Contents

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

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

1.2 Default Code

所有模板的 define 都在這

```cpp
#include<bits/stdc++.h>
#include <chrono>
using namespace std;

#ifdef LOCAL // ======== Local ======== g++ -DLOCAL ...
void dbg() { cerr << '\n'; }
template<class T, class ...U> void dbg(T a, U ...b) {
  cerr << a << ' ', dbg(b...); }
template<class T> void org(T l, T r) {
  while (l != r) cerr << *l++ << ' '; cerr << '\n'; }
#define DEBUG(args...) \
  (dbg("#> (" + string(#args) + ") = (", args, ")"))
#define ORANGE(args...) \
  (cerr << "#> [" + string(#args) + ") = ", org(args))
#else          // ======== OnlineJudge ========
#define DEBUG(...) ((void)0)
#define ORANGE(...) ((void)0)
#endif

#define ll long long
#define ld long double
#define INF 0x3f3f3f3f
#define LLINF 0x3f3f3f3f3f3f3f3f
#define NINF 0xc1c1c1c1
#define NLLINF 0xc1c1c1c1c1c1c1c1
#define X first
#define Y second
#define PB emplace_back
#define pll pair<ll, ll>
#define MEM(a,n) memset(a, n, sizeof(a))
#define io ios::sync_with_stdio(0); cin.tie(0); cout.
    tie(0);
const int MXN = + 5;
mt19937 rng(chrono::sQteady_clock::now().
    time_since_epoch().count());

void sol(){}
int main(){
  io int t=1;
  // cin >> t;
  while(t--){ sol(); } }
```

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 $u \to v$, 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

```cpp
// unique
sort(a.begin(), a.end());
a.resize(unique(a.begin(), a.end()) - a.begin());
// O(n) a[k] = kth small, a[i] < a[k] if i < k
nth_element(a.begin(), a.begin()+k, a.end());
// stable_sort(a.begin(), a.end())
// lower_bound: first element >= val
// upper_bound: first element >  val
// set_union, set_intersection, set_difference,
//   set_symmetric_difference
set_union(a.begin(), a.end(), b.begin(), b.end(),
  inserter(c, c.begin()));
//next_permutation prev_permutation(sort/reverse first)
do{ for(auto i : a) cout << i << ' ';
```

```
} while(next_permutation(a.begin(), a.end()));
```

1.5 Bi/Ternary Search

```
while(l < r){ // first l of check(l) == true
  ll m = (l + r) >> 1;
  if(!check(m)) l = m + 1; else r = m; }
while(l < r){ // last l of check(l) == false
  ll m = (l + r + 1) >> 1;
  if(!check(m)) l = m;     else r = m - 1; }
while(l < r){
  ll ml = l + (r - l) / 3, mr = r - (r - l) / 3;
  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：
您是否在本地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

```
struct zkwflow{
  static const int MXN = 10000;
  struct Edge{ int v, f, re; ll w;};
  int n, s, t, ptr[MXN]; bool vis[MXN]; ll dis[MXN];
  vector<Edge> E[MXN];
  void init(int _n,int _s,int _t){
    n=_n,s=_s,t=_t;
    for(int i=0;i<n;i++) E[i].clear();
  }
  void addEdge(int u, int v, int f, ll w){
    E[u].push_back({v, f, E[v].size(), w});
    E[v].push_back({u, 0 ,E[u].size()-1, -w});
  }
  bool SPFA(){
    fill_n(dis, n, LLINF); memset(vis, 0, 4 * n);
    queue<int> q; q.push(s); dis[s] = 0;
    while (!q.empty()){
      int u = q.front(); q.pop(); vis[u] = false;
      for(auto &it : E[u]){
        if(it.f > 0 && dis[it.v] > dis[u] + it.w){
          dis[it.v] = dis[u] + it.w;
          if(!vis[it.v]){
            vis[it.v] = 1; q.push(it.v);
          } } } }
    return dis[t] != LLINF;
  }
  int DFS(int u, int nf){
    if(u == t) return nf;
    int res =0; vis[u] = 1;
    for(int &i = ptr[u]; i < (int)E[u].size(); ++i){
      auto &it = E[u][i];
      if(it.f>0&&dis[it.v]==dis[u]+it.w&&!vis[it.v]){
        int tf = DFS(it.v, min(nf,it.f));
        res += tf, nf -= tf, it.f -= tf;
        E[it.v][it.re].f += tf;
        if(nf == 0){ vis[u] = false; break; }
      }
    }
    return res;
  }
  pair<int,ll> flow(){
    int flow = 0; ll cost=0;
    while (SPFA()){
      memset(ptr, 0, 4 * n);
      int f = DFS(s, INF);
      flow += f; cost += dis[t] * f;
    }
    return{ flow, cost };
  }
} flow;
```

2.2 Dinic

求最大流 $O(N^2E)$，求二分最大匹配 $O(E\sqrt{N})$)
dinic.init(n, st, en) $\Rightarrow$ 0-base
dinic.addEdge(u, v, f) $\Rightarrow u \to v$, flow f units
dinic.run() $\Rightarrow$ return max flow from st to en
反向邊為該邊的流量
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

```
const int MXN = 10005;
struct Dinic{
  struct Edge{ ll v, f, re; };
  int n, s, t, lvl[MXN];
  vector<Edge> e[MXN];
  void init(int _n, int _s, int _t){
    n = _n; s = _s; t = _t;
    for(int i = 0; i < n; ++i) e[i].clear(); }
  void addEdge(int u, int v, ll f = 1){
    e[u].push_back({v, f, e[v].size()});
    e[v].push_back({u, 0, e[u].size() - 1}); }
  bool bfs(){
    memset(lvl, -1, n * 4);
    queue<int> q;
    q.push(s);
    lvl[s] = 0;
    while(!q.empty()){
      int u = q.front(); q.pop();
      for(auto &i : e[u])
        if(i.f > 0 && lvl[i.v] == -1)
          lvl[i.v] = lvl[u] + 1, q.push(i.v); }
    return lvl[t] != -1; }
  ll dfs(int u, ll nf){
    if(u == t) return nf;
    ll res = 0;
    for(auto &i : e[u])
      if(i.f > 0 && lvl[i.v] == lvl[u] + 1){
        ll tmp = dfs(i.v, min(nf, i.f));
        res += tmp, nf -= tmp, i.f -= tmp;
        e[i.v][i.re].f += tmp;
        if(nf == 0) return res; }
    if(!res) lvl[u] = -1;
    return res; }
  ll run(ll res){
    while(bfs()) res += dfs(s, LLINF);
    return res; } };
```

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

```
const int MXN = 1005;
struct KM{ // 1-base
  int n, mx[MXN], my[MXN], pa[MXN];
  ll g[MXN][MXN], lx[MXN], ly[MXN], sy[MXN];
  bool vx[MXN], vy[MXN];
  void init(int _n){
    n = _n;
    MEM(g, 0); }
  void addEdge(int x, int y, ll w){ g[x][y] = w; }
  void augment(int y){
    for(int x, z; y; y = z)
      x = pa[y], z = mx[x], my[y] = x, mx[x] = y; }
  void bfs(int st){
    for(int i = 1; i <= n; ++i)
      sy[i] = LLINF, vx[i] = vy[i] = 0;
    queue<int> q; q.push(st);
    for(;;){
      while(!q.empty()){
        int x = q.front(); q.pop();
        vx[x] = 1;
        for(int y = 1; y <= n; ++y)
          if(!vy[y]){
            ll t = lx[x] + ly[y] - g[x][y];
            if(t == 0){
              pa[y] = x;
              if(!my[y]){ augment(y); return; }
              vy[y] = 1, q.push(my[y]); }
            else if(sy[y] > t) pa[y] = x, sy[y] = t;} }
      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; } };
```

2.4 Directed MST *

```
struct DMST {
  struct Edge{ int u, v, c;
    Edge(int u, int v, int c):u(u),v(v),c(c){} };
  int v, e, root;
  Edge edges[MXN];
  int newV(){ return ++v; }
  void addEdge(int u, int v, int c)
   { edges[++e] = Edge(u, v, c); }
  bool con[MXN];
  int mnInW[MXN], prv[MXN], cyc[MXN], vis[MXN];
  int run(){
    memset(con, 0, 4*(V+1));
    int r1 = 0, r2 = 0;
    while(1){
      fill(mnInW, mnInW+V+1, INF);
      fill(prv, prv+V+1, -1);
      for(int i = 1; i <= e; ++i){
        int u=edges[i].u, v=edges[i].v, c=edges[i].c;
        if(u != v && v != root && c < mnInW[v])
          mnInW[v] = c, prv[v] = u; }
      fill(vis, vis+V+1, -1);
      fill(cyc, cyc+V+1, -1);
      r1 = 0;
      bool jf = 0;
      for(int i = 1; i <= v; ++i){
        if(con[i]) continue ;
        if(prv[i] == -1 && i != root) return -1;
        if(prv[i] > 0) r1 += mnInW[i];
        int s;
        for(s = i; s != -1 && vis[s] == -1; s = prv[s])
          vis[s] = i;
        if(s > 0 && vis[s] == i){
          jf = 1; int v = s;
          do{ cyc[v] = s, con[v] = 1;
            r2 += mnInW[v]; v = prv[v];
          }while(v != s);
          con[s] = 0;
      } }
      if(!jf) break ;
      for(int i = 1; i <= e; ++i){
        int &u = edges[i].u;
        int &v = edges[i].v;
        if(cyc[v] > 0) edges[i].c -= mnInW[edges[i].v];
        if(cyc[u] > 0) edges[i].u = cyc[edges[i].u];
        if(cyc[v] > 0) edges[i].v = cyc[edges[i].v];
        if(u == v) edges[i--] = edges[E--];
    } }
    return r1+r2;}};
```

2.5 SW min-cut (不限 S-T 的 min-cut) *

```
struct SW{ // O(V^3)
  int n,vst[MXN],del[MXN];
  int edge[MXN][MXN],wei[MXN];
  void init(int _n){
    n = _n; memset(del, 0, sizeof(del));
    memset(edge, 0, sizeof(edge));
  }
  void addEdge(int u, int v, int w){
    edge[u][v] += w; edge[v][u] += w;
  }
  void search(int &s, int &t){
    memset(vst, 0, sizeof(vst)); memset(wei, 0, sizeof(
        wei));
    s = t = -1;
    while (true){
      int mx=-1, cur=0;
      for (int i=0; i<n; i++)
        if (!del[i] && !vst[i] && mx<wei[i])
          cur = i, mx = wei[i];
      if (mx == -1) break;
      vst[cur] = 1;
      s = t; t = cur;
      for (int i=0; i<n; i++)
        if (!vst[i] && !del[i]) wei[i] += edge[cur][i];
    }
  }
  int solve(){
    int res = 2147483647;
    for (int i=0,x,y; i<n-1; i++){
      search(x,y);
      res = min(res,wei[y]);
      del[y] = 1;
      for (int j=0; j<n; j++)
        edge[x][j] = (edge[j][x] += edge[y][j]);
    }
    return res;
}  }graph;
```

2.6 Bounded Max Flow

```
// flow use ISAP
// Max flow with lower/upper bound on edges
// source = 1 , sink = n
int in[ N ] , out[ N ];
int l[ M ] , r[ M ] , a[ M ] , b[ M ];//0-base,a下界,b
    上界
int solve(){
  flow.init( n );   //n為點的數量,m為邊的數量,點是1-
      base
  for( int i = 0 ; i < m ; i ++ ){
    in[ r[ i ] ] += a[ i ];
    out[ l[ i ] ] += a[ i ];
    flow.addEdge( l[ i ] , r[ i ] , b[ i ] - a[ i ] );
    // flow from l[i] to r[i] must in [a[ i ], b[ i ]]
  }
  int nd = 0;
  for( int i = 1 ; i <= n ; i ++ ){
    if( in[ i ] < out[ i ] ){
      flow.addEdge( i , flow.t , out[ i ] - in[ i ] );
```

```
    nd += out[ i ] - in[ i ];
  }
  if( out[ i ] < in[ i ] )
    flow.addEdge( flow.s , i , in[ i ] - out[ i ] );
}
// original sink to source
flow.addEdge( n , 1 , INF );
if( flow.maxflow() != nd )
  return -1; // no solution
int ans = flow.G[ 1 ].back().c; // source to sink
flow.G[ 1 ].back().c = flow.G[ n ].back().c = 0;
// take out super source and super sink
for( size_t i = 0 ; i < flow.G[ flow.s ].size() ; i
    ++ ){
  flow.G[ flow.s ][ i ].c = 0;
  Edge &e = flow.G[ flow.s ][ i ];
  flow.G[ e.v ][ e.r ].c = 0;
}
for( size_t i = 0 ; i < flow.G[ flow.t ].size() ; i
    ++ ){
  flow.G[ flow.t ][ i ].c = 0;
  Edge &e = flow.G[ flow.t ][ i ];
  flow.G[ e.v ][ e.r ].c = 0;
}
flow.addEdge( flow.s , 1 , INF );
flow.addEdge( n , flow.t , INF );
flow.reset();
return ans + flow.maxflow();
}
```

2.7 Flow Method *

建模方式:
限制條件有幾大類: 分層建 flow
每個點有不同限制: 拆點 (出入點等等)
全部同限制: 超級源點、匯點
可以設定流量，利用費用找答案
例如要找 s-t 中 k 條路徑，可以用超源連 s 流量 k，超匯同理
每個點之間流量 1，費用是邊的長度，跑最小費用流
Maximize c^T x subject to $Ax \le b, x \ge 0$;
with the corresponding symmetric dual problem,
Minimize b^T y subject to $A^T y \ge c, y \ge 0$.
Maximize c^T x subject to $Ax \le b$;
with the corresponding asymmetric dual problem,
Minimize b^T y subject to $A^T y = c, y \ge 0$.
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->v,cap=w), (v->u,cap=w)
3. For each node v, from v to sink with cap = S + 2 * D - deg[v] - 2 * (W of v)
where $deg[v] = \sum$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 $x \to y$ 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 $S \to v$ with capacity $in(v)$, otherwise, connect $v \to T$ with capacity $-in(v)$.
     - To maximize, connect $t \to s$ with capacity ∞ (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 $t \to s$ with capacity ∞ 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: $y \to x$ if $(x, y) \in M$, $x \to y$ 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 $s \to v$, $v \in G$ with capacity K
  4. For each edge (u, v, w) in G, connect $u \to v$ and $v \to u$ with capacity w
  5. For $v \in G$, connect it with sink $v \to t$ 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 $u' \to v'$ with weight $w(u, v)$.
  2. Connect $v \to v'$ 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

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

```
ll fpow(ll a, ll b, ll m){
  ll ret = 1;
  a %= m;
  while(b){
    if(b&1) ret = ret * a % m;
    a = a * a % m;
    b >>= 1; }
  return ret; }

ll fa[MXN], fi[MXN];
void init(){
  fa[0] = 1;
  for(ll i = 1; i < MXN; ++i)
    fa[i] = fa[i - 1] * i % MOD;
  fi[MXN - 1] = fpow(fa[MXN - 1], MOD - 2, MOD);
  for(ll i = MXN - 1; i > 0; --i)
    fi[i - 1] = fi[i] * i % MOD; }

ll c(ll a, ll b){
  return fa[a] * fi[b] % MOD * fi[a - b] % MOD; }
```

3.2 Ext GCD

```
//a * p.first + b * p.second = gcd(a, b)
pair<ll, ll> extgcd(ll a, ll b) {
  pair<ll, ll> res;
  if (a < 0) {
    res = extgcd(-a, b);
    res.first *= -1;
    return res;
  }
  if (b < 0) {
    res = extgcd(a, -b);
    res.second *= -1;
    return res;
  }
  if (b == 0) return {1, 0};
  res = extgcd(b, a % b);
  return {res.second, res.first - res.second * (a / b)
      };
}
```

3.3 Sieve 質數篩

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

3.4 FFT *

```
// const int MAXN = 262144;
// (must be 2^k)
// before any usage, run pre_fft() first
typedef long double ld;
typedef complex<ld> cplx; //real() ,imag()
const ld PI = acosl(-1);
const cplx I(0, 1);
cplx omega[MAXN+1];
void pre_fft(){
  for(int i=0; i<=MAXN; i++)
    omega[i] = exp(i * 2 * PI / MAXN * I);
}
// n must be 2^k
void fft(int n, cplx a[], bool inv=false){
  int basic = MAXN / n;
  int theta = basic;
  for (int m = n; m >= 2; m >>= 1) {
    int mh = m >> 1;
    for (int i = 0; i < mh; i++) {
      cplx w = omega[inv ? MAXN-(i*theta%MAXN)
                         : i*theta%MAXN];
      for (int j = i; j < n; j += m) {
        int k = j + mh;
        cplx x = a[j] - a[k];
        a[j] += a[k];
        a[k] = w * x;
    } }
    theta = (theta * 2) % MAXN;
  }
  int i = 0;
  for (int j = 1; j < n - 1; j++) {
    for (int k = n >> 1; k > (i ^= k); k >>= 1);
    if (j < i) swap(a[i], a[j]);
  }
  if(inv) for (i = 0; i < n; i++) a[i] /= n;
}
cplx arr[MAXN+1];
inline void mul(int _n,ll a[],int _m,ll b[],ll ans[])
{
  int n=1,sum=_n+_m-1;
  while(n<sum)
    n<<=1;
  for(int i=0;i<n;i++)
  {
    double x=(i<_n?a[i]:0),y=(i<_m?b[i]:0);
    arr[i]=complex<double>(x+y,x-y);
  }
  fft(n,arr);
  for(int i=0;i<n;i++)
    arr[i]=arr[i]*arr[i];
  fft(n,arr,true);
  for(int i=0;i<sum;i++)
    ans[i]=(long long int)(arr[i].real()/4+0.5);
}
```

3.5 NTT *

```
// Remember coefficient are mod P
/* p=a*2^n+1
   n    2^n         p           a     root
   16   65536       65537       1     3
   20   1048576     7340033     7     3 */
// (must be 2^k)
template<LL P, LL root, int MAXN>
struct NTT{
  static LL bigmod(LL a, LL b) {
    LL res = 1;
    for (LL bs = a; b; b >>= 1, bs = (bs * bs) % P)
      if(b&1) res=(res*bs)%P;
    return res;
  }
  static LL inv(LL a, LL b) {
    if(a==1)return 1;
    return (((LL)(a-inv(b%a,a))*b+1)/a)%b;
  }
  LL omega[MAXN+1];
  NTT() {
    omega[0] = 1;
    LL r = bigmod(root, (P-1)/MAXN);
    for (int i=1; i<=MAXN; i++)
      omega[i] = (omega[i-1]*r)%P;
  }
  // n must be 2^k
  void tran(int n, LL a[], bool inv_ntt=false){
    int basic = MAXN / n , theta = basic;
    for (int m = n; m >= 2; m >>= 1) {
      int mh = m >> 1;
      for (int i = 0; i < mh; i++) {
        LL w = omega[i*theta%MAXN];
        for (int j = i; j < n; j += m) {
          int k = j + mh;
          LL x = a[j] - a[k];
          if (x < 0) x += P;
          a[j] += a[k];
          if (a[j] > P) a[j] -= P;
          a[k] = (w * x) % P;
        }
      }
      theta = (theta * 2) % MAXN;
    }
    int i = 0;
    for (int j = 1; j < n - 1; j++) {
      for (int k = n >> 1; k > (i ^= k); k >>= 1);
      if (j < i) swap(a[i], a[j]);
    }
    if (inv_ntt) {
      LL ni = inv(n,P);
      reverse( a+1 , a+n );
      for (i = 0; i < n; i++)
        a[i] = (a[i] * ni) % P;
    }
  }
};
const LL P=2013265921,root=31;
const int MAXN=4194304;
NTT<P, root, MAXN> ntt;
```

3.6 Linear Recurrence *

```
// Usage: linearRec({0, 1}, {1, 1}, k) //k'th fib
typedef vector<ll> Poly;
//S:前i項的值,tr:遞迴系數,k:求第k項
ll linearRec(Poly& S, Poly& tr, ll k) {
  int n = tr.size();
  auto combine = [&](Poly& a, Poly& b) {
    Poly res(n * 2 + 1);
    rep(i,0,n+1) rep(j,0,n+1)
      res[i+j]=(res[i+j] + a[i]*b[j])%mod;
    for(int i = 2*n; i > n; --i) rep(j,0,n)
      res[i-1-j]=(res[i-1-j] + res[i]*tr[j])%mod;
    res.resize(n + 1);
    return res;
  };
  Poly pol(n + 1), e(pol);
  pol[0] = e[1] = 1;
  for (++k; k; k /= 2) {
    if (k % 2) pol = combine(pol, e);
    e = combine(e, e);
  }
  ll res = 0;
  rep(i,0,n) res=(res + pol[i+1]*S[i])%mod;
  return res;
}
```

3.7 Miller Rabin

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

```
// magic numbers when n <
// 4,759,123,141      : 2, 7, 61
```

```
// 1,122,004,669,633 : 2, 13, 23, 1662803
// 3,474,749,660,383 : 2, 3, 5, 7, 11, 13
// 2^64              : 2, 325, 9375, 28178, 450775,
   9780504, 1795265022
// Make sure testing integer is in range [2, n□2] if
   you want to use magic.
vector<ll> magic = {};
bool witness(ll a, ll n, ll u, ll t){
  if(!a) return 0;
  ll x = fpow(a, u, n);
  while(t--) {
    ll nx = x * x % n;
    if(nx == 1 && x != 1 && x != n - 1) return 1;
    x = nx; }
  return x != 1; }
bool isprime(ll n) {
  if(n < 2) return 0;
  if(~n & 1) return n == 2;
  ll u = n - 1, t = 0;
  while(~u & 1) u >>= 1, t++;
  for(auto i : magic){
    ll a = i % n;
    if(witness(a, n, u, t)) return 0; }
  return 1; }
```

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

```
/* faulhaber’ s formula -
 * cal power sum formula of all p=1~k in O(k^2) */
#define MAXK 2500
const int mod = 1000000007;
int b[MAXK]; // bernoulli number
int inv[MAXK+1]; // inverse
int cm[MAXK+1][MAXK+1]; // combinactories
int co[MAXK][MAXK+2]; // coeeficient of x^j when p=i
inline int getinv(int x) {
  int a=x,b=mod,a0=1,a1=0,b0=0,b1=1;
  while(b) {
    int q,t;
    q=a/b; t=b; b=a-b*q; a=t;
    t=b0; b0=a0-b0*q; a0=t;
    t=b1; b1=a1-b1*q; a1=t;
  }
  return a0<0?a0+mod:a0;
}
inline void pre() {
  /* combinational */
  for(int i=0;i<=MAXK;i++) {
    cm[i][0]=cm[i][i]=1;
    for(int j=1;j<i;j++)
      cm[i][j]=add(cm[i-1][j-1],cm[i-1][j]);
  }
  /* inverse */
  for(int i=1;i<=MAXK;i++) inv[i]=getinv(i);
  /* bernoulli */
  b[0]=1; b[1]=getinv(2); // with b[1] = 1/2
  for(int i=2;i<MAXK;i++) {
    if(i&1) { b[i]=0; continue; }
    b[i]=1;
    for(int j=0;j<i;j++)
      b[i]=sub(b[i],
               mul(cm[i][j],mul(b[j], inv[i-j+1])));
  }
  /* faulhaber */
  // sigma_x=1~n {x^p} =
  //   1/(p+1) * sigma_j=0~p {C(p+1,j)*Bj*n^(p-j+1)}
  for(int i=1;i<MAXK;i++) {
    co[i][0]=0;
    for(int j=0;j<=i;j++)
      co[i][i-j+1]=mul(inv[i+1], mul(cm[i+1][j], b[j]))
          ;
  }
}
/* sample usage: return f(n,p) = sigma_x=1~n (x^p) */
inline int solve(int n,int p) {
  int sol=0,m=n;
  for(int i=1;i<=p+1;i++) {
    sol=add(sol,mul(co[p][i],m));
    m = mul(m, n);
  }
  return sol;
}
```

3.9 Chinese Remainder *

```
LL x[N],m[N];
LL CRT(LL x1, LL m1, LL x2, LL m2) {
  LL g = __gcd(m1, m2);
  if((x2 - x1) % g) return -1;// no sol
  m1 /= g; m2 /= g;
  pair<LL,LL> p = gcd(m1, m2);
  LL lcm = m1 * m2 * g;
  LL res = p.first * (x2 - x1) * m1 + x1;
  return (res % lcm + lcm) % lcm;
}
LL solve(int n){ // n>=2,be careful with no solution
  LL res=CRT(x[0],m[0],x[1],m[1]),p=m[0]/__gcd(m[0],m
      [1])*m[1];
  for(int i=2;i<n;i++){
    res=CRT(res,p,x[i],m[i]);
    p=p/__gcd(p,m[i])*m[i];
  }
  return res;
}
```

3.10 Pollard Rho *

```
// does not work when n is prime  O(n^(1/4))
LL f(LL x, LL mod){ return add(mul(x,x,mod),1,mod); }
LL pollard_rho(LL n) {
  if(!(n&1)) return 2;
  while(true){
    LL y=2, x=rand()%(n-1)+1, res=1;
    for(int sz=2; res==1; sz*=2) {
      for(int i=0; i<sz && res<=1; i++) {
        x = f(x, n);
        res = __gcd(abs(x-y), n);
      }
      y = x;
    }
    if (res!=0 && res!=n) return res;
} }
```

3.11 Josephus Problem *

```
int josephus(int n, int m){ //n人每m次
    int ans = 0;
    for (int i=1; i<=n; ++i)
        ans = (ans + m) % i;
    return ans;
}
```

3.12 Gaussian Elimination *

```
const int GAUSS_MOD = 100000007LL;
struct GAUSS{
    int n;
    vector<vector<int>> v;
    int ppow(int a , int k){
        if(k == 0) return 1;
        if(k % 2 == 0) return ppow(a * a % GAUSS_MOD ,
            k >> 1);
        if(k % 2 == 1) return ppow(a * a % GAUSS_MOD ,
            k >> 1) * a % GAUSS_MOD;
    }
    vector<int> solve(){
        vector<int> ans(n);
        REP(now , 0 , n){
            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
            ][i] , GAUSS_MOD - 2) % GAUSS_MOD;
```

```
        return ans;
    }
    // gs.v.clear() , gs.v.resize(n , vector<int>(n + 1
        , 0));
} gs;
```

3.13 歐拉函數降冪公式

```
ll eulerFunction(ll x) {
  ll ret = x;
  for(ll i = 2; i * i <= x; ++i) {
    if(x % i == 0) {
      ret -= ret / i;
      while(x % i == 0) x /= i;
    }
  }
  if(x > 1) ret -= ret / x;
  return ret;
}

ll eulerPow(ll a, string b, ll mod) {
  ll ret = eulerFunction(mod);
  ll p = 0;
  for(ll i = 0; i < b.size(); ++i) {
    p = (p * 10 + b[i] - '0') % ret;
  }
  p += ret;
  return fastPow(a, p, mod);
}
```

3.14 貝爾數 Bell

```
ll bell[MXN][MXN];

void bellf(int n) {
  bell[1][1] = 1;
  for(int i = 2; i <= n; ++i) {
    bell[i][1] = bell[i - 1][i - 1];
    for(int j = 2; j <= i; ++j) {
      bell[i][j] = bell[i - 1][j - 1] + bell[i][j - 1];
    }
  }
}
```

3.15 Result *

- Lucas' Theorem :
  For $n, m \in \mathbb{Z}^*$ and prime P, $C(m,n)$ mod $P = \Pi(C(m_i, n_i))$ where m_i is the i-th digit of m in base P.

- Stirling approximation :
  $n! \approx \sqrt{2\pi n}(\frac{n}{e})^n e^{\frac{1}{12n}}$

- Stirling Numbers(permutation $|P| = n$ with k cycles):
  $S(n,k) =$ coefficient of x^k in $\Pi_{i=0}^{n-1}(x+i)$

- Stirling Numbers(Partition n elements into k non-empty set):
  $S(n,k) = \frac{1}{k!}\sum\limits_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^n$

- 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 : $C_n = \binom{2n}{n}/(n+1)$
  $C_n^{n+m} - C_{n+1}^{n+m} = (m+n)!\frac{n-m+1}{n+1} \quad for \quad n \geq m$
  $C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$
  $C_0 = 1 \quad and \quad C_{n+1} = 2(\frac{2n+1}{n+2})C_n$
  $C_0 = 1 \quad and \quad C_{n+1} = \sum_{i=0}^{n} C_i C_{n-i} \quad for \quad n \geq 0$

- 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):
  $(\sum_{i=1}^{m} c^{gcd(i,m)})/m$

- Burnside lemma:
  $|X/G| = \frac{1}{|G|}\sum\limits_{g \in G}|X^g|$

- 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) :
  $B_0 = 1$
  $B_n = \sum_{k=0}^{n} s(n,k) \quad (second - stirling)$
  $B_{n+1} = \sum_{k=0}^{n}\binom{n}{k}B_k$

- Wilson's theorem :
  $(p-1)! \equiv -1(mod \ p)$

- Fermat's little theorem :
  $a^p \equiv a(mod \ p)$

- Euler's totient function:
  $A^{B^C} mod \ p = pow(A, pow(B, C, p-1)) mod \ p$

- 歐拉函數降冪公式:
  $A^B \bmod C = A^{B \bmod \phi(c) + \phi(c)} \bmod C$

- 6 的倍數:
  $(a-1)^3 + (a+1)^3 + (-a)^3 + (-a)^3 = 6a$

4 Geometry

4.1 definition

```
const ld EPS = 1e-8;
const ld PI = acos(-1);
int dcmp(ld x){ // float x (<, ==, >) y -> (-1, 0, 1)
  if(abs(x) < EPS) return 0;
  else return x < 0 ? -1 : 1;
}
struct Pt{
  ld x, y; // 改三維記得其他函式都要改
  Pt(ld _x = 0, ld _y = 0): x(_x), y(_y){}
  Pt operator+(const Pt &a) const{
    return Pt(x + a.x, y + a.y); }
  Pt operator-(const Pt &a) const{
    return Pt(x - a.x, y - a.y); }
  Pt operator*(const ld &a) const{
    return Pt(x * a, y * a); }
  Pt operator/(const ld &a) const{
    return Pt(x / a, y / a); }
  ld operator*(const Pt &a) const{ // dot product
    return x * a.x + y * a.y; }
  ld operator^(const Pt &a) const{ // cross product
    return x *a.y - y * a.x; }
  bool operator<(const Pt &a) const{
    return x < a.x || (x == a.x && y < a.y); }
    // return dcmp(x-a.x) < 0 ||
    //   (dcmp(x-a.x) == 0 && dcmp(y-a.y) < 0); }
  bool operator==(const Pt &a) const{
    return dcmp(x - a.x) == 0 && dcmp(y - a.y) == 0; }
  int qua() { // 在哪個象限(軸上點歸類到逆時針的象限)
    if(x > 0 && y >= 0) return 1;
    if(x <= 0 && y > 0) return 2;
    if(x < 0 && y <= 0) return 3;
    if(x >= 0 && y < 0) return 4; }
  ld angle() const{ // -pi ~ pi
    if(dcmp(x) == 0 && dcmp(y) == 0) return 0;
    return atan2(y, x); } };
ld norm2(const Pt &a){
  return a * a; }
ld norm(const Pt &a){ // norm(a - b) = dis of a, b
  return sqrt(norm2(a)); }
Pt perp(const Pt &a){ // 垂直向量(順時針旋轉90度)
  return Pt(-a.y, a.x); }
Pt rotate(const Pt &a, ld ang){
  return Pt(a.x * cos(ang) - a.y * sin(ang),
            a.x * sin(ang) + a.y * cos(ang)); }
struct Line{
  Pt s, e, v; // start, end, end - start
  ld ang; // angle of v
  Line(Pt _s = Pt(0, 0), Pt _e = Pt(0, 0)):
    s(_s), e(_e) { v = e - s; ang = atan2(v.y, v.x); }
  bool operator<(const Line &L) const{ // sort by angle
    return ang < L.ang; } };
struct Circle{
  Pt o; ld r;
  Circle(Pt _o = Pt(0, 0), ld _r = 0): o(_o), r(_r){}
  bool inside(const Pt &a) const {
    return norm2(a - o) <= r * r; } };
```

4.2 halfPlaneIntersection *

```
#define N 100010
#define EPS 1e-8
#define SIDE 10000000
struct PO{ double x , y ; } p[ N ], o ;
struct LI{
  PO a, b;
  double angle;
  void in( double x1 , double y1 , double x2 , double
      y2 ){
    a.x = x1 ; a.y = y1 ; b.x = x2 ; b.y = y2;
  }
}li[ N ] , deq[ N ];
int n , m , cnt;
inline int dc( double x ){
  if ( x > EPS ) return 1;
  else if ( x < -EPS ) return -1;
  return 0;
}
inline PO operator-( PO a, PO b ){
  PO c;
  c.x = a.x - b.x ; c.y = a.y - b.y;
  return c;
}
inline double cross( PO a , PO b , PO c ){
  return ( b.x - a.x ) * ( c.y - a.y ) - ( b.y - a.y )
      * ( c.x - a.x );
}
inline bool cmp( const LI &a , const LI &b ){
  if( dc( a.angle - b.angle ) == 0 ) return dc( cross(
      a.a , a.b , b.a ) ) < 0;
  return a.angle > b.angle;
}
inline PO getpoint( LI &a , LI &b ){
  double k1 = cross( a.a , b.b , b.a );
  double k2 = cross( a.b , b.a , b.b );
  PO tmp = a.b - a.a , ans;
  ans.x = a.a.x + tmp.x * k1 / ( k1 + k2 );
  ans.y = a.a.y + tmp.y * k1 / ( k1 + k2 );
  return ans;
}
inline void getcut(){
  sort( li + 1 , li + 1 + n , cmp ); m = 1;
  for( int i = 2 ; i <= n ; i ++ )
    if( dc( li[ i ].angle - li[ m ].angle ) != 0 )
      li[ ++ m ] = li[ i ];
  deq[ 1 ] = li[ 1 ]; deq[ 2 ] = li[ 2 ];
  int bot = 1 , top = 2;
  for( int i = 3 ; i <= m ; i ++ ){
    while( bot < top && dc( cross( li[ i ].a , li[ i ].
        b , getpoint( deq[ top ] , deq[ top - 1 ] ) ) )
         < 0 ) top -- ;
    while( bot < top && dc( cross( li[ i ].a , li[ i ].
        b , getpoint( deq[ bot ] , deq[ bot + 1 ] ) ) )
         < 0 ) bot ++ ;
    deq[ ++ top ] = li[ i ] ;
  }
  while( bot < top && dc( cross( deq[ bot ].a , deq[
      bot ].b , getpoint( deq[ top ] , deq[ top - 1 ] )
       ) ) < 0 ) top --;
  while( bot < top && dc( cross( deq[ top ].a , deq[
      top ].b , getpoint( deq[ bot ] , deq[ bot + 1 ] )
       ) ) < 0 ) bot ++;
  cnt = 0;
  if( bot == top ) return;
  for( int i = bot ; i < top ; i ++ ) p[ ++ cnt ] =
      getpoint( deq[ i ] , deq[ i + 1 ] );
  if( top - 1 > bot ) p[ ++ cnt ] = getpoint( deq[ bot
      ] , deq[ top ] );
}
double px[ N ] , py[ N ];
void read( int rm ) {
  for( int i = 1 ; i <= n ; i ++ ) px[ i + n ] = px[ i
      ] , py[ i + n ] = py[ i ];
  for( int i = 1 ; i <= n ; i ++ ){
    // half-plane from li[ i ].a -> li[ i ].b
    li[ i ].a.x = px[ i + rm + 1 ]; li[ i ].a.y = py[ i
       + rm + 1 ];
    li[ i ].b.x = px[ i ]; li[ i ].b.y = py[ i ];
    li[ i ].angle = atan2( li[ i ].b.y - li[ i ].a.y ,
        li[ i ].b.x - li[ i ].a.x ) ;
  }
}
inline double getarea( int rm ){
  read( rm ); getcut();
  double res = 0.0;
  p[ cnt + 1 ] = p[ 1 ];
  for( int i = 1 ; i <= cnt ; i ++ ) res += cross( o ,
      p[ i ] , p[ i + 1 ] ) ;
  if( res < 0.0 ) res *= -1.0;
  return res;
}
```

4.3 Convex Hull *

```
double cross(Pt o, Pt a, Pt b){
  return (a-o) ^ (b-o);
}
vector<Pt> convex_hull(vector<Pt> pt){
  sort(pt.begin(),pt.end());
  int top=0;
  vector<Pt> stk(2*pt.size());
  for (int i=0; i<(int)pt.size(); i++){
    while (top >= 2 && cross(stk[top-2],stk[top-1],pt[i
        ]) <= 0)
      top--;
    stk[top++] = pt[i];
  }
  for (int i=pt.size()-2, t=top+1; i>=0; i--){
    while (top >= t && cross(stk[top-2],stk[top-1],pt[i
        ]) <= 0)
      top--;
    stk[top++] = pt[i];
  }
  stk.resize(top-1);
  return stk;
}
```

4.4 Convex Hull trick *

```
/* Given a convexhull, answer querys in O(\lg N)
CH should not contain identical points, the area should
be > 0, min pair(x, y) should be listed first */
double det( const Pt& p1 , const Pt& p2 )
{ return p1.X * p2.Y - p1.Y * p2.X; }
struct Conv{
  int n;
  vector<Pt> a;
  vector<Pt> upper, lower;
  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){
    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);
     return 1;
   }
   return 0;
}  };
```

4.5 掃描的線

```
ScanLine sl;
sl.add(兩點座標);
sl.run()

template <typename T>
struct SegmentTree{
  struct Node{
    T len = 0, tag = 0;
    int nl, nr;
    Node *l, *r;
  } *root;
  vector<T> vec;
  int n;
  SegmentTree(){}
  void init(vector<T> _vec){
    vec = _vec;
    n = vec.size() - 1;
    root = build(0, n - 1);
  }
  Node* build(int l, int r){
    Node *res = new Node();
    res->nl = l, res->nr = r;
    if(l == r){
      res->l = res->r = nullptr;
      return res;
    }
    int mid = (l + r) >> 1;
    res->l = build(l, mid);
    res->r = build(mid + 1, r);
    return res;
  }
  void push(Node *cur){
    int l = cur->nl, r = cur->nr;
    if(cur->tag) cur->len = vec[r + 1] - vec[l];
    else cur->len = l == r ? 0 : cur->l->len + cur->r->
        len;
  }
  void update(Node *cur, int ql, int qr, int x){
    int l = cur->nl, r = cur->nr;
    if(vec[r + 1] <= ql || qr <= vec[l]) return;
    if(ql <= vec[l] && vec[r + 1] <= qr){
      cur->tag += x;
      push(cur);
      return;
    }
    update(cur->l, ql, qr, x);
    update(cur->r, ql, qr, x);
    push(cur);
  }
  void update(int l, int r, int x){
    update(root, l, r, x);
  }
};
template <typename T>
struct ScanLine{
  struct Line{
    T l, r, h, flag;
    bool operator<(const Line &rhs){
      return h < rhs.h;
    }
  };
  vector<T> vec; vector<Line> line; SegmentTree<T> seg;
  int n, cnt = 0;
  ScanLine(int _n): n(_n << 1) {
    line.resize(n), vec.resize(n);
  }
  void add(int x1, int y1, int x2, int y2){
    line[cnt] = {x1, x2, y1, 1}, line[cnt + 1] = {x1,
        x2, y2, -1};
    vec[cnt] = x1, vec[cnt + 1] = x2;
    cnt += 2;
  }
  T run(){
    T res = 0;
    sort(line.begin(), line.end());
    sort(vec.begin(), vec.end());
    vec.erase(unique(vec.begin(), vec.end()), vec.end()
        );
    seg.init(vec);
    for(int i = 0; i < n - 1; ++i){
      seg.update(line[i].l, line[i].r, line[i].flag);
      res += seg.root->len * (line[i + 1].h - line[i].h
          );
    }
    return res;
  }
};
```

4.6 Polar sort

```
sort(pl.begin(), pl.end(), [&](Pt a, Pt b){
  // a = a - o, b = b - o;
  if(a.qua() == b.qua()) return (a ^ b) > 0;
  return a.qua() < b.qua();
```

```
}); // degree 0 to 359
sort(pl.begin(), pl.end(), [&](Pt a, Pt b){
  return (a - pt[i]).angle() < (b - pt[i]).angle();
}); // degree -180 to 180, slower
```

4.7 Li Chao Segment Tree *

```
struct LiChao_min{
  struct line{
    ll m,c;
    line(ll _m=0,ll _c=0){ m=_m; c=_c; }
    ll eval(ll x){ return m*x+c; } // overflow
  };
  struct node{
    node *l,*r; line f;
    node(line v){ f=v; l=r=NULL; }
  };
  typedef node* pnode;
  pnode root; ll sz,ql,qr;
#define mid ((l+r)>>1)
  void insert(line v,ll l,ll r,pnode &nd){
    /* if(!(ql<=l&&r<=qr)){
      if(!nd) nd=new node(line(0,INF));
      if(ql<=mid) insert(v,l,mid,nd->l);
      if(qr>mid) insert(v,mid+1,r,nd->r);
      return;
    } used for adding segment */
    if(!nd){ nd=new node(v); return; }
    ll trl=nd->f.eval(l),trr=nd->f.eval(r);
    ll vl=v.eval(l),vr=v.eval(r);
    if(trl<=vl&&trr<=vr) return;
    if(trl>vl&&trr>vr) { nd->f=v; return; }
    if(trl>vl) swap(nd->f,v);
    if(nd->f.eval(mid)<v.eval(mid))
      insert(v,mid+1,r,nd->r);
    else swap(nd->f,v),insert(v,l,mid,nd->l);
  }
  ll query(ll x,ll l,ll r,pnode &nd){
    if(!nd) return INF;
    if(l==r) return nd->f.eval(x);
    if(mid>=x)
      return min(nd->f.eval(x),query(x,l,mid,nd->l));
    return min(nd->f.eval(x),query(x,mid+1,r,nd->r));
  }
  /* -sz<=ll query_x<=sz */
  void init(ll _sz){ sz=_sz+1; root=NULL; }
  void add_line(ll m,ll c,ll l=-INF,ll r=INF){
    line v(m,c); ql=l; qr=r; insert(v,-sz,sz,root);
  }
  ll query(ll x) { return query(x,-sz,sz,root); }
};
```

4.8 KD Tree *

```
struct KDTree{  // O(sqrtN + K)
  struct Nd{
    LL x[MXK],mn[MXK],mx[MXK];
    int id,f;
    Nd *l,*r;
  }tree[MXN],*root;
  int n,k;
  LL dis(LL a,LL b){return (a-b)*(a-b);}
  LL dis(LL a[MXK],LL b[MXK]){
    LL ret=0;
    for(int i=0;i<k;i++) ret+=dis(a[i],b[i]);
    return ret;
  }
  void init(vector<vector<LL>> &ip,int _n,int _k){
    n=_n,k=_k;
    for(int i=0;i<n;i++){
      tree[i].id=i;
      copy(ip[i].begin(),ip[i].end(),tree[i].x);
    }
    root=build(0,n-1,0);
  }
  Nd* build(int l,int r,int d){
    if(l>r) return NULL;
    if(d==k) d=0;
    int m=(l+r)>>1;
    nth_element(tree+l,tree+m,tree+r+1,[&](const Nd &a,
        const Nd &b){return a.x[d]<b.x[d];});
    tree[m].f=d;
    copy(tree[m].x,tree[m].x+k,tree[m].mn);
    copy(tree[m].x,tree[m].x+k,tree[m].mx);
    tree[m].l=build(l,m-1,d+1);
    if(tree[m].l){
      for(int i=0;i<k;i++){
        tree[m].mn[i]=min(tree[m].mn[i],tree[m].l->mn[i
            ]);
        tree[m].mx[i]=max(tree[m].mx[i],tree[m].l->mx[i
            ]);
    } }
    tree[m].r=build(m+1,r,d+1);
    if(tree[m].r){
      for(int i=0;i<k;i++){
        tree[m].mn[i]=min(tree[m].mn[i],tree[m].r->mn[i
            ]);
        tree[m].mx[i]=max(tree[m].mx[i],tree[m].r->mx[i
            ]);
    } }
    return tree+m;
  }
  LL pt[MXK],md;
  int mID;
  bool touch(Nd *r){
    LL d=0;
    for(int i=0;i<k;i++){
      if(pt[i]<=r->mn[i]) d+=dis(pt[i],r->mn[i]);
        else if(pt[i]>=r->mx[i]) d+=dis(pt[i],r->mx[i])
            ;
    }
    return d<md;
  }
  void nearest(Nd *r){
    if(!r||!touch(r)) return;
    LL td=dis(r->x,pt);
    if(td<md) md=td,mID=r->id;
    nearest(pt[r->f]<r->x[r->f]?r->l:r->r);
    nearest(pt[r->f]<r->x[r->f]?r->r:r->l);
  }
  pair<LL,int> query(vector<LL> &_pt,LL _md=1LL<<57){
    mID=-1,md=_md;
    copy(_pt.begin(),_pt.end(),pt);
    nearest(root);
    return {md,mID};
} }tree;
```

4.9 多邊形面積

```
ld polygonArea(vector<Point> &poly, int n) {
  ld res = 0;
  for(int i = 0, j = 0; i < n; ++i) {
    j = (i + 1) % n;
    res += poly[i].x * poly[j].y - poly[j].x * poly[i].
        y;
  }
  return abs(res) / 2;
}
```

4.10 Min Enclosing Circle

```
const int MXN = 1e7;
int n; Pt p[MXN]; // input n, p[0] ~ p[n - 1]
const Circle circumcircle(Pt a,Pt b,Pt c){
  Circle cir;
  ld fa,fb,fc,fd,fe,ff,dx,dy,dd;
  if( iszero( ( b - a ) ^ ( c - a ) ) ){
    if( ( ( b - a ) * ( c - a ) ) <= 0 )
      return Circle((b+c)/2,norm(b-c)/2);
    if( ( ( c - b ) * ( a - b ) ) <= 0 )
      return Circle((c+a)/2,norm(c-a)/2);
    if( ( ( a - c ) * ( b - c ) ) <= 0 )
      return Circle((a+b)/2,norm(a-b)/2);
  }else{
    fa=2*(a.x-b.x);
    fb=2*(a.y-b.y);
    fc=norm2(a)-norm2(b);
    fd=2*(a.x-c.x);
    fe=2*(a.y-c.y);
    ff=norm2(a)-norm2(c);
    dx=fc*fe-ff*fb;
    dy=fa*ff-fd*fc;
    dd=fa*fe-fd*fb;
    cir.o=Pt(dx/dd,dy/dd);
```

```cpp
    cir.r=norm(a-cir.o);
    return cir; } }
inline Circle mec(int fixed,int num){
  int i;
  Circle cir;
  if(fixed==3) return circumcircle(p[0],p[1],p[2]);
  cir=circumcircle(p[0],p[0],p[1]);
  for(i=fixed;i<num;i++) {
    if(cir.inside(p[i])) continue;
    swap(p[i],p[fixed]);
    cir=mec(fixed+1,i+1); }
  return cir;
}
inline ld min_radius() {
  if(n<=1) return 0.0;
  if(n==2) return norm(p[0]-p[1])/2;
  random_shuffle(p, p+n);
  return mec(0,n).r; }
```

4.11 Min Enclosing Ball

```cpp
// Pt : { x , y , z }
const int MXN = 202020;
int n, nouter; Pt pt[MXN], outer[4], res;
ld radius,tmp;
void ball() {
  Pt q[3]; ld m[3][3], sol[3], L[3], det;
  int i,j; res.x = res.y = res.z = radius = 0;
  switch (nouter) {
    case 1: res=outer[0]; break;
    case 2: res=(outer[0]+outer[1])/2;
      radius=norm2(res - outer[0]); break;
    case 3:
      for (i=0; i<2; ++i) q[i]=outer[i+1]-outer[0];
      for (i=0; i<2; ++i) for(j=0; j<2; ++j)
        m[i][j]=(q[i] * q[j])*2;
      for (i=0; i<2; ++i) sol[i]=(q[i] * q[i]);
      if(fabs(det=m[0][0]*m[1][1]-m[0][1]*m[1][0])<EPS)
        return;
      L[0]=(sol[0]*m[1][1]-sol[1]*m[0][1])/det;
      L[1]=(sol[1]*m[0][0]-sol[0]*m[1][0])/det;
      res=outer[0]+q[0]*L[0]+q[1]*L[1];
      radius=norm2(res - outer[0]);
      break;
    case 4:
      for (i=0; i<3; ++i)
        q[i]=outer[i+1]-outer[0], sol[i]=(q[i] * q[i]);
      for (i=0;i<3;++i) for(j=0;j<3;++j)
        m[i][j]=(q[i] * q[j])*2;
      det= m[0][0]*m[1][1]*m[2][2]
        + m[0][1]*m[1][2]*m[2][0]
        + m[0][2]*m[2][1]*m[1][0]
        - m[0][2]*m[1][1]*m[2][0]
        - m[0][1]*m[1][0]*m[2][2]
        - m[0][0]*m[1][2]*m[2][1];
      if (fabs(det)<EPS) return;
      for (j=0; j<3; ++j) {
        for (i=0; i<3; ++i) m[i][j]=sol[i];
        L[j]=(m[0][0]*m[1][1]*m[2][2]
              + m[0][1]*m[1][2]*m[2][0]
              + m[0][2]*m[2][1]*m[1][0]
              - m[0][2]*m[1][1]*m[2][0]
              - m[0][1]*m[1][0]*m[2][2]
              - m[0][0]*m[1][2]*m[2][1]
            ) / det;
        for (i=0; i<3; ++i) m[i][j]=(q[i] * q[j])*2;
      } res=outer[0];
      for (i=0; i<3; ++i) res = res + q[i] * L[i];
      radius=norm2(res - outer[0]);
}}
void minball(int n){ ball();
  if(nouter < 4) for(int i = 0 ; i < n ; i ++)
    if(norm2(res - pt[i]) - radius > EPS){
      outer[nouter ++] = pt[i]; minball(i); --nouter;
      if(i>0){ Pt Tt = pt[i];
        memmove(&pt[1], &pt[0], sizeof(Pt)*i);pt[0]=Tt;
}}}
ld solve(){
  // n points in pt
  random_shuffle(pt, pt+n); radius=-1;
  for(int i=0;i<n;i++) if(norm2(res-pt[i])-radius>EPS)
    nouter=1, outer[0]=pt[i], minball(i);
  return sqrt(radius);
}
```

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) $\Rightarrow O(nlgn)$
lca.qlca(u, v) $\Rightarrow O(lgn)$ u, v 的 LCA
lca.qdis(u, v) $\Rightarrow O(lgn)$ 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

```cpp
const int MXN = 5e5 + 5;
struct LCA{
  int n, lgn, ti = 0;
  int anc[MXN][24], in[MXN], out[MXN];
  ll ancw[MXN][24];
  vector<pll> g[MXN];
  void init(int _n) {
    n = _n, lgn = __lg(n) + 5;
    for(int i = 0; i < n; i++) g[i].clear(); }
  void addEdge(int u, int v, ll w = 1){
    g[u].PB(w, v), g[v].PB(w, u); }
  void build(int u, int f, ll w = 0) {
    in[u] = ti++;
    int cur = f;
    ll curw = w;
    for(int i = 0; i < lgn; ++i) {
      ancw[u][i] = curw, curw += ancw[cur][i];
      anc[u][i] = cur, cur = anc[cur][i]; }
    for(auto i : g[u]) if(i.Y != f) build(i.Y, u, i.X);
    out[u] = ti++; }
  bool isanc(int a, int u) {
    return in[a] <= in[u] && out[u] <= out[a]; }
  int qlca(int u, int v) {
    if(isanc(u, v)) return u;
    if(isanc(v, u)) return v;
    for(int i = lgn - 1; i >= 0; --i)
      if(!isanc(anc[u][i], v)) u = anc[u][i];
    return anc[u][0]; }
  ll qdis(int u, int v) {
    ll dis = 0;
    for(int i = lgn - 1; i >= 0; --i) {
      if(!isanc(anc[u][i], v)) {
        dis += ancw[u][i];
        u = anc[u][i]; }
      if(!isanc(anc[v][i], u)) {
        dis += ancw[v][i];
        v = anc[v][i]; } }
    if(!isanc(u, v)) dis += ancw[u][0];
    if(!isanc(v, u)) dis += ancw[v][0];
    return dis; } };
```

6 Graph

6.1 HeavyLightDecomposition *

```cpp
const int MXN = 200005;
template <typename T>
struct HeavyDecompose{ // 1-base, Need "ulimit -s
    unlimited"
  SegmentTree<T> st;
  vector<T> vec, tmp; // If tree point has weight
  vector<int> e[MXN];
  int sz[MXN], dep[MXN], fa[MXN], h[MXN];
  int cnt = 0, r = 0, n = 0;
  int root[MXN], id[MXN];
  void addEdge(int a, int b){
    e[a].emplace_back(b);
    e[b].emplace_back(a);
  }
  HeavyDecompose(int n, int r): n(n), r(r){
    vec.resize(n + 1); tmp.resize(n + 1);
  }
  void build(){
    dfs1(r, 0, 0);
    dfs2(r, r);
    st.init(tmp); // SegmentTree Need Add Method
  }
  void dfs1(int x, int f, int d){
    dep[x] = d, fa[x] = f, sz[x] = 1, h[x] = 0;
```

```
    for(int i : e[x]){
      if(i == f) continue;
      dfs1(i, x, d + 1);
      sz[x] += sz[i];
      if(sz[i] > sz[h[x]]) h[x] = i;
    }
  }
  void dfs2(int x, int f){
    id[x] = cnt++, root[x] = f, tmp[id[x]] = vec[x];
    if(!h[x]) return;
    dfs2(h[x], f);
    for(int i : e[x]){
      if(i == fa[x] || i == h[x]) continue;
      dfs2(i, i);
    }
  }
  void update(int x, int y, T v){
    while(root[x] != root[y]){
      if(dep[root[x]] < dep[root[y]]) swap(x, y);
      st.update(id[root[x]], id[x], v);
      x = fa[root[x]];
    }
    if(dep[x] > dep[y]) swap(x, y);
    st.update(id[x], id[y], v);
  }
  T query(int x, int y){
    T res = 0;
    while(root[x] != root[y]){
      if(dep[root[x]] < dep[root[y]]) swap(x, y);
      res = (st.query(id[root[x]], id[x]) + res) % MOD;
      x = fa[root[x]];
    }
    if(dep[x] > dep[y]) swap(x, y);
    res = (st.query(id[x], id[y]) + res) % MOD;
    return res;
  }
  void update(int x, T v){
    st.update(id[x], id[x] + sz[x] - 1, v);
  }
  T query(int x){
    return st.query(id[x], id[x] + sz[x] - 1);
  }
  int getLca(int x, int y){
    while(root[x] != root[y]){
      if(dep[root[x]] > dep[root[y]]) x = fa[root[x]];
      else y = fa[root[y]];
    }
    return dep[x] > dep[y] ? y : x;
  }
};
```

6.2 Centroid Decomposition *

```
struct CentroidDecomposition {
    int n;
    vector<vector<int>> G, out;
    vector<int> sz, v;
    CentroidDecomposition(int _n) : n(_n), G(_n), out(
        _n), sz(_n), v(_n) {}
    int dfs(int x, int par){
        sz[x] = 1;
        for (auto &&i : G[x]) {
            if(i == par || v[i]) continue;
            sz[x] += dfs(i, x);
        }
        return sz[x];
    }
    int search_centroid(int x, int p, const int mid){
        for (auto &&i : G[x]) {
            if(i == p || v[i]) continue;
            if(sz[i] > mid) return search_centroid(i, x
                , mid);
        }
        return x;
    }
    void add_edge(int l, int r){
        G[l].PB(r); G[r].PB(l);
    }
    int get(int x){
        int centroid = search_centroid(x, -1, dfs(x,
            -1)/2);
        v[centroid] = true;
        for (auto &&i : G[centroid]) {
            if(!v[i]) out[centroid].PB(get(i));
        }
        v[centroid] = false;
        return centroid;
} };
```

6.3 DominatorTree *

```
struct DominatorTree{ // O(N)
#define REP(i,s,e) for(int i=(s);i<=(e);i++)
#define REPD(i,s,e) for(int i=(s);i>=(e);i--)
  int n , m , s;
  vector< int > g[ MAXN ] , pred[ MAXN ];
  vector< int > cov[ MAXN ];
  int dfn[ MAXN ] , nfd[ MAXN ] , ts;
  int par[ MAXN ]; //idom[u] s到u的最後一個必經點
  int sdom[ MAXN ] , idom[ MAXN ];
  int mom[ MAXN ] , mn[ MAXN ];
  inline bool cmp( int u , int v )
  { return dfn[ u ] < dfn[ v ]; }
  int eval( int u ){
    if( mom[ u ] == u ) return u;
    int res = eval( mom[ u ] );
    if(cmp( sdom[ mn[ mom[ u ] ] ] , sdom[ mn[ u ] ] ))
      mn[ u ] = mn[ mom[ u ] ];
    return mom[ u ] = res;
  }
  void init( int _n , int _m , int _s ){
    ts = 0; n = _n; m = _m; s = _s;
    REP( i, 1, n ) g[ i ].clear(), pred[ i ].clear();
  }
  void addEdge( int u , int v ){
    g[ u ].push_back( v );
    pred[ v ].push_back( u );
  }
  void dfs( int u ){
    ts++;
    dfn[ u ] = ts;
    nfd[ ts ] = u;
    for( int v : g[ u ] ) if( dfn[ v ] == 0 ){
      par[ v ] = u;
      dfs( v );
  } }
  void build(){
    REP( i , 1 , n ){
      dfn[ i ] = nfd[ i ] = 0;
      cov[ i ].clear();
      mom[ i ] = mn[ i ] = sdom[ i ] = i;
    }
    dfs( s );
    REPD( i , n , 2 ){
      int u = nfd[ i ];
      if( u == 0 ) continue ;
      for( int v : pred[ u ] ) if( dfn[ v ] ){
        eval( v );
        if( cmp( sdom[ mn[ v ] ] , sdom[ u ] ) )
          sdom[ u ] = sdom[ mn[ v ] ];
      }
      cov[ sdom[ u ] ].push_back( u );
      mom[ u ] = par[ u ];
      for( int w : cov[ par[ u ] ] ){
        eval( w );
        if( cmp( sdom[ mn[ w ] ] , par[ u ] ) )
          idom[ w ] = mn[ w ];
        else idom[ w ] = par[ u ];
      }
      cov[ par[ u ] ].clear();
    }
    REP( i , 2 , n ){
      int u = nfd[ i ];
      if( u == 0 ) continue ;
      if( idom[ u ] != sdom[ u ] )
        idom[ u ] = idom[ idom[ u ] ];
} } }domT;
```

6.4 MaximumClique 最大團 *

```
#define N 111
struct MaxClique{ // 0-base
  typedef bitset<N> Int;
  Int linkto[N] , v[N];
```

```
  int n;
  void init(int _n){
    n = _n;
    for(int i = 0 ; i < n ; i ++){
      linkto[i].reset(); v[i].reset();
  } }
  void addEdge(int a , int b)
  { v[a][b] = v[b][a] = 1; }
  int popcount(const Int& val)
  { return val.count(); }
  int lowbit(const Int& val)
  { return val._Find_first(); }
  int ans , stk[N];
  int id[N] , di[N] , deg[N];
  Int cans;
  void maxclique(int elem_num, Int candi){
    if(elem_num > ans){
      ans = elem_num; cans.reset();
      for(int i = 0 ; i < elem_num ; i ++)
        cans[id[stk[i]]] = 1;
    }
    int potential = elem_num + popcount(candi);
    if(potential <= ans) return;
    int pivot = lowbit(candi);
    Int smaller_candi = candi & (~linkto[pivot]);
    while(smaller_candi.count() && potential > ans){
      int next = lowbit(smaller_candi);
      candi[next] = !candi[next];
      smaller_candi[next] = !smaller_candi[next];
      potential --;
      if(next == pivot || (smaller_candi & linkto[next
          ]).count()){
        stk[elem_num] = next;
        maxclique(elem_num + 1, candi & linkto[next]);
  } } }
  int solve(){
    for(int i = 0 ; i < n ; i ++){
      id[i] = i; deg[i] = v[i].count();
    }
    sort(id , id + n , [&](int id1, int id2){
          return deg[id1] > deg[id2]; });
    for(int i = 0 ; i < n ; i ++) di[id[i]] = i;
    for(int i = 0 ; i < n ; i ++)
      for(int j = 0 ; j < n ; j ++)
        if(v[i][j]) linkto[di[i]][di[j]] = 1;
    Int cand; cand.reset();
    for(int i = 0 ; i < n ; i ++) cand[i] = 1;
    ans = 1;
    cans.reset(); cans[0] = 1;
    maxclique(0, cand);
    return ans;
} }solver;
```

6.5 MaximalClique 極大團 *

```
#define N 80
struct MaxClique{ // 0-base
  typedef bitset<N> Int;
  Int lnk[N] , v[N];
  int n;
  void init(int _n){
    n = _n;
    for(int i = 0 ; i < n ; i ++){
      lnk[i].reset(); v[i].reset();
  } }
  void addEdge(int a , int b)
  { v[a][b] = v[b][a] = 1; }
  int ans , stk[N], id[N] , di[N] , deg[N];
  Int cans;
  void dfs(int elem_num, Int candi, Int ex){
    if(candi.none()&&ex.none()){
      cans.reset();
      for(int i = 0 ; i < elem_num ; i ++)
        cans[id[stk[i]]] = 1;
      ans = elem_num; // cans is a maximal clique
      return;
    }
    int pivot = (candi|ex)._Find_first();
    Int smaller_candi = candi & (~lnk[pivot]);
    while(smaller_candi.count()){
      int nxt = smaller_candi._Find_first();
      candi[nxt] = smaller_candi[nxt] = 0;
      ex[nxt] = 1;
      stk[elem_num] = nxt;
      dfs(elem_num+1,candi&lnk[nxt],ex&lnk[nxt]);
  } }
  int solve(){
    for(int i = 0 ; i < n ; i ++){
      id[i] = i; deg[i] = v[i].count();
    }
    sort(id , id + n , [&](int id1, int id2){
          return deg[id1] > deg[id2]; });
    for(int i = 0 ; i < n ; i ++) di[id[i]] = i;
    for(int i = 0 ; i < n ; i ++)
      for(int j = 0 ; j < n ; j ++)
        if(v[i][j]) lnk[di[i]][di[j]] = 1;
    ans = 1; cans.reset(); cans[0] = 1;
    dfs(0, Int(string(n,'1')), 0);
    return ans;
} }solver;
```

6.6 Minimum Steiner Tree

```
const int MXNN = 105;
const int MXNK = 10 + 1;
template<typename T>
struct SteinerTree{ // 有重要點的MST權重和, 1-base
  int n, k;
  T inf;
  vector<vector<T> > dp;
  vector<vector<pair<int, T> > > edge;
  priority_queue<pair<T, int>, vector<pair<T, int> >,
    greater<pair<T, int> > > pq;
  vector<int> vis;
  void init(int _n, int _k, T _inf){
    // n points, 1~k 是重要點, type T的INF
    n = _n, k = _k, inf = _inf;
    dp.assign(n + 1, vector<T>(1 << k, inf));
    edge.resize(n + 1); }
  void addEdge(int u, int v, T w){ // u <-(w)-> v
    edge[u].emplace_back(v, w);
    edge[v].emplace_back(u, w); }
  void dijkstra(int s, int cnt){
    vis.assign(n + 1, 0);
    while(!pq.empty()){
      auto [d, u] = pq.top(); pq.pop();
      if(vis[u]) continue;
      vis[u] = 1;
      for(auto &[v, w] : edge[u])
        // if(cnt > 1 && v <= k) continue;
        if(dp[v][s] > dp[u][s] + w){
          dp[v][s] = dp[u][s] + w;
          pq.push({dp[v][s], v}); } } }
  T run(){ // return total cost O(nk*2^k + n^2*2^k)
    for(int i = 1; i <= k; ++i)dp[i][1 << (i - 1)] = 0;
    for(int s = 1; s < (1 << k); ++s){
      int cnt = 0, tmp = s;
      while(tmp) cnt += (tmp & 1), tmp >>= 1;
      for(int i = k + 1; i <= n; ++i)
        for(int sb = s & (s-1); sb; sb = s & (sb-1))
          dp[i][s] =
            min(dp[i][s], dp[i][sb] + dp[i][s ^ sb]);
      for(int i = (cnt > 1 ? k + 1 : 1); i <= n; ++i)
        if(dp[i][s] != inf) pq.push({dp[i][s], i});
      dijkstra(s, cnt); }
    T res = inf;
    for(int i = 1; i <= n; ++i)
      res = min(res, dp[i][(1 << k) - 1]);
    return res; } };
```

6.7 BCC based on vertex

```
struct BccVertex { // 沒有橋的連通分量
  int n,nScc,step,dfn[MXN],low[MXN];
  vector<int> E[MXN],sccv[MXN];
  int top,stk[MXN];
  void init(int _n) { // 0-base
    n = _n; nScc = step = 0;
    for (int i=0; i<n; i++) E[i].clear();
  }
  void addEdge(int u, int v)
  { E[u].PB(v); E[v].PB(u); }
  void DFS(int u, int f) {
    dfn[u] = low[u] = step++;
```

```
    stk[top++] = u;
    for (auto v:E[u]) {
      if (v == f) continue;
      if (dfn[v] == -1) {
        DFS(v,u);
        low[u] = min(low[u], low[v]);
        if (low[v] >= dfn[u]) {
          int z;
          sccv[nScc].clear();
          do {
            z = stk[--top];
            sccv[nScc].PB(z);
          } while (z != v);
          sccv[nScc++].PB(u);
        }
      }else
        low[u] = min(low[u],dfn[v]);
  } }
  // 回傳分組結果，一個點出現在多個連通分量就是關節點
  // 一個連通分量只有兩個點就是橋
  vector<vector<int>> solve() {
    vector<vector<int>> res;
    for (int i=0; i<n; i++)
      dfn[i] = low[i] = -1;
    for (int i=0; i<n; i++)
      if (dfn[i] == -1) {
        top = 0;
        DFS(i,i);
      }
    for (int i = 0; i < nScc; ++i)
      res.PB(sccv[i]);
    return res;
  }
}graph;
```

6.8 Strongly Connected Component

```
struct Scc{ //O(V + E)
  int n, nScc, vst[MXN], bln[MXN];
  vector<int> E[MXN], rE[MXN], vec;
  void init(int _n){ // 0-base
    n = _n;
    for (int i=0; i<MXN; i++)
      E[i].clear(), rE[i].clear();
  }
  void addEdge(int u, int v){
    E[u].PB(v); rE[v].PB(u);
  }
  void DFS(int u){
    vst[u]=1;
    for (auto v : E[u]) if (!vst[v]) DFS(v);
    vec.PB(u);
  }
  void rDFS(int u){
    vst[u] = 1; bln[u] = nScc;
    for (auto v : rE[u]) if (!vst[v]) rDFS(v);
  }
  void solve(){
    nScc = 0;
    vec.clear();
    MEM(vst, 0);
    for (int i=0; i<n; i++)
      if (!vst[i]) DFS(i);
    reverse(vec.begin(),vec.end());
    MEM(vst, 0);
    for (auto v : vec)
      if (!vst[v]){
        DEBUG(v, nScc);
        rDFS(v); nScc++;
      }
  }
};
```

6.9 尤拉路徑

```
尤拉路徑: 所有邊恰好經過一次
無向圖: 最多只有兩個度數為奇數的點
有向圖: 只有一個出度-入度=1(起點)，反之亦然，其餘都是差
  0
有解的話，隨便dfs都可以
```

6.10 差分約束 *

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

7 String

7.1 PalTree *

```
// len[s]是對應的回文長度
// num[s]是有幾個回文後綴
// cnt[s]是這個回文子字串在整個字串中的出現次數
// fail[s]是他長度次長的回文後綴，aba的fail是a
const int MXN = 1000010;
struct PalT{
  int nxt[MXN][26],fail[MXN],len[MXN];
  int tot,lst,n,state[MXN],cnt[MXN],num[MXN];
  int diff[MXN],sfail[MXN],fac[MXN],dp[MXN];
  char s[MXN]={-1};
  int newNode(int l,int f){
    len[tot]=l,fail[tot]=f,cnt[tot]=num[tot]=0;
    memset(nxt[tot],0,sizeof(nxt[tot]));
    diff[tot]=(l>0?l-len[f]:0);
    sfail[tot]=(l>0&&diff[tot]==diff[f]?sfail[f]:f);
    return tot++;
  }
  int getfail(int x){
    while(s[n-len[x]-1]!=s[n]) x=fail[x];
    return x;
  }
  int getmin(int v){
    dp[v]=fac[n-len[sfail[v]]-diff[v]];
    if(diff[v]==diff[fail[v]])
        dp[v]=min(dp[v],dp[fail[v]]);
    return dp[v]+1;
  }
  int push(){
    int c=s[n]-'a',np=getfail(lst);
    if(!(lst=nxt[np][c])){
      lst=newNode(len[np]+2,nxt[getfail(fail[np])][c]);
      nxt[np][c]=lst; num[lst]=num[fail[lst]]+1;
    }
    fac[n]=n;
    for(int v=lst;len[v]>0;v=sfail[v])
        fac[n]=min(fac[n],getmin(v));
    return ++cnt[lst],lst;
  }
  void init(const char *_s){
    tot=lst=n=0;
    newNode(0,1),newNode(-1,1);
    for(;_s[n];) s[n+1]=_s[n],++n,state[n-1]=push();
    for(int i=tot-1;i>1;i--) cnt[fail[i]]+=cnt[i];
  }
}palt;
```

7.2 SuffixArray

```
const int MXN = 1e6;
// sa[i]: idx of ith rank, rk[i]: rank of idx
// he[i]: sa[i], sa[i - 1] 前he[i]個字元相同
int ct[MXN], he[MXN], rk[MXN];
int sa[MXN], tsa[MXN], tp[MXN][2];
void suffix_array(string ip){ // 0-base
  int len = ip.size();
  int alp = 256;
  MEM(ct, 0);
  for(int i = 0; i < len; i++) ct[ip[i] + 1]++;
  for(int i = 1; i < alp; i++) ct[i] +=ct[i - 1];
  for(int i = 0; i < len; i++) rk[i] = ct[ip[i]];
  for(int i = 1; i < len; i *= 2){
    for(int j = 0; j < len; j++){
      if(j + i >= len) tp[j][1] = 0;
      else tp[j][1] = rk[j + i] + 1;
      tp[j][0] = rk[j];
    }
    memset(ct, 0, sizeof(ct));
    for(int j = 0; j < len; j++) ct[tp[j][1] + 1]++;
    for(int j = 1; j < len+2; j++) ct[j] += ct[j - 1];
    for(int j = 0; j < len; j++) tsa[ct[tp[j][1]]++]=j;
    memset(ct, 0, sizeof(ct));
    for(int j = 0; j < len; j++) ct[tp[j][0] + 1]++;
    for(int j = 1; j < len+1; j++) ct[j] += ct[j - 1];
    for(int j = 0; j < len; j++)
      sa[ct[tp[tsa[j]][0]]++] = tsa[j];
```

```
    rk[sa[0]] = 0;
    for(int j = 1; j < len; j++){
      if( tp[sa[j]][0] == tp[sa[j - 1]][0] &&
          tp[sa[j]][1] == tp[sa[j - 1]][1] )
        rk[sa[j]] = rk[sa[j - 1]];
      else
        rk[sa[j]] = j; } }
  for(int i = 0, h = 0; i < len; i++){
    if(rk[i] == 0) h = 0;
    else{
      int j = sa[rk[i] - 1];
      h = max(0, h - 1);
      for(; ip[i + h] == ip[j + h]; h++); }
    he[rk[i]] = h; } }
```

7.3 MinRoation *

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

7.4 RollingHash

```
struct RollingHash { // 0-base, need MOD
  const int p1 = 44129; // 65537, 40961, 90001, 971651
  int n; vector<ll> pre, ppow;
  void init(string s) { // O(n)
    n = s.size();
    pre.resize(n + 1); ppow.resize(n + 1);
    pre[0] = 0, ppow[0] = 1;
    for (int i = 0; i < n; i++)
      pre[i + 1] = (pre[i] * p1 + s[i]) % MOD,
      ppow[i + 1] = ppow[i] * p1 % MOD;
  }
  ll query(int l, int r) { // [l, r], O(1)
    ll ret = pre[r + 1] - pre[l] * ppow[r - l + 1];
    return (ret % MOD + MOD) % MOD; } };
```

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

```
const int MXN = 2e7 + 5;
int fail[MXN]; vector<int> mi;
void kmp(string &t, string &p){ // O(n), 0-base
  // pattern match in target, idx store in mi
  mi.clear();
  if (p.size() > t.size()) return;
  for (int i = 1, j = fail[0] = -1; i < p.size(); ++i){
    while (j >= 0 && p[j + 1] != p[i]) j = fail[j];
    if (p[j + 1] == p[i]) j++;
    fail[i] = j; }
  for (int i = 0, j = -1; i < t.size(); ++i){
    while (j >= 0 && p[j + 1] != t[i]) j = fail[j];
    if (p[j + 1] == t[i]) j++;
    if (j == p.size() - 1)
      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

```
int lis(vector<ll> &v){ // O(nlgn)
  vector<ll> p;
  for(int i = 0; i < v.size(); ++i)
    if(p.empty() || p.back() < v[i]) p.PB(v[i]);
    else *lower_bound(p.begin(), p.end(), v[i]) = v[i];
  return p.size(); }

int lcs(string s, string t){ // O(nlgn)
  map<char, vector<int> > mp;
  for(int i = 0; i < s.size(); ++i) mp[s[i]].PB(i);
  vector<int> p;
  for(int i = 0; i < t.size(); ++i){
    auto &v = mp[t[i]];
    for(int j = v.size() - 1; j >= 0; --j)
      if(p.empty() || p.back() < v[j]) p.PB(v[j]);
      else *lower_bound(p.begin(),p.end(), v[j])=v[j];}
  return p.size(); }
```

7.7 Aho-Corasick *

```
struct ACautomata{
  struct Node{
    int cnt,i;
    Node *go[26], *fail, *dic;
    Node (){
      cnt = 0; fail = 0; dic = 0; i = 0;
      memset(go,0,sizeof(go));
    }
  }pool[1048576],*root;
  int nMem,n_pattern;
  Node* new_Node(){
    pool[nMem] = Node();
    return &pool[nMem++];
  }
  void init() {
    nMem=0;root=new_Node();n_pattern=0;
    add("");
  }
  void add(const string &str) { insert(root,str,0); }
  void insert(Node *cur, const string &str, int pos){
    for(int i=pos;i<str.size();i++){
      if(!cur->go[str[i]-'a'])
        cur->go[str[i]-'a'] = new_Node();
      cur=cur->go[str[i]-'a'];
    }
    cur->cnt++; cur->i=n_pattern++;
  }
  void make_fail(){
    queue<Node*> que;
    que.push(root);
    while (!que.empty()){
      Node* fr=que.front(); que.pop();
      for (int i=0; i<26; i++){
        if (fr->go[i]){
          Node *ptr = fr->fail;
          while (ptr && !ptr->go[i]) ptr = ptr->fail;
          fr->go[i]->fail=ptr=(ptr?ptr->go[i]:root);
          fr->go[i]->dic=(ptr->cnt?ptr:ptr->dic);
          que.push(fr->go[i]);
  } } } }
  void query(string s){
      Node *cur=root;
      for(int i=0;i<(int)s.size();i++){
        while(cur&&!cur->go[s[i]-'a']) cur=cur->fail;
        cur=(cur?cur->go[s[i]-'a']:root);
        if(cur->i>=0) ans[cur->i]++;
        for(Node *tmp=cur->dic;tmp;tmp=tmp->dic)
          ans[tmp->i]++;
  } }// ans[i] : number of occurrence of pattern i
}AC;
```

7.8 Z Value *

```
int z[MAXN];
void Z_value(const string& s) { //z[i] = lcp(s[1...],s[
    i...])
  int i, j, left, right, len = s.size();
  left=right=0; z[0]=len;
  for(i=1;i<len;i++) {
    j=max(min(z[i-left],right-i),0);
    for(;i+j<len&&s[i+j]==s[j];j++);
    z[i]=j;
    if(i+z[i]>right) {
      right=i+z[i];
      left=i;
}   }   }
```

7.9 manacher

```
const int MXN = 1e7 + 5;
struct Manacher{ // 0-base 每個點為中心的最長回文長度
  string st; int p[MXN * 2];
  void init(string s){ // O(n)
```

```
    MEM(p, 0); st.clear();
    st.push_back('$'); st.push_back('#');
    for(int i = 0; i < s.size(); ++i)
      st.push_back(s[i]), st.push_back('#');
    st.push_back('*');
    int mx = 0, id = 0;
    for(int i = 1; i < st.size(); ++i){
      p[i] = mx>i ? min(p[(id << 1) - i], mx - i) : 1;
      while(st[i + p[i]] == st[i - p[i]]) p[i]++;
      if(i + p[i] > mx) mx = i + p[i], id = i; } }
  // bt=1: middle between mid, mid+1
  int query(int mid, bool bt = 0) {
    return p[mid * 2 + 2 + bt] - 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

```
struct Treap{
  Treap *l, *r;
  int pri, sz, rev;
  ll val, sum;
  Treap(int _val): l(0), r(0),
    pri(rand()), sz(1), rev(0),
    val(_val), sum(_val){} };

ll gsz(Treap *x){ return x ? x->sz : 0; }
ll gsum(Treap *x){ return x ? x->sum : 0; }

Treap* pull(Treap *x){
  x->sz = gsz(x->l) + gsz(x->r) + 1;
  x->sum = x->val + gsum(x->l) + gsum(x->r);
  return x; }
void push(Treap *x){
  if(x->rev){
    swap(x->l, x->r);
    if(x->l) x->l->rev ^= 1;
    if(x->r) x->r->rev ^= 1;
    x->rev = 0; } }

Treap* merge(Treap* a, Treap* b){
  if(!a || !b) return a ? a : b;
  push(a), push(b);
  if(a->pri > b->pri){
    a->r = merge(a->r, b);
    return pull(a); }
  else{
    b->l = merge(a, b->l);
    return pull(b); } }

void split(Treap *x, int k, Treap *&a, Treap *&b){
  if(!x) a = b = 0;
  else{
    push(x);
    if(x->val <= k) a = x, split(x->r, k, a->r, b);
    else            b = x, split(x->l, k, a, b->l);
    pull(x); } }

void kth(Treap *x, int k, Treap *&a, Treap *&b){
  if(!x) a = b = 0;
  else{
    push(x);
    if(gsz(x->l) < k)
         a = x, kth(x->r, k - gsz(x->l) - 1, a->r, b);
    else b = x, kth(x->l, k, a, b->l);
    pull(x); } }
```

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

```
const int MXN = 1e6+5;
struct BIT{
  ll n, a[MXN];
  void init(int _n){ n = _n; MEM(a, 0); }
  void add(int i, int x){
    for(; i <= n; i += i & -i) a[i] += x; }
  int sum(int i){
    int ret = 0;
    for(; i > 0; i -= i & -i) ret += a[i];
    return ret; }
  int kth(int k){
    int res = 0;
    for(int i = 1 << __lg(n); i > 0; i >>= 1)
      if(res + i <= n && a[res+i] < k) k -= a[res+=i];
    return res; } };
```

8.3 二維偏序 *

```
struct Node {
  int x, y, id;
  bool operator < (const Node &b) const {
    if(x == b.x) return y < b.y;
    return x < b.x;}};
struct TDPO {
  vector<Node> p; vector<ll> ans;
  void init(vector<Node> _p) {
    p = _p; bit.init(MXN);
    ans.resize(p.size());
    sort(p.begin(), p.end());}
  void bulid() {
    int sz = p.size();
    for(int i = 0; i < sz; ++i) {
      ans[p[i].id] = bit.sum(p[i].y - 1);
      bit.add(p[i].y, 1);}}}};
```

8.4 三維偏序

```
struct Node {
  int x, y, z;
  int ans, id;
};

bool cmp1(const Node &a, const Node &b) {
  if(a.x != b.x) return a.x < b.x;
  if(a.y != b.y) return a.y < b.y;
  return a.z < b.z;
}

bool cmp2(const Node &a, const Node &b) {
  if(a.y != b.y) return a.y < b.y;
  if(a.z != b.z) return a.z < b.z;
  return a.x < b.x;
}

void cdq(int l, int r) {
  if(l == r) return;
  int mid = (l + r) >> 1, target = 0;
  for(int i = l; i < r; ++i) {
    if(vec[i].x != vec[i + 1].x) {
      if(abs(i - mid) < abs(target - mid)) target = i;
    }
  }
  mid = target;
  cdq(l, mid);
  cdq(mid + 1, r);
  sort(vec.begin() + l, vec.begin() + mid + 1, cmp2);
  sort(vec.begin() + mid + 1, vec.begin() + r + 1, cmp2
      );

  int p = l;
  for(int i = mid + 1; i <= r; ++i) {
    while(p <= mid && vec[p].y < vec[i].y) {
      bit.add(vec[p].z, 1);
      p++;
    }
    vec[i].ans += bit.sum(vec[i].z - 1);
  }

  for(int i = l; i < p; ++i) bit.add(vec[i].z, -1);
}
```

8.5 持久化 *

```
struct Seg {
```

```cpp
  // Persistent Segment Tree, single point modify,
      range query sum
  // 0-indexed, [l, r)
  static Seg mem[M], *pt;
  int l, r, m, val;
  Seg* ch[2];
  Seg () = default;
  Seg (int _l, int _r) : l(_l), r(_r), m(l + r >> 1),
      val(0) {
    if (r - l > 1) {
      ch[0] = new (pt++) Seg(l, m);
      ch[1] = new (pt++) Seg(m, r);
    }
  }
  void pull() {val = ch[0]->val + ch[1]->val;}
  Seg* modify(int p, int v) {
    Seg *now = new (pt++) Seg(*this);
    if (r - l == 1) {
      now->val = v;
    } else {
      now->ch[p >= m] = ch[p >= m]->modify(p, v);
      now->pull();
    }
    return now;
  }
  int query(int a, int b) {
    if (a <= l && r <= b) return val;
    int ans = 0;
    if (a < m) ans += ch[0]->query(a, b);
    if (m < b) ans += ch[1]->query(a, b);
    return ans;
  }
} Seg::mem[M], *Seg::pt = mem;
// Init Tree
Seg *root = new (Seg::pt++) Seg(0, n);
```

8.6 2D 線段樹

```cpp
// 2D range add, range sum in log^2
struct seg {
  int l, r;
  ll sum, lz;
  seg *ch[2]{};
  seg(int _l, int _r) : l(_l), r(_r), sum(0), lz(0) {}
  void push() {
    if (lz) ch[0]->add(l, r, lz), ch[1]->modify(l, r,
        lz), lz = 0;
  }
  void pull() {sum = ch[0]->sum + ch[1]->sum;}
  void add(int _l, int _r, ll d) {
    if (_l <= l && r <= _r) {
      sum += d * (r - l);
      lz += d;
      return;
    }
    if (!ch[0]) ch[0] = new seg(l, l + r >> 1), ch[1] =
        new seg(l + r >> 1, r);
    push();
    if (_l < l + r >> 1) ch[0]->add(_l, _r, d);
    if (l + r >> 1 < _r) ch[1]->add(_l, _r, d);
    pull();
  }
  ll qsum(int _l, int _r) {
    if (_l <= l && r <= _r) return sum;
    if (!ch[0]) return lz * (min(r, _r) - max(l, _l));
    push();
    ll res = 0;
    if (_l < l + r >> 1) res += ch[0]->qsum(_l, _r);
    if (l + r >> 1 < _r) res += ch[1]->qsum(_l, _r);
    return res;
  }
};
struct seg2 {
  int l, r;
  seg v, lz;
  seg2 *ch[2]{};
  seg2(int _l, int _r) : l(_l), r(_r), v(0, N), lz(0, N
      ) {
    if (l < r - 1) ch[0] = new seg2(l, l + r >> 1), ch
        [1] = new seg2(l + r >> 1, r);
  }
  void add(int _l, int _r, int _l2, int _r2, ll d) {
    v.add(_l2, _r2, d * (min(r, _r) - max(l, _l)));
    if (_l <= l && r <= _r) {
      lz.add(_l2, _r2, d);
      return;
    }
    if (_l < l + r >> 1) ch[0]->add(_l, _r, _l2, _r2, d
        );
    if (l + r >> 1 < _r) ch[1]->add(_l, _r, _l2, _r2, d
        );
  }
  ll qsum(int _l, int _r, int _l2, int _r2) {
    ll res = v.qsum(_l2, _r2);
    if (_l <= l && r <= _r) return res;
    res += lz.qsum(_l2, _r2) * (min(r, _r) - max(l, _l)
        );
    if (_l < l + r >> 1) res += ch[0]->query(_l, _r,
        _l2, _r2);
    if (l + r >> 1 < _r) res += ch[1]->query(_l, _r,
        _l2, _r2);
    return res;
  }
};
```

8.7 Disjoint Set

```cpp
struct DisjointSet {
  int fa[MXN], h[MXN], top;
  struct Node {
    int x, y, fa, h;
    Node(int _x = 0, int _y = 0, int _fa = 0, int _h=0)
        : x(_x), y(_y), fa(_fa), h(_h) {}
  } stk[MXN];
  void init(int n) {
    top = 0;
    for (int i = 1; i <= n; i++) fa[i] = i, h[i] = 0; }
  int find(int x){return x == fa[x] ? x : find(fa[x]);}
  void merge(int u, int v) {
    int x = find(u), y = find(v);
    if (h[x] > h[y]) swap(x, y);
    stk[top++] = Node(x, y, fa[x], h[y]);
    if (h[x] == h[y]) h[y]++;
    fa[x] = y; }
  void undo(int k=1) {  //undo k times
    for (int i = 0; i < k; i++) {
      Node &it = stk[--top];
      fa[it.x] = it.fa;
      h[it.y] = it.h; } } }djs;
```

8.8 Black Magic

```cpp
#include <bits/extc++.h>
using namespace __gnu_pbds;
typedef tree<int,null_type,less<int>,rb_tree_tag,
    tree_order_statistics_node_update> set_t;
#include <ext/pb_ds/assoc_container.hpp>
typedef cc_hash_table<int,int> umap_t;
typedef priority_queue<int> heap;
#include<ext/rope>
using namespace __gnu_cxx;
int main(){
  // Insert some entries into s.
  set_t s; s.insert(12); s.insert(505);
  // The order of the keys should be: 12, 505.
  assert(*s.find_by_order(0) == 12);
  assert(*s.find_by_order(3) == 505);
  // The order of the keys should be: 12, 505.
  assert(s.order_of_key(12) == 0);
  assert(s.order_of_key(505) == 1);
  // Erase an entry.
  s.erase(12);
  // The order of the keys should be: 505.
  assert(*s.find_by_order(0) == 505);
  // The order of the keys should be: 505.
  assert(s.order_of_key(505) == 0);

  heap h1 , h2; h1.join( h2 );

  rope<char> r[ 2 ];
  r[ 1 ] = r[ 0 ]; // persistenet
  string t = "abc";
  r[ 1 ].insert( 0 , t.c_str() );
  r[ 1 ].erase( 1 , 1 );
```

```
  cout << r[ 1 ].substr( 0 , 2 );
}
```

9 DP

9.1 DP Method

有向圖求合法路徑方法數

1. $f_k(i,j)$ 表示從 i 到 j 恰好 k 步的方法數

$f_k(i,j) = \sum_{x=1}^{n} f_{k-1}(i,x) * a(x,j)$

2. $S_k(i,j)$ 表示從 i 到 j 不超過 k 步的方法數

$S_k(i,j) = \sum_{k=1}^{K} f_k(i,j)$

多人背包

要求好幾個人的背包結果 (第 k 優解背包問題)

dp[i][j] 代表體積為 i 的第 k 優解

分組背包

當有分組問題，如買 A 物品前要先買 B 物品。

dp[i] = max(dp[i], dp[i - B - A] + val[B] + val[A])

多重背包

當每種物品為有限個時，求最大價值。

dp[i][j] = max(dp[i][j], dp[i - 1][j - k * w[i]] + k * v[i])

需要轉換成單調對列優化。

$d = j \ mod \ w[i]$， $s = \lfloor j/w[i] \rfloor$

dp[i] = max(dp[d + w[i] * k] - v[i] * k) + v * s

樹上背包

dp(u, i, j) 代表 u 根節點，遍歷 i 個子節點，且體積為 j 的最大價值。

dp(u, i, j) = max(dp(u, i - 1, j - k) + dp(v, s, k))

(s 為 v 子樹的節點數)

數位 DP

1. 要求統計滿足一定條件的數的數量 (即，最終目的為計數)

2. 這些條件經過轉化後可以使用「數位」的思想去理解和判斷

3. 輸入會提供一個數字區間 (有時也只提供上界) 來作為統計的限制

4. 上界很大 (比如 10^{18})，暴力枚舉驗證會超時。

dp[位數][限制 1][限制 2]...

dfs 從高到低

區間 DP

合併：即將兩個或多個部分進行整合，當然也可以反過來

特徵：能將問題分解為能兩兩合併的形式

求解：對整個問題設最優值，枚舉合併點，將問題分解為左右兩個部分，最後合併兩個部分的最優值得到原問題的最優值

dp[i][j] = min(dp[i][j], dp[i][k] + dp[k + 1][j] + cost)

SOS DP

= 子集和 DP

DP[mask] = $\sum_{i \in mask} A[i]$

9.2 Bag Problem

```
// 多人背包
for(int i = 1; i <= n; ++i) {
  for(int j = V; j >= v[i]; --j) {
    int c1 = 1, c2 = 2;
    for(int k = 1; k <= K; ++k) {
      if(dp[j][c1] > dp[j - v[i]][c2] + w[i])
        now[k] = f[j][c1], c1++;
      else
        now[k] = f[j - v[i]][c2] + w[i], c2++;
    }
    for(int k = 1; k <= K; ++k) f[j][k] = now[k];
  }
}

// 多重背包
for(int k = 0; k <= K; ++k) {
  while(!dq.empty() &&
    dq.front().first <= dp[d + k * w] - v * k) dq.
        pop_back();
  dq.push_back({dp[d + k * w] - v * k, k});
  while(!dq.empty() && dq.back().second > s) dq.
      pop_front();
  dp[d + k * w] = dq.front().first + v * k;
}
```

9.3 Matrix

```
struct Matrix{
  ll v[MXN][MXN]; int n;
  void init(int n): n(n){ MEM(v, 0); }
  Matrix operator*(const Matrix &rhs){
    Matrix z; z.init(n);
    for(int k = 0; k < n; ++k) for(int i = 0; i < n; ++
        i)
    for(int j = 0; j < n; ++j)
      (z.v[i][j] += v[i][k] * rhs.v[k][j] % MOD) %= MOD
          ;
    return z;
  }
};

Matrix operator^(Matrix m, ll a){
  Matrix ret; ret.init(m.n);
  for(int i = 0; i < m.n; ++i) ret.v[i][i] = 1;
  while(a){
    if(a & 1) ret = (ret * m);
    m = m * m;
    a >>= 1;
  }
  return ret;
}
```

9.4 SOS dp *

```
for(int i = 0; i<(1<<N); ++i)
  F[i] = A[i];
for(int i = 0;i < N; ++i) for(int mask = 0; mask < (1<<
    N); ++mask){
  if(mask & (1<<i))
    F[mask] += F[mask^(1<<i)];
}
```

10 Others

10.1 MO's Algorithm *

```
struct MoSolver {
  struct query {
    int l, r, id;
    bool operator < (const query &o) {
      if (l / C == o.l / C) return (l / C) & 1 ? r > o.
          r : r < o.r;
      return l / C < o.l / C;
    }
  };
  int cur_ans;
  vector <int> ans;
  void add(int x) {
    // do something
  }
  void sub(int x) {
    // do something
  }
  vector <query> Q;
  void add_query(int l, int r, int id) {
    // [l, r)
    Q.push_back({l, r, id});
    ans.push_back(0);
  }
  void run() {
    sort(Q.begin(), Q.end());
    int pl = 0, pr = 0;
    cur_ans = 0;
    for (query &i : Q) {
```

```
      while (pl > i.l)
        add(a[--pl]);
      while (pr < i.r)
        add(a[pr++]);
      while (pl < i.l)
        sub(a[pl++]);
      while (pr > i.r)
        sub(a[--pr]);
      ans[i.id] = cur;
    }
  }
};
```