template

WUST

So Like Coding? You Baldy

July 17, 2019

Contents

0 Header **1**
0.1 pbds . . . 1
0.2 FastIO . . . 1
0.2.1 FastScanner . . . 1
0.2.2 MLE . . . 1

1 Math **2**
1.1 Prime . . . 2
1.2 GCD . . . 2
1.2.1 ex-GCD . . . 2
1.3 CRT . . . 2
1.3.1 CRT . . . 2
1.3.2 ex-CRT . . . 2
1.4 Mobius . . . 3
1.5 Linear Basis . . . 3
1.5.1 Linear Basis . . . 3
1.6 Inv . . . 3
1.7 Phi . . . 3
1.7.1 phi . . . 3
1.8 Combinatorics . . . 4
1.8.1 Lucas . . . 4
1.9 Polygon . . . 5
1.10 BM . . . 5
1.11 Others . . . 5

2 Graph **6**
2.1 Dijkstra . . . 6
2.1.1 Dijkstra . . . 6
2.2 MST . . . 6
2.2.1 Kruskal . . . 6
2.2.2 Prim . . . 7
2.3 Components . . . 7
2.3.1 Cut Vertex . . . 7
2.3.2 Bridge . . . 8
2.3.3 Strongly Connected Components . . . 8
2.4 Bipartite Matching . . . 9
2.4.1 Hungary Algorithm . . . 9
2.4.2 Hopcroft-karp Algorithm . . . 10
2.4.3 Multiple Matching . . . 11
2.4.4 Kuhn-Munkres Algorithm . . . 12
2.4.5 Edmonds's Matching Algorithm . . . 13
2.5 Network Flows . . . 15
2.5.1 Dinic . . . 15
2.5.2 ISAP(undo) . . . 17
2.5.3 MCMF . . . 17
2.6 Directed-MST . . . 18
2.6.1 Directed-MST . . . 18
2.7 Toposort . . . 19
2.7.1 Toposort . . . 19
2.8 2-SAT . . . 20
2.8.1 2-SAT . . . 20
2.9 System of Difference Constraints . . . 20
2.9.1 System of Difference Constraints . . . 20

3 DataStructrue **22**
3.1 HLD . . . 22
3.1.1 HLD . . . 22
3.2 RMQ . . . 25
3.2.1 RMQ . . . 25
3.2.2 RMQbyIndex . . . 25
3.2.3 RMQinNM . . . 26
3.3 MO . . . 27
3.3.1 MO . . . 27
3.3.2 MObyModify . . . 28
3.4 VirtualTree . . . 29
3.4.1 VirtualTree . . . 29
3.5 PersistentDS . . . 30
3.5.1 主席树区间 k 大 . . . 30
3.6 Others . . . 31
3.6.1 BITinNM . . . 31

4 Geometry **33**
4.1 Class . . . 33
4.1.1 geo . . . 33

5 String **43**
5.1 KMP . . . 43
5.1.1 KMP . . . 43
5.1.2 exKMP . . . 43
5.2 Trie . . . 44
5.2.1 Trie . . . 44
5.2.2 Persistence Trie . . . 45
5.3 Manachar . . . 45
5.3.1 Manachar . . . 45
5.4 Aho-Corasick Automation . . . 46
5.4.1 AC Automation . . . 46
5.5 Suffix Array . . . 47
5.5.1 Suffix Array . . . 47
5.6 PalindromicTree . . . 48
5.6.1 PalindromicTree . . . 48

6 dp **51**
6.1 BitDP . . . 51
6.1.1 数位 dp 计和 . . . 51
6.1.2 一般数位 dp . . . 51
6.2 StateDP . . . 52
6.3 Subsequence . . . 52
6.3.1 MaxSum . . . 52
6.3.2 LIS . . . 52
6.3.3 LongestCommonIncrease . . . 53
6.4 Others . . . 53
6.4.1 矩阵快速幂 . . . 54

7 Others **56**

0 Header

0.1 pbds

0.2 FastIO

0.2.1 FastScanner

```cpp
// 适用于正负整数
template <class T>
inline bool scan(T &ret){
    char c;
    int sgn;
    if (c = getchar(), c == EOF) return 0; //EOF
    while (c != '-' && (c < '0' || c > '9')) c = getchar();
    sgn = (c == '-') ? -1 : 1;
    ret = (c == '-') ? 0 : (c - '0');
    while (c = getchar(), c >= '0' && c <= '9') ret = ret * 10 + (c - '0');
    ret *= sgn;
    return 1;
}

template <class T>
inline void out(T x) {
    if (x > 9) out(x / 10);
    putchar(x % 10 + '0');
}
```

0.2.2 MLE

```cpp
// 解决爆栈问题
#pragma comment(linker, "/STACK:1024000000,1024000000")
```

1 Math

1.1 Prime

1.2 GCD

1.2.1 ex-GCD

```
1  void exgcd(int a, int b, int &x, int &y)
2  {
3      if(b == 0) { x = 1; y = 0; return; }
4      exgcd(b, a % b, x, y);
5      int t = x; x = y, y = t - a / b * y;
6  }
```

1.3 CRT

1.3.1 CRT

```
1  typedef long long ll;
2
3  void exgcd(ll a, ll b, ll &x, ll &y)
4  {
5      if(b == 0) { x = 1; y = 0; return; }
6      exgcd(b, a % b, x, y);
7      ll t = x; x = y, y = t - a / b * y;
8  }
9
10 ll crt(ll *a, ll *m, int n)
11 {
12     ll M = 1, ans = 0;
13     for(int i = 1; i <= n; i ++) M *= m[i];
14     for(int i = 1; i <= n; i ++)
15     {
16         ll x = 0, y = 0;
17         ll Mi = M / m[i];
18         exgcd(Mi, m[i], x, y);
19         ans = (ans + Mi % M * x % M * a[i] % M + M) % M;
20     }
21     if(ans < 0) ans += M;
22     return ans;
23 }
```

1.3.2 ex-CRT

```
1  typedef long long ll;
2
3  const int N = 1e5 + 10;
4
5  int n;
6  ll a[N], r[N];
7
8  ll exgcd(ll a, ll b, ll& x, ll& y)
9  {
10     if(b == 0) { x = 1, y = 0; return a; }
11     ll ret = exgcd(b, a % b, y, x); y -= a / b * x;
12     return ret;
13 }
```

```

ll excrt()
{
    ll M = a[1], R = r[1], x, y, d;
    for(int i = 2; i <= n; i ++)
    {
        d = exgcd(M, a[i], x, y);
        if((R - r[i]) % d) return -1;
        x = (R - r[i]) / d * x % a[i];
        R -= M * x;
        M = M / d * a[i];
        R %= M;
    }
    return (R % M + M) % M;
}
```

1.4 Mobius

1.5 Linear Basis

1.5.1 Linear Basis

```
typedef long long ll;

const int MAX_BASE = 63;
const int maxn = 1e5 + 10;

int n;
ll a[maxn], b[MAX_BASE + 5];

void cal()
{
    for (int i = 0; i < n; i++)
    {
        for (int j = MAX_BASE; j >= 0; j--)
        {
            if (a[i] >> j & 1)
            {
                if (b[j]) a[i] ^= b[j];
                else
                {
                    b[j] = a[i];
                    for (int k = j - 1; k >= 0; k--) if (b[k] && (b[j] >> k & 1)) b[j] ^= b[k];
                    for (int k = j + 1; k <= MAX_BASE; k++) if (b[k] >> j & 1) b[k] ^= b[j];
                    break;
                }
            }
        }
    }
}
```

1.6 Inv

1.7 Phi

1.7.1 phi

```
//计算欧拉phi函数, phi(n)且与n互素的正整数个数
```

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

//用类似筛法的方法计算phi(1),phi(2),...,phi(n)
int phi[maxn];

void phi_table(int n)
{
    for (int i = 2; i <= n; i++) phi[i] = 0;
    phi[1] = 1;
    for (int i = 2; i <= n; i++) if (!phi[i])
        for (int j = i; j <= n; j += i)
        {
            if (!phi[j]) phi[j] = j;
            phi[j] = phi[j] / i * (i - 1);
        }
}
```

1.8 Combinatorics

1.8.1 Lucas

```cpp
const int maxn = 1e6 + 10;

ll fac[maxn], inv[maxn], facinv[maxn];

void init()
{
    fac[0] = inv[0] = facinv[0] = 1;
    fac[1] = inv[1] = facinv[1] = 1;
    for(int i = 2; i < maxn; i++)
    {
        fac[i] = fac[i - 1] * i % mod;
        inv[i] = mod - mod / i * inv[mod % i] % mod;
        facinv[i] = facinv[i - 1] * inv[i] % mod;
    }
}

ll C(int n, int k)
{
    if(k > n || k < 0) return 0;
    return fac[n] * facinv[k] % mod * facinv[n - k] % mod;
}

ll lucas(ll n, ll m)
{
    ll res = 1;
    while(n && m)
    {
        res = res * C(n % mod, m % mod) % mod;
```

```
29          n /= mod;
30          m /= mod;
31      }
32      return res;
33  }
```

1.9 Polygon

1.10 BM

1.11 Others

2 Graph

2.1 Dijkstra

2.1.1 Dijkstra

```
const int maxn = 1e5 + 10;
const int inf = 0x3f3f3f3f;

int head[maxn], dis[maxn], cnt, n;

struct Edge { int nex,to,w; }edge[20*maxn];

void add(int u,int v,int w)
{
    edge[++cnt].nex=head[u];
    edge[cnt].w=w;
    edge[cnt].to=v;
    head[u]=cnt;
}

void dijkstra(int s)
{
    priority_queue<pair<int, int>, vector<pair<int, int> >, greater<pair<int, int> > > que;
    memset(dis, 0x3f, sizeof dis);
    que.push({0, s}); dis[s] = 0;
    while(!que.empty())
    {
        auto f = que.top(); que.pop();
        int u = f.second, d = f.first;
        if(d != dis[u]) continue;
        for(int i = head[u]; ~i; i = edge[i].nex)
        {
            int v = edge[i].to, w = edge[i].w;
            if(dis[u] + w < dis[v])
            {
                dis[v] = dis[u] + w;
                que.push({dis[v], v});
            }
        }
    }
}
```

2.2 MST

2.2.1 Kruskal

```
const int maxn = 1e5 + 10;

int n, m, pre[maxn];
struct edge {int u, v, w; } es[maxn];
int Find(int x) { return x == pre[x] ? x : pre[x] = Find(pre[x]); }
bool cmp(const edge &x, const edge &y) { return x.cost < y.cost; }

int kruskal()
{
    sort(es, es + m, cmp);
    int res = 0;
    for(int i = 0; i < m; i ++)
```

```
    {
        int fx = Find(es[i].u), fy = Find(es[i].v);
        if(fx != fy) pre[fx] = fy, res += es[i].cost;
    }
    return res;
}
```

2.2.2 Prim

```
const int maxn = 1000 + 10;
const int inf = 0x3f3f3f3f;

int n, mp[maxn][maxn], cost[maxn];
bool vis[maxn];

int prim()
{
    for(int i = 0; i < n; i ++) cost[u] = inf, vis[u] = false;
    int res = 0; cost[0] = 0;
    for(;;)
    {
        int v = -1;
        for(int u = 0; u < n; u ++)
            if(!vis[u] && (v == -1 || cost[u] < cost[v])) v = u;
        if(v == -1) break;
        res += cost[v];
        vis[v] = true;
        for(int u = 0; u < n; u ++) cost[u] = min(cost[u], mp[v][u]);
    }
    return res;
}
```

2.3 Components

2.3.1 Cut Vertex

```
const int maxn = 1e4 + 10;

vector<int> edge[maxn];
int n, dfn[maxn], low[maxn], cnt = 0;
bool vis[maxn], cut[maxn];

void Tarjan(int u, int fa)
{
    dfn[u] = low[u] = ++cnt;
    vis[u] = true;
    int children = 0;
    for (int i = 0; i < edge[u].size(); i++)
    {
        int v = edge[u][i];
        if (v != fa && vis[v])
            low[u] = min(low[u], dfn[v]);
        else if (!vis[v])
        {
            Tarjan(v, u);
            children++;
            low[u] = min(low[u], low[v]);
            if (fa == -1 && children > 1) //若u是根节点且子节点数大于1
```

```
                cut[u] = true;    //u是割点
            else if (fa != -1 && low[v] >= dfn[u])    //若u不是根节点且v不能访问到u的父节点
                cut[u] = true;    //u是割点
        }
    }
}
```

2.3.2 Bridge

```
const int maxn = 1e4 + 10;

vector<int> edge[maxn];
int n, dfn[maxn], low[maxn], father[maxn], cnt = 0;
bool bridge[maxn][maxn];

void Tarjan(int u, int fa)
{
    dfn[u] = low[u] = ++cnt;
    for (int i = 0; i < edge[u].size(); i++)
    {
        int v = edge[u][i];
        if (!dfn[v])    //未访问节点v
        {
            Tarjan(v, u);
            low[u] = min(low[u], low[v]);
            if (low[v] > dfn[u]) //节点v到达祖先必须经过(u,v)
                bridge[u][v] = bridge[v][u] = true;    //(u,v)是桥
        }
        else if (fa != v)    //u的父节点不是v, (u,v)不存在重边
            low[u] = min(low[u], dfn[v]);
    }
}
```

2.3.3 Strongly Connected Components

```
const int maxn=1000+10;

vector<int> edge[maxn];

int dfn[maxn], low[maxn];
int stack[maxn], index, tot;
int belong[maxn], inde[maxn], outde[maxn], scc;
bool vis[maxn];

void add(int u, int v)
{
    edge[u].push_back(v);
    edge[v].push_back(u);
}

void Tarjan(int u)
{
    dfn[u] = low[u] = ++tot;
    stack[++index] = u;
    vis[u] = true;
    int v;
    for(int i = 0;i < edge[u].size(); i++)
    {
```

```
        v=edge[u][i];
        if(!dfn[v])
        {
            Tarjan(v);
            low[u] = min(low[v], low[u]);
        }
        else if(vis[v]) low[u] = min(low[v], dfn[u]);
    }
    if(dfn[u] == low[u])
    {
        scc++;
        do
        {
            v = stack[index--];
            vis[v] = false;
            belong[v] = scc;
        }while(v != u);
    }
}
```

2.4 Bipartite Matching

2.4.1 Hungary Algorithm

```
const int maxn = 150;

int n;
int edge[maxn][maxn];
int linker[maxn];
bool vis[maxn];

bool path(int u)
{
    for (int v = 1; v <= n; v++)
    {
        if (edge[u][v] && !vis[v])
        {
            vis[v] = true;
            if (linker[v] == -1 || path(linker[v]))
            {
                linker[v] = u;
                return true;
            }
        }
    }
    return false;
}

int hungary()
{
    int res = 0;
    memset(linker, 0xff, sizeof(linker));
    for (int i = 1; i <= n; i++)
    {
        memset(vis, false, sizeof(vis));
        res += path(i);
    }
    return res;
}
```

2.4.2 Hopcroft-karp Algorithm

```
const int MAXN = 3010;//左边节点数量、右边节点数量
const int MAXM = 3010 * 3010;//边的数量
const int INF = 0x7FFFFFFF;

struct Edge
{
    int v;
    int next;
} edge[MAXM];

int nx, ny;
int cnt;
int dis;

int first[MAXN];
int xlink[MAXN], ylink[MAXN];
/*xlink[i]表示左集合顶点所匹配的右集合顶点序号，ylink[i]表示右集合i顶点匹配到的左集合顶点序号。*/
int dx[MAXN], dy[MAXN];
/*dx[i]表示左集合i顶点的距离编号，dy[i]表示右集合i顶点的距离编号*/
int vis[MAXN]; //寻找增广路的标记数组

void init()
{
    cnt = 0;
    memset(first, -1, sizeof(first));
    memset(xlink, -1, sizeof(xlink));
    memset(ylink, -1, sizeof(ylink));
}

void read_graph(int u, int v)
{
    edge[cnt].v = v;
    edge[cnt].next = first[u], first[u] = cnt++;
}

int bfs()
{
    queue<int> q;
    dis = INF;
    memset(dx, -1, sizeof(dx));
    memset(dy, -1, sizeof(dy));
    for (int i = 0; i < nx; i++)
    {
        if (xlink[i] == -1)
        {
            q.push(i);
            dx[i] = 0;
        }
    }
    while (!q.empty())
    {
        int u = q.front();
        q.pop();
        if (dx[u] > dis) break;
        for (int e = first[u]; e != -1; e = edge[e].next)
        {
            int v = edge[e].v;
```

```
58              if (dy[v] == -1)
59              {
60                  dy[v] = dx[u] + 1;
61                  if (ylink[v] == -1) dis = dy[v];
62                  else
63                  {
64                      dx[ylink[v]] = dy[v] + 1;
65                      q.push(ylink[v]);
66                  }
67              }
68          }
69      }
70      return dis != INF;
71  }
72
73  int find(int u)
74  {
75      for (int e = first[u]; e != -1; e = edge[e].next)
76      {
77          int v = edge[e].v;
78          if (!vis[v] && dy[v] == dx[u] + 1)
79          {
80              vis[v] = 1;
81              if (ylink[v] != -1 && dy[v] == dis) continue;
82              if (ylink[v] == -1 || find(ylink[v]))
83              {
84                  xlink[u] = v, ylink[v] = u;
85                  return 1;
86              }
87          }
88      }
89      return 0;
90  }
91
92  int MaxMatch()
93  {
94      int ans = 0;
95      while (bfs())
96      {
97          memset(vis, 0, sizeof(vis));
98          for (int i = 0; i < nx; i++)
99              if (xlink[i] == -1)
100                 ans += find(i);
101     }
102     return ans;
103 }
```

2.4.3 Multiple Matching

```
1  const int maxn = 1e2 + 5;//左边最大点数
2  const int maxm = 1e2 + 5;//右边最大点数
3  int graph[maxn][maxm], vis[maxm];//图G和增广路访问标记
4  int match[maxm][maxn];//左边元素与右边元素第n次匹配
5  int nx, ny, m;//左边点数, 右边点数,边数
6  int vol[maxm];//右边点多重匹配可容纳值
7  int cnt[maxm];//右边点已匹配值
8
9  bool find_path(int u)//找增广路
```

```
{
    for (int i = 0; i < ny; i++)//注意，这里节点是从0开始编号，题目有时是从1开始编号
    {
        if (graph[u][i] && !vis[i])//不在增广路
        {
            vis[i] = 1;//放进增广路
            if (cnt[i] < vol[i])//如果当前已匹配数量小于可容纳量，则直接匹配
            {
                match[i][cnt[i]++] = u;
                return true;
            }
            for (int j = 0; j < cnt[i]; j++)
            {
                if (find_path(match[i][j]))//如果先前已匹配右边的点能另外找到增广路，则此点仍可匹配
                {
                    match[i][j] = u;
                    return true;
                }
            }
        }
    }
    return false;
}

int max_match()//计算多重匹配的最大匹配数
{
    int res = 0;
    memset(match, -1, sizeof(match));
    memset(cnt, 0, sizeof(cnt));
    for (int i = 0; i < nx; i++)
    {
        memset(vis, 0, sizeof(vis));
        if (find_path(i)) res++;
    }
    return res;
}

bool all_match()//判断左边的点是否都与右边的点匹配了
{
    memset(cnt, 0, sizeof(cnt));
    for (int i = 0; i < nx; i++)
    {
        memset(vis, 0, sizeof(vis));
        if (!find_path(i)) return false;
    }
    return true;
}
```

2.4.4 Kuhn-Munkres Algorithm

```
const int maxn=1000+10;
const int inf=0x3f3f3f3f;

int n;
int lx[maxn],ly[maxn],edge[maxn][maxn];
int match[maxn],delta;
bool vx[maxn],vy[maxn];

```

```
bool dfs(int x) //DFS增广，寻找相等子图的完备匹配
{
    vx[x]=true;
    for(int y=1;y<=n;y++)
    {
        if(!vy[y])
        {
            int tmp=lx[x]+ly[y]-edge[x][y];
            if(!tmp)    //edge(x,y)为可行边
            {
                vy[y]=true;
                if(!match[y]||dfs(match[y]))
                {
                    match[y]=x;
                    return true;
                }
            }
            else delta=min(delta,tmp);
        }
    }
    return false;
}

void KM()
{
    for(int i=1;i<=n;i++)   //初始化可行顶标的值
    {
        lx[i]=-inf;
        ly[i]=0;
        for(int j=1;j<=n;j++)
            lx[i]=max(lx[i],edge[i][j]);
    }
    memset(match,0,sizeof(match));
    for(int x=1;x<=n;x++)
    {
        for(;;)
        {
            delta=inf;
            memset(vx,0,sizeof(vx));
            memset(vy,0,sizeof(vy));
            if(dfs(x)) break;
            for(int i=1;i<=n;i++)   //修改顶标
            {
                if(vx[i]) lx[i]-=delta;
                if(vy[i]) ly[i]+=delta;
            }
        }
    }
}
```

2.4.5 Edmonds's Matching Algorithm

```
//一般图匹配，带花树算法
const int maxn = 1000 + 10;

vector<int> edge[maxn];
queue<int> que;

```

```
7  int n, pre[maxn], type[maxn], link[maxn], nex[maxn], vis[maxn];
8
9  void add(int u, int v)
10 {
11     edge[u].push_back(v);
12     edge[v].push_back(u);
13 }
14
15 int Find(int x)
16 {
17     return x == pre[x] ? x : pre[x] = Find(pre[x]);
18 }
19
20 void combine(int x, int lca)    //如果找到奇环，对当前点x和找到的
21 {
22     while (x != lca)
23     {
24         int u = link[x], v = nex[u];
25         if (Find(v) != lca) nex[v] = u;
26         if (type[u] == 1) type[u] = 2, que.push(u);
27         pre[Find(x)] = Find(u);
28         pre[Find(u)] = Find(v);
29         x = v;
30     }
31 }
32
33 void contrack(int x, int y)
34 {
35     int lca = x;
36     memset(vis, 0, sizeof(vis));
37     for (int i = x; i; i = nex[link[i]])
38     {
39         i = Find(i);
40         vis[i] = 1;
41     }
42     for (int i = y; i; i = nex[link[i]])
43     {
44         i = Find(i);
45         if (vis[i])
46         {
47             lca = i;
48             break;
49         }
50     }
51     if (lca != Find(x)) nex[x] = y;
52     if (lca != Find(y)) nex[y] = x;
53     combine(x, lca);
54     combine(y, lca);
55 }
56
57 void bfs(int s)
58 {
59     memset(type, 0, sizeof(type));
60     memset(nex, 0, sizeof(nex));
61     for (int i = 1; i <= n; i++) pre[i] = i;
62     while (!que.empty()) que.pop();
63     que.push(s);
64     type[s] = 2;
65     while (!que.empty())
```

```
66      {
67          int x = que.front();
68          que.pop();
69          for (int i = 0; i < edge[x].size(); i++)
70          {
71              int y = edge[x][i];
72              if (Find(x) == Find(y) || link[x] == y || type[y] == 1) continue;
73              if (type[y] == 2) contrack(x, y);
74              else if (link[y])
75              {
76                  nex[y] = x;
77                  type[y] = 1;
78                  type[link[y]] = 2;
79                  que.push(link[y]);
80              } else
81              {
82                  nex[y] = x;
83                  int pos = y, u = nex[pos], v = link[u];
84                  while (pos)
85                  {
86                      link[pos] = u;
87                      link[u] = pos;
88                      pos = v;
89                      u = nex[pos];
90                      v = link[u];
91                  }
92                  return;
93              }
94          }
95      }
96  }
97
98  int maxmatch()
99  {
100     for (int i = 1; i <= n; i++) if (!link[i]) bfs(i);
101     int ans = 0;
102     for (int i = 1; i <= n; i++) if (link[i]) ans++;
103     return ans / 2;
104 }
105
106 void init()
107 {
108     for (int i = 1; i <= n; i++) edge[i].clear();
109     memset(link, 0, sizeof(link));
110 }
```

2.5 Network Flows

2.5.1 Dinic

```
1  const int MAX_V = 1000 + 10;
2  const int INF = 0x3f3f3f3f;
3
4  //用于表示边的结构体（终点，流量，反向边）
5  struct edge{int to, cap, rev;};
6
7  vector<edge> G[MAX_V];  //图的邻接表表示
8  int level[MAX_V];   //顶点到源点的距离标号
9  int iter[MAX_V];    //当前弧
```

```
10
11  void add(int from, int to, int cap)
12  {
13      G[from].push_back((edge){to, cap, G[to].size()});
14      G[to].push_back((edge){from, 0, G[from].size() - 1});
15  }
16
17  //计算从源点出发的距离标号
18  void bfs(int s)
19  {
20      memset(level, -1, sizeof(level));
21      queue<int> que;
22      level[s] = 0;
23      que.push(s);
24      while(!que.empty())
25      {
26          int v = que.front(); que.pop();
27          for(int i = 0; i < G[v].size(); i++)
28          {
29              edge &e = G[v][i];
30              if(e.cap > 0 && level[e.to] < 0)
31              {
32                  level[e.to] = level[v] + 1;
33                  que.push(e.to);
34              }
35          }
36      }
37  }
38
39  //通过DFS寻找增广路
40  int dfs(int v, int t, int f)
41  {
42      if(v == t) return f;
43      for(int &i = iter[v]; i<G[v].size(); i++)
44      {
45          edge &e = G[v][i];
46          if(e.cap > 0 && level[v] < level[e.to])
47          {
48              int d = dfs(e.to, t, min(f, e.cap));
49              if(d > 0)
50              {
51                  e.cap -= d;
52                  G[e.to][e.rev].cap += d;
53                  return d;
54              }
55          }
56      }
57      return 0;
58  }
59
60  //求解从s到t的最大流
61  int max_flow(int s, int t)
62  {
63      int flow = 0;
64      for(;;)
65      {
66          bfs(s);
67          if(level[t] < 0) return flow;
68          memset(iter, 0, sizeof(iter));
```

```
69          int f;
70          while((f = dfs(s,t,INF)) > 0) flow += f;
71      }
72  }
```

2.5.2 ISAP(undo)

2.5.3 MCMF

```
1   const int maxn = 10000 + 10;
2   const int inf = 0x3f3f3f3f;
3
4   struct Edge { int from, to, cap, flow, cost; };
5
6   struct MCMF
7   {
8       int n, m;
9       vector<Edge> edges;
10      vector<int> G[maxn];
11      bool inq[maxn];
12      int dis[maxn], path[maxn], a[maxn];
13
14      void init(int n)
15      {
16          this->n = n;
17          for(int i = 0;i <= n;i ++)
18              G[i].clear();
19          edges.clear();
20      }
21
22      void addEdge(int from, int to, int cap, int cost)
23      {
24          edges.push_back(Edge{from, to, cap, 0, cost});
25          edges.push_back(Edge{to, from, 0, 0, -cost});
26          m = edges.size();
27          G[from].push_back(m - 2);
28          G[to].push_back(m - 1);
29      }
30
31      bool Bellman_Ford(int s, int t, int& flow, int& cost)
32      {
33          for(int i = 0; i<= n; i++) dis[i] = inf;
34          memset(inq, 0, sizeof inq);
35          dis[s]=0, inq[s]=true, path[s]=0, a[s]=inf;
36          queue<int> Q;
37          Q.push(s);
38          while(!Q.empty())
39          {
40              int u = Q.front(); Q.pop();
41              inq[u] = false;
42              for(int i = 0; i < G[u].size(); i++)
43              {
44                  Edge& e = edges[G[u][i]];
45                  if(e.cap > e.flow && dis[e.to] > dis[u] + e.cost)
46                  {
47                      dis[e.to] = dis[u] + e.cost;
48                      path[e.to] = G[u][i];
```

```
                    a[e.to] = min(a[u], e.cap - e.flow);
                    if(!inq[e.to])
                    {
                        Q.push(e.to);
                        inq[e.to] = true;
                    }
                }
            }
        }
        if(dis[t] == inf) return false;
        flow += a[t];
        cost += dis[t] * a[t];
        for(int u = t; u != s; u = edges[path[u]].from)
        {
            edges[path[u]].flow += a[t];
            edges[path[u] ^ 1].flow -= a[t];
        }
        return true;
    }

    int mincostMaxFlow(int s, int t)
    {
        int flow = 0, cost = 0;
        while(Bellman_Ford(s, t, flow, cost));
        return cost;
    }
};
```

2.6 Directed-MST

2.6.1 Directed-MST

```
const int INF = 0x3f3f3f3f;
const int maxn = 10000;
const int maxm = 10000;

struct Edge{int u,v,cost; } edge[maxm];

int pre[maxn], id[maxn], vis[maxn], in[maxn];

int zhuliu(int root, int n, int m)
{
    int res=0, u, v;
    for(;;)
    {
        for(int i=0; i<n; i++) in[i] = INF;
        for(int i=0; i<m; i++) if(edge[i].u != edge[i].v && edge[i].cost < in[edge[i].v])
        {
            pre[edge[i].v] = edge[i].u;
            in[edge[i].v] = edge[i].cost;
        }
        for(int i=0; i<n; i++) if(i != root && in[i] ==INF) return -1;
        int tn=0;
        memset(id, 0xff, sizeof id);
        memset(vis, 0xff, sizeof vis);
        in[root] = 0;
        for(int i=0; i<n;i++)
        {
            res += in[i];
```

```
28              v = i;
29              while( vis[v] != i && id[v] == -1 && v!= root) vis[v] = i, v = pre[v];
30              if(v != root && id[v] == -1)
31              {
32                  for(int u = pre[v]; u != v; u = pre[u]) id[u] = tn;
33                  id[v] = tn++;
34              }
35          }
36          if(tn == 0) break;
37          for(int i=0; i<n; i++) if(id[i] == -1) id[i] = tn++;
38          for(int i=0; i<m; )
39          {
40              v = edge[i].v;
41              edge[i].u = id[edge[i].u];
42              edge[i].v = id[edge[i].v];
43              if(edge[i].u != edge[i].v) edge[i++].cost -= in[v];
44              else swap(edge[i], edge[--m]);
45          }
46          n = tn;
47          root = id[root];
48      }
49      return res;
50  }
```

2.7 Toposort

2.7.1 Toposort

```
1   const int maxn = 1e5 + 10;
2
3   vector<int> edge[maxn];
4   int indegree[maxn];
5
6   void add(int u, int v)
7   {
8       edge[u].push_back(v);
9       indegree[v]++;
10  }
11
12  void Toposort(int n)
13  {
14      queue<int> que;
15      for (int i = 1; i <= n; i++)
16          if (!indegree[i]) que.push(i);    //将图中没有前驱，即入度为0的点加入队列
17      while (!que.empty())
18      {
19          int u = que.front();
20          que.pop();
21          indegree[u] = -1;    //从图中删去此顶点
22          for (int i = 0; i < edge[u].size(); i++)
23          {
24              int v = edge[u][i];
25              indegree[v]--;    //删去图中以u为尾的弧
26              if (!indegree[v]) que.push(v);    //将新增的当前入度为0的点压入队列中
27          }
28      }
29  }
```

2.8 2-SAT

2.8.1 2-SAT

```
const int maxn = 2e6 + 10;

int n, m, a, va, b, vb;
int low[maxn], dfn[maxn], color[maxn], cnt, scc_cnt;
bool instack[maxn];

vector<int> g[maxn];

void Tarjan(int u)
{
    low[u] = dfn[u] = ++cnt;
    st.push(u);
    instack[u] = true;
    for(const auto &v : g[u])
    {
        if(!dfn[v]) Tarjan(v), low[u] = min(low[u], low[v]);
        else if(instack[v]) low[u] = min(low[u], dfn[v]);
    }
    if(low[u] == dfn[u])
    {
        ++scc_cnt;
        do {
            color[u] = scc_cnt;
            u = st.top(); st.pop();
            instack[u] = false;
        } while(low[u] != dfn[u]);
    }
}

void 2_SAT()
{
    scanf("%d%d", &n, &m);
    for(int i = 0; i < m; i ++)
    {
        scanf("%d%d%d%d", &a, &va, &b, &vb);
        g[ a + n * (va & 1) ].push_back(b + n * (vb ^ 1));
        g[ b + n * (vb & 1) ].push_back(a + n * (va ^ 1));
    }
    cnt = scc_cnt = 0;
    for(int i = 1; i <= (n << 1); i ++) if(!dfn[i]) Tarjan(i);
}
```

2.9 System of Difference Constraints

2.9.1 System of Difference Constraints

```
const int maxn = 1000 + 10;
const int inf = 0x3f3f3f3f;

struct Edge
{
    int nex, to, w;
} edge[10 * maxn];

int head[maxn], cnt, dis[maxn], n;
```

```
10  bool vis[maxn];
11
12  void init()
13  {
14      cnt = 0;
15      memset(head, 0xff, sizeof head);
16  }
17
18  void add(int u, int v, int w)
19  {
20      edge[cnt].nex = head[u];
21      edge[cnt].to = v;
22      edge[cnt].w = w;
23      head[u] = ++cnt;
24  }
25
26  void spfa(int u)
27  {
28      int u, v, w;
29      for (int i = 1; i <= n; i++) dis[i] = inf, vis[i] = false;
30      dis[u] = 0;
31      queue<int> que;
32      que.push(u);
33      vis[u] = true;
34      while (!que.empty())
35      {
36          u = que.front();
37          que.pop();
38          vis[u] = false;
39          for (int i = head[u]; ~i; i = edge[i].nex)
40          {
41              v = edge[i].v, w = edge[i].w;
42              if (dis[u] + w < dis[v])
43              {
44                  dis[v] = dis[u] + w;
45                  if (!vis[v])
46                  {
47                      que.push(v);
48                      vis[v] = true;
49                  }
50              }
51          }
52      }
53  }
```

3 DataStructrue

3.1 HLD

3.1.1 HLD

```cpp
#include <bits/stdc++.h>
#define ll long long
using namespace std;
/*
node 计算点权， path 下放后计算边权， edge 根据边的编号计算边权
work 中没有build需手动写
sz[]数组，以x为根的子树节点个数
top[]数组，当前节点的所在链的顶端节点
son[]数组，重儿子
deep[]数组，当前节点的深度
fa[]数组，当前节点的父亲
idx[]数组，树中每个节点剖分后的新编号
rnk[]数组，idx的逆，表示线段上中当前位置表示哪个节点
*/

const int maxn = 1e5+5;

int sz[maxn], top[maxn], son[maxn], deep[maxn], fa[maxn], idx[maxn], rnk[maxn];
int tot;
int n, le, re;
ll k;

struct HLD {
#define type int

    struct edge {
        int a, b;
        type v;

        edge(int _a, int _b, type _v = 0) : a(_a), b(_b), v(_v) {}
    };

    struct node {
        int to;
        type w;

        node() {}

        node(int _to, type _w) : to(_to), w(_w) {}
    };

    vector<int> mp[maxn];
    vector<edge> e;

    void init(int _n) {
        n = _n;
        for (int i = 0; i <= n; i++) mp[i].clear();
        e.clear();
        e.push_back(edge(0, 0));
    }

    void add_edge(int a, int b, type v = 0) {
//        e.push_back(edge(a,b,v));
```

```
54          mp[a].push_back(b);
55          mp[b].push_back(a);
56      }
57
58      void dfs1(int x, int pre, int h) {
59          int i, to;
60          deep[x] = h;
61          fa[x] = pre;
62          sz[x] = 1;
63          for (i = 0; i < (int) (mp[x].size()); i++) {
64              to = mp[x][i];
65              if (to == pre) continue;
66              dfs1(to, x, h + 1);
67              sz[x] += sz[to];
68              if (son[x] == -1 || sz[to] > sz[son[x]]) son[x] = to;
69          }
70      }
71
72      void dfs2(int x, int tp) {
73          int i, to;
74          top[x] = tp;
75          idx[x] = ++tot;
76          rnk[idx[x]] = x;
77          if (son[x] == -1) return;
78          dfs2(son[x], tp);
79          for (i = 0; i < (int) (mp[x].size()); i++) {
80              to = mp[x][i];
81              if (to != son[x] && to != fa[x]) dfs2(to, to);
82          }
83      }
84
85      void work(int _rt = 1) {
86          memset(son, -1, sizeof son);
87          tot = 0;
88          dfs1(_rt, 0, 0);
89          dfs2(_rt, _rt);
90      }
91
92      int LCA(int x, int y) {
93          while (top[x] != top[y]) {
94              if (deep[top[x]] < deep[top[y]]) swap(x, y);
95              x = fa[top[x]];
96          }
97          if (deep[x] > deep[y]) swap(x, y);
98          return x;
99      }
100
101     void modify_node(int x, int y, type val) {
102         while (top[x] != top[y]) {
103             if (deep[top[x]] < deep[top[y]]) swap(x, y);
104             le = idx[top[x]], re = idx[x];
105             k = val;
106             update(1, 1, n);
107             x = fa[top[x]];
108         }
109         if (deep[x] > deep[y]) swap(x, y);
110         le = idx[x], re = idx[y];
111         k = val;
112         update(1, 1, n);
```

```
113     }
114
115     type query_node(int x, int y) {
116         type res = 0;
117         while (top[x] != top[y]) {
118             if (deep[top[x]] < deep[top[y]]) swap(x, y);
119             le = idx[top[x]], re = idx[x];
120             res += query(1, 1, n);
121             x = fa[top[x]];
122         }
123         if (deep[x] > deep[y]) swap(x, y);
124         le = idx[x], re = idx[y];
125         res += query(1, 1, n);
126         return res;
127     }
128
129     //path
130 //    void init_path()
131 //    {
132 //        v[idx[rt]]=0;
133 //        for(int i=1;i<n;i++)
134 //        {
135 //            if(deep[e[i].a]<deep[e[i].b]) swap(e[i].a,e[i].b);
136 //            a[idx[e[i].a]]=e[i].v;
137 //        }
138 //        build(n);
139 //    }
140     void modify_edge(int id, type val) {
141         if (deep[e[id].a] > deep[e[id].b]) {
142             le = idx[e[id].a], re = idx[e[id].a];
143             k = val;
144             update(1, 1, n);
145         } else {
146             le = idx[e[id].b], re = idx[e[id].b];
147             k = val;
148             update(1, 1, n);
149         }
150     }
151
152     void modify_path(int x, int y, type val) {
153         while (top[x] != top[y]) {
154             if (deep[top[x]] < deep[top[y]]) swap(x, y);
155             le = idx[top[x]], re = idx[x];
156             k = val;
157             update(1, 1, n);
158             x = fa[top[x]];
159         }
160         if (deep[x] > deep[y]) swap(x, y);
161         if (x != y) {
162             le = idx[x] + 1, re = idx[y];
163             k = val;
164             update(1, 1, n);
165         }
166     }
167
168     type query_path(int x, int y) {
169         type res = 0;
170         while (top[x] != top[y]) {
171             if (deep[top[x]] < deep[top[y]]) swap(x, y);
```

```
            le = idx[top[x]], re = idx[x];
            res += query(1, 1, n);
            x = fa[top[x]];
        }
        if (deep[x] > deep[y]) swap(x, y);
        if (x != y) {
            le = idx[x] + 1, re = idx[y];
            res += query(1, 1, n);
        }
        return res;
    }

#undef type
} hld;
```

3.2 RMQ

3.2.1 RMQ

```
//一维RMQ
//MAX=1e6时 第二维开22 内存(int型)占10w
int v[MAX],maxx[MAX][22],minn[MAX][22];
void RMQ(int n)
{
    int i,j;
    for(i=1;i<=n;i++)
    {
        maxx[i][0]=minn[i][0]=v[i];//下标rmq 初始化赋值成i
        for(j=1;1<<(j-1)<=n;j++)
        {
            maxx[i][j]=0;
            minn[i][j]=INF;
        }
    }
    for(j=1;1<<(j-1)<=n;j++)
    {
        for(i=1;i+(1<<j)-1<=n;i++)
        {
            int t=1<<(j-1);
            maxx[i][j]=max(maxx[i][j-1],maxx[i+t][j-1]);
            minn[i][j]=min(minn[i][j-1],minn[i+t][j-1]);
        }
    }
}
int query(int l,int r)
{
    int j=(int)(log10(r-l+1)/log10(2))+1;
    int i=r-(1<<(j-1))+1;
    return max(maxx[l][j-1],maxx[i][j-1]);
//  return min(minn[l][j-1],minn[i][j-1]);
}
```

3.2.2 RMQbyIndex

```
//下标RMQ
int v[MAX],maxx[MAX][22],minn[MAX][22];
int pmax(int a,int b){return v[a]>v[b]?a:b;}
int pmin(int a,int b){return v[a]<v[b]?a:b;}
```

```
5  void RMQ(int n)
6  {
7      int i,j;
8      for(i=1;i<=n;i++)
9      {
10         maxx[i][0]=minn[i][0]=i;
11     }
12     for(j=1;1<<(j-1)<=n;j++)
13     {
14         for(i=1;i+(1<<j)-1<=n;i++)
15         {
16             int t=1<<(j-1);
17             maxx[i][j]=pmax(maxx[i][j-1],maxx[i+t][j-1]);
18             minn[i][j]=pmin(minn[i][j-1],minn[i+t][j-1]);
19         }
20     }
21 }
22 int query(int l,int r)
23 {
24     int j=(int)(log10(r-l+1)/log10(2))+1;
25     int i=r-(1<<(j-1))+1;
26     return pmax(maxx[l][j-1],maxx[i][j-1]);
27 //  return pmin(minn[l][j-1],minn[i][j-1]);
28 }
```

3.2.3 RMQinNM

```
1  //二维RMQ
2  int v[302][302];
3  int maxx[302][302][9][9],minn[302][302][9][9];
4  void RMQ(int n,int m)
5  {
6      int i,j,ii,jj;
7      for(i=1;i<=n;i++)
8      {
9          for(j=1;j<=m;j++)
10         {
11             maxx[i][j][0][0]=minn[i][j][0][0]=v[i][j];
12         }
13     }
14     for(ii=0;(1<<ii)<=n;ii++)
15     {
16         for(jj=0;(1<<jj)<=m;jj++)
17         {
18             if(ii+jj)
19             {
20                 for(i=1;i+(1<<ii)-1<=n;i++)
21                 {
22                     for(j=1;j+(1<<jj)-1<=m;j++)
23                     {
24                         if(ii)
25                         {
26                             minn[i][j][ii][jj]=min(minn[i][j][ii-1][jj],minn[i+(1<<(ii-1))][j][ii
       -1][jj]);
27                             maxx[i][j][ii][jj]=max(maxx[i][j][ii-1][jj],maxx[i+(1<<(ii-1))][j][ii
       -1][jj]);
28                         }
29                         else
```

```
                        {
                            minn[i][j][ii][jj]=min(minn[i][j][ii][jj-1],minn[i][j+(1<<(jj-1))][ii][
    jj-1]);
                            maxx[i][j][ii][jj]=max(maxx[i][j][ii][jj-1],maxx[i][j+(1<<(jj-1))][ii][
    jj-1]);
                        }
                    }
                }
            }
        }
    }
}
int query(int x1,int y1,int x2,int y2)
{
    int k1=0;
    while((1<<(k1+1))<=x2-x1+1) k1++;
    int k2=0;
    while((1<<(k2+1))<=y2-y1+1) k2++;
    x2=x2-(1<<k1)+1;
    y2=y2-(1<<k2)+1;
    return max(max(maxx[x1][y1][k1][k2],maxx[x1][y2][k1][k2]),max(maxx[x2][y1][k1][k2],maxx[x2][y2
    ][k1][k2]))
//  return min(min(minn[x1][y1][k1][k2],minn[x1][y2][k1][k2]),min(minn[x2][y1][k1][k2],minn[x2][y2
    ][k1][k2]));
}
```

3.3 MO

3.3.1 MO

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
const int maxn = 200005;

struct MO {
    int l, r, id;
}q[maxn];

int n, m, col[maxn], block, belong[maxn];
ll vis[maxn * 10], ans;
ll res[maxn];
bool cmp(const MO& a, const MO& b) { return belong[a.l] == belong[b.l] ? a.r < b.r : a.l < b.l; }
void add(ll x) {
    vis[x] ++;
    ans += x * (vis[x] * vis[x] - (vis[x] - 1) * (vis[x] - 1));
}

void del(ll x) {
    vis[x] --;
    ans -= x * ((vis[x] + 1) * (vis[x] + 1) - vis[x] * vis[x]);
}

int main() {
    scanf("%d%d", &n, &m);
    block = sqrt(n);
    for (int i = 1; i <= n; ++i) {
        scanf("%d", &col[i]);
        belong[i] = i / block + 1;
```

```
    }
    for (int i = 1; i <= m; ++i) {
        scanf("%d%d", &q[i].l, &q[i].r);
        q[i].id = i;
    }
    sort(q + 1, q + 1 + m, cmp);
    int l = 1, r = 0;
    for (int i = 1; i <= m; ++i) {
        while(r < q[i].r) add(col[++r]);
        while(r > q[i].r) del(col[r--]);
        while(l < q[i].l) del(col[l++]);
        while(l > q[i].l) add(col[--l]);
        res[q[i].id] = ans;
    }
    for (int i = 1; i <= m; ++i) printf("%lld\n", res[i]);
    return 0;
}
```

3.3.2 MObyModify

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
const int maxn = 50005;

struct MO {
    int l, r, id, oppre;
}q[maxn];

int n, m, col[maxn], block, belong[maxn], colpre[maxn];
int changepos[maxn], changepre[maxn], changenow[maxn];
int vis[maxn * 20];
int ans;
int res[maxn];
bool cmp(const MO& a, const MO& b) {
    if (belong[a.l] != belong[b.l]) return a.l < b.l;
    if (belong[a.r] != belong[b.r]) return a.r < b.r;
    return a.oppre < b.oppre;
}
void add(int x) {}

void del(int x) {}

void unmodify(int pos, int now) {
    if (q[pos].l <= changepos[now] && changepos[now] <= q[pos].r) {
        del(changenow[now]);
        add(changepre[now]);
    }
    col[changepos[now]] = changepre[now];
}

void modify(int pos, int now) {
    if (q[pos].l <= changepos[now] && changepos[now] <= q[pos].r) {
        del(changepre[now]);
        add(changenow[now]);
    }
    col[changepos[now]] = changenow[now];
}
```

```

int main() {
    scanf("%d%d", &n, &m);
    block = pow(n, 0.66666);
    for (int i = 1; i <= n; ++i) {
        scanf("%d", &col[i]);
        colpre[i] = col[i];
        belong[i] = i / block + 1;
    }
    char s[2];
    int t = 0, t2 = 0;
    for (int i = 1; i <= m; ++i) {
        scanf("%s", s);
        if (s[0] == 'Q') {
            ++t;
            scanf("%d%d", &q[t].l, &q[t].r);
            q[t].oppre = t2;
            q[t].id = t;
        } else {
            ++t2;
            scanf("%d%d", &changepos[t2], &changenow[t2]);
            changepre[t2] = colpre[changepos[t2]];
            colpre[changepos[t2]] = changenow[t2];
        }
    }
    sort(q + 1, q + 1 + t, cmp);
    int l = 1, r = 0, now = 0;
    for (int i = 1; i <= t; ++i) {
        while(r < q[i].r) add(col[++r]);
        while(r > q[i].r) del(col[r--]);
        while(l < q[i].l) del(col[l++]);
        while(l > q[i].l) add(col[--l]);
        while (now < q[i].oppre) modify(i, ++now);
        while (now > q[i].oppre) unmodify(i, now--);
        res[q[i].id] = ans;
    }
    for (int i = 1; i <= t; ++i) printf("%d\n", res[i]);
    return 0;
}
```

3.4 VirtualTree

3.4.1 VirtualTree

```
const int maxn = "Edit";
vector<int> vtree[maxn];
void build(vector<int>& vec)
{
    sort(vec.begin(), vec.end(), [&](int x, int y) { return dfn[x] < dfn[y]; });
    static int s[maxn];
    int top = 0;
    s[top] = 0;
    vtree[0].clear();
    for (auto& u : vec)
    {
        int vlca = lca(u, s[top]);
        vtree[u].clear();
        if (vlca == s[top])
            s[++top] = u;
```

```
        else
        {
            while (top && dep[s[top - 1]] >= dep[vlca])
            {
                vtree[s[top - 1]].push_back(s[top]);
                top--;
            }
            if (s[top] != vlca)
            {
                vtree[vlca].clear();
                vtree[vlca].push_back(s[top--]);
                s[++top] = vlca;
            }
            s[++top] = u;
        }
    }
    for (int i = 0; i < top; ++i) vtree[s[i]].push_back(s[i + 1]);
}
```

3.5 PersistentDS

3.5.1 主席树区间 k 大

```
/*************************************************************************
    > File Name: a.cpp
    > Author: badcw
    > Mail: 952223482@qq.com
    > Created Time: 2018年07月21日 星期六 08时47分54秒
 ************************************************************************/

#include <bits/stdc++.h>
#define ll long long
using namespace std;

const int maxn = 100005;
int n, m;
int a[maxn];
int root[maxn];
int cnt = 0;
vector<int> b;
struct node {
    int l, r, val;
}p[maxn * 40];

void update(int l, int r, int pre, int &now, int pos) {
    now = ++cnt;
    p[now] = p[pre];
    p[now].val++;
    if (l == r) {
        return;
    }
    int mid = l + r >> 1;
    if (pos <= mid) update(l, mid, p[pre].l, p[now].l, pos);
    else update(mid + 1, r, p[pre].r, p[now].r, pos);
}

int query(int l, int r, int x, int y, int k) {
    if (l == r) return b[l - 1];
    int mid = l + r >> 1;
```

```
37      int temp = p[p[y].l].val - p[p[x].l].val;
38      if (k <= temp) return query(l, mid, p[x].l, p[y].l, k);
39      return query(mid + 1, r, p[x].r, p[y].r, k - temp);
40  }
41
42  int main(int argc,char *argv[])
43  {
44      while (scanf("%d%d", &n, &m) != EOF) {
45          b.clear();
46          cnt = 0;
47          for (int i = 1; i <= n; ++i) scanf("%d", &a[i]), b.push_back(a[i]);
48          sort(b.begin(), b.end());
49          b.erase(unique(b.begin(), b.end()), b.end());
50          for (int i = 1; i <= n; ++i) {
51              update(1, b.size(), root[i - 1], root[i], lower_bound(b.begin(), b.end(), a[i]) - b.
        begin() + 1);
52          }
53          int L, R, k;
54          while (m--) {
55              scanf("%d%d%d", &L, &R, &k);
56              printf("%d\n", query(1, b.size(), root[L - 1], root[R], k));
57          }
58      }
59      return 0;
60  }
```

3.6 Others

3.6.1 BITinNM

```
1   struct Fenwick_Tree
2   {
3       #define type int
4       type bit[MAX][MAX];
5       int n,m;
6       void init(int _n,int _m){n=_n;m=_m;mem(bit,0);}
7       int lowbit(int x){return x&(-x);}
8       void update(int x,int y,type v)
9       {
10          int i,j;
11          for(i=x;i<=n;i+=lowbit(i))
12          {
13              for(j=y;j<=m;j+=lowbit(j))
14              {
15                  bit[i][j]+=v;
16              }
17          }
18      }
19      type get(int x,int y)
20      {
21          type i,j,res=0;
22          for(i=x;i>0;i-=lowbit(i))
23          {
24              for(j=y;j>0;j-=lowbit(j))
25              {
26                  res+=bit[i][j];
27              }
28          }
29          return res;
```

```
30      }
31      type query(int x1,int x2,int y1,int y2)
32      {
33          x1--;
34          y1--;
35          return get(x2,y2)-get(x1,y2)-get(x2,y1)+get(x1,y1);
36      }
37      #undef type
38  }tr;
```

4 Geometry

4.1 Class

4.1.1 geo

```
#define mp make_pair
#define fi first
#define se second
#define pb push_back
typedef double db;
const db eps=1e-6;
const db pi=acos(-1);
int sign(db k){
    if (k>eps) return 1; else if (k<-eps) return -1; return 0;
}
int cmp(db k1,db k2){return sign(k1-k2);}
int inmid(db k1,db k2,db k3){return sign(k1-k3)*sign(k2-k3)<=0;}// k3 在 [k1,k2] 内
struct point{
    db x,y;
    point operator + (const point &k1) const{return (point){k1.x+x,k1.y+y};}
    point operator - (const point &k1) const{return (point){x-k1.x,y-k1.y};}
    point operator * (db k1) const{return (point){x*k1,y*k1};}
    point operator / (db k1) const{return (point){x/k1,y/k1};}
    int operator == (const point &k1) const{return cmp(x,k1.x)==0&&cmp(y,k1.y)==0;}
    // 逆时针旋转
    point turn(db k1){return (point){x*cos(k1)-y*sin(k1),x*sin(k1)+y*cos(k1)};}
    point turn90(){return (point){-y,x};}
    bool operator < (const point k1) const{
        int a=cmp(x,k1.x);
        if (a==-1) return 1; else if (a==1) return 0; else return cmp(y,k1.y)==-1;
    }
    db abs(){return sqrt(x*x+y*y);}
    db abs2(){return x*x+y*y;}
    db dis(point k1){return ((*this)-k1).abs();}
    point unit(){db w=abs(); return (point){x/w,y/w};}
    void scan(){double k1,k2; scanf("%lf%lf",&k1,&k2); x=k1; y=k2;}
    void print(){printf("%.11lf %.11lf\n",x,y);}
    db getw(){return atan2(y,x);}
    point getdel(){if (sign(x)==-1||(sign(x)==0&&sign(y)==-1)) return (*this)*(-1); else return (*
    this);}
    int getP() const{return sign(y)==1||(sign(y)==0&&sign(x)==-1);}
};
int inmid(point k1,point k2,point k3){return inmid(k1.x,k2.x,k3.x)&&inmid(k1.y,k2.y,k3.y);}
db cross(point k1,point k2){return k1.x*k2.y-k1.y*k2.x;}
db dot(point k1,point k2){return k1.x*k2.x+k1.y*k2.y;}
db rad(point k1,point k2){return atan2(cross(k1,k2),dot(k1,k2));}
// -pi -> pi
int compareangle (point k1,point k2){
    return k1.getP()<k2.getP()||(k1.getP()==k2.getP()&&sign(cross(k1,k2))>0);
}
point proj(point k1,point k2,point q){ // q 到直线 k1,k2 的投影
    point k=k2-k1; return k1+k*(dot(q-k1,k)/k.abs2());
}
point reflect(point k1,point k2,point q){return proj(k1,k2,q)*2-q;}
int clockwise(point k1,point k2,point k3){// k1 k2 k3 逆时针 1 顺时针 -1 否则 0
    return sign(cross(k2-k1,k3-k1));
}
int checkLL(point k1,point k2,point k3,point k4){// 求直线 (L) 线段 (S)k1,k2 和 k3,k4 的交点
```

```
53      return cmp(cross(k3-k1,k4-k1),cross(k3-k2,k4-k2))!=0;
54  }
55  point getLL(point k1,point k2,point k3,point k4){
56      db w1=cross(k1-k3,k4-k3),w2=cross(k4-k3,k2-k3); return (k1*w2+k2*w1)/(w1+w2);
57  }
58  int intersect(db l1,db r1,db l2,db r2){
59      if (l1>r1) swap(l1,r1); if (l2>r2) swap(l2,r2); return cmp(r1,l2)!=-1&&cmp(r2,l1)!=-1;
60  }
61  int checkSS(point k1,point k2,point k3,point k4){
62      return intersect(k1.x,k2.x,k3.x,k4.x)&&intersect(k1.y,k2.y,k3.y,k4.y)&&
63      sign(cross(k3-k1,k4-k1))*sign(cross(k3-k2,k4-k2))<=0&&
64      sign(cross(k1-k3,k2-k3))*sign(cross(k1-k4,k2-k4))<=0;
65  }
66  db disSP(point k1,point k2,point q){
67      point k3=proj(k1,k2,q);
68      if (inmid(k1,k2,k3)) return q.dis(k3); else return min(q.dis(k1),q.dis(k2));
69  }
70  db disSS(point k1,point k2,point k3,point k4){
71      if (checkSS(k1,k2,k3,k4)) return 0;
72      else return min(min(disSP(k1,k2,k3),disSP(k1,k2,k4)),min(disSP(k3,k4,k1),disSP(k3,k4,k2)));
73  }
74  int onS(point k1,point k2,point q){return inmid(k1,k2,q)&&sign(cross(k1-q,k2-k1))==0;}
75  struct circle{
76      point o; db r;
77      void scan(){o.scan(); scanf("%lf",&r);}
78      int inside(point k){return cmp(r,o.dis(k));}
79  };
80  struct line{
81      // p[0]->p[1]
82      point p[2];
83      line(point k1,point k2){p[0]=k1; p[1]=k2;}
84      point& operator [] (int k){return p[k];}
85      int include(point k){return sign(cross(p[1]-p[0],k-p[0]))>0;}
86      point dir(){return p[1]-p[0];}
87      line push(){ // 向外 ( 左手边 ) 平移 eps
88          const db eps = 1e-6;
89          point delta=(p[1]-p[0]).turn90().unit()*eps;
90          return {p[0]-delta,p[1]-delta};
91      }
92  };
93  point getLL(line k1,line k2){return getLL(k1[0],k1[1],k2[0],k2[1]);}
94  int parallel(line k1,line k2){return sign(cross(k1.dir(),k2.dir()))==0;}
95  int sameDir(line k1,line k2){return parallel(k1,k2)&&sign(dot(k1.dir(),k2.dir()))==1;}
96  int operator < (line k1,line k2){
97      if (sameDir(k1,k2)) return k2.include(k1[0]);
98      return compareangle(k1.dir(),k2.dir());
99  }
100 int checkpos(line k1,line k2,line k3){return k3.include(getLL(k1,k2));}
101 vector<line> getHL(vector<line> &L){ // 求半平面交 , 半平面是逆时针方向 , 输出按照逆时针
102     sort(L.begin(),L.end()); deque<line> q;
103     for (int i=0;i<(int)L.size();i++){
104         if (i&&sameDir(L[i],L[i-1])) continue;
105         while (q.size()>1&&!checkpos(q[q.size()-2],q[q.size()-1],L[i])) q.pop_back();
106         while (q.size()>1&&!checkpos(q[1],q[0],L[i])) q.pop_front();
107         q.push_back(L[i]);
108     }
109     while (q.size()>2&&!checkpos(q[q.size()-2],q[q.size()-1],q[0])) q.pop_back();
110     while (q.size()>2&&!checkpos(q[1],q[0],q[q.size()-1])) q.pop_front();
111     vector<line>ans; for (int i=0;i<q.size();i++) ans.push_back(q[i]);
```

```
    return ans;
}
db closepoint(vector<point>&A,int l,int r){ // 最近点对 , 先要按照 x 坐标排序
    if (r-l<=5){
        db ans=1e20;
        for (int i=l;i<=r;i++) for (int j=i+1;j<=r;j++) ans=min(ans,A[i].dis(A[j]));
        return ans;
    }
    int mid=l+r>>1; db ans=min(closepoint(A,l,mid),closepoint(A,mid+1,r));
    vector<point>B; for (int i=l;i<=r;i++) if (abs(A[i].x-A[mid].x)<=ans) B.push_back(A[i]);
    sort(B.begin(),B.end(),[](point k1,point k2){return k1.y<k2.y;});
    for (int i=0;i<B.size();i++) for (int j=i+1;j<B.size()&&B[j].y-B[i].y<ans;j++) ans=min(ans,B[i
    ].dis(B[j]));
    return ans;
}
int checkposCC(circle k1,circle k2){// 返回两个圆的公切线数量
    if (cmp(k1.r,k2.r)==-1) swap(k1,k2);
    db dis=k1.o.dis(k2.o);  int w1=cmp(dis,k1.r+k2.r),w2=cmp(dis,k1.r-k2.r);
    if (w1>0) return 4; else if (w1==0) return 3; else if (w2>0) return 2;
    else if (w2==0) return 1; else return 0;
}
vector<point> getCL(circle k1,point k2,point k3){ // 沿着 k2->k3 方向给出 , 相切给出两个
    point k=proj(k2,k3,k1.o); db d=k1.r*k1.r-(k-k1.o).abs2();
    if (sign(d)==-1) return {};
    point del=(k3-k2).unit()*sqrt(max((db)0.0,d)); return {k-del,k+del};
}
vector<point> getCC(circle k1,circle k2){// 沿圆 k1 逆时针给出 , 相切给出两个
    int pd=checkposCC(k1,k2); if (pd==0||pd==4) return {};
    db a=(k2.o-k1.o).abs2(),cosA=(k1.r*k1.r+a-k2.r*k2.r)/(2*k1.r*sqrt(max(a,(db)0.0)));
    db b=k1.r*cosA,c=sqrt(max((db)0.0,k1.r*k1.r-b*b));
    point k=(k2.o-k1.o).unit(),m=k1.o+k*b,del=k.turn90()*c;
    return {m-del,m+del};
}
vector<point> TangentCP(circle k1,point k2){// 沿圆 k1 逆时针给出
    db a=(k2-k1.o).abs(),b=k1.r*k1.r/a,c=sqrt(max((db)0.0,k1.r*k1.r-b*b));
    point k=(k2-k1.o).unit(),m=k1.o+k*b,del=k.turn90()*c;
    return {m-del,m+del};
}
vector<line> TangentoutCC(circle k1,circle k2){
    int pd=checkposCC(k1,k2); if (pd==0) return {};
    if (pd==1){point k=getCC(k1,k2)[0]; return {(line){k,k}};}
    if (cmp(k1.r,k2.r)==0){
        point del=(k2.o-k1.o).unit().turn90().getdel();
        return {(line){k1.o-del*k1.r,k2.o-del*k2.r},(line){k1.o+del*k1.r,k2.o+del*k2.r}};
    } else {
        point p=(k2.o*k1.r-k1.o*k2.r)/(k1.r-k2.r);
        vector<point>A=TangentCP(k1,p),B=TangentCP(k2,p);
        vector<line>ans; for (int i=0;i<A.size();i++) ans.push_back((line){A[i],B[i]});
        return ans;
    }
}
vector<line> TangentinCC(circle k1,circle k2){
    int pd=checkposCC(k1,k2); if (pd<=2) return {};
    if (pd==3){point k=getCC(k1,k2)[0]; return {(line){k,k}};}
    point p=(k2.o*k1.r+k1.o*k2.r)/(k1.r+k2.r);
    vector<point>A=TangentCP(k1,p),B=TangentCP(k2,p);
    vector<line>ans; for (int i=0;i<A.size();i++) ans.push_back((line){A[i],B[i]});
    return ans;
}
```

```
vector<line> TangentCC(circle k1,circle k2){
    int flag=0; if (k1.r<k2.r) swap(k1,k2),flag=1;
    vector<line>A=TangentoutCC(k1,k2),B=TangentinCC(k1,k2);
    for (line k:B) A.push_back(k);
    if (flag) for (line &k:A) swap(k[0],k[1]);
    return A;
}
db getarea(circle k1,point k2,point k3){
    // 圆 k1 与三角形 k2 k3 k1.o 的有向面积交
    point k=k1.o; k1.o=k1.o-k; k2=k2-k; k3=k3-k;
    int pd1=k1.inside(k2),pd2=k1.inside(k3);
    vector<point>A=getCL(k1,k2,k3);
    if (pd1>=0){
        if (pd2>=0) return cross(k2,k3)/2;
        return k1.r*k1.r*rad(A[1],k3)/2+cross(k2,A[1])/2;
    } else if (pd2>=0){
        return k1.r*k1.r*rad(k2,A[0])/2+cross(A[0],k3)/2;
    }else {
        int pd=cmp(k1.r,disSP(k2,k3,k1.o));
        if (pd<=0) return k1.r*k1.r*rad(k2,k3)/2;
        return cross(A[0],A[1])/2+k1.r*k1.r*(rad(k2,A[0])+rad(A[1],k3))/2;
    }
}
circle getcircle(point k1,point k2,point k3){
    db a1=k2.x-k1.x,b1=k2.y-k1.y,c1=(a1*a1+b1*b1)/2;
    db a2=k3.x-k1.x,b2=k3.y-k1.y,c2=(a2*a2+b2*b2)/2;
    db d=a1*b2-a2*b1;
    point o=(point){k1.x+(c1*b2-c2*b1)/d,k1.y+(a1*c2-a2*c1)/d};
    return (circle){o,k1.dis(o)};
}
circle getScircle(vector<point> A){
    random_shuffle(A.begin(),A.end());
    circle ans=(circle){A[0],0};
    for (int i=1;i<A.size();i++)
        if (ans.inside(A[i])==-1){
            ans=(circle){A[i],0};
            for (int j=0;j<i;j++)
                if (ans.inside(A[j])==-1){
                    ans.o=(A[i]+A[j])/2; ans.r=ans.o.dis(A[i]);
                    for (int k=0;k<j;k++)
                        if (ans.inside(A[k])==-1)
                            ans=getcircle(A[i],A[j],A[k]);
                }
        }
    return ans;
}
db area(vector<point> A){ // 多边形用 vector<point> 表示 , 逆时针
    db ans=0;
    for (int i=0;i<A.size();i++) ans+=cross(A[i],A[(i+1)%A.size()]);
    return ans/2;
}
int checkconvex(vector<point>A){
    int n=A.size(); A.push_back(A[0]); A.push_back(A[1]);
    for (int i=0;i<n;i++) if (sign(cross(A[i+1]-A[i],A[i+2]-A[i]))==-1) return 0;
    return 1;
}
int contain(vector<point>A,point q){ // 2 内部 1 边界 0 外部
    int pd=0; A.push_back(A[0]);
    for (int i=1;i<A.size();i++){
```

```
229         point u=A[i-1],v=A[i];
230         if (onS(u,v,q)) return 1; if (cmp(u.y,v.y)>0) swap(u,v);
231         if (cmp(u.y,q.y)>=0||cmp(v.y,q.y)<0) continue;
232         if (sign(cross(u-v,q-v))<0) pd^=1;
233     }
234     return pd<<1;
235 }
236 vector<point> ConvexHull(vector<point>A,int flag=1){ // flag=0 不严格 flag=1 严格
237     int n=A.size(); vector<point>ans(n*2);
238     sort(A.begin(),A.end()); int now=-1;
239     for (int i=0;i<A.size();i++){
240         while (now>0&&sign(cross(ans[now]-ans[now-1],A[i]-ans[now-1]))<flag) now--;
241         ans[++now]=A[i];
242     } int pre=now;
243     for (int i=n-2;i>=0;i--){
244         while (now>pre&&sign(cross(ans[now]-ans[now-1],A[i]-ans[now-1]))<flag) now--;
245         ans[++now]=A[i];
246     } ans.resize(now); return ans;
247 }
248 db convexDiameter(vector<point>A){
249     int now=0,n=A.size(); db ans=0;
250     for (int i=0;i<A.size();i++){
251         now=max(now,i);
252         while (1){
253             db k1=A[i].dis(A[now%n]),k2=A[i].dis(A[(now+1)%n]);
254             ans=max(ans,max(k1,k2)); if (k2>k1) now++; else break;
255         }
256     }
257     return ans;
258 }
259 vector<point> convexcut(vector<point>A,point k1,point k2){
260     // 保留 k1,k2,p 逆时针的所有点
261     int n=A.size(); A.push_back(A[0]); vector<point>ans;
262     for (int i=0;i<n;i++){
263         int w1=clockwise(k1,k2,A[i]),w2=clockwise(k1,k2,A[i+1]);
264         if (w1>=0) ans.push_back(A[i]);
265         if (w1*w2<0) ans.push_back(getLL(k1,k2,A[i],A[i+1]));
266     }
267     return ans;
268 }
269 int checkPoS(vector<point>A,point k1,point k2){
270     // 多边形 A 和直线 ( 线段 )k1->k2 严格相交 , 注释部分为线段
271     struct ins{
272         point m,u,v;
273         int operator < (const ins& k) const {return m<k.m;}
274     }; vector<ins>B;
275     //if (contain(A,k1)==2||contain(A,k2)==2) return 1;
276     vector<point>poly=A; A.push_back(A[0]);
277     for (int i=1;i<A.size();i++) if (checkLL(A[i-1],A[i],k1,k2)){
278         point m=getLL(A[i-1],A[i],k1,k2);
279         if (inmid(A[i-1],A[i],m)/*&&inmid(k1,k2,m)*/) B.push_back((ins){m,A[i-1],A[i]});
280     }
281     if (B.size()==0) return 0; sort(B.begin(),B.end());
282     int now=1; while (now<B.size()&&B[now].m==B[0].m) now++;
283     if (now==B.size()) return 0;
284     int flag=contain(poly,(B[0].m+B[now].m)/2);
285     if (flag==2) return 1;
286     point d=B[now].m-B[0].m;
287     for (int i=now;i<B.size();i++){
```

```
        if (!(B[i].m==B[i-1].m)&&flag==2) return 1;
        int tag=sign(cross(B[i].v-B[i].u,B[i].m+d-B[i].u));
        if (B[i].m==B[i].u||B[i].m==B[i].v) flag+=tag; else flag+=tag*2;
    }
    //return 0;
    return flag==2;
}
int checkinp(point r,point l,point m){
    if (compareangle(l,r)){return compareangle(l,m)&&compareangle(m,r);}
    return compareangle(l,m)||compareangle(m,r);
}
int checkPosFast(vector<point>A,point k1,point k2){ // 快速检查线段是否和多边形严格相交
    if (contain(A,k1)==2||contain(A,k2)==2) return 1; if (k1==k2) return 0;
    A.push_back(A[0]); A.push_back(A[1]);
    for (int i=1;i+1<A.size();i++)
        if (checkLL(A[i-1],A[i],k1,k2)){
            point now=getLL(A[i-1],A[i],k1,k2);
            if (inmid(A[i-1],A[i],now)==0||inmid(k1,k2,now)==0) continue;
            if (now==A[i]){
                if (A[i]==k2) continue;
                point pre=A[i-1],ne=A[i+1];
                if (checkinp(pre-now,ne-now,k2-now)) return 1;
            } else if (now==k1){
                if (k1==A[i-1]||k1==A[i]) continue;
                if (checkinp(A[i-1]-k1,A[i]-k1,k2-k1)) return 1;
            } else if (now==k2||now==A[i-1]) continue;
            else return 1;
        }
    return 0;
}
// 拆分凸包成上下凸壳 凸包尽量都随机旋转一个角度来避免出现相同横坐标
// 尽量特判只有一个点的情况 凸包逆时针
void getUDP(vector<point>A,vector<point>&U,vector<point>&D){
    db l=1e100,r=-1e100;
    for (int i=0;i<A.size();i++) l=min(l,A[i].x),r=max(r,A[i].x);
    int wherel,wherer;
    for (int i=0;i<A.size();i++) if (cmp(A[i].x,l)==0) wherel=i;
    for (int i=A.size();i;i--) if (cmp(A[i-1].x,r)==0) wherer=i-1;
    U.clear(); D.clear(); int now=wherel;
    while (1){D.push_back(A[now]); if (now==wherer) break; now++; if (now>=A.size()) now=0;}
    now=wherel;
    while (1){U.push_back(A[now]); if (now==wherer) break; now--; if (now<0) now=A.size()-1;}
}
// 需要保证凸包点数大于等于 3,2 内部 ,1 边界 ,0 外部
int containCoP(const vector<point>&U,const vector<point>&D,point k){
    db lx=U[0].x,rx=U[U.size()-1].x;
    if (k==U[0]||k==U[U.size()-1]) return 1;
    if (cmp(k.x,lx)==-1||cmp(k.x,rx)==1) return 0;
    int where1=lower_bound(U.begin(),U.end(),(point){k.x,-1e100})-U.begin();
    int where2=lower_bound(D.begin(),D.end(),(point){k.x,-1e100})-D.begin();
    int w1=clockwise(U[where1-1],U[where1],k),w2=clockwise(D[where2-1],D[where2],k);
    if (w1==1||w2==-1) return 0; else if (w1==0||w2==0) return 1; return 2;
}
// d 是方向 , 输出上方切点和下方切点
pair<point,point> getTangentCow(const vector<point> &U,const vector<point> &D,point d){
    if (sign(d.x)<0||(sign(d.x)==0&&sign(d.y)<0)) d=d*(-1);
    point whereU,whereD;
    if (sign(d.x)==0) return mp(U[0],U[U.size()-1]);
    int l=0,r=U.size()-1,ans=0;
```

```
347     while (l<r){int mid=l+r>>1; if (sign(cross(U[mid+1]-U[mid],d))<=0) l=mid+1,ans=mid+1; else r=
        mid;}
348     whereU=U[ans]; l=0,r=D.size()-1,ans=0;
349     while (l<r){int mid=l+r>>1; if (sign(cross(D[mid+1]-D[mid],d))>=0) l=mid+1,ans=mid+1; else r=
        mid;}
350     whereD=D[ans]; return mp(whereU,whereD);
351 }
352 // 先检查 contain, 逆时针给出
353 pair<point,point> getTangentCoP(const vector<point>&U,const vector<point>&D,point k){
354     db lx=U[0].x,rx=U[U.size()-1].x;
355     if (k.x<lx){
356         int l=0,r=U.size()-1,ans=U.size()-1;
357         while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid+1])==1) l=mid+1; else ans=mid,r=
        mid;}
358         point w1=U[ans]; l=0,r=D.size()-1,ans=D.size()-1;
359         while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid+1])==-1) l=mid+1; else ans=mid,r=
        mid;}
360         point w2=D[ans]; return mp(w1,w2);
361     } else if (k.x>rx){
362         int l=1,r=U.size(),ans=0;
363         while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid-1])==-1) r=mid; else ans=mid,l=mid
        +1;}
364         point w1=U[ans]; l=1,r=D.size(),ans=0;
365         while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid-1])==1) r=mid; else ans=mid,l=mid
        +1;}
366         point w2=D[ans]; return mp(w2,w1);
367     } else {
368         int where1=lower_bound(U.begin(),U.end(),(point){k.x,-1e100})-U.begin();
369         int where2=lower_bound(D.begin(),D.end(),(point){k.x,-1e100})-D.begin();
370         if ((k.x==lx&&k.y>U[0].y)||(where1&&clockwise(U[where1-1],U[where1],k)==1)){
371             int l=1,r=where1+1,ans=0;
372             while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid-1])==1) ans=mid,l=mid+1; else
        r=mid;}
373             point w1=U[ans]; l=where1,r=U.size()-1,ans=U.size()-1;
374             while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid+1])==1) l=mid+1; else ans=mid,
        r=mid;}
375             point w2=U[ans]; return mp(w2,w1);
376         } else {
377             int l=1,r=where2+1,ans=0;
378             while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid-1])==-1) ans=mid,l=mid+1; else
         r=mid;}
379             point w1=D[ans]; l=where2,r=D.size()-1,ans=D.size()-1;
380             while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid+1])==-1) l=mid+1; else ans=mid
        ,r=mid;}
381             point w2=D[ans]; return mp(w1,w2);
382         }
383     }
384 }
385 struct P3{
386     db x,y,z;
387     P3 operator + (P3 k1){return (P3){x+k1.x,y+k1.y,z+k1.z};}
388     P3 operator - (P3 k1){return (P3){x-k1.x,y-k1.y,z-k1.z};}
389     P3 operator * (db k1){return (P3){x*k1,y*k1,z*k1};}
390     P3 operator / (db k1){return (P3){x/k1,y/k1,z/k1};}
391     db abs2(){return x*x+y*y+z*z;}
392     db abs(){return sqrt(x*x+y*y+z*z);}
393     P3 unit(){return (*this)/abs();}
394     int operator < (const P3 k1) const{
395         if (cmp(x,k1.x)!=0) return x<k1.x;
```

```
396         if (cmp(y,k1.y)!=0) return y<k1.y;
397         return cmp(z,k1.z)==-1;
398     }
399     int operator == (const P3 k1){
400         return cmp(x,k1.x)==0&&cmp(y,k1.y)==0&&cmp(z,k1.z)==0;
401     }
402     void scan(){
403         double k1,k2,k3; scanf("%lf%lf%lf",&k1,&k2,&k3);
404         x=k1; y=k2; z=k3;
405     }
406 };
407 P3 cross(P3 k1,P3 k2){return (P3){k1.y*k2.z-k1.z*k2.y,k1.z*k2.x-k1.x*k2.z,k1.x*k2.y-k1.y*k2.x};}
408 db dot(P3 k1,P3 k2){return k1.x*k2.x+k1.y*k2.y+k1.z*k2.z;}
409 //p=(3,4,5),l=(13,19,21),theta=85 ans=(2.83,4.62,1.77)
410 P3 turn3D(db k1,P3 l,P3 p){
411     l=l.unit(); P3 ans; db c=cos(k1),s=sin(k1);
412     ans.x=p.x*(l.x*l.x*(1-c)+c)+p.y*(l.x*l.y*(1-c)-l.z*s)+p.z*(l.x*l.z*(1-c)+l.y*s);
413     ans.y=p.x*(l.x*l.y*(1-c)+l.z*s)+p.y*(l.y*l.y*(1-c)+c)+p.z*(l.y*l.z*(1-c)-l.x*s);
414     ans.z=p.x*(l.x*l.z*(1-c)-l.y*s)+p.y*(l.y*l.z*(1-c)+l.x*s)+p.z*(l.x*l.x*(1-c)+c);
415     return ans;
416 }
417 typedef vector<P3> VP;
418 typedef vector<VP> VVP;
419 db Acos(db x){return acos(max(-(db)1,min(x,(db)1)));}
420 // 球面距离 , 圆心原点 , 半径 1
421 db Odist(P3 a,P3 b){db r=Acos(dot(a,b)); return r;}
422 db r; P3 rnd;
423 vector<db> solve(db a,db b,db c){
424     db r=sqrt(a*a+b*b),th=atan2(b,a);
425     if (cmp(c,-r)==-1) return {0};
426     else if (cmp(r,c)<=0) return {1};
427     else {
428         db tr=pi-Acos(c/r); return {th+pi-tr,th+pi+tr};
429     }
430 }
431 vector<db> jiao(P3 a,P3 b){
432     // dot(rd+x*cos(t)+y*sin(t),b) >= cos(r)
433     if (cmp(Odist(a,b),2*r)>0) return {0};
434     P3 rd=a*cos(r),z=a.unit(),y=cross(z,rnd).unit(),x=cross(y,z).unit();
435     vector<db> ret = solve(-(dot(x,b)*sin(r)),-(dot(y,b)*sin(r)),-(cos(r)-dot(rd,b)));
436     return ret;
437 }
438 db norm(db x,db l=0,db r=2*pi){ // change x into [l,r)
439     while (cmp(x,l)==-1) x+=(r-l); while (cmp(x,r)>=0) x-=(r-l);
440     return x;
441 }
442 db disLP(P3 k1,P3 k2,P3 q){
443     return (cross(k2-k1,q-k1)).abs()/(k2-k1).abs();
444 }
445 db disLL(P3 k1,P3 k2,P3 k3,P3 k4){
446     P3 dir=cross(k2-k1,k4-k3); if (sign(dir.abs())==0) return disLP(k1,k2,k3);
447     return fabs(dot(dir.unit(),k1-k2));
448 }
449 VP getFL(P3 p,P3 dir,P3 k1,P3 k2){
450     db a=dot(k2-p,dir),b=dot(k1-p,dir),d=a-b;
451     if (sign(fabs(d))==0) return {};
452     return {(k1*a-k2*b)/d};
453 }
454 VP getFF(P3 p1,P3 dir1,P3 p2,P3 dir2){// 返回一条线
```

```
455     P3 e=cross(dir1,dir2),v=cross(dir1,e);
456     db d=dot(dir2,v); if (sign(abs(d))==0) return {};
457     P3 q=p1+v*dot(dir2,p2-p1)/d; return {q,q+e};
458 }
459 // 3D Covex Hull Template
460 db getV(P3 k1,P3 k2,P3 k3,P3 k4){ // get the Volume
461     return dot(cross(k2-k1,k3-k1),k4-k1);
462 }
463 db rand_db(){return 1.0*rand()/RAND_MAX;}
464 VP convexHull2D(VP A,P3 dir){
465     P3 x={(db)rand(),(db)rand(),(db)rand()}; x=x.unit();
466     x=cross(x,dir).unit(); P3 y=cross(x,dir).unit();
467     P3 vec=dir.unit()*dot(A[0],dir);
468     vector<point>B;
469     for (int i=0;i<A.size();i++) B.push_back((point){dot(A[i],x),dot(A[i],y)});
470     B=ConvexHull(B); A.clear();
471     for (int i=0;i<B.size();i++) A.push_back(x*B[i].x+y*B[i].y+vec);
472     return A;
473 }
474 namespace CH3{
475     VVP ret; set<pair<int,int> >e;
476     int n; VP p,q;
477     void wrap(int a,int b){
478         if (e.find({a,b})==e.end()){
479             int c=-1;
480             for (int i=0;i<n;i++) if (i!=a&&i!=b){
481                 if (c==-1||sign(getV(q[c],q[a],q[b],q[i]))>0) c=i;
482             }
483             if (c!=-1){
484                 ret.push_back({p[a],p[b],p[c]});
485                 e.insert({a,b}); e.insert({b,c}); e.insert({c,a});
486                 wrap(c,b); wrap(a,c);
487             }
488         }
489     }
490     VVP ConvexHull3D(VP _p){
491         p=q=_p; n=p.size();
492         ret.clear(); e.clear();
493         for (auto &i:q) i=i+(P3){rand_db()*1e-4,rand_db()*1e-4,rand_db()*1e-4};
494         for (int i=1;i<n;i++) if (q[i].x<q[0].x) swap(p[0],p[i]),swap(q[0],q[i]);
495         for (int i=2;i<n;i++) if ((q[i].x-q[0].x)*(q[1].y-q[0].y)>(q[i].y-q[0].y)*(q[1].x-q[0].x))
    swap(q[1],q[i]),swap(p[1],p[i]);
496         wrap(0,1);
497         return ret;
498     }
499 }
500 VVP reduceCH(VVP A){
501     VVP ret; map<P3,VP> M;
502     for (VP nowF:A){
503         P3 dir=cross(nowF[1]-nowF[0],nowF[2]-nowF[0]).unit();
504         for (P3 k1:nowF) M[dir].pb(k1);
505     }
506     for (pair<P3,VP> nowF:M) ret.pb(convexHull2D(nowF.se,nowF.fi));
507     return ret;
508 }
509 //  把一个面变成 ( 点 , 法向量 ) 的形式
510 pair<P3,P3> getF(VP F){
511     return mp(F[0],cross(F[1]-F[0],F[2]-F[0]).unit());
512 }
```

```
513 // 3D Cut 保留 dot(dir,x-p)>=0 的部分
514 VVP ConvexCut3D(VVP A,P3 p,P3 dir){
515     VVP ret; VP sec;
516     for (VP nowF: A){
517         int n=nowF.size(); VP ans; int dif=0;
518         for (int i=0;i<n;i++){
519             int d1=sign(dot(dir,nowF[i]-p));
520             int d2=sign(dot(dir,nowF[(i+1)%n]-p));
521             if (d1>=0) ans.pb(nowF[i]);
522             if (d1*d2<0){
523                 P3 q=getFL(p,dir,nowF[i],nowF[(i+1)%n])[0];
524                 ans.push_back(q); sec.push_back(q);
525             }
526             if (d1==0) sec.push_back(nowF[i]); else dif=1;
527             dif|=(sign(dot(dir,cross(nowF[(i+1)%n]-nowF[i],nowF[(i+1)%n]-nowF[i])))==-1);
528         }
529         if (ans.size()>0&&dif) ret.push_back(ans);
530     }
531     if (sec.size()>0) ret.push_back(convexHull2D(sec,dir));
532     return ret;
533 }
534 db vol(VVP A){
535     if (A.size()==0) return 0; P3 p=A[0][0]; db ans=0;
536     for (VP nowF:A)
537         for (int i=2;i<nowF.size();i++)
538             ans+=abs(getV(p,nowF[0],nowF[i-1],nowF[i]));
539     return ans/6;
540 }
541 VVP init(db INF) {
542     VVP pss(6,VP(4));
543     pss[0][0] = pss[1][0] = pss[2][0] = {-INF, -INF, -INF};
544     pss[0][3] = pss[1][1] = pss[5][2] = {-INF, -INF, INF};
545     pss[0][1] = pss[2][3] = pss[4][2] = {-INF, INF, -INF};
546     pss[0][2] = pss[5][3] = pss[4][1] = {-INF, INF, INF};
547     pss[1][3] = pss[2][1] = pss[3][2] = {INF, -INF, -INF};
548     pss[1][2] = pss[5][1] = pss[3][3] = {INF, -INF, INF};
549     pss[2][2] = pss[4][3] = pss[3][1] = {INF, INF, -INF};
550     pss[5][0] = pss[4][0] = pss[3][0] = {INF, INF, INF};
551     return pss;
552 }
```

5 String

5.1 KMP

5.1.1 KMP

```
const int maxn = 1e6 + 10;

char a[maxn], b[maxn];
int nex[maxn];

void getNext()
{
    int len = strlen(b), i = 0, j = -1;
    nex[i] = j;
    for (int i = 1; i < len; i++)
    {
        while (j != -1 && b[i + 1] != b[j]) j = nex[j];
        if (b[i] == b[j + 1]) j++;
        nex[i] = j;
    }
}

void KMP()
{
    int n = strlen(a), m = strlen(b);
    getNext();
    int j = -1;
    for (int i = 0; i < n; i++)
    {
        while (j != -1 && a[i] != b[j + 1]) j = nex[j];
        if (b[j + 1] == a[i]) j++;
    }
}
```

5.1.2 exKMP

```
const int maxn = 1e5 + 10;
int nex[maxn], extend[maxn];

//预处理计算Next数组
void getNext(char *str)
{
    int i = 0, j, po, len = strlen(str);
    nex[0] = len;     //初始化nex[0]
    while (str[i] == str[i + 1] && i + 1 < len) i++;   //计算nex[1]
    nex[1] = i;
    po = 1;   //初始化po的位置
    for (int i = 2; i < len; i++)
    {
        if (nex[i - po] + i < nex[po] + po)  //第一种情况，可以直接得到nex[i]的值
            nex[i] = nex[i - po];
        else    //第二种情况，要继续匹配才能得到nex[i]的值
        {
            j = nex[po] + po - i;
            if (j < 0) j = 0;    //如果i>po+nex[po],则要从头开始匹配
            while (i + j < len && str[j] == str[j + i]) j++;
            nex[i] = j;
            po = i;   //更新po的位置
```

```
        }
    }
}

void EXKMP(char *s1, char *s2)
{
    int i = 0, j, po, len = strlen(s1), l2 = strlen(s2);
    getNext(s2);
    while (s1[i] == s2[i] && i < l2 && i < len) i++;
    extend[0] = i;
    po = 0;
    for (int i = 1; i < len; i++)
    {
        if (nex[i - po] + i < extend[po] + po)
            extend[i] = nex[i - po];
        else
        {
            j = extend[po] + po - i;
            if (j < 0) j = 0;
            while (i + j < len && j < l2 && s1[j + i] == s2[j]) j++;
            extend[i] = j;
            po = i;
        }
    }
}
```

5.2 Trie

5.2.1 Trie

```
const int maxn = 2e6 + 10;

int trie[maxn][30], tot;
bool flag[maxn];

void insert_ch(char *str)
{
    int len = strlen(str);
    int root = 0;
    for (int i = 0; i < len; i++)
    {
        int id = str[i] - 'a';
        if (!trie[root][id]) trie[root][id] = ++tot;
        root = trie[root][id];
    }
    flag[root] = true;
}

bool find_ch(char *str)
{
    int len = strlen(str);
    int root = 0;
    for (int i = 0; i < len; i++)
    {
        int id = str[i] - 'a';
        if (!trie[root][id]) return false;
        root = trie[root][id];
    }
    return true;
```

```
}
```

5.2.2 Persistence Trie

```
const int maxn = 1e5 + 10;

int a[maxn], rt[maxn], n;

struct Trie
{
    int tot;
    int child[maxn * 32][2], sum[maxn *32];
    int insert(int x, int val)
    {
        int tmp, y;
        tmp = y= ++tot;
        for(int i = 30; i >= 0; --i)
        {
            child[y][0] = child[x][0];
            child[y][1] = child[x][1];
            sum[y] = sum[x] + 1;
            int t = val >> i & 1;
            x = child[x][t];
            child[y][t] = ++tot;
            y = child[y][t];
        }
        sum[y] = sum[x] + 1;
        return tmp;
    }
    int query(int l, int r, int val)
    {
        int tmp = 0;
        for(int i =30; i >= 0; --i)
        {
            int t = val >> i & 1;
            if(sum[child[r][t^1]] - sum[child[l][t^1]]) tmp += (1<<i), r = child[r][t^1], l = child
    [l][t ^ 1];
            else r = child[r][t], l = child[l][t];
        }
        return tmp;
    }
}trie;
```

5.3 Manachar

5.3.1 Manachar

```
const int maxn = 1e5 + 10;

char s[maxn];

char tmp[maxn << 1];
int Len[maxn << 1];

int init(char *str)
{
    int i, len = strlen(str);
    tmp[0] = '@';
```

```
    for (int i = 1; i <= 2 * len; i += 2)
    {
        tmp[i] = '#';
        tmp[i + 1] = str[i / 2];
    }
    tmp[2 * len + 1] = '#';
    tmp[2 * len + 2] = '$';
    tmp[2 * len + 3] = 0;
    return 2 * len + 1;
}

int manacher(char *str)
{
    int mx = 0, ans = 0, pos = 0;
    int len = init(str);
    for (int i = 1; i <= len; i++)
    {
        if (mx > i) Len[i] = min(mx - i, Len[2 * pos - i]);
        else Len[i] = 1;
        while (tmp[i - Len[i]] == tmp[i + Len[i]]) Len[i]++;
        if (Len[i] + i > mx) mx = Len[i] + i, pos = i;
    }
}
```

5.4 Aho-Corasick Automation

5.4.1 AC Automation

```
const int maxn = 5e5 + 10;

class AC_automation
{
public:
    int trie[maxn][26], cnt;
    int tag[maxn];
    int fail[maxn];

    void init()
    {
        memset(trie, 0, sizeof trie);
        memset(tag, 0, sizeof tag);
        memset(fail, 0, sizeof fail);
        cnt = 0;
    }

    void insert(char *str)
    {
        int root = 0;
        for (int i = 0; str[i]; i++)
        {
            int id = str[i] - 'a';
            if (!trie[root][id]) trie[root][id] = ++cnt;
            root = trie[root][id];
        }
        tag[root]++;
    }

    void build()
    {
```

```
        queue<int> que;
        for (int i = 0; i < 26; i++) if (trie[0][i]) que.push(trie[0][i]);
        while (!que.empty())
        {
            int k = que.front();
            que.pop();
            for (int i = 0; i < 26; i++)
            {
                if (trie[k][i])
                {
                    fail[trie[k][i]] = trie[fail[k]][i];
                    que.push(trie[k][i]);
                } else trie[k][i] = trie[fail[k]][i];
            }
        }
    }

    int query(char *str)
    {
        int p = 0, res = 0;
        for (int i = 0; str[i]; i++)
        {
            p = trie[p][str[i] - 'a'];
            for (int j = p; j && ~tag[j]; j = fail[j]) res += tag[j], tag[j] = -1;
        }
        return res;
    }
} AC;
```

5.5 Suffix Array

5.5.1 Suffix Array

```
char s[maxn];
int sa[maxn], t[maxn], t2[maxn], c[maxn], n;

void build_sa(int n, int m)
{
    int *x = t, *y = t2;
    for(int i = 0; i < m; i++) c[i] = 0;
    for(int i = 0; i < n; i++) c[x[i] = s[i]]++;
    for(int i = 1; i < m; i++) c[i] += c[i - 1];
    for(int i = n - 1; i >= 0; i--) sa[--c[x[i]]] = i;
    for(int k = 1; k <= n; k <<= 1)
    {
        int p = 0;
        for(int i = n - k; i < n; i++) y[p++] = i;
        for(int i = 0; i < n; i++) if(sa[i] >= k) y[p++] = sa[i] - k;
        for(int i = 0; i < m; i++) c[i] = 0;
        for(int i = 0; i < n; i++) c[x[y[i]]]++;
        for(int i = 0; i < m; i++) c[i] += c[i - 1];
        for(int i = n - 1; i >= 0; i--) sa[--c[x[y[i]]]] = y[i];
        swap(x, y);
        p = 1; x[sa[0]] = 0;
        for(int i = 1; i < n; i++)
            x[sa[i]] = y[sa[i - 1]] == y[sa[i]] && y[sa[i - 1] + k] == y[sa[i] + k] ? p - 1 : p++;
        if(p >= n) break;
        m = p;
    }
```

```
}

int rk[maxn], height[maxn];

void getHeight()
{
    for(int i = 0; i < n; i++) rk[sa[i]] = i;
    for(int i = 0, k = 0; i < n; i++)
    {
        if(k) k--;
        int j = sa[rk[i] - 1];
        while(s[i + k] == s[j + k]) k++;
        height[rk[i]] = k;
    }
}
```

5.6 PalindromicTree

5.6.1 PalindromicTree

```
// 求相交回文串数量

#include<bits/stdc++.h>

#define ll long long
using namespace std;

const int maxn = 2e6+6;
const int N = 26;
const int mod = 51123987;

struct Palindromic_Tree {
    vector<pair<int, int> > next[maxn];
//    int next[maxn][N];//next指针，next指针和字典树类似，指向的串为当前串两端加上同一个字符构成
    int fail[maxn]{};//fail指针，失配后跳转到fail指针指向的节点
    int cnt[maxn]{}; //表示节点i表示的本质不同的串的个数（建树时求出的不是完全的，最后count()函数跑一遍以后才是正确的）
    int num[maxn]{}; //表示以节点i表示的最长回文串的最右端点为回文串结尾的回文串个数
    int len[maxn]{};//len[i]表示节点i表示的回文串的长度（一个节点表示一个回文串）
    int S[maxn]{};//存放添加的字符
    int last{};//指向新添加一个字母后所形成的最长回文串表示的节点。
    int n{};//表示添加的字符个数。
    int p{};//表示添加的节点个数。

    int newnode(int l) {//新建节点
        next[p].clear();
//        for (int i = 0; i < N; ++i) next[p][i] = 0;
//        cnt[p] = 0;
//        num[p] = 0;
        len[p] = l;
        return p++;
    }

    void init() {//初始化
        n = last = p = 0;
        newnode(0);
        newnode(-1);
        S[n] = -1;//开头放一个字符集中没有的字符，减少特判
        fail[0] = 1;
```

```
39      }
40
41      int get_fail(int x) {//和KMP一样，失配后找一个尽量最长的
42          while (S[n - len[x] - 1] != S[n]) x = fail[x];
43          return x;
44      }
45
46      int find(int u, int c) {
47          vector<pair<int, int> > & x = next[u];
48          int sz = x.size();
49          for(int i = 0; i < sz; ++i) {
50              if(x[i].first == c) return x[i].second;
51          }
52          return 0;
53      }
54
55      int add(int c) {
56          S[++n] = c;
57          int cur = get_fail(last);//通过上一个回文串找这个回文串的匹配位置
58          int x = find(cur, c);
59          if (!x) {
60  //          if (!next[cur][c]) {//如果这个回文串没有出现过，说明出现了一个新的本质不同的回文串
61              int now = newnode(len[cur] + 2);//新建节点
62              x = now;
63              fail[now] = find(get_fail(fail[cur]), c);
64              next[cur].emplace_back(make_pair(c, now));
65  //              fail[now] = next[get_fail(fail[cur])][c];//和AC自动机一样建立fail指针，以便失配后跳转
66  //              next[cur][c] = now;
67              num[now] = num[fail[now]] + 1;
68          }
69          last = x;
70  //          last = next[cur][c];
71  //          cnt[last]++;
72          return num[last];
73      }
74
75      void count() {
76          for (int i = p - 1; i >= 0; --i) cnt[fail[i]] += cnt[i];
77          //父亲累加儿子的cnt，因为如果fail[v]=u，则u一定是v的子回文串!
78      }
79  } solve;
80
81  char s[maxn];
82
83  ll a[maxn], b[maxn];
84  int main() {
85      solve.init();
86      int n;
87      scanf("%d", &n);
88      scanf("%s", s);
89      for (int i = 0; i < n; ++i) {
90          a[i] = solve.add(s[i] - 'a');
91      }
92      solve.init();
93      for (int i = n - 1; i >= 0; --i) {
94          b[i] = (b[i + 1] + solve.add(s[i] - 'a')) % mod;
95      }
96      ll res = (b[0] * (b[0] - 1) / 2) % mod;
97      for (int i = 0; i < n; ++i) {
```

```
98          res = ((res - (a[i] * b[i + 1]) + mod) % mod) % mod;
99      }
100     printf("%lld\n", res);
101     return 0;
102 }
```

6 dp

6.1 BitDP

6.1.1 数位 dp 计和

```cpp
#include <bits/stdc++.h>
#define ll long long
using namespace std;
const int mod = 998244353;
pair<ll, ll> dp[20][1<<10];
bool vis[20][1<<10];
int k;
int t[20];
ll base[20];

pair<ll, ll> dfs(int pos, int state, bool limit, bool lead) {
    if (pos == -1) return __builtin_popcount(state) <= k ? make_pair(1, 0) : make_pair(0, 0);
    if (!limit && !lead && vis[pos][state]) return dp[pos][state];
    int up = limit ? t[pos] : 9;
    pair<ll, ll> res = {0, 0};
    for (int i = 0; i <= up; ++i) {
        int n_s = state;
        if (lead && i == 0) n_s = 0;
        else n_s = state | (1 << i);
        auto tmp = dfs(pos - 1, n_s, limit && i == t[pos], lead && i == 0);
        ll pre = 1ll * i * base[pos] % mod;
        (res.first += tmp.first) %= mod;
        (res.second += tmp.second + pre * tmp.first) %= mod;
    }
    if (!limit && !lead) dp[pos][state] = res, vis[pos][state] = 1;
    return res;
}

ll solve(ll x) {
    int pos = 0;
    do {
        t[pos ++] = x % 10;
    } while (x /= 10);
    return dfs(pos - 1, 0, true, true).second;
}

int main(int argc,char *argv[])
{
    base[0] = 1;
    for (int i = 1; i < 20; ++i) base[i] = base[i - 1] * 10;
    ll l, r;
    scanf("%lld%lld%d", &l, &r, &k);
    printf("%lld\n", (solve(r) - solve(l - 1) + mod) % mod);
    return 0;
}
```

6.1.2 一般数位 dp

```cpp
int a[20];
ll dp[20][state];
ll dfs(int pos, /*state变量*/, bool lead /*前导零*/, bool limit /*数位上界变量*/)
{
    //递归边界，既然是按位枚举，最低位是0，那么pos==-1说明这个数枚举完了
```

```
    if (pos == -1) return 1;
    /*这里一般返回1，表示枚举的这个数是合法的，那么这里就需要在枚举时必须每一位都要满足题目条件，
    也就是说当前枚举到pos位，一定要保证前面已经枚举的数位是合法的。*/
    if (!limit && !lead && dp[pos][state] != -1) return dp[pos][state];
    /*常规写法都是在没有限制的条件记忆化，这里与下面记录状态是对应*/
    int up = limit ? a[pos] : 9; //根据limit判断枚举的上界up
    ll ans = 0;
    for (int i = 0; i <= up; i++) //枚举，然后把不同情况的个数加到ans就可以了
    {
        if () ...
        else if () ...
        ans += dfs(pos - 1, /*状态转移*/, lead && i == 0, limit && i == a[pos])
        //最后两个变量传参都是这样写的
        /*当前数位枚举的数是i，然后根据题目的约束条件分类讨论
        去计算不同情况下的个数，还有要根据state变量来保证i的合法性*/
    }
    //计算完，记录状态
    if (!limit && !lead) dp[pos][state] = ans;
    /*这里对应上面的记忆化，在一定条件下时记录，保证一致性，
    当然如果约束条件不需要考虑lead，这里就是lead就完全不用考虑了*/
    return ans;
}
ll solve(ll x)
{
    int pos = 0;
    do //把数位都分解出来
        a[pos++] = x % 10;
    while (x /= 10);
    return dfs(pos - 1 /*从最高位开始枚举*/, /*一系列状态 */, true, true);
    //刚开始最高位都是有限制并且有前导零的，显然比最高位还要高的一位视为0
}
```

6.2 StateDP

6.3 Subsequence

6.3.1 MaxSum

```
// 传入序列a和长度n，返回最大子序列和
int MaxSeqSum(int a[], int n)
{
    int rt = 0, cur = 0;
    for (int i = 0; i < n; i++)
        cur += a[i], rt = max(cur, rt), cur = max(0, cur);
    return rt;
}
```

6.3.2 LIS

```
// 序列下标从1开始，LIS()返回长度，序列存在lis[]中
const int N = "Edit";
int len, a[N], b[N], f[N];
int Find(int p, int l, int r)
{
    while (l <= r)
    {
        int mid = (l + r) >> 1;
        if (a[p] > b[mid])
```

```
            l = mid + 1;
        else
            r = mid - 1;
    }
    return f[p] = l;
}
int LIS(int lis[], int n)
{
    int len = 1;
    f[1] = 1, b[1] = a[1];
    for (int i = 2; i <= n; i++)
    {
        if (a[i] > b[len])
            b[++len] = a[i], f[i] = len;
        else
            b[Find(i, 1, len)] = a[i];
    }
    for (int i = n, t = len; i >= 1 && t >= 1; i--)
        if (f[i] == t) lis[--t] = a[i];
    return len;
}

// 简单写法(下标从0开始,只返回长度)
int dp[N];
int LIS(int a[], int n)
{
    memset(dp, 0x3f, sizeof(dp));
    for (int i = 0; i < n; i++) *lower_bound(dp, dp + n, a[i]) = a[i];
    return lower_bound(dp, dp + n, INF) - dp;
}
```

6.3.3 LongestCommonIncrease

```
// 序列下标从1开始
int LCIS(int a[], int b[], int n, int m)
{
    memset(dp, 0, sizeof(dp));
    for (int i = 1; i <= n; i++)
    {
        int ma = 0;
        for (int j = 1; j <= m; j++)
        {
            dp[i][j] = dp[i - 1][j];
            if (a[i] > b[j]) ma = max(ma, dp[i - 1][j]);
            if (a[i] == b[j]) dp[i][j] = ma + 1;
        }
    }
    return *max_element(dp[n] + 1, dp[n] + 1 + m);
}
```

6.4 Others

问题 设 $f(i) = \min(y[k] - s[i] \times x[k]), k \in [1, i-1]$, 现在要求出所有 $f(i), i \in [1, n]$

考虑两个决策 j 和 k，如果 j 比 k 优，则

$$y[j] - s[i] \times x[j] < y[k] - s[i] \times x[k]$$

化简得:

$$\frac{y_j - y_k}{x_j - x_k} < s_i$$

不等式左边是个斜率，我们把它设为 slope(j,k)

我们可以维护一个单调递增的队列，为什么呢?

因为如果 slope$(q[i-1],q[i])$ > slope$(q[i],q[i+1])$，那么当前者成立时，后者必定成立。即 $q[i]$ 决策优于 $q[i-1]$ 决策时，$q[i+1]$ 必然优于 $q[i]$，因此 $q[i]$ 就没有存在的必要了。所以我们要维护递增的队列。

那么每次的决策点 i，都要满足

$$\begin{cases} \text{slope}(q[i-1],q[i]) < s[i] \\ \text{slope}(q[i],q[i+1]) \geq s[i] \end{cases}$$

一般情况去二分这个 i 即可。

如果 $s[i]$ 是单调不降的，那么对于决策 j 和 $k(j<k)$ 来说，如果决策 k 优于决策 j，那么对于 $i \in [k+1,n]$，都存在决策 k 优于决策 j，因此决策 j 就可以舍弃了。这样的话我们可以用单调队列进行优化，可以少个 log。

单调队列滑动窗口最大值

```
// k为滑动窗口的大小
deque<int> q;
for (int i = 0, j = 0; i + k <= d; i++)
{
    while (j < i + k)
    {
        while (!q.empty() && a[q.back()] < a[j]) q.pop_back();
        q.push_back(j++);
    }
    while (q.front() < i) q.pop_front();
    // a[q.front()]为当前滑动窗口的最大值
}
```

6.4.1 矩阵快速幂

```
const int mod = 1e9 + 7;
typedef long long ll;

int cur;
struct Matrix {ll a[105][105]; };

Matrix mul(Matrix a, Matrix b)
{
    Matrix res;
    memset(res.a, 0, sizeof res.a);
    for(int i = 0; i < cur; i++)
        for(int j = 0; j < cur; j++)
            for(int k = 0; k < cur; k++)
                (res.a[i][j] +=a.a[i][k] * b.a[k][j] % mod) %= mod;
    return res;
}

Matrix pow(Matrix a, ll n)
{
    Matrix ans, base = a;
    for(int i = 0; i < cur; i++) ans.a[i][i] = 1;
    while(n)
    {
        if(n & 1) ans = mul(ans, base);
        base = mul(base, base);
        n >>= 1;
    }
```

```
28      return ans;
29  }
```

7 Others