算法篇

1.字符串

1.1 KMP

```cpp
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1e6 + 5;
int nxt[maxn], n, m;
char str[maxn], ptr[maxn];
int main(){
        scanf("%s%s", str + 1, ptr + 1);
        m = strlen(ptr + 1), n = strlen(str + 1);
        for(int i = 2, j = 0; i <= m; i ++){
                while(j and ptr[j + 1] != ptr[i]) j = nxt[j];
                if(ptr[i] == ptr[j + 1]) j ++;
                nxt[i] = j;
        }
        for(int i = 1, j = 0; i <= n; i ++){
                while(j and ptr[j + 1] != str[i]) j = nxt[j];
                if(str[i] == ptr[j + 1]) j ++;
                if(j == m) printf("%d\n", i - j + 1);
        }
        for(int i = 1; i <= m; i ++) printf("%d ", nxt[i]);
        return 0;
}
```

1.2 manacher

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 11000005;
char str[maxn], ptr[maxn<<1]; int N, cnt, p[maxn<<1];
int main(){
        scanf("%s", str+1);
        N = strlen(str+1);
        ptr[0] = '$';
        for(int i=1; i<=N; i++) ptr[++cnt] = '#', ptr[++cnt] = str[i];
        ptr[++cnt] = '#', ptr[++cnt] = '\0';
        int mx = 0, id, ans = -1; p[0] = 1;
        for(int i = 1; i <= cnt; i++){
                if(i < mx) p[i] = min(p[id*2 - i], mx - i);
                else p[i] = 1;
                while(ptr[i - p[i]] == ptr[i + p[i]]) p[i] ++;
                if(mx < i + p[i]){
                        id = i; mx = i + p[i];
                }
                ans = max(ans, p[i] - 1);
        }
        printf("%d", ans);
        return 0;
}
```

1.3 AC自动机

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1e6 + 5;
char ptr[maxn];
queue<int> q;
struct Aho_Corasick_Automaton{
        int trie[maxn][26], end[maxn], fail[maxn], cnt;
        void insert(char *str){
                int len = strlen(str); int now = 0;
                for(int i=0; i<len; i++){
                        if(!trie[now][str[i] - 'a']) trie[now][str[i] - 'a'] = ++cnt;
                        now = trie[now][str[i] - 'a'];
                }
                end[now] ++;
        }
        void build(){
                for(int i=0; i<26; i++) if(trie[0][i]) fail[trie[0][i]] = 0, q.push(trie[0][i]);
                while(!q.empty()){
                        int x = q.front(); q.pop();
                        for(int i=0; i<26; i++){
                                if(trie[x][i]) fail[trie[x][i]] = trie[fail[x]][i], q.push(trie[x]
                                else trie[x][i] = trie[fail[x]][i];
                        }
                }
        }
        int ask(char *str){
                int len = strlen(str); int now = 0, ans = 0;
                for(int i=0; i<len; i++){
                        now = trie[now][str[i] - 'a'];
                        for(int j=now; j and end[j] != -1; j = fail[j]) ans += end[j], end[j] = -1
                }
                return ans;
        }
}AC;
int main(){
        int N;
        scanf("%d", &N);
        for(int i=1; i<=N; i++){
                scanf("%s", ptr);
                AC.insert(ptr);
        }
        AC.build();
        scanf("%s", ptr);
        printf("%d", AC.ask(ptr));
        return 0;
}
```

1.4 SA

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1e6+5;
int sa[maxn], rank[maxn], newRank[maxn], sum[maxn], key2[maxn];
int n, m;
char  str[maxn];
void getHeight()
{
        int k = 0;
        for(int i=1; i<=n; i++)
        {
                if(rank[i] == 1) continue;
                int j = sa[rank[i]-1];
                while(str[j+k] == str[i+k] and j+k<=n and i+k<=n) k++;
                height[rank[i]] = k;
                if(k != 0) k--;
        }
}
bool cmp(int a, int b, int l)
{
        if(rank[a] != rank[b]) return false;
        if( (a+l > n and b+l <= n) or (a+l <= n and b+l > n) ) return false;
        if(a+l > n and b+l > n) return true;
        return rank[a+l] == rank[b+l];
}

int main()
{
        scanf("%s", str+1);
        n = strlen(str+1);
        for(int i=1; i<=n; i++) sum[rank[i] = str[i]]++;
        m = max(n, 256);
        for(int i=1; i<=m; i++) sum[i]+=sum[i-1];
        for(int i=n; i>=1; i--) sa[sum[rank[i]]--] = i;

        for(int l=1; l<n; l<<=1)//倍增
        {
                int k = 0;
                for(int i=n-l+1; i<=n; i++) key2[++k] = i;
```

```
            for(int i=1; i<=n; i++) if(sa[i] > l) key2[++k] = sa[i]-l;
            for(int i=1; i<=m; i++) sum[i] = 0;
            for(int i=1; i<=n; i++) sum[rank[i]]++;//分类
            for(int i=1; i<=m; i++) sum[i]+=sum[i-1];//前缀和
            for(int i=n; i>=1; i--)
            {
                int j = key2[i];
                sa[sum[rank[j]]--] = j;
            }

            int rk = 1;
            newRank[sa[1]] = rk;//计算排名，排序后的第一名的排名为1
            for(int i=2; i<=n; i++)
            {
                if(cmp(sa[i-1], sa[i], l)) newRank[sa[i]] = rk;
                else newRank[sa[i]] = ++rk;
            }
            for(int i=1; i<=n; i++) rank[i] = newRank[i];

            if(rk == n) break;
        }

        for(int i=1; i<=n; i++) printf("%d ",sa[i]);
        return 0;
}
```

1.5 SAM

```cpp
#include<bits/stdc++.h>
using namespace std;
const int maxn = 2e6 + 5;
int N;
struct SuffixAutomata{
        int maxlen[maxn], trans[maxn][26], fail[maxn], pos[maxn], siz, last;
        int id[maxn], rak[maxn], num[maxn];
        SuffixAutomata()
        {siz = last = 1;}
        void clear(){
                memset(maxlen, 0, sizeof(maxlen)), memset(trans, 0, sizeof(trans)), memset(fail,
                siz = last = 1;
        }
        inline void insert(int id){
                int cur = ++siz, p;
                maxlen[cur] = maxlen[last] + 1;
                pos[siz] = maxlen[siz];
                for(p = last; p and !trans[p][id]; p = fail[p]) trans[p][id] = cur;
                if(!p) fail[cur] = 1;
                else{
                        int q = trans[p][id];
                        if(maxlen[q] == maxlen[p] + 1) fail[cur] = q;
                        else{
                                int clone = ++siz;
                                pos[clone] = pos[q];
                                fail[clone] = fail[q];
                                maxlen[clone] = maxlen[p] + 1;
                                for(int i=0; i<26; i++) trans[clone][i] = trans[q][i];
                                for(; p and trans[p][id] == q; p = fail[p]) trans[p][id] = clone;
                                fail[cur] = fail[q] = clone;
                        }
                }
                last = cur;
                num[cur] = 1;
        }
        inline int calc(){
                for(int i=2; i<=siz; i++) rak[maxlen[i]] ++;
                for(int i=2; i<=siz; i++) rak[i] += rak[i-1];
                for(int i=2; i<=siz; i++) id[rak[maxlen[i]] --] = i;
                int ans = 0;
                for(int i=siz-1; i>=1; i--){
                        int now = id[i]; num[fail[now]] += num[now];
                        if(num[now] > 1) ans = max(ans, num[now]*maxlen[now]);
                }
                return ans;
        }
}trie;
char str[maxn];
int main(){
        scanf("%s", str + 1);
        int len = strlen(str + 1);
        for(int i=1; i<=len; i++) trie.insert(str[i] - 'a');
        printf("%d", trie.calc());
        return 0;
}
```

2. 图论

2.1 缩点

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1e4+5, maxm = 1e5+5, INF = 1e9+7;
int dval[maxn], dist[maxn];
int head[maxn], Next[maxm], ver[maxm], tot;
int head_c[maxn], Next_c[maxm], ver_c[maxm], tot_c;
int Stack[maxn], ins[maxn], dfn[maxn], c[maxn], low[maxn], top, num;
struct node
{
    int q[maxn];
    int tail;
    void clear()
    {
        tail=0;
    }
    bool cmp(int x,int y)
    {
        return dist[x] > dist[y];
    }
    void heap_up(int p)
    {
        while(p > 1 and cmp(q[p],q[p/2])) swap(q[p],q[p/2]),p/=2;
    }
    void heap_down(int p)
    {
        while(p*2 <= tail)
        {
            int tmp;
            if(p*2==tail or cmp(q[p*2],q[p*2+1])) tmp=p*2;
            else tmp=p*2+1;
            if(cmp(q[tmp],q[p])) swap(q[tmp],q[p]),p=tmp;
            else return ;
        }
    }
    void push(int x)
    {
        q[++tail] = x;
        heap_up(tail);
    }
    void pop()
    {
        q[1] = q[tail];
        tail--;
        heap_down(1);
    }
    int top()
    {
        return q[1];
    }
    bool empty()
    {
            return tail == 0;
    }
};
node Q; int in[maxn];
void add(int x,int y)
{
        ver[++tot] = y, Next[tot] = head[x], head[x] = tot;
}
void add_c(int x,int y)
{
        ver_c[++tot_c] = y, Next_c[tot_c] = head_c[x], head_c[x] = tot_c;
}
int n, m;
void tarjan(int x)
{
        dfn[x] = low[x] = ++num;
        ins[x] = 1, Stack[++top] = x;
        for(int i=head[x]; i; i=Next[i])
           if(!dfn[ver[i]])
           {
                   tarjan(ver[i]);
                   low[x] = min(low[x], low[ver[i]]);
           }
           else if(ins[ver[i]]) low[x] = min(low[x], dfn[ver[i]]);
        if(dfn[x] == low[x])
        {
                int y;
                do
                {
                        y=Stack[top--],ins[y] = 0, c[y] = x;
                        if(x == y) break;
                        dval[x]+=dval[y];
                }while(x!=y);
        }
}
void bfs()
{
        for(int i=1; i<=n; i++)
           if(in[i] == 0 and c[i] == i) Q.push(i), dist[i] = dval[i];
        while(!Q.empty())
        {
                int x = Q.top();Q.pop();
                for(int i=head_c[x]; i; i=Next_c[i])
                {
                        int y = ver_c[i];
                        in[y]--;
                        if(dist[y] < dist[x]+dval[y]) dist[y] = dist[x]+dval[y];
                        if(in[y] == 0) Q.push(y);
                }
        }
}
int main()
{
        scanf("%d%d",&n,&m);
        for(int i=1; i<=n; i++) scanf("%d",&dval[i]);
        for(int i=1; i<=m; i++)
        {
                int x,y;
                scanf("%d%d",&x,&y);
                add(x, y);
        }
        for(int i=1; i<=n; i++) if(!dfn[i]) tarjan(i);
        for(int i=1; i<=n; i++)
           for(int j=head[i]; j; j=Next[j])
           {
                     int x = i, y = ver[j];
```

```
                        if(c[x] == c[y]) continue;
                        add_c(c[x], c[y]); in[c[y]]++;
              }
          bfs();
          int ans = -INF;
          for(int i=1; i<=n; i++)
         ans = max(ans, dist[i]);
     printf("%d",ans);
          return 0;
}
```

2.2 dinic

```cpp
#include<bits/stdc++.h>
using namespace std;
#define int long long
const int maxn = 205, maxm = 5005, INF = 1e15 + 7;
inline int read(){
        int w = 0, f = 1; char ch = getchar();
        while(ch < '0' or ch > '9') {if(ch == '-') f = -f; ch = getchar();}
        while(ch >= '0' and ch <= '9') w = w*10 + ch - '0', ch = getchar();
        return w*f;
}
int N, M, S, T, head[maxn], ver[maxm<<1], edge[maxm<<1], Next[maxm<<1], tot;
void add(int x, int y, int z){
        ver[++tot] = y, edge[tot] = z, Next[tot] = head[x], head[x] = tot;
        ver[++tot] = x, edge[tot] = 0, Next[tot] = head[y], head[y] = tot;
}
int d[maxn], cur[maxn];
bool bfs(){
        for(int i=1; i<=N; i++) d[i] = 0, cur[i] = head[i];
        queue<int> q;
        q.push(S); d[S] = 1;
        while(!q.empty()){
                int x = q.front(); q.pop();
                for(int i=head[x]; i; i=Next[i]){
                        int y = ver[i];
                        if(edge[i] and !d[y]){
                                d[y] = d[x] + 1;
                                q.push(y);
                                if(y == T) return 1;
                        }
                }
        }
        return 0;
}
int dfs(int x, int flow){
        if(x == T) return flow;
        int rest = flow, k;
        for(int i=cur[x]; i and rest; i=Next[i]){
                cur[x] = i;
                if(edge[i] and d[ver[i]] == d[x] + 1){
                        k = dfs(ver[i], min(rest, edge[i]));
                        rest -= k, edge[i] -= k, edge[i^1] += k;
                }
        }
        return flow - rest;
}
signed main(){
        tot = 1;
        N = read(), M = read(), S = read(), T = read();
        for(int i=1; i<=M; i++){
                int x = read(), y = read(), z = read();
                add(x, y, z);
        }
        int maxflow = 0, flow = 0;
        while(bfs()) while(flow = dfs(S, INF)) maxflow += flow;
        cout<<maxflow<<endl;
        return 0;
}
```

2.3 费用流

```
//Edmonds-Karp
#include<bits/stdc++.h>
using namespace std;
const int maxn = 5005, maxm = 50005, INF = 1e9+7;
int head[maxn], ver[maxm<<1], edge[maxm<<1], Next[maxm<<1], cost[maxm<<1], tot;
void add(int x,int y,int z,int w)
{
    ver[++tot] = y, edge[tot] = z, cost[tot] = w, Next[tot] = head[x], head[x] = tot;
    ver[++tot] = x, edge[tot] = 0, cost[tot] = -w, Next[tot] = head[y], head[y] = tot;
}
int pre[maxn], incf[maxn], dis[maxn], s, t, maxflow, ans;
int n, m;
bool spfa()
{
    bool vis[maxn] = {};
    queue<int> Q;
    Q.push(s);incf[s] = INF;vis[s] = true;
    for(int i=1; i<=n; i++) dis[i] = INF;
    dis[s] = 0;
    while(!Q.empty())
    {
        int x = Q.front(); Q.pop();
        for(int i = head[x]; i; i=Next[i])
           if(edge[i])
           {
              int y = ver[i];
              if(dis[y] > dis[x]+cost[i])
              {
                        dis[y] = dis[x]+cost[i];
                        incf[y] = min(incf[x], edge[i]);
                        pre[y] = i;
                        if(!vis[y]) vis[y] = true, Q.push(y);
              }
           }
        vis[x] = false;
    }
    if(dis[t] == INF) return false;
    return true;
}
```

```
void update()
{
    int x = t;
    while(x!=s)
    {
        int i = pre[x];
        edge[i] -= incf[t];
        edge[i^1] += incf[t];
        x = ver[i^1];
    }
    maxflow+=incf[t];
    ans+=dis[t]*incf[t];
}
int main()
{
//    freopen("testdata.in","r",stdin);
        scanf("%d%d%d%d",&n,&m,&s,&t);
    tot = 1;
    for(int i=1; i<=m; i++)
    {
        int ui, vi, wi, fi;
        scanf("%d%d%d%d",&ui,&vi,&wi,&fi);
        add(ui, vi, wi, fi);
    }
    while(spfa()) update();
    printf("%d %d", maxflow, ans);
    return 0;
}
```

2.4 dijkstra

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
inline int read(){
        int w = 0, f = 1; char ch = getchar();
        while(ch < '0' or ch > '9') {if(ch == '-') f = -f; ch = getchar();}
        while(ch >= '0' and ch <= '9') w = w*10 + ch - '0', ch = getchar();
        return w*f;
}
const int maxn = 2e5 + 5, INF = 1e19 + 7ll;
int head[maxn], ver[maxn<<1], Next[maxn<<1], edge[maxn<<1], tot;
void add(int x, int y, int z){
        ver[++tot] = y, edge[tot] = z, Next[tot] = head[x], head[x] = tot;
}
int n, m, s, d[maxn];
priority_queue< pair<int, int> > q;
void dijkstra(){
        for(int i = 1; i <= n; i++) d[i] = INF;
        d[s] = 0;
        q.push(make_pair(0, s));
        while(!q.empty()){
                pair<int, int> u = q.top();
                q.pop();
                if(d[u.second] != -u.first) continue;
                for(int i = head[u.second]; i; i = Next[i]){
                        int v = ver[i];
                        if(d[v] > d[u.second] + edge[i]){
                                d[v] = d[u.second] + edge[i];
                                q.push(make_pair(-d[v], v));
                        }
                }
        }
}
signed main(){
        n = read(), m = read(), s = read();
        for(int i = 1; i <= m; i++){
                int x = read(), y = read(), z = read();
                add(x, y, z);
        }
        dijkstra();
        for(int i=1; i<=n; i++) printf("%lld ", d[i]);
        return 0;
}
```

3. 暴力

3.1 莫队

```
void move(int pos, int sign) {
  // update nowAns
}

void solve() {
  BLOCK_SIZE = int(ceil(pow(n, 0.5)));
  sort(querys, querys + m);
  for (int i = 0; i < m; ++i) {
    const query &q = querys[i];
    while (l > q.l) move(--l, 1);
    while (r < q.r) move(++r, 1);
    while (l < q.l) move(l++, -1);
    while (r > q.r) move(r--, -1);
    ans[q.id] = nowAns;
  }
}
```

3.2 Dsu On Tree

```cpp
#include<bits/stdc++.h>
using namespace std;
const int maxn = 5e5 + 5;
inline int read(){
        int w = 0, f = 1; char ch = getchar();
        while(ch < '0' or ch > '9') {if(ch == '-') f = -f; ch = getchar();}
        while(ch >= '0' and ch <= '9') w = w*10 + ch - '0', ch = getchar();
        return w*f;
}
vector< pair<int, int> > Q[maxn];
int N, M, f[maxn], head[maxn], ver[maxn<<1], Next[maxn<<1], tot;
void add(int x, int y){
        ver[++tot] = y, Next[tot] = head[x], head[x] = tot;
}
char str[maxn];
int siz[maxn], d[maxn], son[maxn];
void dfs1(int x, int deep){
        siz[x] = 1, d[x] = deep;
        for(int i=head[x]; i; i=Next[i]){
                int y = ver[i];
                dfs1(y, deep+1);
                siz[x] += siz[y];
                if(siz[son[x]] < siz[y]) son[x] = y;
        }
}
bool ans[maxn]; int Set[maxn];
void getans(int x, int p){
        Set[d[x]] ^= (1<<(str[x] - 'a'));
        for(int i=head[x]; i; i=Next[i]){
                int y = ver[i];
                if(y == p) continue;
                getans(y, p);
        }
}
void clear(int x){
        Set[d[x]] ^= (1<<(str[x] - 'a'));
        for(int i=head[x]; i; i=Next[i]){
                int y = ver[i];
                clear(y);
        }
}
void dfs2(int x){
        for(int i=head[x]; i; i=Next[i]){
                int y = ver[i];
                if(y == son[x]) continue;
                dfs2(y);//暴力统计每一个轻儿子答案
                clear(y);//清空轻儿子用的桶
        }
        if(son[x]) dfs2(son[x]);//暴力统计重儿子的答案
        getans(x, son[x]);//保留重儿子答案的同时把轻儿子内的答案也算上
        for(vector< pair<int, int> >::iterator it = Q[x].begin(); it != Q[x].end(); ++it){
                int S = Set[(*it).first];
                ans[(*it).second] = (S == (S&-S));//回答每一个离线下来的询问
        }
}
int main(){
        N = read(), M = read();
        for(int i=2; i<=N; i++) f[i] = read(), add(f[i], i);
        scanf("%s", str+1);
        for(int i=1; i<=M; i++){
                int a = read(), b = read();
                Q[a].push_back(make_pair(b, i));
        }
        dfs1(1, 1);
        dfs2(1);
        for(int i=1; i<=M; i++) printf(ans[i]?"Yes\n":"No\n");
        return 0;
}
```

4. 树上算法

4.1 LCA

```cpp
#include<bits/stdc++.h>
using namespace std;
const int maxn = 500005;
int head[maxn], ver[maxn<<1], Next[maxn<<1], tot;
void add(int x, int y){
        ver[++tot] = y, Next[tot] = head[x], head[x] = tot;
}
int N, M, S, f[maxn][20], deep[maxn]; bool vis[maxn];
queue<int> q;
void bfs(){
        q.push(S); deep[S] = 1;
        while(!q.empty()){
                int x = q.front();
                q.pop();
                for(int i=head[x]; i; i=Next[i]){
                        int y = ver[i];
                        if(deep[y]) continue;
                        f[y][0] = x; deep[y] = deep[x] + 1;
                        for(int j=1; j<20; j++) f[y][j] = f[f[y][j-1]][j-1];
                        q.push(y);
                }
        }
}
int lca(int x, int y){
        if(deep[x] > deep[y]) swap(x, y);
        for(int i=19; i>=0; i--) if(deep[f[y][i]] >= deep[x]) y = f[y][i];
        if(x == y) return x;
        for(int i=19; i>=0; i--) if(f[x][i] != f[y][i]) x = f[x][i], y = f[y][i];
        return f[x][0];
}
int main(){
        scanf("%d%d%d",&N,&M,&S);
        for(int i=1; i<N; i++){
                int x, y;
                scanf("%d%d",&x,&y);
                add(x, y); add(y, x);
        }
        bfs();
        for(int i=1; i<=M; i++){
                int a, b;
                scanf("%d%d",&a,&b);
                printf("%d\n", lca(a, b));
        }
        return 0;
}
```

5. 动态规划

5.1 数位dp

```
int dfs(int n, int pre, bool limit, bool lead){// 状态 n limit lead 基本上是固定的
        int sum = 0;
        if(n == 0) return 1;//基本不变
        if(!limit and !lead and f[n][pre] != -1) return f[n][pre];//基本不变
        for(int i=0; i <= (limit?digit[n]:9); i++)
            if(lead or (!lead and abs(i - pre) >= 2)) //不同题目的转移条件不同
                    sum += dfs(n-1, i, limit&(i == digit[n]), lead&(i == 0));
        if(!limit and !lead) f[n][pre] = sum;//基本不变
        return sum;
}
```

数据结构篇

1. ST表

```cpp
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1e5+5;
int rmq[20][maxn], Log2[maxn], n, m;
int main()
{
        scanf("%d%d",&n,&m);
        for(int i=1; i<=n; i++)
           scanf("%d",&rmq[0][i]);
        for(int i=1; i<20; i++)
           for(int j=1; j<=n; j++)
           {
                       int k=min(n, j+(1<<(i-1)));
                       rmq[i][j] = max(rmq[i-1][j], rmq[i-1][k]);
           }
        for(int i=1; i<=n; i++)
        {
                if(i > (1<<Log2[i-1])*2) Log2[i] = Log2[i-1]+1;
                else Log2[i] = Log2[i-1];
        }
        for(int i=1; i<=m; i++)
        {
                int x,y;
                scanf("%d%d",&x,&y);
                int len = Log2[y-x+1];
                printf("%d\n",max(rmq[len][x], rmq[len][y-(1<<len)+1]));
        }
        return 0;
}
```

2. Treap

```cpp
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1e5 + 5;
inline int read(){
        int w = 0, f = 1; char ch = getchar();
        while(ch < '0' or ch > '9') {if(ch == '-') f = -f; ch = getchar();}
        while(ch >= '0' and ch <= '9') w = w*10 + ch - '0', ch = getchar();
        return w*f;
}
struct Treap{
        int node[maxn][2], val[maxn], key[maxn], siz[maxn], cnt, root;
        Treap() {
                memset(node, 0, sizeof(node));
                memset(val, 0, sizeof(val));
                memset(key, 0, sizeof(key));
                memset(siz, 0, sizeof(siz));
                cnt = 0;
        }
        inline void update(int now){
                siz[now] = 1 + siz[node[now][0]] + siz[node[now][1]];
        }
        inline void split_val(int now, int k, int &x, int &y){
                if(!now) x = 0, y = 0;
                else {
                        if(val[now] <= k) x = now, split_val(node[now][1], k, node[now][1], y);
                        else y = now, split_val(node[now][0], k, x, node[now][0]);
                        update(now);
                }
        }
        inline void split_rak(int now, int k, int &x, int &y){
                if(!now) x = 0, y = 0;
                else if(k <= siz[node[now][0]]) y = now, split_rak(node[now][0], k, x, node[now][0
                else x = now, split_rak(node[now][1], k-siz[node[now][0]]-1, node[now][1], y), upd
        }
        inline int merge(int x, int y){
                if(!x or !y) return x|y;
                if(key[x] < key[y]){
                        node[y][0] = merge(x, node[y][0]);
                        update(y); return y;
                }
                else{
                        node[x][1] = merge(node[x][1], y);
                        update(x); return x;
                }
        }
        inline int new_node(int x){
                val[++cnt] = x, key[cnt] = rand(), siz[cnt] = 1;
                return cnt;
        }
        inline void insert(int x){
                int A, B;
                split_val(root, x, A, B);
                root = merge(merge(A, new_node(x)), B);
        }
        inline void del(int x){
                int A, B, C;
                split_val(root, x, A, B);
                split_val(A, x-1, A, C);
                C = merge(node[C][0], node[C][1]);
                root = merge(merge(A, C), B);
        }
        inline int getrak(int x){
                int A, B;
                split_val(root, x-1, A, B);
                int rak = siz[A] + 1;
                root = merge(A, B);
                return rak;
        }
        inline int findKth(int k, int now){
                while(1){
                        if(k <= siz[node[now][0]]) now = node[now][0];
                        else if(k == siz[node[now][0]] + 1) return val[now];
                        else k -= siz[node[now][0]] + 1, now = node[now][1];
                }
        }
        inline int pre(int x){
                int A, B;
                split_val(root, x-1, A, B);
                int Kth = findKth(siz[A], A);
                root = merge(A, B);
                return Kth;
        }
        inline int nxt(int x){
                int A, B;
                split_val(root, x, A, B);
                int Kth = findKth(1, B);
                root = merge(A, B);
                return Kth;
        }
}FHQ_treap;
int main(){
        int N = read();
        while(N--){
                int opt = read(), x = read();
                switch (opt){
                        case 1: FHQ_treap.insert(x); break;
                        case 2: FHQ_treap.del(x); break;
                        case 3: printf("%d\n", FHQ_treap.getrak(x)); break;
                        case 4: printf("%d\n", FHQ_treap.findKth(x, FHQ_treap.root)); break;
                        case 5: printf("%d\n", FHQ_treap.pre(x)); break;
                        case 6: printf("%d\n", FHQ_treap.nxt(x)); break;
                }
        }
        return 0;
}
```

指针版

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

struct Node {
        Node* ch[2];
        int val, rank;
        int rep_cnt;
        int siz;

        Node(int val): val(val), rep_cnt(1), siz(1) {
                ch[0] = ch[1] = nullptr;
                rank = rand();
        }

        void upd_siz() {
                siz = rep_cnt;
                if (ch[0] != nullptr) siz += ch[0]->siz;
                if (ch[1] != nullptr) siz += ch[1]->siz;
        }
};

class Treap {
private:
        Node* root;

        const static int NIL = -1;  // 用于表示查询的值不存在

        enum rot_type { LF = 1, RT = 0 };

        int q_prev_tmp = 0, q_nex_tmp = 0;

        void _rotate(Node*& cur, rot_type dir) {  // 0为右旋，1为左旋
                Node* tmp = cur->ch[dir];
                cur->ch[dir] = tmp->ch[!dir];
                tmp->ch[!dir] = cur;
                cur->upd_siz(), tmp->upd_siz();
                cur = tmp;
        }
```

```
    void _insert(Node*& cur, int val) {
        if (cur == nullptr) {
            cur = new Node(val);
            return;
        }
        else if (val == cur->val) {
            cur->rep_cnt++;
            cur->siz++;
        }
        else if (val < cur->val) {
            _insert(cur->ch[0], val);
            if (cur->ch[0]->rank < cur->rank) {
                _rotate(cur, RT);
            }
            cur->upd_siz();
        }
        else {
            _insert(cur->ch[1], val);
            if (cur->ch[1]->rank < cur->rank) {
                _rotate(cur, LF);
            }
            cur->upd_siz();
        }
    }

    void _del(Node*& cur, int val) {
        if (val > cur->val) {
            _del(cur->ch[1], val);
            cur->upd_siz();
        }
        else if (val < cur->val) {
            _del(cur->ch[0], val);
            cur->upd_siz();
        }
        else {
            if (cur->rep_cnt > 1) {
                cur->rep_cnt--, cur->siz--;
                return;
            }
```

```
                uint8_t state = 0;
                state |= (cur->ch[0] != nullptr);
                state |= ((cur->ch[1] != nullptr) << 1);
                // 00都无，01有左无右，10，无左有右，11都有
                Node* tmp = cur;
                switch (state) {
                case 0:
                        delete cur;
                        cur = nullptr;
                        break;
                case 1:  // 有左无右
                        cur = tmp->ch[0];
                        delete tmp;
                        break;
                case 2:  // 有右无左
                        cur = tmp->ch[1];
                        delete tmp;
                        break;
                case 3:
                        rot_type dir = cur->ch[0]->rank < cur->ch[1]->rank ? RT : LF;
                        _rotate(cur, dir);
                        _del(cur->ch[!dir], val);
                        cur->upd_siz();
                        break;
                }
        }
}

int _query_rank(Node* cur, int val) {
        int less_siz = cur->ch[0] == nullptr ? 0 : cur->ch[0]->siz;
        if (val == cur->val)
                return less_siz + 1;
        else if (val < cur->val) {
                if (cur->ch[0] != nullptr)
                        return _query_rank(cur->ch[0], val);
                else
                        return 1;
        }
        else {
```

```
                        if (cur->ch[1] != nullptr)
                                return less_siz + cur->rep_cnt + _query_rank(cur->ch[1], val);
                        else
                                return cur->siz + 1;
                }
        }

        int _query_val(Node* cur, int rank) {
                int less_siz = cur->ch[0] == nullptr ? 0 : cur->ch[0]->siz;
                if (rank <= less_siz)
                        return _query_val(cur->ch[0], rank);
                else if (rank <= less_siz + cur->rep_cnt)
                        return cur->val;
                else
                        return _query_val(cur->ch[1], rank - less_siz - cur->rep_cnt);
        }

        int _query_prev(Node* cur, int val) {
                if (val <= cur->val) {
                        if (cur->ch[0] != nullptr) return _query_prev(cur->ch[0], val);
                }
                else {
                        q_prev_tmp = cur->val;
                        if (cur->ch[1] != nullptr) _query_prev(cur->ch[1], val);
                        return q_prev_tmp;
                }
                return NIL;
        }

        int _query_nex(Node* cur, int val) {
                if (val >= cur->val) {
                        if (cur->ch[1] != nullptr) return _query_nex(cur->ch[1], val);
                }
                else {
                        q_nex_tmp = cur->val;
                        if (cur->ch[0] != nullptr) _query_nex(cur->ch[0], val);
                        return q_nex_tmp;
                }
                return NIL;
        }

public:
        void insert(int val) { _insert(root, val); }

        void del(int val) { _del(root, val); }

        int query_rank(int val) { return _query_rank(root, val); }

        int query_val(int rank) { return _query_val(root, rank); }

        int query_prev(int val) { return _query_prev(root, val); }

        int query_nex(int val) { return _query_nex(root, val); }
};

Treap tr;

int main() {
        srand(0);
        int t;
        scanf("%d", &t);
        while (t--) {
                int mode;
                int num;
                scanf("%d%d", &mode, &num);
                switch (mode) {
                case 1:
                        tr.insert(num);
                        break;
                case 2:
                        tr.del(num);
                        break;
                case 3:
                        printf("%d\n", tr.query_rank(num));
                        break;
                case 4:
                        printf("%d\n", tr.query_val(num));
                        break;
                case 5:
                        printf("%d\n", tr.query_prev(num));
                        break;
                case 6:
                        printf("%d\n", tr.query_nex(num));
                        break;
                }
        }
}
```

3. 主席树

静态区间第k小

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 2e5 + 5;
struct node{
        int ls, rs, cnt;
}t[maxn*20];
int a[maxn], N, M, root[maxn], tot, Map[maxn];
pair<int, int> key[maxn];
int build(int l, int r){
        int p = ++tot;
        if(l == r) {
                t[p].cnt = 0;
                return p;
        }
        int mid = (l+r)>>1;
        t[p].ls = build(l, mid);
        t[p].rs = build(mid+1, r);
        t[p].cnt = t[t[p].ls].cnt + t[t[p].rs].cnt;
        return p;
}
int insert(int now, int l, int r, int x, int val){
        int p = ++tot;
        t[p] = t[now];
        if(l == r){
                t[p].cnt += val;
                return p;
        }
        int mid = (l+r)>>1;
        if(x <= mid) t[p].ls = insert(t[now].ls, l, mid, x, val);
        else t[p].rs = insert(t[now].rs, mid+1, r, x, val);
        t[p].cnt = t[t[p].ls].cnt + t[t[p].rs].cnt;
        return p;
}
int ask(int p, int q, int l, int r, int k){
        if(l == r) return l;
        int mid = (l+r)>>1;
        int num = t[t[p].ls].cnt - t[t[q].ls].cnt;
        if(k <= num) return ask(t[p].ls, t[q].ls, l, mid, k);
        else return ask(t[p].rs, t[q].rs, mid+1, r, k - num);
}
int Find(int x){
        return Map[x];
}
int main(){
        scanf("%d%d", &N, &M);
        for(int i=1; i<=N; i++) scanf("%d", &a[i]), key[i] = make_pair(a[i], i);
        sort(key+1, key+N+1);
        int rak = 0;
        for(int i=1; i<=N; i++){
                if(key[i].first != key[i-1].first or i == 1) rak ++;
                a[key[i].second] = rak; Map[rak] = key[i].first;
        }
        root[0] = build(1, rak);
        for(int i=1; i<=N; i++){
                root[i] = insert(root[i-1], 1, rak, a[i], 1);
        }
        for(int i=1; i<=M; i++){
                int l, r, k;
                scanf("%d%d%d", &l, &r, &k);
                printf("%d\n", Find(ask(root[r], root[l-1], 1, rak, k)));
        }
        return 0;
}
```

4. 重链剖分

```
//树链剖分 2020.8.19
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1e5 + 5;
int id[maxn], f[maxn], d[maxn], siz[maxn], son[maxn], top[maxn], val[maxn], cnt;
int head[maxn], Next[maxn<<1], ver[maxn<<1], tot;
struct node{
        int l, r;
        int sum, add;
}t[maxn<<2];
int N, M, root, mod, a[maxn];
//以下为区间修改线段树
void build(int p, int l, int r){
        t[p].l = l, t[p].r = r;
        if(l == r){
                t[p].sum = val[l];
                return ;
        }
        int mid = (l+r)>>1;
        build(p<<1, l, mid);
        build(p<<1|1, mid+1, r);
        t[p].sum = (t[p<<1].sum + t[p<<1|1].sum)%mod;
}
void spread(int p){
        if(t[p].add){
                t[p<<1].sum = (t[p<<1].sum + t[p].add*(t[p<<1].r-t[p<<1].l+1)%mod)%mod;
                t[p<<1|1].sum = (t[p<<1|1].sum + t[p].add*(t[p<<1|1].r-t[p<<1|1].l+1)%mod)%mod;
                t[p<<1].add = (t[p<<1].add + t[p].add)%mod;
                t[p<<1|1].add = (t[p<<1|1].add + t[p].add)%mod;
                t[p].add = 0;
        }
}
void change(int p, int l, int r, int x){
        if(t[p].l >= l and t[p].r <= r){
                t[p].sum = (t[p].sum + x*(t[p].r-t[p].l+1)%mod)%mod;
                t[p].add = (t[p].add + x)%mod;
                return ;
        }
        spread(p);
        int mid = (t[p].l + t[p].r)>>1;
        if(l <= mid) change(p<<1, l, r, x);
        if(r > mid) change(p<<1|1, l, r, x);
        t[p].sum = (t[p<<1].sum + t[p<<1|1].sum)%mod;
}
int ask(int p, int l, int r){
        if(t[p].l >= l and t[p].r <= r) return t[p].sum;
        spread(p);
        int mid = (t[p].l + t[p].r)>>1;
        int v = 0;
        if(l <= mid) v = (v + ask(p<<1, l, r))%mod;
        if(r > mid) v = (v + ask(p<<1|1, l, r))%mod;
        return v;
}
//邻接链表建边
void add(int x, int y){
        ver[++tot] = y, Next[tot] = head[x], head[x] = tot;
}
//处理出f,d,siz,son数组
void dfs1(int x, int fa, int deep){
        f[x] = fa, d[x] = deep, siz[x] = 1;
        for(int i=head[x]; i; i=Next[i]){
                int y = ver[i];
                if(y == fa) continue;
                dfs1(y, x, deep+1);
                siz[x] += siz[y];
                if(siz[y] > siz[son[x]] or son[x] == 0) son[x] = y;
        }
        return ;
}
//处理出top,id数组
void dfs2(int x, int t){
        top[x] = t, id[x] = ++cnt, val[cnt] = a[x];
        if(!son[x]) return ;
        dfs2(son[x], t);
        for(int i=head[x]; i; i=Next[i]){
                int y = ver[i];
                if(y == son[x] or y == f[x]) continue;
                dfs2(y, y);
        }
        return ;
}
void updRange(int x, int y, int z){
        z %= mod;
        while(top[x] != top[y]){//如果x,y不在一条重链上
                if(d[top[x]] < d[top[y]]) swap(x, y);//保证x所在的重链始终比y更深
                change(1, id[top[x]], id[x], z);//区间修改链
                x = f[top[x]];//x上升到重链顶端的父节点
        }
        if(d[x] > d[y]) swap(x, y);//保证x始终比y浅(即保证x的id始终比y小，x在连续序列上始终在y前)
        change(1, id[x], id[y], z);//区间修改
}
//同上
int qRange(int x, int y){
        int ans = 0;
        while(top[x] != top[y]){
                if(d[top[x]] < d[top[y]]) swap(x, y);
                ans = (ans + ask(1, id[top[x]], id[x]))%mod;
                x = f[top[x]];
        }
        if(d[x] > d[y]) swap(x, y);
        ans = (ans + ask(1, id[x], id[y]))%mod;
        return ans;
}
void updSon(int x, int z){
        change(1, id[x], id[x] + siz[x] - 1, z);//一个子树的编号一定是连续的，直接区间修改即可
}
int qSon(int x){
        return ask(1, id[x], id[x] + siz[x] - 1);//同上
}
int main(){
        scanf("%d%d%d%d", &N,&M,&root,&mod);
        for(int i=1; i<=N; i++) scanf("%d", &a[i]);
        for(int i=1; i<N; i++){
                int x, y;
                scanf("%d%d", &x, &y);
                add(x, y); add(y, x);
        }
        dfs1(root, 0, 1);
        dfs2(root, root);
        build(1, 1, N);
        for(int i=1; i<=M; i++){
                int opt, x, y, z;
                scanf("%d", &opt);
                if(opt == 1){
                        scanf("%d%d%d", &x, &y, &z);
                        updRange(x, y, z);
                }
                else if(opt == 2){
                        scanf("%d%d", &x, &y);
                        printf("%d\n", qRange(x, y));
                }
                else if(opt == 3){
                        scanf("%d%d", &x, &z);
                        updSon(x, z);
                }
                else if(opt == 4){
                        scanf("%d", &x);
                        printf("%d\n", qSon(x));
                }
        }
        return 0;
}
```

数学篇

1. 线性代数

1.1 高斯消元

```cpp
#include <bits/stdc++.h>
using namespace std;
inline int read() {
    int w = 0, f = 1; char ch = getchar();
    while(ch < '0' or ch > '9') {if(ch == '-') f = -f; ch = getchar();}
    while(ch >= '0' and ch <= '9') w = w*10 + ch - '0', ch = getchar();
    return w*f;
}
const int maxn = 105;
double a[maxn][maxn];
int N;
int main() {
    N = read();
    for(int i = 1; i <= N; i ++)
        for(int j = 1; j <= N + 1; j ++)
            scanf("%lf", &a[i][j]);
    for(int i = 1; i <= N; i ++) {
        int maxi = i;
        for(int j = i + 1; j <= N; j ++)
            if(fabs(a[j][i]) > fabs(a[maxi][i]))
                maxi = j;
        if(i != maxi) swap(a[i], a[maxi]);
        if(!a[i][i]) return printf("No Solution"), 0;
        for(int j = 1; j <= N; j ++) {
            if(j ^ i) {
                for(int k = N + 1; k >= i; k --)
                    a[j][k] -= a[j][i] / a[i][i] * a[i][k];
            }
        }
    }
    for(int i = 1; i <= N; i ++) printf("%.2lf\n", a[i][N + 1] / a[i][i]);
    return 0;
}
```

1.2 行列式

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
inline int read()
{
    int w = 0, f = 1;
    char ch = getchar();
    while (ch < '0' or ch > '9')
    {
        if (ch == '-')
            f = -f;
        ch = getchar();
    }
    while (ch >= '0' and ch <= '9')
        w = w * 10 + ch - '0', ch = getchar();
    return w * f;
}
const int maxn = 606;
int det(int a[maxn][maxn], int N, int mod)
{
    int opt = 1, ans = 1;
    for (int i = 1; i <= N; i++)
    {
        for (int j = i + 1; j <= N; j++)
        {
            while (a[i][i])
            {
                int l = a[j][i] / a[i][i];
                for (int k = i; k <= N; k++)
                    a[j][k] = (a[j][k] - l * a[i][k] % mod) % mod;
                swap(a[i], a[j]), opt ^= 1;
            }
            swap(a[i], a[j]), opt ^= 1;
        }
        ans = ans * a[i][i] % mod;
    }
    if (!opt)
        ans = -ans;
    return (ans + mod) % mod;
}
int N, a[maxn][maxn], mod;
signed main()
{
    N = read(), mod = read();
    for (int i = 1; i <= N; i++)
        for (int j = 1; j <= N; j++)
            a[i][j] = read();
    printf("%lld", det(a, N, mod));
    return 0;
}
```

1.3 矩阵树定理

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int mod = 1e9 + 7;
inline int read()
{
    int w = 0, f = 1;
    char ch = getchar();
    while (ch < '0' or ch > '9')
    {
        if (ch == '-')
            f = -f;
        ch = getchar();
    }
    while (ch >= '0' and ch <= '9')
        w = w * 10 + ch - '0', ch = getchar();
    return w * f;
}
const int maxn = 606;
int det(int a[maxn][maxn], int N)
{
    int opt = 1, ans = 1;
    for (int i = 1; i <= N; i++)
    {
        for (int j = i + 1; j <= N; j++)
        {
            while (a[i][i])
            {
                int l = a[j][i] / a[i][i];
                for (int k = i; k <= N; k++)
                    a[j][k] = (a[j][k] - l * a[i][k] % mod) % mod;
                swap(a[i], a[j]), opt ^= 1;
            }
            swap(a[i], a[j]), opt ^= 1;
        }
        ans = ans * a[i][i] % mod;
    }
    if (!opt)
        ans = -ans;
    return (ans + mod) % mod;
}
int N, a[maxn][maxn], b[maxn][maxn];
signed main()
{
    int N, M, t;
    N = read(), M = read(), t = read();
    for (int i = 1; i <= M; i++)
    {
        int u = read(), v = read(), w = read();
        if (t)
        {
            a[v][v] += w;
            a[u][v] -= w;
            a[u][v] += mod;
            a[v][v] %= mod, a[u][v] %= mod;
        }
        else
        {
            a[u][u] += w, a[v][v] += w;
            a[u][v] -= w, a[u][v] += mod, a[v][u] -= w, a[v][u] += mod;
            a[u][u] %= mod, a[v][v] %= mod;
            a[u][v] %= mod, a[v][u] %= mod;
        }
    }
    for (int i = 1; i < N; i++)
        for (int j = 1; j < N; j++)
        {
            b[i][j] = a[i + 1][j + 1];
        }
    cout << det(b, N - 1);
    return 0;
}
```

1.4 矩阵快速幂

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
inline int read()
{
    int w = 0, f = 1;
    char ch = getchar();
    while (ch < '0' or ch > '9')
    {
        if (ch == '-')
            f = -f;
        ch = getchar();
    }
    while (ch >= '0' and ch <= '9')
        w = w * 10 + ch - '0', ch = getchar();
    return w * f;
}
const int maxn = 105, mod = 1e9 + 7;
struct Matrix
{
    int a[maxn][maxn], n;
    void clear(int x)
    {
        n = x;
        for (int i = 1; i <= n; i++)
            a[i][i] = 1;
    }
    void init(int x)
    {
        n = x;
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++)
                a[i][j] = 0;
    }
    Matrix operator*(const Matrix &x) const
    {
        Matrix c;
        c.init(n);
        for (int k = 1; k <= n; k++)
```

```
            for (int i = 1; i <= n; i++)
                for (int j = 1; j <= n; j++)
                    (c.a[i][j] += a[i][k] * x.a[k][j] % mod) %= mod;
        return c;
    }
    void debug()
    {
        for (int i = 1; i <= n; i++)
        {
            for (int j = 1; j <= n; j++)
                cout << a[i][j] << " ";
            cout << endl;
        }
    }
};
int n, k;
Matrix qpow(Matrix x, int y)
{
    Matrix cnt, basic = x;
    cnt.clear(n);
    while (y)
    {
        if (y & 1)
            cnt = cnt * basic;
        basic = basic * basic, y >>= 1;
    }
    return cnt;
}
signed main()
{
    n = read(), k = read();
    Matrix a;
    a.clear(n);
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
        {
            a.a[i][j] = read();
        }
    qpow(a, k).debug();
    return 0;
}
```

2. 数论

2.1 扩展欧几里得

```cpp
#include<bits/stdc++.h>
using namespace std;
inline long long read(){
        long long w = 0ll, f = 1ll; char ch = getchar();
        while(ch < '0' or ch > '9') {if(ch == '-') f = -f; ch = getchar();}
        while(ch >= '0' and ch <= '9') w = w*10ll + ch - '0', ch = getchar();
        return w*f;
}
long long exgcd(long long a, long long b, long long c, long long &x, long long &y){
        if(b == 0){
                x = c/a, y = 0ll;
                return a;
        }
        long long t;
        long long d = exgcd(b, a%b, c, t, x);
        y = t - a/b*x;
        return d;
}
int main(){
        int T = read();
        while(T--){
                long long a = read(), b = read(), c = read(), x, y, GCD;
                GCD = exgcd(a, b, c, x, y);
                if(c%GCD != 0) printf("-1\n");
                else{
                        long long A = a/GCD, B = b/GCD, C = c/GCD;
                        x = (x%B + B)%B;
                        if(x == 0) x += B;
                        y = (C - A*x)/B;
                        if(y <= 0ll){
                                printf("%lld ", x);
                                y = (y%A + A)%A; if(y == 0ll) y += A;
                                printf("%lld\n", y);
                        }
                        else{
                                long long cnt = y/A + 1;
                                if(y%A == 0ll) cnt --;
                                long long miniy = (y%A + A)%A;
                                if(miniy == 0ll) miniy += A;
```

```
                                            printf("%lld %lld %lld %lld %lld\n", cnt, x, miniy, (C - B*miniy)/
                                  }
                        }
              }
              return 0;
    }
```

2.2 扩展中国剩余定理

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int maxn = 1e5 + 5;
inline int read(){
        int w = 0, f = 1; char ch = getchar();
        while(ch < '0' or ch > '9') {if(ch == '-') f = -f; ch = getchar();}
        while(ch >= '0' and ch <= '9') w = w*10 + ch - '0', ch = getchar();
        return w*f;
}
int N, M[maxn], m, a[maxn];
pair<int, int> exgcd(int a, int b){
        if(b == 0) return make_pair(1, 0);
        pair<int, int> tmp = exgcd(b, a%b);
        return make_pair(tmp.second, tmp.first - a/b*tmp.second);
}
int gcd(int x, int y){
        if(y == 0) return x;
        return gcd(y, x%y);
}
int lcm(int x, int y){
        return x/gcd(x, y)*y;
}
int ksc(int x, int y, int mod){
        y = (y%mod + mod)%mod;
        int cnt = 0, basic = x;
        while(y){
                if(y&1) cnt = (cnt + basic)%mod;
                basic = (basic + basic)%mod, y >>= 1;
        }
        return cnt;
}
signed main(){
        int N = read(), x = 0;
        m = read(), x = read();
        for(int i=2; i<=N; i++){
                M[i] = read(), a[i] = read();
                int tmp = ((a[i] - x)%M[i] + M[i])%M[i];
                int GCD = gcd(m, M[i]);
                int k = ksc(exgcd(m, M[i]).first, tmp/GCD, M[i]/GCD);
                x = x + k*m; m = lcm(m, M[i]); x %= m;
        }
        cout<<(x%m + m)%m<<endl;
        return 0;
}
```

1.3 扩展欧拉定理

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int maxn = 1e8 + 5;
inline int read(){
        int w = 0, f = 1; char ch = getchar();
        while(ch < '0' or ch > '9') {if(ch == '-') f = -f; ch = getchar();}
        while(ch >= '0' and ch <= '9') w = w*10 + ch - '0', ch = getchar();
        return w*f;
}
int a, m, b;
int quickpow(int x, int y){
        int cnt = 1, basic = x%m;
        while(y){
                if(y&1) cnt = cnt*basic%m;
                basic = basic*basic%m, y >>= 1;
        }
        return cnt;
}
signed main(){
        a = read(), m = read();
        int x = m, phi = 1;
        for(int i=2; i*i <= m; i++){
                if(x%i) continue;
                phi = phi*(i - 1); x /= i;
                while(x%i == 0) phi *= i, x /= i;
        }
        if(x > 1) phi *= x-1;
        // cout<<phi<<endl;
        int w = 0; char ch = getchar();
        while(ch < '0' or ch > '9') ch = getchar();
        bool flag = 0;
        while(ch >= '0' and ch <= '9'){
                w = w*10 + ch - '0', ch = getchar();
                if(w >= phi) flag = 1, w %= phi;
        }
        if(flag) w += phi;
        // cout<<w<<endl;
        cout<<quickpow(a, w);
        return 0;
}
```

1.4 Lucas定理

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
const int maxn = 1e5 + 5;
inline int read(){
        int w = 0, f = 1; char ch = getchar();
        while(ch < '0' or ch > '9') {if(ch == '-') f = -f; ch = getchar();}
        while(ch >= '0' and ch <= '9') w = w*10 + ch - '0', ch = getchar();
        return w*f;
}
int Fac[maxn<<1], FacInverse[maxn<<1], mod, N, M;
int quickpow(int x, int y){
        x = (x%mod + mod)%mod;
        int cnt = 1, basic=  x;
        while(y){
                if(y&1) cnt = cnt*basic%mod;
                basic = basic*basic%mod, y >>= 1;
        }
        return cnt;
}
int C(int x, int y){
        if(x < y) return 0ll;
        return Fac[x]*FacInverse[x-y]%mod*FacInverse[y]%mod;
}
int lucas(int x, int y){
        if(y == 0) return 1;
        return lucas(x/mod, y/mod) * C(x%mod, y%mod) % mod;
}
signed main(){
        int T = read();
        while(T --){
                N = read(), M = read(), mod = read();
                Fac[0] = 1;
                for(int i=1; i<=mod-1; i++) Fac[i] = Fac[i-1]*i%mod;
                FacInverse[mod-1] = quickpow(Fac[mod-1], mod - 2);
                for(int i=mod-2; i>=0; i--) FacInverse[i] = FacInverse[i+1]*(i+1)%mod;
                printf("%lld\n", lucas(N+M, N));
        }
        return 0;
}
```

1.5 Polya定理

n个点n中颜色的环求本质不同的染色方案

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll mod = 1e9 + 7;
inline ll quickpow(ll x, ll y){
        ll basic = x, cnt = 1ll;
        while(y){
                if(y&1) cnt = cnt*basic%mod;
                basic = basic*basic%mod, y >>= 1;
        }
        return cnt;
}
inline ll phi(ll N){
        ll ans = N, x = N;
        for(ll i=2; i*i <= N; i++){
                if(x%i == 0){
                        while(x%i == 0) x /= i;
                        ans -= ans/i;
                }
        }
        if(x != 1) ans -= ans/x;
        return ans;
}
int main(){
        int T;
        scanf("%d", &T);
        while(T--){
                int N; ll ans = 0;
                scanf("%d", &N);
                for(int d=1; d*d <= N; d++){
                        if(N%d == 0){
                                ans += phi(N/d)*quickpow(N, d)%mod; ans %= mod;
                                if(d*d != N) ans += phi(d)*quickpow(N, N/d)%mod; ans %= mod;
                        }
                }
                printf("%lld\n", ans*quickpow(N, mod-2)%mod);
        }
        return 0;
}
```

1.6 莫比乌斯反演

gcd(x, y) 为质数

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int maxn = 1e7 + 5;
inline int read(){
	int w = 0, f = 1; char ch = getchar();
	while(ch < '0' or ch > '9') {if(ch == '-') f = -f; ch = getchar();}
	while(ch >= '0' and ch <= '9') w = w*10 + ch - '0', ch = getchar();
	return w*f;
}
int mu[maxn], prime[maxn/10], cnt, f[maxn], sum[maxn]; bool vis[maxn];
void init(){
	mu[1] = 1;
	for(int i=2; i<=10000000; i++){
		if(!vis[i]) prime[++cnt] = i, mu[i] = -1;
		for(int j=1; j<=cnt and i*prime[j] <= 10000000; j++){
			vis[i*prime[j]] = 1;
			if(i%prime[j] == 0){
				mu[i*prime[j]] = 0;
				break;
			}
			mu[i*prime[j]] = -mu[i];
		}
	}
	for(int i=1; i<=cnt; i++)
		for(int j=1; j*prime[i] <= 10000000; j++) f[j*prime[i]] += mu[j];
	for(int i=1; i<=10000000; i++) sum[i] = sum[i-1] + f[i];
}
signed main(){
	init();
	int T = read();
	while(T--){
		int N = read(), M = read(), ans = 0;
		if(N > M) swap(N, M);
		for(int l = 1, r = 0; l <= N; l = r+1){
			r = min(N/(N/l), M/(M/l));
			ans += (sum[r] - sum[l-1])*(N/l)*(M/l);
		}
		printf("%lld\n", ans);
```

```
        }
        return 0;
}
```

1.7 FFT

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
const int maxn = 4e6 + 5, mod = 998244353;
inline double read(){
        int w = 0, f = 1; char ch = getchar();
        while(ch < '0' or ch > '9') {if(ch == '-') f = -f; ch = getchar();}
        while(ch >= '0' and ch <= '9') w = w*10 + ch - '0', ch = getchar();
        return (double)w*f;
}
int quickpow(int x, int y = mod - 2){
        y = (y%(mod-1) + mod-1)%(mod-1);
        x = (x%mod + mod)%mod;
        int cnt = 1, basic = x;
        while(y){
                if(y&1) cnt = cnt*basic%mod;
                basic = basic*basic%mod, y >>= 1;
        }
        return cnt;
}
int g = 3, Gi;
struct FastFourierTransform{
        int omega[maxn], omegaInverse[maxn], reserve[maxn];
        void init(const int& N){
                for(int i=0; i<N; i++){
                        omega[i] = quickpow(3, ((mod - 1)/N)*i);
                        omegaInverse[i] = quickpow(omega[i], mod - 2);
                        reserve[i] = (reserve[i>>1]>>1)|((i&1)?N>>1:0);
                }
        }
        void transform(int *a, const int& N, const int* omega){
                for(int i=0; i<N; i++) if(i < reserve[i]) swap(a[i], a[reserve[i]]);
                for(int l=2; l<=N; l<<=1){
                        int m = l/2;
                        for(int j = 0; j < N; j += l){
                                for(int i=0; i<m; i++) {
                                        int t = omega[N/l*i] * a[j+m+i]%mod;
                                        a[j+m+i] = (a[j + i] - t + mod)%mod;
                                        a[j + i] = (a[j + i] + t)%mod;
                                }
                        }
                }
        }
        void dft(int *a, const int& N){
                transform(a, N, omega);
        }
        void idft(int *a, const int& N){
                transform(a, N, omegaInverse);
        }
}ntt;
int F[maxn], G[maxn], Ans[maxn];
signed main(){
        int N, M;
        N = read(), M = read();
        Gi = quickpow(g);
        for(int i=0; i<=N; i++) scanf("%d", &F[i]);
        for(int j=0; j<=M; j++) scanf("%d", &G[j]);
        int limit = 1;
        while(limit <= N+M) limit <<= 1;
        ntt.init(limit);
        ntt.dft(F, limit);
        ntt.dft(G, limit);
        for(int i=0; i< limit; i++) Ans[i] = F[i]*G[i]%mod;
        ntt.idft(Ans, limit); int invl = quickpow(limit, mod - 2);
        for(int i=0; i<=N+M; i++) printf("%d ", Ans[i]*invl%mod);
        return 0;
}
```