ACM 模板

Buns_out

2023 年 8 月 14 日

目录

1 STL

1.1 快读快写

```
inline void read(int &x){
    int s = 0, w = 1; char ch = getchar();
    while(ch < '0' || ch > '9'){ if(ch == '-') w = -1; ch =
        getchar(); }
    while(ch >= '0' && ch <= '9') s = s * 10 + ch - '0', ch
        = getchar();
    x = s*w;
    return ;
}
inline void write(int x){
     if(x<0){
         putchar('-');
         x=-x;
     }
     if(x>9)write(x/10);
     putchar(x%10+'0');
}
```

```
unordered_map<int,int>mp;
unordered_set<int>s;
multiset<int>S;

int x=floor(0.1);//xia
int y=ceil(1.0);//shang
```

1.2 自定义开 O2 开栈

```
#pragma GCC optimize(2)
#pragma GCC optimize(3,"Ofast","inline")

#pragma comment(linker, "/STACK:1024000000,1024000000")

//自定义开栈
    int size(512<<20); // 512M
    __asm__ ( "movq␣%0,␣%%rsp\n"::"r"((char*)malloc(size)+
       size)); // YOUR CODE
```

1.3 set

```
unordered_set<int>s;
multiset<int>S;
fl=lower_bound(a,a+13,x)-a;//在从小到大的序列中找出大于等于x的数的地址
fl=upper_bound(a,a+13,x)-a;//在从小到大的序列中找出大于x的书的地址

fl=lower_bound(b,b+13,x,greater<int>() )-b;//在从大到小的序列中找出小于等于x的数的地址
fl=upper_bound(b,b+13,x,greater<int>() )-b;//在从小到大的序列中找出小于x的数的地址
s.erase(s.find(tmp));
```

1.4 string

```
string a=string(10,'9');
int x=stoll(a);
a=to_string(x);
```

1.5 python math 库

```
import math
math.e #常量e,2.718281828459045
math.pi #常量pi,3.141592653589793
math.factorial(x) #x的阶乘
math.gcd(x,y) #x,y的gcd
math.sqrt(x) #x的平方根
x=math.log(n,a) #以a为底n的对数x,a^x=n,默认底数为e
math.log(32,2) #5.0
math.degrees(math.pi/4) #将□/4转为角度
math.radians(45) #将45度转为弧度
math.cos(math.pi/4) #参数都为弧度
```

1.6 rand()、测速

```
srand(time(0));
clock_t start, finish;start = clock();
finish = clock();
solve();
cout << "the time cost is" <<double(finish - start)/
    CLOCKS_PER_SEC;
```

1.7 struct

```
struct node
{
    int sum,ls,rs,ts;
    node():sum(0),ls(0),rs(0),ts(0){}
    node(int a,int b,int c,int d):sum(a),ls(b),rs(c),ts(d)
        {}
};
```

1.8 vector

```
int sum=accumulate(v.begin(),v.end(),0); // 求和
//去重
vector<int> vector_unique(vector<int>v ){
    sort(v.begin(),v.end());
    auto v_it = unique(v.begin(),v.end());
    if(v_it != v.end())
        v.erase(v_it,v.end());
    return v;
}
//两个vector求并集
vector<int> vector_set_union(vector<int>v1 ,vector<int>v2)
    {
    vector<int>v;
    sort(v1.begin(),v1.end());
```

```
14      sort(v2.begin(),v2.end());
15      set_union(v1.begin(),v1.end(),v2.begin(),v2.end(),
            back_inserter(v));
16      return v;
17  }
18  //两个vector求交集
19  vector<int> vector_set_intersection(vector<int>v1 ,vector<
        int>v2){
20      vector<int>v;
21      sort(v1.begin(),v1.end());
22      sort(v2.begin(),v2.end());
23      set_intersection(v1.begin(),v1.end(),v2.begin(),v2.end
            (),back_inserter(v));
24      return v;
25  }
26  //判断vector的某一元素是否存在
27  bool is_element_in_vector(vector<int>v,int element){
28      // vector<int>::iterator it;
29      auto it=find(v.begin(),v.end(),element);
30      if (it!=v.end()){
31          return true;
32      }
33      else{
34          return false;
35      }
36  }
37  void Erase()
38  {
39      vector<int>v;
40      v.push_back(1); v.push_back(2);v.push_back(3);
41      //直接引用
42      v.back()-=1;
43      //删除最后一个元素
44      v.erase(v.end()-1,v.end());
45      //删除第一个元素
46      v.erase(v.begin(),v.begin()+1);
47      for(auto i:v)
```

```
48          cout<<i<<"␣";cout<<endl;
49      //sum
50      int sum=accumulate(v.begin(),v.end(),0);
51      // cout<<sum<<endl;
52      string str= accumulate(
53          next(v.begin()),
54          v.end(),
55          to_string(v[0]),
56          [](string a,int b){
57              return a+'-'+to_string(b);
58          }
59      );
60      cout<<str<<endl;
61      int MA = *max_element(v.begin(),v.end());//取值
62      int MA_fl = max_element(v.begin(),v.end())-v.begin();
          //取下标
63
64      int MI = *min_element(v.begin(),v.end());//取值
65      int MI_fk = min_element(v.begin(),v.end())-v.begin();
          //取下标
66  }
```

1.9 STL 函数

```
1  next_permutation(p+1,p+1+9);
2  prev_permutation(p+1,p+1+9);
3  //数字里面的 1 的个数
4  cnt=__builtin_popcount(x);
5  cnt=__builtin_popcountll(X);
6  //二进制表示形式中末尾0的个数
7  cnt=__builtin_ctz(x);
8  cnt=__builtin_ctzll(X);
9  //二进制表示形式中前导0的个数
10 cnt=__builtin_clz(x);
11 cnt=__builtin_clzll(X);
12 //二进制表示形式中1的个数的奇偶性（偶：0，奇：1）
```

```
13 cnt=__builtin_parity(x);
14 cnt=__builtin_parityll(X);
15 //二进制表示形式中最后一个1在第几位
16 cnt= __builtin_ffs(x);
17 cnt= __builtin_ffsll(X);
18 //快速开平方
19 cnt=__builtin_sqrt(x);
```

1.10 差分矩阵

```
1  void insert(int x1,int y1,int x2,int y2,int c)
2  {
3      s[x1][y1]+=c;
4      s[x2+1][y2+1]+=c;
5      s[x1][y2+1]-=c;
6      s[x2+1][y1]-=c;
7  }
8  void solve()
9  {
10     cin>>n>>m>>q;
11     for(int i=1;i<=n;i++)
12         for(int j=1;j<=m;j++)
13             cin>>p[i][j],insert(i,j,i,j,p[i][j]);
14     while(q--)
15     {
16         int a,b,c,d,k;
17         cin>>a>>b>>c>>d>>k;
18         insert(a,b,c,d,k);
19     }
20     for(int i=1;i<=n;i++)
21     {
22         for(int j=1;j<=m;j++){
23             p[i][j]=p[i-1][j]+p[i][j-1]-p[i-1][j-1]+s[i][j
                   ];
24             cout<<p[i][j]<<"␣";
25         }
26         cout<<endl;
27     }
28 }
```

1.11 前缀和矩阵

```
1  void solve(int a)
2  {
3      cin>>n>>m>>q;
4      for(int i=1;i<=n;i++)
5          for(int j=1;j<=m;j++){
6              cin>>p[i][j];
7              s[i][j]=s[i-1][j]+s[i][j-1]-s[i-1][j-1]+p[i][j
                   ];
8          }
9      while(q--)
10     {
11         int x1,x2,y1,y2;
12         cin>>x1>>y1>>x2>>y2;
13         cout<<s[x2][y2]-s[x2][y1-1]-s[x1-1][y2]+s[x1-1][y1
               -1]<<endl;
14     }
15 }
```

1.12 分治，最大字段和

```
1  int p[maxn];
2  int n,m;
3  //分治
4  int MaxIn(int *a)
5  {
6      vector<int>dp(n+1,0);
7      int ma=0;
8      for(int i=1;i<=n;i++)
9      {
10         if(dp[i-1]>0)
11         {
12             dp[i]=dp[i-1]+a[i];
13         }
14         else dp[i]=a[i];
```

```
15          ma=max(ma,dp[i]);
16      }
17      return ma;
18  }
19  int MaxIn(int *a,int l,int r)
20  {
21      if(l==r)
22          return a[l]>0?a[l]:0;
23      int mid=(l+r)>>1;
24      //左边区间的最大子段和
25      int l_max = MaxIn(a,l,mid);
26      //右边区间的最大子段和
27      int r_max = MaxIn(a,mid+1,r);
28
29      int sum = 0;
30      int left_max = 0;
31      for(int i=mid;i>=l;i--)
32      {
33          sum+=a[i];
34          if(sum>left_max)
35              left_max = sum;
36      }
37
38      sum = 0;
39      int right_max = 0;
40      for(int i=mid+1;i<=r;i++)
41      {
42          sum+=a[i];
43          if(sum>right_max)
44              right_max = sum;
45      }
46      int res = left_max + right_max;
47      if(res < l_max)
48          res = l_max;
49      if(res < r_max)
50          res = r_max;
51      return res;
```

```
52 }
53 void solve()
54 {
55     cin>>n;
56     for(int i=1;i<=n;i++)
57         cin>>p[i];
58     cout<<MaxIn(p)<<endl;
59 }
```

2 字符串

2.1 kmp

```
struct KMP{
    int nxt[maxn];
    int len;
    void init(char *s)
    {
        nxt[1]=0;
        len=strlen(s+1);
        for(int i=2;i<=len;i++)
        {
            nxt[i]=nxt[i-1];
            while(s[nxt[i]+1]!=s[i]&&nxt[i])
                nxt[i]=nxt[nxt[i]];
            if(s[nxt[i]+1]==s[i])nxt[i]++;
        }
    }
    void kmp(char *s,char *t)
    {
        int lens=strlen(s+1);
        int lent=strlen(t+1);
        int cnt=0;
        for(int i=0,j=0;i<lens;i++)
        {
            while(j&&s[i+1]!=t[j+1])j=nxt[j];
            if(s[i+1]==t[j+1])j++;
            if(j==lent)
            {
                cnt++;
                j=nxt[j];
            }
        }
    }
    /* 循环周期 形如 acaca 中 ac 是一个合法周期 */
    vector<int> periodic()
```

```
34      {
35          vector<int>ret;
36          int now=len;
37          while(now)
38          {
39              now=nxt[now];
40              ret.push_back(len-now);
41          }
42          return ret;
43      }
44      /* 循环节 形如 acac 中ac、acac是循环节，aca不是*/
45      vector<int> periodic_loop()
46      {
47          vector<int>ret;
48          for(auto i:periodic())
49          {
50              if(len%i==0)
51                  ret.push_back(i);
52          }
53          return ret;
54      }
55      void debug(){
56          for (int i=0;i<=len;i++){
57              printf("[debug] nxt[%d]=%d\n",i,nxt[i]);
58          }
59      }
60  }kmp;
```

2.2 exKMP

对于个长度为 n 的字符串。定义函数表示 $z[i]$ 表示 s 和 $s[i,n-1]$（即以 $s[i]$ 开头的后缀）的最长公共前缀（LCP）的长度。被称为的 Z 函数。

```
1  struct EXKMP{
2  //result: ext[i] = LCP(S[i,lens],T)
3  //require: nxt[i] = LCP(T[i,lent],T)
4  //nxt : s  Mode_String
5  //ext : s  Text_String t Mode_String
6  void exkmp(char *s,int lens,char *t,int lent,int *ext,int
       *nxt)
7  {
8      ext[0]=0;
9      for(int i=1,p0=0,p=0;i<=lens;i++)
10     {
11         ext[i]=i<=p?min(nxt[i-p0+1],p-i+1):0;
12         while(i+ext[i]<=lens&&ext[i]<lent&&s[i+ext[i]]==t[
               ext[i]+1])ext[i]++;
13         if(i+ext[i]-1>=p&&i!=1)p0=i,p=i+ext[i]-1;
                                  //最右端
14     }
15 }
16 }exKMP;
17 char s[maxn];
18 char t[maxn];
19 int nxt[maxn];
20 int ext[maxn];
21 void solve()
22 {
23     scanf("%s",t+1);
24     scanf("%s",s+1);
25     exKMP.exkmp(s,strlen(s+1),s,strlen(s+1),nxt,nxt);//
26     exKMP.exkmp(t,strlen(t+1),s,strlen(s+1),ext,nxt);
27 }
```

2.3 Manacher

```
struct Manacher{
    char ch[maxn];
    int lc[maxn];
    int N;
    void init(char *s){
        int n=strlen(s+1);// puts(s+1);
        ch[n*2+1]='#';
        ch[0]='@';
        ch[n*2+2]='\0';
        for(int i=1;i<=n;i++)ch[i*2]=s[i],ch[i*2-1]='#';
        N=n*2+1;
    }
    void manacher(){
        lc[1]=1;  int k=1,ma=1;
        for (int i=2;i<=N;i++){
            int p = k+lc[k]-1;
            if (i<=p){lc[i]=min(lc[2*k-i],p-i+1);
            }else{  lc[i]=1;}
            while(ch[i+lc[i]]==ch[i-lc[i]])lc[i]++;
            if(i+lc[i]>k+lc[k])k=i;
            ma=max(ma,lc[i]-1);
        }
    }
}Manch;
char s[maxn];
void solve()
{
    scanf("%s",s+1);
    Manch.init(s);
    Manch.manacher();
}
```

2.4 字符串哈希

```
#define ull unsigned long long
using ll=long long;
using pii=pair<int,int>;
const int inf=0x3f3f3f3f;
const int INF=1e9+7;
const int maxn=1e6;
const ull mod=2123704401301379571l;
const ull prime=233317;
const ull base=131;
const ull more=19260817;
const ull seed[]={911,146527,19260817,91815541};
ull bas[maxn];
map<ull,bool>mp;
ull a[maxn];
int n;
namespace EEE{
    class StringHash{
    public:
        const ull mod=2123704401301379571l;
        const ull prime=233317;
        const ull base=131;
        vector<ull>h;
        vector<ull>bas;
        StringHash():h(1),bas(1,1){}
        void push_back(char ch){
            h.push_back((h.back()*base+ch)%mod + prime);
            bas.push_back(bas.back()*base%mod);
        }
        ull get(int l,int r){
            return (h[r] + __int128(h[l])*(mod-bas[r-l]))%
               mod;
        }
    };
    void A()
    {
```

```
35          string str;
36          StringHash hs,rhs;
37          int N=int(str.size());
38          for(int i=0;i<N;i++)
39              hs.push_back(str[i]);
40          for(int i=N-1;i>=0;i--)
41              rhs.push_back(str[i]);
42      }
43  };
44  ull get_hash(string &s)
45  {
46      ull ans=0;
47      for(int i=0;i<s.size();i++)
48          ans=(ans*base+(ull)s[i])%mod+prime;
49      return ans;
50
51
52      bas[0]=1;
53      for(int i=1;i<=n;i++){
54          bas[i]=bas[i-1]*seed[0]%mod;
55      }
56  }
57  void solve()
58  {
59      cin>>n;
60      for(int i=1;i<=n;i++)
61      {
62          string str;
63          cin>>str;
64          a[i]=get_hash(str);
65      }
66      sort(a+1,a+1+n);
67      int ans=1;
68      for(int i=2;i<=n;i++)
69          if(a[i]!=a[i-1])
70              ans++;
71      cout<<ans<<endl;
```

```
72 }
73 signed main()
74 {
75     ios::sync_with_stdio(false);
76     cin.tie(nullptr);cout.tie(nullptr);
77         solve();
78     return 0;
79 }
```

2.5 最小表示法

```
int Get_Min(int *p,int n)
{
    int i=0,j=1,k=0;
    while(i<n&&j<n&&k<n)
    {
        if(p[(i+k)%n]==p[(j+k)%n])k++;
        else
        {
            if(p[(i+k)%n]>p[(j+k)%n])i=i+k+1;
            else j=j+k+1;
            if(i==j)i++;
            k=0;
        }
    }
    return min(i,j);
}
string str;
int p[maxn];
int n;
void solve()
{
    cin>>n;
    for(int i=0;i<n;i++)
        cin>>p[i];
    int tmp=Get_Min(p,n);
    // cout<<tmp<<endl;
    for(int i=0;i<n;i++)
        cout<<p[(i+tmp)%n]<<"␣";
}
```

2.6 AC 自动机

```
string mp[maxn];
vector<string>v;
int n;
struct Trie{
    int nxt[maxn][26],fail[maxn];
    int val[maxn];
    int cnt[maxn];
    int id[maxn];
    int q[maxn];
    int root=0,indx=0;
    int top=0,low=1;
    void clear(){
        for(int i=0;i<=indx;i++){
            val[i]=cnt[i]=id[i]=q[i]=fail[i]=0;
            for(int j=0;j<26;j++)
                nxt[i][j]=0;
        }
        indx=top=0;low=1;//v.clear();
    }
    void insert(string& str,int x)
    {
        int rt=0;
        for(int i=0;i<str.size();i++){
            int tmp=str[i]-'a';
            if(!nxt[rt][tmp])nxt[rt][tmp]=++indx;
            rt=nxt[rt][tmp];
        }
        val[rt]++;id[x]=rt;
    }
    void build()
    {
        for(int i=0;i<26;i++)
            if(nxt[0][i])
                q[++top]=nxt[0][i];
        while(low<=top){
```

```
36              int x=q[low++];
37              for(int i=0;i<26;i++){
38                  int &rt=nxt[x][i];
39                  if(!rt)rt=nxt[fail[x]][i];
40                  else{
41                      fail[rt]=nxt[fail[x]][i];
42                      q[++top]=rt;
43                  }
44              }
45          }
46      }
47      void query(string& s)
48      {
49          for(int i=0,j=0;i<s.size();i++){
50              j=nxt[j][s[i]-'a'];
51              cnt[j]++;
52          }
53          for(int i=indx;i;i--)
54              cnt[fail[q[i]]]+=cnt[q[i]];
55          //模式串i出现次数
56          // for(int i=1;i<=n;i++)
57          //  cout<<cnt[id[i]]<<endl;
58          //模式串出现次数
59          // int ans=0;
60          // int now=0;
61          // for(int i=0;i<s.size();i++)
62          // {
63          //  now=nxt[now][s[i]-'a'];
64          //  for(int t=now;t&&val[t];t=fail[t])
65          //      ans+=val[t],val[t]=0;
66          // }
67          // cout<<ans<<endl;
68      }
69  }trie;
70  void solve()
71  {
72      while(cin>>n&&n)
```

```
73      {
74          trie.clear();
75          for(int i=1;i<=n;i++)
76          {
77              string str;
78              cin>>str;
79              mp[i]=str;
80              trie.insert(str,i);
81          }
82          trie.build();
83          string t;
84          cin>>t;
85          trie.query(t);
86          break;
87      }
88  }
```

3 图论

3.1 差分约束

```
1  a-b<=c add(b,a,c)
2  求最小生成树
3
4  a-b>=c add(b,a,c)
5  求最大生成树
6  或add(a,b,-c)
7
8  a-b==c
9  add(b,a,c)
10 add(a,b,-c)
```

3.2 Floyd

```
1 for(int k=1;k<=n;k++)
2         for(int i=1;i<=n;i++)
3             for(int j=1;j<=n;j++)
4                 dis[i][j]=min(dis[i][j],dis[i][k]+dis[k][j
                      ]);
5 }
```

3.3 2-SET

```
void solve()
{
    cin>>n>>m;
    /*
    如果指定 x 一定为真
    add(x,x+n);
    如果x为假那么x一定为真
    */
    for(int i=1;i<=m;i++)
    {
        int x,a,y,b;
        cin>>x>>a>>y>>b;
        if(a==1&&b==1)//x为真或者y为真
        {
            v[x].push_back(y+n);//x为假那么y一定为真
            v[y].push_back(x+n);//y为假那么x一定为真
        }
        else if(a==0&&b==1)//x为假或者y为真
        {
            v[x+n].push_back(y+n);//x为真那么y一定为真
            v[y].push_back(x);//y为假那么x一定为假
        }
        else if(a==1&&b==0)//x为真或者y为假
        {
            v[x].push_back(y);//x为假那么y一定为假
            v[y+n].push_back(x+n);//y为真那么x一定为真
        }
        else if(a==0&&b==0)//x为假或者y为假
        {
            v[x+n].push_back(y);
            v[y+n].push_back(x);
        }
    }
    for(int i=1;i<=n*2;i++)
        if(!dfn[i])
```

```
36              tarjan(i);
37      //判断无解情况
38      for(int i=1;i<=n;i++)
39          if(col[i]==col[i+n]){
40              cout<<"IMPOSSIBLE"<<endl;
41              return ;
42          }
43      //输出拓扑序靠后(col较小)
44      cout<<"POSSIBLE"<<endl;
45      for(int i=1;i<=n;i++)
46          cout<<(col[i]>col[i+n])<<"␣";
47      cout<<endl;
48  }
```

3.4 倍增优化建图

```
int find(int x,int y){
    if(id[x][y])return id[x][y];
    id[x][y]=++Flow::tot;
    if(!y){
        Flow::Add(id[x][y],Flow::t,val[x]);
    }
    else {
        Flow::Add(id[x][y],find(x,y-1),INF);
        Flow::Add(id[x][y],find(f[x][y-1],y-1),INF);
    }
    return id[x][y];
}

// 与倍增求lca相同的循环
int lca=LCA(x,y);
for(int i=14;i>=0;i--)
    if((dep[x]-dep[lca])>>i&1)
    {
        Flow::Add(now,find(x,i),INF);
        x=f[x][i];
    }
for(int i=14;i>=0;i--)
    if((dep[y]-dep[lca])>>i&1)
    {
        Flow::Add(now,find(y,i),INF);
        y=f[y][i];
    }
```

3.5 prim

```
#include<bits/stdc++.h>
using namespace std;
#define endl ”\n”
using ull=unsigned long long;
using ll=long long;
using pii=pair<int,int>;
const int dx[]={0,0,1,-1,1,-1,1,-1};
const int dy[]={1,-1,0,0,1,-1,-1,1};
const ull str_mod=2123704401301379571l;
const ull more=19260817;
const ull prime=233317;
const ull base=131;
const int inf=0x3f3f3f3f;
const int INF=1e9+7;
const int maxn=1e6;
struct node
{
    int t;int d;
    bool operator < (const node &a) const
    {
        return d>a.d;
    }
};
struct Edge{
    int to,next,w;
}edge[maxn];
int head[maxn],cnt;
bitset<maxn>vis;
int n,m;
void add(int from,int to,int w){
    edge[++cnt].w=w;
    edge[cnt].to=to;
    edge[cnt].next=head[from];
    head[from]=cnt;
}
```

```
36 void prim()
37 {
38     priority_queue<node>q;
39     q.push({1,0});
40     int ans=0;
41     while(!q.empty())
42     {
43         auto [x,X]=q.top();q.pop();
44         if(vis[x])continue;
45         vis[x]=1;
46         ans+=X;
47         for(int i=head[x];i;i=edge[i].next)
48             if(!vis[edge[i].to])
49                 q.push({edge[i].to,edge[i].w});
50     }
51     int flag=1;
52     for(int i=1;i<=n;i++)
53         if(!vis[i])flag=0;
54     if(!flag)cout<<”orz”<<endl;
55     else cout<<ans<<endl;
56 }
57 void solve()
58 {
59     cin>>n>>m;
60     for(int i=1;i<=m;i++)
61     {
62         int x,y,w;
63         cin>>x>>y>>w;
64         add(x,y,w);
65         add(y,x,w);
66     }
67     prim();
68 }
69 signed main(){
70     ios::sync_with_stdio(false);
71     cin.tie(nullptr);cout.tie(nullptr);
72         solve();
```

```
73      return 0;
74  }
```

3.6 匈牙利

```cpp
#include<bits/stdc++.h>
using namespace std;
using ll=long long;
using pii=pair<int,int>;
const int maxn=1e6;
vector<int>v[maxn];
int mt[maxn];
int n,m,e;
bitset<maxn>vis;
bool dfs(int x)
{
    if(vis[x])return 0;
    vis[x]=1;
    for(auto y:v[x]){
        if(!mt[y]||dfs(mt[y])){
            mt[y]=x;
            return 1;
        }
    }
    return 0;
}
int main()
{
    cin>>n>>m>>e;
    for(int i=1;i<=e;i++){
        int x,y;
        cin>>x>>y;
        v[x].push_back(y);
        // v[y].push_back(x);
    }
    int ans=0;
    for(int i=1;i<=n;i++){
        vis.reset();
        if(dfs(i))ans++;
    }
```

```
36      cout<<ans<<endl;
37  }
```

3.7 KM

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
using ll=long long;
using pii=pair<int,int>;
const int INF=1e9+7;
const int mod=998244353;
const int N=303;
int w[N][N];
int n,m;
bitset<N>vx,vy;
namespace KM {
    long long cal(int n, int m) {
        std::vector<long long> u(n + 1), v(m + 1), p(m + 1), way(m + 1);
        for (int i = 1; i <= n; i++) {
            p[0] = i;
            long long j0 = 0;
            std::vector<long long> minv(m + 1, 1e18);
            std::vector<char> used(m + 1, false);
            do {
                used[j0] = true;
                long long i0 = p[j0], delta = 1e18, j1;
                for (int j = 1; j <= m; ++j) {
                    if (!used[j]) {
                        long long cur = w[i0][j] - u[i0] - v[j];
                        if (cur < minv[j]) {
                            minv[j] = cur, way[j] = j0;
                        }
                        if (minv[j] < delta) {
                            delta = minv[j], j1 = j;
                        }
                    }
                }
```

```
34                  for (int j = 0; j <= m; ++j) {
35                      if (used[j]) {
36                          u[p[j]] += delta, v[j] -= delta;
37                      }
38                      else {
39                          minv[j] -= delta;
40                      }
41                  }
42                  j0 = j1;
43              } while (p[j0] != 0);
44              do {
45                  long long j1 = way[j0];
46                  p[j0] = p[j1];
47                  j0 = j1;
48              } while (j0);
49          }
50          long long res = 0;
51          for (int i = 1; i <= m; i++) {
52              res += w[p[i]][i];
53          }
54          return res;
55      }
56  }
57  void solve()
58  {
59      while(cin>>n)
60      {
61          for(int i=1;i<=n;i++)
62          {
63              for(int j=1;j<=n;j++)
64              {
65                  cin>>w[i][j];
66              }
67          }
68          /*最小费用*/
69          cout<<KM::cal(n, n)<<endl;
70      }
```

```
71 }
72 signed main()
73 {
74     ios::sync_with_stdio(false);
75     cin.tie(0);cout.tie(0);
76     int __;cin>>__;
77     while(__--)
78         solve();
79 }
```

3.8 点双

```
struct Edge{
    int to,next,w;
}edge[maxn];
int head[maxn],cnt;
int dfn[maxn],low[maxn],indx;
vector<int>bcc[maxn];
int col[maxn],tot;
bitset<maxn>vis;
stack<int>s;
int bri[maxn];
int cut[maxn];
int n,m,bcc_cnt;
void add(int from,int to)
{
    edge[++cnt].to=to;
    edge[cnt].next=head[from];
    head[from]=cnt;
}
void tarjan(int x,int fa)
{
    dfn[x]=low[x]=++indx;
    vis[x]=1;
    s.push(x);
    int num=0;
    for(int i=head[x];i;i=edge[i].next)
    {
        int y=edge[i].to;
        if(!dfn[y])
        {
            num++;
            tarjan(y,i);
            low[x]=min(low[x],low[y]);
            if(dfn[x]<low[y])
                bri[i]=bri[i^1]=1;
            if(low[y]>=dfn[x])
```

```
36              {
37                  cut[x]=1;
38                  bcc_cnt++;
39                  bcc[bcc_cnt].push_back(x);
40                  int v;
41                  do{
42                      v=s.top();s.pop();
43                      bcc[bcc_cnt].push_back(v);
44                  }while(v!=y);
45              }
46          }
47          else if(vis[y]&&dfn[y]<dfn[x]&&i!=(fa^1))
48              low[x]=min(low[x],dfn[y]);
49      }
50      if(fa==0&&num==1)cut[x]=0;
51  }
```

3.9 边双

```
1  void tarjan(int x,int fa)
2  {
3      dfn[x]=low[x]=++indx;
4      vis[x]=1;
5      s.push(x);
6      for(int i=head[x];i;i=edge[i].next)
7      {
8          int y=edge[i].to;
9          if(!dfn[y])
10         {
11             tarjan(y,i);
12             low[x]=min(low[x],low[y]);
13             if(dfn[x]<low[y])
14                 bri[i]=bri[i^1]=1;
15         }
16         else if(vis[y]&&dfn[y]<dfn[x]&&i!=(fa^1))
17             low[x]=min(low[x],dfn[y]);
18     }
19     if(dfn[x]==low[x])
20     {
21         tot++;
22         int y;
23         do{
24             y=s.top();s.pop();
25             vis[y]=0;
26             col[y]=tot;
27         }while(x!=y);
28     }
29 }
```

3.10 仙人掌图求环路径长度

```
const int mod=998244353;
const int inf=0x3f3f3f3f;
const int INF=1e14+7;
const int maxn=4e6+100;
struct Edge{
    int to,w,next;
}edge[maxn];
int head[maxn],cnt;
int dfn[maxn],low[maxn],indx;
stack<pii>s;
int n,m,ans;
void init()
{
    while(!s.empty())s.pop();
    for(int i=1;i<=n;i++)
    {
        dfn[i]=low[i]=head[i]=0;
    }
    cnt=1;
    indx=0;
    ans=INF;
}
void add(int from,int to,int w)
{
    edge[++cnt].w=w;
    edge[cnt].to=to;
    edge[cnt].next=head[from];
    head[from]=cnt;
}
void tarjan(int x,int fa)
{
    dfn[x]=low[x]=++indx;
    for(int i=head[x];i;i=edge[i].next)
    {
        int y=edge[i].to;
```

```
36          if(!dfn[y])
37          {
38              s.push({x,edge[i].w});
39              tarjan(y,i);
40              low[x]=min(low[x],low[y]);
41              if(low[y]>=dfn[x])
42              {
43                  vector<int>g;
44                  pii v;
45                  do{
46                      v=s.top();s.pop();
47                      g.push_back(v.second);
48                  }while(v.first!=x);
49
50                  sort(g.begin(),g.end());
51
52                  if(g.size()==1)ans=min(ans,g[0]);
53                  else if(g.size()==2)ans=min(ans,g[0]+g[1])
                        ;
54                  else ans=min(ans,min(g[0]+g[1],g[2]));
55              }
56          }
57          else if(dfn[x]>dfn[y]&&i!=(fa^1))
58          {
59              low[x]=min(low[x],dfn[y]);
60              s.push({x,edge[i].w});
61          }
62      }
63  }
64  void solve()
65  {
66      cin>>n>>m;
67      init();
68      for(int i=1;i<=m;i++)
69      {
70          int x,y,w;
71          cin>>x>>y>>w;
```

```
72          add(x,y,w);
73          add(y,x,w);
74      }
75      tarjan(1,0);
76      cout<<ans<<endl;
77  }
78  signed main(){
79      int size(512<<20); // 512M
80      __asm__ ( "movq %0, %%rsp\n"::"r"((char*)malloc(size)+
            size)); // YOUR CODE
81      ios::sync_with_stdio(false);
82      cin.tie(nullptr);cout.tie(nullptr);
83      int __;cin>>__;
84      while(__--)
85          solve();
86      exit(0);
87  }
```

3.11 DINIC

```
struct Edge{
    int to,next,w;
}edge[maxn];
int head[maxn],cnt=1;
int dep[maxn];
int n,m,s,t,tot;
void init(){
    memset(head+1,0,sizeof(int)*tot);
    cnt=1;
    tot=0;
}
void add(int from,int to,int w)
{
    edge[++cnt].w=w;
    edge[cnt].to=to;
    edge[cnt].next=head[from];
    head[from]=cnt;
}
bool bfs(int s,int t)
{
    memset(dep+1,0,sizeof(int)*tot);
    queue<int>q;
    dep[s]=1;
    q.push(s);
    while(!q.empty())
    {
        int x=q.front();q.pop();
        if(x==t)return true;
        for(int i=head[x];i;i=edge[i].next)
        {
            int y=edge[i].to;
            int w=edge[i].w;
            if(dep[y]==0&&w>0)
                q.push(y),dep[y]=dep[x]+1;
        }
```

```
36      }
37      return false;
38  }
39  int dfs(int x,int flow,int t)
40  {
41      if(x==t)return flow;
42      int out=flow;
43      for(int i=head[x];i;i=edge[i].next)
44      {
45          int y=edge[i].to;
46          int w=edge[i].w;
47          if(w!=0&&dep[y]==dep[x]+1)
48          {
49              int tmp=dfs(y,min(out,w),t);
50              edge[i].w-=tmp;
51              edge[i^1].w+=tmp;
52              out-=tmp;
53              if(!out)break;
54          }
55      }
56      if(out==flow)dep[x]=0;
57      return flow-out;
58  }
59  int DINIC(int s,int t)
60  {
61      int ans=0;
62      while(bfs(s,t))
63          ans+=dfs(s,INF,t);
64      return ans;
65  }
```

3.12 MCMF

```
struct node{
    int to,next,w,c;
}edge[maxn];
int head[maxn],cnt;
int dis[maxn],flow[maxn],last[maxn];
bitset<maxn>vis;
int n,m,s,t;
void init(){
    memset(head+1,0,sizeof(int)*n);
    cnt=1;
}
void add(int from,int to,int w,int c)
{
    edge[++cnt].w=w;
    edge[cnt].c=c;
    edge[cnt].to=to;
    edge[cnt].next=head[from];
    head[from]=cnt;
}
bool spfa(int s,int t)
{
    memset(dis+1,INF,sizeof(int)*n);
    memset(last+1,-1,sizeof(int)*n);
    vis.reset();
    queue<int>q;
    q.push(s);
    dis[s]=0;
    vis[s]=1;
    flow[s]=INF;
    while(!q.empty())
    {
        int x=q.front();q.pop();
        vis[x]=0;
        for(int i=head[x];i;i=edge[i].next)
        {
```

```
36              int y=edge[i].to;
37              int w=edge[i].w;
38              int c=edge[i].c;
39              if(w>0&&dis[y]>dis[x]+c)
40              {
41                  dis[y]=dis[x]+c;
42                  last[y]=i;
43                  flow[y]=min(flow[x],w);
44                  if(!vis[y])
45                      q.push(y),vis[y]=1;
46              }
47          }
48      }
49      return last[t]!=-1;
50  }
51  void MCMF(int s,int t)
52  {
53      int maxf=0,maxc=0;
54      while(spfa(s,t))
55      {
56          maxf+=flow[t];
57          maxc+=flow[t]*dis[t];
58          for(int i=t;i!=s;i=edge[last[i]^1].to)
59          {
60              edge[last[i]].w-=flow[t];
61              edge[last[i]^1].w+=flow[t];
62          }
63      }
64      cout<<maxf<<" "<<maxc<<endl;
65  }
```

3.13 朱流算法（Edmonds）

```
#include<bits/stdc++.h>
using namespace std;
#define endl ”\n”
const int INF=2e9+7;
const int maxn=1e6+100;
struct node{
    int x,y,w;
}e[maxn];
int vis[maxn];//标记是否走过
int pre[maxn];//前驱节点
int inv[maxn];//节点权值
int id[maxn];//新的标号
int n,m,root;
int Edmonds()
{
    int ans=0;
    while(1)
    {
        /**********1.找最短弧集合**********/
        //初始化
        for(int i=1;i<=n;i++)
            inv[i]=INF;
        for(int i=1;i<=m;i++)
        {
            int x=e[i].x;
            int y=e[i].y;
            //找到每个点的最小入边 以及前驱
            if(x!=y&&e[i].w<inv[y])
                inv[y]=e[i].w,pre[y]=x;
        }
        //如果一个点不是根 并且没有入边则不是树形图
        for(int i=1;i<=n;i++)
            if(i!=root&&inv[i]==INF)
                return -1;
        int cnt=0;
```

```
36          //初始化标记
37          for(int i=1;i<=n;i++)
38              vis[i]=id[i]=0;
39          /*******2.判断集合E中有没有有向环，如果有转步骤3，
                否则转4*********/
40          for(int i=1;i<=n;i++)
41          {
42              if(i==root)continue;
43              ans+=inv[i];
44              int v=i;
45              while(vis[v]!=i&&!id[v]&&v!=root)
46              {
47                  vis[v]=i;
48                  v=pre[v];
49              }
50              /**********3.收缩G中的有向环*********/
51              if(!id[v]&&v!=root)
52              {
53                  id[v]=++cnt;
54                  for (int u=pre[v];u!=v;u=pre[u])
55                      id[u]=cnt;
56              }
57          }
58          if(cnt==0)break;
59          for(int i=1;i<=n;i++)
60              if(!id[i])id[i]=++cnt;
61          for(int i=1;i<=m;i++)
62          {
63              int u=e[i].x;
64              int v=e[i].y;
65              e[i].x=id[u];
66              e[i].y=id[v];
67              if(id[u]!=id[v])
68                  e[i].w-=inv[v];
69          }
70          root=id[root];
71          n=cnt;
```

```
72      }
73      return ans;
74  }
75  void solve()
76  {
77      cin>>n>>m>>root;
78      for(int i=1;i<=m;i++)
79          cin>>e[i].x>>e[i].y>>e[i].w;
80      cout<<Edmonds()<<endl;
81  }
82  signed main(){
83          solve();
84      return 0;
85  }
```

3.14 最小生成树 Boruvka

```
#define mp make_pair
const int N=200005;
const int M=500005;
const LL inf=1e12;
int f[N],pd,n,m;
struct node{ int a,b; LL c; }p[M];
pair<LL,LL> E[N];
int find(int r){
    return f[r]=(r==f[r])?r:find(f[r]);
}
LL Boruvka(){
    LL res=0; pd=1; int num=0;
    for(int i=1;i<=n;i++)
        f[i]=i;
    while(num<n-1)
    {
        int tmp=0;
        for(int i=1;i<=n;i++)
        {
            int fa=find(i);
            E[fa]={inf,inf};
        }
        for(int i=1;i<=m;i++)
        {
            int fa=find(p[i].a);
            int fb=find(p[i].b);
            if (fa==fb) continue;
            tmp++;
            E[fa]=min(E[fa],{p[i].c,i*1ll});
            E[fb]=min(E[fb],{p[i].c,i*1ll});
        }
        if (tmp==0) break;
        for(int i=1;i<=m;i++)
        {
            int fa=find(p[i].a); int fb=find(p[i].b);
```

```
36              if (fa==fb) continue;
37              if ((E[fa]==mp(p[i].c,i*1ll))||(E[fb]==mp(p[i
                   ].c,i*1ll)))
38              { f[fa]=fb; res+=p[i].c; num++; }
39          }
40      }
41      if (num<n-1) pd=0;
42      return res;
43  }
44  int main(){
45      scanf("%d%d",&n,&m);
46      for(int i=1;i<=m;i++)
47          scanf("%d%d%lld",&p[i].a,&p[i].b,&p[i].c);
48      LL ans=Boruvka();
49      if (!pd) printf("orz\n");
50      else printf("%lld\n",ans);
51      return 0;
52  }
```

3.15 基尔霍夫矩阵

```
/*
基尔霍夫局矩阵树
用于解决:
    给定n个点m条边的无向图，求图的生成树个数
基本定义:
1.无向图G:给定n个点，m条边的无向图，设点击为V，边集为E，记为G(V,E)
2.度数矩阵D[G]:当i!=j时，D[i][j]=0,当i==j时,D[i][j]=点v的度数
3.邻接矩阵A[G]:当vi，vj有边链接时，A[i][j]=1,当vi，vj无边连接时，A[i][j]=0
4.基尔霍夫矩阵:K[G]:K[G]=D[G]-A[G],K[i][j]=D[i][j]-A[i][j]
定理:
    对于已经得出的基尔霍夫矩阵，去掉随意一行一列得出的矩阵的行列式
其绝对值为生成树个数

一般来说求(1,1)~(n-1,n-1)这个行列式
*/
ll det(int n)
{
    ll res=1;
    for(int i=2;i<=n;i++){
        for(int j=i+1;j<=n;j++){
            while(a[j][i]){
                ll t=a[i][i]/a[j][i];
                for(int k=i;k<=n;k++)
                    a[i][k]=(a[i][k]-a[j][k]*t);
                for(int k=i;k<=n;k++)
                    swap(a[i][k],a[j][k]);
                res=-res;
            }
        }
        if(a[i][i]==0)res=-res;
        res*=a[i][i];
```

```
32      }
33      if(res<0)res=-res;
34      return res;
35  }
36  void solve()
37  {
38      init();
39      cin>>n>>m;
40      while(m--)
41      {
42          int x,y;
43          cin>>x>>y;
44          a[x][x]++;
45          a[y][y]++;
46          a[x][y]--;
47          a[y][x]--;
48      }
49      cout<<det(n)<<endl;
50  }
```

3.16 全源最短路

```
struct node{
    int x;ll dis;
    bool operator<(const node&a)const{
        return a.dis<dis;
    }
};
struct Edge{int to,next;ll w;}edge[maxn];
int head[maxn],cnt;
ll dis[maxn];
ll h[maxn];
int n,m;
void add(int from,int to,ll w){
    edge[++cnt].w=w;
    edge[cnt].to=to;
    edge[cnt].next=head[from];
    head[from]=cnt;
}
bool spfa(int s)
{
    for(int i=0;i<=n;i++)h[i]=INF;
    vector<int>in(n+10,0);
    bitset<maxn>vis;
    queue<int>q;
    vis[s]=1;
    h[s]=0;
    q.push(s);
    while(!q.empty()){
        int x=q.front();q.pop();
        vis[x]=0;
        in[x]++;
        if(in[x]>n)return 0;
        for(int i=head[x];i;i=edge[i].next){
            int y=edge[i].to;
            if(h[y]>h[x]+edge[i].w){
                h[y]=h[x]+edge[i].w;
```

```
36                  if(!vis[y])q.push(y),vis[y]=1;
37              }
38          }
39      }
40      return 1;
41  }
42  void dij(int s)
43  {
44      for(int i=1;i<=n;i++)dis[i]=INF;
45      priority_queue<node>q;
46      bitset<maxn>vis;
47      q.push({s,0});
48      dis[s]=0;
49      while(!q.empty()){
50          auto [x,now]=q.top();q.pop();
51          if(vis[x])continue;
52          vis[x]=1;
53          for(int i=head[x];i;i=edge[i].next){
54              int y=edge[i].to;
55              if(dis[y]>dis[x]+edge[i].w){
56                  dis[y]=dis[x]+edge[i].w;
57                  q.push({y,dis[y]});
58              }
59          }
60      }
61  }
62  int main()
63  {
64      cin>>n>>m;
65      for(int i=1;i<=m;i++){
66          int x,y,z;
67          cin>>x>>y>>z;
68          add(x,y,z);
69      }
70      for(int i=1;i<=n;i++)
71          add(0,i,0);
72      //建立虚点，从0开始单源最短路
```

```
73      //0开始单源最短路即为势能
74      //同时spfa判断是否存在负环（in[x]>n)
75      if(!spfa(0)){
76          cout<<-1<<endl;
77          return 0;
78      }
79      //每一条路加上势能差
80      for(int u=1;u<=n;u++){
81          for(int i=head[u];i;i=edge[i].next){
82              edge[i].w+=h[u]-h[edge[i].to];
83          }
84      }
85      for(int i=1;i<=n;i++){
86          dij(i);
87          ll ans=0;
88          for(int j=1;j<=n;j++){
89              if(dis[i]==INF)ans+=j*INF;
90              else ans+=j*(dis[j]+h[j]-h[i]);
91              //减去势能差
92          }
93          cout<<ans<<endl;
94      }
95  }
```

3.17 全局最小割

```
int G[MAX_N][MAX_N];
int v[MAX_N];//v[i]代表节点i合并到的顶点
int w[MAX_N];//定义w(A,x) = ∑w(v[i],x), v[i]∈A
bool visited[MAX_N]; //用来标记是否该点加入了A集合
int squ[MAX_N];  //记录移除的节点次序
int index;//记录最小割的位置，以便分开整个图的节点

int stoer_wagner(int n)
{
    int min_cut = INF,r=0;
    for (int i = 0; i < n; ++i){
        v[i] = i; //初始还未合并，都代表节点本身
    }
    while (n > 1){
        int pre = 0;//pre用来表示之前加入A集合的点（在t之前一个加进去的点）
        memset(visited, 0, sizeof(visited));
        memset(w, 0, sizeof(w));
        for (int i = 1; i < n; ++i){
            //求出 某一轮最大生成树的最后两个节点，并且去除最后的t，将与t连接的边归并
            int k = -1;
            for (int j = 1; j < n; ++j){ //选取V-A中的w(A,x)最大的点x加入集合
                if (!visited[v[j]]){
                    w[v[j]] += G[v[pre]][v[j]];
                    if (k == -1 || w[v[k]] < w[v[j]])
                        k = j;
                }
            }
            visited[v[k]] = true; //标记该点x已经加入A集合
            if (i == n - 1) //若|A|=|V|（所有点都加入了A），结束
            {
                const int s = v[pre], t = v[k];
```

```
32                  //令倒数第二个加入A的点（v[pre]）为s，最后
                        一个加入A的点（v[k]）为t
33                  cout<<t<<"--->"<<s<<endl;
34                  squ[r++]=t;
35                  if(w[t]<min_cut){
36                      min_cut=w[t];
37                      index=r;
38                  }
39                  //min_cut = min(min_cut, w[t]);
40                  // 则s-t最小割为w(A,t)，用其更新min_cut
41                  for (int j = 0; j < n; ++j) //Contract(s,
                        t)
42                  {
43                      G[s][v[j]] += G[v[j]][t];
44                      G[v[j]][s] += G[v[j]][t];
45                  }
46                  v[k] = v[--n];//删除最后一个点（即删除t，
                        也即将t合并到s）
47              }
48              pre = k;
49          }
50      }
51      return min_cut;
52  }
53  int main(int argc, char *argv[]){
54      int n, m;
55      while (scanf("%d%d", &n, &m) != EOF)
56      {
57          memset(G, 0, sizeof(G));
58          while (m--)
59          {
60              int u, v, w;
61              scanf("%d%d%d", &u, &v, &w);
62              G[u][v] += w;
63              G[v][u] += w;
64          }
65          int z=n;
```

```
66          //printf("%d\n", stoer_wagner(n));
67          cout<<"\r\n归并的步骤为："<<endl;
68          int res=stoer_wagner(n);
69          cout<<"\r\n最小割的总权值为：␣"<<res<<"\r\n图划分
                为部分A："";
70          //cout<<"图划分为部分A："";
71          for(int i=0;i<z;i++)
72          {
73              if(i==index)
74                  cout<<"部分B："";
75              cout<<squ[i]<<"␣␣";
76          }
77      }
78      return 0;
79  }
```

4 数据结构

4.1 MultiplyLCA

```
struct Multiply_LCA{
    vector<int>v[maxn];
    int dep[maxn];
    int f[maxn][25];

    void dfs(int x,int fa)
    {       dep[x]=dep[fa]+1;
        f[x][0]=fa;
        for(int i=1;i<=20;i++)
            f[x][i]=f[f[x][i-1]][i-1];
        for(auto y:v[x])
            if(y!=fa)
                RMQ(y,x);
    }
    int LCA(int x,int y)
    {
        if(dep[x]<dep[y])swap(x,y);
        for(int i=20;i>=0;i--)
            if(dep[f[x][i]]>=dep[y])
                x=f[x][i];
        if(x==y)return x;
        for(int i=20;i>=0;i--)
            if(dep[f[x][i]]!=dep[f[y][i]])
                x=f[x][i],y=f[y][i];
        return f[x][0];
    }

};
```

4.2 ST

```
1  struct ST{
2      int f[maxn][30];
3      void init(){
4          for(int i=1;i<=n;i++)//距离为0初始化
5              f[i][0]=p[i];
6          int t=log(n)/log(2)+1;//j定位
7          for(int j=1;j<t;j++){
8              for(int i=1;i<=n-(1<<j)+1;i++){
9                  f[i][j]=max(f[i][j-1],f[i+(1<<(j-1))][j
                       -1]);
10                 //f[i][j-1]为f[i][j]的长为2^j-1的前半段,
                       另一段为f[i+(1<<(j-1))][j-1]
11             }
12         }
13     }
14     int query(int l,int r){
15         int k=log2(r-l+1);
16         return max(f[l][k],f[r-(1<<k)+1][k]);
17     }
18 }st;
```

4.3 GrayCode

```
// n位格雷码
vector<int> grayCode(int n) {
    int count = 1 << n;
    vector<int> res(count,0);
    for(int i = 1 ; i < count; i ++)
    {
        int bin = i,cur = bin >> (n - 1);
        for(int k = n - 1;k > 0;k --)
            cur = (cur << 1) + (((bin >> k) & 1) ^ ((bin
               >>(k - 1)) & 1));
        res[i] = cur;
    }
    return res;
}
//递归实现n位格雷码
vector<string> gray_code(int n){
    if(n==1)return {”0”,”1”};
    else{
        vector<string>v,v1;
        v1=gray_code(n-1);
        for(int i=0;i<v1.size();i++)
            v.push_back(”0”+v1[i]);
        for(int i=(v1.size()-1);i>-1;i--)
            v.push_back(”1”+v1[i]);
        return v;
    }
}
void solve()
{
    int n;
    cin>>n;
    vector<string>v=gray_code(n);
    for(int i=0;i<v.size();i++)
        cout<<v[i]<<endl;
}
```

4.4 归并排序

```
void doit()
{
    int n;
    vector<int>p(n+1,0);
    auto mg_sort = [&](auto mg_sort,int l,int r)->ll{
        if(l>=r)return 0;
        int mid=(l+r)>>1;
        ll t = mg_sort(mg_sort,l,mid)+mg_sort(mg_sort,mid
           +1,r);
        vector<int>temp;
        int i=l,j=mid+1;
        while(i<=mid&&j<=r)
        {
            if(p[i]<=p[j])temp.push_back(p[i++]);
            else t+=mid-i+1,temp.push_back(p[j++]);
        }
        while(i<=mid)temp.push_back(p[i++]);
        while(j<=r)temp.push_back(p[j++]);
        for(int i=l,j=0;i<=r;i++,j++)
            p[i]=temp[j];
        return t;
    };
    mg_sort(mg_sort,1,n);
}
```

4.5 树链剖分

```
struct Seg{int l,r,val,lazy;}t[maxn<<2];
struct Edge{int to,next,w;}edge[maxn];
int head[maxn],cnt;
int siz[maxn],f[maxn],dep[maxn],son[maxn];
int id[maxn],di[maxn],top[maxn],tot;
int w[maxn];
int n,root,q,P;
inline void add(int from,int to,int w=0){
    edge[++cnt]={to,head[from],w};
    head[from]=cnt;
}
inline void pushup(int rt){
    t[rt].val=t[rt<<1].val+t[rt<<1|1].val;
}
inline void pushdown(int rt){
    if(t[rt].lazy){
        (t[rt<<1].val+=(t[rt].lazy*(t[rt<<1].r-t[rt<<1].l
           +1))%P)%=P;
        (t[rt<<1|1].val+=(t[rt].lazy*(t[rt<<1|1].r-t[rt
           <<1|1].l+1))%P)%=P;
        (t[rt<<1].lazy+=t[rt].lazy)%=P;
        (t[rt<<1|1].lazy+=t[rt].lazy)%=P;
        t[rt].lazy=0;
    }
}
inline void build(int rt,int l,int r)
{
    t[rt]={l,r,0,0};
    if(l==r){
        (t[rt].val=w[di[l]])%=P;
        return ;
    }
    int mid=(l+r)>>1;
    build(rt<<1,l,mid);
    build(rt<<1|1,mid+1,r);
```

```
34      pushup(rt);
35  }
36  inline void update(int rt,int l,int r,int k)
37  {
38      if(l<=t[rt].l&&t[rt].r<=r){
39          (t[rt].val+=(k*(t[rt].r-t[rt].l+1))%P)%=P;
40          (t[rt].lazy+=k)%=P;
41          return ;
42      }
43      pushdown(rt);
44      int mid=(t[rt].l+t[rt].r)>>1;
45      if(l<=mid)update(rt<<1,l,r,k);
46      if(r>mid)update(rt<<1|1,l,r,k);
47      pushup(rt);
48  }
49  inline int query(int rt,int l,int r){
50      if(l<=t[rt].l&&t[rt].r<=r)
51          return t[rt].val;
52      pushdown(rt);
53      int mid=(t[rt].l+t[rt].r)>>1;
54      int ans=0;
55      if(l<=mid)(ans+=query(rt<<1,l,r))%=P;
56      if(r>mid)(ans+=query(rt<<1|1,l,r))%=P;
57      return ans;
58  }
59  inline void upRange(int x,int y,int k)
60  {
61      while(top[x]!=top[y]){
62          if(dep[top[x]]<dep[top[y]])swap(x,y);
63          update(1,id[top[x]],id[x],k);
64          x=f[top[x]];
65      }
66      if(dep[x]>dep[y])swap(x,y);
67      update(1,id[x],id[y],k);
68  }
69  inline void upSon(int x,int k){
70      update(1,id[x],id[x]+siz[x]-1,k);
```

```
}
inline int qRange(int x,int y)
{
    int ans=0;
    while(top[x]!=top[y]){
        if(dep[top[x]]<dep[top[y]])swap(x,y);
        (ans+=query(1,id[top[x]],id[x]))%=P;
        x=f[top[x]];
    }
    if(dep[x]>dep[y])swap(x,y);
    (ans+=query(1,id[x],id[y]))%=P;
    return ans;
}
inline int qSon(int x){
    return query(1,id[x],id[x]+siz[x]-1);
}
inline int LCA(int x,int y)
{
    while(top[x]!=top[y]){
        if(dep[top[x]]<dep[top[y]])swap(x,y);
        x=f[top[x]];
    }
    if(dep[x]>dep[y])swap(x,y);
    return x;
}
inline void dfs1(int x)
{
    siz[x]=1;
    for(int i=head[x];i;i=edge[i].next){
        int y=edge[i].to;
        if(y==f[x])continue;
        f[y]=x;dep[y]=dep[x]+1;
        dfs1(y);
        siz[x]+=siz[y];
        if(siz[y]>siz[son[x]])
            son[x]=y;
    }
```

```
108 }
109 inline void dfs2(int x,int topf)
110 {
111     id[x]=++tot;
112     di[tot]=x;
113     top[x]=topf;
114     if(son[x])dfs2(son[x],topf);
115     for(int i=head[x];i;i=edge[i].next)
116     {
117         int y=edge[i].to;
118         if(y==f[x]||y==son[x])continue;
119         dfs2(y,y);
120     }
121 }
122 void solve()
123 {
124     cin>>n>>q>>root>>P;
125     for(int i=1;i<=n;i++)
126         cin>>w[i];
127     for(int i=1;i<n;i++)
128     {
129         int x,y;
130         cin>>x>>y;
131         add(x,y);
132         add(y,x);
133     }
134     f[root]=0;
135     dfs1(root);
136     dfs2(root,root);
137     build(1,1,n);
138     while(q--)
139     {
140         int op,x,y,k;
141         cin>>op;
142         if(op==1)
143         {
144             cin>>x>>y>>k;
```

```
145             upRange(x,y,k);
146         }
147         else if(op==2)
148         {
149             cin>>x>>y;
150             cout<<qRange(x,y)<<endl;
151         }
152         else if(op==3)
153         {
154             cin>>x>>k;
155             upSon(x,k);
156         }
157         else
158         {
159             cin>>x;
160             cout<<qSon(x)<<endl;
161         }
162     }
163 }
```

4.6 扫描线线段树

```
struct Seg{
    int l,r,len,cv,_l,_r,cnt;
}t[maxn];
pp(4) p[maxn];
int w[maxn];
int n,m,cnt;
void build(int rt,int l,int r)
{
    t[rt]={l,r,0,0,0,0,0};
    if(l==r)return ;
    int mid=(l+r)>>1;
    build(rt<<1,l,mid);
    build(rt<<1|1,mid+1,r);
}
void pushup(int rt)
{
    if(t[rt].cv)
    {
        t[rt]._l=t[rt]._r=1;
        t[rt].len=w[t[rt].r+1]-w[t[rt].l];
        t[rt].cnt=1;
    }
    else if(t[rt].l==t[rt].r)
    {
        t[rt]._l=t[rt]._r=0;
        t[rt].len=0;
        t[rt].cnt=0;
    }
    else
    {
        t[rt]._l=t[rt<<1]._l;
        t[rt]._r=t[rt<<1|1]._r;
        t[rt].len=t[rt<<1].len+t[rt<<1|1].len;
        t[rt].cnt=t[rt<<1].cnt+t[rt<<1|1].cnt-(t[rt<<1]._r
           &t[rt<<1|1]._l);
```

```
35      }
36  }
37  void update(int rt,int l,int r,int c)
38  {
39      if(l<=t[rt].l&&t[rt].r<=r)
40      {
41          t[rt].cv+=c;
42          pushup(rt);
43          return ;
44      }
45      int mid=(t[rt].l+t[rt].r)>>1;
46      if(l<=mid)update(rt<<1,l,r,c);
47      if(r>mid)update(rt<<1|1,l,r,c);
48      pushup(rt);
49  }
50  void solve()
51  {
52      cin>>n;
53      for(int i=1;i<=n;i++)
54      {
55          int x,y,_x,_y;
56          cin>>x>>y>>_x>>_y;
57          w[++m]=y;
58          w[++m]=_y;
59          p[i*2-1]={x,y,_y,1};
60          p[i*2]={_x,y,_y,-1};
61      }
62      sort(w+1,w+1+m);
63      m=unique(w+1,w+1+m)-w-1;
64      sort(p+1,p+1+n+n,[&](pp(4)a,pp(4)b){
65          if(a[0]==b[0])return a[3]>b[3];
66          return a[0]<b[0];
67      });
68      for(int i=1;i<=n<<1;i++)
69      {
70          p[i][1]=lower_bound(w+1,w+1+m,p[i][1])-w;
71          p[i][2]=lower_bound(w+1,w+1+m,p[i][2])-w;
```

```
72      }
73      build(1,1,m);
74      int pre=0;
75      int ans=0;
76      for(int i=1;i<=n<<1;i++)
77      {
78          auto [x,y,_y,op]=p[i];
79          update(1,y,_y-1,op);
80          ans+=abs(t[1].len-pre);
81          ans+=(p[i+1][0]-p[i][0])*2*t[1].cnt;
82          pre=t[1].len;
83      }
84      cout<<ans<<endl;
85  }
```

4.7 莫队

```
1  struct Query{
2      int l,r,id,block;
3      bool operator < (const Query& q)const
4      {
5          if(block==q.block)
6              return r<q.r;
7          else
8              return block<q.block;
9      }
10 }p[maxn];
11 int cnt_a[maxn],cnt_b[maxn];
12 int a[maxn],b[maxn];
13 bool ans[maxn];
14 int n,m,ok=0,cnt=0;
15 void Debug()
16 {
17     printf("[Debug]:");
18     int ma=0;
19     for(int i=1;i<=n;i++)
20         ma=max({ma,a[i],b[i]});
21     for(int i=1;i<=ma;i++)
22         cout<<cnt_a[i];cout<<endl;
23     printf("[Debug]:");
24     for(int i=1;i<=ma;i++)
25         cout<<cnt_b[i];
26     cout<<endl;
27     cout<<cnt<<endl;
28 }
29 inline void upd_a(int x,int fl)
30 {
31     int fl_1=0,fl_2=0;
32     if((cnt_a[x]>0&&cnt_b[x]>0)||(cnt_a[x]==0&&cnt_b[x
          ]==0))fl_1=1;
33     if((cnt_a[x]+fl>0&&cnt_b[x]>0)||(cnt_a[x]+fl==0&&cnt_b
          [x]==0))fl_2=1;
```

```
34      if(fl_1>fl_2)cnt++;
35      else if(fl_1<fl_2)cnt--;
36      cnt_a[x]+=fl;
37      if(cnt)ok=1;
38      else ok=0;
39  }
40  inline void upd_b(int x,int fl)
41  {
42      int fl_1=0,fl_2=0;
43      if((cnt_b[x]>0&&cnt_a[x]>0)||(cnt_b[x]==0&&cnt_a[x
            ]==0))fl_1=1;
44      if((cnt_b[x]+fl>0&&cnt_a[x]>0)||(cnt_b[x]+fl==0&&cnt_a
            [x]==0))fl_2=1;
45      if(fl_1>fl_2)cnt++;
46      else if(fl_1<fl_2)cnt--;
47      cnt_b[x]+=fl;
48      if(cnt)ok=1;
49      else ok=0;
50  }
51  void solve()
52  {
53      cin>>n>>m;
54      for(int i=1;i<=n;i++)
55          cin>>a[i];
56      for(int i=1;i<=n;i++)
57          cin>>b[i];
58      int tmp=sqrt(n);
59      for(int i=1;i<=m;i++)
60      {
61          cin>>p[i].l>>p[i].r;
62          p[i].id=i;
63          p[i].block=i/tmp;
64      }
65      sort(p+1,p+1+m);
66      for(int i=1;i<=max(p[1].l,p[1].r);i++)
67      {
68          if(i<=p[1].l)
```

```
69              upd_a(a[i],1);
70          if(i<=p[1].r)
71              upd_b(b[i],1);
72      }
73      ans[p[1].id]=ok;
74      int l=p[1].l;
75      int r=p[1].r;
76      // Debug();
77      for(int i=2;i<=m;i++)
78      {
79          while(l<p[i].l)//l++;
80              l++,upd_a(a[l],1);
81          while(l>p[i].l)//l--;
82              upd_a(a[l],-1),l--;
83          while(r<p[i].r)//r++;
84              r++,upd_b(b[r],1);
85          while(r>p[i].r)//r--;
86              upd_b(b[r],-1),r--;
87          ans[p[i].id]=ok;
88          // Debug();
89      }
90      for(int i=1;i<=m;i++)
91      {
92          if(ans[i])cout<<"NO"<<endl;
93          else cout<<"YES"<<endl;
94      }
95  }
```

4.8 点分治 1

```
#include<bits/stdc++.h>
using namespace std;
#define endl ”\n”
const int INF=10000005;
const int maxn=1e7+10;
const int N=2e4+10;
struct Edge{int to,next,w;}edge[N];
int head[N],_cnt;
int del[N],siz[N],maxson,root,sum;
int dis[N],d[N],cnt;
int ans[N],q[maxn],judge[maxn],ask[maxn];
int n,m;
void add(int from,int to,int w){
    edge[++_cnt].w=w;
    edge[_cnt].to=to;
    edge[_cnt].next=head[from];
    head[from]=_cnt;
}
void getroot(int x,int fa)
{
    siz[x]=1;
    int sx=0;
    for(int i=head[x];i;i=edge[i].next)
    {
        int y=edge[i].to;
        if(y==fa||del[y])continue;
        getroot(y,x);
        siz[x]+=siz[y];
        sx=max(sx,siz[y]);
    }
    sx=max(sx,sum-siz[x]);
    if(sx<maxson)maxson=sx,root=x;
}
void getdis(int x,int fa)
{
```

```
36      dis[++cnt]=d[x];
37      for(int i=head[x];i;i=edge[i].next)
38      {
39          int y=edge[i].to;
40          if(y==fa||del[y])continue;
41          d[y]=d[x]+edge[i].w;
42          getdis(y,x);
43      }
44  }
45  void calc(int x)
46  {
47      judge[0]=1;
48      int p=0;
49      for(int i=head[x];i;i=edge[i].next)
50      {
51          int y=edge[i].to;
52          if(del[y])continue;
53          cnt=0;
54          d[y]=edge[i].w;
55          getdis(y,x);
56          for(int j=1;j<=cnt;j++)
57              for(int k=1;k<=m;k++)
58                  if(ask[k]>=dis[j])
59                      ans[k]|=judge[ask[k]-dis[j]];
60          for(int j=1;j<=cnt;j++)
61              if(dis[j]<INF)
62                  q[++p]=dis[j],judge[q[p]]=1;
63      }
64      for(int i=1;i<=p;i++)
65          judge[q[i]]=0;
66  }
67  void divide(int x)
68  {
69      calc(x);
70      del[x]=1;
71      for(int i=head[x];i;i=edge[i].next)
72      {
```

```
73          int y=edge[i].to;
74          if(del[y])continue;
75          maxson=sum=siz[y];
76          getroot(y,0);
77          divide(root);
78      }
79  }
80  char ch[maxn];
81  void solve()
82  {
83      cin>>n>>m;
84      for(int i=1;i<n;i++)
85      {
86          int x,y,w;
87          cin>>x>>y>>w;
88          add(x,y,w);
89          add(y,x,w);
90      }
91      for(int i=1;i<=m;i++)
92          cin>>ask[i];
93      maxson=sum=n;
94      getroot(1,0);
95      getroot(root,0);
96      divide(root);
97      for(int i=1;i<=m;i++)
98          cout<<(ans[i]?"AYE\n":"NAY\n");
99  }
100 signed main(){
101     ios::sync_with_stdio(false);
102     cin.tie(nullptr);cout.tie(nullptr);
103         solve();
104     return 0;
105 }
```

4.9 点分治 2

```
#include<bits/stdc++.h>
using namespace std;
#define endl ”\n”
const int INF=10000005;
const int maxn=1e7+10;
const int N=2e4+10;
struct Edge{int to,next,w;}edge[N];
int head[N],_cnt;
int del[N],siz[N],maxson,root,sum;
int dis[N],d[N],cnt;
int ans[N],q[maxn],judge[maxn],ask[maxn];
int n,m;
void add(int from,int to,int w){
    edge[++_cnt].w=w;
    edge[_cnt].to=to;
    edge[_cnt].next=head[from];
    head[from]=_cnt;
}
void getroot(int x,int fa)
{
    siz[x]=1;
    int sx=0;
    for(int i=head[x];i;i=edge[i].next)
    {
        int y=edge[i].to;
        if(y==fa||del[y])continue;
        getroot(y,x);
        siz[x]+=siz[y];
        sx=max(sx,siz[y]);
    }
    sx=max(sx,sum-siz[x]);
    if(sx<maxson)maxson=sx,root=x;
}
void getdis(int x,int fa)
{
```

```
36      dis[++cnt]=d[x];
37      for(int i=head[x];i;i=edge[i].next)
38      {
39          int y=edge[i].to;
40          if(y==fa||del[y])continue;
41          d[y]=d[x]+edge[i].w;
42          getdis(y,x);
43      }
44  }
45  void calc(int x,int w,int op)
46  {
47      cnt=0,d[x]=w;
48      getdis(x,0);
49      sort(dis+1,dis+1+cnt);
50      for(int i=1;i<=m;i++)
51      {
52          int l=1,r=cnt;
53          while(l<r){
54              if(dis[l]+dis[r]<=ask[i]){
55                  if(dis[l]+dis[r]==ask[i])ans[i]+=op;
56                  ++l;
57              }
58              else --r;
59          }
60      }
61  }
62  void divide(int x)
63  {
64      calc(x,0,1);
65      del[x]=1;
66      for(int i=head[x];i;i=edge[i].next)
67      {
68          int y=edge[i].to;
69          if(del[y])continue;
70          calc(y,edge[i].w,-1);
71          maxson=sum=siz[y];
72          getroot(y,0);
```

```
73          divide(root);
74      }
75  }
76  char ch[maxn];
77  void solve()
78  {
79      cin>>n>>m;
80      for(int i=1;i<n;i++)
81      {
82          int x,y,w;
83          cin>>x>>y>>w;
84          add(x,y,w);
85          add(y,x,w);
86      }
87      for(int i=1;i<=m;i++)
88          cin>>ask[i];
89      maxson=sum=n;
90      getroot(1,0);
91      getroot(root,0);
92      divide(root);
93      for(int i=1;i<=m;i++)
94          cout<<(ans[i]?"AYE\n":"NAY\n");
95  }
96  signed main(){
97      ios::sync_with_stdio(false);
98      cin.tie(nullptr);cout.tie(nullptr);
99          solve();
100     return 0;
101 }
```

5 数论

5.1 ExGcd

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

5.2 快速幂

```
int mull(int a,int b){int qw=0;while(b){if(b&1)qw=(qw+a)%mod;a=(a+a)%mod;b>>=1;}return qw;}

int ksm(int b,int p){int r=1;b%=mod;while(p){if(p&1)r=(r*b)%mod;p>>=1;b=(b*b)%mod;}return r;}

int ksm(int b,int p,int mod){int r=1;b%=mod;while(p){if(p&1)r=(r*b)%mod;p>>=1;b=(b*b)%mod;}return r;}
```

5.3 整除分块

```
ll division_block(ll n){
    ll res = 0;
    for(ll l = 1, r; l <= n; l = r + 1){
        r = n / (n / l);
        // cout<<r<<endl;
        res += n / l * (r - l + 1);
    }
    return res;
}
```

5.4 欧拉函数

```
int prime[maxn],phi[maxn];
bool vis[maxn];
int n;
void doit()
{
    for(int i=2;i<=maxn;i++){
        if(!vis[i])prime[++prime[0]]=i,phi[i]=i-1;
        for(int j=1;j<=prime[0]&&prime[j]*i<=maxn;j++){
            vis[i*prime[j]]=1;
            if(i%prime[j]==0){
                phi[i*prime[j]]=phi[i]*prime[j];
                break;
            }
            phi[i*prime[j]]=phi[i]*(prime[j]-1);
        }
    }
}
int PHI(int x)
{
    //O(sqrt(x)
    int ans=x;
    for(int i=2;i*i<=x;i++)
        if(x%i==0)
        {
            while(x%i==0)x/=i;
            ans=ans*(i-1)/i;
        }
    if(x>1)ans=ans*(x-1)/x;
    return ans;
}
```

5.5 等比数列求和

```
1  int ksm(int b,int p,int mod){int r=1;b%=mod;while(p){if(p
       &1)r=(r*b)%mod;p>>=1;b=(b*b)%mod;}return r;}
2  int sumq(int q,int n,int p)
3  {
4      if(n==1)return 1;
5      if(n%2==0)return ((1+ksm(q,n/2,p))*sumq(q,n/2,p)%p)%p;
6      else return (1+q*sumq(q,n-1,p))%p;
7  }
8  // q^0 + q^1 + q^2 +q^(n-1)
9  void solve()
10 {
11     int q,n,p;
12     cin >> q >> n >> p;
13     cout << sumq(q,n,p) % p << endl;
14 }
```

5.6 素数

```
1  bool su(int x)
2  {
3      if(x==1)return 0;
4      if(x==2||x==3)return 1;
5      if(x%6!=5&&x%6!=1)return 0;
6      int tmp=sqrt(x);
7      for(int i=5;i<=tmp;i+=6)
8          if(x%i==0||x%(i+2)==0)
9              return 0;
10     return 1;
11 }
```

5.7 排列组合其一

```
int gcd(int a,int b){
    if(b==0)return a;
    else return gcd(b,a%b);
}
//解线性同余方程，扩展欧几里德定理
int x,y;
void Extended_gcd(int a,int b){
    if(b==0){x=1;y=0;}
    else{Extended_gcd(b,a%b);long t=x;x=y;y=t-(a/b)*y;}
}
//计算不大的C(n,m)
int C(int a,int b){
    if(b>a)return 0;b=(ff[a-b]%mod*ff[b])%mod;a=ff[a];
    int c=gcd(a,b);a/=c;b/=c;Extended_gcd(b,M);x=(x+mod)%
       mod;
    x=(x*a)%mod;return x;
}
//Lucas定理
int Combination(int n, int m)
{
    int ans=1;int a,b;
    while(m||n){a=n%mod;b=m%mod;n/=mod;m/=mod;
        ans=(ans*C(a,b))%mod;}
    return ans;
}
int A(int n,int m){return (Combination(n,m)*ff[m])%mod;}

signed main()
{
    int i,m,n;
    ff[0]=1;
    for(int i=1; i<=M; i++) //预计算n!
        ff[i]=(ff[i-1]*i)%mod;
    while(~scanf("%lld%lld",&n, &m))
    {
```

```
35          printf("%lld\n",Combination(n,m));
36      }
37      return 0;
38  }
```

5.8 排列组合其二

```
 1  ll mod=998244353;
 2  ll qmi(ll a, ll k)//快速幂模板
 3  {
 4      int res = 1;
 5      while (k)
 6      {
 7          if (k & 1) res = (ll)res * a % mod;
 8          a = (ll)a * a % mod;
 9          k >>= 1;
10      }
11      return res;
12  }
13  ll C(ll a, ll b)//通过定理求组合数C(a, b)
14  {
15      ll res = 1;
16      for (int i = 1, j = a; i <= b; i ++, j -- )
17      {
18          res = (ll)res * j % mod;
19          res = (ll)res * qmi(i, mod - 2) % mod;
20      }
21      return res;
22  }
23  ll lucas(ll a, ll b)
24  {
25      if (a < mod && b < mod) return C(a, b);
26      return (ll)C(a % mod, b % mod)*lucas(a / mod, b / mod)
            % mod;
27  }
```

5.9 排列组合其三

```
int ksm(int b,int p){int r=1;b%=mod;while(p){if(p&1)r=(r*b)%mod;p>>=1;b=(b*b)%mod;}return r;}

int C(int n,int m){
    return F[n]*INF[n-m]*INF[m];
}
F[0]=INF[0]=1;
for(int i=1;i<M;i++)
{
    F[i]=F[i-1]*i%mod;
    INF[i]=ksm(F[i],mod-2);
}
```

5.10 单调队列维护 DP

```
int pre[N];
int dp[N];
int q[N];
int n,m;
void solve()
{
    cin>>n>>m;
    for(int i=1;i<=n;i++)
    {
        int v,w,s;
        cin>>v>>w>>s;
        // 复制上一次的i-1 的DP
        memcpy(pre,dp,sizeof(dp));
        //  枚举余数为J
        for(int j=0;j<v;j++)
        {
            int head=0;
            int tail=-1;
            for(int k=j;k<=m;k+=v)
            {
                //k-q[head]>s*v 长度长了，最多s个
                if(head<=tail&&q[head]<k-s*v)
                    head++;
                while(head<=tail&&pre[q[tail]]-(q[tail]-j)
                   /v*w<=pre[k]-(k-j)/v*w)
                    tail--;
                if(head<=tail)
                    dp[k]=max(dp[k],pre[q[head]]+(k-q[head
                       ])/v*w);
                q[++tail]=k;
            }
        }
    }
    cout<<dp[m]<<endl;
}
```

6 计算几何

6.1 绕点旋转后坐标

```
pii f(double x1,double y1,double x0,double y0,double d)
{
    d=d/180*acos(-1);//如果是360一周
    double x=(x1-x0)*cos(d)-(y1-y0)*sin(d)+x0;
    double y=(x1-x0)*sin(d)+(y1-y0)*cos(d)+y0;
    return {x,y};
}
```

6.2 正方形已知两点求另外两点

```
void solve()
{
    cin>>a>>b>>c>>d;
    double x=(a-b+c+d)*0.5;
    double y=(a+b-c+d)*0.5;
    if(ceil(x) == floor(x)&&ceil(y) == floor(y))cout<<(int
       )x<<"␣"<<(int)y<<endl;
    else cout<<"No␣Answer!"<<endl;
}
```

6.3 三角形

```
class triangle{
    double S(pii x,pii y,pii z)
    {
        double s=0.5*abs((y.first-x.first)*(z.second-x.
           second)-(z.first-x.first)*(y.second-x.second));
        return s;
    }
    double S(double x,double y,double z)
    {
```

```
 9          //海伦公式
10          double p = 0.5*(x+y+z);
11          double s;
12          s=sqrt(p*(p-x)*(p-y)*(p-z));
13          // 或者
14          s=0.25*sqrt((x+y+z)*(x+y-z)*(x+z-y)*(y+z-x));
15          return s;
16      }
17      //能否构成三角形
18      bool ok(pii x,pii y,pii z)
19      {
20          if(x.first!=y.first&&y.first!=z.first&&z.first!=x.
               first)
21          {
22              double k1 = 1.0 * (y.second - x.second) / (y.
                   first - x.first);
23              double k2 = 1.0 * (z.second - y.second) / (z.
                   first - y.first);
24              double k3 = 1.0 * (z.second - x.second) / (z.
                   first - x.first);
25              if(k1 == k2 && k2 == k3)
26                  return 0;
27              else
28                  return 1;
29          }
30          else if(x.first == y.first && y.first == z.first)
31              return 0;
32          else
33          {
34              if(x.second == y.second && y.second == z.
                   second)
35                  return 0;
36              else
37                  return 1;
38          }
39      }
40      //能否构成三角形
```

```
41      bool ok(int x1,int y1,int x2,int y2,int x3,int y3)
42      {
43          if (x1 != x2 && x2 != x3 && x3 != x1)
44          {
45              double k1 = 1.0 * (y2 - y1) / (x2 - x1);
46              double k2 = 1.0 * (y3 - y2) / (x3 - x2);
47              double k3 = 1.0 * (y3 - y1) / (x3 - x1);
48              if (k1 == k2 && k2 == k3)
49                  return 0;
50              else
51                  return 1;
52
53          }
54          else if (x1 == x2 && x2 == x3)
55                  return 0;
56          else
57          {
58              if (y1 == y2 && y2 == y3)
59                  return 0;
60              else
61                  return 1;
62          }
63      }
64  };
```

7 网络流建图

7.1 最小路径覆盖

最小路径覆盖：在一个有向无环图中，找出最少的路径，使得这些路径经过了所有的点。最小路径覆盖分为**最小不相交路径覆盖**和**最小可相交路径覆盖**，区别是这些路径是否可以相交

7.1.1 最小不相交路径覆盖

建图方法：把原图的每个点 u 拆成两个点 u_1,u_2，如果有一条有向边 (a,b)，则连边 (a_2,b_1)，容易发现这是一个二分图，那么用下面的定理就可以求出答案

定理： 最小路径覆盖 = 原图节点数-新图最大匹配数

证明：

一开始每个点都是一条路径，每次找一条匹配边，代表合并两条路径

由于路径不相交（即每个点的入度和出度至少有一个为 1），所以二分图上的边也不相交

（如果相交则说明某个点的入度或出度大于 1），这正好是匹配的定义

每条匹配边代表答案 -1，所以**最小路径覆盖 = 原图节点数-新图最大匹配数**

7.1.2 最小可相交路径覆盖

对原图传递闭包，即若原图中 (u,v) 连通，则增加边 (u,v)．这可以用 $Floyd$ 算法 $O(n^3)$ 实现。然后对新图做最小不相交路径覆盖即可。因为在原图中相交的路径在传递闭包后可以拆分成另一条边，这样就不相交了

7.1.3 最多不相交路径

这种问题变化比较多，但都能表示成以下形式：**已知一些路径，每个节点只能属于一条路径，求能选择多少条路径使它们不相交．**

主要的方法是拆点，将一个点拆成两个，然后连边，容量表示该点最多经过次数

7.2 最小割

7.2.1 最大权闭合子图

定义：有一个有向图，每一个点都有一个权值，选择一个权值和最大的子图，使得每个点的后继都在子图里面，这个子图就叫最大权闭合子图。

建图方法：从源点 s 向每个正权点连一条容量为权值的边，每个负权点向汇点 t 连一条容量为权值的绝对值的边，有向图原来的边容量全部为无限大。

定理： 最大权闭合子图 = 所有正权点之和-最小割

关键性质： 如果 s 与 i 有边，表示 i 在子图中。如果 i 与 t 有边，表示 i 不在于子图中。即：割掉 s 与 i 表示不选 i，割掉 i 与 t 表示选 i。

性质 1： 原图之间的边一定不会被割掉

边权为无穷大，当然不会被选进最小割

性质 2： 只有 s 到 t 不联通时，才得到最大权闭合子图

反证法：若 s 到 t 连通，则一定存在节点 i,j 使 s 到 i 有边，i 到 j 有边（引理 1），j 到 t 有边．而根据性质 1:i 在子图中，j 不在子图中，这与最大权闭合子图的定义矛盾，证毕由引理 2 可得，图的一个割就是一个闭合子图

由于一个割的边权和 = 不选的正权点 + 选的负权点绝对值 = 不选的正权点 − 选的负权点一个割的边权和 = 不选的正权点 + 选的负权点绝对值 = 不选的正权点 − 选的负权点．

闭合子图 = 正权点 + 负权点 = 所有正权和 − 不选的正权点 + 选的负权点 = 所有正权和 − 割的边权和闭合子图 = 正权点 + 负权点 = 所有正权和 − 不选的正权点 + 选的负权点 = 所有正权和 − 割的边权和

显然割的边权和最小的时候得到最大权闭合子图，证毕

7.3 最小割

定理： 二分图最大独立集 =n-二分图最大匹配

其实二分图独立集是**特殊的一种最大权闭合子图**。我们根据上文“收益”的思想，把选某个点的收益看为 1，左部节点为正权点，右部节点为负权点．按照最大权闭合子图的方式建图，答案为正权和-最小割 =n-最小割 =n-最大流。我们发现把最大权闭合子图中 INF 的边换成 1 也不影响答案，因为图中其他边的容量都为 1。这样图就变成了二分图匹配中的图，最大流 = 二分图最大匹配

7.4 最大密度子图

定义： 图的密度是图上的边的数量除以点数。求密度最大的子图。

建图： 看到平均数想到 01 分数规划。二分答案 mid，那么问题转化为判定是否存在一个子图，使得边数 $-mid\cdot$ 点数 $>$

$$0$$

边数 $-mid\cdot$ 点数 >0．那么可以把每条边的权看成 1，每个点的权看成 $-mid$，限制是选择一条边就必须选择边连接的两个点。于是把边看成左部点，点看成右部点，跑最大权闭合子图，若答案 >0，则合法。

7.5 二元关系最小割模型

定义： 有若干个变量，每个变量有 2 种取值，有若干个现在，每个限制形如”若变量 $x = a, y = b$，就要付出 c 的代价”。最大化所有变量的值之和减去最小代价。

建图： 每个变量建一个点，S 到 x 连边表示 x 的一种取值的代价，x 到 T 连边表示 x 的另一种取值的代价。对于一个限制，在两个点之间连边。边权需要列方程解出。

7.6 二分图带权匹配

定义： 每条边有边权，求匹配边权值之和最大的匹配

建图： 在边上加上权，跑费用流即可

7.7 最大权不相交路径

定义： 每条路径有一个权值（一般是边权和），在 ** 不相交路径数最多的情况下 **，最大化费用

建图： 同最多不相交路径，在连接两个拆点的边上加上费用跑费用流即可

7.8 不等式差分模型（网络流解线性规划）

定义： 对于一些不太好直接想到建图的问题，我们可以数学建模，列出方程然后用线性规划求解。这样的好处是思维量较小，只要做代数变换就可以建图，而不用考虑建图的实际意义。我们需要把式子做差，使得每个未知数仅在两个等式中出现。

根据网络流中每个点流量平衡的思想，我们可以把 $-xi$ 看成从点 i 流出 x_i 的流量，$+x_i$ 看成流入 x_i 的流量。等式为 0 就代表流量平衡。

建图： 每个等式为图中一个顶点，添加源点 S 和汇点 T。

如果一个等式右边为非负整数 c，从源点 S 向该等式对应的顶点连接一条容量为 c，权值为 0 的有向边；如果一个等式右边为负整数 c，从该等式对应的顶点向汇点 T 连接一条容量为 c，权值为 0 的有向边。

如果一个变量 x_i 在第 j 个等式中出现为 x_i，在第 k 个等式中出现为 $-x_i$，且在目标函数里的系数为 c_i，从顶点 j 向顶点 k 连接一条容量为 $+\infty$，费用为 c_i 的有向边。- 如果一个变量 y_i 在第 j 个等式中出现为 y_i，在第 k 个等式中出现为 $-y_i$，且在目标函数里没有出现，从顶点 j 向顶点 k 连接一条容量为 $+\infty$，权值为 0 的有向边。

7.9 有上下界的网络流

7.9.1 无源汇有上下界可行流

定义： 无源汇网络指的是没有源点和汇点，每个点都有入边和出边且满足流量守恒的网络。在这个网络上求一个流量方案，使得每条边的流量必须在 $[l_i, r_i]$ 之间，且每个点流量守恒。

有上下界的费用流的核心是”补偿”。我们先假设每条边的流量均为 l_i，那么一定会有一些点流量不守恒。现在我们需要构造一个附加网络，使得把附加网络和原网络叠加（即对应边流量相加）之后的图满足流量守恒。

因为 Dinic 只能求有源汇最大流，所以是不能直接求出附加网络的流量的。那么我们可以在附加网络上添加一些不在原网络上的边和点，来实现我们的限制.

记 d_i= 点 D 的入流 − 点 i 的出流，然后建附加网络：

1. 新建源点 ss 和汇点 tt

2. 对于原图中的每条边 $e_i = (u, v)$，连边 $(u, v, r_i - l_i)$，也就是说附加网络包括原网络的边。

3. 新建边来满足流量守恒

若 $d_i = 0$ 则该点流量平衡，不用处理

若 $d_i > 0$ 则入流 D> 出流，那么附加网络中 i 的出边需要增加流量，我们连边 (ss, i, d_i)，这样求最大流的时候出边的流量会增加 $d + i$，叠加后满足流量守恒

若 $d_[i] < 0$ 则入流 < 出流，那么附加网络中 i 的入边需要增加流量，同理连边 $(i, tt, -d_i)$，这样求最大流的时候入边的流量会增加 $-d_i$，叠加后满足流量守恒

那么当且仅当步骤 3 中新建边满流时有解，总可行流为 $maxflow(ss, tt) + \sum l_i$。每条边在原图中流量 = 容量下界 + 附加流中它的流量

7.9.2 有源汇有上下界可行流

定义： 在有源汇网络上求一个流量方案，使得每条边的流量必须在 $[l_i, r_i]$ 之间，且除源汇外每个点流量守恒。

设原网络的源和汇分别为 s, t 我们在原网络上加一条边 $(t, s, +\infty)$，相当于把到汇点的所有流量都流回源点，这样每个点流量都守恒。然后套无源汇的方法即可。注意总流量 =t 到 s 的无穷边在原图中的流量

7.9.3 有源汇有上下界最大流和最小流

定义： 在有源汇网络上求一个流量方案，使得每条边的流量必须在 $[l_i, r_i]$ 之间，且除源汇外每个点流量守恒。在这个条件下使得总流量最大/最小。

先按上面的方法求出一个有源汇有上下界可行流. 然后在附加网络上再跑一次 s 到 t 的最大流（注意不是 ss,tt!)。最大流 = 可行流 + 第二次跑的 s 到 t 最大流。

再跑一次最大流是因为附加网络上属于原图的边还有流量没被“榨干”。容易发现只要附加网络上不属于原图的边满流，属于原图的边怎么跑流量都是守恒的。因为第一次跑最大流已经保证所有点守恒，第二次跑最大流不会经过不属于原图的边，因此等价于对原图跑一次普通的最大流，除源汇外流量守恒。两次合起来总流量一定守恒，这就保证了正确性。

同理求最小流就跑一次 t 到 s 的最大流。最小流 = 可行流-第二次跑的 t 到 s 最大流。这是因为 Dinic 过程中反向边的流量增加等价于正向边的的流量减少。

7.9.4 有源汇有上下界最小费用流

定义： 在有源汇网络上求一个流量方案，使得每条边的流量必须在 $[l_i, r_i]$ 之间，且除源汇外每个点流量守恒。每条边单位流量的费用为 c_i. 在这个条件下使得总费用最小，费用定义同一般费用流。(不要求总流量最大)

这是有上下界费用流常被误解的一点，即最小费用流求的是费用最小的可行流，而不是最大流。

因此按有源汇可行流的方法建图，把原图中的边带上费用。总费用

$$= mincost(ss, tt) + \sum l_i c_i$$