

ACM 算法模板

Buns_out

2024 年 4 月 20 日



目录

1	STL	6
1.1	快读快写	6
1.2	自定义开 02 开栈	7
1.3	set	8
1.4	string	8
1.5	bitset	8
1.6	python math 库	9
1.7	rand()、测速、小数	9
1.8	vector	9
1.9	struct	11
1.10	STL 函数	12
1.11	差分矩阵	14
1.12	前缀和矩阵	15
1.13	分治，最大字段和	15
1.14	模拟退火	18
1.15	约瑟夫环问题	20
1.16	PBDS	21
1.16.1	超快哈希 MAP	21
1.16.2	Tree	22
1.16.3	平衡树	24
2	字符串	27
2.1	kmp	27
2.2	kmp 可持久化（优化）	29
2.3	exKMP	31
2.4	Manacher	32
2.5	字符串哈希	33
2.6	最小表示法	39
2.7	AC 自动机	40
2.8	SA 后缀数组	43
2.9	SAM 后缀数组	47
2.10	PAM 回文自动机	52
2.11	序列自动机	54
3	搜索	55
3.1	精确 DLX	55

4	图论	59
4.1	差分约束	59
4.2	Floyd	59
4.3	2-SET	60
4.4	倍增优化建图	62
4.5	prim	63
4.6	匈牙利	66
4.7	KM	68
4.8	点双	71
4.9	边双	73
4.10	仙人掌图求环路径长度	74
4.11	DINIC(超级无敌快!)	77
4.12	MCMF	79
4.13	同余最短路	81
4.14	朱流算法 (Edmonds)	84
4.15	最小生成树 Boruvka	87
4.16	MST	89
4.17	基尔霍夫矩阵	99
4.18	全源最短路	101
4.19	全局最小割	104
4.20	前缀和优化建图	107
5	数据结构	109
5.1	MultiplyLCA	109
5.2	一维 ST	110
5.3	二维 ST	111
5.4	划分树	112
5.5	GrayCode	114
5.6	归并排序	115
5.7	树链剖分	116
5.8	扫描线线段树	121
5.9	莫队	124
5.10	点分治 1	127
5.11	点分治 2	130
6	数论	133
6.1	ExGcd	133
6.2	快速幂	133

6.3	整除分块	133
6.4	欧拉函数	134
6.5	等比数列求和	135
6.6	素数	135
6.7	排列组合其一	136
6.8	排列组合其二	137
6.9	排列组合其三	138
6.10	单调队列维护 DP	139
7	计算几何	140
7.1	绕点旋转后坐标	140
7.2	正方形已知两点求另外两点	140
7.3	三角形	140
8	字符串模型	143
8.1	SAM: LCS	143
8.2	SAM: Multy_LCS	148
8.3	SAM: 出现次数大于等于 k 次的最长子串	153
8.4	SQAM: 公共子串数量	158
8.5	SQAM: 公共子序列数量	160
9	网络流建图	163
9.1	最小路径覆盖	163
9.1.1	最小不相交路径覆盖	163
9.1.2	最小可相交路径覆盖	163
9.1.3	最多不相交路径	163
9.2	最小割	164
9.2.1	最大权闭合子图	164
9.3	最小割	165
9.4	最大密度子图	165
9.5	二元关系最小割模型	165
9.6	二分图带权匹配	165
9.7	最大权不相交路径	165
9.8	不等式差分模型 (网络流解线性规划)	166
9.9	有上下界的网络流	166
9.9.1	无源汇有上下界可行流	166
9.9.2	有源汇有上下界可行流	167
9.9.3	有源汇有上下界最大流和最小流	167

9.9.4 有源汇有上下界最小费用流	167
------------------------------	-----

1 STL

1.1 快读快写

```
1 inline void read(int &x){
2     int s = 0, w = 1; char ch = getchar();
3     while(ch < '0' || ch > '9'){ if(ch == '-') w = -1; ch =
        getchar(); }
4     while(ch >= '0' && ch <= '9') s = s * 10 + ch - '0', ch
        = getchar();
5     x = s*w;
6     return ;
7 }
8 inline void write(int x){
9     if(x<0){
10         putchar('-');
11         x=-x;
12     }
13     if(x>9)write(x/10);
14     putchar(x%10+'0');
15 }
```

```
1 unordered_map<int,int>mp;
2 unordered_set<int>s;
3 multiset<int>S;
4
5 int x=floor(0.1);//xia
6 int y=ceil(1.0);//shang
```

1.2 自定义开 02 开栈

```
1 #pragma GCC optimize(2)
2 #pragma GCC optimize(3,"Ofast","inline")
3
4 #pragma comment(linker, "/STACK:1024000000,1024000000")
5
6 //自定义开栈
7 int size(512<<20); // 512M
8 __asm__ ( "movq %0, %%rsp\n"::"r"((char*)malloc(size)+
    size)); // YOUR CODE
```

1.3 set

```
1 unordered_set<int>s;  
2 multiset<int>S;  
3 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  
4 fl=upper_bound(a,a+13,x)-a;//在从小到大的序列中找出大于x的地址  
5  
6 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  
7 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  
8 s.erase(s.find(tmp));
```

1.4 string

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

1.5 bitset

```
1 foo.size() 返回大小（位数）  
2 foo.count() 返回1的个数  
3 foo.any()  返回是否有1  
4 foo.none() 返回是否没有1  
5 foo.set()  全都变成1  
6 foo.set(p) 将第p + 1位变成1  
7 foo.set(p, x) 将第p + 1位变成x  
8 foo.reset() 全都变成0  
9 foo.reset(p) 将第p + 1位变成0  
10 foo.flip() 全都取反  
11 foo.flip(p) 将第p + 1位取反
```


- 12 `foo.to_ulong()` 返回它转换为`unsigned long`的结果，如果超出范围则报错
- 13 `foo.to_ullong()` 返回它转换为`unsigned long long`的结果，如果超出范围则报错
- 14 `foo.to_string()` 返回它转换为`string`的结果

1.6 python math 库

```

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

```

1.7 rand()、测速、小数

```

1 srand(time(0));
2 clock_t start, finish;start = clock();
3 finish = clock();
4 solve();
5 cout << "the_time_cost_is" <<double(finish - start)/
    CLOCKS_PER_SEC;
6 // cout << fixed << setprecision(2) <<ans<<"\n";

```

1.8 vector

```

1 int sum=accumulate(v.begin(),v.end(),0); // 求和
2 //去重

```

```
3 vector<int> vector_unique(vector<int>v ){
4     sort(v.begin(),v.end());
5     auto v_it = unique(v.begin(),v.end());
6     if(v_it != v.end())
7         v.erase(v_it,v.end());
8     return v;
9 }
10 //两个vector求并集
11 vector<int> vector_set_union(vector<int>v1 ,vector<int>v2)
12 {
13     vector<int>v;
14     sort(v1.begin(),v1.end());
15     sort(v2.begin(),v2.end());
16     set_union(v1.begin(),v1.end(),v2.begin(),v2.end(),
17         back_inserter(v));
18     return v;
19 }
20 //两个vector求交集
21 vector<int> vector_set_intersection(vector<int>v1 ,vector<
22 int>v2){
23     vector<int>v;
24     sort(v1.begin(),v1.end());
25     sort(v2.begin(),v2.end());
26     set_intersection(v1.begin(),v1.end(),v2.begin(),v2.end
27         (),back_inserter(v));
28     return v;
29 }
30 //判断vector的某一元素是否存在
31 bool is_element_in_vector(vector<int>v,int element){
32     // vector<int>::iterator it;
33     auto it=find(v.begin(),v.end(),element);
34     if (it!=v.end()){
35         return true;
36     }
37     else{
38         return false;
39     }
40 }
```

```
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();
63     //取下标
64     int MI = *min_element(v.begin(),v.end());//取值
65     int MI_fk = min_element(v.begin(),v.end())-v.begin();
66     //取下标
66 }
```

1.9 struct

```
1 struct node
2 {
3     int sum,ls,rs,ts;
4     node():sum(0),ls(0),rs(0),ts(0){}
5     node(int a,int b,int c,int d):sum(a),ls(b),rs(c),ts(d)
6     {}
7 };
```

1.10 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);
20
21 void solve()
22 {
23     int x;cin>>x;
24     int fl;
25     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
```

```
26     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  
27  
28     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  
29     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  
30 }  
31  
32 next_permutation(p+1,p+1+9);  
33 prev_permutation(p+1,p+1+9);
```

1.11 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-->0)
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 }
```

1.12 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-->0)
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 }
```

1.13 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     }
```

```
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 }
```

1.14 模拟退火

```
1  const int maxn=1e6+100;
2  struct node{
3      double x,y,z;
4  }a[maxn];
5  int n;
6  struct SaDown{
7      //退火系数
8      #define down 0.996
9      double ansx,ansy,ansz,answ;
10     //评估函数
11     double energy(double x,double y,double z){
12         double r=0;
13         for(int i=1;i<=n;i++){
14             r=max(r,sqrt((x-a[i].x)*(x-a[i].x)+(y-a[i].y)
15                 *(y-a[i].y)+(z-a[i].z)*(z-a[i].z)));
16         }
17         return r;
18     }
19     void sa()
20     {
21         double t=3000;
22         while(t>1e-15)
23         {
24             double ex=ansx+(rand()*2-RAND_MAX)*t;
25             double ey=ansy+(rand()*2-RAND_MAX)*t;
26             double ez=ansz+(rand()*2-RAND_MAX)*t;
27             double ew=energy(ex,ey,ez);
28             double de=ew-answ;
29             if(answ>ew)
30             {
31                 ansx=ex;
32                 ansy=ey;
33                 ansz=ez;
34                 answ=ew;
35             }
36         }
37     }
38 }
```

```
35         else if(exp(-de/t)*RAND_MAX>rand())
36         {
37             ansx=ex;
38             ansy=ey;
39             ansz=ez;
40         }
41         t*=down;
42     }
43 }
44 void doit(){
45     //退火次数
46     for(int i=1;i<=4;i++)
47         sa();
48 }
49 }sa;
50 void solve()
51 {
52     cin>>n;
53     for(int i=1;i<=n;i++){
54         scanf("%lf%lf%lf",&a[i].x,&a[i].y,&a[i].z);
55         sa.ansx+=a[i].x;
56         sa.ansy+=a[i].y;
57         sa.ansz+=a[i].z;
58     }
59     sa.ansx/=n;
60     sa.ansy/=n;
61     sa.ansz/=n;
62     sa.answ=sa.energy(sa.ansx,sa.ansy,sa.ansz);
63     sa.doit();
64     printf("%.3lf\n",sa.answ);
65 }
66 signed main(){
67     ios::sync_with_stdio(false);
68     cin.tie(nullptr);cout.tie(nullptr);
69     solve();
70     return 0;
71 }
```

1.15 约瑟夫环问题

```
1 ll n,m,k;
2 ll f(ll n,ll m,ll k)
3 {
4     ll ans=(k-1)%(n-m+1);
5     if(k==1) ans=m-1;
6     else{
7         for(ll i=n-m+2;i<=n;i++){
8             ans=(ans+k)%i;
9             ll js=i-ans-1;
10            js/=k;
11            if(n-i-1<js)js=n-i-1;
12            if(js>0){
13                i+=js;
14                ans=ans+k*js;
15            }
16        }
17    }
18    return ans+1;
19 }
20 void solve()
21 {
22     cin>>n>>m>>k;
23     cout<<f(n,m,k)<<endl;
24 }
```

1.16 PBDS

1.16.1 超快哈希 MAP

```
1 #include <ext/pb_ds/assoc_container.hpp>
2 #include <bits/stdc++.h>
3 using namespace __gnu_pbds;
4 using namespace std;
5 const int RANDOM = chrono::high_resolution_clock::now().
    time_since_epoch().count();
6 struct chash {
7     int operator()(int x) const { return x ^ RANDOM; }
8 };
9 using hash_map=gp_hash_table<string, int>;
10
11
12 hash_map a;
13 int main()
14 {
15     a["sdf"]=1;
16     a["q"]=2;
17     for(auto x:a)
18         cout<<x.first<<"_ "<<x.second<<endl;
19 }
```

1.16.2 Tree

```
1 // #include <bits/extc++.h>
2 #include <bits/stdc++.h>
3 #include <ext/pb_ds/assoc_container.hpp>
4 using namespace std;
5 using namespace __gnu_cxx;
6 using namespace __gnu_pbds;
7
8 typedef tree<int, null_type, less<int>, rb_tree_tag,
   tree_order_statistics_node_update> Tree;
9 Tree tr;
10
11
12 // 求平衡树内排名为k的值是多少
13 template<typename T>
14 T getval(int k)
15 {
16     auto it=tr.find_by_order(k-1);
17     if(it!=tr.end())
18         return *it;
19     else
20         return -1;
21 }
22
23
24 template<typename T>
25 int getRank(T x)
26 {
27     return tr.order_of_key(x)+1;
28 }
29
30
31 void test()
32 {
33     tr.insert(10);
34     tr.insert(20);
```

```
35     cout<<getval<int>(1)<<endl;
36     cout<<getval<int>(2)<<endl;
37
38
39
40     cout<<endl;
41     cout<<getRank<int>(11)<<endl;
42     cout<<getRank<int>(21)<<endl;
43 }
44
45 namespace A{
46     using pii=pair<int,int>;
47     tree<pii,null_type,less<pii>,rb_tree_tag,
        tree_order_statistics_node_update> tr;
48 // pii //存储的类型
49 // null_type //无映射(低版本g++为null_mapped_type)
50 // less<pii> //从小到大排序
51 // rb_tree_tag //红黑树
52 // tree_order_statistics_node_update //更新方式
53 // tr.insert(mp(x,y)); //插入;
54 // tr.erase(mp(x,y)); //删除;
55 // tr.order_of_key(pii(x,y)); //求排名
56 // tr.find_by_order(x); //找k小值, 返回迭代器
57 // tr.join(b); //将b并入tr, 前提是两棵树类型一样且没有重复
    元素
58 // tr.split(v,b); //分裂, key小于等于v的元素属于tr, 其余的
    属于b
59 // tr.lower_bound(x); //返回第一个大于等于x的元素的迭代器
60 // tr.upper_bound(x); //返回第一个大于x的元素的迭代器
61 // //以上所有操作的时间复杂度均为O(logn)
62 }
63 int main()
64 {
65     test();
66 }
```

1.16.3 平衡树

```
1  typedef tree<pii, null_type, less<pii>, rb_tree_tag,  
    tree_order_statistics_node_update> Tree;  
2  Tree t;  
3  
4  template<typename T>  
5  int getRank(T x)  
6  {  
7      return t.order_of_key(x)+1;  
8  }  
9  
10 template<typename T>  
11 T getVal(int k)  
12 {  
13     auto it=t.find_by_order(k-1);  
14     if(it!=t.end())  
15         return *it;  
16     else  
17         return {INF,0};  
18 }  
19  
20 void solve()  
21 {  
22     int n;  
23     cin>>n;  
24     for(int i=1;i<=n;i++)  
25     {  
26         int op,x;  
27         cin>>op>>x;  
28         if(op==1) // 插入x数  
29         {  
30             t.insert({x,i});  
31         }  
32         else if(op==2) // 删除x数(若有多个相同的数, 应只删  
            除一个)  
33         {
```



```
34         auto it=t.upper_bound({x,0});
35         t.erase(it);
36     }
37     else if(op==3) //查询 x 数的排名(排名定义为比当前
        数小的数的个数 +1
38     {
39         int k=getRank<pii>({x,0});
40         cout<<k<<endl;
41     }
42     else if(op==4) // 查询排名为 x 的数
43     {
44         auto v=getVal<pii>(x);
45         cout<<v.first<<endl;
46     }
47     else if(op==5) // 求 x 的前驱(前驱定义为小于 x，且
        最大的数)
48     {
49         auto it=t.upper_bound({x,0});
50         --it;
51         auto v=*it;
52         cout<<v.first<<endl;
53     }
54     else if(op==6) // 求 x 的后驱(前驱定义为大于 x，且
        最小的数)
55     {
56         auto v=*t.upper_bound({x,INF});
57         cout<<v.first<<endl;
58     }
59 }
60 }
61 signed main(){
62     // freopen("data.in","r",stdin);
63     // freopen("data.out","w",stdout);
64     ios::sync_with_stdio(false);
65     cin.tie(nullptr);cout.tie(nullptr);
66     solve();
67     return 0;
```

68 }

2 字符串

2.1 kmp

```
1 struct KMP{
2     int nxt[maxn];
3     int len;
4     void init(char *s)
5     {
6         nxt[1]=0;
7         len=strlen(s+1);
8         for(int i=2;i<=len;i++)
9         {
10             nxt[i]=nxt[i-1];
11             while(s[nxt[i]+1]!=s[i]&&nxt[i])
12                 nxt[i]=nxt[nxt[i]];
13             if(s[nxt[i]+1]==s[i])nxt[i]++;
14         }
15     }
16     void kmp(char *s,char *t)
17     {
18         int lens=strlen(s+1);
19         int lent=strlen(t+1);
20         int cnt=0;
21         for(int i=0,j=0;i<lens;i++)
22         {
23             while(j&&s[i+1]!=t[j+1])j=nxt[j];
24             if(s[i+1]==t[j+1])j++;
25             if(j==lent)
26             {
27                 cnt++;
28                 j=nxt[j];
29             }
30         }
31     }
32     /* 循环周期 形如 acaca 中 ac 是一个合法周期 */
33     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;
```



```
                                nxt[i]-nxt[nxt[i]]));
34         }
35     }
36     if(s[nxt[i]+1]==s[i])nxt[i]++;
37     cout<<nxt[i]<<" ";
38 }
39 cout<<endl;
40 }
41 }
42 signed main(){
43     // freopen("data.in","r",stdin);
44     // freopen("data.out","w",stdout);
45     ios::sync_with_stdio(false);
46     cin.tie(nullptr);cout.tie(nullptr);
47     solve();
48     return 0;
49 }
```

2.3 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 }

```

2.4 Manacher

```
1 struct Manacher{
2     char ch[maxn];
3     int lc[maxn];
4     int N;
5     void init(char *s){
6         int n=strlen(s+1); // puts(s+1);
7         ch[n*2+1]='#';
8         ch[0]='@';
9         ch[n*2+2]='\0';
10        for(int i=1;i<=n;i++)ch[i*2]=s[i],ch[i*2-1]='#';
11        N=n*2+1;
12    }
13    void manacher(){
14        lc[1]=1; int k=1,ma=1;
15        for (int i=2;i<=N;i++){
16            int p = k+lc[k]-1;
17            if (i<=p){lc[i]=min(lc[2*k-i],p-i+1);
18            }else{ lc[i]=1;}
19            while(ch[i+lc[i]]==ch[i-lc[i]])lc[i]++;
20            if(i+lc[i]>k+lc[k])k=i;
21            ma=max(ma,lc[i]-1);
22        }
23    }
24 }Manch;
25 char s[maxn];
26 void solve()
27 {
28     scanf("%s",s+1);
29     Manch.init(s);
30     Manch.manacher();
31 }
```


2.5 字符串哈希

```
1  /*
2  双哈希 测试
3  https://ac.nowcoder.com/acm/contest/64384/D
4  */
5  #pragma GCC optimize(2)
6  #pragma GCC optimize(3,"Ofast","inline")
7  #include<bits/stdc++.h>
8  using namespace std;
9  #define endl "\n"
10 using ull=unsigned long long;
11 using ll=long long;
12 using pii=pair<int,int>;
13 using PLL=pair<ull,ull>;
14 const int maxn=1e6+200;
15 char s[maxn];
16 int n,m,k;
17 namespace EA{
18     class Hash{
19     public:
20         ull Prime_Pool[3] = {233317ull,1998585857ul
21             ,2333333333ull};
22         ull Seed_Pool[4]={911,146527,19260817,91815541};
23         ull Mod_Pool
24             [5]={29123,998244353,1000000009,4294967291ull
25             ,21237044013013795711};
26         ull sum[maxn],bas[maxn];
27         int perm[maxn];
28         int sigma;
29         ull Seed,Mod;
30         int N;
31         Hash(int x,int y):Seed(Seed_Pool[x]),Mod(Mod_Pool[
32             y]){
33             bas[0]=1;
34             for (int i = 1; i <= N; i++){
35                 bas[i] = bas[i - 1] * Seed % Mod;
```

```

32         }
33     }
34     void init(char *s){
35         N=strlen(s+1);
36         for(int i=1;i<=N;i++)
37             sum[i]=(sum[i-1]*Seed%Mod+s[i])%Mod;
38         bas[0]=1;
39         for(int i=1;i<=N;i++)
40             bas[i]=bas[i-1]*Seed%Mod;
41     }
42     void indexInit(char *s){
43         N=strlen(s+1);
44         sigma=N+100;
45         iota(perm+1,perm+1+sigma,1);
46         random_shuffle(perm+1,perm+1+sigma);
47         // for(int i=1;i<=N;i++)
48         //     cout<<perm[i]<<" ";
49         // cout<<endl;
50         for(int i=1;i<=N;i++)
51             sum[i]=(sum[i-1]*Seed%Mod+perm[s[i]])%Mod;
52         bas[0]=1;
53         for(int i=1;i<=N;i++)
54             bas[i]=bas[i-1]*Seed%Mod;
55     }
56     ull get(int l,int r){
57         return (sum[r]-sum[l-1]*bas[r-l+1]%Mod+Mod)%
58             Mod;
59     }
60     }h[2]{Hash(2,2),Hash(3,3)};
61     void solve()
62     {
63         cin>>n>>m>>k>>s+1;
64         h[0].init(s);
65         h[1].init(s);
66         // h[0].indexInit(s);
67         // h[1].indexInit(s);
68         map<PLL,int>cnt;

```

```
68     map<PLL,int>vis;
69     for(int i=m;i<=n;i++)
70     {
71         ull now[2];
72         now[0]=h[0].get(i-m+1,i);
73         now[1]=h[1].get(i-m+1,i);
74         PLL fl={now[0],now[1]};
75         if(vis.count(fl)&&vis[fl]<i-m+1)
76         {
77             cnt[fl]++;
78             vis[fl]=i;
79         }
80         else if(!vis.count(fl))
81         {
82             cnt[fl]=1;
83             vis[fl]=i;
84         }
85     }
86     int ans=0;
87     for(auto [x,y]:cnt)
88         if(y==k)ans++;
89     cout<<ans<<endl;
90 }
91 };
92 namespace EB{
93     class StringHash{
94     public:
95         ull Mod_Pool
96             [5]={29123,998244353,1000000009,4294967291ull,
97                 2123704401301379571l};
98         ull Prime_Pool[3] = {233317ull,1998585857ul
99             ,2333333333ul};
100         ull seed[4]={911,146527,19260817,91815541};
101         ull mod=2123704401301379571l;
102         ull prime=233317;
103         ull base=131;
104         vector<ull>h;
```

```

102     vector<ull>bas;
103     StringHash():h(1),bas(1,1),base(131){}
104     StringHash(int x,int y):h(1),bas(1,1){base=seed[x
        ];mod=Mod_Pool[y];}
105     void push_back(char ch){
106         h.push_back((h.back()*base+ch)%mod + prime);
107         bas.push_back(bas.back()*base%mod);
108     }
109     // ull get(int l,int r){
110     //     return (h[r] + __int128(h[l])*(mod-bas[r-l]))%
        mod;
111     // }
112     ull get(int l,int r){
113         return (h[r]-h[l-1]*bas[r-l+1]%mod+mod)%mod;
114     }
115 };
116 void A()
117 {
118     string str;
119     StringHash hs,rhs;
120     int N=int(str.size());
121     for(int i=0;i<N;i++)
122         hs.push_back(str[i]);
123     for(int i=N-1;i>=0;i--)
124         rhs.push_back(str[i]);
125 }
126 void solve()
127 {
128     cin>>n>>m>>k>>s+1;
129     StringHash h[2]{StringHash(1,2),StringHash(2,3)};
130     for(int i=1;i<=n;i++)
131     {
132         h[0].push_back(s[i]);
133         h[1].push_back(s[i]);
134     }
135     map<PLL,int>cnt;
136     map<PLL,int>vis;

```

```
137         for(int i=m;i<=n;i++)
138         {
139             ull now[2];
140             now[0]=h[0].get(i-m+1,i);
141             now[1]=h[1].get(i-m+1,i);
142             PLL fl={now[0],now[1]};
143             if(vis.count(fl)&&vis[fl]<i-m+1)
144             {
145                 cnt[fl]++;
146                 vis[fl]=i;
147             }
148             else if(!vis.count(fl))
149             {
150                 cnt[fl]=1;
151                 vis[fl]=i;
152             }
153         }
154         int ans=0;
155         for(auto [x,y]:cnt)
156             if(y==k)ans++;
157         cout<<ans<<endl;
158     }
159 };
160 namespace shuang{
161     class DoubleHash{
162     public:
163         const ull b1=137,b2=149,i1=1'603'801'661,i2=1'024'
            053'074;
164         const ull p1=2'034'452'107,p2=2'013'074'419;
165         ull m1[maxn],m2[maxn],r1,r2;
166         PLL h[maxn];
167         int N;
168         void init(char*s){
169             h[0]={0,0};
170             int n=strlen(s+1);
171             m1[0]=m2[0]=1;
172             N=n+10;
```

```
173         for(int i=1;i<=N;i++)
174         {
175             m1[i]=m1[i-1]*b1%p1;
176             m2[i]=m2[i-1]*b2%p2;
177         }
178         h[0]={0,0};
179         for(int i=1;i<=n;i++)
180         {
181             r1=(r1+s[i]*m1[i])%p1;
182             r2=(r2+s[i]*m2[i])%p2;
183             h[i]={r1,r2};
184         }
185     }
186     PLL getv(int l,int r){
187         return {(p1+h[r].first-h[l].first)*m1[N-l]%p1
188                 ,(p2+h[r].second-h[l].second)*m2[N-l]%p2};
189     }
190     ull get(int l,int r){
191         PLL a=getv(l,r);
192         return (ull)a.first<<32|a.second;
193     }h;
194 };
195 signed main()
196 {
197     ios::sync_with_stdio(false);
198     cin.tie(nullptr);cout.tie(nullptr);
199     EA::solve();
200     return 0;
201 }
```

2.6 最小表示法

```
1 int Get_Min(int *p,int n)
2 {
3     int i=0,j=1,k=0;
4     while(i<n&& j<n&& k<n)
5     {
6         if(p[(i+k)%n]==p[(j+k)%n])k++;
7         else
8         {
9             if(p[(i+k)%n]>p[(j+k)%n])i=i+k+1;
10            else j=j+k+1;
11            if(i==j)i++;
12            k=0;
13        }
14    }
15    return min(i,j);
16 }
17 string str;
18 int p[maxn];
19 int n;
20 void solve()
21 {
22     cin>>n;
23     for(int i=0;i<n;i++)
24         cin>>p[i];
25     int tmp=Get_Min(p,n);
26     // cout<<tmp<<endl;
27     for(int i=0;i<n;i++)
28         cout<<p[(i+tmp)%n]<<" ";
29 }
```

2.7 AC 自动机

```
1 struct ACAM{
2     int ch[maxn][26],nxt[maxn];
3     int val[maxn],cnt[maxn];
4     int id[maxn];
5     int in[maxn];
6     int root=0,indx=0;
7     void clear(){
8         for(int i=0;i<=indx;i++){
9             val[i]=cnt[i]=id[i]=nxt[i]=0;
10            for(int j=0;j<26;j++)
11                ch[i][j]=0;
12        }
13        indx=0;
14    }
15    void insert(const string &str,int x)
16    {
17        int rt=0;
18        for(int i=0;i<str.size();i++){
19            int tmp=str[i]-'a';
20            if(!ch[rt][tmp])ch[rt][tmp]=++indx;
21            rt=ch[rt][tmp];
22        }
23        val[rt]++;
24        id[x]=rt;
25    }
26    void build()
27    {
28        queue<int>q;
29        for(int i=0;i<26;i++)
30            if(ch[0][i])
31                q.push(ch[0][i]);
32        while(!q.empty()){
33            int x=q.front();q.pop();
34            for(int i=0;i<26;i++){
35                int &rt=ch[x][i];
```



```
36         if(!rt)
37             rt=ch[nxt[x]][i];
38         else{
39             nxt[rt]=ch[nxt[x]][i];
40             in[ch[nxt[x]][i]]++;
41             q.push(rt);
42         }
43     }
44 }
45 }
46 void query(const string&s,int n)
47 {
48     int rt=0;
49     for(int i=0;i<s.size();i++)
50     {
51         int tmp=s[i]-'a';
52         rt=ch[rt][tmp];
53         cnt[rt]++;
54     }
55     queue<int>q;
56     for(int i=1;i<=indx;i++)
57         if(!in[i])q.push(i);
58     while(!q.empty())
59     {
60         int x=q.front();q.pop();
61         int y=nxt[x];
62         cnt[y]+=cnt[x];
63         in[y]--;
64         if(!in[y])q.push(y);
65     }
66     for(int i=1;i<=n;i++)
67         cout<<cnt[id[i]]<<endl;
68 }
69 }ac;
70 int n;
71 void solve()
72 {
```

```
73     cin>>n;
74     for(int i=1;i<=n;i++){
75         string str;cin>>str;
76         ac.insert(str,i);
77     }
78     string t;
79     cin>>t;
80     ac.build();
81     ac.query(t,n);
82 }
```

2.8 SA 后缀数组

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define maxn 1000050
4  int x[maxn],y[maxn],c[maxn],sa[maxn],rk[maxn],height[maxn
    ],wt[30];
5  char s[maxn];
6  int mm[maxn];
7  int best[maxn][20];
8  int RMQ[maxn];
9  int n,m;
10 void putout(int x)
11 {
12     if(!x) {putchar(48);return;}
13     int l=0;
14     while(x) wt[++l]=x%10,x/=10;
15     while(l) putchar(wt[l--]+48);
16 }
17 void get_SA()
18 {
19     int i,j,k;
20     for (i=1;i<=n;++i) ++c[x[i]=s[i]];
21     //c数组是桶
22     //x[i]是第i个元素的第一关键字
23     for (i=2;i<=m;++i) c[i]+=c[i-1];
24     //做c的前缀和，我们就可以得出每个关键字最多是在第几名
25     for (i=n;i>=1;--i) sa[c[x[i]]--]=i;
26     for (k=1;k<=n;k<=1)//log(n)轮
27     {
28         int num=0;
29         for (i=n-k+1;i<=n;++i) y[++num]=i;
30         //y[i]表示第二关键字排名为i的数，第一关键字的位置
31         //第n-k+1到第n位是没有第二关键字的 所以排名在最前
            面
32         for (i=1;i<=n;++i) if (sa[i]>k) y[++num]=sa[i]-k;
33         //排名为i的数 在数组中是否在第k位以后
```

```
34 //如果满足(sa[i]>k) 那么它可以作为别人的第二关键字，就把它的第一关键字的位置添加进y就行了
35 //所以i枚举的是第二关键字的排名，第二关键字靠前的先入队
36 for (i=1;i<=m;++i) c[i]=0;
37 //初始化c桶
38 for (i=1;i<=n;++i) ++c[x[i]];
39 //因为上一次循环已经算出了这次的第一关键字 所以直接加就行了
40 for (i=2;i<=m;++i) c[i]+=c[i-1];//第一关键字排名为1~i的数有多少个
41 for (i=n;i>=1;--i) sa[c[x[y[i]]]--]=y[i],y[i]=0;
42 //因为y的顺序是按照第二关键字的顺序来排的
43 //第二关键字靠后的，在同一个第一关键字桶中排名越靠后
44 //基数排序
45 swap(x,y);
46 //这里不用想太多，因为要生成新的x时要用到旧的，就把旧的复制下来，没别的意思
47 x[sa[1]]=1;num=1;
48 for (i=2;i<=n;++i)
49     x[sa[i]]=(y[sa[i]]==y[sa[i-1]] && y[sa[i]+k]==y[sa[i-1]+k]) ? num : ++num;
50 //因为sa[i]已经排好序了，所以可以按排名枚举，生成下一次的第一个关键字
51 if (num==n) break;
52 m=num;
53 //这里就不用那个122了，因为都有新的编号了
54 }
55 for(int i=1;i<=n;i++)
56     cout<<sa[i]<<" ";cout<<endl;
57 }
58 void get_height()
59 {
60     int k=0;
61     for (int i=1;i<=n;++i) rk[sa[i]]=i;
62     for (int i=1;i<=n;++i)
```

```

63     {
64         if (rk[i]==1) continue;//第一名height为0
65         if (k) --k;//h[i]>=h[i-1]+1;
66         int j=sa[rk[i]-1];
67         while (j+k<=n && i+k<=n && s[i+k]==s[j+k]) ++k;
68         height[rk[i]]=k;//h[i]=height[rk[i]];
69     }
70     for(int i=1;i<=n;i++)RMQ[i]=height[i];
71     for(int i=1;i<=n;i++)
72         cout<<height[i]<<" ";cout<<endl;
73 }
74
75 void initST(int n){
76     mm[0]=-1;
77     for(int i=1;i<=n;i++)
78         mm[i]=((1<(i-1))&&0)?mm[i-1]+1:mm[i-1];
79     for(int i=1;i<=n;i++)best[i][0]=i;
80     for(int i=1;i<=mm[n];i++)
81     {
82         for(int j=1;j+(1<<i)-1<=n;j++){
83             int a=best[j][i-1];
84             int b=best[j+(1<<i-1)][j];
85             if(RMQ[a]<RMQ[b])best[j][i]=a;
86             else best[j][i]=b;
87         }
88     }
89 }
90 int queryST(int a,int b){
91     int t;
92     t=mm[b-a+1];
93     b--=(1<<t)-1;
94     a=best[a][t];
95     b=best[b][t];
96     return RMQ[a]<RMQ[b]?a:b;
97 }
98 int lcp(int a,int b){
99     a=rk[a];b=rk[b];

```

```
100     if(a>b)swap(a,b);
101     return height[queryST(a+1,b)];
102 }
103 void solve()
104 {
105     cin>>s+1;
106     n=strlen(s+1);m=122;
107     //n表示原字符串长度，m表示字符个数，ascii('z')=122
108     get_SA();
109     get_height();
110     initST(n);
111     cout<<lcp(1,3)<<endl;
112
113 }
114 signed main(){
115     ios::sync_with_stdio(false);
116     cin.tie(0);cout.tie(0);
117     solve();
118 }
119 /*
120 */
```

2.9 SAM 后缀数组

```
1  class SAM{
2  public:
3      vector<int>v[maxn];
4      int ch[maxn][26],siz[maxn],len[maxn],fa[maxn];
5      bitset<maxn>vis;
6      int f[maxn];
7      int tot=1,np=1;
8      ll ans;
9      void clear(){
10         for(int i=0;i<=tot;i++){
11             for(int j=0;j<26;j++)
12                 ch[i][j]=0;siz[i]=len[i]=fa[i]=vis[i]=0;
13         }tot=np=1;
14     }
15     void insert(const int&c){
16         int p=np; np=++tot;
17         len[np]=len[p]+1; siz[np]=1;
18         for(;p&&!ch[p][c];p=fa[p])ch[p][c]=np;
19         if(p==0)fa[np]=1;
20         else{
21             int q=ch[p][c];
22             if(len[q]==len[p]+1)fa[np]=q;
23             else{
24                 int nq=++tot;
25                 len[nq]=len[p]+1;
26                 fa[nq]=fa[q]; fa[q]=nq; fa[np]=nq;
27                 for(;p&&ch[p][c]==q;p=fa[p])ch[p][c]=nq;
28                 memcpy(ch[nq],ch[q],sizeof(ch[q]));
29             }
30         }
31     }
32     void insert(const char*s){for(int i=0;s[i]!='\0';i++)
33         insert(s[i]-'a');}
34     void insert(const string&s){for(int i=0;i<(int)s.size()
35         ();i++)insert(s[i]-'a');}
```

```
34     void build_tree(){
35         for(int i=2;i<=tot;i++)
36             v[fa[i]].push_back(i);
37     }
38     /* op=0本质不同子串 op=1位置不同子串 */
39     void build(int op=0){
40         build_tree();
41         if(!op)
42         {
43             for(int i=1;i<=tot;i++)
44                 f[i]=siz[i]=1;
45         }
46         else get_siz(1);
47         f[1]=siz[1]=0;
48         get_f(1);
49     }
50     void get_kth(int k,int rt=1)
51     {
52         if(k>f[rt]){
53             cout<<-1<<endl;
54             return ;
55         }
56         if(k<=siz[rt]){cout<<"\n";return ;}
57         k-=siz[rt];
58         if(k==0){return;}
59         for(int i=0;i<26;i++){
60             if(ch[rt][i]){
61                 if(f[ch[rt][i]]>=k){
62                     cout<<(char)(i+'a');
63                     get_kth(k,ch[rt][i]);
64                     return ;
65                 }
66                 else
67                     k-=f[ch[rt][i]];
68             }
69         }
70         cout<<-1<<endl;
```



```

71     }
72     void match(const string&s)
73     {
74         // queue<int>q;
75         // for(int i=1;i<=tot;i++){
76         //     f[i]=0,in[i]=main_in[i];
77         //     if(!in[i])q.push(i);
78         // }
79         // int rt=1;
80         // int length=0;
81         // for(int i=0,tmp;i<(int)s.size();i++)
82         // {
83         //     tmp=s[i]-'a';
84         //     while(rt&&!ch[rt][tmp])
85         //         rt=fa[rt],length=len[rt];
86         //     if(!rt)rt=1;
87         //     else{
88         //         rt=ch[rt][tmp];length++;
89         //     }
90         //     f[rt]=max(f[rt],length);
91         // }
92         // while(!q.empty()){
93         //     int x=q.front();q.pop();
94         //     mi[x]=min(mi[x],f[x]);
95         //     int y=fa[x];
96         //     if(f[x]>=len[y])
97         //         f[y]=len[y];
98         //     in[y]--;
99         //     if(!in[y])q.push(y);
100        // }
101    }
102    void debug(){
103        for(int i=0;i<=tot;i++)
104            for(int j=0;j<26;j++)
105                if(ch[i][j])
106                    cout<<"ins_"<<i<<"_--"<<(char)(j+'a')
                        <<"-->_"<<ch[i][j]<<endl;

```

```
107         for(int i=1;i<=tot;i++)
108             cout<<"fail_"<<i<<"_->"<<fa[i]<<endl;
109         for(int i=1;i<=tot;i++)
110             cout<<"len["<<i<<"]="<<len[i]<<endl;
111     }
112     void query();
113 private:
114     void get_siz(int x){
115         for(auto y:v[x]){
116             get_siz(y);
117             siz[x]+=siz[y];
118         }
119         f[x]=siz[x];
120     }
121     void get_f(int x){
122         if(vis[x])return ;
123         vis[x]=1;
124         for(int i=0;i<26;i++){
125             int y=ch[x][i];
126             if(!y)continue;
127             get_f(y);
128             f[x]+=f[y];
129         }
130     }
131     int LCS(string&str)
132     {
133         int rt=1;
134         int ans=0;
135         int length=0;
136         for(int i=0;i<str.size();i++)
137         {
138             int tmp=str[i]-'a';
139             while(rt&&!ch[rt][tmp])
140                 rt=fa[rt],length=len[rt];
141             if(!rt)rt=1;
142             else
143                 rt=ch[rt][tmp],length++;

```

```
144         ans=max(ans,length);
145     }
146     return ans;
147 }
148 }sam;
149 char ch[maxn];
150 int op,k;
151 void solve()
152 {
153     cin>>ch;
154     sam.insert(ch);
155     // sam.debug();
156     sam.build(0);
157     int q;cin>>q;
158     while(q--){
159         cin>>k;
160         sam.get_kth(k);
161     }
162 }
```

2.10 PAM 回文自动机

```
1  class PAM{
2  public:
3      int s[maxn],now;
4      int ch[maxn][26],fa[maxn],len[maxn],last,tot;
5      int num[maxn];
6      int pre=0;
7      void clear()
8      {
9          s[0]=len[1]=-1;
10         fa[0]=tot=now=1;
11         last=len[0]=0;
12         memset(ch[0],0,sizeof(ch[0]));
13         memset(ch[1],0,sizeof(ch[1]));
14     }
15     PAM(){clear();}
16     int newnode(int length){
17         tot++;
18         memset(ch[tot],0,sizeof(ch[tot]));
19         fa[tot]=num[tot]=0;
20         len[tot]=length;
21         return tot;
22     }
23     int get_fail(int rt){
24         while(s[now-len[rt]-2]!=s[now-1])rt=fa[rt];
25         return rt;
26     }
27     void insert(int c)
28     {
29         s[now++]=c;
30         int rt=get_fail(last);
31         if(!ch[rt][c]){
32             int np=newnode(len[rt]+2);
33             fa[np]=ch[get_fail(fa[rt])][c];
34             ch[rt][c]=np;
35             // num[np]=num[fa[np]]+1;
```

```
36     }
37     last=ch[rt][c];
38     num[last]++;
39 }
40 void build(){
41     for(int i=tot;i>=2;i--)
42         num[fa[i]]+=num[i];
43     num[0]=num[1]=0;
44 }
45 void insert(char*s){while(*s){insert(*s-'a');s++;}}
46 void insert(const string&s){for(auto i:s)insert(i-'a')
47     ;}
48 void debug(){
49     for(int i=0;i<=tot;i++)
50         for(int j=0;j<26;j++)
51             if(ch[i][j])
52                 cout<<"ins_"<<i<<"_--"<<(char)(j+'a')
53                     <<"-->_"<<ch[i][j]<<endl;
54     for(int i=0;i<=tot;i++)
55         cout<<"len["<<i<<"]="<<len[i]<<endl;
56     for(int i=0;i<=tot;i++)
57         cout<<"fa["<<i<<"]="<<fa[i]<<endl;
58 }
59 }pam;
60 string s;
61 void solve()
62 {
63     cin>>s;
64     pam.insert(s);
65 }
```

2.11 序列自动机

```
1  class SQAM{
2  public:
3      int ch[maxn][26];
4      string s;
5      int tot;
6      void init(){init(s);}
7      void init(string&s)
8      {
9          int n=s.size();
10         // for(int i=0;i<26;i++)
11         //   ch[n][i]=INF;
12         for(int i=n-1;i>=0;i--){
13             for(int j=0;j<26;j++){
14                 ch[i][j]=ch[i+1][j];
15                 ch[i][s[i]-'a']=i+1;
16             }
17         }
18         void debug(){
19             for(int i=0;i<=tot;i++){
20                 for(int j=0;j<26;j++){
21                     if(ch[i][j]){
22                         // cout<<i<<" --"<<(char)(j+'a')<<"-->
23                         //   "<<ch[i][j]<<endl;
24                         cout<<i<<"_ "<<ch[i][j]<<"_ "<<(char)(j+
25                         'a')<<endl;
26                     }
27                 }
28             }
29         }
30     };
31 }
```

3 搜索

3.1 精确 DLX

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define endl "\n"
4  #define pp(x) array<int,x>
5  using ull=unsigned long long;
6  using ll=long long;
7  using pii=pair<int,int>;
8  const int dx[]={0,0,1,-1,1,-1,1,-1};
9  const int dy[]={1,-1,0,0,1,-1,-1,1};
10 const int mod=998244353;
11 const int inf=0x3f3f3f3f;
12 const int INF=1e9+7;
13 const int maxn=1e6+100;
14 const int MaxM=450010;
15 const int MaxP=510;
16 struct DLX{
17     int U[MaxM],D[MaxM],L[MaxM],R[MaxM];
18     int Row[MaxM],Col[MaxM];
19     int First[MaxP];
20     int Size[MaxM];
21     int tot=0;
22     int ans;
23     void build(int n,int num){
24         for(int i=0;i<=num;i++){
25             L[i]=i-1;R[i]=i+1;
26             U[i]=i;D[i]=i;
27             Size[i]=0;
28         }
29         L[0]=num;
30         R[num]=0;
31         tot=num;
32         for(int i=1;i<=n;i++)
33             First[i]=-1;
```

```
34     }
35     void insert(int r,int c){
36         Row[++tot]=r; Col[tot]=c;
37         D[tot]=D[c]; U[D[c]]=tot;
38         U[tot]=c; D[c]=tot;
39         if(First[r]<0){
40             L[tot]=tot;
41             R[tot]=tot;
42             First[r]=tot;
43         }
44         else{
45             R[tot]=R[First[r]];
46             L[R[First[r]]]=tot;
47             L[tot]=First[r];
48             R[First[r]]=tot;
49         }
50         Size[c]++;
51     }
52     void remove(int c){
53         L[R[c]]=L[c];
54         R[L[c]]=R[c];
55         for(int i=D[c];i!=c;i=D[i]){
56             for(int j=R[i];j!=i;j=R[j]){
57                 U[D[j]]=U[j];
58                 D[U[j]]=D[j];
59                 Size[Col[j]]--;
60             }
61         }
62     }
63     void resume(int c){
64         for(int i=U[c];i!=c;i=U[i]){
65             for(int j=L[i];j!=i;j=L[j]){
66                 D[U[j]]=j;
67                 U[D[j]]=j;
68                 Size[Col[j]]++;
69             }
70         }
```



```
71         R[L[c]]=c;
72         L[R[c]]=c;
73     }
74     void dance(int depth){
75         if(depth>=ans)
76             return ;
77         if(R[0]==0){
78             ans=depth;
79             return ;
80         }
81         int c=R[0];
82         for(int i=R[0];i!=0;i=R[i])
83             if(Size[i]<Size[c])
84                 c=i;
85         remove(c);
86         for(int i=D[c];i!=c;i=D[i]){
87             for(int j=R[i];j!=i;j=R[j])
88                 remove(Col[j]);
89             dance(depth+1);
90             for(int j=L[i];j!=i;j=L[j])
91                 resume(Col[j]);
92         }
93         resume(c);
94     }
95 }dlx;
96 int n,m,p;
97 void solve()
98 {
99     cin>>n>>m>>p;
100     dlx.build(p,n*m);
101     for(int i=1;i<=p;i++)
102     {
103         int x,y,_x,_y;
104         cin>>x>>y>>_x>>_y;
105         for(int I=x+1;I<=_x;I++)
106             for(int J=y+1;J<=_y;J++)
107                 dlx.insert(i,(I-1)*m+J);
```

```
108     }
109     dlx.ans=inf;
110     dlx.dance(0);
111     if(dlx.ans==inf)cout<<-1<<endl;
112     else cout<<dlx.ans<<endl;
113 }
114 signed main(){
115     // freopen("C:\\Users\\tob\\Desktop\\P1000_0.in.txt","r",
116     //         stdin);
117     // freopen("C:\\Users\\tob\\Desktop\\P1000_0.out.txt","w
118     //         ",stdout);
119     ios::sync_with_stdio(false);
120     cin.tie(nullptr);cout.tie(nullptr);
121     int __;cin>>__;
122     while(__--){
123         solve();
124     }
125     // fclose(stdin);
126     // fclose(stdout);
127     return 0;
128 }
129 /*
130 (x,y,w)
131 x w y w
132 */
```

4 图论

4.1 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)
```

4.2 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 }
```

4.3 2-SET

```
1 void solve()
2 {
3     cin>>n>>m;
4     /*
5     如果指定 x 一定为真
6     add(x,x+n);
7     如果x为假那么x一定为真
8     */
9     for(int i=1;i<=m;i++)
10    {
11        int x,a,y,b;
12        cin>>x>>a>>y>>b;
13        if(a==1&&b==1)//x为真或者y为真
14        {
15            v[x].push_back(y+n);//x为假那么y一定为真
16            v[y].push_back(x+n);//y为假那么x一定为真
17        }
18        else if(a==0&&b==1)//x为假或者y为真
19        {
20            v[x+n].push_back(y+n);//x为真那么y一定为真
21            v[y].push_back(x);//y为假那么x一定为假
22        }
23        else if(a==1&&b==0)//x为真或者y为假
24        {
25            v[x].push_back(y);//x为假那么y一定为假
26            v[y+n].push_back(x+n);//y为真那么x一定为真
27        }
28        else if(a==0&&b==0)//x为假或者y为假
29        {
30            v[x+n].push_back(y);
31            v[y+n].push_back(x);
32        }
33    }
34    for(int i=1;i<=n*2;i++)
35        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     }
```

4.4 倍增优化建图

```
1 int find(int x,int y){
2     if(id[x][y])return id[x][y];
3     id[x][y]=++Flow::tot;
4     if(!y){
5         Flow::Add(id[x][y],Flow::t,val[x]);
6     }
7     else {
8         Flow::Add(id[x][y],find(x,y-1),INF);
9         Flow::Add(id[x][y],find(f[x][y-1],y-1),INF);
10    }
11    return id[x][y];
12 }
13
14 // 与倍增求lca相同的循环
15 int lca=LCA(x,y);
16 for(int i=14;i>=0;i--)
17     if((dep[x]-dep[lca])>>i&1)
18     {
19         Flow::Add(now,find(x,i),INF);
20         x=f[x][i];
21     }
22 for(int i=14;i>=0;i--)
23     if((dep[y]-dep[lca])>>i&1)
24     {
25         Flow::Add(now,find(y,i),INF);
26         y=f[y][i];
27     }
```

4.5 prim

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define endl "\n"
4  using ull=unsigned long long;
5  using ll=long long;
6  using pii=pair<int,int>;
7  const int dx[]={0,0,1,-1,1,-1,1,-1};
8  const int dy[]={1,-1,0,0,1,-1,-1,1};
9  const ull str_mod=21237044013013795711;
10 const ull more=19260817;
11 const ull prime=233317;
12 const ull base=131;
13 const int inf=0x3f3f3f3f;
14 const int INF=1e9+7;
15 const int maxn=1e6;
16 struct node
17 {
18     int t;int d;
19     bool operator < (const node &a) const
20     {
21         return d>a.d;
22     }
23 };
24 struct Edge{
25     int to,next,w;
26 }edge[maxn];
27 int head[maxn],cnt;
28 bitset<maxn>vis;
29 int n,m;
30 void add(int from,int to,int w){
31     edge[++cnt].w=w;
32     edge[cnt].to=to;
33     edge[cnt].next=head[from];
34     head[from]=cnt;
35 }
```

```
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 }
```

4.6 匈牙利

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  using ll=long long;
4  using pii=pair<int,int>;
5  const int maxn=1e6;
6  vector<int>v[maxn];
7  int mt[maxn];
8  int n,m,e;
9  bitset<maxn>vis;
10 bool dfs(int x)
11 {
12     if(vis[x])return 0;
13     vis[x]=1;
14     for(auto y:v[x]){
15         if(!mt[y]||dfs(mt[y])){
16             mt[y]=x;
17             return 1;
18         }
19     }
20     return 0;
21 }
22 int main()
23 {
24     cin>>n>>m>>e;
25     for(int i=1;i<=e;i++){
26         int x,y;
27         cin>>x>>y;
28         v[x].push_back(y);
29         // v[y].push_back(x);
30     }
31     int ans=0;
32     for(int i=1;i<=n;i++){
33         vis.reset();
34         if(dfs(i))ans++;
35     }
```

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

4.7 KM

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define int long long
4  using ll=long long;
5  using pii=pair<int,int>;
6  const int INF=1e9+7;
7  const int mod=998244353;
8  const int N=303;
9  int w[N][N];
10 int n,m;
11 bitset<N>vx,vy;
12 namespace KM {
13     long long cal(int n, int m) {
14         std::vector<long long> u(n + 1), v(m + 1), p(m +
            1), way(m + 1);
15         for (int i = 1; i <= n; i++) {
16             p[0] = i;
17             long long j0 = 0;
18             std::vector<long long> minv(m + 1, 1e18);
19             std::vector<char> used(m + 1, false);
20             do {
21                 used[j0] = true;
22                 long long i0 = p[j0], delta = 1e18, j1;
23                 for (int j = 1; j <= m; ++j) {
24                     if (!used[j]) {
25                         long long cur = w[i0][j] - u[i0] -
                            v[j];
26                         if (cur < minv[j]) {
27                             minv[j] = cur, way[j] = j0;
28                         }
29                         if (minv[j] < delta) {
30                             delta = minv[j], j1 = j;
31                         }
32                     }
33                 }
            }
        }
    }
}

```

```
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 }
```

4.8 点双

```
1  struct Edge{
2      int to,next,w;
3  }edge[maxn];
4  int head[maxn],cnt;
5  int dfn[maxn],low[maxn],indx;
6  vector<int>bcc[maxn];
7  int col[maxn],tot;
8  bitset<maxn>vis;
9  stack<int>s;
10 int bri[maxn];
11 int cut[maxn];
12 int n,m,bcc_cnt;
13 void add(int from,int to)
14 {
15     edge[++cnt].to=to;
16     edge[cnt].next=head[from];
17     head[from]=cnt;
18 }
19 void tarjan(int x,int fa)
20 {
21     dfn[x]=low[x]=++indx;
22     vis[x]=1;
23     s.push(x);
24     int num=0;
25     for(int i=head[x];i;i=edge[i].next)
26     {
27         int y=edge[i].to;
28         if(!dfn[y])
29         {
30             num++;
31             tarjan(y,i);
32             low[x]=min(low[x],low[y]);
33             if(dfn[x]<low[y])
34                 bri[i]=bri[i^1]=1;
35             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 }
```


4.9 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 }
```

4.10 仙人掌图求环路径长度

```
1  const int mod=998244353;
2  const int inf=0x3f3f3f3f;
3  const int INF=1e14+7;
4  const int maxn=4e6+100;
5  struct Edge{
6      int to,w,next;
7  }edge[maxn];
8  int head[maxn],cnt;
9  int dfn[maxn],low[maxn],indx;
10 stack<pii>s;
11 int n,m,ans;
12 void init()
13 {
14     while(!s.empty())s.pop();
15     for(int i=1;i<=n;i++)
16     {
17         dfn[i]=low[i]=head[i]=0;
18     }
19     cnt=1;
20     indx=0;
21     ans=INF;
22 }
23 void add(int from,int to,int w)
24 {
25     edge[++cnt].w=w;
26     edge[cnt].to=to;
27     edge[cnt].next=head[from];
28     head[from]=cnt;
29 }
30 void tarjan(int x,int fa)
31 {
32     dfn[x]=low[x]=++indx;
33     for(int i=head[x];i;i=edge[i].next)
34     {
35         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                     ;
55                 else ans=min(ans,min(g[0]+g[1],g[2]));
56             }
57         }
58         else if(dfn[x]>dfn[y]&&!(fa^1))
59         {
60             low[x]=min(low[x],dfn[y]);
61             s.push({x,edge[i].w});
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 }
```

4.11 DINIC(超级无敌快!)

```
1 namespace Flow{
2     struct Edge{
3         int to,next,w;
4     }edge[maxn];
5     int head[maxn],cnt=1;
6     int dep[maxn];
7     int now[maxn];
8     int s,t,tot;
9     void init(){
10         memset(head+1,0,sizeof(int)*tot);
11         cnt=1;
12         tot=0;
13     }
14     void add(int from,int to,int w){
15         edge[++cnt].w=w;
16         edge[cnt].to=to;
17         edge[cnt].next=head[from];
18         head[from]=cnt;
19     }
20     void Add(int u,int v,int w){
21         add(u,v,w);
22         add(v,u,0);
23     }
24     bool bfs(int s,int t){
25         for(int i=1;i<=tot;i++)
26             dep[i]=INF;
27         queue<int>q;
28         q.push(s);
29         dep[s]=0;
30         now[s]=head[s];
31         while(!q.empty()){
32             int x=q.front();q.pop();
33             for(int i=head[x];i;i=edge[i].next){
34                 int y=edge[i].to;
35                 if(edge[i].w>0&&dep[y]==INF){
```

```
36             q.push(y);
37             now[y]=head[y];
38             dep[y]=dep[x]+1;
39             if(y==t)return 1;
40         }
41     }
42 }
43 return 0;
44 }
45 int dfs(int x,int flow,int t){
46     if(x==t)return flow;
47     int ans=0,k;
48     for(int i=now[x];i&&flow;i=edge[i].next){
49         now[x]=i;
50         int y=edge[i].to;
51         if(edge[i].w>0&&dep[y]==dep[x]+1){
52             k=dfs(y,min(flow,edge[i].w),t);
53             if(!k)dep[y]=inf;
54             edge[i].w-=k;
55             edge[i^1].w+=k;
56             ans+=k;
57             flow-=k;
58         }
59     }
60     return ans;
61 }
62 int DINIC(){
63     int ans=0;
64     while(bfs(s,t))
65         ans+=dfs(s,INF,t);
66     return ans;
67 }
68 }
```

4.12 MCMF

```
1  struct node{
2      int to,next,w,c;
3  }edge[maxn];
4  int head[maxn],cnt;
5  int dis[maxn],flow[maxn],last[maxn];
6  bitset<maxn>vis;
7  int n,m,s,t;
8  void init(){
9      memset(head+1,0,sizeof(int)*n);
10     cnt=1;
11 }
12 void add(int from,int to,int w,int c)
13 {
14     edge[++cnt].w=w;
15     edge[cnt].c=c;
16     edge[cnt].to=to;
17     edge[cnt].next=head[from];
18     head[from]=cnt;
19 }
20 bool spfa(int s,int t)
21 {
22     memset(dis+1,INF,sizeof(int)*n);
23     memset(last+1,-1,sizeof(int)*n);
24     vis.reset();
25     queue<int>q;
26     q.push(s);
27     dis[s]=0;
28     vis[s]=1;
29     flow[s]=INF;
30     while(!q.empty())
31     {
32         int x=q.front();q.pop();
33         vis[x]=0;
34         for(int i=head[x];i;i=edge[i].next)
35         {
```

```
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 }
```


4.13 同余最短路

同余最短路常常用于解决这样一类问题：

给出 n 个整数，每个整数可以取任意多次，询问关于它们能拼凑出的数的一些信息

例如，可能询问这些整数可以拼凑出多少 $[l, r]$ 范围内的数。

当然很容易想到完全背包之类的东西，它的时间复杂度是 $O(nV)$ (V 表示询问的值域)。。同余最短路为解决这类问题提供了一种复杂度与询问无关的方法。

我们从同余的角度考虑问题。模 A_1 的每个同余类 $[x]$ ，一旦我们能用 A_2, \dots, A_n 凑出 $[x]$ 中的某个数，那么 $[x]$ 中的每个大于等于 x 的数都可以凑出，因为可以不断地加 A_1 。例如，如果 $A = (5, 7, 11)$ ，因为 7 和 11 可以凑出模 $[3]$ 中的 18，所以 $[3]$ 中大于等于 18 的数 $(18, 23, 28, \dots)$ 都可以用 A_1 凑出来，因为可以不断的加 A_1 。

这启示我们，只要知道每个同余类中能被凑出来的最小的数，就可以解决问题。设 m 同余类 $[x]$ 中能凑出来的最小的数位 d ，则在 $[0, r]$ 中能凑出的 $[x]$ 中的数的个数位 $\lfloor \frac{r-d}{m} \rfloor + 1$ ($r \geq d$)。

这可以被转化为一个图论问题。我们对于模 A_1 的每个同余类建出一个节点，向它加 A_i ($i = 2, 3, \dots, n$) 后的对应节点连一条边，边权为 A_i 。那么每个同余类中能被凑出的最小数就是从 $[0]$ 到它的最短路的长度。

如：给定 x, y, z ，对于 $1 \leq k \leq h$ ，有多少个 k 能够满足 $ax + by + cz = k$

```

1 void solve()
2 {
3     cin>>h>>x>>y>>z;
4     if(x==1 || y==1 || z==1){cout<<h;return ;}
5     if(x>y)swap(x,y);
6     if(x>z)swap(x,z);
7     for(int i=0;i<x;i++)
8     {
9         v[i].push_back({(i+y)%x,y});
10        v[i].push_back({(i+z)%x,z});
11    }
12    dij();
13    ll ans=0;

```

```

14     for(int i=0;i<x;i++)
15         if(dis[i]<=h)
16             ans+=(h-dis[i])/x+1;
17     cout<<ans<<endl;
18 }

```

题目描述:

墨墨突然对等式很感兴趣,他正在研究 $(\sum_{i=1}^n a_i x_i) = b$ 存在非负整数解的条件,他要求你编写一个程序,给定 $n, a_1 \dots n, l, r$, 求出有多少 $b \in [l, r]$ 可以使等式存在非负整数解。

输入格式:

第一行三个整数 n, l, r 。

第二行 n 个整数 $a_1 \dots n$ 。

输出格式:

一行一个整数,表示有多少 $b \in [l, r]$ 可以使等式存在非负整数解。

对于 100% 的数据, $n \leq 12, 0 \leq a_i \leq 5 \times 10^5, 1 \leq l \leq r \leq 10^{12}$ 。

```

1 void solve()
2 {
3     cin>>n>>l>>r;l--;
4     for(int i=1;i<=n;i++)
5         cin>>p[i];
6     sort(p+1,p+1+n);
7     for(int i=0;i<p[1];i++)
8         for(int j=2;j<=n;j++)
9             add(i,(i+p[j])%p[1],p[j]);
10    dij(0);
11    ll ans=0;
12    for(int i=0;i<p[1];i++)
13    {
14        ll R=(r-dis[i])/p[1]+1;
15        ll L=(l-dis[i])/p[1]+1;
16        if(r<dis[i])R=0;
17        if(l<dis[i])L=0;
18        ans+=R-L;
19    }
20    cout<<ans<<endl;
21 }
22 /*

```

```
23 in:
24 2 5 10
25 3 5
26 out:
27 5
28 */
```

4.14 朱流算法 (Edmonds)

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define endl "\n"
4  const int INF=2e9+7;
5  const int maxn=1e6+100;
6  struct node{
7      int x,y,w;
8  }e[maxn];
9  int vis[maxn]; //标记是否走过
10 int pre[maxn]; //前驱节点
11 int inv[maxn]; //节点权值
12 int id[maxn]; //新的标号
13 int n,m,root;
14 int Edmonds()
15 {
16     int ans=0;
17     while(1)
18     {
19         /*****1.找最短弧集合*****/
20         //初始化
21         for(int i=1;i<=n;i++)
22             inv[i]=INF;
23         for(int i=1;i<=m;i++)
24         {
25             int x=e[i].x;
26             int y=e[i].y;
27             //找到每个点的最小入边 以及前驱
28             if(x!=y&&e[i].w<inv[y])
29                 inv[y]=e[i].w,pre[y]=x;
30         }
31         //如果一个点不是根 并且没有入边则不是树形图
32         for(int i=1;i<=n;i++)
33             if(i!=root&&inv[i]==INF)
34                 return -1;
35         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 }
```

4.15 最小生成树 Boruvka

```
1  #define mp make_pair
2  const int N=200005;
3  const int M=500005;
4  const LL inf=1e12;
5  int f[N],pd,n,m;
6  struct node{ int a,b; LL c; }p[M];
7  pair<LL,LL> E[N];
8  int find(int r){
9      return f[r]=(r==f[r])?r:find(f[r]);
10 }
11 LL Boruvka(){
12     LL res=0; pd=1; int num=0;
13     for(int i=1;i<=n;i++)
14         f[i]=i;
15     while(num<n-1)
16     {
17         int tmp=0;
18         for(int i=1;i<=n;i++)
19         {
20             int fa=find(i);
21             E[fa]={inf,inf};
22         }
23         for(int i=1;i<=m;i++)
24         {
25             int fa=find(p[i].a);
26             int fb=find(p[i].b);
27             if (fa==fb) continue;
28             tmp++;
29             E[fa]=min(E[fa],{p[i].c,i*1ll});
30             E[fb]=min(E[fb],{p[i].c,i*1ll});
31         }
32         if (tmp==0) break;
33         for(int i=1;i<=m;i++)
34         {
35             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 }
```


4.16 MST

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4
5  using ll=long long;
6  const ll INF64=1e18+7;
7  const int maxn=1e6+10;
8  const int INF=1e9+7;
9  const int N=1001;
10
11
12 class MstPrim{
13 public:
14     struct node{
15         int id;int dis;
16         bool operator<(const node&a)const{
17             return a.dis<dis;
18         }
19     };
20     int maxD[N][N],mp[N][N];
21     bool mst[N][N];
22     int dis[N];
23     bool vis[N];
24     int pre[N];
25     int n;
26     MstPrim(){}
27     void init(int _n){
28         n=_n;
29         for(int i=1;i<=n;i++){
30             for(int j=1;j<=n;j++){
31                 mp[i][j]=INF,mst[i][j]=maxD[i][j]=0;
32                 vis[i]=0;
33                 dis[i]=INF;
34                 pre[i]=1;
35             }
```

```
36     }
37     int prim(int s=1)
38     {
39         priority_queue<node>q;
40         q.push({s,dis[s]=0});
41         int ans=0;
42         while(!q.empty())
43         {
44             auto [x,W]=q.top();q.pop();
45             if(vis[x])continue;
46             vis[x]=1;
47             ans+=W;
48             mst[x][pre[x]]=mst[pre[x]][x]=1;
49             for(int y=1;y<=n;y++){
50                 if(vis[y]&&mp[x][y]<INF)
51                     maxD[x][y]=maxD[y][x]=max(maxD[pre[x]
52                                     ][y],dis[x]);
53                 if(mp[x][y]<dis[y]){
54                     dis[y]=mp[x][y];
55                     pre[y]=x;
56                     q.push({y,dis[y]});
57                 }
58             }
59             return ans;
60         }
61         void add(int x,int y,int w){mp[x][y]=w;}
62         bool had(int x,int y){return mst[x][y];}
63         int getmaxD(int x,int y){return maxD[x][y];}
64     }_mst;
65
66
67     class DSU{
68     public:
69         vector<int>f,num;
70         DSU(){}
71         void init(int n){
```

```
72         f.resize(n+1);
73         iota(f.begin(),f.end(),0);
74     }
75     int find(int r){
76         return f[r]=f[r]==r?f[r]:find(f[r]);
77     }
78     bool same(int x,int y){
79         return find(x)==find(y);
80     }
81     void merge(int x,int y){
82         int fa=find(x);
83         int fb=find(y);
84         if(fa!=fb)
85             f[fa]=fb;
86     }
87 }dsu;
88
89
90 class MstKru{
91 public:
92     struct node{int x,y,w,fl;}p[maxn];
93     bool mst[maxn][maxn];
94     int maxD[maxn][maxn];
95     vector<int>v[maxn];
96     int n,m;
97     MstKru(){}
98     void init(int _n){
99         n=_n;m=0;
100         for(int i=1;i<=n;i++){
101             for(int j=1;j<=n;j++){
102                 mst[i][j]=0;
103                 maxD[i][j]=0;
104             }
105             v[i]={i};
106         }
107     }
108     void add(int x,int y,int w){
```

```
109         p[++m]={x,y,w};
110     }
111     int run()
112     {
113         sort(p+1,p+1+m,[&](node a,node b){return a.w<b.w
            ;});
114         dsu.init(n);
115         ll ans=0;
116         int cnt=n;
117         for(int i=1;i<=m;i++)
118         {
119             auto &[x,y,w,fl]=p[i];
120             if(!dsu.same(x,y))
121             {
122                 cnt--;
123                 int fa=dsu.find(x);
124                 int fb=dsu.find(y);
125                 ans+=w; fl=1;
126                 dsu.merge(x,y);
127                 for(auto _i:v[fa])
128                     for(auto _j:v[fb])
129                         maxD[_i][_j]=maxD[_j][_i]=w;
130                 for(auto _i:v[fb])
131                     v[fa].push_back(_i);
132             }
133             if(cnt==1)break;
134         }
135         if(cnt!=1)return -2;
136         ll res=INF;
137         for(int i=1;i<=m;i++)
138         {
139             if(p[i].fl)continue;
140             auto [x,y,w,fl]=p[i];
141             res=min(res,ans+w-maxD[x][y]);
142         }
143         if(res==INF)return -1;
144         return res;
```

```
145     }
146 }mst;
147
148
149
150
151 class STR_Tr{
152 public:
153     struct Edge{int to,next,w;}edge[maxn];
154     int head[maxn],cnt;
155     int f[maxn][22];
156     int dep[maxn];
157     int maxx[maxn][22];
158     int minn[maxn][22];
159     void add(int from,int to,int w){
160         edge[++cnt].w=w;
161         edge[cnt].to=to;
162         edge[cnt].next=head[from];
163         head[from]=cnt;
164     }
165     void Add(int x,int y,int w){
166         add(x,y,w);
167         add(y,x,w);
168     }
169     void dfs(int x,int fa)
170     {
171         dep[x]=dep[fa]+1;
172         f[x][0]=fa;
173         minn[x][0]=-INF;
174         for(int i=1;(1<i)<=dep[x];i++)
175         {
176             f[x][i]=f[f[x][i-1]][i-1];
177             int kk[4]={maxx[x][i-1],maxx[f[x][i-1]][i-1],
178                 minn[x][i-1],minn[f[x][i-1]][i-1]};
179             sort(kk,kk+4);
180             maxx[x][i]=kk[3];
181             int ptr=2;
```

```

182         while(ptr>=0&&kk[ptr]==kk[3])ptr--;
183         minn[x][i]=(ptr==-1?-INF:kk[ptr]);
184     }
185     for(int i=head[x];i;i=edge[i].next)
186         if(edge[i].to!=fa){
187             maxx[edge[i].to][0]=edge[i].w;
188             dfs(edge[i].to,x);
189         }
190 }
191 int lca(int x,int y){
192     if(dep[x]<dep[y])swap(x,y);
193     for (int j = 0, D = dep[x] - dep[y]; D; ++j, D >>=
194         1)
195         if (D & 1) x = f[x][j];
196     if(x==y)return x;
197     for (int j = 20; ~j; --j)
198         if (f[x][j] != f[y][j])
199             x = f[x][j], y = f[y][j];
200     return f[x][0];
201 }
202 int querySTR(int x,int y,int val){
203     int res=-INF;
204     for(int i=21;i>=0;i--){
205         if(dep[f[x][i]]>=dep[y]){
206             if(val!=maxx[x][i])
207                 res=max(res,maxx[x][i]);
208             else
209                 res=max(res,minn[x][i]);
210             x=f[x][i];
211         }
212     }
213     return res;
214 }
215 int query(int x,int y)
216 {
217     int res=-INF;
218     for(int i=21;i>=0;i--){

```

```

218         if(dep[f[x][i]]>=dep[y]){
219             res=max(res,maxx[x][i]);
220             x=f[x][i];
221         }
222     }
223     return res;
224 }
225 }tr;
226
227
228
229 class Tr{
230 public:
231     struct Edge{int to,next,w;}edge[maxn];
232     int head[maxn],cnt;
233     int maxx[maxn][22];
234     int f[maxn][22];
235     int dep[maxn];
236     void add(int from,int to,int w){
237         edge[++cnt].w=w;
238         edge[cnt].to=to;
239         edge[cnt].next=head[from];
240         head[from]=cnt;
241     }
242     void Add(int x,int y,int w){
243         add(x,y,w);
244         add(y,x,w);
245     }
246     void dfs(int x,int fa)
247     {
248         dep[x]=dep[fa]+1;
249         f[x][0]=fa;
250         for(int i=1;(1<i)<=dep[x];i++)
251         {
252             f[x][i]=f[f[x][i-1]][i-1];
253             maxx[x][i]=max(maxx[x][i-1],maxx[f[x][i-1]][i
-1]);

```

```

254     }
255     for(int i=head[x];i;i=edge[i].next)
256         if(edge[i].to!=fa){
257             maxx[edge[i].to][0]=edge[i].w;
258             dfs(edge[i].to,x);
259         }
260 }
261 int lca(int x,int y){
262     if(dep[x]<dep[y])swap(x,y);
263     for (int j = 0, D = dep[x] - dep[y]; D; ++j, D >>=
        1)
264         if (D & 1) x = f[x][j];
265     if(x==y)return x;
266     for (int j = 20; ~j; --j)
267         if (f[x][j] != f[y][j])
268             x = f[x][j], y = f[y][j];
269     return f[x][0];
270 }
271 int query(int x,int y)
272 {
273     int res=-INF;
274     for(int i=21;i>=0;i--){
275         if(dep[f[x][i]]>=dep[y]){
276             res=max(res,maxx[x][i]);
277             x=f[x][i];
278         }
279     }
280     return res;
281 }
282 }_tr;
283
284
285 class Kru{
286 public:
287     struct node{int x,y,w,fl;};
288     vector<node>p;
289     int n,m;

```



```
290     ll ans;
291     Kru(){}
292     void init(int _n){
293         n=_n;ans=0;m=0;
294         p.clear();
295         p.resize(m+1);
296         dsu.init(n);
297     }
298     void add(int x,int y,int w){
299         // p[++m]={x,y,w};
300         p.push_back({x,y,w});
301     }
302     int run(){
303         sort(p.begin()+1,p.end(),[](node a,node b){return
304             a.w<b.w;});
305         int num=n;
306         for(int i=1;i<=m;i++)
307         {
308             auto &[x,y,w,fl]=p[i];
309             if(!dsu.same(x,y)){
310                 num--;
311                 dsu.merge(x,y);
312                 tr.Add(x,y,w);
313                 ans+=w;
314                 fl=1;
315             }
316             if(num==1)break;
317         }
318         if(num!=1)ans=-1;
319         return ans;
320     }
321     ll query()
322     {
323         tr.dfs(1,0);
324         ll res=INF64;
325         for(int i=1;i<=m;i++)
326         {
```

```
326         if(p[i].fl)continue;
327         auto [x,y,w,fl]=p[i];
328         int _lca=tr.lca(x,y);
329         int tmpa=tr.query(x,_lca);
330         int tmpb=tr.query(y,_lca);
331         if(max(tmpa,tmpb)>-INF)
332             res=min(res,ans-max(tmpa,tmpb)+w);
333     }
334     return (res==INF64?-1:res);
335 }
336 }_k;
337
338
339
340
341
342 int main(){
343     ios::sync_with_stdio(0);
344     cin.tie(0);cout.tie(0);
345     // int __;cin>>__;
346     // for(int i=1;i<=__;i++){
347     //     cout<<"Case #"<<i<<": ";
348     //     solve();
349     // }
350     return 0;
351 }
```

4.17 基尔霍夫矩阵

```

1  /*
2  基尔霍夫局矩阵树
3  用于解决：
4      给定n个点m条边的无向图，求图的生成树个数
5  基本定义：
6  1.无向图G:给定n个点，m条边的无向图，设点集为V，边集为E，记
    为G(V,E)
7  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
8  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
9  4.基尔霍夫矩阵:K[G]:K[G]=D[G]-A[G],K[i][j]=D[i][j]-A[i][j]
10 定理：
11      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
12 其绝对值为生成树个数
13
14 一般来说求(1,1)~(n-1,n-1)这个行列式
15 */
16 ll det(int n)
17 {
18     ll res=1;
19     for(int i=2;i<=n;i++){
20         for(int j=i+1;j<=n;j++){
21             while(a[j][i]){
22                 ll t=a[i][i]/a[j][i];
23                 for(int k=i;k<=n;k++)
24                     a[i][k]=(a[i][k]-a[j][k]*t);
25                 for(int k=i;k<=n;k++)
26                     swap(a[i][k],a[j][k]);
27                 res=-res;
28             }
29         }
30         if(a[i][i]==0)res=-res;
31         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-->0)
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 }
```

4.18 全源最短路

```
1 struct node{
2     int x;ll dis;
3     bool operator<(const node&a)const{
4         return a.dis<dis;
5     }
6 };
7 struct Edge{int to,next;ll w;}edge[maxn];
8 int head[maxn],cnt;
9 ll dis[maxn];
10 ll h[maxn];
11 int n,m;
12 void add(int from,int to,ll w){
13     edge[++cnt].w=w;
14     edge[cnt].to=to;
15     edge[cnt].next=head[from];
16     head[from]=cnt;
17 }
18 bool spfa(int s)
19 {
20     for(int i=0;i<=n;i++)h[i]=INF;
21     vector<int>in(n+10,0);
22     bitset<maxn>vis;
23     queue<int>q;
24     vis[s]=1;
25     h[s]=0;
26     q.push(s);
27     while(!q.empty()){
28         int x=q.front();q.pop();
29         vis[x]=0;
30         in[x]++;
31         if(in[x]>n)return 0;
32         for(int i=head[x];i;i=edge[i].next){
33             int y=edge[i].to;
34             if(h[y]>h[x]+edge[i].w){
35                 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 }
```

4.19 全局最小割

```

1  int G[MAX_N][MAX_N];
2  int v[MAX_N]; //v[i]代表节点i合并到的顶点
3  int w[MAX_N]; //定义 $w(A,x) = \sum w(v[i],x)$ ,  $v[i] \in A$ 
4  bool visited[MAX_N]; //用来标记是否该点加入了A集合
5  int squ[MAX_N]; //记录移除的节点次序
6  int index; //记录最小割的位置, 以便分开整个图的节点
7
8  int stoer_wagner(int n)
9  {
10     int min_cut = INF, r=0;
11     for (int i = 0; i < n; ++i){
12         v[i] = i; //初始还未合并, 都代表节点本身
13     }
14     while (n > 1){
15         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)
16         memset(visited, 0, sizeof(visited));
17         memset(w, 0, sizeof(w));
18         for (int i = 1; i < n; ++i){
19             //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
20             int k = -1;
21             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
22                 if (!visited[v[j]]){
23                     w[v[j]] += G[v[pre]][v[j]];
24                     if (k == -1 || w[v[k]] < w[v[j]])
25                         k = j;
26                 }
27             }
28             visited[v[k]] = true; //标记该点x已经加入A集合
29             if (i == n - 1) //若|A|=|V| (所有点都加入了A), 结束
30                 {
31                     const int s = v[pre], t = v[k];

```



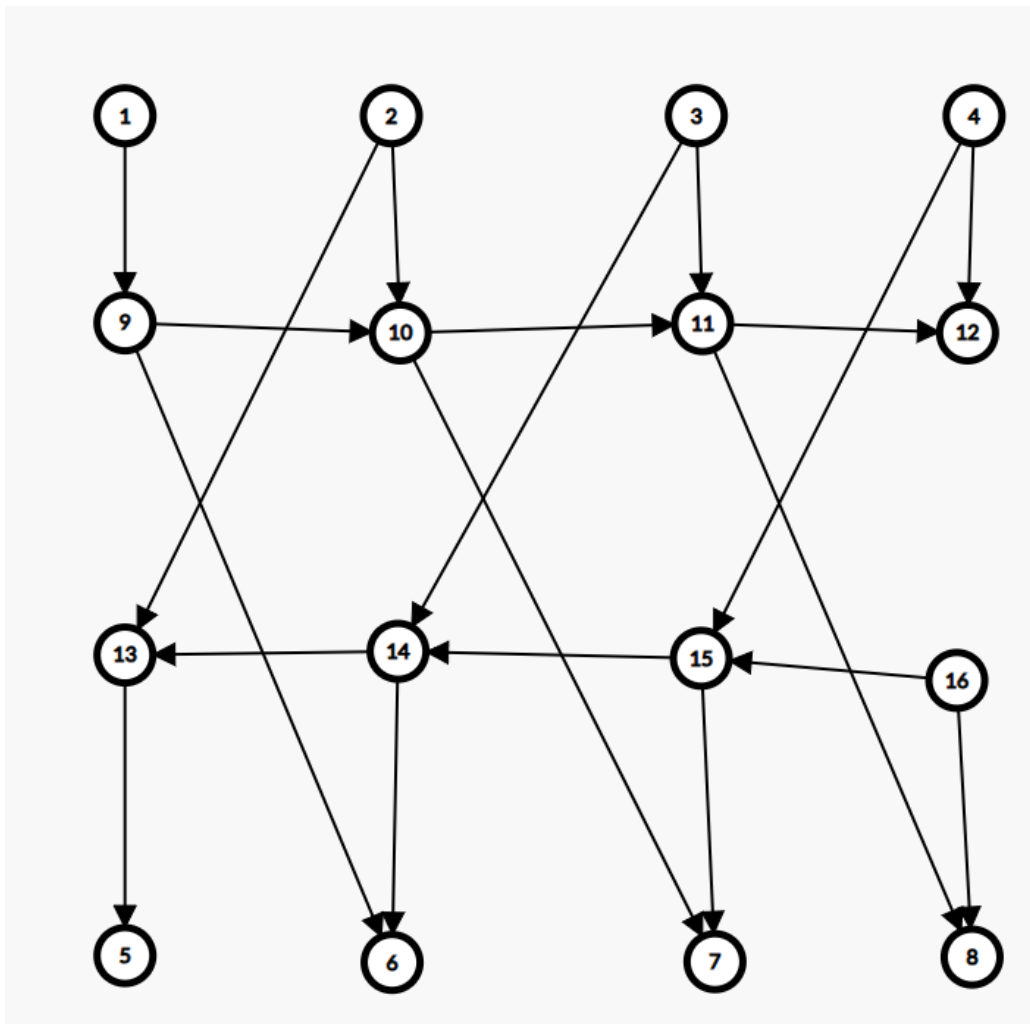
```

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 }
```

4.20 前缀和优化建图



```

1 vector<int>v[maxn];
2 vector<int>p;
3 int n,m,k;
4 void f()
5 {
6     int t=p.size();
7     for(int i=0;i<t;i++)
8     {
9         int x=p[i];
10        v[x].push_back(x+n*2);
11        v[x+n*3].push_back(x+n);
12        if(i<t-1)
13        {
14            int y=p[i+1];

```

```
15         v[x+n*2].push_back(y+n*2);
16         v[y+n*3].push_back(x+n*3);
17         v[x+n*2].push_back(y+n);
18         v[y].push_back(x+n*3);
19     }
20 }
21 }
```

5 数据结构

5.1 MultiplyLCA

```
1 struct Multiply_LCA{
2     vector<int>v[maxn];
3     int dep[maxn];
4     int f[maxn][25];
5
6     void dfs(int x,int fa)
7     {
8         dep[x]=dep[fa]+1;
9         f[x][0]=fa;
10        for(int i=1;i<=20;i++)
11            f[x][i]=f[f[x][i-1]][i-1];
12        for(auto y:v[x])
13            if(y!=fa)
14                RMQ(y,x);
15    }
16    int LCA(int x,int y)
17    {
18        if(dep[x]<dep[y])swap(x,y);
19        for(int i=20;i>=0;i--)
20            if(dep[f[x][i]]>=dep[y])
21                x=f[x][i];
22        if(x==y)return x;
23        for(int i=20;i>=0;i--)
24            if(dep[f[x][i]]!=dep[f[y][i]])
25                x=f[x][i],y=f[y][i];
26        return f[x][0];
27    }
28 };
```

5.2 一维 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,
11                //另一段为f[i+(1<<(j-1))][j-1]
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;
```

5.3 二维 ST

```

1  struct St{
2  int f[309][309][9][9];
3  void init(int m,int n){
4      for(int i=1;i<=m;i++)
5          for(int j=1;j<=n;j++)
6              f[i][j][0][0]=mp[i][j];
7      int t=log(n)/log(2)+1;//j定位
8      for(int i=0;i<=t;i++){for(int j=0;j<=t;j++){
9          if(i==0&&j==0)continue;
10         for(int row=1;row+(1<<i)-1<=m;row++){
11             for(int col=1;col+(1<<j)-1<=n;col++){
12                 if(i)f[row][col][i][j]=min(f[row][col
13                     ][i-1][j],
14                     f[row+(1<<(i-1))][col][i-1][j]);
15                 else f[row][col][i][j]=min(f[row][col
16                     ][i][j-1],
17                     f[row][col+(1<<(j-1))][i][j-1]);
18             }
19         }
20     }
21 }
22 int query(int x1,int y1,int x2,int y2){
23     int k1 = log(double(x2 - x1 + 1)) / log(2.0);
24     int k2 = log(double(y2 - y1 + 1)) / log(2.0);
25     int m1 = f[x1][y1][k1][k2];
26     int m2 = f[x2 - (1<<k1) + 1][y1][k1][k2];
27     int m3 = f[x1][y2 - (1<<k2) + 1][k1][k2];
28     int m4 = f[x2 - (1<<k1) + 1][y2 - (1<<k2) + 1 ][k1][k2
29         ];
30     int _max = min(min(m1,m2),min(m3,m4));
31     return _max;
32 }
33 }sT;

```

5.4 划分树

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define endl "\n"
4  using ll=long long;
5  const int maxn=100010;
6  int tree[20][maxn]; //表示每层每个位置的值
7  int sorted[maxn]; //已经排序好的数
8  int toleft[20][maxn]; //toleft[p][i]表示第i层从1到i有数分入
    左边
9  int n,m;
10
11 void build(int l,int r,int dep){
12     if(l==r)return ;
13     int mid=(l+r)>>1;
14     int same=mid-l+1;
15     for(int i=l;i<=r;i++)
16         if(tree[dep][i]<sorted[mid])
17             same--;
18     int lpos=l;
19     int rpos=mid+1;
20     for(int i=l;i<=r;i++){
21         if(tree[dep][i]<sorted[mid])
22             tree[dep+1][lpos++]=tree[dep][i];
23         else if(tree[dep][i]==sorted[mid]&&same>0){
24             tree[dep+1][lpos++]=tree[dep][i];
25             same--;
26         }
27         else
28             tree[dep+1][rpos++]=tree[dep][i];
29         toleft[dep][i]=toleft[dep][l-1]+lpos-l;
30     }
31     build(l,mid,dep+1);
32     build(mid+1,r,dep+1);
33 }
34 int query(int L,int R,int l,int r,int dep,int k){
```



```
35     if(l==r)return tree[dep][l];
36     int mid=(L+R)>>1;
37     int cnt=toleft[dep][r]-toleft[dep][l-1];
38     if(cnt>=k){
39         int newl=L+toleft[dep][l-1]-toleft[dep][L-1];
40         int newr=newl+cnt-1;
41         return query(L,mid,newl,newr,dep+1,k);
42     }
43     else{
44         int newr=r+toleft[dep][R]-toleft[dep][r];
45         int newl=newr-(r-l-cnt);
46         return query(mid+1,R,newl,newr,dep+1,k-cnt);
47     }
48 }
49 int query(int l,int r,int k){
50     return query(1,n,l,r,0,k);
51 }
52 void solve()
53 {
54
55     cin>>n>>m;
56     memset(tree,0,sizeof(tree));
57     for(int i=1;i<=n;i++){
58         cin>>tree[0][i];
59         sorted[i]=tree[0][i];
60     }
61     sort(sorted+1,sorted+n+1);
62     build(1,n,0);
63     int s,t,k;
64     while(m--){
65         {
66             cin>>s>>t>>k;
67             cout<<query(s,t,k)<<endl;
68         }
69     }
```

5.5 GrayCode

```
1 // n位格雷码
2 vector<int> grayCode(int n) {
3     int count = 1 << n;
4     vector<int> res(count,0);
5     for(int i = 1 ; i < count; i ++ )
6     {
7         int bin = i, cur = bin >> (n - 1);
8         for(int k = n - 1; k > 0; k --)
9             cur = (cur << 1) + (((bin >> k) & 1) ^ ((bin
10                 >>(k - 1)) & 1));
11         res[i] = cur;
12     }
13     return res;
14 }
15 //递归实现n位格雷码
16 vector<string> gray_code(int n){
17     if(n==1)return {"0","1"};
18     else{
19         vector<string>v,v1;
20         v1=gray_code(n-1);
21         for(int i=0;i<v1.size();i++)
22             v.push_back("0"+v1[i]);
23         for(int i=(v1.size()-1);i>-1;i--)
24             v.push_back("1"+v1[i]);
25         return v;
26     }
27 }
28 void solve()
29 {
30     int n;
31     cin>>n;
32     vector<string>v=gray_code(n);
33     for(int i=0;i<v.size();i++)
34         cout<<v[i]<<endl;
35 }
```

5.6 归并排序

```
1 void doit()
2 {
3     int n;
4     vector<int>p(n+1,0);
5     auto mg_sort = [&](auto mg_sort,int l,int r)->ll{
6         if(l>=r)return 0;
7         int mid=(l+r)>>1;
8         ll t = mg_sort(mg_sort,l,mid)+mg_sort(mg_sort,mid
9             +1,r);
10        vector<int>temp;
11        int i=l,j=mid+1;
12        while(i<=mid&&j<=r)
13        {
14            if(p[i]<=p[j])temp.push_back(p[i++]);
15            else t+=mid-i+1,temp.push_back(p[j++]);
16        }
17        while(i<=mid)temp.push_back(p[i++]);
18        while(j<=r)temp.push_back(p[j++]);
19        for(int i=l,j=0;i<=r;i++,j++)
20            p[i]=temp[j];
21        return t;
22    };
23    mg_sort(mg_sort,1,n);
24 }
```

5.7 树链剖分

```

1  struct Seg{int l,r,val,lazy;}t[maxn<<2];
2  struct Edge{int to,next,w;}edge[maxn];
3  int head[maxn],cnt;
4  int siz[maxn],f[maxn],dep[maxn],son[maxn];
5  int id[maxn],di[maxn],top[maxn],tot;
6  int w[maxn];
7  int n,root,q,P;
8  inline void add(int from,int to,int w=0){
9      edge[++cnt]={to,head[from],w};
10     head[from]=cnt;
11 }
12 inline void pushup(int rt){
13     t[rt].val=t[rt<<1].val+t[rt<<1|1].val;
14 }
15 inline void pushdown(int rt){
16     if(t[rt].lazy){
17         (t[rt<<1].val+=(t[rt].lazy*(t[rt<<1].r-t[rt<<1].l
18             +1))%P)%=P;
19         (t[rt<<1|1].val+=(t[rt].lazy*(t[rt<<1|1].r-t[rt
20             <<1|1].l+1))%P)%=P;
21         (t[rt<<1].lazy+=t[rt].lazy)%=P;
22         (t[rt<<1|1].lazy+=t[rt].lazy)%=P;
23         t[rt].lazy=0;
24     }
25 }
26 inline void build(int rt,int l,int r)
27 {
28     t[rt]={l,r,0,0};
29     if(l==r){
30         (t[rt].val=w[di[l]])%=P;
31         return ;
32     }
33     int mid=(l+r)>>1;
34     build(rt<<1,l,mid);
35     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&&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&&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);
```

```
71 }
72 inline int qRange(int x,int y)
73 {
74     int ans=0;
75     while(top[x]!=top[y]){
76         if(dep[top[x]]<dep[top[y]])swap(x,y);
77         (ans+=query(1,id[top[x]],id[x]))%=P;
78         x=f[top[x]];
79     }
80     if(dep[x]>dep[y])swap(x,y);
81     (ans+=query(1,id[x],id[y]))%=P;
82     return ans;
83 }
84 inline int qSon(int x){
85     return query(1,id[x],id[x]+siz[x]-1);
86 }
87 inline int LCA(int x,int y)
88 {
89     while(top[x]!=top[y]){
90         if(dep[top[x]]<dep[top[y]])swap(x,y);
91         x=f[top[x]];
92     }
93     if(dep[x]>dep[y])swap(x,y);
94     return x;
95 }
96 inline void dfs1(int x)
97 {
98     siz[x]=1;
99     for(int i=head[x];i;i=edge[i].next){
100         int y=edge[i].to;
101         if(y==f[x])continue;
102         f[y]=x;dep[y]=dep[x]+1;
103         dfs1(y);
104         siz[x]+=siz[y];
105         if(siz[y]>siz[son[x]])
106             son[x]=y;
107     }
```

```
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 }
```


5.8 扫描线线段树

```
1 struct Seg{
2     int l,r,len,cv,_l,_r,cnt;
3 }t[maxn];
4 pp(4) p[maxn];
5 int w[maxn];
6 int n,m,cnt;
7 void build(int rt,int l,int r)
8 {
9     t[rt]={l,r,0,0,0,0,0};
10    if(l==r)return ;
11    int mid=(l+r)>>1;
12    build(rt<<1,l,mid);
13    build(rt<<1|1,mid+1,r);
14 }
15 void pushup(int rt)
16 {
17     if(t[rt].cv)
18     {
19         t[rt]._l=t[rt]._r=1;
20         t[rt].len=w[t[rt].r+1]-w[t[rt].l];
21         t[rt].cnt=1;
22     }
23     else if(t[rt].l==t[rt].r)
24     {
25         t[rt]._l=t[rt]._r=0;
26         t[rt].len=0;
27         t[rt].cnt=0;
28     }
29     else
30     {
31         t[rt]._l=t[rt<<1]._l;
32         t[rt]._r=t[rt<<1|1]._r;
33         t[rt].len=t[rt<<1].len+t[rt<<1|1].len;
34         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 }
```

5.9 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]
33         ]==0))fl_1=1;
34     if((cnt_a[x]+fl>0&&cnt_b[x]>0)|| (cnt_a[x]+fl==0&&cnt_b
35         [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 }
```

5.10 点分治 1

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define endl "\n"
4  const int INF=10000005;
5  const int maxn=1e7+10;
6  const int N=2e4+10;
7  struct Edge{int to,next,w;}edge[N];
8  int head[N],_cnt;
9  int del[N],siz[N],maxson,root,sum;
10 int dis[N],d[N],cnt;
11 int ans[N],q[maxn],judge[maxn],ask[maxn];
12 int n,m;
13 void add(int from,int to,int w){
14     edge[++_cnt].w=w;
15     edge[_cnt].to=to;
16     edge[_cnt].next=head[from];
17     head[from]=_cnt;
18 }
19 void getroot(int x,int fa)
20 {
21     siz[x]=1;
22     int sx=0;
23     for(int i=head[x];i;i=edge[i].next)
24     {
25         int y=edge[i].to;
26         if(y==fa||del[y])continue;
27         getroot(y,x);
28         siz[x]+=siz[y];
29         sx=max(sx,siz[y]);
30     }
31     sx=max(sx,sum-siz[x]);
32     if(sx<maxson)maxson=sx,root=x;
33 }
34 void getdis(int x,int fa)
35 {
```

```
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 }
```

5.11 点分治 2

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define endl "\n"
4  const int INF=10000005;
5  const int maxn=1e7+10;
6  const int N=2e4+10;
7  struct Edge{int to,next,w;}edge[N];
8  int head[N],_cnt;
9  int del[N],siz[N],maxson,root,sum;
10 int dis[N],d[N],cnt;
11 int ans[N],q[maxn],judge[maxn],ask[maxn];
12 int n,m;
13 void add(int from,int to,int w){
14     edge[++_cnt].w=w;
15     edge[_cnt].to=to;
16     edge[_cnt].next=head[from];
17     head[from]=_cnt;
18 }
19 void getroot(int x,int fa)
20 {
21     siz[x]=1;
22     int sx=0;
23     for(int i=head[x];i;i=edge[i].next)
24     {
25         int y=edge[i].to;
26         if(y==fa||del[y])continue;
27         getroot(y,x);
28         siz[x]+=siz[y];
29         sx=max(sx,siz[y]);
30     }
31     sx=max(sx,sum-siz[x]);
32     if(sx<maxson)maxson=sx,root=x;
33 }
34 void getdis(int x,int fa)
35 {
```

```
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 }
```

6 数论

6.1 ExGcd

```
1 int exgcd(int a,int b,int& x,int& y)
2 {
3     if(b==0)return x=1,y=0,a;
4     int d=exgcd(b,a%b,y,x);
5     y-=x*(a/b);
6     return d;
7 }
```

6.2 快速幂

```
1 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;}
2
3 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;}
4
5 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;}
```

6.3 整除分块

```
1 ll division_block(ll n){
2     ll res = 0;
3     for(ll l = 1, r; l <= n; l = r + 1){
4         r = n / (n / l);
5         // cout<<r<<endl;
6         res += n / l * (r - l + 1);
7     }
8     return res;
9 }
```

6.4 欧拉函数

```
1  int prime[maxn],phi[maxn];
2  bool vis[maxn];
3  int n;
4  void doit()
5  {
6      for(int i=2;i<=maxn;i++){
7          if(!vis[i])prime[++prime[0]]=i,phi[i]=i-1;
8          for(int j=1;j<=prime[0]&&prime[j]*i<=maxn;j++){
9              vis[i*prime[j]]=1;
10             if(i%prime[j]==0){
11                 phi[i*prime[j]]=phi[i]*prime[j];
12                 break;
13             }
14             phi[i*prime[j]]=phi[i]*(prime[j]-1);
15         }
16     }
17 }
18 int PHI(int x)
19 {
20     //0(sqrt(x)
21     int ans=x;
22     for(int i=2;i*i<=x;i++)
23         if(x%i==0)
24         {
25             while(x%i==0)x/=i;
26             ans=ans*(i-1)/i;
27         }
28     if(x>1)ans=ans*(x-1)/x;
29     return ans;
30 }
```

6.5 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 //  $q^0 + q^1 + q^2 + \dots + q^{n-1}$ 
9 void solve()
10 {
11     int q,n,p;
12     cin >> q >> n >> p;
13     cout << sumq(q,n,p) % p << endl;
14 }
```

6.6 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 }
```

6.7 排列组合其一

```
1 int gcd(int a,int b){
2     if(b==0)return a;
3     else return gcd(b,a%b);
4 }
5 //解线性同余方程，扩展欧几里德定理
6 int x,y;
7 void Extended_gcd(int a,int b){
8     if(b==0){x=1;y=0;}
9     else{Extended_gcd(b,a%b);long t=x;x=y;y=t-(a/b)*y;}
10 }
11 //计算不大的C(n,m)
12 int C(int a,int b){
13     if(b>a)return 0;b=(ff[a-b]%mod*ff[b])%mod;a=ff[a];
14     int c=gcd(a,b);a/=c;b/=c;Extended_gcd(b,M);x=(x+mod)%
        mod;
15     x=(x*a)%mod;return x;
16 }
17 //Lucas定理
18 int Combination(int n, int m)
19 {
20     int ans=1;int a,b;
21     while(m||n){a=n%mod;b=m%mod;n/=mod;m/=mod;
22         ans=(ans*C(a,b))%mod;}
23     return ans;
24 }
25 int A(int n,int m){return (Combination(n,m)*ff[m])%mod;}
26
27 signed main()
28 {
29     int i,m,n;
30     ff[0]=1;
31     for(int i=1; i<=M; i++) //预计算n!
32         ff[i]=(ff[i-1]*i)%mod;
33     while(~scanf("%lld%lld",&n, &m))
34     {
```



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

6.8 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)
27         % mod;
28 }
```

6.9 排列组合其三

```
1  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;}
2
3  int C(int n,int m){
4      return F[n]*INF[n-m]*INF[m];
5  }
6  F[0]=INF[0]=1;
7  for(int i=1;i<M;i++)
8  {
9      F[i]=F[i-1]*i%mod;
10     INF[i]=ksm(F[i],mod-2);
11 }
```

6.10 单调队列维护 DP

```
1  int pre[N];
2  int dp[N];
3  int q[N];
4  int n,m;
5  void solve()
6  {
7      cin>>n>>m;
8      for(int i=1;i<=n;i++)
9      {
10         int v,w,s;
11         cin>>v>>w>>s;
12         // 复制上一次的i-1 的DP
13         memcpy(pre,dp,sizeof(dp));
14         // 枚举余数为J
15         for(int j=0;j<v;j++)
16         {
17             int head=0;
18             int tail=-1;
19             for(int k=j;k<=m;k+=v)
20             {
21                 //k-q[head]>s*v 长度长了, 最多s个
22                 if(head<=tail&&q[head]<k-s*v)
23                     head++;
24                 while(head<=tail&&pre[q[tail]]-(q[tail]-j)
25                     /v*w<=pre[k]-(k-j)/v*w)
26                     tail--;
27                 if(head<=tail)
28                     dp[k]=max(dp[k],pre[q[head]]+(k-q[head]
29                         )/v*w);
30                 q[++tail]=k;
31             }
32         }
33     }
```

7 计算几何

7.1 绕点旋转后坐标

```

1 pii f(double x1,double y1,double x0,double y0,double d)
2 {
3     d=d/180*acos(-1);//如果是360一周
4     double x=(x1-x0)*cos(d)-(y1-y0)*sin(d)+x0;
5     double y=(x1-x0)*sin(d)+(y1-y0)*cos(d)+y0;
6     return {x,y};
7 }

```

7.2 正方形已知两点求另外两点

```

1 void solve()
2 {
3     cin>>a>>b>>c>>d;
4     double x=(a-b+c+d)*0.5;
5     double y=(a+b-c+d)*0.5;
6     if(ceil(x) == floor(x)&&ceil(y) == floor(y))cout<<(int
7         )x<<"_"<<(int)y<<endl;
8     else cout<<"No_Answer!"<<endl;
9 }

```

7.3 三角形

```

1 class triangle{
2     double S(pii x,pii y,pii z)
3     {
4         double s=0.5*abs((y.first-x.first)*(z.second-x.
5             second)-(z.first-x.first)*(y.second-x.second));
6         return s;
7     }
8     double S(double x,double y,double z)
9     {

```

```
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         else if (x1 == x2 && x2 == x3)
54             return 0;
55         else
56         {
57             if (y1 == y2 && y2 == y3)
58                 return 0;
59             else
60                 return 1;
61         }
62     }
63 }
64 };
```

8 字符串模型

8.1 SAM: LCS

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define endl "\n"
4  #define pp(x) array<int,x>
5  using ull=unsigned long long;
6  using ll=long long;
7  using pii=pair<int,int>;
8  using pdd=pair<double,double>;
9  const int dx[]={0,0,1,-1,1,-1,1,-1};
10 const int dy[]={1,-1,0,0,1,-1,-1,1};
11 const int mod=998244353;
12 const int inf=0x3f3f3f3f;
13 const int INF=1e9+7;
14 const int maxn=1e6+100;
15 class SAM{
16 public:
17     vector<int>v[maxn];
18     int ch[maxn][26],siz[maxn],len[maxn],fa[maxn];
19     bitset<maxn>vis;
20     int f[maxn];
21     int tot=1,np=1;
22     ll ans;
23     void clear(){
24         for(int i=0;i<=tot;i++){
25             for(int j=0;j<26;j++)
26                 ch[i][j]=0;siz[i]=len[i]=fa[i]=vis[i]=0;
27         }tot=np=1;
28     }
29     void insert(const int&c){
30         int p=np; np=++tot;
31         len[np]=len[p]+1; siz[np]=1;
32         for(;p&&!ch[p][c];p=fa[p])ch[p][c]=np;
33         if(p==0)fa[np]=1;
```

```

34         else{
35             int q=ch[p][c];
36             if(len[q]==len[p]+1)fa[np]=q;
37             else{
38                 int nq=++tot;
39                 len[nq]=len[p]+1;
40                 fa[nq]=fa[q]; fa[q]=nq; fa[np]=nq;
41                 for(;p&&ch[p][c]==q;p=fa[p])ch[p][c]=nq;
42                 memcpy(ch[nq],ch[q],sizeof(ch[q]));
43             }
44         }
45     }
46     void insert(const char*s){for(int i=0;s[i]!='\0';i++)
47         insert(s[i]-'a');}
47     void insert(const string&s){for(int i=0;i<(int)s.size
48         ();i++)insert(s[i]-'a');}
48     void build_tree(){
49         for(int i=2;i<=tot;i++)
50             v[fa[i]].push_back(i);
51     }
52     /* op=0本质不同子串 op=1位置不同子串 */
53     void build(int op=0){
54         build_tree();
55         if(!op)
56         {
57             for(int i=1;i<=tot;i++)
58                 f[i]=siz[i]=1;
59         }
60         else get_siz(1);
61         f[1]=siz[1]=0;
62         get_f(1);
63     }
64     void get_kth(int k,int rt=1)
65     {
66         if(k>f[rt]){
67             cout<<-1<<endl;
68             return ;

```



```

69         }
70         if(k<=siz[rt]){cout<<"\n";return ;}
71         k--siz[rt];
72         if(k==0){return;}
73         for(int i=0;i<26;i++){
74             if(ch[rt][i]){
75                 if(f[(ch[rt][i])]>=k){
76                     cout<<(char)(i+'a');
77                     get_kth(k,ch[rt][i]);
78                     return ;
79                 }
80                 else
81                     k-=f[ch[rt][i]];
82             }
83         }
84         cout<<-1<<endl;
85     }
86     void debug(){
87         for(int i=0;i<=tot;i++)
88             for(int j=0;j<26;j++)
89                 if(ch[i][j])
90                     cout<<"ins_"<<i<<"_--"<<(char)(j+'a')
91                     <<"-->_"<<ch[i][j]<<endl;
92         for(int i=1;i<=tot;i++)
93             cout<<"fail_"<<i<<"_->_"<<fa[i]<<endl;
94         for(int i=1;i<=tot;i++)
95             cout<<"len["<<i<<"]="<<len[i]<<endl;
96     }
97     int LCS(string&str)
98     {
99         int rt=1;
100        int ans=0;
101        int length=0;
102        for(int i=0;i<str.size();i++)
103        {
104            int tmp=str[i]-'a';
105            while(rt&&!ch[rt][tmp])

```

```
105         rt=fa[rt],length=len[rt];
106         if(!rt)rt=1;
107         else
108             rt=ch[rt][tmp],length++;
109         ans=max(ans,length);
110     }
111     return ans;
112 }
113 private:
114     void get_siz(int x){
115         for(auto y:v[x]){
116             get_siz(y);
117             siz[x]+=siz[y];
118         }
119         f[x]=siz[x];
120     }
121     void get_f(int x){
122         if(vis[x])return ;
123         vis[x]=1;
124         for(int i=0;i<26;i++){
125             int y=ch[x][i];
126             if(!y)continue;
127             get_f(y);
128             f[x]+=f[y];
129         }
130     }
131 }sam;
132 string s,t;
133 void solve()
134 {
135     cin>>s>>t;
136     sam.insert(s);
137     cout<<sam.LCS(t);
138 }
139 signed main(){
140     // freopen("data.in","r",stdin);
141     // freopen("data.out","w",stdout);
```

```
142     ios::sync_with_stdio(false);  
143     cin.tie(nullptr);cout.tie(nullptr);  
144     solve();  
145     return 0;  
146 }
```

8.2 SAM: Multy_LCS

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define endl "\n"
4  #define pp(x) array<int,x>
5  using ull=unsigned long long;
6  using ll=long long;
7  using pii=pair<int,int>;
8  using pdd=pair<double,double>;
9  const int dx[]={0,0,1,-1,1,-1,1,-1};
10 const int dy[]={1,-1,0,0,1,-1,-1,1};
11 const int mod=998244353;
12 const int inf=0x3f3f3f3f;
13 const int INF=1e9+7;
14 const int maxn=2e6+100;
15 class SAM{
16 public:
17     vector<int>v[maxn];
18     int ch[maxn][26],siz[maxn],len[maxn],fa[maxn],mi[maxn]
19         ];
20     bitset<maxn>vis;
21     int f[maxn];
22     int tot=1,np=1;
23     ll ans;
24     void clear(){
25         for(int i=0;i<=tot;i++){
26             for(int j=0;j<26;j++)
27                 ch[i][j]=0;siz[i]=len[i]=fa[i]=vis[i]=0;
28         }tot=np=1;
29     }
30     void insert(const int&c){
31         int p=np; np=++tot;
32         len[np]=len[p]+1; siz[np]=1;
33         for(;p&&!ch[p][c];p=fa[p])ch[p][c]=np;
34         if(p==0)fa[np]=1;
35         else{
```

```

35         int q=ch[p][c];
36         if(len[q]==len[p]+1)fa[np]=q;
37         else{
38             int nq=++tot;
39             len[nq]=len[p]+1;
40             fa[nq]=fa[q]; fa[q]=nq; fa[np]=nq;
41             for(;p&&ch[p][c]==q;p=fa[p])ch[p][c]=nq;
42             memcpy(ch[nq],ch[q],sizeof(ch[q]));
43         }
44     }
45 }
46 void insert(const char*s){for(int i=0;s[i]!='\0';i++)
47     insert(s[i]-'a');}
47 void insert(const string&s){for(int i=0;i<(int)s.size
48     ();i++)insert(s[i]-'a');build_tree();}
48 void build_tree(){
49     for(int i=2;i<=tot;i++)
50         v[fa[i]].push_back(i);
51     for(int i=1;i<=tot;i++)
52         mi[i]=len[i];
53 }
54 /* op=0本质不同子串 op=1位置不同子串 */
55 void build(int op=0){
56     build_tree();
57     if(!op)
58     {
59         for(int i=1;i<=tot;i++)
60             f[i]=siz[i]=1;
61     }
62     else get_siz(1);
63     f[1]=siz[1]=0;
64     get_f(1);
65 }
66 void get_kth(int k,int rt=1)
67 {
68     if(k>f[rt]){
69         cout<<-1<<endl;

```

```

70         return ;
71     }
72     if(k<=siz[rt]){cout<<"\n";return ;}
73     k--siz[rt];
74     if(k==0){return;}
75     for(int i=0;i<26;i++){
76         if(ch[rt][i]){
77             if(f[(ch[rt][i])>=k]){
78                 cout<<(char)(i+'a');
79                 get_kth(k,ch[rt][i]);
80                 return ;
81             }
82             else
83                 k-=f[ch[rt][i]];
84         }
85     }
86     cout<<-1<<endl;
87 }
88 void debug(){
89     for(int i=0;i<=tot;i++)
90         for(int j=0;j<26;j++)
91             if(ch[i][j])
92                 cout<<"ins_"<<i<<"_--"<<(char)(j+'a')
93                 <<"-->_"<<ch[i][j]<<endl;
94     for(int i=1;i<=tot;i++)
95         cout<<"fail_"<<i<<"_>_"<<fa[i]<<endl;
96     for(int i=1;i<=tot;i++)
97         cout<<"len["<<i<<"]="<<len[i]<<endl;
98 }
99 void match(string&str)
100 {
101     for(int i=1;i<=tot;i++)f[i]=0;
102     int rt=1;
103     int length=0;
104     for(int i=0;i<str.size();i++)
105     {
106         int tmp=str[i]-'a';

```

```
106         while(rt&&!ch[rt][tmp])
107             rt=fa[rt],length=len[rt];
108         if(!rt)rt=1;
109         else
110             rt=ch[rt][tmp],length++;
111         f[rt]=max(f[rt],length);
112     }
113     dfs(1,1);
114     for(int i=1;i<=tot;i++)
115         mi[i]=min(mi[i],f[i]);
116 }
117 int query()
118 {
119     int ans=0;
120     for(int i=1;i<=tot;i++)
121         ans=max(ans,mi[i]);
122     return ans;
123 }
124 private:
125     void get_siz(int x){
126         for(auto y:v[x]){
127             get_siz(y);
128             siz[x]+=siz[y];
129         }
130         f[x]=siz[x];
131     }
132     void get_f(int x){
133         if(vis[x])return ;
134         vis[x]=1;
135         for(int i=0;i<26;i++){
136             int y=ch[x][i];
137             if(!y)continue;
138             get_f(y);
139             f[x]+=f[y];
140         }
141     }
142     void dfs(int x,int fa)
```

```
143     {
144         for(auto y:v[x])
145         {
146             dfs(y,x);
147             f[x]=max(f[x],min(len[x],f[y]));
148         }
149     }
150 }sam;
151 string p[maxn];
152 int n;
153 void solve()
154 {
155     cin>>n;
156     int root=1;
157     for(int i=1;i<=n;i++){
158         cin>>p[i];
159         if(p[i].size()<p[root].size())root=i;
160     }
161     swap(p[1],p[root]);
162     sam.insert(p[1]);
163     for(int i=2;i<=n;i++)
164         sam.match(p[i]);
165     cout<<sam.query();
166 }
167 signed main(){
168     // freopen("data.in","r",stdin);
169     // freopen("data.out","w",stdout);
170     ios::sync_with_stdio(false);
171     cin.tie(nullptr);cout.tie(nullptr);
172     solve();
173     return 0;
174 }
```


8.3 SAM: 出现次数大于等于 k 次的最长子串

首先建出后缀树那么对于后缀树上 $siz=1$ 的节点都是字符串的后缀, 那么 siz 向上回溯和大于等于 k 就是出现了至少 k 次

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define endl "\n"
4  #define pp(x) array<int,x>
5  using ull=unsigned long long;
6  using ll=long long;
7  using pii=pair<int,int>;
8  using pdd=pair<double,double>;
9  const int dx[]={0,0,1,-1,1,-1,1,-1};
10 const int dy[]={1,-1,0,0,1,-1,-1,1};
11 const int mod=998244353;
12 const int inf=0x3f3f3f3f;
13 const int INF=1e9+7;
14 const int maxn=1e6+100;
15 class SAM{
16 public:
17     vector<int>v[maxn];
18     int ch[maxn][26],siz[maxn],len[maxn],fa[maxn];
19     bitset<maxn>vis;
20     int f[maxn];
21     int h[maxn];
22     int tot=1,np=1;
23     ll ans;
24     void clear(){
25         for(int i=0;i<=tot;i++){
26             for(int j=0;j<26;j++)
27                 ch[i][j]=0;siz[i]=len[i]=fa[i]=vis[i]=0;
28             }tot=np=1;
29     }
30     int L(int rt){ return len[rt]-len[fa[rt]]; }
31     void insert(const int&c,int pos){
32         int p=np; np=++tot;
33         len[np]=len[p]+1; siz[np]=1;
```

```

34     for(;p&&!ch[p][c];p=fa[p])ch[p][c]=np;
35     if(p==0)fa[np]=1,ans+=L(np);
36     else{
37         int q=ch[p][c];
38         if(len[q]==len[p]+1)fa[np]=q,ans+=L(np);
39         else{
40             int nq=++tot;
41             len[nq]=len[p]+1;
42             ans-=L(q);
43             fa[nq]=fa[q]; fa[q]=nq; fa[np]=nq;
44             ans+=L(nq)+L(q)+L(np);
45             for(;p&&ch[p][c]==q;p=fa[p])ch[p][c]=nq;
46             memcpy(ch[nq],ch[q],sizeof(ch[q]));
47         }
48     }
49     f[pos]=ans;
50 }
51 // void insert(const char*s){for(int i=0;s[i]!='\0';i
    ++)insert(s[i]-'a');}
52 void insert(const string&s){
53     for(int i=0;i<(int)s.size();i++)
54         insert(s[i]-'a',i+1);
55     for(int i=0;i<(int)s.size();i++)
56         h[s[i]-'a']+=f[i+1]-f[i];
57     for(int i=0;i<(int)s.size();i++)
58         cout<<f[i]<<"␣";cout<<endl;
59     cout<<endl;
60     for(int i=0;i<26;i++)
61         cout<<(char)(i+'a')<<"␣"<<h[i]<<endl;
62 }
63 void build_tree(){
64     for(int i=2;i<=tot;i++)
65         v[fa[i]].push_back(i);
66 }
67 /* op=0本质不同子串 op=1位置不同子串 */
68 void build(int op=0){
69     build_tree();

```

```
70         if(!op)
71         {
72             for(int i=1;i<=tot;i++)
73                 f[i]=siz[i]=1;
74         }
75         else get_siz(1);
76         f[1]=siz[1]=0;
77         get_f(1);
78     }
79     void get_kth(int k,int rt=1)
80     {
81         if(k>f[rt]){
82             cout<<-1<<endl;
83             return ;
84         }
85         if(k<=siz[rt]){cout<<"\n";return ;}
86         k-=siz[rt];
87         if(k==0){return;}
88         for(int i=0;i<26;i++){
89             if(ch[rt][i]){
90                 if(f[(ch[rt][i])]>=k){
91                     cout<<(char)(i+'a');
92                     get_kth(k,ch[rt][i]);
93                     return ;
94                 }
95                 else
96                     k-=f[ch[rt][i]];
97             }
98         }
99         cout<<-1<<endl;
100     }
101     void debug(){
102         for(int i=0;i<=tot;i++)
103             for(int j=0;j<26;j++)
104                 if(ch[i][j])
105                     cout<<"ins_"<<i<<"_--"<<(char)(j+'a')
                        <<"-->_"<<ch[i][j]<<endl;
```

```
106         for(int i=1;i<=tot;i++)
107             cout<<"fail_"<<i<<"_->"<<fa[i]<<endl;
108         for(int i=1;i<=tot;i++)
109             cout<<"len["<<i<<"]="<<len[i]<<endl;
110     }
111 private:
112     void get_siz(int x){
113         for(auto y:v[x]){
114             get_siz(y);
115             siz[x]+=siz[y];
116         }
117         f[x]=siz[x];
118     }
119     void get_f(int x){
120         if(vis[x])return ;
121         vis[x]=1;
122         for(int i=0;i<26;i++){
123             int y=ch[x][i];
124             if(!y)continue;
125             get_f(y);
126             f[x]+=f[y];
127         }
128     }
129 }sam;
130 string s,t;
131 int n,m;
132 void solve()
133 {
134     cin>>n>>m>>s>>t;
135     sam.insert(s);
136     for(int i=1;i<=n;i++)
137         cout<<sam.f[i]<<" ";cout<<endl;
138
139 }
140 signed main(){
141     // freopen("data.in","r",stdin);
142     // freopen("data.out","w",stdout);
```

```
143     ios::sync_with_stdio(false);
144     cin.tie(nullptr);cout.tie(nullptr);
145     solve();
146     return 0;
147 }
148 /*
149 3 2
150 aab
151 bc
152
153 4 3
154 abca
155 bba
156 */
```

8.4 SQAM: 公共子串数量

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define endl "\n"
4  #define pp(x) array<int,x>
5  #define int long long
6  const int INF=1e18+7;
7  const int maxn=111;
8  int dp[155][155][155];
9  class SQAM{
10 public:
11     int ch[maxn][26];
12     string s;
13     int tot;
14     void init(){init(s);}
15     void init(string&s)
16     {
17         int n=s.size();
18         // for(int i=0;i<26;i++)
19         //   ch[n][i]=INF;
20         for(int i=n-1;i>=0;i--){
21             for(int j=0;j<26;j++){
22                 ch[i][j]=ch[i+1][j];
23                 ch[i][s[i]-'a']=i+1;
24             }
25         }
26     void debug(){
27         for(int i=0;i<=tot;i++){
28             for(int j=0;j<26;j++){
29                 if(ch[i][j]){
30                     // cout<<i<<" --"<<(char)(j+'a')<<"-->
31                     //   "<<ch[i][j]<<endl;
32                     cout<<i<<"_ "<<ch[i][j]<<"_ "<<(char)(j+
33                     'a')<<endl;
```

```
34 }sa,sb,sc;
35
36 string a,b,c;
37 int n;
38 int dfs(int x,int y,int z)
39 {
40     if(dp[x][y][z])return dp[x][y][z];
41     int res=1;
42     for(int i=0;i<26;i++)
43         if(sa.ch[x][i]&&sb.ch[y][i]&&sc.ch[z][i])
44             res+=dfs(sa.ch[x][i],sb.ch[y][i],sc.ch[z][i]);
45     return dp[x][y][z]=res;
46 }
47 void solve()
48 {
49     cin>>a>>b>>c;
50     sa.init(a);
51     sb.init(b);
52     sc.init(c);
53     cout<<dfs(0,0,0)-1<<endl;
54 }
55 signed main(){
56     // freopen("data.in","r",stdin);
57     // freopen("data.out","w",stdout);
58     ios::sync_with_stdio(false);
59     cin.tie(nullptr);cout.tie(nullptr);
60     solve();
61     return 0;
62 }
```

8.5 SQAM: 公共子序列数量

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define endl "\n"
4  using ll=long long;
5  const int mod=1e8;
6  const int INF=1e9+7;
7  const int maxn=1e6+100;
8  class SQAM{
9  public:
10     int ch[maxn][26];
11     int f[maxn];
12     string s;
13     int tot;
14     void init(){init(s);}
15     void init(string&s)
16     {
17         int n=s.size();
18         // for(int i=0;i<26;i++)
19         //   ch[n][i]=INF;
20         for(int i=n-1;i>=0;i--){
21             for(int j=0;j<26;j++){
22                 ch[i][j]=ch[i+1][j];
23                 ch[i][s[i]-'a']=i+1;
24             }
25         }
26     void debug(){
27         for(int i=0;i<=tot;i++){
28             for(int j=0;j<26;j++){
29                 if(ch[i][j]){
30                     // cout<<i<<" --"<<(char)(j+'a')<<"-->
31                     //   "<<ch[i][j]<<endl;
32                     cout<<i<<"_ "<<ch[i][j]<<"_ "<<(char)(j+
33                     'a')<<endl;
```



```
34     void query(string&str)
35     {
36         int rt=1;
37         f[1]=1;
38         for(auto i:str)
39         {
40             int tmp=i-'a';
41             if(ch[rt][tmp]){
42                 f[ch[rt][tmp]]+=f[rt];
43                 rt=ch[rt][tmp];
44             }
45             else rt=1;
46         }
47         for(int i=1;i<=tot;i++)
48             cout<<f[i]<<" ";cout<<endl;
49     }
50 }sa,sb,sc;
51 ll dp[155][155][155];
52 ll dfs(int x,int y,int z)
53 {
54     if(dp[x][y][z])return dp[x][y][z];
55     for(int i=0;i<26;i++)
56         if(sa.ch[x][i]&&sb.ch[y][i]&&sc.ch[z][i])
57         {
58             dp[x][y][z]+=dfs(sa.ch[x][i],sb.ch[y][i],sc.ch
59                               [z][i]);
60             dp[x][y][z]%=mod;
61         }
62     if(x>1||y>1||z>1)dp[x][y][z]++;
63     return dp[x][y][z]%mod;
64 }
65 int n;
66 void solve()
67 {
68     cin>>n;
69     cin>>sa.s>>sb.s>>sc.s;
70     sa.init();
```

```
70     sb.init();
71     sc.init();
72     cout<<dfs(0,0,0)<<endl;
73 }
74 signed main(){
75     // freopen("data.in","r",stdin);
76     // freopen("data.out","w",stdout);
77     ios::sync_with_stdio(false);
78     cin.tie(nullptr);cout.tie(nullptr);
79     solve();
80     return 0;
81 }
82 /*
83 apartment
84 apache
85 approach
86
87 a p ap pa aa apa
88 */
```

9 网络流建图

9.1 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

9.1.1 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

9.1.2 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

9.1.3 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

9.2 最小割

9.2.1 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

9.3 最小割

定理：

二分图最大独立集 $= n - \text{二分图最大匹配}$

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 $= n - \text{最小割} = n - \text{最大流}$ 。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

9.4 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 $\text{边数} - mid \cdot \text{点数} > 0$ 。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。

9.5 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，有若干个限制，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。

9.6 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

9.7 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 k 条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

9.8 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 $-x_i$ 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。

9.9 有上下界的网络流

9.9.1 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 and 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 $d_i = \text{点 } D \text{ 的入流} - \text{点 } i \text{ 的出流}$ ，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

9.9.2 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

9.9.3 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流量减少。

9.9.4 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
 $= \mincost(ss, tt) + \sum l_i c_i$