int MXN = 5e5+5;
2 struct LCA{
3     int n, lgn, ti = 0;
4     int anc[MXN][24], in[MXN], out[MXN];
5     vector<int> g[MXN];
6     void init(int _n){
7         n = _n, lgn = __lg(n) + 5;
8         for(int i = 0; i < n; ++i) g[i].clear();
9     }
10    void addEdge(int u, int v){ g[u].PB(v), g[v].PB(u); }
11    void build(int u, int f){
12        in[u] = ti++;
13        int cur = f;
14        for(int i = 0; i < lgn; ++i)
15            anc[u][i] = cur, cur = anc[cur][i];
16        for(auto i : g[u]) if(i != f) build(i, u);
17        out[u] = ti++;
18    }
19    bool isanc(int a, int u){
20        return in[a] <= in[u] && out[a] >= out[u];
21    }
22    int qlca(int u, int v){
23        if(isanc(u, v)) return u;
24        if(isanc(v, u)) return v;
25        for(int i = lgn-1; i >= 0; --i)
26            if(!isanc(anc[u][i], v)) u = anc[u][i];
27        return anc[u][0];
28    }
29    int qdis(int u, int v){
30        int dis = !isanc(u, v) + !isanc(v, u);
31        for(int i = lgn-1; i >= 0; --i){
32            if(!isanc(anc[u][i], v))
33                u = anc[u][i], dis += 1<<i;
34            if(!isanc(anc[v][i], u))
35                v = anc[v][i], dis += 1<<i;
36        }
37        return dis;
38    } };
```

## 6 Graph

### 6.1 HeavyLightDecomposition \*

```
1 const int MXN = 200005;
2 template <typename T>
3 struct HeavyDecompose{ // 1-base, Need "ulimit -s unlimited"
4     SegmentTree<T> st;
5     vector<T> vec, tmp; // If tree point has weight
6     vector<int> e[MXN];
7     int sz[MXN], dep[MXN], fa[MXN], h[MXN];
8     int cnt = 0, r = 0, n = 0;
9     int root[MXN], id[MXN];
10    void addEdge(int a, int b){
11        e[a].emplace_back(b);
12        e[b].emplace_back(a);
13    }
14    HeavyDecompose(int n, int r): n(n), r(r){
15        vec.resize(n+1); tmp.resize(n+1);
16    }
17    void build(){
18        dfs1(r, 0, 0);
19        dfs2(r, r);
20        st.init(tmp); // SegmentTree Need Add Method
21    }
22    void dfs1(int x, int f, int d){
23        dep[x] = d, fa[x] = f, sz[x] = 1, h[x] = 0;
24        for(int i : e[x]){
25            if(i == f) continue;
26            dfs1(i, x, d+1);
27            sz[x] += sz[i];
28            if(sz[i] > sz[h[x]]) h[x] = i;
29        }
30    }
31    void dfs2(int x, int f){
32        id[x] = cnt++, root[x] = f, tmp[id[x]] = vec[x];
33        if(!h[x]) return;
34        dfs2(h[x], f);
35        for(int i : e[x]){
36            if(i == fa[x] || i == h[x]) continue;
37            dfs2(i, i);
38        }
39    }
```

```

39 }
40 void update(int x, int y, T v){
41     while(root[x] != root[y]){
42         if(dep[root[x]] < dep[root[y]]) swap(x, y);
43         st.update(id[root[x]], id[x], v);
44         x = fa[root[x]];
45     }
46     if(dep[x] > dep[y]) swap(x, y);
47     st.update(id[x], id[y], v);
48 }
49 T query(int x, int y){
50     T res = 0;
51     while(root[x] != root[y]){
52         if(dep[root[x]] < dep[root[y]]) swap(x, y);
53         res = (st.query(id[root[x]], id[x]) + res) % MOD;
54         x = fa[root[x]];
55     }
56     if(dep[x] > dep[y]) swap(x, y);
57     res = (st.query(id[x], id[y]) + res) % MOD;
58     return res;
59 }
60 void update(int x, T v){
61     st.update(id[x], id[x] + sz[x] - 1, v);
62 }
63 T query(int x){
64     return st.query(id[x], id[x] + sz[x] - 1);
65 }
66 int getLca(int x, int y){
67     while(root[x] != root[y]){
68         if(dep[root[x]] > dep[root[y]]) x = fa[root[x]];
69         else y = fa[root[y]];
70     }
71     return dep[x] > dep[y] ? y : x;
72 }
73 };

```

## 6.2 Centroid Decomposition \*

```

1 struct CentroidDecomposition {
2     int n;
3     vector<vector<int>> G, out;
4     vector<int> sz, v;
5     CentroidDecomposition(int _n) : n(_n), G(_n), out(
6         _n), sz(_n), v(_n) {}
7     int dfs(int x, int par){
8         sz[x] = 1;
9         for (auto &i : G[x]) {
10             if(i == par || v[i]) continue;
11             sz[x] += dfs(i, x);
12         }
13         return sz[x];
14     }
15     int search_centroid(int x, int p, const int mid){
16         for (auto &i : G[x]) {
17             if(i == p || v[i]) continue;
18             if(sz[i] > mid) return search_centroid(i,
19                 x, mid);
20         }
21         return x;
22     }
23     void add_edge(int l, int r){
24         G[l].PB(r); G[r].PB(l);
25     }
26     int get(int x){
27         int centroid = search_centroid(x, -1, dfs(x,
28             -1)/2);
29         v[centroid] = true;
30         for (auto &i : G[centroid]) {
31             if(!v[i]) out[centroid].PB(get(i));
32         }
33         v[centroid] = false;
34         return centroid;
35     }
36 };

```

## 6.3 DominatorTree \*

```

1 struct DominatorTree{ // O(N)
2     #define REP(i,s,e) for(int i=(s);i<=(e);i++)
3     #define REPD(i,s,e) for(int i=(s);i>=(e);i--)
4     int n, m, s;
5     vector<int> g[ MAXN ], pred[ MAXN ];
6     vector<int> cov[ MAXN ];

```

```

7     int dfn[ MAXN ], nfd[ MAXN ], ts;
8     int par[ MAXN ]; //idom[u] s到u的最後一個必經點
9     int sdom[ MAXN ], idom[ MAXN ];
10    int mom[ MAXN ], mn[ MAXN ];
11    inline bool cmp( int u, int v )
12    { return dfn[ u ] < dfn[ v ]; }
13    int eval( int u ){
14        if( mom[ u ] == u ) return u;
15        int res = eval( mom[ u ] );
16        if(cmp( sdom[ mn[ mom[ u ] ] ], sdom[ mn[ u ] ] ))
17            mn[ u ] = mn[ mom[ u ] ];
18        return mom[ u ] = res;
19    }
20    void init( int _n, int _m, int _s ){
21        ts = 0; n = _n; m = _m; s = _s;
22        REP( i, 1, n ) g[ i ].clear(), pred[ i ].clear();
23    }
24    void addEdge( int u, int v ){
25        g[ u ].push_back( v );
26        pred[ v ].push_back( u );
27    }
28    void dfs( int u ){
29        ts++;
30        dfn[ u ] = ts;
31        nfd[ ts ] = u;
32        for( int v : g[ u ] ) if( dfn[ v ] == 0 ){
33            par[ v ] = u;
34            dfs( v );
35        }
36    }
37    void build(){
38        REP( i, 1, n ){
39            dfn[ i ] = nfd[ i ] = 0;
40            cov[ i ].clear();
41            mom[ i ] = mn[ i ] = sdom[ i ] = i;
42        }
43        dfs( s );
44        REPD( i, n, 2 ){
45            int u = nfd[ i ];
46            if( u == 0 ) continue;
47            for( int v : pred[ u ] ) if( dfn[ v ] ){
48                eval( v );
49                if( cmp( sdom[ mn[ v ] ], sdom[ u ] ) )
50                    sdom[ u ] = sdom[ mn[ v ] ];
51            }
52            cov[ sdom[ u ] ].push_back( u );
53            mom[ u ] = par[ u ];
54            for( int w : cov[ par[ u ] ] ){
55                eval( w );
56                if( cmp( sdom[ mn[ w ] ], par[ u ] ) )
57                    idom[ w ] = mn[ w ];
58                else idom[ w ] = par[ u ];
59            }
60            cov[ par[ u ] ].clear();
61        }
62        REP( i, 2, n ){
63            int u = nfd[ i ];
64            if( u == 0 ) continue;
65            if( idom[ u ] != sdom[ u ] )
66                idom[ u ] = idom[ idom[ u ] ];
67        }
68    }
69 };

```

## 6.4 MaximumClique 最大團 \*

```

1 #define N 111
2 struct MaxClique{ // 0-base
3     typedef bitset<N> Int;
4     Int linkto[N], v[N];
5     int n;
6     void init(int _n){
7         n = _n;
8         for(int i = 0 ; i < n ; i++){
9             linkto[i].reset(); v[i].reset();
10        }
11    }
12    void addEdge(int a, int b)
13    { v[a][b] = v[b][a] = 1; }
14    int popcount(const Int& val)
15    { return val.count(); }
16    int lowbit(const Int& val)
17    { return val._Find_first(); }
18    int ans, stk[N];
19    int id[N], di[N], deg[N];
20    Int cans;

```

```

20 void maxclique(int elem_num, Int candi){
21     if(elem_num > ans){
22         ans = elem_num; cans.reset();
23         for(int i = 0 ; i < elem_num ; i ++){
24             cans[id[stk[i]]] = 1;
25         }
26         int potential = elem_num + popcount(candi);
27         if(potential <= ans) return;
28         int pivot = lowbit(candi);
29         Int smaller_candi = candi & (~linkto[pivot]);
30         while(smaller_candi.count() && potential > ans){
31             int next = lowbit(smaller_candi);
32             candi[next] = !candi[next];
33             smaller_candi[next] = !smaller_candi[next];
34             potential --;
35             if(next == pivot || (smaller_candi & linkto[next
36                 ]).count()){
37                 stk[elem_num] = next;
38                 maxclique(elem_num + 1, candi & linkto[next]);
39             } }
40         int solve(){
41             for(int i = 0 ; i < n ; i ++){
42                 id[i] = i; deg[i] = v[i].count();
43             }
44             sort(id , id + n , [&](int id1, int id2){
45                 return deg[id1] > deg[id2]; });
46             for(int i = 0 ; i < n ; i ++){ di[id[i]] = i;
47             for(int i = 0 ; i < n ; i ++){
48                 for(int j = 0 ; j < n ; j ++){
49                     if(v[i][j]) linkto[di[i]][di[j]] = 1;
50                 }
51                 Int cand; cand.reset();
52                 for(int i = 0 ; i < n ; i ++){ cand[i] = 1;
53                 ans = 1;
54                 cans.reset(); cans[0] = 1;
55                 maxclique(0, cand);
56                 return ans;
57             } }
58     } }
59 } } solver;

```

## 6.5 MaximalClique 極大團 \*

```

1 #define N 80
2 struct MaxClique{ // 0-base
3     typedef bitset<N> Int;
4     Int lnk[N] , v[N];
5     int n;
6     void init(int _n){
7         n = _n;
8         for(int i = 0 ; i < n ; i ++){
9             lnk[i].reset(); v[i].reset();
10        } }
11     void addEdge(int a , int b)
12     { v[a][b] = v[b][a] = 1; }
13     int ans , stk[N], id[N] , di[N] , deg[N];
14     Int cans;
15     void dfs(int elem_num, Int candi, Int ex){
16         if(candi.none() && ex.none()){
17             cans.reset();
18             for(int i = 0 ; i < elem_num ; i ++){
19                 cans[id[stk[i]]] = 1;
20                 ans = elem_num; // cans is a maximal clique
21                 return;
22             }
23             int pivot = (candilex)._Find_first();
24             Int smaller_candi = candi & (~lnk[pivot]);
25             while(smaller_candi.count()){
26                 int nxt = smaller_candi._Find_first();
27                 candi[nxt] = smaller_candi[nxt] = 0;
28                 ex[nxt] = 1;
29                 stk[elem_num] = nxt;
30                 dfs(elem_num+1, candi & lnk[nxt], ex & lnk[nxt]);
31             } }
32     int solve(){
33         for(int i = 0 ; i < n ; i ++){
34             id[i] = i; deg[i] = v[i].count();
35         }
36         sort(id , id + n , [&](int id1, int id2){
37             return deg[id1] > deg[id2]; });
38         for(int i = 0 ; i < n ; i ++){ di[id[i]] = i;
39         for(int i = 0 ; i < n ; i ++){
40             for(int j = 0 ; j < n ; j ++){
41                 if(v[i][j]) lnk[di[i]][di[j]] = 1;
42             }
43             ans = 1; cans.reset(); cans[0] = 1;

```

```

43     dfs(0, Int(string(n, '1')), 0);
44     return ans;
45 } } solver;

```

## 6.6 Minimum Steiner Tree

```

1 const int MXNN = 105;
2 const int MXNK = 10 + 1;
3 template<typename T>
4 struct SteinerTree{ // 有重要點的MST權重和, 1-base
5     int n, k;
6     T inf;
7     vector<vector<T>> dp;
8     vector<vector<pair<int, T>>> edge;
9     priority_queue<pair<T, int>, vector<pair<T, int>>,
10         greater<pair<T, int>>> pq;
11     vector<int> vis;
12     void init(int _n, int _k, T _inf){
13         // n points, 1~k 是重要點, type T的INF
14         n = _n, k = _k, inf = _inf;
15         dp.assign(n + 1, vector<T>(1 << k, inf));
16         edge.resize(n + 1); }
17     void addEdge(int u, int v, T w){ // u <-(w)-> v
18         edge[u].emplace_back(v, w);
19         edge[v].emplace_back(u, w); }
20     void dijkstra(int s, int cnt){
21         vis.assign(n + 1, 0);
22         while(!pq.empty()){
23             auto [d, u] = pq.top(); pq.pop();
24             if(vis[u]) continue;
25             vis[u] = 1;
26             for(auto &[v, w] : edge[u]){
27                 // if(cnt > 1 && v <= k) continue;
28                 if(dp[v][s] > dp[u][s] + w){
29                     dp[v][s] = dp[u][s] + w;
30                     pq.push({dp[v][s], v}); } } }
31     T run(){ // return total cost 0(nk*2^k + n^2*2^k)
32         for(int i = 1; i <= k; ++i) dp[i][1 << (i - 1)] = 0;
33         for(int s = 1; s < (1 << k); ++s){
34             int cnt = 0, tmp = s;
35             while(tmp) cnt += (tmp & 1), tmp >>= 1;
36             for(int i = k + 1; i <= n; ++i){
37                 for(int sb = s & (s-1); sb; sb = s & (sb-1))
38                     dp[i][s] =
39                     min(dp[i][s], dp[i][sb] + dp[i][s ^ sb]);
40             for(int i = (cnt > 1 ? k + 1 : 1); i <= n; ++i){
41                 if(dp[i][s] != inf) pq.push({dp[i][s], i});
42             dijkstra(s, cnt); }
43         T res = inf;
44         for(int i = 1; i <= n; ++i){
45             res = min(res, dp[i][(1 << k) - 1]);
46         return res; } } }

```

## 6.7 BCC based on vertex \*

```

1 struct BccVertex {
2     int n, nScc, step, dfn[MXN], low[MXN];
3     vector<int> E[MXN], sccv[MXN];
4     int top, stk[MXN];
5     void init(int _n) {
6         n = _n; nScc = step = 0;
7         for (int i=0; i<n; i++) E[i].clear();
8     }
9     void addEdge(int u, int v)
10     { E[u].PB(v); E[v].PB(u); }
11     void DFS(int u, int f) {
12         dfn[u] = low[u] = step++;
13         stk[top++] = u;
14         for (auto v:E[u]) {
15             if (v == f) continue;
16             if (dfn[v] == -1) {
17                 DFS(v, u);
18                 low[u] = min(low[u], low[v]);
19                 if (low[v] >= dfn[u]) {
20                     int z;
21                     sccv[nScc].clear();
22                     do {
23                         z = stk[--top];
24                         sccv[nScc].PB(z);
25                     } while (z != v);
26                     sccv[nScc++].PB(u);
27                 }

```



```

28     }else
29         low[u] = min(low[u],dfn[v]);
30 } }
31 vector<vector<int>> solve() {
32     vector<vector<int>> res;
33     for (int i=0; i<n; i++)
34         dfn[i] = low[i] = -1;
35     for (int i=0; i<n; i++)
36         if (dfn[i] == -1) {
37             top = 0;
38             DFS(i,i);
39         }
40     REP(i,nScc) res.PB(sccv[i]);
41     return res;
42 }
43 }graph;

```

## 6.8 Strongly Connected Component \*

```

1 struct Scc{
2     int n, nScc, vst[MXN], bln[MXN];
3     vector<int> E[MXN], rE[MXN], vec;
4     void init(int _n){
5         n = _n;
6         for (int i=0; i<MXN; i++)
7             E[i].clear(), rE[i].clear();
8     }
9     void addEdge(int u, int v){
10         E[u].PB(v); rE[v].PB(u);
11     }
12     void DFS(int u){
13         vst[u]=1;
14         for (auto v : E[u]) if (!vst[v]) DFS(v);
15         vec.PB(u);
16     }
17     void rDFS(int u){
18         vst[u] = 1; bln[u] = nScc;
19         for (auto v : rE[u]) if (!vst[v]) rDFS(v);
20     }
21     void solve(){
22         nScc = 0;
23         vec.clear();
24         FZ(vst);
25         for (int i=0; i<n; i++)
26             if (!vst[i]) DFS(i);
27         reverse(vec.begin(),vec.end());
28         FZ(vst);
29         for (auto v : vec)
30             if (!vst[v]){
31                 rDFS(v); nScc++;
32             }
33     }
34 };

```

## 6.9 差分約束 \*

約束條件  $V_j - V_i \leq W$  建邊  $V_i \rightarrow V_j$  權重為  $W$  → bellman-ford or spfa

# 7 String

## 7.1 PalTree \*

```

1 // len[s]是對應的回文長度
2 // num[s]是有幾個回文後綴
3 // cnt[s]是這個回文字串在整個字串中的出現次數
4 // fail[s]是他長度次長的回文後綴，aba的fail是a
5 const int MXN = 1000010;
6 struct PalT{
7     int nxt[MXN][26], fail[MXN], len[MXN];
8     int tot, lst, n, state[MXN], cnt[MXN], num[MXN];
9     int diff[MXN], sfail[MXN], fac[MXN], dp[MXN];
10    char s[MXN]={'-1'};
11    int newNode(int l, int f){
12        len[tot]=l, fail[tot]=f, cnt[tot]=num[tot]=0;
13        memset(nxt[tot], 0, sizeof(nxt[tot]));
14        diff[tot]=(l>0?l-len[f]:0);
15        sfail[tot]=(l>0&&diff[tot]==diff[f]?sfail[f]:f);
16        return tot++;
17    }
18    int getfail(int x){
19        while(s[n-len[x]-1]!=s[n]) x=fail[x];
20        return x;
21    }

```

```

22    int getmin(int v){
23        dp[v]=fac[n-len[sfail[v]]-diff[v]];
24        if(diff[v]==diff[fail[v]])
25            dp[v]=min(dp[v], dp[fail[v]]);
26        return dp[v]+1;
27    }
28    int push(){
29        int c=s[n]-'a', np=getfail(lst);
30        if(!(lst=nxt[np][c])){
31            lst=newNode(len[np]+2, nxt[getfail(fail[np])][c]);
32            nxt[np][c]=lst; num[lst]=num[fail[lst]]+1;
33        }
34        fac[n]=n;
35        for(int v=lst; len[v]>0; v=sfail[v])
36            fac[n]=min(fac[n], getmin(v));
37        return ++cnt[lst], lst;
38    }
39    void init(const char *_s){
40        tot=lst=n=0;
41        newNode(0,1), newNode(-1,1);
42        for(; _s[n];) s[n+1]=_s[n], ++n, state[n-1]=push();
43        for(int i=tot-1; i>1; i--) cnt[fail[i]]+=cnt[i];
44    }
45 }palt;

```

## 7.2 SuffixArray \*

```

1 const int MAX = 1020304;
2 int ct[MAX], he[MAX], rk[MAX];
3 int sa[MAX], tsa[MAX], tp[MAX][2];
4 void suffix_array(char *ip){
5     int len = strlen(ip);
6     int alp = 256;
7     memset(ct, 0, sizeof(ct));
8     for(int i=0; i<len; i++) ct[ip[i]+1]++;
9     for(int i=1; i<alp; i++) ct[i]+=ct[i-1];
10    for(int i=0; i<len; i++) rk[i]=ct[ip[i]];
11    for(int i=1; i<len; i*=2){
12        for(int j=0; j<len; j++){
13            if(j+i>len) tp[j][1]=0;
14            else tp[j][1]=rk[j+i]+1;
15            tp[j][0]=rk[j];
16        }
17        memset(ct, 0, sizeof(ct));
18        for(int j=0; j<len; j++) ct[tp[j][1]+1]++;
19        for(int j=1; j<len+2; j++) ct[j]+=ct[j-1];
20        for(int j=0; j<len; j++) sa[ct[tp[j][1]]++] = j;
21        memset(ct, 0, sizeof(ct));
22        for(int j=0; j<len; j++) ct[tp[j][0]+1]++;
23        for(int j=1; j<len+1; j++) ct[j]+=ct[j-1];
24        for(int j=0; j<len; j++)
25            sa[ct[tp[tsa[j]][0]]++] = tsa[j];
26        rk[sa[0]]=0;
27        for(int j=1; j<len; j++){
28            if( tp[sa[j]][0] == tp[sa[j-1]][0] &&
29               tp[sa[j]][1] == tp[sa[j-1]][1] )
30                rk[sa[j]] = rk[sa[j-1]];
31            else
32                rk[sa[j]] = j;
33        }
34    }
35    for(int i=0, h=0; i<len; i++){
36        if(rk[i]==0) h=0;
37        else{
38            int j=sa[rk[i]-1];
39            h=max(0, h-1);
40            for(; ip[i+h]==ip[j+h]; h++);
41        }
42        he[rk[i]]=h;
43    }
44 }

```

## 7.3 MinRoation \*

```

1 //rotate(begin(s),begin(s)+minRotation(s),end(s))
2 int minRotation(string s) {
3     int a = 0, N = s.size(); s += s;
4     rep(b,0,N) rep(k,0,N) {
5         if(a+k == b || s[a+k] < s[b+k])
6             {b += max(0, k-1); break;}
7         if(s[a+k] > s[b+k]) {a = b; break;}
8     } return a;

```

9|}

## 7.4 RollingHash

```
1 struct RollingHash {
2     const int p1 = 44129; // 65537, 40961, 90001, 971651
3     vector<ll> pre;
4     void init(string s) {
5         pre.resize(s.size() + 1); pre[0] = 0;
6         for (int i = 0; i < (int)s.size(); i++)
7             pre[i + 1] = (pre[i] * p1 + s[i]) % MOD;
8     }
9     ll query(int l, int r) {return (pre[r + 1] - pre[l] *
10         fpow(p1, r - l + 1));}
```

## 7.5 KMP

在 k 結尾的情況下，這個子字串可以由開頭長度為 (k + 1) - (fail[k] + 1) 的部分重複出現來表達  
fail[k] + 1 為次長相同前綴後綴長度  
如果我們不只想求最多，那可能的長度由大到小會是 fail[k]+1, fail[fail[k]+1], fail[fail[fail[k]]]+1...  
直到有值為 -1 為止

```
1 const int MXN = 2e7 + 5;
2 int fail[MXN]; vector<int> mi;
3 void kmp(string &t, string &p){ // O(n), 0-base
4     // pattern match in target, idx store in mi
5     mi.clear();
6     if (p.size() > t.size()) return;
7     for (int i = 1, j = fail[0] = -1; i < p.size(); ++i){
8         while (j >= 0 && p[j + 1] != p[i]) j = fail[j];
9         if (p[j + 1] == p[i]) j++;
10        fail[i] = j; }
11    for (int i = 0, j = -1; i < t.size(); ++i){
12        while (j >= 0 && p[j + 1] != t[i]) j = fail[j];
13        if (p[j + 1] == t[i]) j++;
14        if (j == p.size() - 1)
15            j = fail[j], mi.pb(i - p.size() + 1); }
```

## 7.6 LCS & LIS

LIS: 最長遞增子序列  
LCS: 最長共同子字串 (利用 LIS), 但常數可能較大

```
1 int lis(vector<ll> &v){ // O(nlgn)
2     vector<ll> p;
3     for(int i = 0; i < v.size(); ++i)
4         if(p.empty() || p.back() < v[i]) p.pb(v[i]);
5         else *lower_bound(p.begin(), p.end(), v[i]) = v[i];
6     return p.size(); }
7
8 int lcs(string s, string t){ // O(nlgn)
9     map<char, vector<int>> > mp;
10    for(int i = 0; i < s.size(); ++i) mp[s[i]].pb(i);
11    vector<int> p;
12    for(int i = 0; i < t.size(); ++i){
13        auto &v = mp[t[i]];
14        for(int j = v.size() - 1; j >= 0; --j)
15            if(p.empty() || p.back() < v[j]) p.pb(v[j]);
16            else *lower_bound(p.begin(), p.end(), v[j]) = v[j];
17    return p.size(); }
```

## 7.7 Aho-Corasick \*

```
1 struct ACautomata{
2     struct Node{
3         int cnt,i;
4         Node *go[26], *fail, *dic;
5         Node (){
6             cnt = 0; fail = 0; dic = 0; i = 0;
7             memset(go,0,sizeof(go));
8         }
9     }pool[1048576],*root;
10    int nMem,n_pattern;
11    Node* new_Node(){
12        pool[nMem] = Node();
13        return &pool[nMem++];
14    }
15    void init() {
16        nMem=0;root=new_Node();n_pattern=0;
17        add("");
18    }
19    void add(const string &str) { insert(root,str,0); }
```

```
20 void insert(Node *cur, const string &str, int pos){
21     for(int i=pos;i<str.size();i++){
22         if(!cur->go[str[i]-'a'])
23             cur->go[str[i]-'a'] = new_Node();
24         cur=cur->go[str[i]-'a'];
25     }
26     cur->cnt++; cur->i=n_pattern++;
27 }
28 void make_fail(){
29     queue<Node*> que;
30     que.push(root);
31     while (!que.empty()){
32         Node* fr=que.front(); que.pop();
33         for (int i=0; i<26; i++){
34             if (fr->go[i]){
35                 Node *ptr = fr->fail;
36                 while (ptr && !ptr->go[i]) ptr = ptr->fail;
37                 fr->go[i]->fail=ptr=(ptr?ptr->go[i]:root);
38                 fr->go[i]->dic=(ptr->cnt?ptr:ptr->dic);
39                 que.push(fr->go[i]);
40             } } }
41     void query(string s){
42         Node *cur=root;
43         for(int i=0;i<(int)s.size();i++){
44             while(cur&&!cur->go[s[i]-'a']) cur=cur->fail;
45             cur=(cur?cur->go[s[i]-'a']:root);
46             if(cur->i>=0) ans[cur->i]++;
47             for(Node *tmp=cur->dic;tmp;tmp=tmp->dic)
48                 ans[tmp->i]++;
49         } // ans[i] : number of occurrence of pattern i
50     }AC;
```

## 7.8 Z Value \*

```
1 int z[MAXN];
2 void Z_value(const string& s) { //z[i] = lcp(s[1...],s[
3     i...])
4     int i, j, left, right, len = s.size();
5     left=right=0; z[0]=len;
6     for(i=1;i<len;i++) {
7         j=max(min(z[i-left],right-i),0);
8         for(;i+j<len&&s[i+j]==s[j];j++);
9         z[i]=j;
10        if(i+z[i]>right) {
11            right=i+z[i];
12        } }
```

## 7.9 manacher \*

```
1 struct Manacher {
2     char str[MXN]; int p[MXN], len = 0;
3     void init(string s) {
4         MEM(p, 0);
5         str[len++] = '$', str[len++] = '#';
6         int sz = s.size();
7         for(int i = 0; i < sz; ++i)
8             str[len++] = s[i], str[len++] = '#';
9         str[len] = '*';
10        int mx = 0, id = 0;
11        for(int i = 1; i < len; ++i) {
12            p[i] = mx > i ? min(p[id<1] - i, mx - i) : 1;
13            while(str[i + p[i]] == str[i - p[i]]) p[i]++;
14            if(i + p[i] > mx) {
15                mx = i + p[i];
16                id = i; } }
17    int query(int l, int r) {
18        int ans = 0;
19        l = 2 * l + 2, r = 2 * r + 2;
20        for(int i = l; i < r; i++)
21            ans = max(ans, p[i]);
22        return ans - 1; } }
```

## 8 Data Structure

### 8.1 Treap

Treap \*th = 0  
th = merge(th, new Treap(val)) ⇒ 新增元素到 th  
th = merge(merge(tl, tm), tr) ⇒ 合併 tl,tm,tr 到 th  
split(th, k, tl, tr) ⇒ 分割 th, tl 的元素 ≤ k (失去 BST 性質後不能用)  
kth(th, k, tl, tr) ⇒ 分割 th, gsz(tl) ≤ k (< when gsz(th) < k)  
gsz ⇒ get size | gsum ⇒ get sum | th->rev ^= 1 ⇒ 反轉 th

帶懶標版本，並示範 sum/rev 如何 pull/push  
注意 Treap 複雜度好但常數大，動作能用其他方法就用，並做 io 等優化

```

1 struct Treap{
2     Treap *l, *r;
3     int pri, sz, rev;
4     ll val, sum;
5     Treap(int _val): l(0), r(0),
6         pri(rand()), sz(1), rev(0),
7         val(_val), sum(_val){} };
8
9 ll gsz(Treap *x){ return x ? x->sz : 0; }
10 ll gsum(Treap *x){ return x ? x->sum : 0; }
11
12 Treap* pull(Treap *x){
13     x->sz = gsz(x->l) + gsz(x->r) + 1;
14     x->sum = x->val + gsum(x->l) + gsum(x->r);
15     return x; }
16 void push(Treap *x){
17     if(x->rev){
18         swap(x->l, x->r);
19         if(x->l) x->l->rev ^= 1;
20         if(x->r) x->r->rev ^= 1;
21         x->rev = 0; } }
22
23 Treap* merge(Treap* a, Treap* b){
24     if(!a || !b) return a ? a : b;
25     push(a), push(b);
26     if(a->pri > b->pri){
27         a->r = merge(a->r, b);
28         return pull(a); }
29     else{
30         b->l = merge(a, b->l);
31         return pull(b); } }
32
33 void split(Treap *x, int k, Treap *&a, Treap *&b){
34     if(!x) a = b = 0;
35     else{
36         push(x);
37         if(x->val <= k) a = x, split(x->r, k, a->r, b);
38         else b = x, split(x->l, k, a, b->l);
39         pull(x); } }
40
41 void kth(Treap *x, int k, Treap *&a, Treap *&b){
42     if(!x) a = b = 0;
43     else{
44         push(x);
45         if(gsz(x->l) < k)
46             a = x, kth(x->r, k - gsz(x->l) - 1, a->r, b);
47         else b = x, kth(x->l, k, a, b->l);
48         pull(x); } }

```

## 8.2 BIT

bit.init(n)  $\Rightarrow$  1-base  
bit.add(i, x)  $\Rightarrow$  add a[i] by x  
bit.sum(i)  $\Rightarrow$  get sum of [1, i]  
bit.kth(k)  $\Rightarrow$  get kth small number (by using bit.add(num, 1))  
維護差分可以變成區間加值，單點求值

```

1 const int MXN = 1e6+5;
2 struct BIT{
3     ll n, a[MXN];
4     void init(int _n){ n = _n; MEM(a, 0); }
5     void add(int i, int x){
6         for(; i <= n; i += i & -i) a[i] += x; }
7     int sum(int i){
8         int ret = 0;
9         for(; i > 0; i -= i & -i) ret += a[i];
10        return ret; }
11    int kth(int k){
12        int res = 0;
13        for(int i = 1 << __lg(n); i > 0; i >>= 1)
14            if(res + i <= n && a[res+i] < k) k -= a[res+i];
15        return res; } }

```

## 8.3 二維偏序 \*

```

1 struct Node {
2     int x, y, id;
3     bool operator < (const Node &b) const {
4         if(x == b.x) return y < b.y;
5         return x < b.x; } }
6 struct TDPO {

```

```

7     vector<Node> p; vector<ll> ans;
8     void init(vector<Node> _p) {
9         p = _p; bit.init(MXN);
10        ans.resize(p.size());
11        sort(p.begin(), p.end());
12    void bulid() {
13        int sz = p.size();
14        for(int i = 0; i < sz; ++i) {
15            ans[p[i].id] = bit.sum(p[i].y - 1);
16            bit.add(p[i].y, 1); } }

```

## 8.4 持久化 \*

```

1 struct Seg {
2     // Persistent Segment Tree, single point modify,
3     // range query sum
4     // 0-indexed, [l, r)
5     static Seg mem[M], *pt;
6     int l, r, m, val;
7     Seg* ch[2];
8     Seg () = default;
9     Seg (int _l, int _r) : l(_l), r(_r), m(l + r >> 1),
10        val(0) {
11         if (r - l > 1) {
12             ch[0] = new (pt++) Seg(l, m);
13             ch[1] = new (pt++) Seg(m, r);
14         }
15     }
16     void pull() {val = ch[0]->val + ch[1]->val;}
17     Seg* modify(int p, int v) {
18         Seg *now = new (pt++) Seg(*this);
19         if (r - l == 1) {
20             now->val = v;
21         } else {
22             now->ch[p >= m] = ch[p >= m]->modify(p, v);
23             now->pull();
24         }
25         return now;
26     }
27     int query(int a, int b) {
28         if (a <= l && r <= b) return val;
29         int ans = 0;
30         if (a < m) ans += ch[0]->query(a, b);
31         if (m < b) ans += ch[1]->query(a, b);
32         return ans;
33     }
34     Seg::mem[M], *Seg::pt = mem;
35     // Init Tree
36     Seg *root = new (Seg::pt++) Seg(0, n);

```

## 8.5 2D 線段樹

```

1 // 2D range add, range sum in log^2
2 struct seg {
3     int l, r;
4     ll sum, lz;
5     seg *ch[2];
6     seg(int _l, int _r) : l(_l), r(_r), sum(0), lz(0) {}
7     void push() {
8         if (lz) ch[0]->add(l, r, lz), ch[1]->modify(l, r,
9             lz), lz = 0;
10    }
11    void pull() {sum = ch[0]->sum + ch[1]->sum;}
12    void add(int _l, int _r, ll d) {
13        if (_l <= l && r <= _r) {
14            sum += d * (r - l);
15            lz += d;
16            return;
17        }
18        if (!ch[0]) ch[0] = new seg(l, l + r >> 1), ch[1] =
19            new seg(l + r >> 1, r);
20        push();
21        if (_l < l + r >> 1) ch[0]->add(_l, _r, d);
22        if (l + r >> 1 < _r) ch[1]->add(_l, _r, d);
23        pull();
24    }
25    ll qsum(int _l, int _r) {
26        if (_l <= l && r <= _r) return sum;
27        if (!ch[0]) return lz * (min(r, _r) - max(l, _l));
28        push();
29        ll res = 0;
30        if (_l < l + r >> 1) res += ch[0]->qsum(_l, _r);

```

```

29     if (l + r >> 1 < _r) res += ch[1]->qsum(_l, _r);
30     return res;
31 }
32 };
33 struct seg2 {
34     int l, r;
35     seg v, lz;
36     seg2 *ch[2]{};
37     seg2(int _l, int _r) : l(_l), r(_r), v(0, N), lz(0, N) {
38         if (l < r - 1) ch[0] = new seg2(l, l + r >> 1), ch[1] = new seg2(l + r >> 1, r);
39     }
40     void add(int _l, int _r, int _l2, int _r2, ll d) {
41         v.add(_l2, _r2, d * (min(r, _r) - max(l, _l2)));
42         if (_l <= l && r <= _r) {
43             lz.add(_l2, _r2, d);
44             return;
45         }
46         if (_l < l + r >> 1) ch[0]->add(_l, _r, _l2, _r2, d);
47         if (l + r >> 1 < _r) ch[1]->add(_l, _r, _l2, _r2, d);
48     }
49     ll qsum(int _l, int _r, int _l2, int _r2) {
50         ll res = v.qsum(_l2, _r2);
51         if (_l <= l && r <= _r) return res;
52         res += lz.qsum(_l2, _r2) * (min(r, _r) - max(l, _l2));
53         if (_l < l + r >> 1) res += ch[0]->query(_l, _r, _l2, _r2);
54         if (l + r >> 1 < _r) res += ch[1]->query(_l, _r, _l2, _r2);
55         return res;
56     }
57 };

```

## 8.6 Disjoint Set

```

1 struct DisjointSet {
2     int fa[MXN], h[MXN], top;
3     struct Node {
4         int x, y, fa, h;
5         Node(int _x = 0, int _y = 0, int _fa = 0, int _h = 0) : x(_x), y(_y), fa(_fa), h(_h) {}
6     } stk[MXN];
7     void init(int n) {
8         top = 0;
9         for (int i = 1; i <= n; i++) fa[i] = i, h[i] = 0; }
10    int find(int x){return x == fa[x] ? x : find(fa[x]);}
11    void merge(int u, int v) {
12        int x = find(u), y = find(v);
13        if (h[x] > h[y]) swap(x, y);
14        stk[top++] = Node(x, y, fa[x], h[y]);
15        if (h[x] == h[y]) h[y]++;
16        fa[x] = y; }
17    void undo(int k=1) { //undo k times
18        for (int i = 0; i < k; i++) {
19            Node &it = stk[--top];
20            fa[it.x] = it.fa;
21            h[it.y] = it.h; } } djs;

```

## 8.7 Black Magic

```

1 #include <bits/extc++.h>
2 using namespace __gnu_pbds;
3 typedef tree<int, null_type, less<int>, rb_tree_tag,
4     tree_order_statistics_node_update> set_t;
5 #include <ext/pb_ds/assoc_container.hpp>
6 typedef cc_hash_table<int, int> umap_t;
7 typedef priority_queue<int> heap;
8 #include <ext/rope>
9 using namespace __gnu_cxx;
10 int main(){
11     // Insert some entries into s.
12     set_t s; s.insert(12); s.insert(505);
13     // The order of the keys should be: 12, 505.
14     assert(*s.find_by_order(0) == 12);
15     assert(*s.find_by_order(3) == 505);
16     // The order of the keys should be: 12, 505.
17     assert(s.order_of_key(12) == 0);
18     assert(s.order_of_key(505) == 1);

```

```

18 // Erase an entry.
19 s.erase(12);
20 // The order of the keys should be: 505.
21 assert(*s.find_by_order(0) == 505);
22 // The order of the keys should be: 505.
23 assert(s.order_of_key(505) == 0);
24
25 heap h1, h2; h1.join(h2);
26
27 rope<char> r[2];
28 r[1] = r[0]; // persistenet
29 string t = "abc";
30 r[1].insert(0, t.c_str());
31 r[1].erase(1, 1);
32 cout << r[1].substr(0, 2);
33 }

```

## 9 Others

### 9.1 SOS dp \*

```

1 for(int i = 0; i < (1<<N); ++i)
2     F[i] = A[i];
3 for(int i = 0; i < N; ++i) for(int mask = 0; mask < (1<<N); ++mask){
4     if(mask & (1<<i))
5         F[mask] += F[mask^(1<<i)];
6 }

```

### 9.2 MO's Algorithm \*

```

1 struct MoSolver {
2     struct query {
3         int l, r, id;
4         bool operator < (const query &o) {
5             if (l / C == o.l / C) return (l / C) & 1 ? r > o.r : r < o.r;
6             return l / C < o.l / C;
7         }
8     };
9     int cur_ans;
10    vector<int> ans;
11    void add(int x) {
12        // do something
13    }
14    void sub(int x) {
15        // do something
16    }
17    vector<query> Q;
18    void add_query(int l, int r, int id) {
19        // [l, r]
20        Q.push_back({l, r, id});
21        ans.push_back(0);
22    }
23    void run() {
24        sort(Q.begin(), Q.end());
25        int pl = 0, pr = 0;
26        cur_ans = 0;
27        for (query &i : Q) {
28            while (pl > i.l) add(a[--pl]);
29            while (pr < i.r) add(a[pr++]);
30            while (pl < i.l) sub(a[pl--]);
31            while (pr > i.r) sub(a[pr--]);
32            ans[i.id] = cur;
33        }
34    }
35 };

```





