ICPC Templates

Leachim

August 25, 2020

Contents

1 图论 **3**
1.1 最小生成树 Kruskal . . . 3
1.2 最小生成树 Prim . . . 4
1.3 最短路 Djikstra . . . 5
1.4 最短路 SPFA . . . 7
1.5 最近公共祖先 LCA_Doubling . . . 8
1.6 最近公共祖先 LCA_Tarjan . . . 9
1.7 判断负环 SPFA_Negtive_Cycle . . . 11
1.8 拓扑排序 Topological_Sort_Khan . . . 12

2 多项式 **15**
2.1 FFT 字符串匹配 String_Match_FFT . . . 15
2.2 FFT 递归 Fast_Fourier_Transform_Cooley-Tukey_Recursion . . . 16
2.3 FFT 递推 Fast_Fourier_Transform_Cooley-Tukey_Iteration . . . 17
2.4 NTT 递推 Number_Theoretic_Transforms . . . 19
2.5 多项式求逆 Polynomial_Inverse . . . 20

3 字符串 **23**
3.1 字符串哈希 String_Hash . . . 23
3.2 马拉车 Manacher . . . 23
3.3 字符串匹配 KMP . . . 24
3.4 AC 自动机 AC-Automaton . . . 25
3.5 后缀数组 Suffix_Array . . . 26
3.6 后缀自动机 Suffix-Automaton . . . 27
3.7 广义后缀自动机 General_Suffix-Automaton . . . 29

4 数据结构 **32**
4.1 并查集 Union_Find . . . 32
4.2 分块 Block_1 . . . 32
4.3 分块 Block_2 . . . 33
4.4 分块 Block_4 . . . 35
4.5 树状数组 Binary_Indexed_Tree . . . 36
4.6 树状数组 2D_Binary_Indexed_Tree . . . 37
4.7 线段树 Segment_Tree . . . 38
4.8 线段树 Segment_Tree_Multiply . . . 40
4.9 扫描线 Scanline . . . 42
4.10 zkw 线段树 ZKW_Segment_Tree . . . 44

4.11 李超线段树 Li-Chao_Segment_Tree 45
4.12 可并堆左偏树 Leftist_Tree 47
4.13 Splay 树 Splay_Tree 48
4.14 Splay 树 Splay_Tree_Flip 51
4.15 Splay 树 Splay_Tree_Dye&Flip 53

5 数论 **58**
5.1 乘法逆元 Multiplicative_Inverse_Modulo 58
5.2 卢卡斯 Lucas 58
5.3 拓展欧几里得 Exgcd 59
5.4 拓展欧拉定理 Ex_Euler_Theorem-Automaton 60
5.5 欧拉筛 Eular_Sieve 61
5.6 杜教筛 Dujiao_Sieve 61
5.7 求原根 Get_Primitive_Root 63
5.8 贝祖引理 Bezout_Lemma 64
5.9 除法分块 Block_Division 64

6 网络流 **66**
6.1 最大费用流 Minimum-Cost_Flow_Edmonds-Karp 66
6.2 最大流 Maximum_Flow_Edmonds-Karp 67
6.3 最大流 Maximum_Flow_Dinic 69
6.4 二分题最大匹配 Bipartite_Graph_Maximum_Matching_Dinic 71
6.5 二分图最大匹配 Bipartite_Graph_Maximum_Matching_Hungarian 73
6.6 二分图最大匹配 Bipartite_Graph_Maximum_Matching_Hopcroft-Karp 74

7 计算几何 **76**
7.1 计算几何 Compuational_Geometry 76

1 图论

1.1 最小生成树 Kruskal

```cpp
#include <cstdio>
#include <algorithm>

#define MAXN 5005
#define MAXM 200005
#define _for(i,a,b) for(int i=(a);i<=(b);i++)

using namespace std;

int read(){
    int ng=0,x=0;
    char ch=getchar();
    for(;ch<'0' || ch>'9';ch=getchar()) ng|=ch=='-';
    for(;ch>='0' && ch<='9';ch=getchar()) x=(x<<3)+(x<<1)+ch-'0';
    return ng?-x:x;
}

struct road
{
    int f,g,w;
}r[MAXM];

struct cmpf
{
    bool operator() (const road& a, const road& b) const {
        return a.w<b.w;
    }
};

int n,m,ans=0,fa[MAXN];

int find(int x){
    return fa[x]==x?x:fa[x]=find(fa[x]);
}

void kruskal(){
    int rx,ry;

    sort(r+1, r+m+1, cmpf());
    _for(i,1,m){
        rx=find(r[i].f);
        ry=find(r[i].g);
        if(rx==ry)
            continue;
        fa[rx]=ry;
        ans+=r[i].w;
    }
}

```

```
int main(){
    n=read();m=read();
    _for(i,1,m){
        r[i].f=read();
        r[i].g=read();
        r[i].w=read();
    }
    _for(i,1,n)
        fa[i]=i;
    kruskal();
    _for(i,2,n){
        if(find(i)!=find(i-1)){
            puts("orz");
            return 0;
        }
    }
    printf("%d", ans);
}
```

1.2 最小生成树 Prim

```
#include <cstdio>
#include <cstdlib>
#include <queue>

#define MAXN 5005
#define MAXM 200005

using namespace std;

struct edge
{
    int v,to,next;
}e[MAXM<<1];

int n,m,f,g,w,tot=0,head[MAXN],ans=0,flag[MAXN];

void add(int x, int y, int z){
    e[++tot].v=z;
    e[tot].to=y;
    e[tot].next=head[x];
    head[x]=tot;
}

struct HeapNode
{
    int v,u;
    bool operator <(const HeapNode& a) const{
        return v > a.v;
    }
};

```

```
priority_queue<HeapNode> Q;

void prim(){
    for(int i=1;i<=n;i++)
        flag[i]=0;
    flag[1]=1;
    for(int p=head[1];p;p=e[p].next)
        Q.push((HeapNode){e[p].v,e[p].to});
    for(int i=1,u;i<n;i++){
        while(flag[Q.top().u]){
            Q.pop();
            if(Q.empty()){
                puts("orz");
                exit(0);
            }
        }
        ans+=Q.top().v;
        u=Q.top().u;
        Q.pop();
        flag[u]=1;
        for(int p=head[u];p;p=e[p].next)
            if(!flag[e[p].to])
                Q.push((HeapNode){e[p].v,e[p].to});
    }
}

int main(){
    scanf("%d %d", &n, &m);
    for(int i=1;i<=n;i++)
        head[i]=0;
    for(int i=1;i<=m;i++){
        scanf("%d %d %d",&f,&g,&w);
        add(f,g,w);
        add(g,f,w);
    }
    prim();
    printf("%d",ans);
    return 0;
}
```

1.3 最短路 Djikstra

```
#include <cstdio>
#include <cstring>
#include <climits>
#include <set>

#define LL long long
#define MAXN 100005
#define MAXM 500005

using namespace std;
```

```
11
12 struct edge{
13     int v,next,to;
14 }e[MAXM];
15
16 int n,m,p,tot=0,dist[MAXN],head[MAXN];
17
18 void add(int x, int y, int z){
19     tot++;
20     e[tot].v=z;
21     e[tot].to=y;
22     e[tot].next=head[x];
23     head[x]=tot;
24 }
25
26 struct node{
27     int v,u;
28     bool operator<(const node a)const{
29         if(v!=a.v)
30             return v<a.v;
31         else
32             return u<a.u;
33     }
34     node(int x,int y):v(x),u(y){}
35 };
36
37 set<node> s;//v,u
38 set<node>::iterator ite;
39 int u,q;
40 LL tem;
41
42 void djikstra(int x){
43     for(int i=1;i<=n;i++){
44         dist[i]=INT_MAX;
45     }
46     dist[x]=0;
47     s.insert(node(0,x));
48     while(!s.empty()){
49         ite=s.begin();
50         u=ite->u;
51         s.erase(ite);
52         for(int q=head[u];q;q=e[q].next){
53             int v=e[q].to,w=e[q].v;
54             if(dist[u]+w<dist[v]){
55                 ite=s.find(node(dist[v],v));
56                 if(ite!=s.end()) s.erase(ite);
57                 dist[v]=dist[u]+w;
58                 s.insert(node(dist[v],v));
59             }
60         }
61     }
62 }
63
```

```
int main(){
    scanf("%d %d %d", &n, &m ,&p);
    memset(head+1,0,sizeof(head[0])*n);
    for(int i=1;i<=m;i++){
        int f,g,w;
        scanf("%d %d %d", &f, &g, &w);
        add(f,g,w);
    }
    djikstra(p);
    for(int i=1;i<=n;i++)
        printf("%d ", dist[i]);
    return 0;
}
```

1.4 最短路 SPFA

```
#include <cstdio>
#include <cstring>
#include <queue>

#define MAXN 100005
#define MAXM 500005
const int inf=0x3f3f3f3f;

using namespace std;

struct edge{
    int v,to,next;
}e[MAXM];

int n,m,p,tot,head[MAXN],dist[MAXN];
bool flag[MAXN];

void add(int x,int y,int z){
    tot++;
    e[tot].v=z;
    e[tot].to=y;
    e[tot].next=head[x];
    head[x]=tot;
}

void spfa(int x){
    queue <int> Q;
    memset(dist+1,inf,n*sizeof(dist[0]));
    memset(flag+1,0,n*sizeof(flag[0]));

    Q.push(x);
    flag[x]=true;
    dist[x]=0;
    while(!Q.empty()){
        int u=Q.front();
        Q.pop();
```

```
        flag[u]=false;
        for(int q=head[u];q;q=e[q].next){
            int v=e[q].to;
            if(dist[u]+e[q].v<dist[v]){
                dist[v]=dist[u]+e[q].v;
                if(!flag[v]){
                    Q.push(v);
                    flag[v]=true;
                }
            }
        }
    }
}

int main(){
    scanf("%d %d %d",&n,&m,&p);
    tot=0;
    memset(head+1,0,n*sizeof(head[0]));
    for(int i=1;i<=m;i++){
        int f,g,w;
        scanf("%d %d %d", &f, &g, &w);
        add(f,g,w);
    }
    spfa(p);
    for(int i=1;i<=n;i++)
        printf("%d ", dist[i]==inf?2147483647:dist[i]);
    return 0;
}
```

1.5 最近公共祖先 LCA_Doubling

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#define MAXN 500005
#define MAXM 500005
#define MAXLN 25
using namespace std;

struct edge{
    int to,next;
}e[MAXM<<1];

int tot,head[MAXN];

void add(int x,int y){
    tot++;
    e[tot].to=y;
    e[tot].next=head[x];
    head[x]=tot;
}

```

```
int dep[MAXN],lgd[MAXN],st[MAXN][MAXLN];

void dfs(int cur,int fa){
    dep[cur]=dep[fa]+1;
    st[cur][0]=fa;
    for(lgd[cur]=1;(1<<lgd[cur])<=dep[cur];lgd[cur]++)
        st[cur][lgd[cur]]=st[st[cur][lgd[cur]-1]][lgd[cur]-1];

    for(int p=head[cur];p;p=e[p].next){
        if(e[p].to==fa) continue;
        dfs(e[p].to,cur);
    }
}

int lca(int x,int y){
    if(dep[x]<dep[y]) swap(x,y);
    for(int i=0;dep[x]-dep[y];i++)
        if((dep[x]-dep[y])&(1<<i)) x=st[x][i];
    if(x==y) return x;

    for(int i=lgd[x];i>=0;i--)
        if(st[x][i]!=st[y][i])
            x=st[x][i], y=st[y][i];
    return st[x][0];
}

int n,m,s;

int main(){
    scanf("%d %d %d", &n, &m, &s);
    tot=0;
    memset(head+1,0,n*sizeof(head[0]));
    for(int i=1;i<n;i++){
        int f,g;
        scanf("%d %d", &f, &g);
        add(f,g);
        add(g,f);
    }
    dep[0]=0;
    dfs(s,0);
    for(int i=1;i<=m;i++){
        int f,g;
        scanf("%d %d", &f, &g);
        printf("%d\n", lca(f,g));
    }
    return 0;
}
```

1.6 最近公共祖先 LCA_Tarjan

```
#include <cstdio>
#define MAXN 500005
```

```
3  #define MAXM 500005
4
5  struct edge{
6      int to,next;
7  }e[MAXM<<1],eq[MAXM<<1];
8
9  int n,m,s,tot,totq,head[MAXN],headq[MAXN],fa[MAXN],ans[MAXM];
10
11 void add(int x, int y){
12     tot++;
13     e[tot].to=y;
14     e[tot].next=head[x];
15     head[x]=tot;
16 }
17
18 void addq(int x, int y){
19     tot++;
20     eq[tot].to=y;
21     eq[tot].next=headq[x];
22     headq[x]=tot;
23 }
24
25 int find(int x){
26     if(fa[x]==x)
27         return x;
28     return fa[x]=find(fa[x]);
29 }
30
31 void tarjan(int x){
32     fa[x]=x;
33     for(int p=head[x];p;p=e[p].next){
34         int u=e[p].to;
35         if(!fa[u]){
36             tarjan(u);
37             fa[u]=x;
38         }
39     }
40     for(int p=headq[x];p;p=eq[p].next){
41         int u=eq[p].to;
42         if(fa[u]){
43             ans[(p+1)>>1]=find(u);
44         }
45     }
46 }
47
48 int main(){
49     scanf("%d %d %d", &n, &m, &s);
50     for(int i=1;i<=n;i++){
51         head[i]=0;
52         headq[i]=0;
53     }
54     tot=0;
55     for(int i=1;i<n;i++){
```

```
        int f,g;
        scanf("%d %d",&f,&g);
        add(f,g);
        add(g,f);
    }
    tot=0;
    for(int i=1;i<=m;i++){
        int f,g;
        scanf("%d %d",&f,&g);
        addq(f,g);
        addq(g,f);
    }
    for(int i=1;i<=n;i++){
        fa[i]=0;
    }
    tarjan(s);
    for(int i=1;i<=m;i++)
        printf("%d\n", ans[i]);
    return 0;
}
```

1.7 判断负环 SPFA_Negtive_Cycle

```
#include <cstdio>
#include <cstring>
#include <queue>

#define MAXN 100005
#define MAXM 500005
const int inf=0x3f3f3f3f;

using namespace std;

struct edge{
    int v,to,next;
}e[MAXM];

int n,m,tot,head[MAXN];
int dist[MAXN],cnt[MAXN];
bool flag[MAXN];

void add(int x,int y,int z){
    tot++;
    e[tot].v=z;
    e[tot].to=y;
    e[tot].next=head[x];
    head[x]=tot;
}

bool spfa(int x){//负环return false
    queue <int> Q;
    memset(dist+1,inf,n*sizeof(dist[0]));
```

```
    memset(flag+1,0,n*sizeof(flag[0]));
    memset(cnt+1,0,n*sizeof(cnt[0]));

    Q.push(x);
    flag[x]=true;
    dist[x]=0;
    while(!Q.empty()){
        int u=Q.front();
        Q.pop();
        flag[u]=false;
        for(int q=head[u];q;q=e[q].next){
            int v=e[q].to;
            if(dist[u]+e[q].v<dist[v]){
                dist[v]=dist[u]+e[q].v;
                if(!flag[v]) {
                    Q.push(v);
                    flag[v]=true;
                    if(++cnt[v]>=n){
                        return false;
                    }
                }
            }
        }
    }
    return true;
}

void solve() {
    scanf("%d %d",&n,&m);
    tot=0;
    memset(head+1,0,n*sizeof(head[0]));
    for(int i=1;i<=m;i++){
        int f,g,w;
        scanf("%d %d %d", &f, &g, &w);
        add(f,g,w);
        if(w>=0) add(g,f,w);
    }
    if(spfa(1)) printf("NO\n");
    else printf("YES\n");
}

int main(){
    int T=1;
    scanf("%d", &T);
    while(T--) {
        solve();
    }
    return 0;
}
```

1.8 拓扑排序 Topological_Sort_Khan

```cpp
#include <cstdio>
#include <cstring>
#include <stack>
#define MAXN 100005
#define MAXM 200005

using namespace std;

struct edge{
    int to,next;
}e[MAXM];

int n,m,tot,head[MAXN],indgr[MAXN],list[MAXN];
bool flag[MAXN];

void add(int x,int y){
    tot++;
    e[tot].to=y;
    e[tot].next=head[x];
    head[x]=tot;
}

void khan(){
    stack<int> s;
    int cnt=0;
    memset(flag+1,0,n*sizeof(flag[0]));
    for(int i=1;i<=n;i++){
        if(!flag[i] && indgr[i]==0){
            s.push(i);
            flag[i]=true;
            while(!s.empty()){
                int u=s.top();s.pop();
                list[++cnt]=u;
                flag[u]=true;
                for(int p=head[u];p;p=e[p].next){
                    int v=e[p].to;
                    indgr[v]--;
                    if(indgr[v]==0)
                        s.push(v);
                }
            }
        }
    }
}

int main(){
    scanf("%d %d",&n,&m);
    memset(head+1,0,n*sizeof(head[0]));
    memset(indgr+1,0,n*sizeof(indgr[0]));
    tot=0;
    for(int i=1;i<=m;i++){
        int f,g;
        scanf("%d %d", &f, &g);
```

```
54          add(f,g);
55          indgr[g]++;
56      }
57      khan();
58      for(int i=1;i<=n;i++)
59          printf("%d ", list[i]);
60      return 0;
61  }
```

2 多项式

2.1 FFT 字符串匹配 String_Match_FFT

```cpp
#include <cstdio>
#include <cstring>
#include <cmath>
#include <algorithm>
#include <complex>
#define MAXN 300005
#define MAXL 550005

using namespace std;

const double PI=acos(-1);
complex<double> omg[MAXL],iomg[MAXL];

void init(int n){
    for(int i=0;i<n;i++){
        omg[i]=polar(1.0,2.0*PI*i/n);
        iomg[i]=conj(omg[i]);
    }
}

void FFT(int n,complex<double>* P,complex<double>* w){
    for(int i=0,j=0;i<n;i++){
        if(i<j) swap(P[i],P[j]);
        for(int l=n>>1;(j^=l)<l;l>>=1);
    }

    for(int i=2,l;i<=n;i<<=1){
        l=i>>1;
        complex<double> t;
        for(int j=0;j<n;j+=i){
            for(int k=0;k<l;k++){
                t=P[j+l+k]*w[n/i*k];
                P[j+l+k]=P[j+k]-t;
                P[j+k]=P[j+k]+t;
            }
        }
    }
}

int n,m,lim;
char s1[MAXN],s2[MAXN];
complex<double> A[MAXL],B[MAXL],tA[MAXL],tB[MAXL],ans[MAXL];

int main(){
    scanf("%d %d", &m, &n);
    scanf("%s %s",s1,s2);
    for(int i=0;i<m;i++) A[m-i-1]=(s1[i]=='*'?0:s1[i]-'a'+1);
    for(int i=0;i<n;i++) B[i]=(s2[i]=='*'?0:s2[i]-'a'+1);
    for(lim=1;lim<n-1;lim<<=1);
```

```
    init(lim);
    fill(ans,ans+lim,complex<double>(0.0,0.0));

    for(int i=0;i<m;i++) tA[i]=A[i]*A[i]*A[i];
    for(int i=0;i<n;i++) tB[i]=B[i];
    fill(tA+m,tA+lim,complex<double>(0.0,0.0));
    fill(tB+n,tB+lim,complex<double>(0.0,0.0));
    FFT(lim,tA,omg);
    FFT(lim,tB,omg);
    for(int i=0;i<lim;i++) ans[i]+=tA[i]*tB[i];

    for(int i=0;i<m;i++) tA[i]=A[i];
    for(int i=0;i<n;i++) tB[i]=B[i]*B[i]*B[i];
    fill(tA+m,tA+lim,complex<double>(0.0,0.0));
    fill(tB+n,tB+lim,complex<double>(0.0,0.0));
    FFT(lim,tA,omg);
    FFT(lim,tB,omg);
    for(int i=0;i<lim;i++) ans[i]+=tA[i]*tB[i];

    for(int i=0;i<m;i++) tA[i]=A[i]*A[i];
    for(int i=0;i<n;i++) tB[i]=B[i]*B[i];
    fill(tA+m,tA+lim,complex<double>(0.0,0.0));
    fill(tB+n,tB+lim,complex<double>(0.0,0.0));
    FFT(lim,tA,omg);
    FFT(lim,tB,omg);
    for(int i=0;i<lim;i++) ans[i]-=complex<double>(2.0,0)*tA[i]*tB[i];

    FFT(lim,ans,iomg);

    int cnt=0;
    for(int i=m-1;i<n;i++)
        if((int)(ans[i].real()/lim+0.5)==0) cnt++;
    printf("%d\n", cnt);
    for(int i=m-1;i<n;i++)
        if((int)(ans[i].real()/lim+0.5)==0) printf("%d ", i-m+2);
    return 0;
}
```

2.2 FFT 递归 Fast_Fourier_Transform_Cooley-Tukey_Recursion

```
#include <cstdio>
#include <algorithm>
#include <complex>
#define MAXN 4000005

using namespace std;

complex<double> omg[MAXN],iomg[MAXN],temp[MAXN];

void init(int n){
    double PI=acos(-1);
    for(int i=0;i<n;i++){
```

```
        omg[i]=polar(1.0,2.0*PI*i/n);
        iomg[i]=conj(omg[i]);
    }
}

void FFT(int n,complex<double>* buffer,int offset,int step,complex<double>* omg){
    if(n==1) return;
    int m=n>>1;
    FFT(m,buffer,offset,step<<1,omg);
    FFT(m,buffer,offset+step,step<<1,omg);
    for(int i=0;i<m;i++){
        int pos=2*i*step;
        temp[i]=buffer[offset+pos]+omg[i*step]*buffer[offset+step+pos];
        temp[i+m]=buffer[offset+pos]-omg[i*step]*buffer[offset+step+pos];
    }
    for(int i=0;i<n;i++)
        buffer[offset+i*step]=temp[i];
}

int n,m,lim;
complex<double> A[MAXN],B[MAXN];

int main(){
    scanf("%d %d", &n, &m);
    for(lim=1;lim<=n+m;lim<<=1);
    fill(A,A+lim,complex<double>(0.0,0.0));
    fill(B,B+lim,complex<double>(0.0,0.0));
    for(int i=0,t;i<n+1;i++){
        scanf("%d", &t);
        A[i]+=t;
    }
    for(int i=0,t;i<m+1;i++){
        scanf("%d", &t);
        B[i]+=t;
    }
    init(lim);
    FFT(lim,A,0,1,omg);
    FFT(lim,B,0,1,omg);
    for(int i=0;i<lim;i++)
        A[i]=A[i]*B[i];
    FFT(lim,A,0,1,iomg);
    for(int i=0;i<n+m+1;i++)
        printf("%d ", (int)(A[i].real()/lim+0.5));
    return 0;
}
```

2.3 FFT 递推 Fast_Fourier_Transform_Cooley-Tukey_Iteration

```
#include <bits/stdc++.h>
#define MAXN 2100005

using namespace std;
```

```
5
6   const double PI=acos(-1);
7
8   struct Complex{
9       double real,image;
10      Complex operator+(Complex y)const{
11          return {real+y.real,image+y.image};
12      }
13      Complex operator-(Complex y)const{
14          return {real-y.real,image-y.image};
15      }
16      Complex operator*(Complex y)const{
17          return {real*y.real-image*y.image,real*y.image+image*y.real};
18      }
19  };
20
21  void FFT(int n,Complex* P,int f){
22      for(int i=0,j=0;i<n;i++){
23          if(i<j) swap(P[i],P[j]);
24          for(int l=n>>1;(j^=l)<l;l>>=1);
25      }
26
27      for(int i=2,l;i<=n;i<<=1){
28          l=i>>1;
29          Complex wn={cos(2*PI/i),f*sin(2*PI/i)};
30          for(int j=0;j<n;j+=i){
31              Complex w={1,0};
32              for(int k=0;k<l;k++,w=w*wn){
33                  Complex t=P[j+l+k]*w;
34                  P[j+l+k]=P[j+k]-t;
35                  P[j+k]=P[j+k]+t;
36              }
37          }
38      }
39  }
40
41  int n,m,lim;
42  Complex A[MAXN],B[MAXN];
43
44  int main(){
45      scanf("%d %d", &n, &m);
46      for(lim=1;lim<=n+m;lim<<=1);
47      memset(A,0,lim*sizeof(A[0]));
48      memset(B,0,lim*sizeof(B[0]));
49      for(int i=0,t;i<n+1;i++){
50          scanf("%d", &t);
51          A[i]={1.0*t,0};
52      }
53      for(int i=0,t;i<m+1;i++){
54          scanf("%d", &t);
55          B[i]={1.0*t,0};
56      }
57      FFT(lim,A,1);
```

```
    FFT(lim,B,1);
    for(int i=0;i<lim;i++)
        A[i]=A[i]*B[i];
    FFT(lim,A,-1);
    for(int i=0;i<n+m+1;i++)
        printf("%d ", (int)(A[i].real/lim+0.5));
    return 0;
}
```

2.4 NTT 递推 Number_Theoretic_Transforms

```
#include <cstdio>
#include <algorithm>
#define LL long long
#define MAXN 2100005
#define MOD 998244353
#define RT 3

using namespace std;

LL omg[MAXN],iomg[MAXN];

LL binpow(LL x,LL y,LL mod){
    LL r=1%mod;
    while(y){
        if(y&1) r=(r*x)%mod;
        x=(x*x)%mod;
        y>>=1;
    }
    return r;
}

void init(int n){
    omg[0]=iomg[0]=1;
    omg[1]=binpow(RT,(MOD-1)/n,MOD);
    iomg[1]=binpow(omg[1],MOD-2,MOD);
    for(int i=2;i<n;i++){
        omg[i]=omg[i-1]*omg[1]%MOD;
        iomg[i]=iomg[i-1]*iomg[1]%MOD;
    }
}

void NTT(int n,LL* P,LL* w){
    for(int i=0,j=0;i<n;i++){
        if(i<j) swap(P[i],P[j]);
        for(int l=n>>1;(j^=l)<l;l>>=1);
    }

    for(int i=2,l;i<=n;i<<=1){
        l=i>>1;
        for(int j=0;j<n;j+=i){
            for(int k=0;k<l;k++){
```

```
                LL t=P[j+l+k]*w[n/i*k]%MOD;
                P[j+l+k]=(P[j+k]-t+MOD)%MOD;
                P[j+k]=(P[j+k]+t)%MOD;
            }
        }
    }
}

int n,m,lim;
LL A[MAXN],B[MAXN];

int main(){
    scanf("%d %d", &n, &m);
    for(lim=1;lim<=n+m;lim<<=1);
    for(int i=0;i<n+1;i++){
        scanf("%lld", &A[i]);
    }
    for(int i=0;i<m+1;i++){
        scanf("%lld", &B[i]);
    }
    init(lim);
    NTT(lim,A,omg);
    NTT(lim,B,omg);
    for(int i=0;i<lim;i++)
        A[i]=A[i]*B[i]%MOD;
    NTT(lim,A,iomg);
    LL invn=binpow(lim,MOD-2,MOD);
    for(int i=0;i<n+m+1;i++)
        printf("%lld ", A[i]*invn%MOD);
    return 0;
}
```

2.5 多项式求逆 Polynomial_Inverse

```
#include <cstdio>
#include <algorithm>
#define LL long long
#define MAXN 270005
#define MOD 998244353
#define RT 3

using namespace std;

LL omg[MAXN],iomg[MAXN];

LL binpow(LL x,LL y,LL mod){
    LL r=1%mod;
    while(y){
        if(y&1) r=(r*x)%mod;
        x=(x*x)%mod;
        y>>=1;
    }
```

```
19      return r;
20  }
21
22  void init(int n){
23      omg[0]=iomg[0]=1;
24      omg[1]=binpow(RT,(MOD-1)/n,MOD);
25      iomg[1]=binpow(omg[1],MOD-2,MOD);
26      for(int i=2;i<n;i++){
27          omg[i]=omg[i-1]*omg[1]%MOD;
28          iomg[i]=iomg[i-1]*iomg[1]%MOD;
29      }
30  }
31
32  void NTT(int n,LL* P,LL* w){
33      for(int i=0,j=0;i<n;i++){
34          if(i<j) swap(P[i],P[j]);
35          for(int l=n>>1;(j^=l)<l;l>>=1);
36      }
37
38      for(int i=2,l;i<=n;i<<=1){
39          l=i>>1;
40          for(int j=0;j<n;j+=i){
41              for(int k=0;k<l;k++){
42                  LL t=P[j+l+k]*w[n/i*k]%MOD;
43                  P[j+l+k]=(P[j+k]-t+MOD)%MOD;
44                  P[j+k]=(P[j+k]+t)%MOD;
45              }
46          }
47      }
48  }
49
50  void poly_inv(int dgr,LL* X,LL* Y){
51      if(dgr==1){
52          Y[0]=binpow(X[0],MOD-2,MOD);
53      }
54      else{
55          poly_inv((dgr+1)>>1,X,Y);
56
57          static LL Z[MAXN];
58          int lim;
59          for(lim=1;lim<(dgr<<1);lim<<=1);
60          copy(X,X+dgr,Z);
61          fill(Z+dgr,Z+lim,0);
62          init(lim);
63
64          NTT(lim,Z,omg);
65          NTT(lim,Y,omg);
66          for(int i=0;i<lim;i++)
67              Y[i]=(2-Z[i]*Y[i]%MOD+MOD)*Y[i]%MOD;
68          NTT(lim,Y,iomg);
69          LL invlim=binpow(lim,MOD-2,MOD);
70          for(int i=0;i<dgr;i++)
71              Y[i]=Y[i]*invlim%MOD;
```

```
72          fill(Y+dgr,Y+lim,0);
73      }
74  }
75
76  int n;
77  LL A[MAXN],B[MAXN];
78
79  int main(){
80      scanf("%d", &n);
81      for(int i=0;i<n;i++)
82          scanf("%lld", &A[i]);
83      poly_inv(n,A,B);
84      for(int i=0;i<n;i++)
85          printf("%lld ", B[i]);
86      return 0;
87  }
```

3 字符串

3.1 字符串哈希 String_Hash

```cpp
#include <cstdio>
#include <string>

#define BASE 307
#define MOD 5555567

int hsh(string x){
    int h=0,len=x.length();
    for(int i=0;i<len;i++){
        h=(h*BASE+x[i])%MOD;
    }
    return h;
}

int main(){
    return 0;
}
```

3.2 马拉车 Manacher

```cpp
#include <cstdio>
#include <cstring>
#include <algorithm>
#define MAXN 11000005
using namespace std;

int r[MAXN<<1],mx;
char st[MAXN<<1];

void manacher(char *s){
    int len=strlen(s);
    st[0]='$';
    for(int i=0;i<len;i++){
        st[i<<1|1]='#';
        st[(i+1)<<1]=s[i];
    }
    len=len<<1|1;
    st[len]='#';
    st[len+1]='*';
    r[1]=1;
    mx=0;
    for(int i=2,mid=1;i<=len;i++){
        r[i]=min(mid+r[mid]-i,r[2*mid-i]);
        for(;st[i-r[i]]==st[i+r[i]];r[i]++);
        if(i+r[i]>mid+r[mid]) mid=i;
        mx=max(mx,r[i]-1);
    }
}
```

```
29
30 char s[MAXN];
31
32 int main(){
33     scanf("%s",s);
34     manacher(s);
35     printf("%d\n", mx);
36     return 0;
37 }
```

3.3 字符串匹配 KMP

```
1  #include <cstdio>
2  #include <cstring>
3  #define MAXN 1000005
4
5  int n,m,cnt,next[MAXN];
6  char s1[MAXN],s2[MAXN];
7
8  void kmp(){
9      next[0]=-1;
10     for(int i=1,k=-1;i<=m;i++){
11         //k初始值为-1. next数组可以查询boarder的boarder.
12         while(~k && s2[k+1]!=s2[i])
13             k=next[k];
14         next[i]=++k;
15     }
16
17     cnt=0;
18     for(int i=1,k=0;i<=n;i++){
19         //匹配串前缀与模式串后缀比，上一位前缀的boarder的boarder也能匹配。
20         while(~k && s2[k+1]!=s1[i])
21             k=next[k];
22         if(m==++k){
23             cnt++;
24             printf("%d\n", i-m+1);
25         }
26     }
27 }
28
29 int main(){
30     scanf("%s %s", s1+1, s2+1);
31     n=strlen(s1+1);
32     m=strlen(s2+1);
33     kmp();
34     for(int i=1;i<=m;i++){
35         printf("%d ", next[i]);
36     }
37     return 0;
38 }
```

3.4 AC 自动机 AC-Automaton

```cpp
#include <cstdio>
#include <cstring>
#include <map>
#include <queue>
#define MAXN 1000006
using namespace std;

struct trie{
    int fail,mark,ch[26];
}tt[MAXN];

int tot;

void insert(char *s){
    int len=strlen(s+1), cur=0;
    for(int i=1;i<=len;i++){
        int& next=tt[cur].ch[s[i]-'a'];
        if(!next){
            next=++tot;
            tt[tot].mark=0;
        }
        cur=next;
    }
    tt[cur].mark++;
}

void getfail(){
    queue<int> Q;
    tt[0].fail=0;
    tt[0].mark=0;
    for(int i=0;i<26;i++)
        if(tt[0].ch[i]) tt[tt[0].ch[i]].fail=0, Q.push(tt[0].ch[i]);
    while(!Q.empty()){
        int u=Q.front(); Q.pop();
        for(int i=0;i<26;i++){
            if(!tt[u].ch[i]) continue;
            int k=tt[u].fail;
            while(k && !tt[k].ch[i])
                k=tt[k].fail;
            tt[tt[u].ch[i]].fail=tt[k].ch[i];
            // tt[tt[u].ch[i]].mark+=tt[tt[k].ch[i]].mark;//如果需要重复统计，fail累加标记
            Q.push(tt[u].ch[i]);
        }
    }
}

int query(char *s){
    int len=strlen(s+1),ans=0;
    for(int i=1,k=0;i<=len;i++){
        while(k && !tt[k].ch[s[i]-'a'])
            k=tt[k].fail;
```

```
        k=tt[k].ch[s[i]-'a'];
        ans+=tt[k].mark;

        tt[k].mark=0;//清除该字符串的标记（只求
    }
    return ans;
}

int n;
char s[MAXN];

int main(){
    scanf("%d", &n);
    tot=0;
    memset(tt,0,sizeof(tt));
    for(int i=1;i<=n;i++){
        scanf("%s", s+1);
        insert(s);
    }
    scanf("%s", s+1);
    getfail();
    printf("%d\n", query(s));
    return 0;
}
```

3.5 后缀数组 Suffix_Array

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#define MAXN 1000005
using namespace std;

//sa:排名对应的前缀，rk:前缀的排名，tp:第二关键字排名对应的前缀，tax:排名对应的个数
//height:排名i与i-1后缀的LCP(最长公共前缀)
int sa[MAXN],r1[MAXN],r2[MAXN],tax[MAXN],height[MAXN];
int *rk=r1,*tp=r2;
char s[MAXN];

void rsort(int n,int m){
    memset(tax,0,(m+1)*sizeof(tax[0]));
    for(int i=1;i<=n;i++) tax[rk[i]]++;//当前排名装桶
    for(int i=1;i<=m;i++) tax[i]+=tax[i-1];//计算桶的名次
    for(int i=n;i>=1;i--) sa[tax[rk[tp[i]]]--]=tp[i];//按照第二关键字降序，分配排名。
}

void get_sa(char* s){
    //O(nlogn)
    int n=strlen(s+1), m=0;
    for(int i=1;i<=n;i++)
        m=max(m,rk[i]=s[i]),tp[i]=i;
    rsort(n,m);
```

```
    for(int k=1,p=0;p<n;k<<=1,m=p){
        p=0;
        //重制第二关键字
        for(int i=n-k+1;i<=n;i++) tp[++p]=i; //后续为空，排前面
        for(int i=1;i<=n;i++) if(sa[i]>k) tp[++p]=sa[i]-k; //按照第一关键字排第二关键字

        rsort(n,m);

        swap(tp,rk);
        rk[sa[1]]=p=1;
        for(int i=2;i<=n;i++){
            rk[sa[i]]=(tp[sa[i]]==tp[sa[i-1]] && tp[sa[i]+k]==tp[sa[i-1]+k])?p:++p;
        }
    }

    //利用height[rk[i+1]]>=height[rk[i]]-1
    //O(n)
    for(int i=1,k=0;i<=n;i++){
        if(k) k--;
        while(rk[i]>1 && s[i+k]==s[sa[rk[i]-1]+k]) k++;
        height[rk[i]]=k;
    }
}

int main(){
    scanf("%s",s+1);
    get_sa(s);
    int len=strlen(s+1);
    for(int i=1;i<=len;i++)
        printf("%d ", sa[i]);
    return 0;
}
```

3.6 后缀自动机 Suffix-Automaton

```
#include <bits/stdc++.h>
#define MAXN 1000005
#define LL long long
using namespace std;

struct SAM {
    int len,link,cnt;
    int ch[26];
}sam[MAXN<<1];

int sz,last;

void sam_init() {
    sam[0].len=0;
    sam[0].link=-1;
    sam[0].cnt=0;
    memset(sam[0].ch,0,sizeof(sam[0].ch));
```

```
18      sz=0;
19      last=0;
20  }
21
22  void sam_extend(int c) {
23      int cur=++sz;
24      sam[cur].len=sam[last].len+1;
25      memset(sam[cur].ch,0,sizeof(sam[cur].ch));
26
27      int p=last;
28      for(;~p && !sam[p].ch[c];p=sam[p].link)
29          sam[p].ch[c]=cur;
30
31      if(!~p) {
32          sam[cur].link=0;
33      } else {
34          int q=sam[p].ch[c];
35          if(sam[p].len+1==sam[q].len) {
36              sam[cur].link=q;
37          } else {
38              int clone=++sz;
39              sam[clone]=sam[q];
40              sam[clone].len=sam[p].len+1;
41              sam[clone].cnt=0;
42              sam[q].link=sam[cur].link=clone;
43              for(;~p && sam[p].ch[c]==q;p=sam[p].link)
44                  sam[p].ch[c]=clone;
45          }
46      }
47
48      last=cur;
49      sam[cur].cnt=1;
50  }
51
52  struct edge{
53      int to,next;
54  }e[MAXN<<1];
55
56  int tot,head[MAXN<<1];
57
58  void add(int x,int y) {
59      tot++;
60      e[tot].to=y;
61      e[tot].next=head[x];
62      head[x]=tot;
63  }
64
65  char s[MAXN];
66  LL ans;
67
68  void dfs(int x) {
69      for(int p=head[x];p;p=e[p].next) {
70          int u=e[p].to;
```

```
        dfs(u);
        sam[x].cnt+=sam[u].cnt;
    }
    if(sam[x].cnt!=1) ans=max(ans,1LL*sam[x].len*sam[x].cnt);
}

void solve() {
    scanf("%s", s);
    int len=strlen(s);
    sam_init();
    for(int i=0;i<len;i++)
        sam_extend(s[i]-'a');
    tot=0;
    memset(head,0,(sz+1)*sizeof(head[0]));
    for(int i=1;i<=sz;i++)
        add(sam[i].link,i);
    ans=0;
    dfs(0);
    printf("%lld\n", ans);
}

int main() {
    int T=1,cas=1;
    // scanf("%d", &T);
    while(T--) {
        // printf("Case #%d: ", cas++);
        solve();
    }
    return 0;
}
```

3.7 广义后缀自动机 General_Suffix-Automaton

```
#include <bits/stdc++.h>
#define MAXN 1000005
#define LL long long
using namespace std;
// const int inf=0x3f3f3f3f;

struct SAM {
    int len,link,cnt;
    int ch[26];
}sam[MAXN<<1];

int sz,last;

void sam_init() {
    sam[0].len=0;
    sam[0].link=-1;
    sam[0].cnt=0;
    memset(sam[0].ch,0,sizeof(sam[0].ch));
    sz=0;
```

```
    last=0;
}

void sam_extend(int c) {
    if(sam[last].ch[c] && sam[last].len+1==sam[sam[last].ch[c]].len) {
        last=sam[last].ch[c];
        sam[last].cnt++;
        return;
    }

    int cur=++sz;
    sam[cur].len=sam[last].len+1;
    memset(sam[cur].ch,0,sizeof(sam[cur].ch));

    int p=last;
    for(;~p && !sam[p].ch[c];p=sam[p].link)
        sam[p].ch[c]=cur;

    if(!~p) {
        sam[cur].link=0;
    } else {
        int q=sam[p].ch[c];
        if(sam[p].len+1==sam[q].len) {
            sam[cur].link=q;
        } else {
            int clone;
            if(p==last) {
                clone=cur;
            } else {
                clone=++sz;
                sam[cur].link=clone;
            }

            sam[clone]=sam[q];
            sam[clone].len=sam[p].len+1;
            sam[q].link=clone;
            sam[clone].cnt=0;
            for(;~p && sam[p].ch[c]==q;p=sam[p].link)
                sam[p].ch[c]=clone;
        }
    }

    last=cur;
    sam[cur].cnt=1;
}

char s[MAXN];

void solve() {
    int n;
    scanf("%d", &n);
    sam_init();
    for(int i=1;i<=n;i++) {
```

```
73          scanf("%s", s);
74          int len=strlen(s);
75          last=0;
76          for(int j=0;j<len;j++) {
77              sam_extend(s[j]-'a');
78          }
79      }
80
81      LL ans=0;
82      for(int i=1;i<=sz;i++) {
83          ans+=sam[i].len-sam[sam[i].link].len;
84      }
85
86      printf("%lld\n", ans);
87  }
88
89  int main() {
90      int T=1,cas=1;
91      // scanf("%d", &T);
92      while(T--) {
93          // printf("Case #%d: ", cas++);
94          solve();
95      }
96      return 0;
97  }
```

4 数据结构

4.1 并查集 Union_Find

```cpp
#include <cstdio>
#define maxn 10005

int n,m,a,b,c,fa[maxn];

void ini(){
    for(int i=1;i<=n;i++){
        fa[i]=i;
    }
}

int find(int x){
    if(fa[x] == x)
        return x;
    return fa[x]=find(fa[x]);
}

void join(int x, int y){
    fa[find(x)]=find(y);
}

int main(){
    scanf("%d %d", &n, &m);
    ini();
    for(int i=1; i<=m; i++){
        scanf("%d %d %d", &a, &b, &c);
        if(a==1)
            join(b,c);
        else if(find(b)==find(c))
            puts("Y");
        else puts("N");
    }
    return 0;
}
```

4.2 分块 Block_1

```cpp
#include <cstdio>
#include <cstring>
#include <cmath>
#include <algorithm>
#define MAXN 50005

using namespace std;

int n,blo,v[MAXN],bl[MAXN],atag[MAXN];

void add(int l,int r,int x){
```

```
    if(bl[l]==bl[r]){
        for(int i=l;i<=r;i++)
            v[i]+=x;
    }
    else{
        for(int i=l;i<=bl[l]*blo;i++)
            v[i]+=x;
        for(int i=(bl[r]-1)*blo+1;i<=r;i++)
            v[i]+=x;
    }
    for(int i=bl[l]+1;i<=bl[r]-1;i++)
        atag[i]+=x;
}

int main(){
    scanf("%d", &n);blo=sqrt(n);
    for(int i=1;i<=n;i++)
        scanf("%d", &v[i]);
    for(int i=1;i<=n;i++)
        bl[i]=(i-1)/blo+1;
    memset(atag+1,0,bl[n]*sizeof(atag[0]));
    for(int i=1;i<=n;i++){
        int opt,l,r,c;
        scanf("%d %d %d %d", &opt, &l, &r, &c);
        if(opt==0){
            add(l,r,c);
        }
        else printf("%d\n", v[r]+atag[bl[r]]);
    }
    return 0;
}
```

4.3 分块 Block_2

```
#include <cstdio>
#include <cstring>
#include <cmath>
#include <algorithm>
#include <vector>
#define MAXN 50005
#define MAXB 505

using namespace std;

int n,blo,v[MAXN],bl[MAXN],atag[MAXB];
vector<int> ve[MAXB];

void reset(int x){
    ve[x].clear();
    for(int i=(x-1)*blo+1;i<=min(x*blo,n);i++){
        ve[x].push_back(v[i]);
    }
```

```
    sort(ve[x].begin(),ve[x].end());
}

void add(int l,int r,int x){
    if(bl[l]==bl[r]){
        for(int i=l;i<=r;i++)
            v[i]+=x;
        reset(bl[l]);
    }
    else{
        for(int i=l;i<=bl[l]*blo;i++)
            v[i]+=x;
        reset(bl[l]);
        for(int i=(bl[r]-1)*blo+1;i<=r;i++)
            v[i]+=x;
        reset(bl[r]);
        for(int i=bl[l]+1;i<=bl[r]-1;i++)
        atag[i]+=x;
    }
}

int query(int l,int r,int x){
    int cnt=0;
    if(bl[l]==bl[r]){
        for(int i=l;i<=r;i++)
            if(v[i]+atag[bl[i]]<x) cnt++;
    }
    else{
        for(int i=l;i<=bl[l]*blo;i++)
            if(v[i]+atag[bl[i]]<x) cnt++;
        for(int i=(bl[r]-1)*blo+1;i<=r;i++)
            if(v[i]+atag[bl[i]]<x) cnt++;
        for(int i=bl[l]+1;i<=bl[r]-1;i++)
            cnt+=lower_bound(ve[i].begin(),ve[i].end(),x-atag[i])-ve[i].begin();
    }
    return cnt;
}

int main(){
    scanf("%d", &n);blo=sqrt(n);
    for(int i=1;i<=n;i++)
        scanf("%d", &v[i]);
    for(int i=1;i<=n;i++){
        bl[i]=(i-1)/blo+1;
        ve[bl[i]].push_back(v[i]);
    }
    memset(atag+1,0,bl[n]*sizeof(atag[0]));
    for(int i=1;i<=bl[n];i++)
        sort(ve[i].begin(),ve[i].end());

    for(int i=1;i<=n;i++){
        int opt,l,r,c;
        scanf("%d %d %d %d", &opt, &l, &r, &c);
```

```
72          if(opt==0){
73              add(l,r,c);
74          }
75          else{
76              printf("%d\n", query(l,r,c*c));
77          }
78      }
79      return 0;
80  }
```

4.4 分块 Block_4

```
1   #include <cstdio>
2   #include <cstring>
3   #include <cmath>
4   #include <algorithm>
5   #define LL long long
6   #define MAXN 50005
7   #define MAXB 505
8
9   using namespace std;
10
11  int n,blo;
12  LL v[MAXN],bl[MAXN],atag[MAXB],sum[MAXB];
13
14  void add(int l,int r,int x){
15      if(bl[l]==bl[r]){
16          for(int i=l;i<=r;i++)
17              v[i]+=x,sum[bl[i]]+=x;
18      }
19      else{
20          for(int i=l;i<=bl[l]*blo;i++)
21              v[i]+=x,sum[bl[i]]+=x;
22          for(int i=(bl[r]-1)*blo+1;i<=r;i++)
23              v[i]+=x,sum[bl[i]]+=x;
24          for(int i=bl[l]+1;i<=bl[r]-1;i++)
25              atag[i]+=x;
26      }
27  }
28
29  LL query(int l,int r){
30      LL ans=0;
31      if(bl[l]==bl[r]){
32          for(int i=l;i<=r;i++)
33              ans+=v[i]+atag[bl[i]];
34      }
35      else{
36          for(int i=l;i<=bl[l]*blo;i++)
37              ans+=v[i]+atag[bl[i]];
38          for(int i=(bl[r]-1)*blo+1;i<=r;i++)
39              ans+=v[i]+atag[bl[i]];
40          for(int i=bl[l]+1;i<=bl[r]-1;i++)
```

```
            ans+=sum[i]+atag[i]*blo;
    }
    return ans;
}

int main(){
    scanf("%d", &n);blo=sqrt(n);
    for(int i=1;i<=n;i++)
        bl[i]=(i-1)/blo+1;
    memset(atag+1,0,bl[n]*sizeof(atag[0]));
    memset(sum+1,0,bl[n]*sizeof(sum[0]));
    for(int i=1;i<=n;i++){
        scanf("%lld", &v[i]);
        sum[bl[i]]+=v[i];
    }
    for(int i=1;i<=n;i++){
        int opt,l,r,c;
        scanf("%d %d %d %d", &opt, &l, &r, &c);
        if(opt==0){
            add(l,r,c);
        }
        else printf("%lld\n", query(l,r)%(c+1));
    }
    return 0;
}
```

4.5 树状数组 Binary_Indexed_Tree

```
/*
    Coded with Leachim's ACM Template.
    No errors. No warnings. ~~
*/
#include <bits/stdc++.h>
#pragma GCC diagnostic ignored "-Wunused-const-variable"
#pragma GCC diagnostic ignored "-Wsign-conversion"
#pragma GCC diagnostic ignored "-Wsign-compare"
#define LL long long
using namespace std;
const int inf=0x3f3f3f3f;
const double eps=1e-7;
const int dx[4]={1,-1,0,0};
const int dy[4]={0,0,1,-1};
const int MAXN=2000005;

int n,m,bit[MAXN];

int lowbit(int x){
    return x&(-x);
}

void change(int x,int y){
    for(;x<=n;x+=lowbit(x))
```

```
25          bit[x]+=y;
26  }
27
28  int sum(int x){
29      int s=0;
30      for(;x>0;x-=lowbit(x))
31          s+=bit[x];
32      return s;
33  }
34
35  void build() {
36      for(int x=1;x<<1<=n;x<<=1)
37          for(int i=x;i+x<=n;i+=x<<1)
38              bit[i+x]+=bit[i];
39  }
40
41  void solve() {
42      scanf("%d %d", &n, &m);
43      for(int i=1;i<=n;i++) scanf("%d", &bit[i]);
44      build();
45      for(int i=1;i<=m;i++) {
46          int opt,x,y;
47          scanf("%d %d %d", &opt, &x, &y);
48          if(opt==1) {
49              change(x,y);
50          } else {
51              printf("%d\n", sum(y)-sum(x-1));
52          }
53      }
54  }
55
56  int main() {
57      int T=1,cas=1;(void)(cas);
58      // scanf("%d", &T);
59      while(T--) {
60          // printf("Case #%d: ", cas++);
61          solve();
62      }
63      return 0;
64  }
```

4.6 树状数组 2D_Binary_Indexed_Tree

```
1   #include <cstdio>
2
3   #define _for(i,a,b) for(int (i)=(a);(i)<=(b);(i)++)
4   #define MAXN 1005
5
6   int n,m,H,p,q,bit[MAXN][MAXN];
7
8   int lowbit(int x){
9       return x&(-x);
```

```
}

void change(int x,int y,int k){
    for(int i=x;i<=n;i+=lowbit(i)){
        for(int j=y;j<=m;j+=lowbit(j)){
            bit[i][j]+=k;
        }
    }
}

int sum(int x,int y){
    int s=0;
    for(int i=x;i>0;i-=lowbit(i)){
        for(int j=y;j>0;j-=lowbit(j)){
            s+=bit[i][j];
        }
    }
    return s;
}

int main(){
    scanf("%d %d %d",&n, &m, &H);
    _for(i,1,n+1)
        _for(j,1,m+1)
            bit[i][j]=0;
    _for(i,1,H){
        scanf("%d %d", &p, &q);
        change(1,1,1);
        change(p+1,1,-1);
        change(1,q+1,-1);
        change(p+1,q+1,1);//1-2+1
    }
    _for(i,1,n)
        _for(j,1,m)
            printf("%d\n", sum(i,j));
}
```

4.7 线段树 Segment_Tree

```
#include <cstdio>
#include <algorithm>
#define LL long long
#define MAXN 100005
#define MAXT MAXN<<2
using namespace std;

struct node{
    int le,ri;
    LL sum,tag;
}sgt[MAXT];

int n,m;
```

```
LL a[MAXN];

void build(int cur,int l,int r){
    sgt[cur].le=l;
    sgt[cur].ri=r;
    sgt[cur].tag=0;
    if(l+1<r){
        int le=cur<<1,ri=le+1;
        build(le,l,(l+r)>>1);
        build(ri,(l+r)>>1,r);
        sgt[cur].sum=sgt[le].sum+sgt[ri].sum;
    }
    else
        sgt[cur].sum=a[l];
}

void update(int cur){
    int le=cur<<1,ri=le+1;
    sgt[le].sum+=sgt[cur].tag*(sgt[le].ri-sgt[le].le);
    sgt[le].tag+=sgt[cur].tag;
    sgt[ri].sum+=sgt[cur].tag*(sgt[ri].ri-sgt[ri].le);
    sgt[ri].tag+=sgt[cur].tag;
    sgt[cur].tag=0;

}

void modify(int cur,int l,int r,LL delta){
    if(l<=sgt[cur].le && sgt[cur].ri<=r){
        sgt[cur].sum+=delta*(sgt[cur].ri-sgt[cur].le);
        sgt[cur].tag+=delta;
    }
    else{
        int le=cur<<1,ri=le+1,mid=(sgt[cur].le+sgt[cur].ri)>>1;
        if(sgt[cur].tag)
            update(cur);
        if(l<mid)
            modify(le,l,r,delta);
        if(mid<r)
            modify(ri,l,r,delta);
        // sgt[cur].sum=sgt[cur].tag*(sgt[cur].ri-sgt[cur].le)+sgt[le].sum+sgt[ri].sum;
        sgt[cur].sum=sgt[le].sum+sgt[ri].sum;
    }
}

LL query(int cur,int l,int r){
    if(l<=sgt[cur].le && sgt[cur].ri<=r)
        return sgt[cur].sum;
    else{
        int le=cur<<1,ri=le+1,mid=(sgt[cur].le+sgt[cur].ri)>>1;
        if(sgt[cur].tag) update(cur);
        LL sum=0;
        if(l<mid)
            sum+=query(le,l,r);
```

```
67          if(mid<r)
68              sum+=query(ri,l,r);
69          // sum+=sgt[cur].tag*(min(r,sgt[cur].ri)-max(l,sgt[cur].le));
70          return sum;
71      }
72  }
73
74  int main(){
75      scanf("%d %d",&n,&m);
76      for(int i=1;i<=n;i++)
77          scanf("%lld",&a[i]);
78      build(1,1,n+1);
79      for(int i=1;i<=m;i++){
80          int cmd,le,ri;
81          scanf("%d",&cmd);
82          if(cmd==1){
83              LL k;
84              scanf("%d %d %lld",&le,&ri,&k);
85              modify(1,le,ri+1,k);
86          }
87          else{
88              scanf("%d %d",&le,&ri);
89              printf("%lld\n", query(1,le,ri+1));
90          }
91      }
92      return 0;
93  }
```

4.8 线段树 Segment_Tree_Multiply

```
1   #include <cstdio>
2   #define LL long long
3   #define MAXN 100005
4   #define MAXT 400005
5
6   int n,m,p,a[MAXN];
7
8   struct node{
9       int le,ri;
10      LL sum,del,mul;
11  }sgt[MAXT];
12
13  void built(int cur,int l,int r){
14      sgt[cur].le=l; sgt[cur].ri=r;
15      sgt[cur].del=0; sgt[cur].mul=1;
16      if(l<r-1){
17          built(cur<<1,l,(l+r)>>1);
18          built(cur<<1|1,(l+r)>>1,r);
19          sgt[cur].sum=sgt[cur<<1].sum+sgt[cur<<1|1].sum;
20      }
21      else sgt[cur].sum=a[l];
22  }
```

```
23
24 void update(int cur){
25     int lc=cur<<1,rc=lc|1;
26     sgt[lc].sum=(sgt[lc].sum*sgt[cur].mul+sgt[cur].del*(sgt[lc].ri-sgt[lc].le))%p;
27     sgt[rc].sum=(sgt[rc].sum*sgt[cur].mul+sgt[cur].del*(sgt[rc].ri-sgt[rc].le))%p;
28     sgt[lc].mul=sgt[lc].mul*sgt[cur].mul%p;
29     sgt[rc].mul=sgt[rc].mul*sgt[cur].mul%p;
30     sgt[lc].del=(sgt[lc].del*sgt[cur].mul+sgt[cur].del)%p;
31     sgt[rc].del=(sgt[rc].del*sgt[cur].mul+sgt[cur].del)%p;
32     sgt[cur].mul=1; sgt[cur].del=0;
33 }
34
35 void plus(int cur,int l,int r,LL del){
36     if(l<=sgt[cur].le && sgt[cur].ri<=r){
37         sgt[cur].sum=(sgt[cur].sum+del*(sgt[cur].ri-sgt[cur].le))%p;
38         sgt[cur].del=(sgt[cur].del+del)%p;
39     }
40     else{
41         int lc=cur<<1,rc=lc|1,mid=(sgt[cur].le+sgt[cur].ri)>>1;
42         update(cur);
43         if(l<mid)
44             plus(lc,l,r,del);
45         if(r>mid)
46             plus(rc,l,r,del);
47         sgt[cur].sum=sgt[lc].sum+sgt[rc].sum;
48     }
49 }
50
51 void multi(int cur,int l,int r,LL mul){
52     if(l<=sgt[cur].le && sgt[cur].ri<=r){
53         sgt[cur].sum=(sgt[cur].sum*mul)%p;
54         sgt[cur].del=(sgt[cur].del*mul)%p;
55         sgt[cur].mul=(sgt[cur].mul*mul)%p;
56     }
57     else{
58         int lc=cur<<1,rc=lc|1,mid=(sgt[cur].le+sgt[cur].ri)>>1;
59         update(cur);
60         if(l<mid)
61             multi(lc,l,r,mul);
62         if(r>mid)
63             multi(rc,l,r,mul);
64         sgt[cur].sum=sgt[lc].sum+sgt[rc].sum;
65     }
66 }
67
68 LL query(int cur,int l,int r){
69     if(l<=sgt[cur].le && sgt[cur].ri<=r){
70         return sgt[cur].sum;
71     }
72     else{
73         int lc=cur<<1,rc=lc|1,mid=(sgt[cur].le+sgt[cur].ri)>>1;
74         update(cur);
75         LL sum=0;
```

```
76          if(l<mid)
77              sum=(sum+query(lc,l,r))%p;
78          if(r>mid)
79              sum=(sum+query(rc,l,r))%p;
80          return sum;
81      }
82  }
83
84  int main(){
85      scanf("%d %d %d",&n,&m,&p);
86      for(int i=1;i<=n;i++)
87          scanf("%d", &a[i]);
88      built(1,1,n+1);
89      while(m--){
90          int opt,l,r;
91          LL k;
92          scanf("%d", &opt);
93          if(opt==1){
94              scanf("%d %d %lld", &l, &r, &k);
95              multi(1,l,r+1,k);
96          }
97          else if(opt==2){
98              scanf("%d %d %lld", &l, &r, &k);
99              plus(1,l,r+1,k);
100         }
101         else{
102             scanf("%d %d", &l, &r);
103             printf("%lld\n", query(1,l,r+1));
104         }
105
106     }
107     return 0;
108 }
```

4.9 扫描线 Scanline

```
1   #include <cstdio>
2   #include <algorithm>
3   #define LL long long
4   #define MAXN 100005
5   using namespace std;
6
7   struct line{
8       int x,y1,y2,sign;
9       bool operator<(line b)const{
10          if(x!=b.x) return x<b.x;
11          else return sign>b.sign;
12      }
13  }li[MAXN<<1];
14
15  struct node{
16      int le,ri;
```

```
17      int cnt,len;
18  }sgt[MAXN<<3];
19
20  int dy[MAXN<<1];
21
22  void pushup(int cur){
23      if(sgt[cur].cnt)
24          sgt[cur].len=dy[sgt[cur].ri]-dy[sgt[cur].le];
25      else if(sgt[cur].le<sgt[cur].ri-1)
26          sgt[cur].len=sgt[cur<<1].len+sgt[cur<<1|1].len;
27      else sgt[cur].len=0;
28  }
29
30  void build(int cur,int l,int r){
31      sgt[cur].le=l, sgt[cur].ri=r;
32      sgt[cur].cnt=sgt[cur].len=0;
33      if(l<r-1){
34          build(cur<<1,l,(l+r)>>1);
35          build(cur<<1|1,(l+r)>>1,r);
36      }
37  }
38
39  void modify(int cur,int l,int r,int sign){
40      if(l<=sgt[cur].le && sgt[cur].ri<=r){
41          sgt[cur].cnt+=sign;
42      }
43      else{
44          int mid=(sgt[cur].le+sgt[cur].ri)>>1;
45          if(l<mid) modify(cur<<1,l,r,sign);
46          if(r>mid) modify(cur<<1|1,l,r,sign);
47      }
48      pushup(cur);
49  }
50
51
52  int main(){
53      int n,cnt;
54      scanf("%d", &n);
55      cnt=0;
56      for(int i=1;i<=n;i++){
57          int x1,y1,x2,y2;
58          scanf("%d %d %d %d", &x1, &y1, &x2, &y2);
59          li[(i<<1)-1].x=x1, li[i<<1].x=x2;
60          li[(i<<1)-1].y1=li[i<<1].y1=y1;
61          li[(i<<1)-1].y2=li[i<<1].y2=y2;
62          li[(i<<1)-1].sign=1, li[i<<1].sign=-1;
63          dy[++cnt]=y1, dy[++cnt]=y2;
64      }
65      sort(dy+1,dy+cnt+1);
66      cnt=unique(dy+1,dy+cnt+1)-dy-1;
67      for(int i=1;i<=(n<<1);i++){
68          li[i].y1=lower_bound(dy+1,dy+cnt+1,li[i].y1)-dy;
69          li[i].y2=lower_bound(dy+1,dy+cnt+1,li[i].y2)-dy;
```

```
    }
    sort(li+1,li+(n<<1)+1);
    build(1,1,cnt);
    LL sum=0;
    for(int i=1;i<(n<<1);i++){
        modify(1,li[i].y1,li[i].y2,li[i].sign);
        sum+=1LL*sgt[1].len*(li[i+1].x-li[i].x);
    }
    printf("%lld\n", sum);
    return 0;
}
```

4.10 zkw 线段树 ZKW_Segment_Tree

```
#include <cstdio>
#include <cstring>
#define LL long long
#define MAXN 100005

struct node{
    LL sum,tag;
}sgt[MAXN<<2];

int M;

int a[MAXN];

void built(int n){
    for(M=1;M<n+2;M<<=1);
    memset(sgt+M,0,M*sizeof(sgt[0]));
    for(int i=1;i<=n;i++)
        sgt[M+i].sum=a[i];
    for(int i=M-1;i;i--)
        sgt[i].sum=sgt[i<<1].sum+sgt[i<<1|1].sum;
}

void modify(int l,int r,LL del){
    LL len=1,lc=0,rc=0;
    for(l=l+M-1,r=r+M+1;l^r^1;l>>=1,r>>=1,len<<=1){
        if(~l&1) sgt[l+1].tag+=del, lc+=len;
        if(r&1) sgt[r-1].tag+=del, rc+=len;
        sgt[l>>1].sum+=del*lc;
        sgt[r>>1].sum+=del*rc;
    }
    for(lc+=rc,l>>=1;l;l>>=1)
        sgt[l].sum+=del*lc;
}

LL query(int l,int r){
    LL res=0,len=1,lc=0,rc=0;
    for(l=l+M-1,r=r+M+1;l^r^1;l>>=1,r>>=1,len<<=1){
        if(~l&1) res+=sgt[l+1].sum+sgt[l+1].tag*len, lc+=len;
```

```
        if(r&1) res+=sgt[r-1].sum+sgt[r-1].tag*len, rc+=len;
        res+=sgt[l>>1].tag*lc;
        res+=sgt[r>>1].tag*rc;
    }
    for(lc+=rc,l>>=1;l;l>>=1)
        res+=sgt[l].tag*lc;
    return res;
}

int main(){
    int n,m;
    scanf("%d %d", &n, &m);
    for(int i=1;i<=n;i++)
        scanf("%d", &a[i]);
    built(n);
    for(int i=1;i<=m;i++){
        int opt,x,y;
        scanf("%d %d %d", &opt, &x, &y);
        if(opt==1){
            int k;
            scanf("%d", &k);
            modify(x,y,k);
        }
        else{
            printf("%lld\n", query(x,y));
        }
    }
    return 0;
}
```

4.11 李超线段树 Li-Chao_Segment_Tree

```
#include <cstdio>
#include <algorithm>
#define N 39989
#define MAXN 40005
#define MAXT 160005
const double eps=1e-12;
const double inf=1e9;

using namespace std;

struct line{
    int l,r;
    double k,b;
    int id;
}sgt[MAXT];

double calc(line l,int x){return l.k*x+l.b;}

void modify(int cur,int l,int r,line li){
    if(li.l<=l && r<=li.r){
```

```
        if(calc(li,l)-calc(sgt[cur],l)>eps && calc(li,r)-calc(sgt[cur],r)>eps)
            sgt[cur]=li;
        else if(calc(li,l)-calc(sgt[cur],l)>eps || calc(li,r)-calc(sgt[cur],r)>eps){
            int mid=(l+r)>>1;
            if(calc(li,mid)-calc(sgt[cur],mid)>eps)
                swap(li,sgt[cur]);
            if(calc(li,l)-calc(sgt[cur],l)>eps)
                modify(cur<<1,l,mid,li);
            else modify(cur<<1|1,mid+1,r,li);
        }
    }
    else{
        int mid=(l+r)>>1;
        if(li.l<=mid) modify(cur<<1,l,mid,li);
        if(li.r>mid) modify(cur<<1|1,mid+1,r,li);
    }
}

line query(int cur,int l,int r,int x){
    if(l==r) return sgt[cur];
    else{
        int mid=(l+r)>>1;
        line t;
        if(x<=mid) t=query(cur<<1,l,mid,x);
        else t=query(cur<<1|1,mid+1,r,x);
        if(!t.id || calc(sgt[cur],x)-calc(t,x)>eps) return sgt[cur];
        else return t;
    }
}

void built(int cur,int l,int r){
    sgt[cur].k=sgt[cur].b=0;
    sgt[cur].l=1; sgt[cur].r=N;
    sgt[cur].id=0;
    if(l<r){
        int mid=(l+r)>>1;
        built(cur<<1,l,mid);
        built(cur<<1|1,mid+1,r);
    }
}

int n;

int main(){
    scanf("%d", &n);
    built(1,1,N);
    int last=0,id=0;
    for(int i=1;i<=n;i++){
        int opt;
        scanf("%d", &opt);
        if(opt==0){
            int x;
            scanf("%d", &x);
```

```
            x=(x+last-1)%N+1;
            printf("%d\n", last=query(1,1,N,x).id);
        }
        else{
            int x0,x1,y0,y1;
            scanf("%d %d %d %d", &x0,&y0,&x1,&y1);
            x0=(x0+last-1)%N+1;
            x1=(x1+last-1)%N+1;
            y0=(y0+last-1)%1000000000+1;
            y1=(y1+last-1)%1000000000+1;
            line t;
            t.id=++id;
            t.l=min(x0,x1); t.r=max(x0,x1);
            t.k=x1==x0?0:(double)(y1-y0)/(x1-x0);
            t.b=x1==x0?max(y0,y1):y0-t.k*x0;
            modify(1,1,N,t);
        }
    }
    return 0;
}
```

4.12 可并堆左偏树 Leftist_Tree

```
#include <cstdio>
#include <algorithm>
#define MAXN 100005

using namespace std;

int n,m;

struct node{
    int rt,lc,rc,dis,v;
}lt[MAXN];

int find(int x){
    if(lt[x].rt==x)
        return x;
    return lt[x].rt=find(lt[x].rt);
}

int merge(int x,int y){
    if(!x || !y) return x+y;
    if(lt[x].v>lt[y].v || (lt[x].v==lt[y].v && x>y)) swap(x,y);//后一个条件蜜汁优化？
    lt[x].rc=merge(lt[x].rc,y);
    lt[lt[x].rc].rt=x;
    if(lt[lt[x].lc].dis<lt[lt[x].rc].dis) swap(lt[x].lc,lt[x].rc);
    lt[x].dis=lt[lt[x].rc].dis+1;
    return x;
}

void pop(int x){
```

```
    lt[x].v=-1;
    lt[lt[x].lc].rt=lt[x].lc;
    lt[lt[x].rc].rt=lt[x].rc;
    lt[x].rt=merge(lt[x].lc,lt[x].rc);
}

int main(){
    scanf("%d %d", &n, &m);
    for(int i=1;i<=n;i++){
        scanf("%d", &lt[i].v);
        lt[i].rt=i;
        lt[i].lc=lt[i].rc=0;
        lt[i].dis=0;
    }
    lt[0].dis=0;
    for(int i=1;i<=m;i++){
        int opt;
        scanf("%d",&opt);
        if(opt==1){
            int x,y;
            scanf("%d %d", &x, &y);
            int rx=find(x),ry=find(y);
            if(lt[x].v==-1||lt[y].v==-1||rx==ry)
                continue;
            merge(rx,ry);
        }
        else{
            int x;
            scanf("%d", &x);
            if(lt[x].v==-1)
                printf("-1\n");
            else{
                int rx=find(x);
                printf("%d\n", lt[rx].v);
                pop(rx);
            }
        }
    }
    return 0;
}
```

4.13 Splay 树 Splay_Tree

```
#include <cstdio>
#define MAXN 100005
const int inf=0x3f3f3f3f;

int root,len;

struct node{
    int v,fa,ch[2],size,cnt;
}sp[MAXN];
```

```
10
11 int getch(int x) {return sp[sp[x].fa].ch[1]==x;}
12 void pushup(int x) {sp[x].size=sp[x].cnt+sp[sp[x].ch[0]].size+sp[sp[x].ch[1]].size;}
13
14 void rotate(int x){
15     int f=sp[x].fa, ff=sp[f].fa;
16     int k=getch(x);
17     sp[ff].ch[getch(f)]=x; sp[x].fa=ff;
18     sp[sp[x].ch[k^1]].fa=f; sp[f].ch[k]=sp[x].ch[k^1];
19     sp[x].ch[k^1]=f; sp[f].fa=x;
20     pushup(f); pushup(x);
21 }
22
23 void splay(int x,int goal=0){
24     for(int f;(f=sp[x].fa)!=goal;rotate(x)){
25         if(sp[f].fa!=goal)
26             rotate(getch(x)==getch(f)?f:x);
27     }
28     if(!goal) root=x;
29 }
30
31 void insert(int x){
32     int cur=root,f=0;
33     while(cur&&sp[cur].v!=x){
34         f=cur;
35         cur=sp[cur].ch[x>sp[cur].v];
36     }
37     if(cur)
38         sp[cur].cnt++;
39     else{
40         cur=++len;
41         sp[f].ch[x>sp[f].v]=cur;
42         sp[cur].ch[0]=sp[cur].ch[1]=0;
43         sp[cur].fa=f;
44         sp[cur].v=x;
45         sp[cur].cnt=sp[cur].size=1;
46     }
47     splay(cur);
48 }
49
50 void find(int x){
51     int cur=root;
52     while(x!=sp[cur].v && sp[cur].ch[x>sp[cur].v])
53         cur=sp[cur].ch[x>sp[cur].v];
54     splay(cur);
55 }
56
57 int kth(int x){
58     if(sp[root].size<x) return 0;
59     int cur=root;
60     while(1){
61         if(x<=sp[sp[cur].ch[0]].size)
62             cur=sp[cur].ch[0];
```

```
63          else if(x>sp[sp[cur].ch[0]].size+sp[cur].cnt){
64              x-=sp[sp[cur].ch[0]].size+sp[cur].cnt;
65              cur=sp[cur].ch[1];
66          }
67          else return sp[cur].v;
68      }
69  }
70
71  int pre(int x){
72      find(x);
73      if(x>sp[root].v) return root;
74      int cur=sp[root].ch[0];
75      while(sp[cur].ch[1])
76          cur=sp[cur].ch[1];
77      return cur;
78  }
79
80  int succ(int x){
81      find(x);
82      if(x<sp[root].v) return root;
83      int cur=sp[root].ch[1];
84      while(sp[cur].ch[0])
85          cur=sp[cur].ch[0];
86      return cur;
87  }
88
89  void erase(int x){
90      int last=pre(x),next=succ(x),del;
91      splay(last);splay(next,last);
92      del=sp[next].ch[0];
93      if(sp[del].cnt>1){
94          sp[del].cnt--;
95          splay(del);
96      }
97      else{
98          sp[next].ch[0]=0;
99          sp[del].fa=0;
100         sp[del]=sp[len];
101         int f=sp[del].fa;
102         sp[f].ch[(sp[f].ch[1]==len)]=del;
103         sp[sp[del].ch[0]].fa=del;
104         sp[sp[del].ch[1]].fa=del;
105         if(root==len) root=del;
106         len--;
107
108     }
109 }
110
111
112 int n;
113 int main(){
114     scanf("%d", &n);
115     root=0;len=0;
```

```
    insert(-inf);insert(inf);
    sp[0].size=0;
    for(int i=1;i<=n;i++){
        int opt,x;
        scanf("%d %d", &opt, &x);
        if(opt==1){
            insert(x);
        }
        else if(opt==2){
            erase(x);
        }
        else if(opt==3){
            find(x);
            printf("%d\n", sp[sp[root].ch[0]].size);
        }
        else if(opt==4){
            printf("%d\n", kth(x+1));
        }
        else if(opt==5){
            printf("%d\n", sp[pre(x)].v);
        }
        else{
            printf("%d\n", sp[succ(x)].v);
        }
    }
    return 0;
}
```

4.14 Splay 树 Splay__Tree__Flip

```
#include <cstdio>
#include <algorithm>
#define MAXN 100005
const int inf=0x3f3f3f3f;

using namespace std;

int root,len,a[MAXN];

struct node{
    int v,fa,ch[2],size,cnt,tag;
}sp[MAXN];

int getch(int x) {return sp[sp[x].fa].ch[1]==x;}
void pushup(int x) {sp[x].size=sp[x].cnt+sp[sp[x].ch[0]].size+sp[sp[x].ch[1]].size;}

void pushdown(int x){
    if(sp[x].tag){
        sp[sp[x].ch[0]].tag^=1;
        sp[sp[x].ch[1]].tag^=1;
        swap(sp[x].ch[0],sp[x].ch[1]);
        sp[x].tag=0;
```

```
    }
}

void rotate(int x){
    int f=sp[x].fa, ff=sp[f].fa;
    int k=getch(x);
    sp[ff].ch[getch(f)]=x; sp[x].fa=ff;
    sp[sp[x].ch[k^1]].fa=f; sp[f].ch[k]=sp[x].ch[k^1];
    sp[x].ch[k^1]=f; sp[f].fa=x;
    pushup(f); pushup(x);
}

void splay(int x,int goal=0){
    for(int f;(f=sp[x].fa)!=goal;rotate(x)){
        if(sp[f].fa!=goal)
            rotate(getch(x)==getch(f)?f:x);
    }
    if(!goal) root=x;
}

int find(int x){
    int cur=root;
    while(1){
        pushdown(cur);
        if(x<=sp[sp[cur].ch[0]].size)
            cur=sp[cur].ch[0];
        else if(x>sp[sp[cur].ch[0]].size+sp[cur].cnt){
            x-=sp[sp[cur].ch[0]].size+sp[cur].cnt;
            cur=sp[cur].ch[1];
        }
        else return cur;
    }
}

int built(int f,int l,int r){
    if(l>r) return 0;
    int mid=(l+r)>>1, cur=++len;
    sp[cur].fa=f;
    sp[cur].cnt=1;
    sp[cur].v=a[mid];
    sp[cur].tag=0;
    sp[cur].ch[0]=built(cur,l,mid-1);
    sp[cur].ch[1]=built(cur,mid+1,r);
    pushup(cur);
    return cur;
}

void flip(int l,int r){
    int last=find(l-1),next=find(r+1);
    splay(last);splay(next,last);
    sp[sp[sp[root].ch[1]].ch[0]].tag^=1;
}

```

```
76 void dfs(int cur){
77     pushdown(cur);
78     if(sp[cur].ch[0]) dfs(sp[cur].ch[0]);
79     if(sp[cur].v!=-inf && sp[cur].v!=inf) printf("%d ", sp[cur].v);
80     if(sp[cur].ch[1]) dfs(sp[cur].ch[1]);
81 }
82
83 int n,m;
84
85 int main(){
86     scanf("%d %d", &n, &m);
87     for(int i=1;i<=n;i++) a[i+1]=i;
88     a[1]=-inf;a[n+2]=inf;
89     len=0;
90     root=built(0,1,n+2);
91     sp[0].size=0;
92     for(int i=1;i<=m;i++){
93         int l,r;
94         scanf("%d %d", &l, &r);
95         flip(l+1,r+1);
96     }
97     dfs(root);
98     return 0;
99 }
```

4.15 Splay 树 Splay_Tree_Dye&Flip

```
1  #include <cstdio>
2  #include <cstring>
3  #include <algorithm>
4  #define MAXN 500005
5  const int inf=0x3f3f3f3f;
6  using namespace std;
7
8  struct node{
9      int v,fa,ch[2],cnt;//basic
10     int size,sum,lm,rm,mm;//pushup
11     int flip,color;//pushdown
12 }sp[MAXN];
13
14 int a[MAXN],len,root,recy[MAXN],rlen;
15
16 int getch(int x){return sp[sp[x].fa].ch[1]==x;}
17
18 void pushup(int x){
19     int lc=sp[x].ch[0],rc=sp[x].ch[1];
20     sp[x].size=sp[lc].size+sp[rc].size+sp[x].cnt;
21     sp[x].sum=sp[lc].sum+sp[rc].sum+sp[x].v;
22     sp[x].lm=max(sp[lc].lm, sp[lc].sum+sp[x].v+sp[rc].lm);
23     sp[x].rm=max(sp[rc].rm, sp[rc].sum+sp[x].v+sp[lc].rm);
24     sp[x].mm=max(max(sp[lc].mm,sp[rc].mm),sp[lc].rm+sp[x].v+sp[rc].lm);
25 }
```

```
26
27 void pushdown(int x){
28     int lc=sp[x].ch[0],rc=sp[x].ch[1];
29     if(sp[x].color!=inf){
30         if(lc){
31             sp[lc].v=sp[lc].color=sp[x].color;
32             sp[lc].sum=sp[lc].size*sp[x].color;
33         }
34         if(rc){
35             sp[rc].v=sp[rc].color=sp[x].color;
36             sp[rc].sum=sp[rc].size*sp[x].color;
37         }
38         if(sp[x].color>0){
39             if(lc) sp[lc].lm=sp[lc].rm=sp[lc].mm=sp[lc].sum;
40             if(rc) sp[rc].lm=sp[rc].rm=sp[rc].mm=sp[rc].sum;
41         }
42         else{
43             if(lc) {sp[lc].lm=sp[lc].rm=0; sp[lc].mm=sp[lc].v;}
44             if(rc) {sp[rc].lm=sp[rc].rm=0; sp[rc].mm=sp[rc].v;}
45         }
46         sp[x].color=inf;
47         sp[x].flip=0;
48     }
49     else if(sp[x].flip){
50         if(lc){
51             sp[lc].flip^=1;
52             swap(sp[lc].ch[0],sp[lc].ch[1]);
53             swap(sp[lc].lm,sp[lc].rm);
54         }
55         if(rc){
56             sp[rc].flip^=1;
57             swap(sp[rc].ch[0],sp[rc].ch[1]);
58             swap(sp[rc].lm,sp[rc].rm);
59         }
60         sp[x].flip=0;
61     }
62 }
63
64 void rotate(int x){
65     int f=sp[x].fa, ff=sp[f].fa;
66     int k=getch(x);
67     sp[ff].ch[getch(f)]=x; sp[x].fa=ff;
68     sp[sp[x].ch[k^1]].fa=f; sp[f].ch[k]=sp[x].ch[k^1];
69     sp[x].ch[k^1]=f; sp[f].fa=x;
70     pushup(f); pushup(x);
71 }
72
73 void splay(int x,int goal=0){
74     for(int f;(f=sp[x].fa)!=goal;rotate(x)){
75         if(sp[f].fa!=goal)
76             rotate(getch(x)==getch(f)?f:x);
77     }
78     if(!goal) root=x;
```

```
}

int find(int x){
    int cur=root;
    while(1){
        pushdown(cur);
        if(x<=sp[sp[cur].ch[0]].size)
            cur=sp[cur].ch[0];
        else if(x>sp[sp[cur].ch[0]].size+sp[cur].cnt){
            x-=sp[sp[cur].ch[0]].size+sp[cur].cnt;
            cur=sp[cur].ch[1];
        }
        else return cur;
    }
}

int built(int f,int l,int r){
    if(l>r) return 0;
    int mid=(l+r)>>1, cur=rlen?recy[rlen--]:++len;
    sp[cur].v=a[mid];
    sp[cur].fa=f;
    sp[cur].cnt=1;
    sp[cur].flip=0;
    sp[cur].color=inf;
    sp[cur].ch[0]=built(cur,l,mid-1);
    sp[cur].ch[1]=built(cur,mid+1,r);
    pushup(cur);
    return cur;
}

void insert(int pos,int tot){
    int l=find(pos),r=find(pos+1);
    splay(l);splay(r,l);
    sp[r].ch[0]=built(r,1,tot);
    pushup(r); pushup(l);
}

void recycle(int x){
    if(!x) return;
    recycle(sp[x].ch[0]);
    recycle(sp[x].ch[1]);
    sp[sp[x].fa].ch[getch(x)]=0;
    recy[++rlen]=x;
}

void erase(int pos,int tot){
    int l=find(pos-1),r=find(pos+tot);
    splay(l);splay(r,l);
    recycle(sp[r].ch[0]);
    pushup(r); pushup(l);
}

void dye(int pos,int tot,int c){
```

```
132     int l=find(pos-1),r=find(pos+tot);
133     splay(l);splay(r,l);
134     int x=sp[r].ch[0];
135     sp[x].color=c;
136     sp[x].v=c;
137     sp[x].sum=sp[x].size*c;
138     if(c>0)
139         sp[x].lm=sp[x].rm=sp[x].mm=sp[x].sum;
140     else{
141         sp[x].lm=sp[x].rm=0;
142         sp[x].mm=sp[x].v;
143     }
144     pushup(r); pushup(l);
145 }
146 
147 void reverse(int pos,int tot){
148     int l=find(pos-1),r=find(pos+tot);
149     splay(l);splay(r,l);
150     int x=sp[r].ch[0];
151     sp[x].flip^=1;
152     swap(sp[x].ch[0],sp[x].ch[1]);
153     swap(sp[x].lm,sp[x].rm);
154     pushup(r); pushup(l);
155 }
156 
157 int getsum(int pos,int tot){
158     int l=find(pos-1),r=find(pos+tot);
159     splay(l);splay(r,l);
160     return sp[sp[r].ch[0]].sum;
161 }
162 
163 int n,m;
164 
165 int main(){
166     scanf("%d %d", &n, &m);
167     for(int i=1;i<=n;i++){
168         scanf("%d", &a[i+1]);
169     }
170     memset(sp,0,sizeof(sp[0]));
171     sp[0].mm=a[1]=a[n+2]=-inf;
172     rlen=0;
173     len=0;
174     root=built(0,1,n+2);
175     for(int i=1;i<=m;i++){
176         char opt[10];
177         scanf("%s", opt);
178         if(opt[0]=='I'){//Insert
179             int pos,tot;
180             scanf("%d %d", &pos, &tot);
181             for(int i=1;i<=tot;i++)
182                 scanf("%d", &a[i]);
183             insert(pos+1,tot);
184         }
```

```
185         else if(opt[0]=='D'){//Delete
186             int pos,tot;
187             scanf("%d %d", &pos, &tot);
188             erase(pos+1,tot);
189         }
190         else if(opt[2]=='K'){//Make-Same
191             int pos,tot,c;
192             scanf("%d %d %d", &pos, &tot, &c);
193             dye(pos+1,tot,c);
194         }
195         else if(opt[0]=='R'){//Reverse
196             int pos,tot;
197             scanf("%d %d", &pos, &tot);
198             reverse(pos+1,tot);
199         }
200         else if(opt[0]=='G'){//Get-Sum
201             int pos,tot;
202             scanf("%d %d", &pos, &tot);
203             printf("%d\n", getsum(pos+1,tot));
204         }
205         else if(opt[0]=='M'){//Max-Sum
206             printf("%d\n", sp[root].mm);
207         }
208     }
209     return 0;
210 }
```

5 数论

5.1 乘法逆元 Multiplicative_Inverse_Modulo

```cpp
#include <cstdio>
#define MAXP 20000530

int n,p,inv[MAXP];

int main(){
    scanf("%d %d", &n, &p);
    inv[1]=1;
    for(int i=2;i<=n;i++){
        inv[i]=1LL*(p-p/i)*inv[p%i]%p;
    }
    for(int i=1;i<=n;i++)
        printf("%d\n", inv[i]);
    return 0;
}
```

5.2 卢卡斯 Lucas

```cpp
#include <cstdio>
#define LL long long
#define MAXP 100005

LL f[MAXP];

LL binpow(LL x,LL y,LL mod){
    LL r=1%mod;
    for(;y;y>>=1){
        if(y&1) r=r*x%mod;
        x=x*x%mod;
    }
    return r;
}

void pre(LL p){
    f[0]=1;
    for(int i=1;i<=p-1;i++) f[i]=f[i-1]*i%p;
}

LL C(LL x,LL y,LL p){
    if(x<y) return 0;
    return f[x]*binpow(f[y],p-2,p)%p*binpow(f[x-y],p-2,p)%p;
}

LL lucas(LL x,LL y,LL p){
    if(!y) return 1;
    return C(x%p,y%p,p)*lucas(x/p,y/p,p)%p;
}

```

```
int main(){
    int T;
    scanf("%d", &T);
    while(T--){
        LL n,m,p;
        scanf("%lld %lld %lld", &n, &m, &p);
        pre(p);
        printf("%lld\n", lucas(n+m,m,p));
    }
    return 0;
}
```

5.3 拓展欧几里得 Exgcd

```
#include <cstdio>
#include <cmath>
#define LL long long

LL exgcd(LL a,LL b,LL &x,LL &y){
    if(!b || !a){
        x=(a!=0);y=(b!=0);
        return a+b;
    }
    LL g=exgcd(b,a%b,y,x);
    y=y-(a/b)*x;
    return g;
}

int main(){
    int T;
    scanf("%d", &T);
    while(T--){
        LL a,b,c,x,y,g;
        scanf("%lld %lld %lld", &a, &b, &c);
        g=exgcd(a,b,x,y);
        if(c%g){
            printf("-1\n");
            continue;
        }
        a/=g;b/=g;c/=g;x*=c;y*=c;
        LL kl=ceil((double)(-x+1)/b),kr=floor((double)(y-1)/a);
        if(kr<kl){
            printf("%lld %lld\n",(x+kl*b),(y-kr*a));
        }
        else{
            printf("%lld %lld %lld %lld %lld\n",kr-kl+1,(x+kl*b),(y-kr*a),(x+kr*b),(y-kl*
                a));
        }
    }
    return 0;
}
```

5.4 拓展欧拉定理 Ex_Euler_Theorem-Automaton

```cpp
#include <bits/stdc++.h>
using namespace std;
#define LL long long
#define MAXN 100000005

LL binpow(LL x,LL y,LL m){
    int r=1%m;
    x%=m;
    while(y){
        if(y&1) r=1LL*r*x%m;
        x=1LL*x*x%m;
        y>>=1;
    }
    return r;
}

void solve() {
    int a,m;
    scanf("%d %d", &a, &m);
    int mm=m,phi=m;
    for(int i=2;i*i<=m;i++) {
        if(mm%i==0){
            while(mm%i==0) mm/=i;
            phi=phi/i*(i-1);
        }
    }
    if(mm>1) phi=phi/mm*(mm-1);
    char c;
    while(!isdigit(c=getchar()));
    int b=c-'0';
    bool flag=0;
    while(isdigit(c=getchar())){
        b=10*b+c-'0';
        if(b>=phi) b%=phi, flag=1;
    }
    if(flag) b+=phi;
    printf("%lld\n", binpow(a,b,m));

}

int main() {
    int T=1;
    // scanf("%d", &T);
    while(T--) {
        solve();
    }
    return 0;
}
```

5.5 欧拉筛 Eular_Sieve

```cpp
#include <cstdio>
#include <cstring>
#define MAXN 2000005

int cnt,p[MAXN];
bool inp[MAXN];
int phi[MAXN],mu[MAXN];

void eular_sieve(int n){
    cnt=0;
    memset(inp,0,(n+1)*sizeof(inp[0]));
    inp[0]=inp[1]=1;
    phi[1]=1;
    mu[1]=1;
    for(int i=2;i<=n;i++){
        if(!inp[i]) p[++cnt]=i, phi[i]=i-1, mu[i]=-1;
        for(int j=1;j<=cnt && i*p[j]<=n;j++){
            inp[i*p[j]]=1;
            if(i%p[j]){
                phi[i*p[j]]=phi[i]*(p[j]-1);
                mu[i*p[j]]=-mu[i];
            }
            else{
                phi[i*p[j]]=phi[i]*p[j];
                mu[i*p[j]]=0;
                break;
            }
        }
    }
}

int main(){
    int n,m;
    scanf("%d %d", &n, &m);
    eular_sieve(n);
    for(int i=1;i<=m;i++){
        int k;
        scanf("%d",&k);
        printf("%d\n", p[k]);
    }
}
```

5.6 杜教筛 Dujiao_Sieve

```cpp
#include <cstdio>
#include <cstring>
#define LL long long
#define MAXR R+5
#define R 2000000//r=n^(2/3)

```

```
int n,r,p[MAXR],cnt;
bool inp[MAXR];

LL phi[MAXR],sphi[MAXR],sphir[MAXR],mu[MAXR],smu[MAXR],smur[MAXR];

void pre(){
    memset(inp+1,0,r*sizeof(inp[0]));
    inp[0]=inp[1]=1;
    cnt=0;
    sphi[1]=phi[1]=1;
    smu[1]=mu[1]=1;
    for(int i=2;i<=r;i++){
        if(!inp[i]) p[++cnt]=i, phi[i]=i-1, mu[i]=-1;
        for(int j=1;j<=cnt&&i*p[j]<=r;j++){
            inp[i*p[j]]=1;
            if(i%p[j]){
                phi[i*p[j]]=phi[i]*(p[j]-1);
                mu[i*p[j]]=-mu[i];
            }
            else{
                phi[i*p[j]]=phi[i]*p[j];
                mu[i*p[j]]=0;
                break;
            }
        }
        sphi[i]=sphi[i-1]+phi[i];
        smu[i]=smu[i-1]+mu[i];
    }
}

LL sumphi(LL x){
    if(x<=r) return sphi[x];
    else if(sphir[n/x]) return sphir[n/x];
    LL &sx=sphir[n/x];
    sx=x*(x+1)/2;
    for(LL i=2;i<=x;i++){
        LL t=x/i,j=x/t;
        sx-=(j-i+1)*sumphi(t);
        i=j;
    }
    return sx;
}

LL summu(LL x){
    if(x<=r) return smu[x];
    else if(smur[n/x]) return smur[n/x];
    LL &sx=smur[n/x];
    sx=1;
    for(LL i=2;i<=x;i++){
        LL t=x/i,j=x/t;
        sx-=(j-i+1)*summu(t);
        i=j;
    }
```

```
    return sx;
}

int main(){
    int T;
    scanf("%d", &T);
    r=R;
    pre();
    while(T--){
        scanf("%d", &n);
        memset(sphir+1,0,(n/r)*sizeof(sphir[0]));
        memset(smur+1,0,(n/r)*sizeof(smur[0]));
        printf("%lld %lld\n", sumphi(n), summu(n));
    }
    return 0;
}
```

5.7 求原根 Get_Primitive_Root

```
#include <cstdio>
#include <vector>

using namespace std;

int p;
vector<int> v;

int binpow(int x,int y,int mod){
    int r=1%mod;
    while(y){
        if(y&1) r=(1LL*r*x)%mod;
        x=(1LL*x*x)%mod;
        y>>=1;
    }
    return r;
}

int main(){
    scanf("%d", &p);
    int pp=p-1;
    for(int i=2;i*i<=pp;i++){
        if(pp%i==0){
            v.push_back(i);
            while(pp%i==0) pp/=i;
        }
    }
    if(pp>1) v.push_back(pp);
    for(int g=2;;g++){
        bool isg=true;
        for(int d:v){
            if(binpow(g,(p-1)/d,p)==1){
                isg=false;
```

```
34                 break;
35             }
36         }
37         if(isg){
38             printf("%d\n", g);
39             break;
40         }
41     }
42     return 0;
43 }
```

5.8 贝祖引理 Bezout_Lemma

```
1  #include <cstdio>
2
3  int gcd(int x,int y){
4      if(!x || !y) return x+y;
5      return gcd(y,x%y);
6  }
7
8  int n;
9
10 int main(){
11     scanf("%d", &n);
12     int g;
13     scanf("%d", &g);
14     if(g<0)g=-g;
15     for(int i=2;i<=n;i++){
16         int t;
17         scanf("%d", &t);
18         if(t<0) t=-t;
19         g=gcd(g,t);
20     }
21     printf("%d\n", g);
22     return 0;
23 }
```

5.9 除法分块 Block_Division

```
1  #include <cstdio>
2  #define LL long long
3
4  LL n;
5
6  int main(){
7      scanf("%lld", &n);
8      LL ans=0;
9      for(LL i=1;i<=n;i++){
10         LL t=n/i,j=n/t;
11         ans+=(j-i+1)*t;
12         i=j;
```

```
    }
    printf("%lld\n", ans);
    return 0;
}
```

6 网络流

6.1 最大费用流 Minimum-Cost_Flow_Edmonds-Karp

```cpp
#include <cstdio>
#include <cstring>
#include <queue>
#include <algorithm>
#define MAXN 5005
#define MAXM 50005

const int inf = 0x3f3f3f3f;

using namespace std;

int n,m,s,t,tot,head[MAXN],dis[MAXN],maxflow,mincost;
bool inque[MAXN];

struct edge{//残量网络 residual network
    int to,cf,next,dis;//Cf:residual capacity
}e[(MAXM<<1)+1];

struct node{
    int fr,edge;
}pre[MAXN];

void add(int x,int y,int f,int d){
    tot++;
    e[tot].cf=f;
    e[tot].dis=d;
    e[tot].to=y;
    e[tot].next=head[x];
    head[x]=tot;
}

bool spfa(){//SPFA
    memset(dis+1,inf,n*sizeof(dis[0]));
    memset(inque+1,0,n*sizeof(inque[0]));
    pre[t].fr=0;
    queue<int> q;
    dis[s]=0;
    q.push(s);
    inque[s]=1;
    while(!q.empty()){
        int u=q.front();q.pop();
        inque[u]=0;
        for(int p=head[u];p;p=e[p].next){
            int v=e[p].to;
            if(e[p].cf && dis[v]>dis[u]+e[p].dis){
                dis[v]=dis[u]+e[p].dis;
                pre[v].fr=u;
                pre[v].edge=p;
                if(!inque[v]){
```

```
                q.push(v);
                inque[v]=1;
            }
        }
    }
    return pre[t].fr!=0;
}

int min_flow(){
    int mn=inf;
    for(int u=t;u!=s;u=pre[u].fr){
        mn=min(mn,e[pre[u].edge].cf);
    }
    for(int u=t;u!=s;u=pre[u].fr){
        e[pre[u].edge].cf-=mn;
        e[pre[u].edge^1].cf+=mn;
    }
    return mn;
}

void edmonds_karp(){
    maxflow=0,mincost=0;
    while(spfa()){
        int flow=min_flow();
        maxflow+=flow;
        mincost+=flow*dis[t];
    }
}

int main(){
    scanf("%d %d %d %d", &n, &m, &s, &t);
    tot=1;
    memset(head+1,0,n*sizeof(head[0]));
    for(int i=1;i<=m;i++){
        int f,g,w,d;
        scanf("%d %d %d %d",&f,&g,&w,&d);
        add(f,g,w,d);
        add(g,f,0,-d);
    }
    edmonds_karp();
    printf("%d %d\n",maxflow,mincost);
    return 0;
}
```

6.2 最大流 Maximum_Flow_Edmonds-Karp

```
#include <cstdio>
#include <cstring>
#include <queue>
#include <algorithm>
#define MAXN 10005
```

```cpp
#define MAXM 100005

using namespace std;

const int inf = 0x3f3f3f3f;

int n,m,s,t,tot,head[MAXN],vis[MAXN];

struct edge{
    int to,cf,next;
}e[MAXM<<1];

struct node{
    int fr,edge;
}pre[MAXN];

void add(int x,int y,int z){
    tot++;
    e[tot].cf=z;
    e[tot].to=y;
    e[tot].next=head[x];
    head[x]=tot;
}

bool find_augment(){
    memset(pre+1,0,n*sizeof(pre[0]));
    memset(vis+1,0,n*sizeof(vis[0]));
    queue<int> q;
    vis[s]=1;
    q.push(s);
    while(!q.empty()){
        int u=q.front();q.pop();
        for(int p=head[u];p;p=e[p].next){
            int v=e[p].to;
            if(!vis[v] && e[p].cf){
                pre[v].fr=u;
                pre[v].edge=p;
                vis[v]=1;
                q.push(v);
                if(v==t) return true;
            }
        }
    }
    return false;
}

int min_flow(){
    int mn=inf;
    for(int u=t;u!=s;u=pre[u].fr){
        mn=min(mn,e[pre[u].edge].cf);
    }
    for(int u=t;u!=s;u=pre[u].fr){
        e[pre[u].edge].cf-=mn;
```

```
        e[pre[u].edge^1].cf+=mn;
    }
    return mn;
}

int edmonds_karp(){
    int flow=0;
    while(find_augment()){
        flow+=min_flow();
    }
    return flow;
}

int main(){
    scanf("%d %d %d %d", &n, &m, &s, &t);
    tot=1;
    memset(head+1,0,n*sizeof(head[0]));
    for(int i=1;i<=m;i++){
        int f,g,w;
        scanf("%d %d %d",&f,&g,&w);
        add(f,g,w);
        add(g,f,0);
    }
    printf("%d\n", edmonds_karp());
    return 0;
}
```

6.3 最大流 Maximum_Flow_Dinic

```
#include <cstdio>
#include <cstring>
#include <queue>
#include <algorithm>
#define MAXN 10005
#define MAXM 100005

const int inf = 0x3f3f3f3f;

using namespace std;

int n,m,s,t,tot,head[MAXN],lb[MAXN],cur[MAXN];

struct edge{//残量网络 residual network
    int to,cf,next;//Cf:residual capacity
}e[(MAXM<<1)+1];

void add(int x,int y,int z){
    tot++;
    e[tot].cf=z;
    e[tot].to=y;
    e[tot].next=head[x];
    head[x]=tot;
```

```
}

bool label_vertex(){//BFS
    memset(lb+1,0,n*sizeof(lb[0]));
    queue<int> q;
    lb[s]=1;
    q.push(s);
    while(!q.empty()){
        int u=q.front();q.pop();
        for(int p=head[u];p;p=e[p].next){
            int v=e[p].to;
            if(e[p].cf && !lb[v]){
                lb[v]=lb[u]+1;
                q.push(v);
                if(v==t) return true;
            }
        }
    }
    return false;
}

int multi_augment(int u,int lim){//DFS 多路增广
    if(u == t) return lim;

    int used=0;
    for(int& p=cur[u];p;p=e[p].next){
        int v=e[p].to;
        if(e[p].cf && lb[v]==lb[u]+1){
            int rest=multi_augment(v,min(lim-used,e[p].cf));
            used+=rest;
            e[p].cf-=rest;
            e[p^1].cf+=rest;
            if(used==lim) break;
        }
    }
    return used;
}

int dinic(){
    int flow=0;
    while(label_vertex()){//BFS 标记
        for(int i=1;i<=n;i++) cur[i]=head[i];//当前弧优化
        flow+=multi_augment(s,inf);//DFS 顺着标记找增广路
    }
    return flow;
}

int main(){
    scanf("%d %d %d %d", &n, &m, &s, &t);
    tot=1;
    memset(head+1,0,n*sizeof(head[0]));
    for(int i=1;i<=m;i++){
        int f,g,w;
```

```
77          scanf("%d %d %d",&f,&g,&w);
78          add(f,g,w);
79          add(g,f,0);
80      }
81      printf("%d\n", dinic());
82      return 0;
83  }
```

6.4 二分题最大匹配 Bipartite_Graph_Maximum_Matching_Dinic

```
1   #include <cstdio>
2   #include <cstring>
3   #include <queue>
4   #define MAXN 2005
5   #define MAXM 1000005
6
7   using namespace std;
8
9   const int inf=0x3f3f3f3f;
10
11  struct edge{
12      int to,cf,next;
13  }e[MAXM<<1];
14
15  int n,n1,n2,m,s,t;
16  int tot,head[MAXN],cur[MAXN],lbl[MAXN];
17
18  void add(int x,int y,int z){
19      tot++;
20      e[tot].to=y;
21      e[tot].cf=z;
22      e[tot].next=head[x];
23      head[x]=tot;
24  }
25
26  bool bfs(){
27      memset(lbl+1,0,n*sizeof(lbl[0]));
28      lbl[t]=1;
29      queue<int> q;
30      q.push(t);
31      while(!q.empty()){
32          int u=q.front();q.pop();
33          for(int p=head[u];p;p=e[p].next){
34              int v=e[p].to;
35              if(e[p^1].cf && !lbl[v]){
36                  lbl[v]=lbl[u]+1;
37                  q.push(v);
38                  if(v==s) return true;
39              }
40          }
41      }
42      return lbl[s]!=0;
```

```
}

int dfs(int u,int lim){
    if(u==t)return lim;

    int used=0;
    for(int& p=cur[u];p;p=e[p].next){
        int v=e[p].to;
        if(e[p].cf && lbl[v]==lbl[u]-1){
            int rest=dfs(v,min(lim-used,e[p].cf));
            used+=rest;
            e[p].cf-=rest;
            e[p^1].cf+=rest;
            if(used==lim) break;
        }
    }
    return used;
}

int dinic(){
    int flow=0;
    while(bfs()){
        for(int i=1;i<=n;i++)
            cur[i]=head[i];
        flow+=dfs(s,inf);
    }
    return flow;
}

int main(){
    scanf("%d %d %d", &n1, &n2, &m);
    n=n1+n2+2;//n个点
    s=n-1;t=n;
    tot=1;
    memset(head+1,0,n*sizeof(head[0]));
    for(int i=1;i<=m;i++){
        int f,g;
        scanf("%d %d",&f,&g);
        if(f>n1 || g>n2) continue;
        add(f,n1+g,1);
        add(n1+g,f,0);
    }
    for(int i=1;i<=n1;i++){
        add(s,i,1);
        add(i,s,0);
    }
    for(int i=n1+1;i<=n1+n2;i++){
        add(i,t,1);
        add(t,i,0);
    }
    printf("%d\n", dinic());
    return 0;
}
```

6.5 二分图最大匹配 Bipartite_Graph_Maximum_Matching_Hungarian

```
#include <cstdio>
#include <cstring>
#define MAXN 1005
#define MAXM 1000005

struct node{
    int to,next;
}e[MAXM];

int n1,n2,m,head[MAXN],tot,dfn[MAXN],mat[MAXN];

void add(int x,int y){
    tot++;
    e[tot].to=y;
    e[tot].next=head[x];
    head[x]=tot;
}

bool augment(int x,int stamp){
    for(int p=head[x];p;p=e[p].next){
        int u=e[p].to;
        if(dfn[u] == stamp) continue;
        dfn[u]=stamp;
        if(!mat[u] || augment(mat[u],stamp)){
            mat[u]=x;
            return true;
        }
    }
    return false;
}

int match(){
    memset(mat+1,0,n2*sizeof(mat[0]));
    memset(dfn+1,0,n2*sizeof(dfn[0]));
    int cnt=0;
    for(int i=1;i<=n1;i++){
        if(augment(i,i))
            cnt++;
    }
    return cnt;
}

int main(){
    scanf("%d %d %d", &n1,&n2,&m);
    tot=0;
    memset(head+1,0,n1*sizeof(head[0]));
    for(int i=1;i<=m;i++){
        int f,g;
        scanf("%d %d", &f, &g);
        if(f>n1 || g>n2)
            continue;
```

```
52          add(f,g);
53      }
54      printf("%d\n", match());
55      return 0;
56  }
```

6.6 二分图最大匹配 Bipartite_Graph_Maximum_Matching_Hopcroft-Karp

```
1   #include <cstdio>
2   #include <cstring>
3   #include <queue>
4   #define MAXN 2005
5   #define MAXM 1000005
6
7   const int inf=0x3f3f3f3f;
8
9   using namespace std;
10
11  int n1,n2,n,m,tot,head[MAXN];
12  int mat[MAXN],lb[MAXN],dfn[MAXN];
13
14  struct edge{
15      int to,next;
16  }e[MAXM];
17
18  void add(int x,int y){
19      tot++;
20      e[tot].to=y;
21      e[tot].next=head[x];
22      head[x]=tot;
23  }
24
25  bool bfs(){
26      memset(lb+1,0,n*sizeof(lb[0]));
27      queue<int> q;
28      for(int i=1;i<=n1;i++){
29          if(!mat[i]){
30              q.push(i);
31              lb[i]=1;
32          }
33      }
34      int dis=inf;
35      while(!q.empty()){
36          int u=q.front();q.pop();
37          for(int p=head[u];p;p=e[p].next){
38              int v=e[p].to;
39              if(!lb[v]){
40                  lb[v]=lb[u]+1;
41                  if(!mat[v]) dis=lb[v];
42                  else if(lb[v]<dis){
43                      lb[mat[v]]=lb[v]+1;
```

```
                q.push(mat[v]);
            }
        }
    }
    return dis!=inf;
}

bool dfs(int u,int stamp){
    for(int p=head[u];p;p=e[p].next){
        int v=e[p].to;
        if(dfn[v]!=stamp && lb[v]==lb[u]+1){
            dfn[v]=stamp;
            if(!mat[v] || (lb[mat[v]]==lb[v]+1 && dfs(mat[v],stamp))){
                mat[v]=u;
                mat[u]=v;
                return true;
            }
        }
    }
    return false;
}

int hopcroft_karp(){
    int cnt=0,stamp=0;
    memset(dfn+1,0,n*sizeof(dfn[0]));
    memset(mat+1,0,n*sizeof(mat[0]));
    while(bfs()){
        stamp++;
        for(int i=1;i<=n1;i++){
            if(!mat[i] && dfs(i,stamp)){
                cnt++;
            }
        }
    }
    return cnt;
}

int main(){
    scanf("%d %d %d", &n1, &n2, &m);
    n=n1+n2;
    memset(head+1,0,n*sizeof(head[0]));
    tot=0;
    for(int i=1;i<=m;i++){
        int f,g;
        scanf("%d %d",&f,&g);
        if(f>n1 || g>n2) continue;
        add(f,n1+g);
    }
    printf("%d\n", hopcroft_karp());
    return 0;
}
```

7 计算几何

7.1 计算几何 Compuational_Geometry

```
#include <bits/stdc++.h>
#define MAXN 2000005
#define LL long long
using namespace std;
const double PI=acos(-1.0);
const double inf=1e100;
const double eps=1e-7;

int sgn(double d) {
    if(abs(d)<eps) return 0;
    if(d>0) return 1;
    return -1;
}

int dcmp(double x,double y) {
    if(abs(x-y)<eps) return 0;
    if(x>y) return 1;
    return -1;
}

struct Point{
    double x,y;
    Point(double x=0, double y=0):x(x),y(y){}

    Point operator + (const Point& B) const{
        return Point(x+B.x,y+B.y);
    }
    Point operator - (const Point& B) const{
        return Point(x-B.x,y-B.y);
    }
    Point operator * (const double k) const{
        return Point(x*k,y*k);
    }
    Point operator / (const double k) const{
        return Point(x/k,y/k);
    }
    bool operator < (const Point B) {
        if(dcmp(x,B.x)==0)
            return dcmp(y,B.y)<0;
        else return dcmp(x,B.x)<0;
    }

    double operator * (const Point& B) const{//点积
        return x*B.x+y*B.y;
    }
    double operator ^ (const Point& B) const {//叉积
        return x*B.y-y*B.x;
    }
};
```

```
50
51  typedef Point Vector;
52
53  double Length(Vector A) {
54      return sqrt(A*A);
55  }
56
57  double Angle(Vector A, Vector B) {//弧度
58      return acos(A*B/Length(A)/Length(B));
59  }
60
61  double Area2(Vector A, Vector B){//求平行四边形面积
62      return A^B;
63  }
64
65  Vector Rotate(Vector A, double rad) {//逆时针
66      return Vector(A.x*cos(rad)-A.y*sin(rad), A.x*sin(rad)+A.y*cos(rad));
67  }
68
69  Vector Normal(Vector A) {//逆时针转90度，单位法向量
70      double L=Length(A);
71      return Vector(-A.y/L, A.x/L);
72  }
73
74  bool ToLeftTest(Vector A, Vector B) {//B是不是在A左边
75      return sgn(A^B)>0;
76  }
77
78  struct Line {//点向式+两点。既可以line也可以seg
79      Point p1,p2;//p1->p2
80      Vector v;
81      Line(Point p1, Point p2):p1(p1),p2(p2),v((p2-p1)/Length(p2-p1)){}
82      Point point(double t) {//给t求点
83          return p1+v*t;
84      }
85  };
86  typedef Line Segment;
87
88  bool OnLine(Point P,Line l) {//判断点P是否在直线L上
89      return sgn((P-l.p1)^l.v);
90  }
91
92  Point GetIntersection(Line l1, Line l2) {//求直线交点
93      double t = (l2.v^(l1.p1-l2.p1))/(l1.v^l2.v);
94      return l1.point(t);
95  }
96
97  double DistanceToLine(Point P, Line l) {//点到直线距离
98      return abs(l.v^(P-l.p1));
99  }
100
101 Point GetProjection(Point P, Line l) {//求投影点
102     return l.point((P-l.p1)*l.v);
```

```
}

bool OnSegment(Point P, Segment s) {
    return (OnLine(P,s) && sgn((s.p1-P)*(s.p2-P))<0);
}

bool InSegmentIntersection(Segment s1, Segment s2) {//不允许端点相交
    double c1=(s1.p2-s1.p1)^(s2.p1-s1.p1), c2=(s1.p2-s1.p1)^(s2.p2-s1.p1);
    double c3=(s2.p2-s2.p1)^(s1.p1-s2.p1), c4=(s2.p2-s2.p1)^(s1.p2-s2.p1);
    return (sgn(c1)*sgn(c2)<0 && sgn(c3)*sgn(c4)<0);
}

void solve() {

}

int main() {
    int T=1,cas=1;
    // scanf("%d", &T);
    while(T--) {
        // printf("Case #%d: ", cas++);
        solve();
    }
    return 0;
}
```