Contents

1 basic

1.1 default

```cpp
#include <bits/stdc++.h>
using namespace std;
#define masterspark ios::sync_with_stdio(0), cin.tie(0)
    ,cout.tie(0),cin.exceptions(cin.failbit);

#define int long long
#define pp pair<int, int>
#define ff first
#define ss second

#define forr(i,n) for(int i = 1; i <= n;++i)
#define rep(i,j,n) for(int i = j; i < n;++i)
#define PB push_back
#define PF push_front
#define EB emplace_back
#define all(v) (v).begin(), (v).end()
#define FZ(x) memset(x, 0, sizeof(x)) //fill zero
#define SZ(x) ((int)x.size())
bool chmin(auto &a, auto b) { return (b < a) and (a = b
    , true); }
bool chmax(auto &a, auto b) { return (a < b) and (a = b
    , true); }
using i128 = __int128_t;
using i64 = __int64_t;
using i32 = __int32_t;

void solve(){

}
signed main()
{
    masterspark
    int t = 1;
    // freopen("stdin","r",stdin);
    // freopen("stdout","w",stdout);
    // cin >> t;
    while(t--){
        solve();
    }
    return 0;
}
```

1.2 godcode

```cpp
#pragma GCC optimize("O3,unroll-loops")
#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")
編譯指令: g++ -std=c++20 -w -Wfatal-errors -Wall -
    Wshadow -fsanitize=undefined

mt19937 gen(chrono::steady_clock::now().
    time_since_epoch().count());
int randint(int lb, int ub)
{ return uniform_int_distribution<int>(lb, ub)(gen); }

#define SECs ((double)clock() / CLOCKS_PER_SEC)
```

```
struct KeyHasher {
  size_t operator()(const Key& k) const {
    return k.first + k.second * 100000;
} };
typedef unordered_map<Key,int,KeyHasher> map_t;

__builtin_popcountll    // 二進位有幾個1 (記得這是long long)
__builtin_clzll         // 左起第一個1之前0的個數
__builtin_parityll      // 1的個數的奇偶性
__builtin_mul_overflow(a,b,&h) // a*b是否溢位,h = a * b;
__builtin_add_overflow(a,b,&h)
```

1.3 random

```
mt19937 mt(chrono::steady_clock::now().time_since_epoch().count());
//mt19937_64 mt() -> return randnum
int randint(int l, int r){
    uniform_int_distribution<> dis(l, r); return dis(mt);
}
```

1.4 run.bat

```
@echo off
g++ ac.cpp -o ac.exe
g++ wa.cpp -o wa.exe
set /a num=1
:loop
   echo %num%
   python gen.py > input
   ac.exe < input > ac
   wa.exe < input > wa
   fc ac wa
   set /a num=num+1
if not errorlevel 1 goto loop
```

1.5 run.sh

```
set -e
for ((i=0;;i++))
do
    echo "$i"
    python gen.py > in
    ./ac < in > ac.out
    ./wa < in > wa.out
    diff ac.out wa.out || break
done
```

2 binarysearch

2.1 二分搜

```
int bsearch_1(int l, int r)
{
    while (l < r)
    {
        int mid = l + r >> 1;
        if (check(mid)) r = mid;
        else l = mid + 1;
    }
    return l;
}
// ..............oooooooooo

int bsearch_2(int l, int r)
{
    while (l < r)
    {
        int mid = l + r + 1 >> 1;
        if (check(mid)) l = mid;
        else r = mid - 1;
    }
    return l;
}
// oooooooooo..................

int m = *ranges::partition_point(views::iota(0LL,(int)1e9+9),[&](int a){
    return check(a) > k;
    });
//[begin,last)
//111111100000000000
//搜左邊數過來第一個 0
//都是 1 會回傳 last

int partitionpoint(int L,int R,function<bool(int)> chk){
  int l = L,r = R-1;
  while(r - l > 10){
    int m = l + (r-l)/2;
    if(chk(m)) l = m;
    else r = m;
  }
    int m = l;
    while(m <= r){
        if(!chk(m)) break;
        ++m;
    }
  if(!chk(m)) return m;
  else return R;
}

//手工
```

2.2 三分搜

```
int l = 1,r = 100;
while(l < r) {
    int lmid = l + (r - l) / 3; // l + 1/3區間大小
    int rmid = r - (r - l) / 3;  // r - 1/3區間大小
    lans = cal(lmid),rans = cal(rmid);
    // 求凹函數的極小值
    if(lans <= rans) r = rmid - 1;
    else l = lmid + 1;
}
```

3 dataStructure

3.1 DSU

```
struct DSU {
    vector<i32> f, sz;
    DSU(i32 n) : f(n), sz(n) {
        for (i32 i = 0; i < n; i++) {
            f[i] = i;
            sz[i] = 1;
        }
    }
    i32 find(i32 x) {
        if (x == f[x]) return x;
        f[x] = find(f[x]);
        return f[x];
    }
    void merge(i32 x, i32 y) {
        x = find(x), y = find(y);
        if (x == y) return;
        if (sz[x] < sz[y])
            swap(x, y);
        sz[x] += sz[y];
        f[y] = x;
    }
    bool same(i32 a, i32 b) { return (find(a) == find(b)); }
};
```

3.2 fenwickTree

```
// [0, n] -> fenwick(n)
struct fenwick {
  #define lowbit(x) (x & -x)
  i32 n; // [0, n)
  vector<i64> v;
    fenwick() {}
  fenwick(i32 _n) : n(_n + 1), v(_n + 2, 0) {}
  void _add(i32 x, i64 u){
    for(;x <= n; x += lowbit(x)) v[x] += u;
  }
  i64 _qry(i32 x){
    i64 ret = 0;
```

```
    for(; x ; x -= lowbit(x)) ret += v[x];
    return ret;
  }
    i32 _lowerbound(i64 k) {
        i64 sum = 0;
        i32 p = 0;
        for (i32 i = (1 << __lg(n)); i; i >>= 1) {
            i32 nxt = p + i;
            if (nxt <= n && sum + v[nxt] < k) {
                sum += v[nxt];
                p = nxt;
            }
        }
        if (p + 1 > n) return -1;
        return p;
    }
    void add(i32 x, i64 v) { _add(x + 1, v); }
    i64 qry(i32 x) { return _qry(x + 1); }
  i64 qry(i32 l,i32 r) { return qry(r) - qry(l - 1); }
    i32 lower_bound(i64 k) { return _lowerbound(k); }
};
```

3.3 segmentTree1

```
// [l, r)
template<class Info>
struct SegmentTree {
    inline i32 cl(i32 x) { return x << 1; }
    inline i32 cr(i32 x) { return (x << 1) | 1; }
    i32 n;
    vector<Info> info;
    SegmentTree() : n(0) {}
    SegmentTree(i32 n_, Info v_ = Info()) { init(n_, v_
        ); }
    template<class T>
    SegmentTree(vector<T> init_) { init(init_); }

    void init(i32 n_, Info v_ = Info()) { init(vector(
        n_, v_)); }
    template<class T>
    void init(vector<T> init_) {
        n = init_.size();
        info.assign(4 << __lg(n), Info());
        function<void(i32, i32, i32)> build = [&](i32 p
            , i32 l, i32 r) {
            if (r - l == 1) {
                info[p] = init_[l];
                return;
            }
            i32 m = (l + r) >> 1;
            build(cl(p), l, m);
            build(cr(p), m, r);
            pull(p);
        };
        build(1, 0, n);
    }

    void pull(i32 p) { info[p] = merge(info[cl(p)],
        info[cr(p)]); }

    void modify(i32 p, i32 l, i32 r, i32 x, const Info
        &v) {
        if (r - l == 1) {
            info[p] = v;
            return;
        }
        i32 m = (l + r) >> 1;
        if (x < m) modify(cl(p), l, m, x, v);
        else modify(cr(p), m, r, x, v);
        pull(p);
    }
    void modify(i32 p, const Info &v) { modify(1, 0, n,
         p, v); }

    Info rangeQuery(i32 p, i32 l, i32 r, i32 x, i32 y)
        {
        if (l >= y || r <= x) return Info();
        if (l >= x && r <= y) return info[p];
        i32 m = (l + r) >> 1;
        return merge(rangeQuery(cl(p), l, m, x, y),
            rangeQuery(cr(p), m, r, x, y));
    }
    Info rangeQuery(i32 l, i32 r) { return rangeQuery
        (1, 0, n, l, r); }

    template<class F>
    i32 findFirst(i32 p, i32 l, i32 r, i32 x, i32 y, F
        &&pred) {
        if (l >= y || r <= x) return -1;
        if (l >= x && r <= y && !pred(info[p])) return
            -1;
        if (r - l == 1) return l;
        i32 m = (l + r) >> 1;
        i32 res = findFirst(cl(p), l, m, x, y, pred);
        if (res == -1) res = findFirst(cr(p), m, r, x,
            y, pred);
        return res;
    }
    template<class F>
    i32 findFirst(i32 l, i32 r, F &&pred) { return
        findFirst(1, 0, n, l, r, pred); }

    template<class F>
    i32 findLast(i32 p, i32 l, i32 r, i32 x, i32 y, F
        &&pred) {
        if (l >= y || r <= x) return -1;
        if (l >= x && r <= y && !pred(info[p])) return
            -1;
        if (r - l == 1) return l;
        i32 m = (l + r) >> 1;
        i32 res = findLast(cr(p), m, r, x, y, pred);
        if (res == -1) res = findLast(cl(p), l, m, x, y
            , pred);
        return res;
    }
    template<class F>
    i32 findLast(i32 l, i32 r, F &&pred) { return
        findLast(1, 0, n, l, r, pred); }
};
```

3.4 segmentTree2

```
// [l, r)
template<class Info, class Tag>
struct segTree {
    inline i32 cl(i32 x) { return x << 1; }
    inline i32 cr(i32 x) { return (x << 1) | 1; }
    i32 n;
    vector<Info> info;
    vector<Tag> tag;
    segTree(): n(0) {}
    segTree(i32 n_, Info v_ = Info()) {
        init(n_, v_);
    }
    template<class T>
    segTree(vector<T> init_) {
        init(init_);
    }
    void init(i32 n_, Info v_ = Info()) {
        init(vector(n_, v_));
    }
    template<class T>
    void init(vector<T> init_) {
        n = init_.size();
        info.assign(4 << __lg(n), Info());
        tag.assign(4 << __lg(n), Tag());
        function<void(i32, i32, i32)> build = [&](i32 p
            , i32 l, i32 r) {
            if (r - l == 1) {
                info[p] = init_[l];
                return;
            }
            i32 m = (l + r) >> 1;
            build(cl(p), l, m);
            build(cr(p), m, r);
            pull(p, l, r);
        };
        build(1, 0, n);
    }
    void pull(i32 p, i32 l, i32 r) {
        i32 m = (l + r) >> 1;
        push(cl(p), l, m);
        push(cr(p), m, r);
        info[p] = merge(info[cl(p)], info[cr(p)]);
```

```
    }
    void rangeModify(i32 p, i32 l, i32 r, i32 x, i32 y,
         const Tag &v) {
        push(p, l, r);
        if (l >= x && r <= y) {
            tag[p] += v;
            return;
        }
        i32 m = (l + r) >> 1;
        if (x < m) rangeModify(cl(p), l, m, x, y, v);
        if (y > m) rangeModify(cr(p), m, r, x, y, v);
        pull(p, l, r);
    }
    Info rangeQuery(i32 p, i32 l, i32 r, i32 x, i32 y)
        {
        push(p, l, r);
        if (l >= y || r <= x) {
            return Info();
        }
        if (l >= x && r <= y) {
            return info[p];
        }
        i32 m = (l + r)>> 1;
        return merge(rangeQuery(cl(p), l, m, x, y),
            rangeQuery(cr(p), m, r, x, y));
    }
    Info rangeQuery(i32 l, i32 r) { return rangeQuery
        (1, 0, n, l, r); }
    void rangeModify(i32 l, i32 r, const Tag &v) {
        rangeModify(1, 0, n, l, r, v); }
    void push(i32 p, i32 l, i32 r) { // need compelete
        if (tag[p].add != 0) {
            info[p].v += tag[p].add * (r - l);
            if (r - l != 1) {
                tag[cl(p)].add += tag[p].add;
                tag[cr(p)].add += tag[p].add;
            }
            tag[p].add = 0;
        }
    }
};
```

3.5 persistantSegTree

```
struct pSeg{
    struct node{
        int v;
        node *l,*r;
    };
    int n;
    vector<node*> ver;
    node* build(int l,int r){
        node* x = new node();
        if(l == r){
            x->v = 0;
            return x;
        }
        int m = (l+r)/2;
        x->l = build(l,m);
        x->r = build(m+1,r);
        x->v = x->l->v + x->r->v;
        return x;
    }
    void init(int _n){
        n = _n;
        ver.PB(build(0,n-1));
    }
    int qry(node* now,int l,int r,int ql,int qr){
        if(ql <= l && r <= qr){
            return now->v;
        }
        int m = (l+r)/2,ret = 0;
        if(ql <= m)ret += qry(now->l,l,m,ql,qr);
        if(qr > m )ret += qry(now->r,m+1,r,ql,qr);
        return ret;
    }
    node* upd(node* prv,int l,int r,int p,int v){
        node* x = new node();
        if(l == r){
            x->v = prv->v + v; //累加
            return x;
        }
        int m = (l+r)/2;
        if(p <= m) {
            x->l = upd(prv->l,l,m,p,v);
            x->r = prv->r;
        }else{
            x->l = prv->l;
            x->r = upd(prv->r,m+1,r,p,v);
        }
        x->v = x->l->v + x->r->v;
        return x;
    }
    void addver(int p,int v){
        ver.PB(upd(ver.back(),0,n-1,p,v));
    }
    //(a,b] kth //用segTree統計出現次數 //版本當區間 //
        第 i 個版本為前 區間 [0,i] 有統計
    int qurey(node* a,node* b,int l,int r,int k){
        if(l == r) return l;
        int m = (l+r)/2;
        int num = b->l->v - a->l->v;
        if(num >= k) return qurey(a->l,b->l,l,m,k);//
            左邊大往左搜
        else return qurey(a->r,b->r,m+1,r,k-num);
    }
};
```

3.6 countMinimumSeg

```
//count zeros on segmentTree
struct segTree{
    #define cl (i<<1)
    #define cr ((i<<1)+1)
    pp seg[MXN*4];
    int tag[MXN*4];
    pp comb(pp a,pp b){
        if(a.ff < b.ff) return a;
        if(a.ff > b.ff) return b;
        return pp{a.ff,a.ss+b.ss};
    }
    void push(int i,int l,int r){
        if(tag[i]){
            seg[i].ff += tag[i];
            if(r - l > 1){
                tag[cl] += tag[i];
                tag[cr] += tag[i];
            }
            tag[i] = 0;
        }
    }
    void pull(int i,int l,int r){
        int m = (r-l)/2 + l;
        push(cl,l,m);
        push(cr,m,r);
        seg[i] = comb(seg[cl],seg[cr]);
    }
    void build(int i,int l,int r){
        if(r - l <= 1){
            seg[i] = pp{0,1};
            return;
        }
        int m = (r-l)/2 + l;
        build(cl,l,m);
        build(cr,m,r);
        pull(i,l,r);
    }
    void upd(int i,int l,int r,int ql,int qr,int x){
        push(i,l,r);
        if(ql <= l && r <= qr){
            tag[i] += x;
            return;
        }
        int m = (r-l)/2 + l;
        if(ql < m) upd(cl,l,m,ql,qr,x);
        if(qr > m) upd(cr,m,r,ql,qr,x);
        pull(i,l,r);
    }
    int qry(){
        //count zero
        if(seg[1].ff == 0) return seg[1].ss;
        return 0;
    }
    void upd(int l,int r,int x){
```

```
        upd(1,0,MXN,l,r,x);
    }
}st;
```

3.7 LiChaoSegTree

```
const int inf = numeric_limits<i64>::max()/2;
struct Line {
    // y = ax + b
    i64 a{0}, b{-inf};
    i64 operator()(i64 x) {
        return a * x + b;
    }
};

struct Seg{
    int l, r;
    Seg *ls{},*rs{};
    Line f{};
    Seg(int l, int r) : l(l), r(r) {}
    void add(Line g){
        int m = (l+r)/2;
        if (g(m) > f(m)) swap(g, f);
        if(g.b == -inf || r - l == 1) return;
        if(g.a < f.a){
            if(!ls) ls = new Seg(l,m);
            ls->add(g);
        }else{
            if(!rs) rs = new Seg(m,r);
            rs->add(g);
        }
    }
    i64 qry(i64 x){
        int m = (l+r) / 2;
        i64 y = f(x);
        if(x <  m && ls) y = max({y,ls->qry(x)});
        if(x >= m && rs) y = max({y,rs->qry(x)});
        return y;
    }
};
auto add = [&](Line g,int ql,int qr){ //新增線段 [ql,qr
    )
    auto find = [&](auto &&self,Seg * now,int l,int r)
        -> void {
        if(ql <= l && r <= qr){
            now->add(g);
            return;
        }
        int m = (l+r) / 2;
        if(ql < m) {
            if(!now->ls) now->ls = new Seg(l,m);
            self(self,now->ls,l,m);
        }
        if(qr > m){
            if(!now->rs) now->rs = new Seg(m,r);
            self(self,now->rs,m,r);
        }
    };
    find(find,st,-ninf,ninf);
};
//Seg *st = new Seg(-ninf,ninf); // [l,r)
```

3.8 2Dbit

```
struct fenwick{
    #define lowbit(x) (x&-x)
    int n,m;
    vector<vector<int>> v;
    fenwick(int _n,int _m) : n(_n+1),m(_m+1),v(_n+2,
        vector<int>(_m+2,0)){}
    void add(int x,int y,int u){
        ++x,++y;
        for(;x < n; x += lowbit(x)){
            for(int j = y;j < m; j += lowbit(j)) v[x][j
                ] += u;
        }
    }
    int qry(int x,int y){
        ++x,++y;
        int ret = 0;
        for(; x ; x -= lowbit(x)){
            for(int j = y; j; j -= lowbit(j)) ret += v[
                x][j];
        }
        return ret;
    }
    //(l,u) <= (r,d)
    //d -   +
    //u +   -
    //  l   r
    void add(int l,int u,int r,int d,int x){
        ++r,++d;
        add(l,u,x);
        add(l,d,-x);
        add(r,u,-x);
        add(r,d,x);
    }
    int qry(int l,int u,int r,int d){
        --l,--u;
        return qry(r,d) - qry(r,u) - qry(l,d) + qry(l,u
            );
    }
};
```

4 dp

4.1 digit

```
ll dp[MXN_BIT][PRE_NUM][LIMIT][F0];//字串位置，根據題目
    的值，是否上界，前導0
ll dfs(int i,int pre, bool lim, bool f0, const string&
    str){
    if(v[i][pre][f0][lim])  return dp[i][pre][f0][lim];
    v[i][pre][f0][lim] = true;

    if(i == str.size())
        return dp[i][pre][f0][lim] = 1;

    ll ret = 0, h = lim ? str[i] : '9';

    for(int j='0'; j<=h; j++){
        if(abs(j-pre)>=2 || f0){
            ret += dfs(i+1, j, j==h && lim, f0 && j=='0
                ', str);
    }}
    return dp[i][pre][f0][lim] = ret;
}
```

4.2 p_median

```
void p_Median(){
    for (int i=1; i<=N; ++i)
        for (int j=i; j<=N; ++j){
            m = (i+j)/2,d[i][j] = 0;        // m是中位
                數，d[i][j]為距離的總和
            for (int k=i; k<=j; ++k) d[i][j] += abs(arr
                [k] - arr[m]);
        }
    for (int p=1; p<=P; ++p)
        for (int n=1; n<=N; ++n){
            dp[p][n] = 1e9;
            for (int k=p; k<=n; ++k)
                if (dp[p-1][k-1] + d[k][n] < dp[p][n]){
                    dp[p][n] = dp[p-1][k-1] + d[k][n];
                    r[p][n] = k;    // 從第k個位置往右
                        到第j個位置
                }
        }
}
```

4.3 sosdp

```
// 求子集和 或超集和 -> !(mask & (1 << i))
for(int i = 0; i<(1<<N); ++i) F[i] = A[i]; //預處理 狀
    態權重

for(int i = 0;i < N; ++i)
for (int s = 0; s < (1<<N) ; ++s)
  if (s & (1 << i))
    F[s] += F[s ^ (1 << i)];

//窮舉子集合
for(int s = mask; s ; s = (s-1)&mask;)
```

4.4 MinimumSteinerTree

```
int dp[MXN][(1<<11)],vis[MXN];
//dp[i][S] -> 選了前K個點 以第i個點為第K+1個點的 生成
    (1..K+1)的最小生成樹
rep(s,0,(1<<K)) forr(i,N) dp[i][s] = INF;
  rep(j,0,K) dp[j+1][(1<<j)] = 0;
  rep(s,0,(1<<K)){
    forr(i,N){
      for(int a = s; a ; a=(a-1)&s)
      dp[i][s] = min(dp[i][s],dp[i][s^a] + dp[i][a]);
          // node
    }
    FZ(vis);
    priority_queue<pp,vector<pp>,greater<pp>> Q;
    forr(i,N) Q.emplace(dp[i][s],i);
    while(Q.size()){
      auto [d,u] = Q.top();Q.pop();
      if(vis[u]) continue;
      vis[u] = 1;
      for(auto [v,w]:E[u]){
        if(dp[u][s]+w < dp[v][s]) {
          dp[v][s] = dp[u][s]+w;
          Q.emplace(dp[v][s],v);
        }
      }
    }
  }
rep(i,K+1,N+1) cout << dp[i][(1<<K)-1] <<'\n';
```

4.5 lowConvexHull

```
struct Line {
  mutable ll m, b, p;
  bool operator<(const Line& o) const { return m < o.m;
       }
  bool operator<(ll x) const { return p < x; }
};

struct LineContainer : multiset<Line, less<>> {
  // (for doubles, use inf = 1/.0, div(a,b) = a/b)
  const ll inf = LLONG_MAX;
  ll div(ll a, ll b) { // floored division
    return a / b - ((a ^ b) < 0 && a % b); }
  bool isect(iterator x, iterator y) {
    if (y == end()) { x->p = inf; return false; }
    if (x->m == y->m) x->p = x->b > y->b ? inf : -inf;
    else x->p = div(y->b - x->b, x->m - y->m);
    return x->p >= y->p;
  }
  void insert_line(ll m, ll b) {
    auto z = insert({m, b, 0}), y = z++, x = y;
    while (isect(y, z)) z = erase(z);
    if (x != begin() && isect(--x, y)) isect(x, y =
        erase(y));
    while ((y = x) != begin() && (--x)->p >= y->p)
      isect(x, erase(y));
  }
  ll eval(ll x) {
    assert(!empty());
    auto l = *lower_bound(x);
    return l.m * x + l.b;
  }
};
```

5 flow

5.1 Dinic

```
struct Dinic{
  struct Edge{ int v,f,re; };
  int n,s,t,level[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 add_edge(int u, int v, int f){
    E[u].PB({v,f,SZ(E[v])});
    E[v].PB({u,0,SZ(E[u])-1});
  }
  bool BFS(){
    for (int i=0; i<n; i++) level[i] = -1;
    queue<int> que;
    que.push(s);
    level[s] = 0;
    while (!que.empty()){
      int u = que.front(); que.pop();
      for (auto it : E[u]){
        if (it.f > 0 && level[it.v] == -1){
          level[it.v] = level[u]+1;
          que.push(it.v);
    } } }
    return level[t] != -1;
  }
  int DFS(int u, int nf){
    if (u == t) return nf;
    int res = 0;
    for (auto &it : E[u]){
      if (it.f > 0 && level[it.v] == level[u]+1){
        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) return res;
    } }
    if (!res) level[u] = -1;
    return res;
  }
  int flow(int res=0){
    while ( BFS() )
      res += DFS(s,2147483647);
    return res;
} }flow;
```

5.2 isap

```
struct Maxflow {
  static const int MAXV = 20010;
  static const int INF  = 1000000;
  struct Edge {
    int v, c, r;
    Edge(int _v, int _c, int _r):
      v(_v), c(_c), r(_r) {}
  };
  int s, t;
  vector<Edge> G[MAXV*2];
  int iter[MAXV*2], d[MAXV*2], gap[MAXV*2], tot;
  void init(int x) {
    tot = x+2;
    s = x+1, t = x+2;
    for(int i = 0; i <= tot; i++) {
      G[i].clear();
      iter[i] = d[i] = gap[i] = 0;
  } }
  void addEdge(int u, int v, int c) {
    G[u].push_back(Edge(v, c, SZ(G[v]) ));
    G[v].push_back(Edge(u, 0, SZ(G[u]) - 1));
  }
  int dfs(int p, int flow) {
    if(p == t) return flow;
    for(int &i = iter[p]; i < SZ(G[p]); i++) {
      Edge &e = G[p][i];
      if(e.c > 0 && d[p] == d[e.v]+1) {
        int f = dfs(e.v, min(flow, e.c));
        if(f) {
          e.c -= f;
          G[e.v][e.r].c += f;
          return f;
    } } }
    if( (--gap[d[p]]) == 0) d[s] = tot;
    else {
      d[p]++;
      iter[p] = 0;
      ++gap[d[p]];
    }
    return 0;
  }
  int solve() {
    int res = 0;
    gap[0] = tot;
    for(res = 0; d[s] < tot; res += dfs(s, INF));
    return res;
  }
  void reset() {
```

```
    for(int i=0;i<=tot;i++) {
      iter[i]=d[i]=gap[i]=0;
} } }flow;
```

5.3 KM

```
struct KM{ // max weight, for min negate the weights
  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) { // 1-based, N個節點
    n = _n;
    for(int i=1; i<=n; i++) fill(g[i], g[i]+n+1, 0);
  }
  void addEdge(int x, int y, ll w) {g[x][y] = w;} //左
      邊的集合節點x連邊右邊集合節點y權重為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]=INF, vx[i]=vy[i]=0;
    queue<int> q; q.push(st);
    for(;;) {
      while(q.size()) {
        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 = INF;
      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 solve(){ // 回傳值為完美匹配下的最大總權重
    fill(mx, mx+n+1, 0); fill(my, my+n+1, 0);
    fill(ly, ly+n+1, 0); fill(lx, lx+n+1, -INF);
    for(int x=1; x<=n; ++x) for(int y=1; y<=n; ++y) //
        1-base
      lx[x] = max(lx[x], g[x][y]);
    for(int x=1; x<=n; ++x) bfs(x);
    ll ans = 0;
    for(int y=1; y<=n; ++y) ans += g[my[y]][y];
    return ans;
} }graph;
```

5.4 匈牙利

```
struct hungarian {
    int L, R;
    vector<bool> vis;
    vector<int> match;
    vector<vector<int>> E;
    hungarian(int l, int r): //左邊有幾個，右邊有幾個
    L(l), R(r),
    vis(l+r+1),
    match(l+r+1,-1),
    E(l+r+1){}
    void add_edge(int l, int r){//左側第幾個(1-base) ,
        右側第幾個(1-base)
        r = L+r;
        E[l].push_back(r);
        E[r].push_back(l);
    }
    bool dfs(int u){
        for(int i : E[u]){
            if(vis[i]) continue; // 有連通且未拜訪
            vis[i] = true; // 紀錄是否走過
            if(match[i] == -1 || dfs(match[i])){
                match[i] = u; match[u] = i; // 紀錄匹配
                return true;
            }
        }
        return false;
    }
    int solve(){
        int ans = 0;
        for(int i = 1; i <= L; i++){
            fill(all(vis),0);
            if(dfs(i)) ans++;
        }
        return ans;
    }
};
```

5.5 對偶建圖

```
auto add = [&](int u,int v,int w){
    E[u].EB(v,w);
    E[v].EB(u,w);
};
//A : 橫槓(n*(m-1)); B : 直槓((n-1)*m); C : 斜槓((n-1)
    *(m-1));
//n 列 m 行平面圖 (1-base) S起點 (左上) T 終點 (右下)
forr(s,(n-1)){
    int M = (m-1)*2;
    forr(i,M){
        int id = i + (s-1)*M;
        if(i&1){
            int u = (s < n-1) ? ((i+1) + s*M) : T;
            int e = (i > 1) ? id - 1 : T;
            add(id,e,B[s-1][(i-1)/2]);
            add(id,u,A[s][(i-1)/2]);
        }else{
            if(i == M) add(id,S,B[s-1][m-1]);
            if(s == 1) add(id,S,A[s-1][i/2-1]);
            int w = C[s-1][i/2-1];
            add(id,id-1,w);
        }
    }
}
```

5.6 最小花費最大流 dijkstra 不能負值

```
struct MinCostMaxFlow{
typedef int Tcost;
  static const int MAXV = 20010;
  static const int INFf = 1000000;
  static const Tcost INFc  = 1e9;
  struct Edge{
    int v, cap;
    Tcost w;
    int rev;
    Edge(){}
    Edge(int t2, int t3, Tcost t4, int t5)
    : v(t2), cap(t3), w(t4), rev(t5) {}
  };
  int V, s, t;
  vector<Edge> g[MAXV];
  void init(int n, int _s, int _t){
    V = n; s = _s; t = _t;
    for(int i = 0; i <= V; i++) g[i].clear();
  }
  void addEdge(int a, int b, int cap, Tcost w){
    g[a].push_back(Edge(b, cap, w, (int)g[b].size()));
    g[b].push_back(Edge(a, 0, -w, (int)g[a].size()-1));
  }
  Tcost d[MAXV];
  int id[MAXV], mom[MAXV];
  bool inqu[MAXV];
  queue<int> q;
  pair<int,Tcost> solve(){
    int mxf = 0; Tcost mnc = 0;
    while(1){
      fill(d, d+1+V, INFc);
      fill(inqu, inqu+1+V, 0);
      fill(mom, mom+1+V, -1);
      mom[s] = s;
      d[s] = 0;
      q.push(s); inqu[s] = 1;
      while(q.size()){
```

```
        int u = q.front(); q.pop();
        inqu[u] = 0;
        for(int i = 0; i < (int) g[u].size(); i++){
          Edge &e = g[u][i];
          int v = e.v;
          if(e.cap > 0 && d[v] > d[u]+e.w){
            d[v] = d[u]+e.w;
            mom[v] = u;
            id[v] = i;
            if(!inqu[v]) q.push(v), inqu[v] = 1;
      } } }
      if(mom[t] == -1) break ;
      int df = INFf;
      for(int u = t; u != s; u = mom[u])
        df = min(df, g[mom[u]][id[u]].cap);
      for(int u = t; u != s; u = mom[u]){
        Edge &e = g[mom[u]][id[u]];
        e.cap              -= df;
        g[e.v][e.rev].cap += df;
      }
      mxf += df;
      mnc += df*d[t];
    }
    return {mxf,mnc};
} }flow;
```

5.7 最小花費最大流 SPFA

```
struct zkwflow{
  static const int maxN=10000;
  struct Edge{ int v,f,re; ll w;};
  int n,s,t,ptr[maxN]; bool vis[maxN]; ll dis[maxN];
  vector<Edge> E[maxN];
  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,(int)E[v].size(),w});
    E[v].push_back({u,0,(int)E[u].size()-1,-w});
  }
  bool SPFA(){
    fill_n(dis,n,LLONG_MAX); fill_n(vis,n,false);
    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]=true; q.push(it.v);
    } } } }
    return dis[t]!=LLONG_MAX;
  }
  int DFS(int u,int nf){
    if(u==t) return nf;
    int res=0; vis[u]=true;
    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()){
      fill_n(ptr,n,0);
      int f=DFS(s,INT_MAX);
      flow+=f; cost+=dis[t]*f;
    }
    return{ flow,cost };
  } // reset: do nothing
} flow;
```

6 geometry

6.1 Point

```
using ld = long double;
template<class T>
struct pt{
  T x,y;
  pt(T _x,T _y):x(_x),y(_y){}
  pt():x(0),y(0){}

  pt operator * (T  c){ return pt(x*c,y*c);}
  pt operator / (T  c){ return pt(x/c,y/c);}
  pt operator + (pt a){ return pt(x+a.x,y+a.y);}
  pt operator - (pt a){ return pt(x-a.x,y-a.y);}
  T  operator * (pt a){ return x*a.x + y*a.y;}
  T  operator ^ (pt a){ return x*a.y - y*a.x;}

  auto operator<=>(pt o) const { return (x != o.x) ? x
      <=> o.x : y <=> o.y; } // c++20
  bool operator < (pt a) const { return x < a.x || (x
      == a.x && y < a.y);};
  bool operator== (pt a) const { return x == a.x and y
      == a.y;};
  friend T ori(pt a, pt b, pt c) { return (b - a) ^ (c
      - a); }
  friend T abs2(pt a) { return a * a; }
};
using numbers::pi; // c++20
const ld pi = acos(-1);
const ld eps = 1e-8L;
using Pt = pt<ld>;
int sgn(ld x) { return (x > -eps) - (x < eps); } //
    dcmp == sgn
ld abs(Pt a) { return sqrt(abs2(a)); }
ld arg(Pt x) { return atan2(x.y, x.x); }
bool argcmp(Pt a, Pt b) { // arg(a) < arg(b)
    int f = (Pt{a.y, -a.x} > Pt{} ? 1 : -1) * (a != Pt
        {});
    int g = (Pt{b.y, -b.x} > Pt{} ? 1 : -1) * (b != Pt
        {});
    return f == g ? (a ^ b) > 0 : f < g;
}
Pt unit(Pt x) { return x / abs(x); }
Pt rotate(Pt u) { // pi / 2
    return {-u.y, u.x};
}
Pt rotate(Pt u, ld a) {
    Pt v{sin(a), cos(a)};
    return {u ^ v, u * v};
}


istream &operator>>(istream &s, Pt &a) { return s >> a.
    x >> a.y; }
ostream &operator<<(ostream &s, Pt &a) { return s << "(
    " << a.x << ", " << a.y << ")";}

bool collinearity(Pt a, Pt b, Pt c) {  // 三點共線
    return ((b - a) ^ (c - a)) == 0;
}
```

6.2 Line

```
struct Line {
    Pt a, b;
    Pt dir() const { return b - a; }
};
int PtSide(Pt p, Line L) {
    // return sgn(ori(L.a, L.b, p) / abs(L.a - L.b));
    return sgn(ori(L.a, L.b, p));
}
bool PtOnSeg(Pt p, Line L) {
    return PtSide(p, L) == 0 and sgn((p - L.a) * (p - L
        .b)) <= 0;
}
Pt proj(Pt p, Line l) {
    Pt dir = unit(l.b - l.a);
    return l.a + dir * (dir * (p - l.a));
}
```

6.3 Circle

```
struct Cir {
    Pt o;
    ld r;
```

```
};
bool disjunct(const Cir &a, const Cir &b) { //  分開 || 外切 -> 1
    return sgn(abs(a.o - b.o) - a.r - b.r) >= 0;
}
bool contain(const Cir &a, const Cir &b) {
    return sgn(a.r - b.r - abs(a.o - b.o)) >= 0;
}
```

6.4 圓多邊形面積

```
double CirclePoly(Cir C, const vector<Pt> &P) {
    auto arg = [&](Pt p, Pt q) { return atan2(p ^ q, p * q); };
    double r2 = C.r * C.r / 2;
    auto tri = [&](Pt p, Pt q) {
        Pt d = q - p;
        auto a = (d * p) / abs2(d), b = (abs2(p) - C.r * C.r)/ abs2(d);
        auto det = a * a - b;
        if (det <= 0) return arg(p, q) * r2;
        auto s = max(0., -a - sqrt(det)), t = min(1., -a + sqrt(det));
        if (t < 0 or 1 <= s) return arg(p, q) * r2;
        Pt u = p + d * s, v = p + d * t;
        return arg(p, u) * r2 + (u ^ v) / 2 + arg(v, q) * r2;
    };
    double sum = 0.0;
    for (int i = 0; i < P.size(); i++)
    sum += tri(P[i] - C.o, P[(i + 1) % P.size()] - C.o);
    return sum;
}
```

6.5 圓三角形面積

```
double CircleTriangle(Pt a, Pt b, double r) {
    if (sgn(abs(a) - r) <= 0 and sgn(abs(b) - r) <= 0) {
        return abs(a ^ b) / 2;
    }
    if (abs(a) > abs(b)) swap(a, b);
    auto I = CircleLineInter({{}, r}, {a, b});
    erase_if(I, [&](Pt x) { return !PtOnSeg(x, {a, b}); });
    if (I.size() == 1) return abs(a ^ I[0]) / 2 + SectorArea(I[0], b, r);
    if (I.size() == 2) {
        return SectorArea(a, I[0], r) + SectorArea(I[1], b, r) + abs(I[0] ^ I[1]) / 2;
    }
    return SectorArea(a, b, r);
}
```

6.6 半平面交

```
bool cover(Line L, Line P, Line Q) {
    // PtSide(LineInter(P, Q), L) <= 0 or P, Q parallel
    i128 u = (Q.a - P.a) ^ Q.dir();
    i128 v = P.dir() ^ Q.dir();
    i128 x = P.dir().x * u + (P.a - L.a).x * v;
    i128 y = P.dir().y * u + (P.a - L.a).y * v;
    return sgn(x * L.dir().y - y * L.dir().x) * sgn(v) >= 0;
}
vector<Line> HPI(vector<Line> P) {
    // line P.a -> P.b 的逆時針是半平面
    sort(all(P), [&](Line l, Line m) {
        if (argcmp(l.dir(), m.dir())) return true;
        if (argcmp(m.dir(), l.dir())) return false;
        return ori(m.a, m.b, l.a) > 0;
    });
    int n = P.size(), l = 0, r = -1;
    for (int i = 0; i < n; i++) {
        if (i and !argcmp(P[i - 1].dir(), P[i].dir()))
            continue;
        while (l < r and cover(P[i], P[r - 1], P[r])) r--;
        while (l < r and cover(P[i], P[l], P[l + 1])) l++;
        P[++r] = P[i];
    }
    while (l < r and cover(P[l], P[r - 1], P[r])) r--;
    while (l < r and cover(P[r], P[l], P[l + 1])) l++;
    if (r - l <= 1 or !argcmp(P[l].dir(), P[r].dir()))
        return {}; // empty
    if (cover(P[l + 1], P[l], P[r]))
        return {}; // infinity
    return vector(P.begin() + l, P.begin() + r + 1);
}
```

6.7 圓線交

```
vector<Pt> CircleLineInter(Cir c, Line l) {
    Pt H = proj(c.o, l);
    Pt dir = unit(l.b - l.a);
    double h = abs(H - c.o);
    if (sgn(h - c.r) > 0) return {};
    double d = sqrt(max((double)0., c.r * c.r - h * h));
    if (sgn(d) == 0) return {H};
    return {H - dir *d, H + dir * d};
    // Counterclockwise
}
```

6.8 圓圓交

```
vector<Pt> CircleInter(Cir a, Cir b) {
    double d2 = abs2(a.o - b.o), d = sqrt(d2);
    if (d < max(a.r, b.r) - min(a.r, b.r) || d > a.r + b.r) return {};
    Pt u = (a.o + b.o) / 2 + (a.o - b.o) * ((b.r * b.r - a.r * a.r) / (2 * d2));
    double A = sqrt((a.r + b.r + d) * (a.r - b.r + d) * (a.r + b.r - d) * (-a.r + b.r + d));
    Pt v = rotate(b.o - a.o) * A / (2 * d2);
    if (sgn(v.x) == 0 and sgn(v.y) == 0) return {u};
    return {u - v, u + v}; // counter clockwise of a
}
```

6.9 線線交

```
bool isInter(Line l, Line m) {
    if (PtOnSeg(m.a, l) or PtOnSeg(m.b, l) or
        PtOnSeg(l.a, m) or PtOnSeg(l.b, m))
        return true;
    return PtSide(m.a, l) * PtSide(m.b, l) < 0 and
           PtSide(l.a, m) * PtSide(l.b, m) < 0;
}
Pt LineInter(Line l, Line m) {
    double s = ori(m.a, m.b, l.a), t = ori(m.a, m.b, l.b);
    return (l.b * s - l.a * t) / (s - t);
}
```

6.10 ConvexHull

```
vector<Pt> Hull(vector<Pt> P) {
    sort(all(P));
    P.erase(unique(all(P)), P.end());
    P.insert(P.end(), P.rbegin() + 1, P.rend());
    vector<Pt> stk;
    for (auto p : P) {
        auto it = stk.rbegin();
        while (stk.rend() - it >= 2 and \
            ori(*next(it), *it, p) <= 0 and \
            (*next(it) < *it) == (*it < p)) {
            it++;
        }
        stk.resize(stk.rend() - it);
        stk.push_back(p);
    }
    stk.pop_back();
    return stk;
}
```

6.11 Hulltrick

```
struct Convex {
    int n;
    vector<Pt> A, V, L, U;
    Convex(const vector<Pt> &_A) : A(_A), n(_A.size()) { // n >= 3
```

```
        auto it = max_element(all(A));
        L.assign(A.begin(), it + 1);
        U.assign(it, A.end()), U.push_back(A[0]);
        for (int i = 0; i < n; i++) {
            V.push_back(A[(i + 1) % n] - A[i]);
        }
    }
    int inside(Pt p, const vector<Pt> &h, auto f) {
        auto it = lower_bound(all(h), p, f);
        if (it == h.end()) return 0;
        if (it == h.begin()) return p == *it;
        return 1 - sgn(ori(*prev(it), p, *it));
    }
    // 0: out, 1: on, 2: in
    int inside(Pt p) {
        return min(inside(p, L, less{}), inside(p, U,
            greater{}));
    }
    static bool cmp(Pt a, Pt b) { return sgn(a ^ b) >
        0; }
    // A[i] is a far/closer tangent point
    int tangent(Pt v, bool close = true) {
        assert(v != Pt{});
        auto l = V.begin(), r = V.begin() + L.size() -
            1;
        if (v < Pt{}) l = r, r = V.end();
        if (close) return (lower_bound(l, r, v, cmp) -
            V.begin()) % n;
        return (upper_bound(l, r, v, cmp) - V.begin())
            % n;
    }
    // closer tangent point array[0] -> array[1] 順時針
    array<int, 2> tangent2(Pt p) {
        array<int, 2> t{-1, -1};
        if (inside(p) == 2) return t;
        if (auto it = lower_bound(all(L), p); it != L.
            end() and p == *it) {
            int s = it - L.begin();
            return {(s + 1) % n, (s - 1 + n) % n};
        }
        if (auto it = lower_bound(all(U), p, greater{})
            ; it != U.end() and p == *it) {
            int s = it - U.begin() + L.size() - 1;
            return {(s + 1) % n, (s - 1 + n) % n};
        }
        for (int i = 0; i != t[0]; i = tangent((A[t[0]
            = i] - p), 0));
        for (int i = 0; i != t[1]; i = tangent((p - A[t
            [1] = i]), 1));
        return t;
    }
    int find(int l, int r, Line L) {
        if (r < l) r += n;
        int s = PtSide(A[l % n], L);
        return *ranges::partition_point(views::iota(l,
            r),
            [&](int m) {
                return PtSide(A[m % n], L) == s;
            }) - 1;
    };
    // Line A_x A_x+1 interset with L
    vector<int> intersect(Line L) {
        int l = tangent(L.a - L.b), r = tangent(L.b - L
            .a);
        if (PtSide(A[l], L) * PtSide(A[r], L) >= 0)
            return {};
        return {find(l, r, L) % n, find(r, l, L) % n};
    }
};
```

6.12 點線距

```
double PtSegDist(Pt p, Line l) {
    double ans = min(abs(p - l.a), abs(p - l.b));
    if (sgn(abs(l.a - l.b)) == 0)  return ans;
    if (sgn((l.a - l.b) * (p - l.b)) < 0) return ans;
    if (sgn((l.b - l.a) * (p - l.a)) < 0) return ans;
    return min(ans, abs(ori(p, l.a, l.b)) / abs(l.a - l
        .b));
}
double SegDist(Line l, Line m) {
    return PtSegDist({0, 0}, {l.a - m.a, l.b - m.b});
}
```

6.13 MEC

```
Pt Center(Pt a, Pt b, Pt c) {
    Pt x = (a + b) / 2;
    Pt y = (b + c) / 2;
    return LineInter({x, x + rotate(b - a)}, {y, y +
        rotate(c - b)});
}
Cir MEC(vector<Pt> P) {
    mt19937 rng(time(0));
    shuffle(all(P), rng);
    Cir C = {P[0], 0.0};
    for (int i = 0; i < P.size(); i++) {
        if (C.inside(P[i])) continue;
        C = {P[i], 0};
        for (int j = 0; j < i; j++) {
            if (C.inside(P[j])) continue;
            C = {(P[i] + P[j]) / 2, abs(P[i] - P[j]) /
                2};
            for (int k = 0; k < j; k++) {
                if (C.inside(P[k])) continue;
                C.o = Center(P[i], P[j], P[k]);
                C.r = abs(C.o - P[i]);
            }
        }
    }
    return C;
}
```

6.14 MEC2

```
PT arr[MXN];
int n = 10;
double checky(double x, double y) {
    double cmax = 0;
    for (int i = 0; i < n; i++) {  // 過程中回傳距離^2
        避免不必要的根號運算
        cmax = max(cmax, (arr[i].x - x) * (arr[i].x - x
            ) + (arr[i].y - y) * (arr[i].y - y));
    }
    return cmax;
}
double checkx(double x) {
    double yl = -1e9, yr = 1e9;
    while (yr - yl > EPS) {
        double ml = (yl + yl + yr) / 3, mr = (yl + yr +
             yr) / 3;
        if (checky(x, ml) < checky(x, mr))
            yr = mr;
        else
            yl = ml;
    }
}
signed main() {
    double xl = -1e9, xr = 1e9;
    while (xr - xl > EPS) {
        double ml = (xl + xl + xr) / 3, mr = (xl + xr +
             xr) / 3;
        if (checkx(ml) < checkx(mr))
            xr = mr;
        else
            xl = ml;
    }
}
```

6.15 旋轉卡尺

```
auto RotatingCalipers(const vector<Pt> &hull) { // 最遠
    點對 回傳距離平方
    int n = hull.size();
    auto ret = abs2(hull[0]);
    ret = 0;
    if (hull.size() <= 2) return abs2(hull[0] - hull
        [1]);
    for (int i = 0, j = 2; i < n; i++) {
        Pt a = hull[i], b = hull[(i + 1) % n];
        while(ori(hull[j], a, b) <
             (ori(hull[(j + 1) % n], a, b)))
            j = (j + 1) % n;
        chmax(ret, abs2(a - hull[j]));
```

```
        chmax(ret, abs2(b - hull[j]));
    }
    return ret;
}
```

6.16 Minkowski

```
// P, Q, R(return) are counterclockwise order convex
    polygon
vector<Pt> Minkowski(vector<Pt> P, vector<Pt> Q) {
    auto cmp = [&](Pt a, Pt b) {
        return Pt{a.y, a.x} < Pt{b.y, b.x};
    };
    auto reorder = [&](auto &R) {
        rotate(R.begin(), min_element(all(R), cmp), R.
            end());
        R.push_back(R[0]), R.push_back(R[1]);
    };
    const int n = P.size(), m = Q.size();
    reorder(P), reorder(Q);
    vector<Pt> R;
    for (int i = 0, j = 0, s; i < n or j < m; ) {
        R.push_back(P[i] + Q[j]);
        s = sgn((P[i + 1] - P[i]) ^ (Q[j + 1] - Q[j]));
        if (s >= 0) i++;
        if (s <= 0) j++;
    }
    return R;
}
```

6.17 PointInPolygon

```
int inPoly(Pt p, const vector<Pt> &P) {
    const int n = P.size();
    int cnt = 0;
    for (int i = 0; i < n; i++) {
        Pt a = P[i], b = P[(i + 1) % n];
        if (PtOnSeg(p, {a, b})) return 1; // on edge
        if ((sgn(a.y - p.y) == 1) ^ (sgn(b.y - p.y) ==
            1))
            cnt += sgn(ori(a, b, p));
    }
    return cnt == 0 ? 0 : 2; // out, in
}
```

6.18 UnionOfCircles

```
// Area[i] : area covered by at least i circle
// TODO:!!!aaa!!!
vector<double> CircleUnion(const vector<Cir> &C) {
    const int n = C.size();
    vector<double> Area(n + 1);
    auto check = [&](int i, int j) {
        if (!contain(C[i], C[j]))
            return false;
        return sgn(C[i].r - C[j].r) > 0 or (sgn(C[i].r
            - C[j].r) == 0 and i < j);
    };
    struct Teve {
        double ang; int add; Pt p;
        bool operator<(const Teve &b) { return ang < b.
            ang; }
    };
    auto ang = [&](Pt p) { return atan2(p.y, p.x); };
    for (int i = 0; i < n; i++) {
        int cov = 1;
        vector<Teve> event;
        for (int j = 0; j < n; j++) if (i != j) {
            if (check(j, i)) cov++;
            else if (!check(i, j) and !disjunct(C[i], C
                [j])) {
                auto I = CircleInter(C[i], C[j]);
                assert(I.size() == 2);
                double a1 = ang(I[0] - C[i].o), a2 =
                    ang(I[1] - C[i].o);
                event.push_back({a1, 1, I[0]});
                event.push_back({a2, -1, I[1]});
                if (a1 > a2) cov++;
            }
        }
        if (event.empty()) {
            Area[cov] += pi * C[i].r * C[i].r;
            continue;
        }
        sort(all(event));
        event.push_back(event[0]);
        for (int j = 0; j + 1 < event.size(); j++) {
            cov += event[j].add;
            Area[cov] += (event[j].p ^ event[j + 1].p)
                / 2.;
            double theta = event[j + 1].ang - event[j].
                ang;
            if (theta < 0) theta += 2 * pi;
            Area[cov] += (theta - sin(theta)) * C[i].r
                * C[i].r / 2.;
        }
    }
    return Area;
}
```

6.19 UnionOfPolygons

```
// Area[i] : area covered by at least i polygon
vector<double> PolyUnion(const vector<vector<Pt>> &P) {
    const int n = P.size();
    vector<double> Area(n + 1);
    vector<Line> Ls;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < P[i].size(); j++)
            Ls.push_back({P[i][j], P[i][(j + 1) % P[i].
                size()]});
    auto cmp = [&](Line &l, Line &r) {
        Pt u = l.b - l.a, v = r.b - r.a;
        if (argcmp(u, v)) return true;
        if (argcmp(v, u)) return false;
        return PtSide(l.a, r) < 0;
    };
    sort(all(Ls), cmp);
    for (int l = 0, r = 0; l < Ls.size(); l = r) {
        while (r < Ls.size() and !cmp(Ls[l], Ls[r])) r
            ++;
        Line L = Ls[l];
        vector<pair<Pt, int>> event;
        for (auto [c, d] : Ls) {
            if (sgn((L.a - L.b) ^ (c - d)) != 0) {
                int s1 = PtSide(c, L) == 1;
                int s2 = PtSide(d, L) == 1;
                if (s1 ^ s2) event.emplace_back(
                    LineInter(L, {c, d}), s1 ? 1 : -1);
            } else if (PtSide(c, L) == 0 and sgn((L.a -
                 L.b) * (c - d)) > 0) {
                event.emplace_back(c, 2);
                event.emplace_back(d, -2);
            }
        }
        sort(all(event), [&](auto i, auto j) {
            return (L.a - i.ff) * (L.a - L.b) < (L.a -
                j.ff) * (L.a - L.b);
        });
        int cov = 0, tag = 0;
        Pt lst{0, 0};
        for (auto [p, s] : event) {
            if (cov >= tag) {
                Area[cov] += lst ^ p;
                Area[cov - tag] -= lst ^ p;
            }
            if (abs(s) == 1) cov += s;
            else tag += s / 2;
            lst = p;
        }
    }
    for (int i = n - 1; i >= 0; i--) Area[i] += Area[i
        + 1];
    for (int i = 1; i <= n; i++) Area[i] /= 2;
    return Area;
};
```

6.20 圓公切線

```
vector<Line> CircleTangent(Cir c1, Cir c2, int sign1) {
    // sign1 = 1 for outer tang, -1 for inter tang
    vector<Line> ret;
    ld d_sq = abs2(c1.o - c2.o);
    if (sgn(d_sq) == 0) return ret;
```

```
    ld d = sqrt(d_sq);
    Pt v = (c2.o - c1.o) / d;
    ld c = (c1.r - sign1 * c2.r) / d;
    if (c * c > 1) return ret;
    ld h = sqrt(max(0.0, 1.0 - c * c));
    for (int sign2 = 1; sign2 >= -1; sign2 -= 2) {
        Pt n = Pt(v.x * c - sign2 * h * v.y, v.y * c +
            sign2 * h * v.x);
        Pt p1 = c1.o + n * c1.r;
        Pt p2 = c2.o + n * (c2.r * sign1);
        if (sgn(p1.x - p2.x) == 0 && sgn(p1.y - p2.y)
            == 0)
            p2 = p1 + rotate(c2.o - c1.o);
        ret.push_back({p1, p2});
    }
  return ret;
}
```

6.21 點圓切線

```
vector<Line> CircleTangent(Cir c, Pt p) {
    vector<Line> z;
    double d = abs(p - c.o);
    if (sgn(d - c.r) == 0) {
        Pt i = rotate(p - c.o);
        z.push_back({p, p + i});
    } else if (d > c.r) {
        double o = acos(c.r / d);
        Pt i = unit(p - c.o);
        Pt j = rotate(i, o) * c.r;
        Pt k = rotate(i, -o) * c.r;
        z.push_back({c.o + j, p});
        z.push_back({c.o + k, p});
    }
    return z;
}
```

6.22 最近點對

```
pair<ld, pair<i32, i32>> ClosestPair(vector<Pt> &P) {
    // ans = dis * dis  !!注意ans overflow問題
    if (P.size() == 1) { return {1e200L, {0, 0}}; }
    pair<i32, i32> ansi;
    auto ans = abs2(P[0] - P[1]);
    ansi = {0, 1};
    auto upd = [&](const Pt &a, const Pt &b) {
        auto dis = abs2(a - b);
        if (dis < ans) ans = dis, ansi.FF = a.id, ansi.
            SS = b.id;
    };
    auto cmpy = [](const Pt &a, const Pt &b) { return a
        .y < b.y; };

    vector<Pt> t(P.size() + 1);
    auto rec = [&](auto &&self, i32 l, i32 r) {
        if (r - l <= 3) {
            for (i32 i = l; i <= r; i++)
                for (i32 j = i + 1; j <= r; j++) upd(P[
                    i], P[j]);
            sort(P.begin() + l, P.begin() + r + 1, cmpy
                );
            return;
        }

        i32 m = (l + r) >> 1;
        auto midx = P[m].x;
        self(self, l, m), self(self, m + 1, r);
        i32 tsz = 0;
        inplace_merge(P.begin() + l, P.begin() + m + 1,
             P.begin() + r + 1, cmpy);
        for (i32 i = l; i <= r; i++) {
            if (abs(P[i].x - midx) * abs(P[i].x - midx)
                 >= ans) continue;
            for (i32 j = tsz - 1; j >= 0 && (P[i].y - t
                [j].y) * (P[i].y - t[j].y) < ans; j--)
                upd(P[i], t[j]);
            t[tsz++] = P[i];
        }
    };
    sort(all(P));
    rec(rec, 0, P.size() - 1);
    return make_pair(sqrt(ans), ansi);
}
```

7 graph

7.1 BCC

```
#define REP(i, n) for (int i = 0; i < n; i++)
struct BccVertex {
    int n, nScc, step, dfn[MXN], low[MXN];
    vector<int> E[MXN], sccv[MXN];
    int top, stk[MXN];
    void init(int _n) {
        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]);
        }
    }
    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);
            }
        REP(i, nScc) res.PB(sccv[i]);
        return res;
    }
} graph;
```

7.2 SCC

```
struct Scc{
  int n, nScc, vst[MXN], bln[MXN];
  vector<int> E[MXN], rE[MXN], vec;
  void init(int _n){
    n = _n;
    for (int i=0; i<= n; 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();
    fill(vst, vst+n+1, 0);
    for (int i=0; i<=n; i++)
```

```
      if (!vst[i]) DFS(i);
    reverse(vec.begin(),vec.end());
    fill(vst, vst+n+1, 0);
    for (auto v : vec)
      if (!vst[v]){
        rDFS(v); nScc++;
      }
  }
};
```

7.3 支配樹

```
#define REP(i, s, e) for (int i = (s); i <= (e); i++)
#define REPD(i, s, e) for (int i = (s); i >= (e); i--)
struct DominatorTree {  // O(N) 1-base
    int n, 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 _s) {
        ts = 0;
        n = _n;
        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) {
            idom[i] = par[i] = 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;
```

7.4 最大團

```
struct MaxClique {  // 0-base
    typedef bitset<MXN> Int;
    Int linkto[MXN], v[MXN];
    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[MXN];
    int id[MXN], di[MXN], deg[MXN];
    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;
```

7.5 最小圈

```
/* minimum mean cycle O(VE) */
struct MMC{
#define E 101010
#define V 1021
#define inf 1e9
```

```
#define eps 1e-6
  struct Edge { int v,u; double c; };
  int n, m, prv[V][V], prve[V][V], vst[V];
  Edge e[E];
  vector<int> edgeID, cycle, rho;
  double d[V][V];
  void init( int _n )
  { n = _n; m = 0; }
  // WARNING: TYPE matters
  void addEdge( int vi , int ui , double ci )
  { e[ m ++ ] = { vi , ui , ci }; }
  void bellman_ford() {
    for(int i=0; i<n; i++) d[0][i]=0;
    for(int i=0; i<n; i++) {
      fill(d[i+1], d[i+1]+n, inf);
      for(int j=0; j<m; j++) {
        int v = e[j].v, u = e[j].u;
        if(d[i][v]<inf && d[i+1][u]>d[i][v]+e[j].c) {
          d[i+1][u] = d[i][v]+e[j].c;
          prv[i+1][u] = v;
          prve[i+1][u] = j;
  } } } }
  double solve(){
    // returns inf if no cycle, mmc otherwise
    double mmc=inf;
    int st = -1;
    bellman_ford();
    for(int i=0; i<n; i++) {
      double avg=-inf;
      for(int k=0; k<n; k++) {
        if(d[n][i]<inf-eps) avg=max(avg,(d[n][i]-d[k][i
            ])/(n-k));
        else avg=max(avg,inf);
      }
      if (avg < mmc) tie(mmc, st) = tie(avg, i);
    }
    fill(vst,0); edgeID.clear(); cycle.clear(); rho.
        clear();
    for (int i=n; !vst[st]; st=prv[i--][st]) {
      vst[st]++;
      edgeID.PB(prve[i][st]);
      rho.PB(st);
    }
    while (vst[st] != 2) {
      if(rho.empty()) return inf;
      int v = rho.back(); rho.pop_back();
      cycle.PB(v);
      vst[v]++;
    }
    reverse(ALL(edgeID));
    edgeID.resize(SZ(cycle));
    return mmc;
} }mmc;
```

7.6 kShortestPath

```
while(Q.size()){
    auto [dx,x] = Q.top();Q.pop();
    if(dis[x].size() >= k) continue;
    dis[x].PB(dx);
    for(auto [v,w]:E[x]) Q.emplace(w+dx,v);
}
```

7.7 結論

- 2-SAT :
  $(a_i \vee a_j) = true \ \forall (i,j)$
  對於任意限制 $(x \vee y)$
  建兩條有向邊 (要多編號 $\neg x$)
  $x \to \neg y$ and $y \to \neg x$
  跑 scc
  scc.bln[x] $<$ scc.bln[$\neg x$] $\Leftrightarrow$ x is true
  scc.bln[$\neg x$] $<$ scc.bln[x] $\Leftrightarrow$ x is false
  $\exists x$ which scc.bln[x] $==$ scc.bln[$\neg x$] $\Leftrightarrow$ 無解

- 差分約束:
  n 個變數及 m 個約束條件
  求滿足所有 $x_j - x_i \le b_k \ (i,j \in [1,n], k \in [1,m])$
  的一組 $x_1 \ \dots \ x_n$
  可轉成 $x_j - x_i \le b_k \ \to \ x_j \le x_i + b_k$
  結論就是使得所有 x_j 變小以滿足上式
  建邊跑 SPFA/Bellman
  要多建起點 s 連到所有 i 且邊權 0, dis[s] = 0
  有負環則無解，否則起點到所有 i 的距離為一組解
  $x_j - x_i \le k \Rightarrow$ addEdge $i \xrightarrow{k} j$
  $x_j - x_i \ge k \Rightarrow$ addEdge $j \xrightarrow{-k} i$
  $x_j = x_i \Rightarrow$ addEdge $i \xrightarrow{0} j$ and $j \xrightarrow{0} i$

8 math

8.1 DiscreteSqrt

```
void calcH(i64 &t, i64 &h, const i64 p) {
  i64 tmp=p-1; for(t=0;(tmp&1)==0;tmp/=2) t++; h=tmp;
}
// solve equation x^2 mod p = a
// !!!! (a != 0) !!!!!
bool solve(i64 a, i64 p, i64 &x, i64 &y) {
  if(p == 2) { x = y = 1; return true; }
  int p2 = p / 2, tmp = mypow(a, p2, p);
  if (tmp == p - 1) return false;
  if ((p + 1) % 4 == 0) {
    x=mypow(a,(p+1)/4,p); y=p-x; return true;
  } else {
    i64 t, h, b, pb; calcH(t, h, p);
    if (t >= 2) {
      do {b = rand() % (p - 2) + 2;
      } while (mypow(b, p / 2, p) != p - 1);
      pb = mypow(b, h, p);
    } int s = mypow(a, h / 2, p);
    for (int step = 2; step <= t; step++) {
      int ss = (((i64)(s * s) % p) * a) % p;
      for(int i=0;i<t-step;i++) ss=mul(ss,ss,p);
      if (ss + 1 == p) s = (s * pb) % p;
          pb = ((i64)pb * pb) % p;
    } x = ((i64)s * a) % p; y = p - x;
  } return true;
}
```

8.2 excrt

```
i128 exgcd(i128 a, i128 b, i128 &x, i128 &y){
    if (b == 0) return x=1, y=0, a;
    int d = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return d;
}
// as -> 算式答案
// ns -> 模數 MOD
i128 CRT(vector<i64> as, vector<i64> ns) {
    i32 n = as.size();
    i128 a1, a2, n1, n2;
    bool flag = false;
    auto china = [&]() {
        i128 d = a2 - a1;
        i128 x, y;
        i128 g = exgcd(n1, n2, x, y);
        if (d % g == 0) {
            x = ((x * d / g) % (n2 / g) + (n2 / g)) % (
                n2 / g);
            a1 = x * n1 + a1;
            n1 = (n1 * n2) / g;
        } else {
            flag = true;
        }
    };

    a1 = as[0], n1 = ns[0];
    for (i32 i = 1; i < n; i++) {
        a2 = as[i], n2 = ns[i];
        china();
        if (flag) return -1;
    }
    return a1;
}
```

8.3 exgcd

```
int exgcd(int a,int b,int&x,int&y){
    if(b==0)return x=1,y=0,a;
    int d = exgcd(b,a%b,y,x);
    y-=a/b*x;
    return d;
}
```

8.4 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,i64 a[],int _m,i64 b[],i64 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]=(i64)(arr[i].real()/4+0.5);
}
```

8.5 josephus

```
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;
}
```

8.6 Theorem

- Lucas's 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):
  $S(n,k) = \frac{1}{k!}\sum_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^n$

- Pick's Theorem : $A = i + b/2 - 1$
  A: Area、i: grid number in the inner、b: grid number on the side

- 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 is number of color , m is the number of cycle size):
  $(\sum_{i=1}^{m} c^{gcd(i,m)})/m$

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

- 錯排公式: (n 個人中，每個人皆不再原來位置的組合數):
  $dp[0] = 1; dp[1] = 0;$
  $dp[i] = (i-1) * (dp[i-1] + dp[i-2]);$

- Bell 數 (有 n 個人, 把他們拆組的方法總數) :
  $B_0 = 1$
  $B_n = \sum_{k=0}^{n} s(n,k) \quad (second - stirling)$
  $B_{n+1} = \sum_{k=0}^{n}\binom{n}{k}B_k$

- 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 mod $C = A^{B \bmod \phi(c)+\phi(c)}$ mod C

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

- Standard young tableau (標準楊表):
  $\lambda = (\lambda_1 \geq \cdots \geq \lambda_k), \sum \lambda_i = n$ denoted by $\lambda \vdash n$
  $\lambda \vdash n$ 意思為 λ 整數拆分 n eg. $n = 10, \lambda = (6,4)$ 此拆分可表示一種楊表形狀。
  楊表: 第 1 列 λ_1 行 $\cdots$ 第 k 列 λ_k 行的方格圖。
  標準楊表: 每列從左到右遞增，每行從上到下遞增。
  Let T 為某一 Permutation 跑 RSK 後的標準楊表，則此 Permutation 的 LDS、LIS 長度分別為 T 的列、行數。

- RSK Correspondence:
  A permutation is bijective to (P, Q) 一對標準楊表
  P : Permutation 跑 RSK 算法的結果，可為半標準楊表。
  Q : 可用來還原 Permutation (像排列矩陣)。

- Hook length formula (形狀為 λ 的標準楊表個數):
  $f^\lambda = \frac{n!}{\prod h_\lambda(i,j)}$
  $h_\lambda(i,j) =$ number of pair (x,y) where $(x = i \vee y = j) \wedge (x,y) \geq (i,j)$ 且 (x,y) 落在形狀為 λ 的表上。
  Recursion:
  (i) $f^{(0,\cdots,0)} = 1$
  (ii) $f^{(\lambda_1,\cdots,\lambda_m)} = \sum_{k=1}^{m} f^{(\lambda_1,\cdots,\lambda_{k-1},\lambda_k-1,\lambda_{k+1},\cdots,\lambda_m)}$

8.7 Primes

Prime	Root	Prime	Root
7681	17	167772161	3
12289	11	104857601	3
40961	3	985661441	3
65537	3	998244353	3
786433	10	1107296257	10
5767169	3	2013265921	31
7340033	3	2810183681	11
23068673	3	2885681153	3
469762049	3	605028353	3
924844033	5	0	0

8.8 millerrabin

```
// n < 4,759,123,141          3 :  2, 7, 61
// n < 1,122,004,669,633      4 :  2, 13, 23, 1662803
// n < 3,474,749,660,383          6 :  pirmes <= 13
// n < 2^64                       7 :
// 2, 325, 9375, 28178, 450775, 9780504, 1795265022
// Make sure testing integer is in range [2, n-2] if
// you want to use magic.
bool witness(i64 a,i64 n,i64 u,int t){
```

```
  if(!a) return 0;
  i64 x=mypow(a,u,n);
  for(int i=0;i<t;i++) {
    i64 nx=mul(x,x,n);
    if(nx==1&&x!=1&&x!=n-1) return 1;
    x=nx;
  }
  return x!=1;
}
bool mii64er_rabin(i64 n) {
  int s = 7;
  // iterate s times of witness on n
  if(n<2) return 0;
  if(!(n&1)) return n == 2;
  i64 u=n-1; int t=0;
  // n-1 = u*2^t
  while(!(u&1)) u>>=1, t++;
  while(s--){
    i64 a=magic[s]%n;
    if(witness(a,n,u,t)) return 0;
  }
  return 1;
}
```

8.9 phi

```
ll phi(ll n){   // 計算小於n的數中與n互質的有幾個
    ll res = n, a=n;    // O(sqrtN)
    for(ll i=2;i*i<=a;i++){
        if(a%i==0){
            res = res/i*(i-1);
            while(a%i==0) a/=i;
    }   }
    if(a>1) res = res/a*(a-1);
    return res;
}
```

8.10 pollardrho

```
// does not work when n is prime  O(n^(1/4))
i64 f(i64 x, i64 c, i64 mod){ return add(mul(x,x,mod),c
    ,mod); }
i64 poi64ard_rho(i64 n) {
    i64 c = 1, x = 0, y = 0, p = 2, q, t = 0;
    while (t++ % 128 or gcd(p, n) == 1) {
        if (x == y) c++, y = f(x = 2, c, n);
        if (q = mul(p, abs(x-y), n)) p = q;
        x = f(x, c, n); y = f(f(y, c, n), c, n);
    }
    return gcd(p, n);
}
```

8.11 primes

```
/* 12721, 13331, 14341, 75577, 123457, 222557, 556679
* 999983, 1097774749, 1076767633, 100102021, 999997771
* 1001010013, 1000512343, 987654361, 999991231
* 999888733, 98789101, 987777733, 999991921, 1010101333
* 1010102101, 1000000000039, 1000000000000037
* 2305843009213693951, 4611686018427387847
* 9223372036854775783, 18446744073709551557 */
int mu[ N ] , p_tbl[ N ];
vector<int> primes;
void sieve() {
  mu[ 1 ] = p_tbl[ 1 ] = 1;
  for( int i = 2 ; i < N ; i ++ ){
    if( !p_tbl[ i ] ){
      p_tbl[ i ] = i;
      primes.push_back( i );
      mu[ i ] = -1;
    }
    for( int p : primes ){
      int x = i * p;
      if( x >= M ) break;
      p_tbl[ x ] = p;
      mu[ x ] = -mu[ i ];
      if( i % p == 0 ){
        mu[ x ] = 0;
        break;
} } } }
vector<int> factor( int x ){
  vector<int> fac{ 1 };
  while( x > 1 ){
    int fn = SZ(fac), p = p_tbl[ x ], pos = 0;
    while( x % p == 0 ){
      x /= p;
      for( int i = 0 ; i < fn ; i ++ )
        fac.PB( fac[ pos ++ ] * p );
  } }
  return fac;
}
```

8.12 Euler

```
int Euler(int n){
  int now = n;
  for (int i = 2; i * i <= n; i++)
    if (n % i == 0){
      now = now - now / i;
      while (n % i == 0) n = n / i;
        }
    if (n > 1) now = now - now / n;
    return now;
}
```

8.13 quickeuler

```
vector<int> pri;
bool not_prime[MXN + 10];
int phi[MXN + 10];
void quick_euler(int n) {
    phi[1] = 1;
    for (int i = 2; i <= n; i++) {
        if (!not_prime[i]) {
            pri.push_back(i);
            phi[i] = i - 1;
        }
        for (int pri_j : pri) {
            if (i * pri_j > n)
                break;
            not_prime[i * pri_j] = true;
            if (i % pri_j == 0) {
                phi[i * pri_j] = phi[i] * pri_j;
                break;
            }
            phi[i * pri_j] = phi[i] * phi[pri_j];
        }
    }
}
```

8.14 sieve

```
const int MXN = 1e8 + 50;
const int SQRTMXN = 1e4 + 50;
bitset<MXN> isprime;
void sieve() {
    isprime[1] = 1;
    for (int i = 2; i <= SQRTMXN; i++) {
        if (!isprime[i])
            for (i64 j = i * i; j < MXN; j += i)
                isprime[j] = 1;
    }
}
```

8.15 NTT

```
constexpr i64 power(i64 a, i64 b, i64 m) {
    i64 ret = 1;
    for (; b; b >>= 1, a = a * a % m)
        if (b & 1) ret = ret * a % m;
    return ret;
};
template<i64 M, i64 root>
struct NTT {
  static const int Log = 21;
    array<i64, Log + 1> e{}, ie{};
    NTT() {
    static_assert(__builtin_ctz(M - 1) >= Log);
        e[Log] = power(root, (M - 1) >> Log, M);
        ie[Log] = power(e[Log], M - 2, M);
        for (int i = Log - 1; i >= 0; i--) {
            e[i] = e[i + 1] * e[i + 1] % M;
            ie[i] = ie[i + 1] * ie[i + 1] % M;
        }
```

```
    }
    void operator()(vector<i64> &v, bool inv) {
        int n = v.size();
        for (int i = 0, j = 0; i < n; i++) {
            if (i < j) swap(v[i], v[j]);
            for (int k = n / 2; (j ^= k) < k; k /= 2);
        }
        for (int m = 1; m < n; m *= 2) {
            i64 w = (inv ? ie : e)[__lg(m) + 1];
            for (int i = 0; i < n; i += m * 2) {
                i64 cur = 1;
                for (int j = i; j < i + m; j++) {
                    i64 g = v[j], t = cur * v[j + m] %
                        M;
                    v[j] = (g + t) % M;
                    v[j + m] = (g - t + M) % M;
                    cur = cur * w % M;
                }
            }
        }
        if (inv) {
            i64 in = power(n, M - 2, M);
            for (int i = 0; i < n; i++) v[i] = v[i] *
                in % M;
        }
    }
};
template<int M, int G> //nlogn f*g
vector<i64> convolution(vector<i64> f, vector<i64> g) {
  static NTT<M, G> ntt;
    int n = ssize(f) + ssize(g) - 1;
    int len = bit_ceil(1ull * n);
    f.resize(len);
    g.resize(len);
    ntt(f, 0), ntt(g, 0);
    for (int i = 0; i < len; i++) {
        (f[i] *= g[i]) %= M;
    }
    ntt(f, 1);
    f.resize(n);
    return f;
}
```

9 polynomial

9.1 Definition

```
using poly = vector<i64>;
poly operator * (poly A, poly B){
    return convolution<mod,G>(A,B);
}
poly operator - (i64 r,poly A){
    r %= mod;
    r = (r + mod) % mod;
    for(auto &c:A){
        c %= mod;
        c = (mod - c) % mod;
    }
    A[0] = (A[0] + r) % mod;
    return A;
}
poly operator + (i64 r, poly A){
    r %= mod;
    r = (r + mod) % mod;
    A[0] = (A[0] + r) % mod;
    return A;
}
poly operator * (i64 r, poly A){
    for(auto &c:A){
        c = r * c % mod;
        c = (c + mod) % mod;
    }
    return A;
}
poly operator - (poly A, poly B){
    int n = max(A.size(),B.size());
    A.resize(n),B.resize(n);
    for(int i = 0; i < n;++i){
        A[i] = (A[i] - B[i]) % mod;
        A[i] = (A[i] + mod) % mod;
    }
    return A;
}
poly operator + (poly A, poly B){
    int n = max(A.size(),B.size());
    A.resize(n),B.resize(n);
    for(int i = 0; i < n;++i){
        A[i] = (A[i] + B[i]) % mod;
        A[i] = (A[i] + mod) % mod;
    }
    return A;
}
poly operator % (poly A,int n){
    A.resize(n);
    return A;
}

poly derive(poly P){
    if(!P.size()) return P;
    for(int i = 0; i < P.size();++i){
        P[i] = i * P[i] % mod;
        P[i] = (P[i] + mod) % mod;
    }
    P.erase(P.begin());
    return P;
}
poly integr(poly P){
    if(!P.size()) return P;
    for(int i = 0; i < P.size();++i){
        P[i] = P[i] * inv(i + 1) % mod;
        P[i] = (P[i] + mod) % mod;
    }
    P.insert(P.begin(),0);
    return P;
}
```

9.2 Inverse Series

```
poly inverse(poly A){
    int n = A.size();
    int tn = 1;
    poly Q(tn);
    Q[0] = inv(A[0]);
    while(tn < n){
        tn *= 2;
        Q = Q * (2 - A % tn * Q % tn);
        Q = Q % tn;
    }
    return Q;
}
```

9.3 Poly Shift

```
// nlogn f(x) -> f(x + k)
auto shift = [&](vector<i64> f,i64 k) {
    k %= mod;
    k += mod;
    k %= mod;
    int n = f.size() - 1;
    vector<i64> g(n+1);
    for(int i = 0; i <= n;++i){
        f[i] = f[i] * fac[i] % mod;
        g[n - i] = fpow(k,i) * inv(fac[i]) % mod;
        //x^(-n) -> x^(0)
    }
    auto h = convolution<mod,3>(f,g);
    h.erase(h.begin(),h.begin()+n);
    for(int i = 0; i <= n;++i) h[i] = h[i] * inv(fac[i
        ]) % mod;
    return h;
};
```

9.4 Exponential

```
poly exp(poly P){
    int n = P.size();
    int tn = 1;
    poly Q(tn);
    Q[0] = 1;
    while(tn < n){
        tn *= 2;
        Q = Q * (1 + P % tn - log(Q) % tn);
        Q = Q % tn;
    }
```

```
    return Q;
}
```

9.5 Nature log

```
poly log(poly P){
    return integr(derive(P) * inverse(P));
}
```

10 other

10.1 cdq

```
// 三維偏序 (求 arr[j] < arr[i] (每一維嚴格小於), i!=j
    j 的個數)
// 先照 x 排序 merge sort排y 最後BIT動態求z的順序個數
// 左區間的 x < 右區間的
void cdq(int ll,int rr){
    if(ll == rr) return;
    int m = (ll+rr)/2;
    cdq(ll,m),cdq(m+1,rr);
    int i = ll,j = m+1,t = 0;
    auto work = [&](){
        ans += BIT.qry(arr[j].z); //計數
        temp[t++] = arr[j++];
    };
    while(i <= m && j <= rr){
        if(arr[i].y <= arr[j].y){
            BIT.add(arr[i].z,1); //二維偏序求法
            temp[t++] = arr[i++];
        }
        else work();
    }
    while(i <= m) temp[t++] = arr[i++];
    while(j <= rr) work();
    BIT.reset(); //操作復原
    rep(k,0,t) arr[k+ll] = temp[k];
}
//[l,r)
auto cdq = [&](auto&& self,auto l,auto r){
    if((r - l) <= 1) return;
    auto m = (r - l) / 2 + l;
    self(self,l,m);
    self(self,m,r);
    auto i = l,j = m;
    auto work = [&](){
        ++j;
    };
    while(i != m && j != r){
        if(arr[*i][1] <= arr[*j][1]) {
            ++i;
        }else work();
    }
    while(j != r) work();
    clear();
    inplace_merge(l,m,r,[&](auto a,auto b){
        return arr[a][1] < arr[b][1];
    });
};
cdq(cdq,all(ord));//排ord
```

10.2 DeBruijnSequence

```
//求由所有 N 長度bitstring作為substring 最短的字串 B(2,
    N) //B(k,N) : 以k個字元作為N長度字串節點
//00110 -> 00 01 11 10
//建圖 : 點為substrings 邊用 0 1 連接
//走訪 : 000 -1-> 001
// 解為 Hamiltonian 路徑 (剛好所有節點走過一遍)
// 可同構到 N-1 圖上的Eulerian Circuit (每條邊 N-1 圖上
    的邊 代表 N 圖上的一個點)
vector<int> edges[1<<(N-1)];
vector<int> ans;
void dfs(int x){ // Eulerian Circuit
    while(edges[x].size()){
        int u = edges[x].back();
        edges[x].pop_back();
        ans.push_back(u&1);
        dfs(u);
    }
}
void solve(int n){
    if(n == 1) {
        ans = {1,0};
        return;
    }
     for(int i = 0; i < (1<<(n-1)); ++i){
        edges[i].push_back((i<<1)&((1<<(n-1))-1)); // 0
            的邊
        edges[i].push_back(((i<<1)+1)&((1<<(n-1))-1));
            // 1 的邊
    }
    for(int i = 0; i < n-1;++i) ans.push_back(0); //初
        始狀態
    dfs(0);
}
```

10.3 SmallestLexicographic

```
//對於可化作DAG的回朔問題求最小字典序的選擇
//建反圖 (反著做回來) (把以 i 結尾變成 以 i 開頭)
//結論 : i <- j (i < j) 取最小的 a[j]
for(int j = N; j; --j) {
    for(auto i:E[j])
    dp[i] = min(dp[i],dp[j]);
}
```

11 random

11.1 XORShift

```
const i64 mask = std::chrono::steady_clock::now().
    time_since_epoch().count();
//13 17 5
//13 17 7
i64 shift(i64 x) { // XOR shift (1-1 func)
  x ^= x << 13;
  x ^= x >> 7;
  x ^= x << 17;
  x ^= mask;
  return x;
}
```

12 string

12.1 KMP

```
//pi[i] = 最大的 k 使得 s[0...(k-1)] = s[i-(k-1)...i]
vector<int> prefunc(const string& s){
  int n = s.size();
  vector<int> pi(n);
  for(int i=1,j=0;i<n;++i){
    j = pi[i-1];
    while(j && s[j] != s[i]) j = pi[j-1]; //取次小LCP
    if(s[j] == s[i]) ++j;
    pi[i] = j;
  }
  return pi;
}
//找 s 在 str 中出現的所有位子
vector<int> kmp(string str, string s) {
    vector<int> nxt = prefunc(s);
    vector<int> ans;
    for (int i = 0, j = 0; i < SZ(str); i++) {
        while (j && str[i] != s[j]) j = nxt[j - 1];
        if (str[i] == s[j]) j++;
        if (j == SZ(s)) {
            ans.push_back(i - SZ(s) + 1);
            j = nxt[j - 1];
        }
    }
    return ans;
}
```

12.2 minRotation

```
// rotate(begin(s),begin(s)+minRotation(s),end(s))
#define rep(i, s, e) for (int i = (s); i < (e); i++)
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(0LL, k - 1);
            break;
        }
        if (s[a + k] > s[b + k]) {
            a = b;
            break;
        }
    }
    return a;
}
```

12.3 PalindromeTree

```
// 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
// fail[s] -> s 建邊是顆樹
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,int LEN){
    tot=lst=n=0;
    newNode(0,1),newNode(-1,1);
    for(; n<LEN && _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;
```

12.4 RollingHash

```
struct RollingHash{
#define psz 2
    vector<ll> primes={17, 75577};
    vector<ll> MOD={998244353, 1000000007};
    vector<array<ll, psz>> hash, base;
    void init(const string &s){
        hash.clear(); hash.resize(s.size());
        base.clear(); base.resize(s.size());
        for(int i=0;i<psz;i++){
            hash[0][i] = s[0];
            base[0][i] = 1;
        }
        for(int i=1;i<s.size();i++){
            for(int j=0;j<psz;j++){
                hash[i][j] = (hash[i-1][j] * primes[j]
                    % MOD[j] + s[i]) % MOD[j];
                base[i][j] = base[i-1][j] * primes[j] %
                    MOD[j];
            }
        }
    }
    array<ll, psz> getHash(int l,int r){
        if(l == 0) return hash[r];
        array<ll, psz> ret = hash[r];
        for(int i=0;i<psz;i++){
            ret[i] -= hash[l-1][i] * base[r-l+1][i] %
                MOD[i];
            if(ret[i]<0) ret[i]+=MOD[i];
        }
        return ret;
    }
}Hash;
```

12.5 SuffixArray

```
const int N = 300010;
struct SA{
#define REP(i,n) for ( int i=0; i<int(n); i++ )
#define REP1(i,a,b) for ( int i=(a); i<=int(b); i++ )
  bool _t[N*2];
  int _s[N*2], _sa[N*2], _c[N*2], x[N], _p[N], _q[N*2],
      hei[N], r[N];
  int operator [] (int i){ return _sa[i]; }
  void build(int *s, int n, int m){
    memcpy(_s, s, sizeof(int) * n);
    sais(_s, _sa, _p, _q, _t, _c, n, m);
    mkhei(n);
  }
  void mkhei(int n){
    REP(i,n) r[_sa[i]] = i;
    hei[0] = 0;
    REP(i,n) if(r[i]) {
      int ans = i>0 ? max(hei[r[i-1]] - 1, 0) : 0;
      while(_s[i+ans] == _s[_sa[r[i]-1]+ans]) ans++;
      hei[r[i]] = ans;
    }
  }
  void sais(int *s, int *sa, int *p, int *q, bool *t,
      int *c, int n, int z){
    bool uniq = t[n-1] = true, neq;
    int nn = 0, nmxz = -1, *nsa = sa + n, *ns = s + n,
        lst = -1;
#define MS0(x,n) memset((x),0,n*sizeof(*(x)))
#define MAGIC(XD) MS0(sa, n); \
    memcpy(x, c, sizeof(int) * z); \
    XD; \
    memcpy(x + 1, c, sizeof(int) * (z - 1)); \
    REP(i,n) if(sa[i] && !t[sa[i]-1]) sa[x[s[sa[i
        ]-1]]++] = sa[i]-1; \
    memcpy(x, c, sizeof(int) * z); \
    for(int i = n - 1; i >= 0; i--) if(sa[i] && t[sa[i
        ]-1]) sa[--x[s[sa[i]-1]]] = sa[i]-1;
    MS0(c, z);
    REP(i,n) uniq &= ++c[s[i]] < 2;
    REP(i,z-1) c[i+1] += c[i];
    if (uniq) { REP(i,n) sa[--c[s[i]]] = i; return; }
    for(int i = n - 2; i >= 0; i--) t[i] = (s[i]==s[i
        +1] ? t[i+1] : s[i]<s[i+1]);
    MAGIC(REP1(i,1,n-1) if(t[i] && !t[i-1]) sa[--x[s[i
        ]]]=p[q[i]=nn++]=i);
    REP(i, n) if (sa[i] && t[sa[i]] && !t[sa[i]-1]) {
      neq=lst<0||memcmp(s+sa[i],s+lst,(p[q[sa[i]]+1]-sa
          [i])*sizeof(int));
      ns[q[lst=sa[i]]]=nmxz+=neq;
    }
    sais(ns, nsa, p + nn, q + n, t + n, c + z, nn, nmxz
        + 1);
    MAGIC(for(int i = nn - 1; i >= 0; i--) sa[--x[s[p[
        nsa[i]]]]] = p[nsa[i]]);
  }
}sa;
// H [i] 第 i 跟前面的最大共同前綴
// SA[i] 第 i 小是從第幾個字元開始
int H[ N ], SA[ N ];
void suffix_array(int* ip, int len) {
  // should padding a zero in the back
  // ip is int array, len is array length
  // ip[0..n-1] != 0, and ip[len] = 0
```

```
  ip[len++] = 0;
  sa.build(ip, len, 128); // 注意字元個數
  for (int i=0; i<len; i++) {
    H[i] = sa.hei[i + 1];
    SA[i] = sa._sa[i + 1];
  }
  // resulting height, sa array \in [0,len)
}
```

12.6 trie

```
//01 bitwise trie
struct trie{
    trie *nxt[2];  // 差別
    int cnt;    //紀錄有多少個數字以此節點結尾
    int sz;     //有多少數字的前綴包括此節點
    trie():cnt(0),sz(0){
        memset(nxt,0,sizeof(nxt));
    }
};
//創建新的字典樹
trie *root;
void insert(int x){
    trie *now = root;  // 每次從根節點開始
    for(int i=22;i>=0;i--){ // 從最高位元開始往低位元走
        now->sz++;
        //cout<<(x>>i&1)<<endl;
        if(now->nxt[x>>i&1] == NULL){ //判斷當前第 i 個
            位元是 0 還是 1
            now->nxt[x>>i&1] = new trie();
        }
        now = now->nxt[x>>i&1];  //走到下一個位元
    }
    now->cnt++;
    now->sz++;
}
```

12.7 Z-algorithm

```
//z[i] = s 跟 s[i..n-1] 的最長真共同前綴長度 // z[0] =
    0
vector<int> zfunc(string &s){
  int n = s.size();
  vector<int> z(n);
  for(int i = 1,l = 0,r = 0; i < n;++i){
    if(i <= r && z[i - l] < r - i + 1) z[i] = z[i - l];
    else {
      z[i] = max(0LL,r - i + 1);
      while(i + z[i] < n && s[z[i]] == s[i + z[i]]) ++z
          [i];
    }
    if(i + z[i] - 1 > r) l = i, r = i + z[i] - 1;
  }
  return z;
}
```

12.8 馬拉車

```
//以每個字元為中心的最長迴文長度
//abc -> @a@b@c
void z_value_pal(char* s, int len, int* z) {
    len = (len << 1) + 1;
    for (int i = len - 1; i >= 0; i--)
        s[i] = i & 1 ? s[i >> 1] : '@';
    z[0] = 1;
    for (int i = 1, l = 0, r = 0; i < len; i++) {
        z[i] = i < r ? min(z[l + l - i], r - i) : 1;
        while (i - z[i] >= 0 && i + z[i] < len && s[i -
             z[i]] == s[i + z[i]])
            ++z[i];
        if (i + z[i] > r)
            l = i, r = i + z[i];
    }
}
```

13 tree

13.1 DSUONTREE

```
int ans[MXN], color[MXN], son[MXN];
map<int, int> mp[MXN];
void dfs(int x, int f){
    if(son[x]){
        dfs(son[x], x);
        swap(mp[x], mp[son[x]]);
        ans[x] = ans[son[x]];
    }
    mp[x][color[x]]++;
    ans[x] = max(ans[x], mp[x][color[x]]);
    for(int i : edge[x]){
        if(i == f || i == son[x])    continue;
        dfs(i, x);
        for(auto j : mp[i]){
            mp[x][j.first] += j.second;
            ans[x] = max(ans[x], mp[x][j.first]);
        }
    }
}
```

13.2 EularTour

```
int timing=0;
int in[N],out[N];
void dfs(int u){
    in[u] = ++timing;//這時進入u
    for(int nxt : g[u]){//跑過所有孩子
        dfs(nxt);
    }
    out[u] = timing;//這時離開u 不會++

}
```

13.3 LCA

```
int n, q;
int anc[MAXN][25], in[MAXN], out[MAXN];
vector<int> edge[MAXN];
int timing = 1;
void dfs(int cur, int fa) {
    anc[cur][0] = fa;
    in[cur] = timing++;
    for (int nex : edge[cur]) {
        if (nex == fa) continue;
        dfs(nex, cur);
    }
    out[cur] = timing++;
}
void init() {
    dfs(1, 0);
    for (int i = 1; i < 25; i++) {
        for (int cur = 1; cur <= n; cur++) {
            anc[cur][i] = anc[anc[cur][i - 1]][i - 1];
        }
    }
}
bool isanc(int u, int v) { return (in[u] <= in[v] &&
    out[v] <= out[u]); }
int lca(int a, int b) {
    if (isanc(a, b)) return a;
    if (isanc(b, a)) return b;
    for (int i = 24; i >= 0; i--) {
        if (anc[a][i] == 0) continue;
        if (!isanc(anc[a][i], b)) a = anc[a][i];
    }

    return anc[a][0];
}



int t = 0,tt = 0;
vector<int> dfn(n),in(n),out(n),dep(n);
vector anc(n,vector<int>(20));
auto pdfs = [&](auto &&self,int x,int f,int d = 0) ->
    void {
    in[x] = ++t;
    anc[x][0] = f;
    dep[x] = d;
    dfn[x] = ++tt;
    for(auto u:E[x]){
        if(u == f) continue;
        self(self,u,x,d+1);
    }
```

```
    out[x] = ++t;
};
pdfs(pdfs,0,0);
for(int k = 1; k < 20;++k){
    for(int i = 0; i < n;++i){
        anc[i][k] = anc[anc[i][k-1]][k-1];
    }
}
auto isanc = [&](int u,int v){
    return in[u] <= in[v] && out[v] <= out[u];
};
auto lca = [&](int x,int y){
    if(isanc(x,y)) return x;
    if(isanc(y,x)) return y;
    for(int i = 19; i >= 0; --i){
        if(!isanc(anc[x][i],y)) x = anc[x][i];
    }
    return anc[x][0];
};
```

13.4 treehash

```
map<vector<int>,int> id; //rooted
int dfs(int x,int f){
    vector<int> s;
    for(int u:E[x]){
        if(u == f) continue;
        s.PB(dfs(u,x));
    }
    sort(all(s));
    if(!id.count(s)) id[s] = id.size();
    return id[s];
}

const i64 mask = std::chrono::steady_clock::now().
    time_since_epoch().count();
//13 17 5
//13 17 7
i64 shift(i64 x) { // XOR shift (1-1 func)
  x ^= mask;
  x ^= x << 13;
  x ^= x >> 7;
  x ^= x << 17;
  x ^= mask;
  return x;
}

int dfs(int x,int f){
    int ret = 1; // 需要常數
     for(int u:E[x]){
        if(u == f) continue;
        ret += shift(dfs(u,x));
    }
    // ret ^= rand_mask //如果xor hash被卡
    return ret;
}
```

13.5 HeavyLightDecomposition

```
int t = 0;
vector<int> dep(n+1),p(n+1),sz(n+1),dfn(n+1),son(n+1);
auto dfs = [&](auto &&self,int x,int f,int d = 0) ->
    void {
    ++sz[x],dep[x] = d,p[x] = f;
    for(auto u:E[x]){
        if(u == f) continue;
        self(self,u,x,d+1);
        sz[x] += sz[u];
        if(!son[x] || sz[u] > sz[son[x]]) son[x] = u;
    }
};
vector<int> top(n+1);
auto dfsa = [&](auto &&self,int x,int f,int now) ->
    void {
    dfn[x] = ++t;
    top[x] = now;
    if(son[x]) self(self,son[x],x,now);
    for(auto u:E[x]){
        if(u == f || u == son[x]) continue;
        self(self,u,x,u);
    }
};
dfs(dfs,1,1);
dfsa(dfsa,1,1,1);
auto lca = [&](int x,int y){
    while(top[x] != top[y]){
        if(dep[top[x]] < dep[top[y]]) swap(x,y);
        x = p[top[x]];
    }
    return dep[x] < dep[y] ? x : y ;
};
// 如果要開線段樹 要每個鏈都開一顆 (比較快)
```

13.6 VirtualTree

```
//求關鍵點的虛樹
//thm1 : 照dfn (dfs序) 排序後的 "相鄰點" 求lca可求出全
    點對的lca
auto virTree = [&](vector<int> key){
    auto cmp = [&](int a,int b){return dfn[a] < dfn[b
        ];};
    sort(all(key),cmp);
    auto res = vector<int>(all(key));
    for(int i = 1; i < key.size();++i){
        res.PB(lca(key[i-1],key[i]));
    }
    sort(all(res),cmp);
    res.erase(unique(all(res)),res.end());
    return res; // res : 全點對lca集 + 關鍵點集
};
//詢問
for(int i = 1; i < ret.size(); ++i){
    int LCA = lca(ret[i-1],ret[i]);
    query(LCA,ret[i]); // 2. LCA -> ret[i] 是一條
        virTree的邊
    //query : 路徑詢問
    //且會全部算到
}
```