Template

GummyBear

May 17, 2019

目录

1 A **1**
- 1.1 .vimrc . . . 1
- 1.2 FastIO . . . 1
- 1.3 Head . . . 1
- 1.4 duipai . . . 1

2 DP **2**
- 2.1 DigDP . . . 2

3 DataStructure **2**
- 3.1 2DST . . . 2
- 3.2 2DSegTree . . . 2
- 3.3 CDQ . . . 3
- 3.4 CartesianTree . . . 3
- 3.5 Fenwick . . . 3
- 3.6 IntervalMaximumChangeTimes . . . 4
- 3.7 KDT . . . 4
- 3.8 LCT . . . 5
- 3.9 Rope . . . 6
- 3.10 ST . . . 6
- 3.11 SegIntervalMax . . . 6
- 3.12 Splay . . . 7
- 3.13 fhqTreap . . . 7
- 3.14 lcSegTree . . . 8
- 3.15 perTreap . . . 8
- 3.16 动态 k 大 . . . 9
- 3.17 动态树上路径 k 大 . . . 10
- 3.18 常见转化 . . . 10
- 3.19 覆盖大于 k 次的矩形面积 . . . 10
- 3.20 高维偏序 . . . 11

4 Game **11**
- 4.1 Game . . . 11

5 Geometry **11**
- 5.1 GeoAdd . . . 11
- 5.2 MaxAreaPoly . . . 13
- 5.3 MaxAreaTri . . . 14

6 Graph **14**
- 6.1 BCC . . . 14
- 6.2 DCC . . . 14
- 6.3 DMST . . . 14
- 6.4 Dijkstra . . . 15
- 6.5 Dinic . . . 15
- 6.6 DualMST . . . 16
- 6.7 EulerianPath . . . 16
- 6.8 FindCircle . . . 16
- 6.9 Lindstrom_Gessel_Viennot_Lemma . . . 16
- 6.10 MMST . . . 16
- 6.11 ManhattanDistance . . . 17
- 6.12 MaxMatch . . . 17
- 6.13 Max_clique_BK . . . 17
- 6.14 Max_clique_fastest . . . 17
- 6.15 MinCostMaxFlow . . . 18
- 6.16 SCC . . . 18
- 6.17 SteinerTree . . . 19
- 6.18 StoerWagner_O(n3) . . . 19
- 6.19 StoerWagner_O(nmlog(m)) . . . 20
- 6.20 生成树计数与欧拉回路方案数 . . . 20

7 Math **21**
7.1 BerlekampMassey . . . 21
7.2 Bernoulli . . . 21
7.3 CRT . . . 21
7.4 EulerPower . . . 21
7.5 FFT . . . 22
7.6 FFTMOD . . . 22
7.7 FFT_fast . . . 23
7.8 Fib . . . 23
7.9 GaussDB . . . 23
7.10 GaussInt . . . 24
7.11 GaussXor . . . 24
7.12 LinearBasis . . . 25
7.13 LinearRecursion . . . 25
7.14 MathFunction . . . 25
7.15 NTT . . . 26
7.16 Polya . . . 26
7.17 SternBrocotTree . . . 26
7.18 min_25 . . . 27
7.19 ploynomial . . . 27
7.20 polysum . . . 28
7.21 prime . . . 28

8 Others **29**
8.1 BitOperation . . . 29
8.2 Bitset . . . 29
8.3 RomanNumerals . . . 29
8.4 Strtok . . . 30

9 String **30**
9.1 ACAutomaton . . . 30
9.2 DoublingArray . . . 30
9.3 Exkmp . . . 30
9.4 Kmp . . . 31
9.5 Manacher . . . 31
9.6 PalindromicTree . . . 31
9.7 SAIS . . . 31
9.8 StrHash . . . 32
9.9 SuffixAutomaton . . . 33

10 Tree **33**
10.1 Centroid . . . 33
10.2 DsuOnTree . . . 33
10.3 HeavyChain . . . 34
10.4 LCARMQ . . . 34
10.5 LongChain . . . 34
10.6 MoOnTree_Path . . . 35
10.7 VTree . . . 35

1 A

1.1 .vimrc

```
set nu ai ci si mouse=a ts=2 sts=2 sw=2
nmap<F2> : vs %<.in <CR>
nmap<F3> : !gedit % <CR>
nmap<F8> : !time ./%< < %<.in <CR>
nmap<F9> : :w <CR> :!g++ % -o %< -O2 -g -std=c++11 -Wall <CR>

nmap<F5> : !./%< <CR>
nmap<F10> : :w <CR> :make %< <CR>
```

1.2 FastIO

```
// read untill EOF (xint)
struct FastIO {
    static const int S = 1310720;
    int wpos;
    char wbuf[S];
    bool ed;
    FastIO() : wpos(0), ed(0) { }
    inline int xchar() {
        static char buf[S];
        static int len = 0, pos = 0;
        if (pos == len) pos = 0, len = fread(buf, 1, S, stdin);
        if (pos == len) return -1;
        return buf[pos++];
    }
    inline int xint() {
        int c = xchar(), x = 0, s = 1;
        while (c <= 32) {
            if(!~c) return ed = 1;
            c = xchar();
        }
        if (c == '-') s = -1, c = xchar();
        for (; '0' <= c && c <= '9'; c = xchar()) x = x * 10 + c - '0';
        return x * s;
    }
    inline int xuint() {
        int c = xchar(), x = 0;
        while (c <= 32) c = xchar();
        for (; '0' <= c && c <= '9'; c = xchar()) x = x * 10 + c - '0';
        return x;
    }
    inline void xstring(char *s) {
        int c = xchar();
        while (c <= 32) c = xchar();
        for (; c >  32; c = xchar()) *s++ = c;
        *s = 0;
    }
    inline void wchar(int x) {
        if (wpos == S) fwrite(wbuf, 1, S, stdout), wpos = 0;
        wbuf[wpos++] = x;
    }
    inline void wint(int x) {
        if (x < 0) wchar('-'), x = -x;
        char s[24];
        int n = 0;
        while (x || !n) s[n++] = '0' + x % 10, x /= 10;
        while (n--) wchar(s[n]);
    }
    inline void wstring(const char *s) {
        while (*s) wchar(*s++);
    }
    ~FastIO() {
        if (wpos) fwrite(wbuf, 1, wpos, stdout), wpos = 0;
    }
} io;
```

1.3 Head

```
#include<bits/stdc++.h>
using namespace std;
#define fi first
#define se second
#define mp make_pair
#define pb push_back
#define rep(i, a, b) for(int i=(a); i<(b); i++)
#define per(i, a, b) for(int i=(b)-1; i>=(a); i--)
#define sz(a) (int)a.size()
#define de(a) cout << #a << " = " << a << endl
#define dd(a) cout << #a << " = " << a << " "
#define all(a) a.begin(), a.end()
#define pw(x) (1ll<<(x))
#define endl "\n"
typedef long long ll;
typedef pair<int, int> pii;
typedef vector<int> vi;
typedef double db;

int main() {
   std::ios::sync_with_stdio(false);
   std::cin.tie(0);
// cout << setiosflags(ios::fixed);
// cout << setprecision(3);
   return 0;
}
```

1.4 duipai

```
#!/bin/bash
while true; do
    ./gen > gen.in
    ./sol <gen.in >sol.out
    ./dp <gen.in >dp.out
    if diff sol.out dp.out; then
        printf "AC\n"
```

```
    else
        printf "Wa\n"
        exit 0
    fi
done

// sh duipai.sh
```

2 DP

2.1 DigDP

```
// fill f -1
ll f[];// 自顶向下限制
ll dfs(int pos, ..., bool lim){
   if(pos == -1) return ?;// ...
   if(!lim && ~f[...]) return f[...];
   ll res = 0;
   int up = lim ? dig[pos] : 9;//...
   rep(i,0,up+1) {
      if(..) res += dfs(pos - 1, ..., lim & (i == up));
   }
   if(!lim) f[]=res;
   return res;
}
ll solve(ll x){
   int pos=0;
   while(x) dig[pos++] = x % 10 , x /= 10;
   return dfs(pos-1, ... , 1);
}
```

3 DataStructure

3.1 2DST

```
namespace ST_2D{
   const int N = 1030;
   int LOG[N], P[20], dep1, dep2;
   short st[11][11][N][N];
   void build(int n, int m, short a[][N]){
      rep(i, 0, 11) P[i] = 1<<i;
      rep(i, 2, 1025) LOG[i] = LOG[i>>1]+1;
      for(dep1 = 0; (1<<dep1) < n; dep1++);
      for(dep2 = 0; (1<<dep2) < m; dep2++);
      rep(i, 1, n+1)
         rep(j, 1, m+1)
            st[0][0][i][j] = a[i][j];//modi
      rep(i, 1, n+1)
         rep(j, 1, dep2+1)
            rep(k, P[j], m+1)
               st[0][j][i][k] = max(st[0][j-1][i][k], st[0][j-1][i][k-P[j-1]]);
      rep(i, 1, dep1+1)
         rep(j, P[i], n+1)
            rep(k, 0, dep2+1) //attention to range of k
               rep(l, P[k], m+1)
                  st[i][k][j][l]=max(st[i-1][k][j-P[i-1]][l], st[i-1][k][j][l]);
   }
   int qry(int x1, int y1, int x2, int y2){
      int l1 = LOG[x2-x1+1], l2 = LOG[y2-y1+1];
      int res1 = max(st[l1][l2][x1+P[l1]-1][y1+P[l2]-1], st[l1][l2][x2][y2]);
      int res2 = max(st[l1][l2][x1+P[l1]-1][y2], st[l1][l2][x2][y1+P[l2]-1]);
      return max(res1, res2);
   }
}
```

3.2 2DSegTree

```
// 区域覆盖、标记永久化、标记单调
const int N=1010;
int n,m,q;
struct seg {
    int ma[N<<2], la[N<<2];
    void upd(int L,int R,int c,int l=0,int r=m,int rt=1) {
        ma[rt]=max(ma[rt], c);
        if(L<=l&&r<=R) {
            la[rt]=max(la[rt], c);
            return ;
        }
        int mid=l+r>>1;
        if(L<=mid) upd(L, R, c, l, mid, rt<<1);
        if(R>=mid+1) upd(L, R, c, mid+1, r, rt<<1|1);
    }
    int qry(int L,int R,int l=0,int r=m,int rt=1) {
        int ans=0;
        ans=max(ans, la[rt]);
        if(L<=l&&r<=R) {
            ans=max(ans, ma[rt]);
            return ans;
        }
        int mid=l+r>>1;
        if(L<=mid) ans=max(ans, qry(L, R, l, mid, rt<<1));
        if(R>=mid+1) ans=max(ans, qry(L, R, mid+1, r, rt<<1|1));
        return ans;
    }
};
struct Seg {
    seg ma[N<<2], la[N<<2];
    void upd(int x1,int x2,int y1,int y2,int c,int l=0,int r=n,int rt=1) {
        ma[rt].upd(y1, y2, c);
        if(x1<=l&&r<=x2) {
            la[rt].upd(y1, y2, c);
            return ;
        }
        int mid=l+r>>1;
        if(x1<=mid) upd(x1, x2, y1, y2, c, l, mid, rt<<1);
        if(x2>=mid+1) upd(x1, x2, y1, y2, c, mid+1, r, rt<<1|1);
    }
```

```
    int qry(int x1,int x2,int y1,int y2,int l=0,int r=n,int rt=1) {
        int ans=0;
        ans=max(ans, la[rt].qry(y1, y2));
        if(x1<=l&&r<=x2) {
            ans=max(ans, ma[rt].qry(y1, y2));
            return ans;
        }
        int mid=l+r>>1;
        if(x1<=mid) ans=max(ans, qry(x1, x2, y1, y2, l, mid, rt<<1));
        if(x2>=mid+1) ans=max(ans, qry(x1, x2, y1, y2, mid+1, r, rt<<1|1));
        return ans;
    }
}T;
int main() {
    scanf("%d%d%d",&n,&m,&q);
    while(q--) {
        int d,s,h,x,y;scanf("%d%d%d%d%d",&d,&s,&h,&x,&y);
        int t=T.qry(x, x+d-1, y, y+s-1);
        T.upd(x, x+d-1, y, y+s-1, h+t);
    }
    printf("%d\n",T.qry(0, n, 0, m));
    return 0;
}
```

3.3 CDQ

```
const int N = 200005;
int p1, p2, pos, n, k, nn, ans[N];
struct node{
   int x, y, z, num, ans;
   bool operator == (const node & b) const{
      return x == b.x && y == b.y && z == b.z;
   }
} a[N], tmp[N];
bool cmp(node a, 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(node a, node b){
   //if (a.y != b.y) return a.y < b.y;
   //return a.z < b.z;
   return a.y < b.y;
}
void CDQ(int l, int r){
   if (l == r) {
      a[l].ans = a[l].num - 1;
      return;
   }
   int mid = l + r >> 1;
   CDQ(l, mid); CDQ(mid+1, r);
   pos = l;
   rep(i, mid+1, r+1) {
      while (pos <= mid && a[pos].y <= a[i].y) {
         fen.add(fen.a1, a[pos].z, a[pos].num);
         pos++;
      }
      a[i].ans += fen.sum(fen.a1, a[i].z);
   }
   rep(i, l, pos) fen.add(fen.a1, a[i].z, -a[i].num);
   p1 = l; p2 = mid+1;
   rep(i, l, r+1){
      if (p1 > mid) {tmp[i] = a[p2]; p2++;}
      else if (p2 > r) {tmp[i] = a[p1]; p1++;}
      else if (a[p1].y <= a[p2].y) {tmp[i] = a[p1]; p1++;}
      else  {tmp[i] = a[p2]; p2++;}
   }
   rep(i, l, r+1) a[i] = tmp[i];
}

int main(){
   cin >> n >> k;
   rep(i, 1, n+1) cin >> a[i].x >> a[i].y >> a[i].z;
   sort(a+1, a+n+1, cmp);
   nn = 0;
   rep(i, 1, n+1) {
      if (i > 1 && a[i] == a[i-1]) { a[nn].num++;
      } else {
         a[++nn] = a[i];
         a[nn].num = 1;
      }
   }
   fen.ini(N);
   CDQ(1, nn);
   rep(i, 1, nn+1) ans[a[i].ans] += a[i].num;
   rep(i, 0, n) cout << ans[i] << endl;
   return 0;
}
```

3.4 CartesianTree

```
// desc : bud a cartesion tree from a[0] .. a[n - 1]
// time : O(N)
// !!!! : return rt, a[n] will be rewrite
int ls[N], rs[N];
int cartesionTree(int a[], int n) {
   a[n] = INT_MAX;
   vi v(1, n);
   fill_n(ls, n, -1), fill_n(rs, n, -1);
   rep(i, 0, n) {
      while (a[v.back()] < a[i])
         ls[i] = v.back(), v.pop_back();
      v.pb(rs[v.back()] = i);
   }
   return v[1];
}
```

3.5 Fenwick

```
// index : [1, n]
// time : nlogn
// support : segment add, sum
// !!!! : use before init()!
template<class T>
struct Fenwick{
    static const int N =2e5+7;
    int n;T a1[N],a2[N];
    void ini(int _n){
      fill_n(a1+1,n=_n,0);fill_n(a2+1,n=_n,0);
   }
   void add(T *a,int p,T d) { for(; p<=n; p+=p & -p) a[p]+=d; }
   void add(int l,int r,T d) {
      add(a1, l, d), add(a1, r + 1, -d);
      add(a2, l, d * (l - 1)), add(a2, r + 1, -d * r);
   }
   T sum(T *a,int p) { T r=0; for(; p>=1; p-=p & -p) r+=a[p]; return r; }
   T pre(int p) { return !p ? 0 : sum(a1, p) * p - sum(a2, p);}
   T qry(int l,int r) {return pre(r)-pre(l-1); }
};
```

3.6 IntervalMaximumChangeTimes

```
inline int qry(int L, int R, ll &v, int o, bool spe, int l, int r, int rt) {
  if(L > R) return 0;
  if(!spe) {
    if(ma[rt] < v) return 0;
    if(L <= l && r <= R) {
      if(l == r) return v = ma[rt], 1;
      int mid = l + r >> 1;
      down(l, r, mid, rt);
      if(ma[ls | o] < v) return o ? qry(L, R, v, o, 0, l, mid, ls) : qry(L, R, v, o, 0,
      mid + 1, r, rs);
      int ans = cnt[rt][o] - cnt[ls | o][o] + (o == 0 ? qry(L, R, v, o, 0, l, mid, ls) :
       qry(L, R, v, o, 0, mid + 1, r, rs));
      return v = ma[rt], ans;
    }
  }
  int mid = l + r >> 1, ans = 0;
  down(l, r, mid, rt);
  if(o == 0 && L <= mid) ans += qry(L, R, v, o, 0, l, mid, ls);
  if(R > mid) ans += qry(L, R, v, o, 0, mid + 1, r, rs);
  if(o == 1 && L <= mid) ans += qry(L, R, v, o, 0, l, mid, ls);
  return ans;
}
void up(int l, int r, int rt) {
  ma[rt] = max(ma[ls], ma[rs]);
  rep(o, 0, 2) { ll v = 0; cnt[rt][o] = qry(l, r, v, o, 1, l, r, rt); }
}
```

3.7 KDT

```
// init
typedef int T;// modify
namespace KDT {
  const int N = 1e6 + 7, D = 2;
  const T INF = 1e9 + 7;
  const db al = 0.75;

  int rt,L,top,W,sta[N];
  struct P{ T x[D]; bool operator < (const P &c) const { return x[W] < c.x[W]; } }p[N];
  struct Node{ T mi[D],ma[D]; int son[2],sz; P val; }nd[N];

  void init() { rt = L = top = 0; }
  int newnode() { return top ? sta[top--] : ++L; }
  void up(int k) {
    rep(i, 0, D) {
      nd[k].mi[i]=nd[k].ma[i]=nd[k].val.x[i];
      rep(o, 0, 2) if(nd[k].son[o]) {
   int s = nd[k].son[o];
   nd[k].mi[i] = min(nd[k].mi[i], nd[s].mi[i]);
   nd[k].ma[i] = max(nd[k].ma[i], nd[s].ma[i]);
      }
    }
    nd[k].sz = 1; rep(i, 0, 2) nd[k].sz += nd[nd[k].son[i]].sz;
  }
  int build(int l,int r,int w) {
    if(l > r) return 0;
    int mid = l + r >> 1, k = newnode();
    W=w,nth_element(p+l,p+mid,p+r+1),nd[k].val=p[mid];
    nd[k].son[0] = build(l,mid-1, (w + 1) % D);
    nd[k].son[1] = build(mid+1,r, (w + 1) % D);
    up(k); return k;
  }
  void pia(int k,int &cnt) {
    if(nd[k].son[0]) pia(nd[k].son[0],cnt);
    p[++cnt]=nd[k].val,sta[++top]=k;
    if(nd[k].son[1]) pia(nd[k].son[1],cnt);
  }
  void check(int &k,int w) {
    bool o = 0;
    rep(i, 0, 2) if(al * nd[k].sz < nd[nd[k].son[i]].sz) o = 1;
    if(o) { int cnt = 0; pia(k,cnt),k = build(1,cnt,w); }
  }
  void ins(P p,int &k,int w) {
    if(!k) {k=newnode(),nd[k].val=p,nd[k].son[0]=nd[k].son[1]=0,up(k);return;}
    ins(p, nd[k].son[nd[k].val.x[w] < p.x[w]], (w + 1) % D);
    up(k),check(k,w);
  }
  // 抄上面这部分就好了，下面部分是视具体题目定的
  // 最近点
  T dis(P p,int k) {
    T ans = 0; rep(i, 0, D) ans += max(0, p.x[i] - nd[k].ma[i]) + max(0, nd[k].mi[i] - p
    .x[i]); // modify
    return ans;
  }
  T dis(P a, P b) {
    T ans = 0; rep(i, 0, D) ans += abs(a.x[i] - b.x[i]);
```

```
    return ans;
  }
  void qry(P p, int k, T &ans) {
    ans=min(ans,dis(p,nd[k].val));
    int ls = nd[k].son[0], rs = nd[k].son[1];
    T dl = ls ? dis(p, ls) : INF;
    T dr = rs ? dis(p, rs) : INF;
    if(dl > dr) swap(dl, dr), swap(ls, rs);
    if(dl<ans) qry(p,ls,ans);
    if(dr<ans) qry(p,rs,ans);
  }
  // 矩形区域的最大值（伪代码）
  void qry(int u, int &ans) {
    if(no_in || ma < ans) return ;
    if(all_in) { ans = max(ans, ma); return ; }
    if(u_in) ans = max(ans, u_val);
    rep(i, 0, 2) if(nd[u].son[i]) qry(nd[u].son[i], ans);
  }
  // 距离点 u 第 k 远
  priority_queue<ll> ans;
  void init() {
    while(!ans.empty()) ans.pop();
    rep(i, 0, k) ans.push(1);
  }
  ll sqr(int x) { return 1ll * x * x; }
  ll Dis(P p, int u) {
    ll ans = 0; rep(d, 0, D) ans += max(sqr(nd[u].mi[d] - p.x[d]), sqr(nd[u].ma[d] - p.x
    [d]));
    return ans;
  }
  void qry(P p, int u) {
    ll dis = 0; rep(d, 0, D) dis += sqr(nd[u].val.x[d] - p.x[d]);
    ans.push(-dis), ans.pop();
    int ls = nd[u].son[0], rs = nd[u].son[1];
    ll dl = ls ? Dis(p, ls) : -1;
    ll dr = rs ? Dis(p, rs) : -1;
    if(dl > dr) swap(dl, dr), swap(ls, rs);
    if(dr > -ans.top()) qry(p, rs);
    if(dl > -ans.top()) qry(p, ls);
  }
}
```

3.8 LCT

```
struct Node { int val, sum, fa, son[2]; bool rev; };
struct LCT {
    static const int N = ::N;
    Node nd[N]; int sta[N];
    // if(no root) return 1
    bool nrt(int x) {
        int fa = nd[x].fa;
        return nd[fa].son[0] == x || nd[fa].son[1] == x;
    }
    void up(int x) {
        if(!x) return ;
        int ls = nd[x].son[0], rs = nd[x].son[1];
        nd[x].sum = nd[ls].sum + nd[rs].sum + nd[x].val;
    }
    void gao(int x) {
        if(!x) return ;
        nd[x].rev ^= 1, swap(nd[x].son[0], nd[x].son[1]);
    }
    void down(int x) { if(nd[x].rev) gao(nd[x].son[0]), gao(nd[x].son[1]), nd[x].rev =
    0; }
    int id(int u) { return nd[nd[u].fa].son[1] == u; }
    void rot(int x) {
        int y = nd[x].fa, z = nd[y].fa;
        int l = id(x), r = (l ^ 1), s = nd[x].son[r];
        if(nrt(y)) nd[z].son[id(y)] = x; nd[x].son[r] = y; nd[y].son[l] = s;
        if(s) nd[s].fa = y; nd[y].fa = x; nd[x].fa = z;
        up(y), up(x);
    }
    void splay(int x) {
        int top = 0;
        for(int i = x; ; i = nd[i].fa) {
            sta[++top] = i;
            if(!nrt(i)) break;
        }
        while(top) down(sta[top--]);
        while(nrt(x)) {
            int y = nd[x].fa;
            if(nrt(y)) (id(x) ^ id(y)) ? rot(x) : rot(y);
            rot(x);
        }
    }
    void access(int x) { for(int y = 0; x ; y = x, x = nd[x].fa) splay(x), nd[x].son[1]
    = y, up(x); }
    // 换根
    void makeroot(int x) { access(x); splay(x); gao(x); }
    // 找根
    int findroot(int x) {
        access(x); splay(x);
        while(nd[x].son[0]) down(x), x = nd[x].son[0];
        splay(x);
        return x;
    }
    // 加边
    void link(int x, int y) {
        makeroot(x);
        if(findroot(y) != x) nd[x].fa = y;
    }
    // 删边
    void cut(int x, int y) {
        makeroot(x);
        if(findroot(y) == x && nd[y].fa == x && !nd[y].son[0]) nd[y].fa = nd[x].son[1] =
         0, up(x);
    }
    // nd[y]: 路径信息
    void path(int x, int y) { makeroot(x); access(y); splay(y); }
    // 单点修改
```

```
    void upd(int x, int c) { splay(x); nd[x].val = c; up(x); }
}T;
```

3.9 Rope

```
#include <ext/rope>
using namespace __gnu_cxx;

//index : [0..sz(rp))
rope<char> rp;

// 在任意处插入字符数组
rp.push_back(ch);              // 在末尾插入
rp.insert(cur, 字符数组 );    // 在 cur 处插入字符数组

// 删除任意片段
rp.erase(cur, len);            // 删除 cur 开始的 len 个字符

rp.replace(cur, 字符数组 );   // 删除 cur 处的字符，换成字符数组

// 取出任意片段
rp.copy(cur, len, 字符数组 ); // 复制 cur 处开始的 len 个字符到字符数组
rp.at(cur);                    // 取第 cur 个字符
rp[cur];                       // 同上
rp.substr(cur, len);           // 提取从 cur 处开始的 len 个字符

// 可持久化
rp[i] = rp[i - 1];             // 可持久化, O(1), 直接拷贝根节点

/*

* 一）翻转操作
*  1. 维护一正一反两个 rope
*  2. 翻转等价于交换两个子串

* 二）区间循环位移
*  1. 拆成多个子串连在一起

* 三）区间 a -> b, b -> c, c -> d ... z -> a
*  1. 维护 26 个 rope

*/
```

3.10 ST

```
// [0,n)
// 实现不同功能请谨慎复用
// 求下标最好用 pair 存
struct ST{
   static const int N = 101010;
   int a[20][N], lg[N];
   void build(int *v, int n){
      rep(i, 2, n + 1) lg[i] = lg[i >> 1] + 1;
      rep(i, 0, n) a[0][i] = v[i];
      rep(i, 1, lg[n] + 1) rep(j, 0, n - (1 << i) + 1) {
         a[i][j] = max(a[i - 1][j], a[i - 1][j + (1 << i >> 1)]);
      }
   }
   int qry(int l, int r){
      if(l > r) swap(l, r);
      int i = lg[r - l + 1];
      return max(a[i][l] , a[i][r + 1 - (1 << i)]);
   }
};
```

3.11 SegIntervalMax

```
// O(nlogn)
// 区间取 max, 区间求和
struct Seg {
#define ls rt << 1
#define rs ls | 1
   static const int N = ::N << 2;
   ll sum[N];
   int mi[N][2], cnt[N];
   void up(int rt) {
      sum[rt] = sum[ls] + sum[rs];
      rep(i, 0, 2) mi[rt][i] = min(mi[ls][i], mi[rs][i]);
      cnt[rt] = 0;
      rep(i, 0, 2) {
         if(mi[rt][0] == mi[ls | i][0]) cnt[rt] += cnt[ls | i];
         else mi[rt][1] = min(mi[rt][1], mi[ls | i][0]);
      }
   }
   void build(int l, int r, int rt) {
      if(l == r) {
         sum[rt] = mi[rt][0] = l; //modify
         mi[rt][1] = inf;
         cnt[rt] = 1;
         return ;
      }
      int mid = l + r >> 1;
      build(l, mid, ls);
      build(mid + 1, r, rs);
      up(rt);
   }
   void gao(int rt, int c) {
      if(c <= mi[rt][0]) return ;
      sum[rt] += 1ll * cnt[rt] * (c - mi[rt][0]);
      mi[rt][0] = c;
   }
   void down(int rt) {
      gao(ls, mi[rt][0]);
      gao(rs, mi[rt][0]);
   }
   void upd(int L, int R, int c, int l, int r, int rt) {
      if(L > R) return ;
      if(L <= l && r <= R && c < mi[rt][1]) {
         gao(rt, c);
```

```
        return ;
      }
      int mid = l + r >> 1;
      down(rt);
      if(L <= mid) upd(L, R, c, l, mid, ls);
      if(R > mid) upd(L, R, c, mid + 1, r, rs);
      up(rt);
   }
}seg;
```

3.12 Splay

```
// init
// id starts from 1
// if go to vertex p, must splay(p)
struct Node { int val, fa, son[2], cnt, sz; bool rev; };
struct Splay {
   static const int N = ::N;
   int rt, L; Node nd[N];
   int newnode(int c, int fa = 0, int o = 0) {
      nd[++L].fa = fa;
      nd[fa].son[o] = L;
      nd[L].val = c;
      nd[L].son[0] = nd[L].son[1] = nd[L].rev = 0;
      nd[L].cnt = nd[L].sz = 1;
      return L;
   }
   void init(int n) { rt = L = 0; }
   void gao(int u) {
      if(!u) return ;
      nd[u].rev ^= 1, swap(nd[u].son[0], nd[u].son[1]);
   }
   void down(int u) {
      if(nd[u].rev) gao(nd[u].son[0]), gao(nd[u].son[1]), nd[u].rev = 0;
   }
   void up(int u) {
      if(!u) return ;
      int ls = nd[u].son[0], rs = nd[u].son[1];
      nd[u].sz = nd[ls].sz + nd[rs].sz + nd[u].cnt;
   }
   int id(int u) { return nd[nd[u].fa].son[1] == u; }
   void rot(int x) {
      int y = nd[x].fa, z = nd[y].fa;
      int l = id(x), r = (l ^ 1), s = nd[x].son[r];
      if(z) nd[z].son[id(y)] = x; nd[x].son[r] = y; nd[y].son[l] = s;
      if(s) nd[s].fa = y; nd[y].fa = x; nd[x].fa = z;
      up(y), up(x);
   }
   void splay(int x, int g = 0) {
      while(nd[x].fa != g) {
         int y = nd[x].fa, z = nd[y].fa;
         if(z != g) (id(x) ^ id(y)) ? rot(x) : rot(y);
         rot(x);
      }
      if(!g) rt = x;
   }
}T;
```

3.13 fhqTreap

```
// init
// id starts from 1
// 不要修改 0 节点的值
struct Node { int val, cnt, sz, ls, rs; ll r; bool rev;};
struct fhqTreap {
   static const int N = ::N;
   int rt, L; Node nd[N];
   void init() { rt = L = 0; srand(time(0)); }
   ll Rand() { return ((rand() * 1ll << 32) ^ (rand() * 1ll << 16) ^ rand()); }
   int newnode(int c) {
      nd[++L].r = Rand();
      nd[L].val = c;
      nd[L].cnt = nd[L].sz = 1;
      nd[L].ls = nd[L].rs = nd[L].rev = 0;
      return L;
   }
   void up(int x) {
      if(!x) return ;
      int ls = nd[x].ls, rs = nd[x].rs;
      nd[x].sz = nd[ls].sz + nd[rs].sz + nd[x].cnt;
   }
   void gao(int x) {
      if(!x) return ;
      nd[x].rev ^= 1, swap(nd[x].ls, nd[x].rs);
   }
   void down(int x) {
      if(nd[x].rev) gao(nd[x].ls), gao(nd[x].rs), nd[x].rev = 0;
   }
   // u -> (<= c) (> c)
   void splitc(int u, int c, int &x, int &y) {
      if(u) {
         down(u);
         if(nd[u].val <= c) x = u, splitc(nd[u].rs, c, nd[u].rs, y);
         else y = u, splitc(nd[u].ls, c, x, nd[u].ls);
         up(u);
      } else {
         x = y = 0;
      }
   }
   // u -> (1 ~ k) (k+1 ~ L)
   // !!!: nd[].cnt == 1
   void splitk(int u, int k, int &x, int &y) {
      if(u) {
         down(u);
         int sz = nd[nd[u].ls].sz;
         if(sz < k) x = u, splitk(nd[u].rs, k - sz - 1, nd[u].rs, y);
         else y = u, splitk(nd[u].ls, k, x, nd[u].ls);
         up(u);
      } else {
         x = y = 0;
```

```
    }
  }
  int merge(int x, int y) {
    if(x && y) {
      if(nd[x].r < nd[y].r) { down(x), nd[x].rs = merge(nd[x].rs, y), up(x); return x
      ; }
      else { down(y), nd[y].ls = merge(x, nd[y].ls), up(y); return y; }
    } else return x + y;
  }
}T;
```

3.14 lcSegTree

```
// need init
// 1. use id
// 2. init mi/nd as max/min val
struct Node {
    ll k, b;
    Node() : k(0), b(0) {}
    Node(ll k, ll b) : k(k), b(b) {}
    ll getf(int x) const {
        return k * x + b;
    }
};
struct Seg {
#define ls rt << 1
#define rs ls | 1
    static const int N = ::N << 2;
    Node nd[N], mi[N];
    void _upd(Node k, int l, int r, int rt) {
        int mid = l + r >> 1;
        if(k.getf(v[mid]) > nd[rt].getf(v[mid])) swap(k, nd[rt]);
        if(l == r) return ;
        if(min(nd[rt].getf(v[l]), nd[rt].getf(v[r])) >= max(k.getf(v[l]), k.getf(v[r])))
         return ;
        if(nd[rt].k > k.k) _upd(k, l, mid, ls);
        else _upd(k, mid + 1, r, rs);
    }
    void _min(Node k, int l, int r, int rt) {
        int mid = l + r >> 1;
        if(k.getf(v[mid]) < mi[rt].getf(v[mid])) swap(k, mi[rt]);
        if(l == r) return ;
        if(max(mi[rt].getf(v[l]), mi[rt].getf(v[r])) <= min(k.getf(v[l]), k.getf(v[r])))
         return ;
        if(mi[rt].k <= k.k) _min(k, l, mid, ls);
        else _min(k, mid + 1, r, rs);
    }
    void upd(int L, int R, Node c, int l, int r, int rt) {
        if(L > R) return ;
        if(L <= l && r <= R) {
            _upd(c, l, r, rt);
            _min(c, l, r, rt);
            return ;
        }
        int mid = l + r >> 1;
        if(L <= mid) upd(L, R, c, l, mid, ls);
        if(R > mid) upd(L, R, c, mid + 1, r, rs);
    }
    ll qry(int p, int l, int r, int rt) {
        ll ans = max(abs(nd[rt].getf(v[p])), abs(mi[rt].getf(v[p])));
        if(l == r) return ans;
        int mid = l + r >> 1;
        if(p <= mid) ans = max(ans, qry(p, l, mid, ls));
        else ans = max(ans, qry(p, mid + 1, r, rs));
        return ans;
    }
}seg;
```

3.15 perTreap

```
// init
// id starts from 1
// 不要修改 0 节点的值
struct Node { int val, cnt, sz, ls, rs; ll r, sum; bool rev; };
struct fhqTreap {
  static const int N = 3e7;
  int rt[::N], L; Node nd[N];
  void init() { L = 0; srand(time(0)); }
  ll Rand() { return ((rand() * 1ll << 32) ^ (rand() * 1ll << 16) ^ rand()); }
  int newnode(int c) {
    nd[++L].r = Rand();
    nd[L].val = nd[L].sum = c;
    nd[L].cnt = nd[L].sz = 1;
    nd[L].ls = nd[L].rs = nd[L].rev = 0;
    return L;
  }
  int newcopy(int x) {
    nd[++L] = nd[x];
    return L;
  }
  void up(int x) {
    if(!x) return ;
    int ls = nd[x].ls, rs = nd[x].rs;
    nd[x].sz = nd[ls].sz + nd[rs].sz + nd[x].cnt;
    nd[x].sum = nd[ls].sum + nd[rs].sum + nd[x].val;
  }
  void gao(int &x) {
    if(!x) return ;
    x = newcopy(x);
    nd[x].rev ^= 1, swap(nd[x].ls, nd[x].rs);
  }
  void down(int x) {
    if(nd[x].rev) gao(nd[x].ls), gao(nd[x].rs), nd[x].rev = 0;
  }
  // u -> (<= c) (> c)
  void splitc(int u, int c, int &x, int &y) {
    if(u) {
      u = newcopy(u), down(u);
      if(nd[u].val <= c) x = u, splitc(nd[u].rs, c, nd[u].rs, y);
      else y = u, splitc(nd[u].ls, c, x, nd[u].ls);
```

```
      up(u);
    } else {
      x = y = 0;
    }
  }
  // u -> (1 ~ k) (k+1 ~ L)
  // !!!: nd[].cnt == 1
  void splitk(int u, int k, int &x, int &y) {
    if(u) {
      u = newcopy(u), down(u);
      int sz = nd[nd[u].ls].sz;
      if(sz < k) x = u, splitk(nd[u].rs, k - sz - 1, nd[u].rs, y);
      else y = u, splitk(nd[u].ls, k, x, nd[u].ls);
      up(u);
    } else {
      x = y = 0;
    }
  }
  // sometimes do not need to newcopy
  int merge(int x, int y) {
    if(x && y) {
      if(nd[x].r < nd[y].r) { x = newcopy(x), down(x), nd[x].rs = merge(nd[x].rs, y),
       up(x); return x; }
      else { y = newcopy(y), down(y), nd[y].ls = merge(x, nd[y].ls), up(y); return y;
       }
    } else return x + y;
  }
}T;
```

3.16 动态 k 大

```
// zoj 2112 动态区间 k 大
const int N = 50505, M = 10101;
int n, m, a[N], rt[N<<1];
vi V, add, sub;
inline int rk(int x) { return lower_bound(all(V), x) - V.begin(); }
struct Q {
  bool op;
  int a, b, k;
}q[M];
struct Seg {
  static const int N = 2500005;//(::N + 32 * ::M) * 16;
  int cntn, cnt[N], ls[N], rs[N];
  void init() { fill_n(rt+1, n, cntn = 0); }
  void upd(int pre, int &now, int p, int c, int l, int r) {
    now = ++cntn;
    cnt[now] = cnt[pre] + c;
    ls[now] = ls[pre];
    rs[now] = rs[pre];
    if(l == r) return ;
    int mid = l+r>>1;
    if(p<=mid) upd(ls[pre], ls[now], p, c, l, mid);
    else upd(rs[pre], rs[now], p, c, mid+1, r);
  }
  int qry(int L, int R, int k, int l, int r) {
    if(l == r) return l;
    int mid = l+r>>1;
    int lc = 0;
    for(auto i : add) lc += cnt[ls[i]];
    for(auto i : sub) lc -= cnt[ls[i]];
    if(lc>=k) {
      rep(i, 0, sz(add)) add[i] = ls[add[i]];
      rep(i, 0, sz(sub)) sub[i] = ls[sub[i]];
      return qry(L, R, k, l, mid);
    } else {
      rep(i, 0, sz(add)) add[i] = rs[add[i]];
      rep(i, 0, sz(sub)) sub[i] = rs[sub[i]];
      return qry(L, R, k-lc, mid+1, r);
    }
  }
}seg;
struct Fenwick {
#define lb(x) ((x)&(-x))
  void init() { fill_n(rt+1+n, n, 0); }
  void upd(int x, int p, int c) {
    for(; x<=n; x+=lb(x)) seg.upd(rt[x+n], rt[x+n], p, c, 0, sz(V)-1);
  }
  int qry(int l, int r, int k) {
    add.clear();sub.clear();
    add.pb(rt[r]);sub.pb(rt[l-1]);
    int x = r;
    for(; x; x^=lb(x)) add.pb(rt[n+x]);
    x = l-1;
    for(; x; x^=lb(x)) sub.pb(rt[n+x]);
    return seg.qry(l, r, k, 0, sz(V)-1);
  }
}fw;
int main() {
  int T; cin >> T;
  while(T--) {
    cin >> n >> m;
    V.clear(); seg.init(); fw.init();
    ///read
    rep(i, 1, n+1) cin >> a[i], V.pb(a[i]);
    rep(i, 1, m+1) {
      string s;
      cin >> s >> q[i].a >> q[i].b;
      q[i].op = (s[0]=='Q');
      if(s[0]=='Q') cin >> q[i].k;
      else V.pb(q[i].b);
    }
    ///solve
    sort(all(V));
    V.erase(unique(all(V)), V.end());
    rep(i, 1, n+1) seg.upd(rt[i-1], rt[i], rk(a[i]), 1, 0, sz(V)-1);
    rep(i, 1, m+1) {
      if(q[i].op) { cout << V[fw.qry(q[i].a, q[i].b, q[i].k)] << endl;
      } else {
        int p = q[i].a, c = q[i].b;
        fw.upd(p, rk(a[p]), -1);
```

```
            fw.upd(p, rk(a[p] = c), 1);
        } } }
  return 0; }
```

3.17 动态树上路径 k 大

```
int n, m, L, dfn, val[N], rt[N], cnt[M], ls[M], rs[M], st[N], ed[N], fw[N], par[N];
pair<int, pii> Q[N]; vi V, res[2], g[N]; LCARMQ R;
int F(int x) { return lower_bound(all(V), x) - V.begin() + 1; }
// seg
void upd(int &now, int pre, int p, int c, int l, int r) {
  now = ++L;
  cnt[now] = cnt[pre] + c;
  ls[now] = ls[pre];
  rs[now] = rs[pre];
  if(l == r) return ;
  int mid = l + r >> 1;
  if(p <= mid) upd(ls[now], ls[pre], p, c, l, mid);
  else upd(rs[now], rs[pre], p, c, mid + 1, r);
}
int qry(int k, int l, int r) {
  if(l == r) return l;
  int mid = l + r >> 1;
  int cntr = 0;
  rep(i, 0, sz(res[0])) cntr += cnt[rs[res[0][i]]];
  rep(i, 0, sz(res[1])) cntr -= cnt[rs[res[1][i]]];
  if(cntr >= k) {
    rep(i, 0, sz(res[0])) res[0][i] = rs[res[0][i]];
    rep(i, 0, sz(res[1])) res[1][i] = rs[res[1][i]];
    return qry(k, mid + 1, r);
  } else {
    rep(i, 0, sz(res[0])) res[0][i] = ls[res[0][i]];
    rep(i, 0, sz(res[1])) res[1][i] = ls[res[1][i]];
    return qry(k - cntr, l, mid);
  }
}
// build 主席树
void dfs(int u, int fa) {
  st[u] = ++dfn;
  par[u] = fa;
  upd(rt[u], rt[fa], F(val[u]), 1, 1, sz(V));
  rep(i, 0, sz(g[u])) if(g[u][i] != fa) dfs(g[u][i], u);
  ed[u] = dfn;
}
// fenwick
void upd(int p, int o, int c) { for( ; p <= n; p += lb(p)) upd(fw[p], fw[p], c, o, 1, sz
(V)); }
void upd(int u, int o) {
  res[o].pb(rt[u]);
  int p = st[u]; for( ; p; p ^= lb(p)) res[o].pb(fw[p]);
}
void solve() {
  R.Build(g);
  dfs(1, 0);
  rep(i, 1, m + 1) {
    if(!Q[i].fi) {
      int p = Q[i].se.fi, c = Q[i].se.se;
      upd(st[p], -1, F(val[p]));
      upd(ed[p] + 1, 1, F(val[p]));
      val[p] = c;
      upd(st[p], 1, F(val[p]));
      upd(ed[p] + 1, -1, F(val[p]));
    } else {
      rep(o, 0, 2) res[o].clear();
      int a = Q[i].se.fi, b = Q[i].se.se, k = Q[i].fi;
      int c = R.lca(a, b), d = par[c];
      upd(a, 0); upd(b, 0);
      upd(c, 1); upd(d, 1);
      cout << V[qry(k, 1, sz(V)) - 1] << endl;
    } } }
```

3.18 常见转化

```
/*
 * 单点修改，区间查询 -> 单点修改，前缀查询 -> 后缀修改，单点查询
 */
```

3.19 覆盖大于 k 次的矩形面积

```
/*
 * 这里是覆盖次数大于 1 次的
 */
struct Seg {
#define ls rt << 1
#define rs ls | 1
   static const int N = ::N << 2;
   int la[N], len[2][N];
   void up(int rt, int l, int r) {
      if(la[rt] >= 2) {
         len[0][rt] = r - l + 1;
         len[1][rt] = r - l + 1;
      } else if(la[rt] >= 1) {
         len[0][rt] = r - l + 1;
         len[1][rt] = (l == r) ? 0 : len[0][ls] + len[0][rs];
      } else {
         len[0][rt] = (l == r) ? 0 : len[0][ls] + len[0][rs];
         len[1][rt] = (l == r) ? 0 : len[1][ls] + len[1][rs];
      }
   }
   void upd(int L, int R, int c, int l, int r, int rt) {
      if(L <= l && r <= R) {
         la[rt] += c;
         up(rt, l, r);
         return ;
      }
      int mid = l + r >> 1;
      if(L <= mid) upd(L, R, c, l, mid, ls);
      if(R > mid) upd(L, R, c, mid + 1, r, rs);
      up(rt, l, r);
```

```
    }
}seg;
```

3.20 高维偏序

```
// 如果有比较快O2, 不然可能比较慢要手写bitset
namespace PX{
    const int N = :: N, M = sqrt(N) + 5;
    const int K = 7;
    int n, k, B, pos[K][N];
    bitset<N> s[K][M];
    vector<pii> V[K];

    struct node{
        int d[K];
    } a[N];

    void init(int _n, int _k) {
        n = _n; k = _k;
        rep(i, 1, n+1) rep(j, 0, k) cin >> a[i].d[j];
        rep(i, 1, n+1) rep(j, 0, k) V[j].pb(mp(a[i].d[j], i));
        rep(j, 0, k) sort(all(V[j]));
        rep(i, 1, n+1)
            rep(j, 0, k)
                a[i].d[j] = lower_bound(all(V[j]), mp(a[i].d[j], i)) - V[j].begin(), pos[j][
                a[i].d[j]] = i;
        B = sqrt(n);
        rep(j, 0, k) {
            bitset<N> tmp; int id = 1;
            rep(i, 0, n) {
                tmp.set(pos[j][i]);
                if (i == id * B - 1) s[j][id++] = tmp;
            }
        }
    }
    int qry(node a) {
        bitset<N> ans; ans.set();
        rep(j, 0, k) {
            int ed = lower_bound(all(V[j]), mp(V[j][a.d[j]].fi, n+1)) - V[j].begin() - 1;
            bitset<N> tmp;int id = ed / B, st = id ? (id - 1) * B : 0;
            if (id) tmp = s[j][id - 1];
            rep(i, st, ed+1) tmp[pos[j][i]] = 1;
            ans &= tmp;
        }
        return ans.count();
    }
}
```

4 Game

4.1 Game

```
// 威佐夫博弈
//      * 两堆物品，个数 (n, m)(n <= m) ，两人轮流从某一堆拿任意数量的物品或同时从两堆中取同样多的物品，每次至少一个，不能操作的人败。
//      * 必败态： (m - n) * (1 + sqrt5) / 2 == n
// 威佐夫博弈扩展
//      * 两堆物品，个数 (n, m)(n <= m) ，两人轮流从某一堆拿任意数量的物品或同时从两堆中取绝对值 <=k 的物品，每次至少一个，不能操作的人败。
//      * 必败态：
//          * d = k + 1, t^2 + (d - 2) * t - d = 0 -> 解出 t
//          * 必败： (m - n) / d * t == n
// 博弈fib
//      * 一堆石子，两人轮流取。先手不能在第一次取光，之后可以取的石子数介于 1 到对手刚取的石子数的两倍之间（左闭右闭），不能操作的人败。
//      * 必败态：石子个数是 fib 数
```

5 Geometry

5.1 GeoAdd

```
/*
 * 平面图欧拉定理：V + F - E = 2
 */
 bool cmp(const pii &a, const pii &b) { // 级角排序
    int o = a > pii(0, 0), t = b > pii(0, 0);
    if(o != t) return o < t;
    return det(a, b) > 0;
}
bool isSSr(const L &a, const L &b){ // 线段规范相交
    db c1 = det(a.t - a.s, b.s - a.s), c2 = det(a.t - a.s, b.t - a.s),
       c3 = det(b.t - b.s, a.s - b.s), c4 = det(b.t - b.s, a.t - b.s);
    return sign(c1) * sign(c2) < 0 && sign(c3) * sign(c4) < 0;
}
bool isSS(L a,L b){ // 线段不规范相交
    db c1 = det(a.t - a.s, b.s - a.s), c2 = det(a.t - a.s, b.t - a.s),
       c3 = det(b.t - b.s, a.s - b.s), c4 = det(b.t - b.s, a.t - b.s);
    return sign(c1) * sign(c2) <= 0 && sign(c3) * sign(c4) <= 0 &&
       sign(max(a.s.x, a.t.x) - min(b.s.x, b.t.x)) >= 0 &&
       sign(max(b.s.x, b.t.x) - min(a.s.x, a.t.x)) >= 0 &&
       sign(max(a.s.y, a.t.y) - min(b.s.y, b.t.y)) >= 0 &&
       sign(max(b.s.y, b.t.y) - min(a.s.y, a.t.y)) >= 0;
}
db disSS(L a, L b){ // 线段到线段距离
    if(isSS(a, b)) return 0;
    return min(min(disToSeg(b, a.s), disToSeg(b, a.t)), min(disToSeg(a, b.s), disToSeg(a,
     b.t)));
}
// 判断直线线段是否相交（端点也算）
bool isLS(P a1, P a2, P b1, P b2) {
     db c1 = det(a2 - a1, b1 - a1), c2 = det(a2 - a1, b2 - a1);
     return sign(c1) * sign(c2) <= 0;
}
P outC(P A, P B, P C) { // 外心
    P b = B - A, c = C - A;
    db dB = b.len2(), dC = c.len2(), d = 2 * det(b, c);
    return A - P(b.y * dC - c.y * dB, c.x * dB - b.x * dC) / d;
```

```
}
bool isconvex(vector<P> A) { // 判断是否是凸包逆时针
   bool ok = 1;
   int n = sz(A);
   rep(i, 0, 2) A.pb(A[i]);
   rep(i, 0, n) ok &= det(A[i + 1] - A[i], A[i + 2] - A[i]) >= 0;
   return ok;
}
db diameter(vector<P> A) { // 求凸包最远点对
   int n = sz(A);
   if(n <= 1) return 0;
   int l = 0, r = 0;
   rep(i, 1, n) (A[i] < A[l]) && (l = i), (A[r] < A[i]) && (r = i);
   db res = (A[l]-A[r]).len();
   int i = l, j = r;
   do (++(det(A[(i + 1) % n]- A[i], A[(j + 1) % n] - A[j]) >= 0 ? j : i)) %= n,
      res = max(res, (A[i] - A[j]).len());
   while(i != l || j != r);
   return res;
}
// sz(A) <= 100,000
namespace NearestPoints { // 点集中最近点对
   db solve(int l, int r, vector<P> &p) {
      if(l == r) return 1e100;
      int m = l + r >> 1;
      db Xm = p[m].x, lim = min(solve(l, m, p), solve(m + 1, r, p));
      inplace_merge(p.begin() + l, p.begin() + m + 1, p.begin() + r + 1, [&](P a, P b){
      return a.y < b.y;});
      vector<P> V;
      rep(i, l, r + 1) if(fabs(p[i].x - Xm) <= lim) V.pb(p[i]);
      rep(i, 0, sz(V)) rep(j, i + 1, sz(V)) {
         if(fabs(V[j].y - V[i].y) >= lim) break;
         T dis = (V[i] - V[j]).len();
         lim = min(lim, dis);
      }
      return lim;
   }
   db solve(vector<P> A) {
      sort(all(A), [&](P a, P b){return a.x < b.x;});
      return solve(0, sz(A) - 1, A);
   }
}
// 注意相等关系
// 相离4: 外切3: 相交2: 内切1: 内含0:
int relCC(C A, C B) { // 两圆关系
   db dis = (A.o - B.o).len();
   if(sign(dis - (A.r + B.r)) == 1) return 4;
   if(sign(dis - (A.r + B.r)) == 0) return 3;
   if(sign(dis - fabs(A.r - B.r)) == 1) return 2;
   if(sign(dis - fabs(A.r - B.r)) == 0) return 1;
   return 0;
}
vector<P> tanCC(const C &c1, const C &c2) { // 求圆与圆的切点
   vector<P> res;
   db dis = (c1.o - c2.o).len();
   if(sign(dis - (c1.r + c2.r)) == 0) {
      res.pb(c1.o + (c2.o - c1.o) * c1.r / (c1.r + c2.r));
   }
   if(sign(dis - fabs(c1.r - c2.r) == 0)) {
      res.pb(c1.o + (c2.o - c1.o) * c1.r / (c1.r - c2.r));
   }
   return res;
}
db rad(P p1, P p2) { return atan2l(det(p1, p2), dot(p1, p2)); } // p1 与 p2 的夹角，有方向
db areaCT(db r,P s,P t) { // 求圆与三角形交面积，需要除2
   P p1, p2;
   bool f = isCL(C(P(0, 0), r), L(s, t), p1, p2);
   if(!f) return r * r * rad(s, t);
   bool b1 = sign(s.len2() - r * r) == 1 , b2 = sign(t.len2() - r * r) == 1;
   if(b1 && b2) {
      if(sign(dot(s - p1, t - p1)) <= 0 && sign(dot(s - p2, t - p2) <= 0))
         return r * r * (rad(s, p1) + rad(p2, t)) + det(p1, p2);
      else return r * r * rad(s, t);
   } else if(b1) return r * r * rad(s, p1) + det(p1, t);
   else if(b2) return r * r * rad(p2, t) + det(s, p2);
   return det(s, t);
}
db areaCPoly(C c, vector<P> p) { // 求圆与多边形交面积
   int n = sz(p);
   db ans = 0;
   rep(i, 0, n) {
      P u = p[i], v = p[(i + 1) % n];
      ans += areaCT(c.r, u - c.o, v - c.o);
   }
   return fabs(ans) / 2;
}
C Mincir(P *p,int n){ // ? 最小圆覆盖
   random_shuffle(p , p + n);
   P o = p[0];db r = 0;
   rep(i,1,n) {
      if(sgn(abs(o-p[i])-r) <= 0) continue;
      o = p[i] , r = 0;
      rep(j,0,i) {
         if(sgn(abs(o-p[j])-r) <= 0) continue;
         o = (p[i] + p[j]) / 2 , r = abs(o-p[j]);
         rep(k,0,j) {
            if(sgn(abs(o-p[k])-r) <= 0) continue;
            o = outC(p[i],p[j],p[k]) , r = abs(o-p[k]);
         }}}
   return C(o,r);
}
namespace CircleIntersection{ // ?
   struct E{
      P p;T ang;int delta;
      E(){} E(P p,T ang,int delta):p(p),ang(ang),delta(delta){}
      bool operator < (const E&b) const {return ang<b.ang;}
   };
   bool overlap(C a,C b) {return sgn(a.r-b.r-abs(a.o-b.o))>=0;}
   void solve(C *c,int n,T *ans) {
```

```
    memset(ans , 0 , sizeof(T) * (n + 1));
    rep(i,0,n) {
       int cnt=1;
       vector<E> evt;
       rep(j,0,i) if(c[i]==c[j]) cnt++;
       rep(j,0,n) if(j!=i&&!(c[i]==c[j])&&overlap(c[j],c[i])) cnt++;
       rep(j,0,n) if(j!=i){
          vector<P> pts=insCC(c[i],c[j]);
          if(sz(pts)) {
             T a[2];
             rep(j,0,2) a[j]=(pts[j]-c[i].o).arg();
             evt.pb(E(pts[0],a[0],1));
             evt.pb(E(pts[1],a[1],-1));
             cnt += a[0] > a[1];
          }
       }
       if(!sz(evt)) ans[cnt] += pi*c[i].r*c[i].r;
       else{
          sort(all(evt));
          evt.pb(evt.front());
          rep(j,0,sz(evt)-1) {
             cnt+=evt[j].delta;
             ans[cnt] += evt[j].p / evt[j+1].p / 2;
             db ang = evt[j + 1].ang - evt[j].ang;
             if(ang < 0) ang += pi * 2;
             ans[cnt] += ang * c[i].r * c[i].r / 2 - sin(ang) * c[i].r * c[i].r / 2;
          }}}}}
namespace ConvecIntersection{ // ?
   const int N = 1005;
   struct Rec {
      P d[10];int dn;// d[dn] = d[0]
      P operator [] (const int&n) {return d[n];}
   }r[N];
   typedef pair<db,int> pdi;
   int n;pdi res[1000005];
   db getLoc(P a,P b,P p){
      if(sgn(b.x - a.x)) return (p.x - a.x) / (b.x - a.x);
      return (p.y - a.y) / (b.y - a.y);
   }
   db work() {
      db rt=0;
      rep(i,0,n) rep(j,0,r[i].dn){
         int sz=0;
         res[sz++] = pdi(0,0);res[sz++] = pdi(1,0);
         rep(t,0,n) {
            if(t == i) continue;
            rep(g,0,r[t].dn) {
               int du = sgn((r[i][j+1] - r[i][j]) / (r[t][g] - r[i][j]));
               int dv = sgn((r[i][j+1] - r[i][j]) / (r[t][g+1] - r[i][j]));
               if(!du && !dv) {
                  if(sgn((r[i][j+1] - r[i][j]) * (r[t][g+1] - r[t][g])) < 0 || i < t){
                     res[sz++] = pdi(getLoc(r[i][j] , r[i][j+1] , r[t][g]) , 1);
                     res[sz++] = pdi(getLoc(r[i][j] , r[i][j+1] , r[t][g+1]) , -1);
                  }} else {
                     db s1 = (r[i][j] - r[t][g]) / (r[t][g+1] - r[t][g]);
                     db s2 = (r[t][g+1] - r[t][g]) / (r[i][j+1] - r[t][g]);
                     if(du >= 0 && dv < 0) res[sz++] = pdi(s1 / (s1 + s2) , 1);
                     else if(du < 0 && dv >= 0) res[sz++] = pdi(s1 / (s1 + s2) , -1);
                  }}}
         sort(res , res + sz);
         int cnt = 0; --sz;
         rep(t,0,sz) {
            cnt += res[t].se;
            if(cnt == 0 && sgn(res[t].fi - res[t+1].fi)) {
               db a = res[t].fi;
               if(a < 0) a = 0; if(a > 1) break;
               db b = res[t+1].fi;
               if(b < 0) continue; if(b > 1) b = 1;
               rt += ((r[i][j+1] - r[i][j]) * a + r[i][j]) / ((r[i][j+1]-r[i][j]) * b +
               r[i][j]);
            }}}
      return rt / 2;}}
```

5.2 MaxAreaPoly

```
ld solve_poly(vi &S) {
   assert(sz(S) > 0);
   int sum = 0, hi = S[0];
   vi vals;
   rep(i, 1, sz(S)) {
      int cur = S[i];
      if (cur > hi) swap(cur, hi);
      sum += cur;
      vals.pb(cur);
   }
   if (sum <= hi) return 0;
   auto getAngle = [&](ld D) -> ld{
      ld tot = 0;
      for (int l : vals) tot += 2 * asin(ld(l) / ld(D));
      return tot;
   };
   bool isReflex = (getAngle(hi) < PI);
   auto tooSmall = [&](ld D) {
      ld ang = getAngle(D);
      ld hiAng = 2 * asin(ld(hi) / ld(D));
      if (isReflex) return ang < hiAng;
      else return ang + hiAng >= 2 * PI;
   };
   ld mi = hi, ma = hi + 1;
   int numExpand = 0;
   while (tooSmall(ma)) numExpand++, ma += (ma - mi);
   rep(tim, 0, 50 + numExpand) {
      ld md = mi + (ma - mi) / 2;
      if (tooSmall(md)) mi = md;
      else ma = md;
   }
   ld D = mi, area = 0;
   for (int l : vals) area += ld(l) * sqrt(ld(D) * ld(D) - ld(l) * ld(l)) / 4;
   ld hiArea = ld(hi) * sqrt(ld(D) * ld(D) - ld(hi) * ld(hi)) / 4;
   if (isReflex) area -= hiArea;
```

```
   else area += hiArea;
   return area;
}
```

5.3 MaxAreaTri

```
// O(n ^ 2)
void maxAreaTri(P *p, int n, P &a, P &b, P &c) {
   int i = 0, j = 1, k = 2;
   a = p[i], b = p[j], c = p[k];
   T res = area(a, b, c), cur = res, tmp;
   do {
      while(1) {
         while(cur <= (tmp = area(p[i], p[j], p[(k + 1) % n]))) (++k) %= n, cur = tmp;
         if(cur <= (tmp = area(p[i], p[(j + 1) % n], p[k]))) (++j) %= n, cur = tmp;
         else break;
      }
      if(cur > res) a = p[i], b = p[j], c = p[k], res = cur;
      (++i) %= n;
      if(i == j) (++j) %= n;
      if(j == k) (++k) %= n;
      cur = area(p[i], p[j], p[k]);
   } while(i);
}
```

6 Graph

6.1 BCC

```
// key contains the id of edges
// _ starts from 0
namespace BCC{
    const int N = 202020;
    vi key , bcc[N];
    int dfn[N] , low[N] , id[N] , st[N] , _st , _;
    void dfs(int c,int dep,vector<pii> g[]){
        int cc=0;st[_st++]=c;
        dfn[c]=low[c]=dep;
        for(auto e:g[c]){
            int t=e.fi;
            if(!dfn[t]){
                dfs(t,dep+1,g);
                low[c]=min(low[c],low[t]);
                if(low[t]>dfn[c]) key.pb(e.se);
            } else if(dfn[t] != dfn[c] - 1 || cc++)
                low[c] = min(low[c] , dfn[t]);
        }
        if(low[c]==dfn[c]){
            do{id[st[--_st]]=_;}while(st[_st]!=c);
            _++;
        }
    }
    int solve(int n,vector<pii> g[]){
        fill_n(dfn,n,_=0);
        fill_n(low,n,_st=0);
        fill_n(bcc,n,key=vi());
        rep(i,0,n) if(!dfn[i]) dfs(i,1,g);
        rep(i,0,n) for(auto j:g[i]) if(id[i]!=id[j.fi])
            bcc[id[i]].pb(id[j.fi]);
        return _;
    }
};
```

6.2 DCC

```
// cactus: n multi by 2
// key is cuts
// dcc i->j , i(points) , j(bcc_block)
// st is stack
// _st is top of stack
// _ is number of dcc
// can handle isolate point and not connected graph and muti edge
// can handle self circle ?
namespace DCC{
    const int N = 202020;
    vi key , dcc[N];
    int dfn[N] , low[N] , st[N] , _st , _;
    void dfs(int c,int dep,const vi g[]){
        int cc=0,out=1<dep;st[_st++]=c;
        dfn[c]=low[c]=dep;
        for(auto t:g[c])
            if(!dfn[t]){
                dfs(t,dep+1,g);
                low[c]=min(low[c],low[t]);
                if(low[t]>=dfn[c]){
                    if(++out==2) key.pb(c);
                    while(st[--_st]!=t) dcc[st[_st]].pb(_);
                    dcc[c].pb(_);dcc[t].pb(_++);
                }
            } else if(dfn[t] != dfn[c] - 1 || cc++)
                low[c] = min(low[c] , dfn[t]);
    }
    int solve(int n,const vi g[]){// n is size of points
        fill_n(dfn,n,_=0);
        fill_n(low,n,_st=0);
        fill_n(dcc,n,key=vi());
        rep(i,0,n) if(!dfn[i]) dfs(i,1,g);
        rep(i,0,n) if(sz(dcc[i]) == 0) dcc[i].pb(_++);
        return _;
    }
}
```

6.3 DMST

```
// id starts from 0
// can handle multi edge, self ring
struct edge {int u, v, d, U, V;bitset<1005> b;};
```

```
struct DMST{
   static const int N = ::N , M = N * N , inf = 2e9;
   edge e[M];int n, m, vis[N], pre[N], id[N], index[N], Pre[N];
   bitset<1005> fang;
   int in[N];
   void ini(int n) {this->n = n, m = 0;}
   void addedge(int u, int v, int d) {e[m] = edge({u,v,d,u,v}); e[m].reset();e[m].b[m] =
    1;m++;}
   int run(int root){
      int ans = 0;
      while(1){
         rep(i, 0, n) in[i] = inf;
         rep(i, 0, m){
            int u = e[i].u ,v = e[i].v;
            if(e[i].d < in[v] && u != v) in[v] = e[i].d, pre[v] = u, index[v] = i;
         }
         rep(i, 0, n) {
            if(i == root) continue;
            if(in[i] == inf) return -1;
            fang ^= e[index[i]].b;
         }
         int cnt = 0;in[root] = 0;
         memset(id, -1, sizeof(*id)*n);
         memset(vis, -1, sizeof(*vis)*n);
         rep(i, 0, n){
            ans += in[i]; int v = i;
            int t = index[i];
            while(vis[v] != i && id[v] == -1 && v!=root) vis[v] = i, v = pre[v];
            if(v != root && id[v] == -1) {
               for(int u=pre[v];u != v;u = pre[u]) id[u] = cnt;
               id[v] = cnt++;
            }
         }
         if(cnt == 0) break;
         rep(i, 0, n) if(id[i] == -1) id[i] = cnt++;
         rep(i, 0, m) {
            int v=e[i].v;
            e[i].u = id[e[i].u]; e[i].v = id[e[i].v];
            if(e[i].u != e[i].v) {e[i].d -= in[v];e[i].b ^= e[index[v]].b;}
         }
         n = cnt; root = id[root];
      }
      return ans;
   }
} dmst;
```

6.4 Dijkstra

```
int n, dis[N];
void Dijkstra(int st) {
   priority_queue<pii> q;
   fill_n(dis + 1, n, inf);
   dis[st] = 0;
   q.push(mp(0, st));
   while(!q.empty()) {
      pii u = q.top();q.pop();
      if(dis[u.se] != -u.fi) continue;
      for(auto v : g[u.se]) {
         if(dis[v.fi] > dis[u.se] + v.se) {
            dis[v.fi] = dis[u.se] + v.se;
            q.push(mp(-dis[v.fi], v.fi));
         }
      }
   }
}
```

6.5 Dinic

```
// [0,n) init!!
template<class T>
struct Dinic{
    const static int N = 10101 , M = N * 10;
    int s , t , n , h[N] , cur[N] , level[N] , q[N] , e , ne[M] , to[M];
    T cap[M] , flow;
    void liu(int u,int v,T w){ to[e] = v;ne[e] = h[u];cap[e] = w;h[u] = e++;}
    void link(int u,int v,T w){ liu(u , v , w);liu(v , u , 0);}
    void ini(int _n = N) { fill(h , h + (n=_n) , -1);e = 0;}
    bool bfs(){
        int L = 0 , R = 0;
        fill(level , level + n , -1);
        level[q[R++] = s] = 0;
        while(L < R && level[t] == -1){
            int c = q[L++];
            for(int k=h[c];~k;k=ne[k])
                if(cap[k] > 0 && level[to[k]] == -1)
                    level[q[R++] = to[k]] = level[c] + 1;
        }
        return ~level[t];
    }
    T dfs(int c,T mx){
        if(c == t) return mx;
        T ret = 0;
        for(int &k = cur[c];~k;k = ne[k]){
            if(level[to[k]] == level[c] + 1 && cap[k] > 0){
                T flow = dfs(to[k] , min(mx , cap[k]));
                ret += flow;cap[k] -= flow , cap[k^1] += flow;mx -= flow;
                if(!mx) return ret;
            }
        }
        level[c] = -1;
        return ret;
    }
    T run(int _s,int _t){
        s = _s , t = _t;
        flow = 0;
        while(bfs()){
            copy(h , h + n , cur);
            flow += dfs(s,~0U>>1);
        }
        return flow;
```

```
    }
};
```

6.6 DualMST

对偶图最小生成树，等于平面图所有边边权和减去平面图最大生成树。

6.7 EulerianPath

```
vi ans; bool vis[N]; int p[N];
vector<pii> g[N];

void dfs(int u) {
   for( ; p[u] < sz(g[u]); ++p[u]) {
      auto v = g[u][p[u]];
      if(!vis[abs(v.se)]) {
         vis[abs(v.se)] = 1;
         dfs(v.fi);
         ans.pb(-v.se);
      }
   }
}
```

6.8 FindCircle

```
// 支持基环树森林和自环重边
const int N = 1e5 + 7;
vector<pair<pii>, int> g[N]; //点编号 边权边编号
int tim, dfn[N], fa[N], d[N], k;
vi cir[N];
int ne[N]; // 有向图的出度
int id[N]; //点属于的环编号

void dfs(int u, int pre) { // 为边编号pre
   dfn[u] = ++tim;
   rep(i, 0, sz(g[u])) {
      if (g[u][i].se == pre) continue;
      int v = g[u][i].fi.fi, w = g[u][i].fi.se;
      if (dfn[v] && dfn[v] <= dfn[u]) {
         k++;
         int p = u; cir[k].pb(p); id[p] = k;
         if (p != v) {do { p = fa[p]; cir[k].pb(p); id[p] = k;
         } while (p != v);}
         if (sz(cir[k]) > 1 && ne[cir[k][0]] != cir[k][1]) reverse(all(cir[k]));
         continue;
      }
      if (!dfn[v]) {fa[v] = u; d[v] = d[u] + w; dfs(v, g[u][i].se);}
   }
}
```

6.9 Lindstrom_Gessel_Viennot_Lemma

```
/*
 * 对于一张无边权的 DAG 图，给定 n 个起点和对应的 n 个终点，这 n 条不相交路径的方案数为矩阵
 * e(a1,b1),e(a1,b2)...e(a1,bn)
 * e(a2,b1),e(a2,b2)...e(a2,bn)
 * ....
 * ....
 * e(an,b1),e(an,b2)...e(an,bn)
 * 的行列式。

 * 即 M[i][j]=e(ai,bj)
 * e(a,b) 为 a 到 b 的路径方案数
 */
```

6.10 MMST

```
// 曼哈顿最小距离生成树
// 这份代码处理的区域是 Y 轴右转 45 度
namespace MMST {
#define lb(x) ((x) & -(x))
   const int N = 101010, inf = 1e9 + 7;

   vector<pair<int, pii> > E;
   vi V;

   pii mi[N];
   void init() { rep(i, 1, sz(V) + 1) mi[i] = mp(inf, inf); }
   void upd(int p, pii c) {
      p = sz(V) + 1 - p;
      for( ; p <= sz(V); p += lb(p)) mi[p] = min(mi[p], c);
   }
   pii qry(int p) {
      p = sz(V) + 1 - p;
      pii ans = mp(inf, inf);
      for( ; p >= 1; p ^= lb(p)) ans = min(ans, mi[p]);
      return ans;
   }
   int F(int x) { return lower_bound(all(V), x) - V.begin() + 1; }
   void _solve(vector<pair<pii, int> > v) {
      V.clear();
      rep(i, 0, sz(v)) v[i].fi.se -= v[i].fi.fi, V.pb(v[i].fi.se);
      sort(all(V));
      V.erase(unique(all(V)), V.end());
      sort(all(v));
      reverse(all(v));
      init();
      for(auto u : v) {
         pii t = qry(F(u.fi.se));
         int s = u.fi.fi * 2 + u.fi.se;
         if(t.se != inf) E.pb(mp(t.fi - s, mp(t.se, u.se)));
         upd(F(u.fi.se), mp(s, u.se));
      }
   }
   void solve(vector<pair<pii, int> > v) {
      _solve(v);
```

```
      rep(i, 0, sz(v)) swap(v[i].fi.fi, v[i].fi.se);
      _solve(v);
      rep(i, 0, sz(v)) v[i].fi.fi *= -1;
      _solve(v);
      rep(i, 0, sz(v)) swap(v[i].fi.fi, v[i].fi.se);
      _solve(v);
   }
}
```

6.11 ManhattanDistance

```
(x, y) -> (x + y, x - y)           Manhattan distance -> Chebyshev distance
(x, y) -> (x + y >> 1, x - y >> 1) Chebyshev distance -> Manhattan distance
```

6.12 MaxMatch

```
namespace MaxMatch {
   const int N = 1050;
   int link[N], vis[N], use[N], in[N];
   queue<int> Q;
   int dfs(int u, vi g[]) {
      for(auto v : g[u]) {
         if(!vis[v]) {
            vis[v] = 1;
            if(!link[v] || dfs(link[v], g)) { return link[v] = u, 1; }
         }
      }
      return 0;
   }
   int solve(int n, int m, vi g[]) {
      fill_n(link, m+1, 0);
      int ret = 0;
      rep(i, 1, n+1) {
         fill_n(vis, m+1, 0);
         ret += dfs(i, g);
      }
      return ret;
   }
   void MVC(int n, vi g[]) {
      fill_n(vis, n+1, 0);
      per(i, 1, n+1) link[link[i]] = i;
      rep(i, 1, n+1) if (!link[i]) vis[i] = use[i] = 1, Q.push(i);
      while (!Q.empty()) {
         int u = Q.front(); Q.pop();
         if (use[u] == 1) {
            for (auto v : g[u]) {
               use[v] = 2;
               if (!vis[v]) vis[v] = 1, Q.push(v);
            }
         }else {
            int v = link[u];
            use[v] = 1;
            if (!vis[v]) vis[v] = 1, Q.push(v);
         }
      }
      rep(i, 1, n+1) if (link[i] && !use[link[i]]) use[i] = 2;
      return;
   }
}
```

6.13 Max_clique_BK

```
//g[i][i] should be 0
//g[i] is i's edge
//index [0..N)
//O(n ^ 3)
typedef unsigned long long T;
struct BK {
   static const int N = 100; T g[N];
   inline int ctz(T s){ return s ? __builtin_ctzll(s) : 64;}
   int n, ans;
   void ini(int _n) {
      //per(i, 0, n = _n) g[i] = (1ull << n) - 1 - (1ull << i); }
      n = _n;rep(i, 0, n) g[i] = 0;
      rep(i, 0, n) rep(j, 0, n) if (a[i][j]) g[i] |= 1ull << j;
   }
   void gao(T cur, T can, T ban) {
      if (!can && !ban) { ans = max(ans, __builtin_popcountll(cur)); return; }
      if (!can) return;
      int piv = ctz(can | ban), ret = 0;
      T z = can & ~g[piv];
      for(int u = ctz(z); u < n; u += ctz(z >> (u + 1)) + 1) {
         gao(cur | (1ull << u), can & g[u], ban & g[u]);
         can ^= 1ull << u, ban |= 1ull << u;
      }
   }
   int run() { gao(ans = 0, (1ull << n) - 1, 0); return ans; }
}  bk;
```

6.14 Max_clique_fastest

```
const int N = 130;
typedef bool BB[N];
struct Maxclique {
    const BB *e; int pk, lv; db Tlimit;
    struct ve {int i, d; ve(int i): i(i),d(0) {}}; //ve : Vertex
    struct sc {int a, b; sc(     ): a(0),b(0) {}}; //sc : StepCount
    typedef vector<ve> ves; ves V;                 //ves: Vertices
    typedef vector<int> cc; cc  Q, QMAX;           //cc : ColorClass
    vector<cc> C;
    vector<sc> S;
    Maxclique(BB *conn, int sz, const db tt = 0.025): pk(0), lv(1), Tlimit(tt) {
        rep(i, 0, sz) V.pb(ve(i)); e = conn;
        C.resize(sz + 1);
        S.resize(sz + 1);
    }
    static bool desc_deg(const ve &a, const ve &b) { return a.d > b.d; }
    void ini_col(ves &v) { per(i, 0, sz(v)) v[i].d = min(i, v[0].d) + 1; }
```

```
    void set_deg(ves &v) { rep(i, 0, sz(v)){v[i].d = 0; rep(j, 0, sz(v)) v[i].d += e[v[i
    ].i][v[j].i]; } }
    void deg_sort(ves &R) { set_deg(R); sort(all(R), desc_deg); }
    bool cut1(int pi , cc  &va) { rep(i, 0, sz(va)) if (e[pi][va[i]]) return true;
    return false; }
    void cut2(ves &va, ves &vb) { rep(i, 0, sz(va) - 1) if (e[va.back().i][va[i].i]) vb.
    pb(va[i].i); }
    void co_sort(ves &R) {
        int j = 0, maxno = 1, min_k = max(sz(QMAX) - sz(Q) + 1, 1);
        rep(i, 1, 3) C[i].clear();
        rep(i, 0, sz(R)) {
            int pi = R[i].i, k = 1;
            while (cut1(pi, C[k])) k++;
            if (k > maxno) C[(maxno = k) + 1].clear(); C[k].pb(pi);
            if (k < min_k) R[j++].i = pi;
        }
        if (j > 0) R[j - 1].d = 0;
      rep(k, min_k, maxno + 1)
            rep(i, 0, sz(C[k]))
                R[j].i = C[k][i], R[j++].d = k;
    }
    void exp_dyn(ves &R) { // expand_dyn
        S[lv].a += S[lv - 1].a - S[lv].b;
        S[lv].b  = S[lv - 1].a;
        for (; sz(R); Q.pop_back(), R.pop_back()) {
            if (sz(Q) + R.back().d <= sz(QMAX)) return;
            Q.pb(R.back().i);
            ves Rp; cut2(R, Rp);
            if (sz(Rp)) {
                if ((db) S[lv].a / ++pk < Tlimit) deg_sort(Rp);
                co_sort(Rp); S[lv++].a++;
                exp_dyn(Rp); --lv;
            } else if (sz(Q) > sz(QMAX)) QMAX = Q;
        }
    }
    void mcqdyn(int *mxc, int &sz) { // mcqdyn(int maxclique, int &siz)
        set_deg(V); sort(all(V), desc_deg);
        ini_col(V); rep(i, 0, sz(V) + 1) S[i].a = S[i].b = 0;
        exp_dyn(V); per(i, 0, sz(QMAX)) mxc[i] = QMAX[i];
        sz = sz(QMAX);
    }
};
```

6.15 MinCostMaxFlow

```
// [0,n) , init!! , inf modify
template<class U,class V>
struct MCMF{
    static const int N = 204 , M = 101010;
    int h[N] , ing[N] , pre[N] , to[M] , ne[M] , e , s , t , n;
    U cap[M];V dis[N] , cost[M];
    void ini(int _n = N){ fill(h , h + (n=_n) , -1);e = 0;}
    void liu(int u,int v,U c,V w){ to[e] = v;ne[e] = h[u];cap[e] = c;cost[e] = w;h[u] =
    e++;}
    void link(int u,int v,U c,V w){ liu(u,v,c,w);liu(v,u,0,-w); }
    bool spfa(){
        queue<int> Q;
        fill(dis,dis+n,inf);
        ing[s] = true , dis[s] = 0;
        Q.push(s);
        while(!Q.empty()){
            int c = Q.front();Q.pop();ing[c] = false;
            for(int k=h[c];~k;k=ne[k]){
                int v = to[k];
                if(cap[k] <= 0) continue;
                if(dis[c] + cost[k] < dis[v]){
                    dis[v] = dis[c] + cost[k];
                    pre[v] = k;
                    if(!ing[v]) Q.push(v) , ing[v] = true;
                }
            }
        }
        return dis[t] != inf;
    }
    U flow;V mincost;
    pair<U,V> run(int _s,int _t){
        s = _s , t = _t;
        flow = mincost = 0;
        while(spfa()){
            U pl = inf;int p , k;
            for(p=t;p!=s;p=to[k^1]) pl = min(pl , cap[k=pre[p]]);
            for(p=t;p!=s;p=to[k^1]){
                k = pre[p];
                cap[k] -= pl;
                cap[k^1] += pl;
            }
            mincost += pl * dis[t];
            flow += pl;
        }
        return make_pair(flow , mincost);
    }
};
```

6.16 SCC

```
// _ starts from 0
namespace SCC{
    const int N = 100050;
    int dfn[N],low[N],id[N],st[N],_st,_,cc;
    void dfs(int c,vi g[]){
        dfn[c]=low[c]=++cc;
        st[_st++]=c;
        for(auto t:g[c])
            if(!dfn[t])
                dfs(t,g),low[c]=min(low[c],low[t]);
            else if(!id[t])
                low[c] =min(low[c],dfn[t]);
        if(low[c]==dfn[c]){
            ++_;
            do{id[st[--_st]]=_;}while(st[_st]!=c);
```

```
        }
    }
    vi ng[N];
    int solve(int n,vi g[]){
        fill_n(dfn,n,cc=0);
        fill_n(low,n,_st=0);
        fill_n(id,n,_=0);
        rep(i,0,n) if(!dfn[i]) dfs(i,g);
        rep(i,0,n) --id[i];
        fill_n(ng,_,vi());
        rep(i,0,n) for(auto j:g[i]) if(id[i]!=id[j]) ng[id[i]].pb(id[j]);
        return _;
    }
}
```

6.17 SteinerTree

```
// 要视图的情况使用spfa, dijstra, 多源bfs
const int N = 11, M = 10;
const int inf = 0x3f3f3f3f;
int n, m, k, a[N][N], st[N][N], dp[1 << M][N][N], S, ans;
bool use[N][N], vis[1 << M][N][N];
int dx[] = {1, -1, 0, 0};
int dy[] = {0, 0, 1, -1};
queue<pii> q;

struct node {
   int x, y, msk;
   node(int x = 0, int y = 0, int msk = 0):x(x), y(y), msk(msk){}
} now;

pair<node, node> pre[1 << M][N][N];

void spfa(int msk) {
   while (!q.empty()) {
      pii u = q.front(); q.pop();
      int x = u.fi, y = u.se;
      vis[msk][x][y] = 0;
      rep(i, 0, 4) {
         int nx = x + dx[i], ny = y + dy[i], t = msk | st[nx][ny];
         if (nx > n || nx < 1 || ny > m || ny < 1) continue;
         int &z = dp[t][nx][ny], w = dp[msk][x][y] + a[nx][ny];
         if (z > w) {
            z = w, pre[t][nx][ny] = mp(node(x, y, msk), node(x, y, 0));
            if (t == msk && !vis[msk][nx][ny]) {
               vis[msk][nx][ny] = 1;
               q.push(mp(nx, ny));
            }
         }
      }
   }
}

void dfs(node now) {
   pair<node, node> t = pre[now.msk][now.x][now.y];
   node t1 = t.fi, t2 = t.se;
   use[now.x][now.y] = 1;
   if (!t1.x) return;
   dfs(t1);
   if (t2.msk) dfs(t2);
}

int SteinerTree(int n, int m) {
   memset(dp, 0x3f, sizeof(dp));
   rep(i, 1, n+1) rep(j, 1, m+1) {
      cin >> a[i][j];
      if (!a[i][j]) {
         st[i][j] = pw(k);
         dp[pw(k)][i][j] = 0;
         pre[pw(k++)][i][j] = mp(node(0, 0, 0), node(0, 0, 0));
      }
   }
   S = pw(k) - 1;
   rep(msk, 1, S+1) {
      rep(i, 1, n+1)
         rep(j, 1, m+1) {
            if (st[i][j] && !(st[i][j] & msk)) continue;
            int &z = dp[msk][i][j];
            for (int t = msk & (msk - 1); t > 0; t = (t - 1) & msk) {
               int t1 = t | st[i][j], t2 = msk ^ t | st[i][j];
               int w = dp[t1][i][j] + dp[t2][i][j] - a[i][j];
               if (z > w) z = w, pre[msk][i][j] = mp(node(i, j, t1), node(i, j, t2));
            }
            if (z < inf) q.push(mp(i, j)), vis[msk][i][j] = 1;
         }
      spfa(msk);
   }
   ans = inf;
   rep(i, 1, n+1) rep(j, 1, m+1) if (ans > dp[S][i][j])
      ans = dp[S][i][j], now = node(i, j, S);
   dfs(now);
   return ans == inf ? -1 : ans;
}
```

6.18 StoerWagner_O(n3)

```
struct StoerWagner{
   static const int N = 305;
   static const int INF = 0x3f3f3f3f;;
   int n;
   int g[N][N], val[N];
   bool vis[N], use[N];
   void init(int _n) {
      n = _n;
      fill_n(use + 1, n, 0);
      rep(i, 1, n+1) fill_n(g[i] + 1, n, 0);
   }
   void add_edge(int u, int v, int w) {
      g[u][v] += w;
      g[v][u] += w;
```

```
   }
   void merge(int u, int v) {
      rep(i, 1, n+1) {
         g[v][i] += g[u][i];
         g[i][v] += g[i][u];
      }
      use[u] = 1;
   }
   int MinimumCutPhase(int cnt, int &s, int &t) {
      fill_n(val + 1, n, 0);
      fill_n(vis + 1, n, 0);
      t = 1;
      while (--cnt) {
         vis[s = t] = 1;
         rep(i, 1, n+1) if (!vis[i] && !use[i]) val[i] += g[t][i];
         int ma = 0;
         rep(i, 1, n+1) if (!vis[i] && !use[i] && val[i] >= ma) ma = val[i], t = i;
         if (!ma) return 0;
      }
      return val[t];
   }
   int solve() {
      int res = INF;
      for (int i = n, s, t; i > 1; --i) {
         res = min(res, MinimumCutPhase(i, s, t));
         if (res == 0) break;
         merge(s, t);
      }
      return res;
   }
} SW;
```

6.19 StoerWagner_O(nmlog(m))

```
struct StoerWagner{
   static const int N = 3005, M = 100005 * 2, INF = 0x3f3f3f3f;;
   int head[N], val[N], e, n;
   int to[M], ne[M], data[M];
   bool vis[N];
   int fa[N], link[N];
   void init(int _n) {
      n = _n;
      fill_n(head + 1, n, -1);
      fill_n(link + 1, n, -1);
      rep(i, 1, n+1) fa[i] = i;
      e = 0;
   }
   void add_edge(int u, int v, int w) {
      to[e] = v; data[e] = w; ne[e] = head[u]; head[u] = e++;
      to[e] = u; data[e] = w; ne[e] = head[v]; head[v] = e++;
   }
   int findset(int u) {
      return u == fa[u] ? u : fa[u] = findset(fa[u]);
   }
   void merge(int u, int v) {
      int p = u;
      while (~link[p]) p = link[p];
      link[p] = v;
      fa[v] = u;
   }
   int MinimumCutPhase(int cnt, int &s, int &t) {
      fill_n(val + 1, n, 0);
      fill_n(vis + 1, n, 0);
      priority_queue<pii> q;
      t = 1;
      while (--cnt) {
         vis[s = t] = 1;
         for (int u = s; ~u; u = link[u]) {
            for (int p = head[u]; ~p; p = ne[p]) {
               int v = findset(to[p]);
               if (!vis[v]) q.push(mp(val[v] += data[p], v));
            }
         }
         while (!q.empty() && (vis[q.top().se] || val[q.top().se] != q.top().fi)) {
            q.pop();
         }
         if (q.empty()) return 0;
         t = q.top().se; q.pop();
      }
      return val[t];
   }
   int solve() {
      int res = INF;
      for (int i = n, s, t; i > 1; --i) {
         res = min(res, MinimumCutPhase(i, s, t));
         if (res == 0) break;
         merge(s, t);
      }
      return res;
   }
} SW;
```

6.20 生成树计数与欧拉回路方案数

```
// d[][]:
//     i!=j d[i][j]=0
//     i==j d[i][j]=in_deg(i)
// b[][]:
//     from i to j has b[i][j] directed edges
// a[][] = d[][] - b[][]

// 无向图生成树个数： a[][] 任何一个 n-1 阶主子式的绝对值
// 有向图以 i 为根的生成树个数： a[][] 去掉第 i 行第 i 列的行列式的绝对值

int det(int n) { // det(a[1..n-1][1..n-1])
   int ans=1;
   rep(i, 1, n) {
      rep(j, i+1, n) while(a[j][i]) {
         int t = a[i][i] / a[j][i];
         rep(k, i, n) a[i][k] = sub(a[i][k], mul(a[j][k], t)), swap(a[i][k], a[j][k]);
```

```
         ans = P - ans;
      }
      if(a[i][i] == 0) return 0;
      ans = mul(ans, a[i][i]);
   }
   return ans;
}

// 有向图要记得判断每个点的出度入度是否相等
// 无向图需要转换成有向图
// tw(G): 以 w 为根的生成树个数
// ec(G) = tw(G) * pi((deg[v] - 1)!)
// ans = ec(G) * deg[w]; 如果求的不是本质不同的，就还需要这个
// 本质相同：    1231341 1341231
// 本质不同：    1231341 1312341
```

7 Math

7.1 BerlekampMassey

```
// O(len^2)
vi BM(vi s) {
   vi C(1, 1), B(1, 1);
   int L = 0, m = 1, b = 1;
   rep(n, 0, sz(s)) {
      ll d = 0;
      rep(i, 0, L+1) (d += 1ll * C[i] * s[n-i]) %= P;
      if(d == 0) ++m;
      else {
         vi T = C;
         ll c = P - d * kpow(b, P - 2) % P;
         while(sz(C) < sz(B) + m) C.pb(0);
         rep(i, 0, sz(B)) C[i + m] = add(C[i + m], mul(c, B[i]));
         if(2 * L <= n) L = n + 1 - L, B = T, b = d, m = 1;
         else ++m;
      }
   }
   reverse(all(C));
   rep(i, 0, sz(C)) C[i] = P - C[i];
   return vi(C.begin(), C.end() - 1);
}
```

7.2 Bernoulli

```
// desc : 0^k + 1^k + 2^k + .. + (n-1)^k
// time-ini : O(n^2)
// time-cal : k + log
namespace Bernoulli {
   const int N = 1000;
   int C[N][N], B[N];
   void ini() {
      rep(i, 0, N) C[i][0] = 1;
      rep(i, 0, N) rep(j, 1, i + 1) C[i][j] = add(C[i - 1][j - 1], C[i - 1][j]);
      B[0] = 1;
      rep(i, 1, N) {
         B[i] = 0;
         rep(j, 0, i) B[i] = add(B[i], MOD - mul(C[i + 1][j], B[j]));
         B[i] = mul(B[i], qpow(C[i + 1][i], MOD - 2)) % MOD;
      }
   }
   int cal(int n, int k) {
      int sum = 0;
      rep(i, 0, k + 1) sum = add(sum, mul(C[k + 1][i], mul(B[i], qpow(n, k + 1 - i))));
      return mul(sum, qpow(k + 1, MOD - 2));
   }
};
```

7.3 CRT

```
const int N = 1e5+7;
ll a[N], mod[N];

struct CRT{
   int M, R;
   void exgcd(ll a, ll b, ll &x, ll &y){
      if(b == 0) { x = 1; y = 0; return;}
      exgcd(b, a % b, y, x);
      y -= a / b * x;
   }
   ll Inv(ll a, ll mod){
      ll x = 0, y = 0;
      exgcd(a, mod, x, y);
      x %= mod;
      return x < 0 ? x + mod : x;
   }
   ll CRT(int n, ll *a, ll *mod){
      M = mod[1], R = a[1];
      rep(i, 2, n+1) {
         ll g = __gcd(M, mod[i]);
         ll inv = Inv(M / g, mod[i] / g);
         if ((a[i] - R) % g) return -1; // 无解
         R += inv * ((a[i] - R) / g) % (mod[i] / g) * M;
         M = M / g * mod[i];
         R = (R % M + M) % M; // 可能为 0 看是否需要是正整数
      }
      return R;
   }
} crt;
```

7.4 EulerPower

```
// a[l] ^ a[l+1] ^ a[l+2] ... ^ a[r] % mod 注意结果要再模mod
map<int, int> M;

int phi(int n) {
   if (M.count(n)) return M[n];
   int r = n, nn = n;
```

```
   for(int i = 2; i * i <= n; i++) if (n % i == 0){
      r = r / i * (i-1);
      while (n % i == 0) n /= i;
   }
   if (n > 1) r = r / n * (n-1);
   M[nn] = r;
   return r;
}

ll Euler_qpow(ll a, ll b, ll mod) {
   ll res = 1; bool ok = (b > 0 && a >= mod);
   while (b) {
      if (b & 1) {
         res = res * a;
         ok |= (res >= mod);
         res %= mod;
      }
      a = a * a;
      ok |= (b > 1 && a >= mod);
      a %= mod;
      b >>= 1;
   }
   return res + mod * ok;
}

ll work(int l, int r, int mod) {
   if (mod == 1) return 1;
   if (l == r) return a[l];
   return Euler_qpow(a[l], work(l+1, r, phi(mod)), mod);
}
```

7.5 FFT

```
const int M = 1 << 17 << 1;
const db pi = acos(-1);

struct vir{
   db r, i;
   vir(db r = 0.0, db i = 0.0) : r(r), i(i){}
   void print() {printf("%f %f\n", r, i);}
   vir operator +(const vir &c) {return vir(r + c.r, i + c.i);}
   vir operator -(const vir &c) {return vir(r - c.r, i - c.i);}
   vir operator *(const vir &c) {return vir(r * c.r - i * c.i, r * c.i + i * c.r);}
} a[M], b[M], w[2][M];

struct FFT{
   int N, na, nb, rev[M];
   void fft(vir *a, int f){
      vir x, y;
      rep(i, 0, N) if (i < rev[i]) swap(a[i], a[rev[i]]);
      for (int i = 1; i < N; i <<= 1)
         for (int j = 0, t = N/(i<<1); j < N; j += i<<1)
            for (int k = 0, l = 0; k < i; k++, l += t)
               x = w[f][l] * a[j+k+i], y = a[j+k], a[j+k] = y+x, a[j+k+i] = y-x;
      if (f) rep(i, 0, N) a[i].r /= N;
   }
   void work(){
      int d = __builtin_ctz(N);
      rep(i, 0, N) {
         rev[i] = (rev[i>>1] >> 1) | ((i&1) << (d-1));
         w[1][i] = w[0][i] = vir(cos(2*pi*i/N), sin(2*pi*i/N));
         w[1][i].i = -w[1][i].i;
      }
   }
   void doit(vir *a, vir *b, int na, int nb){ // [0, na)
      for (N = 1; N < na + nb - 1; N <<= 1);
      rep(i, na, N) a[i] = vir(0, 0);
      rep(i, nb, N) b[i] = vir(0, 0);
      work(), fft(a, 0), fft(b, 0);
      rep(i, 0, N) a[i] = a[i] * b[i];
      fft(a, 1);
      //rep(i, 0, N) a[i].print();
   }
} fft;
```

7.6 FFTMOD

```
const int P =1e9+7;
const int MAXN= 1 << 18 << 1;
const db PI = acos(-1);

int N,L,MASK,na,nb;
int a[MAXN], b[MAXN];

struct vir{
   db r, i;
   vir(db r = 0.0, db i = 0.0) : r(r), i(i){}
   void print() {printf("%lf %lf\n", r, i);}
};

vir operator +(const vir &a, const vir &b) {return vir(a.r + b.r, a.i + b.i);}
vir operator -(const vir &a, const vir &b) {return vir(a.r - b.r, a.i - b.i);}
vir operator *(const vir &a, const vir &b) {return vir(a.r * b.r - a.i * b.i, a.r * b.i
+ a.i * b.r);}
vir conj(vir a) {return vir(a.r, -a.i);}

vir w[MAXN];

void FFT(vir p[], int n) {
   for (int i = 1, j = 0; i < n - 1; ++i) {
      for (int s = n; j ^= s >>= 1, ~j & s;);
      if (i < j) swap(p[i], p[j]);
   }
   for (int d = 0; (1 << d) < n; ++d) {
      int m = 1 << d, m2 = m * 2, rm = n >> (d + 1);
      for (int i = 0; i < n; i += m2) {
         for (int j = 0; j < m; ++j) {
            vir &p1 = p[i + j + m], &p2 = p[i + j];
            vir t = w[rm * j] * p1;
            p1 = p2 - t, p2 = p2 + t;
```

```
        }
      }
   }
}
vir A[MAXN], B[MAXN], C[MAXN], D[MAXN];


void doit(int *a, int *b, int na, int nb){
   for (N = 1; N < na + nb - 1; N <<= 1);
   L = 15;
   MASK = (1<<L) - 1;
   rep(i, 0, N) w[i] = vir(cos(2 * i * PI / N), sin(2 * i * PI / N));
   rep(i, 0, N) {
      A[i] = vir(a[i] >> L, a[i] & MASK);
      B[i] = vir(b[i] >> L, b[i] & MASK);
   }
}

void mul() {
   FFT(A, N), FFT(B, N);
   rep(i, 0, N) {
      int j = (N - i) % N;
      vir da = (A[i] - conj(A[j])) * vir(0, -0.5),
         db = (A[i] + conj(A[j])) * vir(0.5, 0),
         dc = (B[i] - conj(B[j])) * vir(0, -0.5),
         dd = (B[i] + conj(B[j])) * vir(0.5, 0);
      C[j] = da * dd + da * dc * vir(0, 1);
      D[j] = db * dd + db * dc * vir(0, 1);
   }
   FFT(C, N), FFT(D, N);
   for (int i = 0; i < N; ++i) {
      ll  da = (ll)(C[i].i / N + 0.5) % P,
         db = (ll)(C[i].r / N + 0.5) % P,
         dc = (ll)(D[i].i / N + 0.5) % P,
         dd = (ll)(D[i].r / N + 0.5) % P;
      a[i] = ((dd << (L * 2)) + ((db + dc) << L) + da) % P;
   }
}
```

7.7 FFT_fast

```
const int N = 1 << 21;
const double pi=acos(-1.0);
struct vir{
    double a,b;
    vir(double r=0.0,double i=0.0) {a=r,b=i;}
    vir operator +(const vir &o) const{return vir(a+o.a,b+o.b);}
    vir operator -(const vir &o) const{return vir(a-o.a,b-o.b);}
    vir operator *(const vir &o) const{return vir(a*o.a-b*o.b,b*o.a+a*o.b);}
    vir operator *(const double &o) const{return vir(a*o,b*o);}
    vir operator !() const{return vir(a,-b);}
} x[N|1], y[N|1], z[N|1], w[N|1];

int K;

void fft(vir x[],int k,int v){
    for(int i=0,j=0; i<k; i++){
        if(i>j)swap(x[i],x[j]);
        for(int l=k>>1; (j^=l)<l; l>>=1);
    }
    w[0] = vir(1, 0);
    for(int i=2; i<=k; i<<=1){
        vir g = vir(cos(2*pi/i), (v ? -1 : 1) * sin(2*pi/i));
        for(int j=(i>>1); j>=0; j-=2) w[j] = w[j>>1];
        for(int j=1; j<i>>1; j+=2) w[j] = w[j-1] * g;
        for(int j=0; j<k; j+=i){
            vir *a = x+j, *b = a+(i>>1);
            for(int l=0; l<i>>1; l++){
                vir o = b[l] * w[l];
                b[l] = a[l] - o;
                a[l] = a[l] + o;
            }
        }
    }
    if (v) for(int i=0; i<k; i++) x[i] = vir(x[i].a/k,x[i].b/k);
}

void doit(int *a, int *b, int na, int nb) {
   for(K = 1; K <= na+nb>>1; K <<= 1);
   rep(i, 0, K) x[i] = y[i] = vir(0, 0);
   for(int i=0; i<=na; i++) (i&1 ? x[i>>1].b : x[i>>1].a) = a[i];
    for(int i=0; i<=nb; i++) (i&1 ? y[i>>1].b : y[i>>1].a) = b[i];
   fft(x, K, 0);
    fft(y, K, 0);
    rep(i, 0, K){
        int j = K-1 & K-i;
        vir tmp = (i&K>>1) ? vir(1, 0) - w[i^K>>1] : w[i] + vir(1, 0);
        z[i] =(x[i]*y[i]*4 - (x[i] - !x[j])*(y[i] - !y[j])*tmp)*0.25;
    }
    fft(z, K, 1);
    rep(i, 0, na+nb+1) a[i] = i&1 ? z[i>>1].b + 0.1 : z[i>>1].a + 0.1;
}
```

7.8 Fib

```
// sum(fib[1..n]) + 1=fib[n + 2]
// gcd(fib[n], fib[m]) = fib[gcd(n, m)]
```

7.9 GaussDB

```
namespace GaussDB{
   static const int N = 505;
   db a[N][N], x[N]; //增广矩阵和解集
   bool ok[N]; // 标记变元是否确定
   int free[N], free_num; // 一组合法自由变元
   const db eps = 1e-14;
   int Gauss(int equ, int var){
      int k, col, p;
      fill_n(ok, var, 0); free_num = 0;
```

```
      fill_n(x, var, 0);
      for(k = col = 0; k < equ && col < var; ++k, ++col){
         p = k;
         rep(i, k+1, equ) if(fabs(a[i][col]) > fabs(a[p][col])) p = i;
         if (p != k) rep(j, k, var+1) swap(a[p][j], a[k][j]);
         if(fabs(a[k][col]) < eps) {k--; continue;}
         rep(i, k+1, equ){
            if(fabs(a[i][col]) < eps) continue;
            db t = a[i][col] / a[k][col];
            rep(j, col, var+1) a[i][j] -= a[k][j] * t;
         }
      }
      rep(i, k, equ) if (fabs(a[i][var]) > eps) return -1;//无解
      if(k < var){
         /*
         int pre = var;
         per(i, 0, k) {
            int num = 0;
            rep(j, 0, var) if(fabs(a[i][j]) > eps) {
               if (!num) p = j; num++;
            }
            rep(j, 0, i) if (fabs(a[j][p]) > eps) {
               db t = a[j][p] / a[i][p];
               rep(l, p, var+1) a[j][l] -= a[i][l] * t;
            }
            rep(j, p+1, pre) free[free_num++] = j; pre = p;
            if(num > 1) continue;
            ok[p] = 1;
            x[p] = a[i][var] / a[i][p];
         }*/
         return var - k;//自由变元个数
      }
      per(i, 0, var) {
         db t = a[i][var];
         rep(j, i+1, var) if (fabs(a[i][j]) > eps) t -= x[j] * a[i][j];
         x[i] = t / a[i][i];
      }
      return 0;
   }
}
```

7.10 GaussInt

```
namespace GaussInt{
   static const int N = ::N, P = 1e9 + 7;
   int a[N][N], x[N]; //增广矩阵和解集
   bool ok[N]; // 标记变元是否确定
   int free[N], free_num; // 一组合法自由变元
   int add(int a, int b) {if ((a += b) >= P) a -= P; return a < 0 ? a + P : a;}
   int mul(int a, int b) {return 1ll * a * b % P;}
   int kpow(int a, int b) {int r=1;for(;b;b>>=1,a=mul(a,a)) {if(b&1)r=mul(r,a);}return r
   ;}
   int Gauss(int equ, int var){
      int k, col, p;
      fill_n(free, var, 0); free_num = 0;
      fill_n(x, var, 0);
      for(k = col = 0; k < equ && col < var; ++k, ++col){
         p = k; rep(i, k+1, equ) if (a[i][col]) {p = i; break;}
         if (p != k) rep(j, k, var+1) swap(a[p][j], a[k][j]);
         if(!a[k][col]) {k--; continue;}
         int inv = kpow(a[k][col], P - 2);
         rep(i, col, var+1) a[k][i] = mul(a[k][i], inv);
         rep(i, k+1, equ){
            if(!a[i][col]) continue;
            int t = a[i][col];
            rep(j, col, var+1) a[i][j] = add(a[i][j], -mul(a[k][j], t));
         }
      }
      rep(i, k, equ) if (a[i][var]) return -1;//无解
      if(k < var){
         /*
         int pre = var;
         per(i, 0, k) {
            int num = 0;
            rep(j, 0, var) if (a[i][j]) {
               if (!num) p = j; num++;
            }
            rep(j, 0, i) if (a[j][p]) {
               int t = a[j][p];
               rep(l, p, var+1) a[j][l] = add(a[j][l], -mul(a[i][l], t));
            }
            rep(j, p+1, pre) free[free_num++] = j; pre = p;
            if(num > 1) continue;
            x[p] = a[i][var];
            ok[p] = 1;
         }*/
         return var - k;//自由变元个数
      }
      per(i, 0, var) {
         int t = a[i][var];
         rep(j, i+1, var) if (a[i][j]) t = add(t, -mul(a[i][j], x[j]));
         x[i] = t;
      }
      return 0;
   }
}
```

7.11 GaussXor

```
//对 2 取模的 01 方程组
namespace Gauss{
   static const int N = 2e3 + 10;
   //有 equ 个方程， var 个变元。增广矩阵行数为 equ 列数为, [0..var]
   bitset<N> a[N]; //增广矩阵 modif
   bool ok[N]; // 标记变元是否确定
   int x[N]; //解集
   int free[N], free_num; //一组合法自由变元（多解枚举自由变元可以使用）
   //返回值为 -1 表示无解，为 0 是唯一解，否则返回自由变元个数
   int Gauss(int equ, int var){
      int p, col, k; // k 为增广矩阵的秩
```

```
      fill_n(ok, var, 0); free_num = 0;
      fill_n(x, var, 0);
      for(k = 0, col = 0; k < equ && col < var; k++, col++){
         p = k; rep(i, k, equ) if (a[i][col]) {p = i; break;}
         if (p != k) swap(a[k], a[p]);
         if (!a[k][col]){
            k--; free[free_num++] = col;//这个是自由变元
            continue;
         }
         rep(i, 0, equ) if (i != k && a[i][col]) a[i] ^= a[k];
      }
      rep(i, col, var) free[free_num++] = i;
      rep(i, k, equ) if (a[i][var]) return -1;
      if(k < var) {
         /*
         per(i, 0, k) {
            int num = 0;
            rep(j, 0, var) if(a[i][j]) {
               if (!num) p = j; num++;
            }
            if(num > 1) continue;
            ok[p] = 1; x[p] = a[i][var];
         }*/
         return var - k;//自由变元个数
      }
      //唯一解，回代
      per(i, 0, var){
         bool t = a[i][var];
         rep(j, i+1, var) t ^= (a[i][j] && x[j]);
         x[i] = t;
      }
      return 0;
   }
}
```

7.12 LinearBasis

```
struct Base{
    ll a[63];
    Base() {memset(a,0,sizeof(a));}
    void ins(ll x){
        for(int i=62;~i;--i) {
            if(x>>i&1) {
                if(a[i]) x^=a[i];
                else{ a[i]=x; break; }
            }
        }
    }
};
```

7.13 LinearRecursion

```
// a_{m} = \sum_{j=0}^{-m1}a_{j}*c_{j} O(m^2lgn)
int linear_recurrence(ll n, int m, vi a, vi c) {
   if (n<m) return (a[n]+P)%P;
   vector<ll> v(m, 0), u(m<<1, 0);
   v[0] = 1;
   for(ll x = 0, W = n ? 1ll<<(63 - __builtin_clzll(n)) : 0; W; W >>= 1, x <<= 1) {
      fill(all(u), 0);
      int b = !!(n & W); if(b) x++;
      if(x < m) u[x] = 1;
      else {
         rep(i, 0, m) rep(j, 0, m) (u[i + b + j] += v[i] * v[j]) %= P;
         per(i, m, 2*m) rep(j, 0, m) (u[i - m + j] += c[j] * u[i]) %= P;
      }
      copy(u.begin(), u.begin() + m, v.begin());
   }
   ll ans = 0;
   rep(i, 0, m) (ans += v[i] * a[i]) %= P;
   return (ans+P)%P;
}
```

7.14 MathFunction

```
const int N = 1e6 + 7;
int n, M, f[N], g[N], h[N], phi[N], u[N], p[N]; // f[n] 为 n 的最小质因子， g[n]=f[n]^k ，
 phi[n] 为欧拉函数， u[n] 为莫比乌斯函数， h[n] 为一般积性函数

void prime(int n) {
   u[1]=phi[1]=1,h[1]=(0); // 1 的时候特判
   rep(i, 2, n+1) {
      if (!f[i]) {
         p[++M]=i;
         f[i] = g[i] = i;
         phi[i] = i - 1;
         u[i] = -1;
         h[i] = (0);
      }// 质数的时候特判
      for (int j = 1, k; j <= M && p[j] <= f[i] && i * p[j] <= n; j++){
         f[k = i * p[j]] = p[j];
         if (p[j] < f[i]) {
            g[k] = p[j];
            phi[k] = phi[i] * phi[p[j]];
            u[k] = u[i] * u[p[j]];
            h[k] = h[i] * h[p[j]];
         }else {
            g[k] = g[i] * p[j];
            phi[k] = phi[i] * p[j];
            u[k] = 0;
            h[k] = h[i / g[i]] * (0);
         } /* 质数次幂特判 */
         //phi[i*p[j]]=phi[i]*(p[j]<f[i]?phi[p[j]]:p[j]);
         //u[i*p[j]]=u[i]*(p[j]<f[i]?u[p[j]]:0);
      }
      /*phi[i*j]=phi[i]*phi[j] (gcd(i,j)=1)
               phi[i]*j (j|i)
        u[i*j]=u[i]*u[j] (gcd(i,j)=1)
               0 (j|i)
      */
```

```
   }
}
```

7.15 NTT

```
const int M = 1 << 17 << 1;
int a[M], b[M];

struct NTT{
   static const int G = 3, P = 1004535809; //P = C*2^k + 1
   int N, na, nb, w[2][M], rev[M];
   ll kpow(ll a, int b){
      ll c = 1;
      for (; b; b >>= 1,a = a * a % P) if (b & 1) c = c * a %P;
      return c;
   }
   void FFT(int *a, int f){
      rep(i, 0, N) if (i < rev[i]) swap(a[i], a[rev[i]]);
      for (int i = 1; i < N; i <<= 1)
         for (int j = 0, t = N / (i << 1); j < N; j += i << 1)
            for (int k = 0, l = 0, x, y; k < i; k++, l += t)
               x = (ll) w[f][l] * a[j+k+i] % P, y = a[j+k], a[j+k] = (y+x) % P, a[j+k+i]
                = (y-x+P) % P;
      if (f) for (int i = 0, x = kpow(N, P-2); i < N; i++) a[i] = (ll)a[i] * x % P;
   }
   void work(){
      int d = __builtin_ctz(N);
      w[0][0] = w[1][0] = 1;
      for (int i = 1, x = kpow(G, (P-1) / N), y = kpow(x, P-2); i < N; i++) {
         rev[i] = (rev[i>>1] >> 1) | ((i&1) << (d-1));
         w[0][i] = (ll)x * w[0][i-1] % P, w[1][i] = (ll) y * w[1][i-1] % P;
      }
   }
   void doit(int *a, int *b, int na, int nb){ // [0, na)
      for (N = 1; N < na + nb - 1; N <<= 1);
      rep(i, na, N) a[i] = 0;
      rep(i, nb, N) b[i] = 0;
      work(), FFT(a,0), FFT(b,0);
      rep(i, 0, N) a[i] = (ll)a[i] * b[i] % P;
      FFT(a, 1);
      //rep(i, 0, N) cout << a[i] << endl;
   }
} ntt;
```

7.16 Polya

```
/*

* Burnside's lemma

* 首先列出所有可能的染色方案，然后找出每个置换下保持不变的方案（不动点）数。
* 等价类数目：所有置换的不动点数的平均值。

* Polya enumeration theorem

* 一个循环的颜色需相同

*/
```

7.17 SternBrocotTree

```
namespace SBT {
   typedef long double db;
   typedef int U;
   typedef pair<U, U> pii;
   const U INF = 1e9 + 7;
   typedef __int128 T;
   typedef pair<T, T> V; // V = [double|long double|fraction]
   inline int cmp(const V &a, const V &b) {
      T x = a.fi * b.se - a.se * b.fi;
      return (x > 0) - (x < 0);
   }
   inline bool in(const V &a, const V &b, const V &c) {
      return 0 <= cmp(c, a) && cmp(c, b) < 0;
   }
   pii operator+(const pii &a, const pii &b) {
      return mp(a.fi + b.fi, a.se + b.se);
   }
   pii operator*(const pii &a, U x) {
      return mp(a.fi * x, a.se * x);
   }
   bool search(V v, U MAXB, pii &lo, pii &hi, int f) {
      V x;
      U l = 0, r = f > 0 ? (hi.se ? (MAXB - lo.se) / hi.se : INF) :
         (lo.se ? (MAXB - hi.se) / lo.se : INF);
      while (l + 1 < r) {
         U z = (l + r) >> 1;
         x = f > 0 ? lo + hi * z : lo * z + hi;
         f * cmp(x, v) <= 0 ? l = z : r = z;
      }
      x = f > 0 ? lo + hi * r : lo * r + hi;
      r = f * cmp(x, v) <= 0 ? r : l;
      f > 0 ? lo = lo + hi * r : hi = lo * r + hi;
      return r > 0;
   }
   pii solve(V v, U MAXB) { // find ROUND_HALF_UP(a / b) = v, b <= MAXB
      V L = mp(v.fi * 10 - 5, v.se * 10);
      V R = mp(v.fi * 10 + 5, v.se * 10);
      pii lo(0, 1), hi(1, 0);
      while (true) {
         bool ok = 0;
         //V m = mp(lo.fi + hi.fi, lo.se + hi.se);
         //if (in(L, R, m)) return mp(m.fi, m.se);
         ok |= search(v, MAXB, lo, hi, 1);
         ok |= search(v, MAXB, lo, hi, -1);
         if (!ok) break;
      }
      db t1 = (db) lo.fi / lo.se;
      db t2 = (db) hi.fi / hi.se;
```

```
    db t3 = (db) v.fi / v.se;
    if (t2 - t3 <= t3 - t1) return hi;else return lo;
    //if (in(L, R, lo)) return lo;
    //if (in(L, R, hi)) return hi;
    return mp(-1, -1);
  }
};
```

7.18 min_25

```
struct Min_25{
  // F(i) 要拆成多个完全积性函数的和
  static const int N = 1e6 + 7;
  int Sqr, m, p[N], id1[N], id2[N], tot, cntp;
  ll g[N], sp[N], h[N], n, w[N];
  bool isp[N];
  // f(p) = p ^ k
  ll f(int p) { return 1;}

  // 要求的积性函数 F(p ^ e)
  ll F(int p, int e) { return e == 1 ? -1 : 0;}

  // 假设都是质数的完全积性函数前缀和去掉  f(1)
  ll calc(ll n) { return n - 1;}

  void prime(int n){
    cntp = 0;isp[1] = 1;
    rep(i, 2, n+1) {
      if(!isp[i]) p[++cntp] = i;
      for(int j = 1; j <= cntp && i * p[j] <= n; j++){
            isp[i * p[j]] = 1;
            if(i % p[j] == 0)break;
      }
    }
    rep(i, 1, cntp+1) sp[i] = sp[i - 1] + f(p[i]);
    p[++cntp] = INT_MAX;
  }

  ll S(ll x, int y){
    if(x <= 1 || p[y] > x) return 0;
    int k = (x <= Sqr ? id1[x] : id2[n/x]);
    ll ret = -(g[k] - sp[y-1]);// 质数的答案
    for(int i = y; i <= tot && 1ll * p[i] * p[i] <= x; i++){
      ll t1 = p[i], t2 = 1ll * p[i] * p[i];
      for(int e = 1; t2 <= x; e++, t1 = t2, t2 *= p[i]) {
        if (F(p[i], e)) ret += S(x / t1, i + 1) * F(p[i], e);
        ret += F(p[i], e + 1);// 合数的答案
      }
    }
    return  ret;
  }

  ll solve(ll _n) {
    n = _n;if (n == 0) return 0;
    m = 0;Sqr = sqrt(n);
    tot = upper_bound(p + 1, p + cntp + 1, Sqr) - (p + 1);
    for(ll i = 1, j; i <= n; i = j + 1){
      j = n / (n / i);
      w[++m] = n / i;
      g[m] = calc(w[m]);
      w[m] <= Sqr ? id1[w[m]] = m : id2[j] = m;
    }
    rep(j, 1, tot + 1)
      for(int i = 1; i <= m && 1ll * p[j] * p[j] <= w[i]; i++){
        ll t = w[i] / p[j];
        int k = t <= Sqr ? id1[t] : id2[n / t];
        g[i] -= f(p[j]) * (g[k] - sp[j - 1]);
      }
    return S(n,1) + 1;
  }
} _U;
```

7.19 ploynomial

```
template<class T>
struct polynomial{
  static const int N = 101010;
  static const int P = 998244353;
  T a1[N], b1[N], c[N], a[N], pre[N], suf[N], ifac[N], fac[N];
  T add(T a, T b) {a = (a + b) % P; return a < 0 ? a + P : a;}
  T mul(T a, T b) {a = 1ll * a * b % P; return a < 0 ? a + P : a;}
  T kpow(T a, T b) {T r=1;for(;b;b>>=1,a=mul(a,a)) {if(b&1)r=mul(r,a);}return r;}
  void calc(int n, T *a, T *b) {
    fill_n(c, n+1, 0);
    rep(i, 0, n+1) rep(j, 0, 2) c[i+j] = add(c[i+j], mul(a[i], b[j]));
    memcpy(a, c, sizeof(a[0]) * (n+1));
  }
  void solve(int n, T *x, T *y){ // a[0]*x^0 ... a[n]*x^n
    fill_n(a, n+1, 0);
    rep(i, 0, n+1) {
      fill_n(a1, n+1, 0); a1[0] = 1;
      rep(j, 0, n+1) if (j != i) a1[0] = mul(a1[0], x[i] - x[j]);
      a1[0] = mul(y[i], kpow(a1[0], P - 2));
      rep(j, 0, n+1) if (j != i) {
        b1[0] = -x[j]; b1[1] = 1;
        calc(n, a1, b1);
      }
      rep(j, 0, n+1) a[j] = add(a[j], a1[j]);
    }
  }
  T get(int n, int k, T *x, T *y) { // f(k)
    T res = 0;
    rep(i, 0, n+1) {
      T s1 = y[i], s2 = 1;
      rep(j, 0, n+1) if (j != i) s1 = mul(s1, k - x[j]);
      rep(j, 0, n+1) if (j != i) s2 = mul(s2, x[i] - x[j]);
      res = add(res, mul(s1, kpow(s2, P - 2)));
    }
    return res;
  }
```

```
   T get(int n, int k, T *y) { // x is [1..n]
      fac[0] = 1;rep(i, 1, n+1) fac[i] = mul(fac[i-1], i);
      ifac[n] = kpow(fac[n], P - 2);
      per(i, 0, n) ifac[i] = mul(ifac[i+1], i+1);
      pre[0] = suf[n+1] = 1;
      rep(i, 1, n+1) pre[i] = mul(pre[i-1], k - i);
      per(i, 1, n+1) suf[i] = mul(suf[i+1], k - i);
      T ans=0;
      rep(i, 1, n+1){
         T s1 = mul(pre[i-1], suf[i+1]);
         T s2 = mul(ifac[i-1], ifac[n-i]);
         T fg = (n-i)&1 ? -1 : 1;
         ans = add(ans, mul(fg*s1, mul(s2, y[i])));
      }
      return ans;
   }
};
```

7.20 polysum

```
struct polysum {
    static const int D = 101000;
    static const int P = 998244353;
    ll a[D], fac[D], ifac[D], p1[D], p2[D], h[D][2], C[D];
    ll add(ll a, ll b) {a = (a + b) % P; return a < 0 ? a + P : a;}
   ll mul(ll a, ll b) {a = 1ll * a * b % P; return a < 0 ? a + P : a;}
   ll kpow(ll a, ll b) {ll r=1;for(;b;b>>=1,a=mul(a,a)) {if(b&1)r=mul(r,a);}return r;}
    void init(int M) {
        fac[0] = 1; rep(i, 1, M+5) fac[i] = mul(fac[i-1], i);
        ifac[M+4] = kpow(fac[M+4], P - 2);
        per(i, 0, M+4) ifac[i] = mul(ifac[i+1], i+1);
    }
    ll calcn(int d, ll *a, ll n) { // a[0].. a[d]  a[n]
        if (n <= d) return a[n];
        p1[0] = p2[0] = 1;
        rep(i, 0, d+1) p1[i+1] = mul(p1[i], (n - i) % P);
        rep(i, 0, d+1) p2[i+1] = mul(p2[i], (n - d + i) % P);
        ll ans=0;
        rep(i, 0, d+1) {
         ll s1 = mul(p1[i], p2[d - i]);
         ll s2 = mul(ifac[i], ifac[d - i]);
         ll t = mul(mul(s1, s2), a[i]);
            ans = (d-i)&1 ? add(ans, -t) : add(ans, t);
        }
        return ans;
    }
    ll Polysum(ll n, ll *a, ll m) { // a[0].. a[m] \sum_{i=0}^{n-1} a[i]
        a[m+1] = calcn(m, a, m+1);
        rep(i, 1, m+2) a[i] = add(a[i-1], a[i]);
        return calcn(m+1, a, n-1);
    }
    ll qpolysum(ll R, ll n, ll *a, ll m) { // a[0].. a[m] \sum_{i=0}^{n-1} a[i]*R^i
        if (R == 1) return Polysum(n, a, m);
        a[m+1] = calcn(m, a, m+1);
        ll r = kpow(R, P - 2), p3 = 0, p4 = 0, c, ans;
        h[0][0] = 0; h[0][1] = 1;
        rep(i, 1, m+2) {
            h[i][0] = mul(h[i-1][0] + a[i-1], r);
            h[i][1] = mul(h[i-1][1], r);
        }
        rep(i, 0, m+2) {
            ll t = mul(ifac[i], ifac[m+1-i]);
            p3 = i & 1 ? add(p3, -mul(h[i][0], t)) : add(p3, mul(h[i][0], t));
            p4 = i & 1 ? add(p4, -mul(h[i][1], t)) : add(p4, mul(h[i][1], t));
        }
        c = mul(kpow(p4, P - 2), -p3);
        rep(i, 0, m+2) h[i][0] = add(h[i][0], h[i][1] * c);
        rep(i, 0 ,m+2) C[i] = h[i][0];
        ans = add(mul(calcn(m, C, n), kpow(R, n)), -c);
        return ans;
    }
};
```

7.21 prime

```
// time : O(n)
// low[] : optional
const int N = 1e6 + 6;
int low[N],cntp,p[N];
bool isp[N];

void getprime() {
   fill_n(isp + 2, N - 2, 1);
   rep(i, 2, N) {
      if (isp[i]) p[cntp++]=i;
      for (int j=0;j<cntp&&p[j]*i<N;j++){
         //low[p[j] * i] = p[j];
         isp[p[j] * i] = 0;
         if (i % p[j] == 0) break;
      }
   }
}

// 优化版欧拉筛法 bitset 需要 O2
const int N = 3e7 + 6;
const int M = 2e6 + 6;
//int low[N],
bitset<N / 3 + 1> isp;
int cntp,p[M];

void getprime(int N) {
   cntp = 2;p[0] = 2;p[1] = 3;
   for (int i = 5, k = 1; i <= N; (k & 1) ? i+=2 : i+=4 , k++){
      if (!isp[k]) {
         p[cntp++]=i;
         // low[i] = i;
      }
      for (int j = 2; j < cntp && p[j] * i <= N; j++) {
         //low[p[j] * i] = p[j];
         isp[p[j] * i / 3] = 1;
```

```
        if (i % p[j] == 0) break;
      }
   }
}

// 优化埃氏筛法空间最小可以不存质数
const int N = 3e8 + 6;
const int M = 2e7 + 6;
int cntp,p[M];
bitset<N / 3 + 1> bit;

void getprime(int n){
   int i, j;
   cntp = 2; p[0] = 2; p[1] = 3;
   for(i = 5, j = 1; i * i <= n; (j & 1) ? i += 2 : i += 4 , j++) {
      if(bit[j] == 0) {
         p[cntp++]=i;
         for(int j = i * i; j <= n ; j += i)
            if(j % 2 != 0 && j % 3 != 0) bit[j / 3] = 1;
      }
   }
   for( ; i <= n; (j&1) ? i+=2 : i+=4 , j++) if(bit[j] == 0) p[cntp++]=i;
}
```

8 Others

8.1 BitOperation

```
// 枚举子集
for(int i = x; i; (--i) & x) {
    //
}
// 统计子集的答案
rep(i, 0, n) {
    rep(j, 0, 1 << n) if(j >> i & 1) {
        upd(s[j], s[j ^ (1 << i)]);
    }
}
// 统计超集的答案
rep(i, 0, n) {
    for(int j = (1 << n) - 1; ~j; --j) if(!(j >> i & 1)) {
        upd(s[j], s[j | (1 << i)]);
    }
}

//
int __builtin_ffs (unsigned int x)
int __builtin_ffsl (unsigned long)
int __builtin_ffsll (unsigned long long)
Returns one plus the index of the least significant 1-bit of x, or if x is zero, returns
 zero.

//
int __builtin_clz (unsigned int x)
Returns the number of leading 0-bits in x, starting at the most significant bit position
. If x is 0, the result is undefined.

//
int __builtin_ctz (unsigned int x)
Returns the number of trailing 0-bits in x, starting at the least significant bit
position. If x is 0, the result is undefined.

//
int __builtin_popcount (unsigned int x)
Returns the number of 1-bits in x.

//
int __builtin_parity (unsigned int x)
Returns the parity of x, i.e. the number of 1-bits in x modulo 2.
```

8.2 Bitset

```
// Base
   b.any();   // has 1 ?
   b.none();  // all 0 ?
   b.count(); // cnt of 1

   b.set();   // all to 1
   b.reset(); // all to 0
   b.flip();  // all = 0 <-> 1

   b.set(p);  // b[p] = 1
   b.test(p); // b[p] is 1
   b.reset(p);// b[p] = 0
   b.flip(p); // b[p] = 0 <-> 1

// Black tech
   // __builtin_ctz in bitst
   b._Find_first();

   // travel all 1
   for (int i = b._Find_first(); i < sz(b); i = b._Find_next(i));
```

8.3 RomanNumerals

```
const int rom[30] = {
   3000,2000,1000,900,800,700,600,500,400,300,200,100,
   90,80,70,60,50,40,30,20,10,
   9,8,7,6,5,4,3,2,1
};
string smb[30]={
   "MMM","MM","M",
   "CM","DCCC","DCC","DC","D","CD","CCC","CC","C",
   "XC","LXXX","LXX","LX","L","XL","XXX","XX","X",
   "IX","VIII","VII","VI","V","IV","III","II","I"
};
string toRoman(ll d) {
   string r;
```

```
    rep(i, 0, 30) if (d >= rom[i]) d -= rom[i], r += smb[i];
    return r;
}
```

8.4 Strtok

```
char s[111];
gets(s);
vector<string> a;
for(char* p=strtok(s," .,()");p;p=strtok(NULL," .,()")) a.pb(p);
```

9 String

9.1 ACAutomaton

```
/*
* [0,L) , N-1 is virtual , 0 is rt
* init!!
* addation: end[] end[c]|=end[fail[c]]
*/
struct Trie{
    static const int N = 101010 , M = 26;
    int ne[N][M] , fail[N] , fa[N] , rt , L;
    void ini(){ fill_n(ne[fail[0] = N-1],M,0);L = 0;rt = newnode();}
    int newnode(){ fill_n(ne[L],M,0); return L++; }
    void add(char *s){
        int p = rt;
        for(int i=0;s[i];++i){
            int c = s[i] - 'a';// modify
            if(!ne[p][c]) ne[p][c] = newnode() , fa[L-1] = p;
            p = ne[p][c];
        }
    }
    void Build(){
        vi v;v.pb(rt);
        rep(i,0,sz(v)){
            int c = v[i];
            rep(i,0,M) ne[c][i] ?
                v.pb(ne[c][i]) , fail[ne[c][i]] = ne[fail[c]][i] :
                ne[c][i] = ne[fail[c]][i];
        }
    }
};
```

9.2 DoublingArray

```
namespace Doubling{
    static const int N = 101010;
   // sa[0~n]: 排名第的后缀是以i sa[i] 开头
   // h[1~n]:S[sa[i-1]] 与 S[sa[i]] 的最长公共前缀长度为 h[i]
    int t[N] , wa[N] , wb[N] , sa[N] , h[N];
    void sort(int *x,int *y,int n,int m){
        rep(i,0,m) t[i] = 0;
        rep(i,0,n) t[x[y[i]]]++;
        rep(i,1,m) t[i] += t[i-1];
        per(i,0,n) sa[--t[x[y[i]]]] = y[i];
    }
    bool cmp(int *x,int a,int b,int d){
        return x[a] == x[b] && x[a+d] == x[b+d];
    }
    void da(int *s,int n,int m){
        int *x=wa,*y=wb;
        rep(i,0,n) x[i] = s[i] , y[i] = i;
        sort(x , y , n , m);
        for(int j=1,p=1;p<n;m=p,j<<=1){
            p = 0;rep(i,n-j,n) y[p++] = i;
            rep(i,0,n) if(sa[i] >= j) y[p++] = sa[i] - j;
            sort(x , y , n , m);
            swap(x , y);p = 1;x[sa[0]] = 0;
            rep(i,1,n) x[sa[i]] = cmp(y,sa[i],sa[i-1],j)?p-1:p++;
        }
    }
    void cal_h(int *s,int n,int *rk){
        int j,k=0;
        for(int i=1;i<=n;++i) rk[sa[i]] = i;
        for(int i=0;i<n;h[rk[i++]] = k)
            for(k&&--k,j=sa[rk[i]-1];s[i+k]==s[j+k];++k);
    }
}
// rank[0~n-1]: 以 i 开头的后缀排名 rank[i]
struct DA{ // [0,n] , in[n] = 0 , n load
    static const int N = 101010;
    int p[18][N] , rk[N] , in[N] , Log[N] , n;
    void Build(){
        Doubling::da(in,n+1,300);
        Doubling::cal_h(in,n,rk);
        Log[0] = -1;for(int i=1;i<=n;++i) Log[i] = Log[i-1] + (i==(i&(-i)));
        for(int i=1;i<=n;++i) p[0][i] = Doubling::h[i];
        for(int j=1;1<<j<=n;++j){
            int lim = n+1-(1<<j);
            for(int i=1;i<=lim;++i)
                p[j][i] = min(p[j-1][i] , p[j-1][i+(1<<j>>1)]);
        }
    }
    // 某两个后缀的最长公共前缀
    int lcp(int a,int b){
        a = rk[a] , b = rk[b];
        if(a > b) swap(a , b);++a;
        int t = Log[b-a+1];
        return min(p[t][a] , p[t][b-(1<<t)+1]);
    }
};
```

9.3 Exkmp

```
/*
 * s 串的每个后缀与 t 串的最长公共前缀
 * t:  a  b  a
```

```
 * nt: 0  0  1
 * s:  a  b  a  c  a  b  a
 * ns: 3  0  1  0  3  0  1
 */
void exkmp(char *s,int *z,char *t,int *p){
    int lens = strlen(s);
    int lent = strlen(t);
    p[0]=0;
    for(int i=0,x=0,y=0;i<lens;++i){
        z[i] = i <= y ? min(y-i,p[i-x]) : 0;
        while(i + z[i] < lens && z[i] < lent && s[i + z[i]] == t[z[i]]) ++z[i];
        if(y <= i + z[i]) x = i, y = i + z[i];
    }
}

void Exkmp(){
    scanf("%s%s",s,t);
    exkmp(t+1,nt+1,t,nt);
    exkmp(s,ns,t,nt);
}
```

9.4 Kmp

```
/*
t:  a  b  a
nt:-1 -1  0
s:  a  b  a  c  a  b  a
ns: 0  1  2 -1  0  1  2
*/
void kmp(char *s,int *ns,char *t,int *nt){
    int lens = strlen(s);
    int lent = strlen(t);
    nt[0] = -1;
    for(int i=0,j=-1;i<lens;++i){
        while(j >= 0 && s[i] != t[j + 1]) j = nt[j];
        if(s[i] == t[j + 1]) ++j;
        ns[i] = j;
        if(j + 1 == lent) j = nt[j];
    }
}
void KMP(){
    scanf("%s%s",s,t);
    kmp(t+1,nt+1,t,nt);
    kmp(s,ns,t,nt);
}
```

9.5 Manacher

```
/*
 * length of pa is two size of str
 * i: [0, n)     pa[i<<1] : odd string 整个回文长度为 2*pa[i<<1]-1
 * i: [0, n - 1) pa[i<<1|1] : even string 整个回文长度为 2*pa[i<<1]
 * N>2*n
*/
void Manacher(char *s,int n,int *pa){
    pa[0] = 1;
    for(int i=1,j=0;i<(n<<1)-1;++i){
        int p = i >> 1 , q = i - p , r = ((j + 1)>>1) + pa[j] - 1;
        pa[i] = r < q ? 0 : min(r - q + 1 , pa[(j<<1) - i]);
        while(0 <= p - pa[i] && q + pa[i] < n && s[p - pa[i]] == s[q + pa[i]])
            pa[i]++;
        if(q + pa[i] - 1 > r) j = i;
    }
}
```

9.6 PalindromicTree

```
// [0,p) , 0(even) and 1(odd) is virtual , init!!
struct Palindromic_Tree {
   static const int N = ::N , M = 26;
   int ne[N][M] , fail[N] , len[N] , S[N] , last , n , p, cnt[N], las[N];
   int newnode(int l){
      fill(ne[p] , ne[p] + M , 0);
      len[p] = l;
      las[p] = n;
      cnt[p] = 0;
      return p++;
   }
   void ini(){
      p = 0;newnode(0);newnode(-1);
      S[n = last = 0] = -1;
      fail[0] = 1;
   }
   int get_fail(int x){
      while(S[n - len[x] - 1] != S[n]) x = fail[x];
      return x;
   }
   void add(int c){
      S[++n] = c;
      int cur = get_fail(last);
      if(!ne[cur][c]){
         int now = newnode(len[cur] + 2);
         fail[now] = ne[get_fail(fail[cur])][c];
         ne[cur][c] = now;
      }
      last = ne[cur][c];
      cnt[last]++;
   }
   void build() {
      for(int i = p - 1; ~i; --i) cnt[fail[i]] += cnt[i];
   }
}pam;
```

9.7 SAIS

```
/**
 * Ensure that str[n] is the unique lexicographically smallest character in str.
 * time complexity: O(n)
```

```
 */
namespace SA {
   const static int N = 100000 + 10;
   int sa[N], rk[N], ht[N], s[N<<1], t[N<<1], p[N], cnt[N], cur[N];
#define pushS(x) sa[cur[s[x]]--] = x
#define pushL(x) sa[cur[s[x]]++] = x
#define inducedSort(v) std::fill_n(sa, n, -1); std::fill_n(cnt, m, 0);      \
   for (int i = 0; i < n; i++) cnt[s[i]]++;                                  \
   for (int i = 1; i < m; i++) cnt[i] += cnt[i-1];                           \
   for (int i = 0; i < m; i++) cur[i] = cnt[i]-1;                            \
   for (int i = n1-1; ~i; i--) pushS(v[i]);                                  \
   for (int i = 1; i < m; i++) cur[i] = cnt[i-1];                            \
   for (int i = 0; i < n; i++) if (sa[i] > 0 &&  t[sa[i]-1]) pushL(sa[i]-1); \
   for (int i = 0; i < m; i++) cur[i] = cnt[i]-1;                            \
   for (int i = n-1;  ~i; i--) if (sa[i] > 0 && !t[sa[i]-1]) pushS(sa[i]-1)
   void sais(int n, int m, int *s, int *t, int *p) {
      int n1 = t[n-1] = 0, ch = rk[0] = -1, *s1 = s+n;
      for (int i = n-2; ~i; i--) t[i] = s[i] == s[i+1] ? t[i+1] : s[i] > s[i+1];
      for (int i = 1; i < n; i++) rk[i] = t[i-1] && !t[i] ? (p[n1] = i, n1++) : -1;
      inducedSort(p);
      for (int i = 0, x, y; i < n; i++) if (~(x = rk[sa[i]])) {
         if (ch < 1 || p[x+1] - p[x] != p[y+1] - p[y]) ch++;
         else for (int j = p[x], k = p[y]; j <= p[x+1]; j++, k++)
            if ((s[j]<<1|t[j]) != (s[k]<<1|t[k])) {ch++; break;}
         s1[y = x] = ch;
      }
      if (ch+1 < n1) sais(n1, ch+1, s1, t+n, p+n1);
      else for (int i = 0; i < n1; i++) sa[s1[i]] = i;
      for (int i = 0; i < n1; i++) s1[i] = p[sa[i]];
      inducedSort(s1);
   }
   template<typename T>
      int mapCharToInt(int n, const T *str) {
         int m = *max_element(str, str+n);
         std::fill_n(rk, m+1, 0);
         for (int i = 0; i < n; i++) rk[str[i]] = 1;
         for (int i = 0; i < m; i++) rk[i+1] += rk[i];
         for (int i = 0; i < n; i++) s[i] = rk[str[i]] - 1;
         return rk[m];
      }
   template<typename T>
      void suffixArray(int n, const T *str) {
         int m = mapCharToInt(++n, str);
         sais(n, m, s, t, p);
         for (int i = 0; i < n; i++) rk[sa[i]] = i;
         for (int i = 0, h = ht[0] = 0; i < n-1; i++) {
            int j = sa[rk[i]-1];
            while (i+h < n && j+h < n && s[i+h] == s[j+h]) h++;
            if (ht[rk[i]] = h) h--;
         }
      }
};
// 出现 i 次的子串有 c[i] 个
namespace S {
   int top;
   int sta[N<<1], cnt[N<<1];
   ll c[N];
   inline int gao(int k) {
      int cc = 0;
      while(top && sta[top] > k) {
         cc += cnt[top];
         cnt[top] = 0;
         c[cc] += sta[top] - max(k, sta[top-1]);
         --top;
      }
      return cc;
   }
   inline void push(int x, int y) {
      if(!top || sta[top] != x) sta[++top] = x;
      cnt[top] += y;
   }
   inline void build(int n) {
      top = 0;
      fill_n(c+1, n, 0);
      rep(i, 1, n+1) {
         int lcp = SA::ht[i], cc = gao(lcp);
         push(lcp, cc);
         push(n - SA::sa[i], 1);
      }
      gao(0);
   }
};
```

9.8 StrHash

```
inline int upd(int a, int b, int P) {
   if((a += b) >= P) a -= P;
   return a < 0 ? a + P : a;
}
inline int mul(int a, int b, int P) {return 1ll * a * b % P; }

struct Int{
   static const int P = 1e9 + 7, Q = 1e9 + 9;
   int a, b;
   Int(int a = 0, int b = 0) : a(a), b(b) {}
   inline Int operator + (const Int &c) const { return Int(upd(a, c.a, P), upd(b, c.b, Q
   )); }
   inline Int operator - (const Int &c) const { return Int(upd(a, -c.a, P), upd(b, -c.b,
    Q)); }
   inline Int operator * (const Int &c) const { return Int(mul(a, c.a, P), mul(b, c.b, Q
   )); }
   inline bool operator == (const Int &c) const {return a == c.a && b == c.b;}
} _0 = Int(), _1 = Int(1, 1), B[N];

void init(int n){
   B[0] = _1; B[1] = Int(233, 233);
   rep(i, 2, n+1) B[i] = B[i-1] * B[1];
}

struct Str{
```

```
   Int a; int len;
   Str(Int a = _0, int len = 0) : a(a), len(len) {}
   Str(int x) {a = Int(x, x); len = 1;}
   inline Str operator + (const Str &c) const { return Str(a * B[c.len] + c.a, len + c.
   len); }
   inline Str operator - (const Str &c) const { return Str(a - c.a * B[len - c.len], len
    - c.len); } // 减去一个前
   缀
   inline bool operator == (const Str &c) const { return a == c.a && len == c.len;}
} ha[N], hb[N];

void init(vi &s, Str *ha) {
   rep(i, 0, sz(s)) ha[i] = i > 0 ? ha[i-1] + Str(s[i] + 1) : Str(s[0] + 1);
}

Str sub(Str *ha, int l, int r) {
   if (l > r) return Str();
   return l > 0 ? ha[r] - ha[l-1] : ha[r];
}
```

9.9 SuffixAutomaton

```
/*
 * [0,L] , 0 is virtual , 1 is rt , init!!
 * [l[par[s]] + 1, l[s]]
 * 好像暴力向上跳的复杂度是对的。
 */
struct SAM {
   static const int N = ::N << 1, M = 26;
   int par[N], l[N], ne[N][M];
   int rt, last, L;
   void add(int c) {
      int p = last;
      /* ex
      if(ne[p][c] && l[ne[p][c]] == l[p] + 1) {
         last = ne[p][c];
         return ;
      }
      */
      int np = ++L;
      fill(ne[np], ne[np] + M, 0);
      l[np] = l[p] + 1;
      last = np;
      while(p && !ne[p][c]) ne[p][c] = np, p = par[p];
      if(!p) par[np] = rt;
      else{
         int q = ne[p][c];
         if(l[q] == l[p] + 1) par[np] = q;
         else {
            int nq = ++L;
            l[nq] = l[p] + 1;
            copy(ne[q], ne[q] + M, ne[nq]);
            par[nq] = par[q];
            par[q] = par[np] = nq;
            while(p && ne[p][c] == q) ne[p][c] = nq, p = par[p];
         }
      }
   }
   void ini() {
      rt = last = L = 1;
      fill(ne[rt], ne[rt] + M, 0);
      l[0] = -1;
   }
};
// BucketSort
rep(i, 1, L + 1) ++cnt[l[i]];
rep(i, 1, L + 1) cnt[i] += cnt[i - 1];
rep(i, 1, L + 1) cur[cnt[l[i]]--] = i;
```

10 Tree

10.1 Centroid

```
// id starts from 1
namespace Centriod {
    const int N = ::N;
    bool vis[N];
   int sz[N];
    void dfssz(int c,int fa,int Sz,int &rt){
        sz[c] = 1;
        for(auto t : g[c]) if(!vis[t]&&t!=fa) dfssz(t,c,Sz,rt) , sz[c]+=sz[t];
        if(!rt && sz[c]*2>Sz) rt=c;
    }
    void dfs(int c){
        int rt=0;dfssz(c,0,0,rt);dfssz(c,0,sz[c],rt=0);
        vis[rt] = true;
        /*
         * 注意计算以 rt 为起点的路径、只包含 rt 的路径
        * 注意 v != vis[rt]
        */
        for(auto t : g[rt]) if(!vis[t]) dfs(t);
    }
};
```

10.2 DsuOnTree

```
// id starts with 1
namespace QuerySubtree{
    static const int N = ::N;
    int sz[N] , wson[N] , par[N];
    void dfs(int c,int fa,vi g[]){
        sz[c]=1;par[c]=fa;int &s=wson[c]=0;
        for(auto t:g[c]) if(t!=fa)
            dfs(t,c,g),sz[c]+=sz[t],(sz[t]>=sz[s])&&(s=t);
    }
   vi nd;
    void solve(int c,int fa,bool iswson,vi g[]){
        for(auto t : g[c]) if(t != wson[c] && t != fa) solve(t , c , false , g);
```

```
        if(wson[c]) solve(wson[c] , c , true , g);
        for(auto t : g[c]) if(t != wson[c] && t != fa) {
            // 将该子树的信息加入
        }
         // 将当前节点的信息加入
         // 查询
        if(!iswson) {
         // 删除整棵子树的信息 nd.clear()
        }
    }
    void solve(vi g[]){
        dfs(1,0,g);
        solve(1,0,false,g); // 如果输入是单组数据，改成 true 可以优化
    }
}
```

10.3 HeavyChain

```
// id starts with 1
struct HeavyChain{
   static const int N = ::N;
   int sz[N], wson[N], top[N], dep[N], id[N], _, par[N], who[N];
   void dfs(int c, int fa, vi g[]){
      sz[c] = 1;
      par[c] = fa;
      dep[c] = dep[fa] + 1;
      int &s = wson[c] = top[c] = 0;
      for(auto t : g[c]) if(t != fa) {
         dfs(t, c, g);
         sz[c] += sz[t];
         if(sz[t] >= sz[s]) s = t;
      }
   }
   void dfs2(int c, int fa, vi g[]){
      id[c] = ++_;
      who[_] = c;
      int s = wson[c];
      if(!top[c]) top[c] = c;
      if(s) top[s] = top[c], dfs2(s, c, g);
      for(auto t : g[c]) if(t != fa && t != s) dfs2(t, c, g);
   }
   void Query(int a, int b){
      int fa = top[a], fb = top[b];
      while(fa != fb){
         if(dep[fa] < dep[fb]) swap(a, b), swap(fa, fb);
         // Cal id[fa] .. id[a]
         a = par[fa]; fa = top[a];
      }
      if(dep[a] < dep[b]) swap(a, b);
      // Cal id[b] .. id[a]
      // b is lca
   }
   void Build(vi g[]){
      dfs(1, 0, g);
      _=0;
      dfs2(1, 0, g);
   }
}hc;
```

10.4 LCARMQ

```
// N is 2 size of tree , id of nodes start from 1
struct LCARMQ{
    static const int N = 101010 << 1;
    int a[20][N] , lft[N] , dep[N] , lg[N] , L;
    int rmin(int x,int y){return dep[x] < dep[y] ? x : y;}
    void add(int x){ a[0][L++] = x;}
    void dfs(int c,int fa,const vi g[]){
        lft[c]=L;add(c);
        for(auto t : g[c]) if(t!=fa) dep[t]=dep[c]+1,dfs(t,c,g),add(c);
    }
    void Build(const vi g[]){
        L = 0;dfs(1,0,g);dep[0] = -1;
        rep(i,2,L) lg[i]=lg[i>>1]+1;
        rep(i,1,20){
            int lim = L+1-(1<<i);
            rep(j,0,lim) a[i][j] = rmin(a[i-1][j] , a[i-1][j+(1<<i>>1)]);
        }
    }
    int lca(int x,int y){
        x = lft[x] , y = lft[y];
        if(x > y) swap(x , y);
        int i = lg[y-x+1];
        return rmin(a[i][x] , a[i][y+1-(1<<i)]);
    }
};
```

10.5 LongChain

```
struct LongChain{
   static const int N = ::N;
   int wson[N] , top[N] , dep[N] , lg[N];
   int jump[N][20] , id[N] , who[N] , rwho[N] , _;
   void dfs(int c,int fa,vi g[]){
      dep[c]=1;int &s=wson[c]=top[c]=0;
      jump[c][0]=fa;rep(i,1,20) jump[c][i]=jump[jump[c][i-1]][i-1];
      for(auto t:g[c]) if(t!=fa)
         dfs(t,c,g),dep[c]=max(dep[t]+1,dep[c]),(dep[t]>=dep[s])&&(s=t);
   }
   void dfs2(int c,int fa,int rc,vi g[]){
      if(!top[c]) top[c]=c,rc=c;
      who[id[c]=++_]=c;rwho[_]=rc;
      int s=wson[c];
      if(s) top[s]=top[c],dfs2(s,c,jump[rc][0],g);
      for(auto t:g[c]) if(t!=fa&&t!=s) dfs2(t,c,t,g);
   }
   void Build(vi g[]){
      dfs(1,0,g);_=0;dfs2(1,0,1,g);
      rep(i,2,N) lg[i]=lg[i>>1]+1;
```

```
   }
   void solve(int c, int fa, vi g[]) {
      for(auto t : g[c]) if(t != fa) solve(t, c, g);
      if(wson[c]) {
         // upd c by wson[c], O(1) or O(log(n))
      } else {
         // c is leaf
      }
      for(auto t : g[c]) if(t != fa && t != wson[c]) {
         // brute force upd c by t
      }
      // 注意统计以 c 为起点的链的答案，注意深度的限制（两棵子树都要注意）
   }
   // kth_par should exist
   int kth_par(int x,int k){
      if(k==0) return x;
      int j0=1<<lg[k];
      int p0=jump[x][lg[k]];
      int j1=k-j0;
      int del=id[p0]-id[top[p0]];
      if(del>=j1) return who[id[p0]-j1];
      else return rwho[id[top[p0]]+j1-del];
   }
}hc;
```

10.6 MoOnTree_Path

```
/*
 * 带修改莫队：块大小 N^(2/3) 按照 l 所在块， r 所在块， time 排序
 */
namespace MoOnTree {

   const int N = ::N, SZ = sqrt(N), M = 17;

   int cd; // starts from 1
   int dep[N], pre[N][M], st[N], ed[N], dfn[N << 1], B[N << 1], cnt[N];
   struct Node {
      int l, r, id, lca;
      Node(int id, int l, int r, int lca = 0) : id(id), l(l), r(r), lca(lca) {}
      bool operator < (const Node &c) const {
         if(B[l] != B[c.l]) return B[l] < B[c.l];
         return (r < c.r) ^ (B[l] & 1);
      }
   };
   vector<Node> nds;

   void dfs(int u, int fa, vi g[]) {
      dep[u] = dep[fa] + 1;
      pre[u][0] = fa;
      for(int i = 1; i < M && pre[u][i - 1]; ++i) pre[u][i] = pre[pre[u][i - 1]][i - 1];
      dfn[++cd] = u, st[u] = cd;
      for(auto v : g[u]) if(v != fa) dfs(v, u, g);
      dfn[++cd] = u, ed[u] = cd;
   }
   int lca(int x, int y) {
      if(dep[x] > dep[y]) swap(x, y);
      per(i, 0, M) if(dep[pre[y][i]] >= dep[x]) y = pre[y][i];
      per(i, 0, M) if(pre[x][i] != pre[y][i]) x = pre[x][i], y = pre[y][i];
      if(x == y) return x;
      return pre[x][0];
   }
   void adde(int u, int v, int id) {
      if(st[u] > st[v]) swap(u, v);
      int f = lca(u, v);
      if(f == u) {
         nds.pb(Node(id, st[u], st[v]));
      } else {
         int l = ed[u], r = st[v];
         if(l > r) swap(l, r);
         nds.pb(Node(id, l, r, f));
      }
   }
   // p is index in tree
   void add(int p) {
   }
   void sub(int p) {
   }
   void upd(int p, int c) {
      p = dfn[p];
      cnt[p] += c;
      (cnt[p] == 1) ? add(p) : sub(p);
   }
   void solve(vi g[]) {
      rep(i, 0, N << 1) B[i] = i / SZ;
      dfs(1, cd = 0, g);
      // adde(u, v)
      sort(all(nds));
      int l = 1, r = 0;
      for(auto &nd : nds) {
         while(r < nd.r) upd(++r, 1);
         while(l > nd.l) upd(--l, 1);
         while(r > nd.r) upd(r--, -1);
         while(l < nd.l) upd(l++, -1);
         if(nd.lca) upd(st[nd.lca], 1);
         // save ans
         if(nd.lca) upd(st[nd.lca], -1);
      }
   }
}
```

10.7 VTree

```
// nodes should sorted in dfs order
namespace Vtree{
    const int N = 101010;
    int tp[N] , _;
    vi g[N];
    void solve(vi&v,LCARMQ&R){
        _ = 0;
        vi del;del.pb(tp[_++] = v[0]);
```

```
rep(i,1,sz(v)){
    int lca = R.lca(tp[_-1] , v[i]);
    vi l;while(_ > 0 && R.dep[lca] < R.dep[tp[_-1]]) l.pb(tp[_--]);
    if(_ == 0 || lca != tp[_-1]) del.pb(tp[_++] = lca);
    l.pb(tp[_-1]);del.pb(tp[_++] = v[i]);
    rep(i,1,sz(l)) g[l[i]].pb(l[i-1]);
}
rep(i,0,_-1) g[tp[i]].pb(tp[i+1]);
// root = tp[0]
// dfs()
for(auto t : del){
    // Cal()
    g[t].clear();
}
    }
}
```