

Contents

1 Basic

- 1.1 .vimrc
- 1.2 Default Code
- 1.3 Common Sense

2 flow

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

3 Math

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

4 Geometry

- 4.1 definition *
- 4.2 halfPlaneIntersection *
- 4.3 Convex Hull *
- 4.4 Li Chao Segment Tree *
- 4.5 Convex Hull trick *
- 4.6 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 BCC based on vertex *
- 6.7 Strongly Connected Component *
- 6.8 差分約束 *

7 String

- 7.1 PalTree *
- 7.2 SuffixArray *
- 7.3 MinRoation *
- 7.4 Aho-Corasick *
- 7.5 Z Value *
- 7.6 ZValue Palindrome *

8 Data Structure

- 8.1 Treap
- 8.2 BIT

9 Others

- 9.1 SOS dp

1 Basic

1.1 .vimrc

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

```
1 se nu rnu mouse=a cin et sw=2 sts=2 nowrap ic nosp bs=238
2 syn on
3 au InsertLeave * se rnu
4 au InsertEnter * se nornu
```

1.2 Default Code

所有模板的 define 都在這

```
1 #include<bits/stdc++.h>
2 #define ll long long
3 #define ld long double
4 #define INF 0x3f3f3f3f
5 #define LLINF 0x3f3f3f3f3f3f3f3f
6 #define NINF 0xc1c1c1c1
7 #define NLLINF 0xc1c1c1c1c1c1c1c1
8 #define X first
9 #define Y second
10 #define PB emplace_back
11 #define pll pair<long, long>
```

```
12 #define MEM(a,n) memset(a, n, sizeof(a))
13 using namespace std;
14 const int MXN = 4e5+5;
15
16 void sol(){
17 int main(){
18     int t=1;
19     cin >> t;
20     while(t--){
21         sol(); } }
2
```

1.3 Common Sense

陣列過大時本機的指令：

windows: g++ -Wl,-stack,40000000 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

2 flow

2.1 MinCostFlow *

```
61 struct zkwflow{
62     static const int MXN = 10000;
63     struct Edge{ int v, f, re; ll w;};
64     int n, s, t, ptr[MXN]; bool vis[MXN]; ll dis[MXN];
65     vector<Edge> E[MXN];
66     void init(int _n,int _s,int _t){
67         n=_n,s=_s,t=_t;
68         for(int i=0;i<n;i++) E[i].clear();
69     }
70     void addEdge(int u, int v, int f, ll w){
71         E[u].emplace_back(v, f, (int)E[v].size(), w);
72         E[v].emplace_back(u, 0, (int)E[u].size()-1, -w);
73     }
74     bool SPFA(){
75         fill_n(dis, n,LLMXN); memset(vis, 0, 4 * n);
76         queue<int> q; q.push(s); dis[s] = 0;
77         while (!q.empty()){
78             int u = q.front(); q.pop(); vis[u] = false;
79             for(auto &it : E[u]){
80                 if(it.f > 0 && dis[it.v] > dis[u] + it.w){
81                     dis[it.v] = dis[u] + it.w;
82                     if(!vis[it.v]){
83                         vis[it.v] = 1; q.push(it.v);
84                     } } } }
85         return dis[t] != LLMXN;
86     }
87     int DFS(int u, int nf){
88         if(u == t) return nf;
89         int res = 0; vis[u] = 1;
90         for(int &i = ptr[u]; i < (int)E[u].size(); ++i){
91             auto &it = E[u][i];
92             if(it.f>0&&dis[it.v]==dis[u]+it.w&&!vis[it.v]){
93                 int tf = DFS(it.v, min(nf,it.f));
94                 res += tf, nf -= tf, it.f -= tf;
95                 E[it.v][it.re].f += tf;
96                 if(nf == 0){ vis[u] = false; break; }
97             }
98         }
99         return res;
100     }
101     pair<int,ll> flow(){
102         int flow = 0; ll cost=0;
103         while (SPFA()){
104             memset(ptr, 0, 4 * n);
105             int f = DFS(s, INF);
106             flow += f; cost += dis[t] * f;
107         }
108         return { flow, cost };
109     }
110 } 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) ⇒ 0-base
dinic.addEdge(u, v, f) ⇒ $u \rightarrow v$, flow f units
dinic.run() ⇒ 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        ll dfs(int u, ll nf){
26            if(u == t) return nf;
27            ll res = 0;
28            for(auto &i : e[u])
29                if(i.f > 0 && lvl[i.v] == lvl[u] + 1){
30                    int tmp = dfs(i.v, min(nf, i.f));
31                    res += tmp, nf -= tmp, i.f -= tmp;
32                    e[i.v][i.re].f += tmp;
33                    if(nf == 0) return res;
34                }
35            if(!res) lvl[u] = -1;
36            return res;
37        }
38        ll run(ll res){
39            while(bfs()) res += dfs(s, LLINF);
40            return res;
41        }
42    };
43 }
```

2.3 Hungarian

2.4 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：

最大權完美匹配 (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     }
10    void addEdge(int x, int y, ll w){ g[x][y] = w; }
11    void augment(int y){
12        for(int x, z; y; y = z)
13            x = pa[y], z = mx[x], my[y] = x, mx[x] = y;
14    }
15    void bfs(int st){
16        for(int i = 1; i <= n; ++i) sy[i] = LLINF, vx[i] = 0;
17        queue<int> q; q.push(st);
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                    }
29                    else if(sy[y] > t) pa[y] = x, sy[y] = t;
30                }
31        }
32    }
```

```
ll cut = LLINF;
for(int y = 1; y <= n; ++y)
    if(!vy[y] && cut > sy[y]) cut = sy[y];
for(int j = 1; j <= n; ++j){
    if(vx[j]) lx[j] -= cut;
    if(vy[j]) ly[j] += cut;
    else sy[j] -= cut;
}
for(int y = 1; y <= n; ++y)
    if(!vy[y] && sy[y] == 0){
        if(!my[y]){ augment(y); return; }
        vy[y] = 1, q.push(my[y]);
    }
}
ll run(){
    MEM(mx, 0), MEM(my, 0), MEM(ly, 0), MEM(lx, -0x3f);
    for(int x = 1; x <= n; ++x) for(int y = 1; y <= n; ++y)
        lx[x] = max(lx[x], g[x][y]);
    for(int x = 1; x <= n; ++x) bfs(x);
    ll ret = 0;
    for(int y = 1; y <= n; ++y) ret += g[my[y]][y];
    return ret;
}
```

2.5 Directed MST *

```
1 /* Edmond's algoirthm for Directed MST
2  * runs in  $O(VE)$  */
3 const int MAXV = 10010;
4 const int MAXE = 10010;
5 const int INF = 2147483647;
6 struct Edge{
7     int u, v, c;
8     Edge(int x=0, int y=0, int z=0) : u(x), v(y), c(z){}
9 };
10 int V, E, root;
11 Edge edges[MAXE];
12 inline int newV(){ return ++V; }
13 inline void addEdge(int u, int v, int c)
14 { edges[++E] = Edge(u, v, c); }
15 bool con[MAXV];
16 int mnInW[MAXV], prv[MAXV], cyc[MAXV], vis[MAXV];
17 inline int DMST(){
18     fill(con, con+V+1, 0);
19     int r1 = 0, r2 = 0;
20     while(1){
21         fill(mnInW, mnInW+V+1, INF);
22         fill(prv, prv+V+1, -1);
23         REP(i, 1, E){
24             int u=edges[i].u, v=edges[i].v, c=edges[i].c;
25             if(u != v && v != root && c < mnInW[v])
26                 mnInW[v] = c, prv[v] = u;
27         }
28         fill(vis, vis+V+1, -1);
29         fill(cyc, cyc+V+1, -1);
30         r1 = 0;
31         bool jf = 0;
32         REP(i, 1, V){
33             if(con[i]) continue;
34             if(prv[i] == -1 && i != root) return -1;
35             if(prv[i] > 0) r1 += mnInW[i];
36             int s;
37             for(s = i; s != -1 && vis[s] == -1; s = prv[s])
38                 vis[s] = i;
39             if(s > 0 && vis[s] == i){
40                 // get a cycle
41                 jf = 1; int v = s;
42                 do{
43                     cyc[v] = s, con[v] = 1;
44                     r2 += mnInW[v]; v = prv[v];
45                 }while(v != s);
46                 con[s] = 0;
47             }
48         }
49         REP(i, 1, E){
50             int &u = edges[i].u;
51             int &v = edges[i].v;
52             if(cyc[v] > 0) edges[i].c -= mnInW[edges[i].v];
53             if(cyc[u] > 0) edges[i].u = cyc[edges[i].u];
54             if(cyc[v] > 0) edges[i].v = cyc[edges[i].v];
55             if(u == v) edges[i--] = edges[E--];
56         }
57         return r1+r2;
58     }
59 }
```

2.6 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            }
21            if (mx == -1) break;
22            vst[cur] = 1;
23            s = t; t = cur;
24            for (int i=0; i<n; i++){
25                if (!vst[i] && !del[i]) wei[i] += edge[cur][i];
26            }
27        }
28        int solve(){
29            int res = 2147483647;
30            for (int i=0,x,y; i<n-1; i++){
31                search(x,y);
32                res = min(res,wei[y]);
33                del[y] = 1;
34                for (int j=0; j<n; j++){
35                    edge[x][j] = (edge[j][x] += edge[y][j]);
36                }
37            }
38            return res;
39        }
40    }graph;

```

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 = $S + 2 * D - \text{deg}[v] - 2 * (W \text{ of } v)$
 where $\text{deg}[v] = \sum \text{weight of edge associated with } v$
 If $\text{maxflow} < S * |V|$, D is an answer.
 Requiring subgraph: all vertex can be reached from source with
 edge whose cap > 0 .

3 Math

3.1 Fast Pow & Inverse & Combination

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

```

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 - 1] = fi[i] * i % MOD; }
18
19 ll c(ll a, ll b){
20     return fa[a] * fi[b] % MOD * fi[a - b] % MOD; }

```

3.2 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); }

```

3.3 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) : i*theta%MAXN];
21                 for (int j = i; j < n; j += m) {
22                     int k = j + mh;
23                     cplx x = a[j] - a[k];
24                     a[j] += a[k];
25                     a[k] = w * x;
26                 }
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=1, sum=_n+_m-1;
41         while(n<sum)
42             n<<=1;
43         for(int i=0; i<n; i++){
44             double x=(i<_n?a[i]:0), y=(i<_m?b[i]:0);
45             arr[i]=complex<double>(x+y, x-y);
46         }
47         fft(n, arr);
48         for(int i=0; i<n; i++){
49             arr[i]=arr[i]*arr[i];
50             fft(n, arr, true);
51             for(int i=0; i<sum; i++){
52                 ans[i]=(long long int)(arr[i].real()/4+0.5);
53             }
54         }
55     }

```

3.4 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;

```

3.5 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 }

```

3.6 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-->0) {
14         ll nx = x * x % n;
15         if(nx == 1 && x != 1 && x != n - 1) return 1;
16         x = nx; }
17     return x != 1; }
18 bool isprime(ll n) {
19     if(n < 2) return 0;
20     if(~n & 1) return n == 2;
21     ll u = n - 1, t = 0;
22     while(~u & 1) u >>= 1, t++;
23     for(auto i : magic){
24         ll a = i % n;
25         if(witness(a, n, u, t)) return 0; }
26     return 1; }

```

3.7 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]; // coefficient 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],
35                 mul(cm[i][j],mul(b[j], inv[i-j+1])));
36     }
37     /* faulhaber */
38     // sigma_x=1~n {x^p} =
39     // 1/(p+1) * sigma_j=0~p {C(p+1,j)*Bj*n^(p-j+1)}
40     for(int i=1;i<=MAXK;i++) {
41         co[i][0]=0;

```

```

42     for(int j=0;j<=i;j++){
43         co[i][i-j+1]=mul(inv[i+1], mul(cm[i+1][j], b[j]));
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 }

```

3.8 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[1])*m[1];
13     for(int i=2;i<n;i++){
14         res=CRT(res,p,x[i],m[i]);
15         p=p/__gcd(p,m[i])*m[i];
16     }
17     return res;
18 }

```

3.9 Pollard Rho *

```

1 // does not work when n is prime 0(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    } }

```

3.10 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 }

```

3.11 Gaussian Elimination *

```

1 const int GAUSS_MOD = 100000007LL;
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][now] != 0)

```

```

        swap(v[i] , v[now]); // det = -det;
        if(v[now][now] == 0) return ans;
        int inv = ppow(v[now][now] , GAUSS_MOD - 2)
        ;
        REP(i, 0, n) if(i != now){
            int tmp = v[i][now] * inv % GAUSS_MOD;
            REP(j, now, n + 1) (v[i][j] +=
                GAUSS_MOD - tmp * v[now][j] %
                GAUSS_MOD) %= GAUSS_MOD;
        }
        REP(i, 0, n) ans[i] = v[i][n + 1] * ppow(v[i][n + 1], GAUSS_MOD - 2) % GAUSS_MOD;
        return ans;
    }
} // gs.v.clear() , gs.v.resize(n , vector<int>(n + 1 , 0));
} gs;

```

3.12 Result *

- Lucas' Theorem :
For $n, m \in \mathbb{Z}^+$ and prime P , $C(m, n) \bmod P = \prod(C(m_i, n_i))$ 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 : $C_n = \binom{2n}{n} / (n+1)$
$$C_n^{n+m} - C_{n+1}^{n+m} = (m+n)! \frac{n-m+1}{n+1} \quad \text{for } n \geq m$$

$$C_n = \frac{1}{n+1} \binom{2n}{n} = \frac{(2n)!}{(n+1)n!}$$

$$C_0 = 1 \quad \text{and} \quad C_{n+1} = 2 \binom{2n+1}{n+2} C_n$$

$$C_0 = 1 \quad \text{and} \quad C_{n+1} = \sum_{i=0}^n C_i C_{n-i} \quad \text{for } n \geq 0$$
- 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, one column, and cal the $\det(A)$
- Polya' theorem (c 為方法數, m 為總數):
$$\left(\sum_{i=1}^m c^{gcd(i,m)}\right)/m$$
- 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) :
$$B_0 = 1$$

$$B_n = \sum_{k=0}^n s(n, k) \quad (\text{second - stirling})$$

$$B_{n+1} = \sum_{k=0}^n \binom{n}{k} B_k$$
- Wilson's theorem :
$$(p-1)! \equiv -1 \pmod{p}$$
- Fermat's little theorem :
$$a^p \equiv a \pmod{p}$$
- Euler's totient function:
$$A^{BC} \bmod p = \text{pow}(A, \text{pow}(B, C, p-1)) \bmod p$$
- 歐拉函數降幂公式:
$$A^B \bmod C = A^{B \bmod \phi(C) + \phi(C)} \bmod C$$
- 6 的倍數:
$$(a-1)^3 + (a+1)^3 + (-a)^3 + (-a)^3 = 6a$$

4 Geometry

4.1 definition *

```

1 typedef long double ld;
2 const ld eps = 1e-8;
3 int dcmp(ld x) {
4     if(abs(x) < eps) return 0;
5     else return x < 0 ? -1 : 1;
6 }
7 struct Pt {
8     ld x, y;
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 {
19        return x*a.x + y*a.y; }
20    ld operator^(const Pt &a) const {
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 || (dcmp(x-a.x) == 0 &&
25        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 }
29 ld norm2(const Pt &a) {
30     return a*a; }
31 ld norm(const Pt &a) {
32     return sqrt(norm2(a)); }
33 Pt perp(const Pt &a) {
34     return Pt(-a.y, a.x); }
35 Pt rotate(const Pt &a, ld ang) {
36     return Pt(a.x*cos(ang)-a.y*sin(ang), a.x*sin(ang)+a.y
37         *cos(ang)); }
38 struct Line {
39     Pt s, e, v; // start, end, end-start
40     ld ang;
41     Line(Pt _s=Pt(0, 0), Pt _e=Pt(0, 0)):s(_s), e(_e) {
42         = e-s; ang = atan2(v.y, v.x); }
43     bool operator<(const Line &l) const {
44         return ang < l.ang; }
45 }
46 struct Circle {
47     Pt o; ld r;
48     Circle(Pt _o=Pt(0, 0), ld _r=0):o(_o), r(_r) {}
49 }

```

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 );

```

```

25 }
26 inline bool cmp( const LI &a, const LI &b ){
27     if( dc( a.angle - b.angle ) == 0 ) return dc( cross(
28         a.a, a.b, b.a ) ) < 0;
29     return a.angle > b.angle;
30 }
31 inline PO getpoint( LI &a, LI &b ){
32     double k1 = cross( a.a, b.b, b.a );
33     double k2 = cross( a.b, b.a, b.b );
34     PO tmp = a.b - a.a, ans;
35     ans.x = a.a.x + tmp.x * k1 / ( k1 + k2 );
36     ans.y = a.a.y + tmp.y * k1 / ( k1 + k2 );
37     return ans;
38 }
39 inline void getcut(){
40     sort( li + 1, li + 1 + n, cmp ); m = 1;
41     for( int i = 2; i <= n; i++ )
42         if( dc( li[ i ].angle - li[ m ].angle ) != 0 )
43             li[ ++m ] = li[ i ];
44     deq[ 1 ] = li[ 1 ]; deq[ 2 ] = li[ 2 ];
45     int bot = 1, top = 2;
46     for( int i = 3; i <= m; i++ ){
47         while( bot < top && dc( cross( li[ i ].a, li[ i ].
48             b, getpoint( deq[ top ], deq[ top - 1 ] ) ) )
49             < 0 ) top --;
50         while( bot < top && dc( cross( li[ i ].a, li[ i ].
51             b, getpoint( deq[ bot ], deq[ bot + 1 ] ) ) )
52             < 0 ) bot ++;
53         deq[ ++top ] = li[ i ];
54     }
55     while( bot < top && dc( cross( deq[ bot ].a, deq[
56         bot ].b, getpoint( deq[ top ], deq[ top - 1 ] )
57         ) ) < 0 ) top --;
58     while( bot < top && dc( cross( deq[ top ].a, deq[
59         top ].b, getpoint( deq[ bot ], deq[ bot + 1 ] )
60         ) ) < 0 ) bot ++;
61     cnt = 0;
62     if( bot == top ) return;
63     for( int i = bot; i < top; i++ ) p[ ++cnt ] =
64         getpoint( deq[ i ], deq[ i + 1 ] );
65     if( top - 1 > bot ) p[ ++cnt ] = getpoint( deq[ bot
66         ], deq[ top ] );
67 }
68 double px[ N ], py[ N ];
69 void read( int rm ) {
70     for( int i = 1; i <= n; i++ ) px[ i + n ] = px[ i
71         ], py[ i + n ] = py[ i ];
72     for( int i = 1; i <= n; i++ ){
73         // half-plane from li[ i ].a -> li[ i ].b
74         li[ i ].a.x = px[ i + rm + 1 ]; li[ i ].a.y = py[ i
75             + rm + 1 ];
76         li[ i ].b.x = px[ i ]; li[ i ].b.y = py[ i ];
77         li[ i ].angle = atan2( li[ i ].b.y - li[ i ].a.y,
78             li[ i ].b.x - li[ i ].a.x );
79     }
80 }
81 inline double getarea( int rm ){
82     read( rm ); getcut();
83     double res = 0.0;
84     p[ cnt + 1 ] = p[ 1 ];
85     for( int i = 1; i <= cnt; i++ ) res += cross( o,
86         p[ i ], p[ i + 1 ] );
87     if( res < 0.0 ) res *= -1.0;
88     return res;
89 }

```

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
10             ]) <= 0)
11             top--;
12         stk[top++] = pt[i];
13     }
14     for (int i=pt.size()-2, t=top+1; i>=0; i--){

```

```

14 while (top >= t && cross(stk[top-2],stk[top-1],pt[i25
    ]) <= 0)
15     top--;
16     stk[top++] = pt[i];
17 }
18 stk.resize(top-1);
19 return stk;
20 }

```

4.4 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,q,r;
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 }

```

4.5 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){
11        n = a.size();
12        int ptr = 0;
13        for(int i=1; i<n; ++i) if (a[ptr] < a[i]) ptr = i;
14        for(int i=0; i<=ptr; ++i) lower.push_back(a[i]);
15        for(int i=ptr; i<n; ++i) upper.push_back(a[i]);
16        upper.push_back(a[0]);
17    }
18    int sign( LL x ){ // fixed when changed to double
19        return x < 0 ? -1 : x > 0; }
20    pair<LL,int> get_tang(vector<Pt> &conv, Pt vec){
21        int l = 0, r = (int)conv.size() - 2;
22        for( ; l + 1 < r; ){
23            int mid = (l + r) / 2;
24            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){
        pair<LL, int> ret = get_tang(upper, vec);
        ret.second = (ret.second+(int)lower.size()-1)%n;
        ret = max(ret, get_tang(lower, vec));
        return ret.second;
    }
    // 4. Find intersection point of a given line
    // return 1 and intersection is on edge (i, next(i))
    // return 0 if no strictly intersection
    bool get_intersection(Pt u, Pt v, int &i0, int &i1){
        int p0 = get_tang(u - v), p1 = get_tang(v - u);
        if(sign(det(v-u,a[p0]-u))*sign(det(v-u,a[p1]-u))<0){
            if (p0 > p1) swap(p0, p1);
            i0 = bi_search(u, v, p0, p1);
            i1 = bi_search(u, v, p1, p0 + n);

```

```

100     return 1;
101 }
102 return 0;
103 } };

```

4.6 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     }
70 } };

```

5 Tree

5.1 LCA

求樹上兩點的最低共同祖先

$lca.init(n) \Rightarrow 0\text{-base}$

$lca.addEdge(u, v) \Rightarrow u \leftrightarrow v$

$lca.build(root, root) \Rightarrow O(n \lg n)$

$lca.qlca(u, v) \Rightarrow O(\lg n)$ 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)) u = anc[u][i];
33            if(!isanc(anc[v][i], u)) v = anc[v][i];
34            dis += 1<<i;
35        }
36        return dis;
37    }
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    }
36 }

```



```

35 for(int i : e[x]){
36     if(i == fa[x] || i == h[x]) continue;
37     dfs2(i, i);
38 }
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 } } domT;

```

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)

```

```

16 { return val._Find_first(); }
17 int ans , stk[N];
18 int id[N] , di[N] , deg[N];
19 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             }
52             Int cand; cand.reset();
53             for(int i = 0 ; i < n ; i ++){ cand[i] = 1;
54             ans = 1;
55             cans.reset(); cans[0] = 1;
56             maxclique(0, cand);
57             return ans;
58         }
59     } }
60 } } 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            }
21            ans = elem_num; // cans is a maximal clique
22            return;
23        }
24        int pivot = (candilex)._Find_first();
25        Int smaller_candi = candi & (~lnk[pivot]);
26        while(smaller_candi.count()){
27            int nxt = smaller_candi._Find_first();
28            candi[nxt] = smaller_candi[nxt] = 0;
29            ex[nxt] = 1;
30            stk[elem_num] = nxt;
31            dfs(elem_num+1, candi&lnk[nxt], ex&lnk[nxt]);
32        } }
33    int solve(){
34        for(int i = 0 ; i < n ; i ++){
35            id[i] = i; deg[i] = v[i].count();
36        }
37        sort(id , id + n , [&](int id1, int id2){
38            return deg[id1] > deg[id2]; });
39        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;
44             dfs(0, Int(string(n, '1')), 0);
45         } } solver;

```

6.6 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            }
41            REP(i,nScc) res.PB(sccv[i]);
42            return res;
43        } }
44    } graph;

```

6.7 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     }
10    void addEdge(int u, int v){
11        E[u].PB(v); rE[v].PB(u);
12    }
13    void DFS(int u){
14        vst[u]=1;
15        for (auto v : E[u]) if (!vst[v]) DFS(v);
16        vec.PB(u);
17    }
18    void rDFS(int u){
19        vst[u] = 1; bln[u] = nScc;
20        for (auto v : rE[u]) if (!vst[v]) rDFS(v);
21    }
22    void solve(){
23        nScc = 0;
24        vec.clear();
25        FZ(vst);
26        for (int i=0; i<n; i++){
27            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 };

```

6.8 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?1-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[sa[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=sa[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++) tsa[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 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            }
42        }
43    }
44    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;

```

7.5 Z Value *

```

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

```

7.6 ZValue Palindrome *

```

1 void z_value_pal(char *s,int len,int *z){
2     len=(len<<1)+1;
3     for(int i=len-1;i>=0;i--)
4         s[i]=i&1?s[i>>1]:'@';
5     z[0]=1;
6     for(int i=1,l=0,r=0;i<len;i++){
7         z[i]=i<r?min(z[l+l-i],r-i):1;
8         while(i-z[i]>=0&&i+z[i]<len&&s[i-z[i]]==s[i+z[i]])
9             ++z[i];
10        if(i+z[i]>r) l=i,r=i+z[i];
11    } }

```

8 Data Structure

8.1 Treap

Treap *th = nullptr
th = merge(th, new Treap(val)) ⇒ 新增元素到 th
th = merge(merge(tl, tm), tr) ⇒ 合併 t1,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(nullptr), r(nullptr),
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 void split(Treap *x, int k, Treap *&a, Treap *&b){
33     if(!x) a = b = nullptr;
34     else{
35         push(x);
36         if(x->val <= k) a = x, split(x->r, k, a->r, b);
37         else b = x, split(x->l, k, a, b->l);
38         pull(x); } }
39
40 void kth(Treap *x, int k, Treap *&a, Treap *&b){
41     if(!x) a = b = nullptr;
42     else{
43         push(x);
44         if(gsz(x->l) < k)
45             a = x, kth(x->r, k - gsz(x->l) - 1, a->r, b);
46         else b = x, kth(x->l, k, a, b->l);
47         pull(x); } }
48

```

8.2 BIT

bit.init(n) ⇒ 1-base
bit.add(i, x) ⇒ add a[i] by x
bit.sum(i) ⇒ get sum of [1, i]
bit.kth(k) ⇒ 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     int n;
4     ll a[MXN];
5     void init(int _n){
6         n = _n;
7         MEM(a, 0); }
8     void add(int i, int x){
9         for(; i <= n; i += i & -i)
10             a[i] += x; }
11     int sum(int i){
12         int ret = 0;
13         for(; i > 0; i -= i & -i)
14             ret += a[i];
15         return ret; }
16     int kth(int k){
17         int res = 0;
18         for(int i = 1 << __lg(n); i > 0; i >>= 1)
19             if(res + i <= n && a[res + i] < k)
20                 k -= a[res += i];
21         return res; } };

```

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 }

```