## Contents

# 1 基础/配置/黑科技

## 1.1 一般母版

```
/*
  Time:
  Prob:
  By RogerRo
 */
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<vector>
#include<queue>
#include<set>
#include<map>
#include<cmath>
#include<algorithm>
#include<ctime>
#include<bitset>
#define ll long long
#define tr(i,l,r) for((i)=(l);(i)<=(r);++i)
#define rtr(i,r,l) for((i)=(r);(i)>=(l);--i)
#define oo 0x7F7F7F7F
using namespace std;
int read()
{
    int x=0; bool f=0;
    char ch=getchar();
    while (ch<'0'||ch>'9') {f|=ch=='-'; ch=getchar();}
    while (ch>='0'&&ch<='9') {x=(x<<3)+(x<<1)+ch-'0'; ch=getchar();}
    return (x^-f)+f;
}
void write(int x)
{
    char a[20],s=0;
    if (x==0){putchar('0'); return ;}
    if (x<0) {putchar('-'); x=-x;}
    while (x) {a[s++]=x%10+'0'; x=x/10;}
    while (s--) putchar(a[s]);
}
void writeln(int x){write(x); putchar('\n');}
int main()
{

    return 0;
}
```

## 1.2 黑科技

```
//====================万能头文件====================
#include<bits/stdc++.h>
```

```
//====================强制O2优化====================
#pragma GCC optimize(2)
//====================开栈=======================
//g++开栈 放在main开头
int __size__=256<<20;//256MB
char *__p__=(char*)malloc(__size__)+__size__;
__asm__ __volatile__("movq %0,%%rsp\n"::"r"(__p__));
//c++开栈
#pragma comment(linker,"/STACK:102400000,102400000")
//====================C++IO加速====================
#include <iomanip>
ios_base::sync_with_stdio(false);
//====================大数mulmod====================
//int128法
ll mulmod(__int128 x,__int128 y,__int128 mod)   //同理存在__float128
{
    return x*y%mod;
}

//快速乘法
ll mulmod(ll x,ll y,ll mod)
{
  ll ret = 0;
  for(;y;y>>=1)
  {
    if (y&1) ret=(ret+x)%mod;
    x=(x+x)%mod;
  }
  return ret;
}

//汇编法
ll mulmod(ll x,ll y,ll mod) //注意！必须保证x，y都比mod小；可long，不可int
{
    ll ans=0;
    __asm__
    (
        "movq %1,%%rax\n imulq %2\n idivq %3\n"
        :"=d"(ans):"m"(x),"m"(y),"m"(mod):"%rax"
    );
    return ans;
}
//=====================其它小东西=====================
int __gcd(int x,int y)  //<algorithm>且g++才能用
```

## 1.3 位运算

```
//==================枚举i的非空子集j==================
for(j=i;j;j=(j-1)&i);
//=================下一个1的个数相等的数==================
int snoob1(int x)
{
    int y=x&-x,z=x+y;
    return z|((x^z)>>2)/y;
}
int snoob2(int x)   //g++
{
    int t=x|(x-1);
    return (t+1)|(((~t&-~t)-1)>>(__builtin_ctz(x)+1));
}
//======================按位反转======================
int reverse(int x)
{
    x=((x&0x55555555)<<1)|((x&0xAAAAAAAA)>>1);
    x=((x&0x33333333)<<2)|((x&0xCCCCCCCC)>>2);
    x=((x&0x0F0F0F0F)<<4)|((x&0xF0F0F0F0)>>4);
    x=((x&0x00FF00FF)<<8)|((x&0xFF00FF00)>>8);
    x=((x&0x0000FFFF)<<16)|((x&0xFFFF0000)>>16);
    return x;
}
//=======注意！！以下g++下才能用；ll则在函数名后加ll========
int __builtin_popcount(unsigned int x); //1的个数
int __builtin_clz(unsigned int x);      //前缀0的个数
//x为int时，31-__builtin_clz(x) 等价于 int(log(x)/log(2))
int __builtin_ctz(unsigned int x);      //后缀0的个数
int __builtin_parity(unsigned int x);   //1的个数%2
```

## 1.4 离散化

```
//dc[1,2,...]=[x1,x2,..]; rdc(x1,x2,...)=1,2,...
int n,a[maxn],dc[maxn];
int rdc(int x){return lower_bound(dc+1,dc+num+1,x)-dc;}
void init()
{
    //...
    memcpy(dc,a,(n+1)*sizeof(int));
    sort(dc+1,dc+n+1);
    num=unique(dc+1,dc+n+1)-(dc+1);
}
```

## 1.5 Linux 对拍

```
g++ $2 -o 1.out
g++ $3 -o 2.out
cnt=0;
while true; do
g++ $1 -o dm.out
./dm.out>dm.txt
./1.out<dm.txt>1.txt
./2.out<dm.txt>2.txt
if diff 1.txt 2.txt; then let "cnt+=1"; echo ${cnt};
else exit 0;
fi
done
```

### 1.6 vimrc

```
runtime! debian.vim

if has("syntax")
  syntax on
endif

if filereadable("/etc/vim/vimrc.local")
  source /etc/vim/vimrc.local
endif


colo torte
set nu
set ts=4
set sw=4
map <C-A> ggVG"+y
map <F2> :w<CR>
map <F3> :browse e<CR>
map <F4> :browse vsp<CR>
map <F5> :call Run()<CR>
func! Run()
  exec "w"
  exec "!g++ -Wall % -o %<"
  exec "!./%<"
endfunc
```

## 2 数学

### 2.1 高精度类

```
//要sqrt就一定要len和dcm是偶数
//不可以出现如big x=y;的东西，必须分开成big x;x=y;
#define len 3000
#define dcm 3000
void carry(int*x,int y){*(x-1)+=((*x+=y)+10000)/10-1000;*x=(*x+10000)%10;}
struct big
{
    int _[len+2];

    int& operator[](int x){return _[x];}
    big(){memset(_,0,sizeof(int)*(len+2));}
    big(char*x)
    {
        memset(_,0,sizeof(int)*(len+2));
        char *y=x+strlen(x)-1,*z=strchr(x,'.'),*i;
        if (!z) z=y+1;
        int t=dcm-(z-x);
        tr(i,x,y) if(i!=z&&t>=1&&t<=len) _[++t]=*i-'0';
    }

    big& operator=(const big&x){memcpy(_,x._,sizeof(int)*(len+2));return *this
        ;}
    char* c_str()
    {
        char *s=new char[len]; int l,r,i=0,k;
        tr(l,1,len) if(_[l]>0||l==dcm) break;
        rtr(r,len,1) if(_[r]>0||r==dcm) break;
        tr(k,l,r){if(k==dcm+1)s[i++]='.';s[i++]=_[k]+'0';}
        s[i]=0; return s;
    }

    friend int comp(big x,big y) //O(len)
    {
        int i;
        tr(i,1,len) if (x[i]!=y[i]) break;
        return i>len?0:(x[i]>y[i]?1:-1);
    }
    friend big operator+(big x,big y) //O(len)
    {
        big z; int i;
        rtr(i,len,1) carry(&z[i],x[i]+y[i]);
        return z;
    }
    friend big operator-(big x,big y) //O(len)
    {
        big z; int i;
        rtr(i,len,1) carry(&z[i],x[i]-y[i]);
        return z;
    }
    friend big operator*(big x,big y) //O(len^2)
    {
        big z; int i,j;
        rtr(i,len,1) rtr(j,min(dcm+len-i,len),max(dcm+1-i,1))
            carry(&z[i+j-dcm],x[i]*y[j]);
        return z;
    }
    friend big operator/(big x,big y) //O(len^2)
    {
        big z,t,tmp[10]; int i,j,k;
        tr(k,1,9) tmp[k]=tmp[k-1]+y;
        tr(j,1,len-dcm) t[j+dcm]=x[j];
        j--;
        tr(i,1,len)
        {
            tr(k,1,len-1) t[k]=t[k+1];
            t[len]=++j<=len?x[j]:0;
            tr(k,1,9) if (comp(tmp[k],t)>0) break;
            z[i]=--k;
            t=t-tmp[k];
        }
        return z;
    }
    friend int sqrt_deal(big&y,int a,int b,int l)
    {
        int t=a+y[b]%10-9;
        if(2*b>l)t-=(y[2*b-l])/10;
        if (b>=0&&!(a=sqrt_deal(y,t/10,b-1,l))) y[b]+=(t+999)%10-y[b]%10;
        return a;
    }
    friend big sqrt(big x) //O(len^2)
    {
        int l,t=dcm/2; big y,z; y=x;
        for(l=1;l<=len;l++)
        {
            y[++l]+=10;
            while (!sqrt_deal(y,0,l,l)) y[l]+=20;
            z[++t]=y[l]/20; y[l]-=10;
        }
        return z;
    }
    friend big floor(big x)
    {
        big z; z=x; int i;
        tr(i,dcm+1,len) z[i]=0;
        return z;
    }
    friend big ceil(big x){return comp(x,floor(x))==0?x:floor(x+big("1"));}
};
```

### 2.2 筛素数-欧拉筛法

$O(N)$

```
int prime[maxm],a[n];
bool pprime[n];
void EulerPrime()
{
  int i,j;
  tr(i,2,n) pprime[i]=1;
  tr(i,2,n)
  {
    if (pprime[i]) prime[++m]=i;
    tr(j,1,m)
    {
      if (i*prime[j]>n) break;
      pprime[i*prime[j]]=0;
      if (i%prime[j]==0) break;
    }
  }
}
```

### 2.3 高阶代数方程求根-求导

$O(N^3 * S)$，S 取决于精度

```
//求导至最高次为t时，a[t][i]表x^i的系数，ans[t]记录根；oo依题而定
double a[maxn][maxn],ans[maxn][maxn];
int n,anss[maxn];
double get(int x,double y)
{
    int i; double res=0;
    rtr(i,x,0) res=res*y+a[x][i];
    return res;
}
void dich(int x,double ll,double rr)
{
    if (cmp(get(x,ll))==0){ans[x][++anss[x]]=ll;return;}
    if (cmp(get(x,rr))==0){ans[x][++anss[x]]=rr;return;}
    if (cmp(get(x,ll)*get(x,rr))>0) return;
    double l=ll,r=rr,mid;
    while (l+eps<r) //亦可改为循环一定次数
    {
        int tl=cmp(get(x,l)),tm=cmp(get(x,mid=(l+r)/2));
        if (tl==0) break;
        if (tl*tm>=0) l=mid; else r=mid;
    }
    ans[x][++anss[x]]=l;
}
void work()
{
    int i,j; double l,r;
    rtr(i,n-1,1) tr(j,0,i) a[i][j]=a[i+1][j+1]*(j+1);
    tr(i,0,n-1)
    {
        l=-oo;
        tr(j,1,anss[i]){dich(i+1,l,r=ans[i][j]); l=r;}
        dich(i+1,l,oo);
    }
    tr(i,1,anss[n]) printf("%.10lf\n",ans[n][i]);
}
```

## 3 几何

### 3.1 平面几何类包

```
#define maxpn 10005
#define nonx 1E100
#define eps 1E-8
const double pi=acos(-1.0);
int cmp(double x)
{
    if (x>eps) return 1;
    if (x<-eps) return -1;
    return 0;
}
double sqr(double a){return a*a;}
int gcd(int a,int b){return a%b==0?b:gcd(b,a%b);}
struct point
{
    double x,y;
    point(){}
    point(double a,double b){x=a;y=b;}

    friend point operator+(point a,point b){return point(a.x+b.x,a.y+b.y);}
```

```cpp
    friend point operator-(point a,point b){return point(a.x-b.x,a.y-b.y);}
    friend point operator-(point a){return point(-a.x,-a.y);}
    friend double operator*(point a,point b){return a.x*b.x+a.y*b.y;}
    friend point operator*(double a,point b){return point(a*b.x,a*b.y);}
    friend point operator*(point a,double b){return point(a.x*b,a.y*b);}
    friend point operator/(point a,double b){return point(a.x/b,a.y/b);}
    friend double operator^(point a,point b){return a.x*b.y-a.y*b.x;}
    friend bool operator==(point a,point b){return cmp(a.x-b.x)==0&&cmp(a.y-b.y)==0;}

    friend double sqr(point a){return a*a;}
    friend double len(point a){return sqrt(sqr(a));}    //模长
    friend point rotate(point a,double b){return point(a.x*cos(b)-a.y*sin(b),a.x*sin(b)+a.y*cos(b));}   //逆时针旋转
    friend double angle(point a,point b){return acos(a*b/len(a)/len(b));}   //夹角
    friend point reflect(point a,point b){return 2*a-b;}    //以a为中心对称
};
const point nonp=point(nonx,nonx);
point quad(double A,double B,double C)
{
    double delta=sqr(B)-4*A*C;
    if (delta<0) return nonp;
    return point((-B-sqrt(delta))/(2*A),(-B+sqrt(delta))/(2*A));
}
struct line
{
    point a,b;
    line(){}
    line(point pa,point pb){a=pa;b=pb;}
    point dir(){return b-a;}

    friend point proj(point a,line b){double t=(a-b.a)*b.dir()/sqr(b.dir());return point(b.a+t*b.dir());}   //垂足
    friend double dist(point a,line b){return ((a-b.a)^(b.b-b.a))/len(b.dir());}    //点到线距离
    friend bool onray(point a,line b){return cmp((a-b.a)^b.dir())==0&&cmp((a-b.a)*b.dir())>=0;} //判断点在射线上
    friend bool onseg(point a,line b){return cmp((a-b.a)^b.dir())==0&&cmp((a-b.a)*(a-b.b))<=0;} //判断点在线段上
    friend bool online(point a,line b){return cmp((a-b.a)^b.dir())==0;} //判断点在直线上
    friend bool parallel(line a,line b){return cmp(a.dir()^b.dir())==0;}    //判断两线平行
    friend point cross(line a,line b)   //线交
    {
        double t;
        if (cmp(t=a.dir()^b.dir())==0) return nonp;
        return a.a+((b.a-a.a)^b.dir())/t*a.dir();
    }
};
const line nonl=line(nonp,nonp);
struct circle
{
    point o; double r;
    circle(){}
    circle(point a,double b){o=a;r=b;}

    friend double S(circle a){return pi*sqr(a.r);}  //面积
    friend double C(circle a){return 2*pi*a.r;} //周长
    friend line cross(line a,circle b)  //线圆交
  {
    point t=quad(sqr(a.dir()),2*a.dir()*(a.a-b.o),sqr(a.a-b.o)-sqr(b.r));
    if (t==nonp) return nonl;
    return line(a.a+t.x*a.dir(),a.a+t.y*a.dir());
  }
    friend int in(point a,circle b){double t=len(a-b.o);return t==b.r?2:t<b.r;} //点与圆位置关系 0外 1内 2上
    //friend line cross(circle a,circle b){}
    //friend line tangent(point a,circle b){}
    //friend pair<line,line> tangent(circle a,circle b){}
    //friend double unionS(int n,circle*a) //圆面积并
    //{}
};
struct triangle//t 因triangle亦属polygon，故省去许多函数
{
    point a,b,c;
    triangle(){}
    triangle(point ta,point tb,point tc){a=ta;b=tb;c=tc;}

    friend double S(triangle a){return abs((a.b-a.a)^(a.c-a.a))/2;} //面积
    friend double C(triangle a){return len(a.a-a.b)+len(a.a-a.c)+len(a.a-a.c);} //周长
    friend circle outcircle(triangle a) //外接圆
    {
        circle res; point t1=a.b-a.a,t2=a.c-a.a;
        double t=2*t1^t2;
        res.o.x=a.a.x+(sqr(t1)*t2.y-sqr(t2)*t1.y)/t;
        res.o.y=a.a.y+(sqr(t2)*t1.x-sqr(t1)*t2.x)/t;
        res.r=len(res.o-a.a);
        return res;
    }
    friend circle incircle(triangle a)  //内切圆
    {
        circle res; double x=len(a.b-a.c),y=len(a.c-a.a),z=len(a.a-a.b);
        res.o=(a.a*x+a.b*y+a.c*z)/(x+y+z);
        res.r=dist(res.o,line(a.a,a.b));
        return res;
    }
    friend point gc(triangle a){return (a.a+a.b+a.c)/3;}    //重心
    friend point hc(triangle a){return 3*gc(a)-2*outcircle(a).o;}   //垂心
};
struct polygon
{
    int n; point a[maxpn];  //逆时针！
    polygon(){}
    polygon(triangle t){n=3;a[1]=t.a;a[2]=t.b;a[3]=t.c;}
    point& operator[](int _){return a[_];}

    friend double S(polygon a)  //面积 O(n)
    {
        int i; double res=0;
        a[a.n+1]=a[1];
        tr(i,1,a.n) res+=a[i]^a[i+1];
        return res/2;
    }
    friend double C(polygon a) //周长 O(n)
    {
        int i; double res=0;
        a[a.n+1]=a[1];
        tr(i,1,a.n) res+=len(a[i+1]-a[i]);
        return res;
    }
    friend int in(point a,polygon b)  //点与多边形位置关系 0外 1内 2上 O(n)
    {
        int s=0,i,d1,d2,k;
        b[b.n+1]=b[1];
        tr(i,1,b.n)
        {
            if (onseg(a,line(b[i],b[i+1]))) return 2;
            k=cmp((b[i+1]-b[i])^(b[i]-a));
            d1=cmp(b[i].y-a.y);
            d2=cmp(b[i+1].y-a.y);
            s=s+(k>0&&d2<=0&&d1>0)-(k<0&&d1<=0&&d2>0);
        }
        return s!=0;
    }
    friend point gc(polygon a)  //重心 O(n)
    {
        double s=S(a); point t(0,0); int i;
        if (cmp(s)==0) return nonp;
        a[a.n+1]=a[1];
        tr(i,1,a.n) t=t+(a[i]+a[i+1])*(a[i]^a[i+1]);
        return t/s/6;
    }
    friend int pick_on(polygon a)  //皮克求边上格点数 O(n)
    {
        int s=0,i;
        a[a.n+1]=a[1];
        tr(i,1,a.n) s+=gcd(abs(int(a[i+1].x-a[i].x)),abs(int(a[i+1].y-a[i].y)));
        return s;
    }
    friend int pick_in(polygon a){return int(S(a))+1-pick_on(a)/2;} //皮克求多边形内格点数 O(n)

    //friend line convex_maxdist(polygon a){}
    //friend line mindist(polygon a){} //a只是点集
    //friend polygon convex_hull(polygon a){} //a只是点集 O(nlogn)
    //friend int convex_in(point a,polygon b){} //0外 1内 2上 O(logn)
    //friend polygon cross(polygon a,polygon b){}
    //friend polygon cross(line a,polygon b){}
    //friend double unionS(circle a,polygon b){}
    friend circle mincovercircle(polygon a) //最小圆覆盖 O(n)
    {
        circle t; int i,j,k;
        srand(time(0));
        random_shuffle(a.a+1,a.a+a.n+1);
        for(i=2,t=circle(a[1],0);i<=a.n;i++) if (!in(a[i],t))
            for(j=1,t=circle(a[i],0);j<i;j++) if (!in(a[j],t))
                for(k=1,t=circle((a[i]+a[j])/2,len(a[i]-a[j])/2);k<j;k++) if (!in(a[k],t))
                    t=outcircle(triangle(a[i],a[j],a[k]));
        return t;
    }
} ;
```

## 4 博弈

## 5 DP

## 6 串

### 6.1 多模匹配-AC 自动机

求 n 个模式串中有多少个出现过，模式串相同算作多个，$O(\sum P_i + T)$

```
//maxt=文本串长，maxp=模式串长，maxn=模式串数
struct ac{int s,to[26],fail;} a[maxn*maxp];
int m,n;
char ts[maxp],s[maxt];
queue<int> b;
void clear(int x)
{
    a[x].s=a[x].fail=0;
    memset(a[x].to,0,sizeof(a[x].to));
}
void ins(char *st)
{
    int i,x=0,c,l=strlen(st);
    tr(i,0,l-1)
    {
        if (!a[x].to[c=st[i]-'a']) {a[x].to[c]=++m; clear(m);}
        x=a[x].to[c];
    }
    a[x].s++;
}
void build()
{
    int i,h,t;
    tr(i,0,25) if (t=a[0].to[i]) b.push(t);
    while (b.size())
    {
        h=b.front(); b.pop();
        tr(i,0,25)
        if (t=a[h].to[i])
        {
            a[t].fail=a[a[h].fail].to[i];
            b.push(t);
        } else a[h].to[i]=a[a[h].fail].to[i];
    }
}
int cnt(char *st)
{
    int i,x=0,c,t,cnt=0,l=strlen(st);
    tr(i,0,l-1)
    {
        c=st[i]-'a';
        while (!a[x].to[c]&&x) x=a[x].fail;
        x=a[x].to[c];
        for(t=x;t&&a[t].s>-1;t=a[t].fail) {cnt+=a[t].s; a[t].s=-1;}
    }
    return cnt;
}
void work()
{
    int i;
    m=0; clear(0);
    scanf("%d",&n);
    tr(i,1,n)
    {
        scanf("%s",ts); ins(ts);
    }
    build();
    scanf("%s",s); printf("%d\n",cnt(s));
}
```

# 7 图/树

## 7.1 单源最短路-Dijkstra

不加堆，$O(V^2+E)$

```
struct edge{int pre,x,y,d;} a[maxm];
int n,m,ah[maxn],d[maxn];
bool p[maxn];
void update(int x)
{
  int e;
  p[x]=true;
  for(e=ah[x];e>-1;e=a[e].pre)
    if (!p[a[e].y]&&(!d[a[e].y]||a[e].d+d[x]<d[a[e].y]))
      d[a[e].y]=a[e].d+d[x];
}
void dijkstra()
{
  int i,j,t;
    memset(p,0,sizeof(p));
  update(1);
  d[0]=oo;
  tr(i,2,n)
  {
    t=0;
    tr(j,1,n) if (!p[j]&&d[j]&&d[j]<d[t]) t=j;
    update(t);
  }
  printf("%d\n",d[n]);
}
```

加堆，$O(ElogE+V)$

```
typedef pair<int,int> pa;
struct edge{int pre,x,y,d;} a[maxm];
int n,m,ah[maxn],ans[maxn];
priority_queue<pa,vector<pa>,greater<pa> >d;
bool p[maxn];
void dijkstra()
{
  int v,s,e;
    memset(p,0,sizeof(p));
  d.push(make_pair(0,1));
  while(!d.empty())
  {
    v=d.top().second;
    s=d.top().first;
    d.pop();
    if (p[v]) continue;
    p[v]=1;
    ans[v]=s;
    for(e=ah[v];e>-1;e=a[e].pre)
      if (!p[a[e].y]) d.push(make_pair(s+a[e].d,a[e].y));
  }
  printf("%d\n",ans[n]);
}
```

## 7.2 最短路-Floyd

$O(V^3+E)$

```
void floyd()
{
  int i,j,k;
  tr(k,1,n) tr(i,1,n)
    if (a[i][k]) tr(j,1,n)
      if (i!=j&&a[k][j]&&(!a[i][j]||(a[i][j]&&a[i][k]+a[k][j]<a[i][j])))
              a[i][j]=a[i][k]+a[k][j];
}
```

## 7.3 单源最短路-SPFA

不加优化，$O(VE+V^2)=O(kE)$

```
struct edge{int pre,x,y,d;} a[maxm];
int n,m,last[maxn],d[maxn],b[maxn];
bool p[maxn];
void spfa()
{
  int h,t,e;
  memset(d,0x7F,sizeof(d));
```

```
    memset(p,0,sizeof(p));
  b[0]=1; p[1]=1; d[1]=0;
  h=n-1; t=0;
  while (h!=t)
  {
    h=(h+1)%n;
    for (e=last[b[h]];e>-1;e=a[e].pre)
      if (d[a[e].x]+a[e].d<d[a[e].y])
      {
        d[a[e].y]=d[a[e].x]+a[e].d;
        if (!p[a[e].y])
        {
          t=(t+1)%n;
          b[t]=a[e].y;
          p[a[e].y]=1;
        }
      }
    p[b[h]]=0;
  }
  printf("%d\n",d[n]);
}
```

SLF+LLL 优化，$O(VE+V^2)=O(kE)$

```
//a从1开始！
struct edge{int pre,x,y,d;} a[maxm];
int n,m,last[maxn],d[maxn],b[maxn];
bool p[maxn];
void spfa()
{
  int e,h,t,sum,num;
  memset(d,0x7F,sizeof(d));
    memset(p,0,sizeof(p));
  b[0]=1; p[1]=1; d[1]=0;
  sum=0; num=1;
  h=0; t=0;
  while (num)
  {
    while (d[h]*num>sum)
    {
      t=(t+1)%n;
      b[t]=b[h];
      h=(h+1)%n;
    }
    e=last[b[h]];
    p[b[h]]=0;
    num--;
    sum-=d[a[e].x];
    h=(h+1)%n;
    for (;a[e].x;e=a[e].pre)
      if (d[a[e].x]+a[e].d<d[a[e].y])
      {
        if (p[a[e].y]) sum-=d[a[e].y];
        d[a[e].y]=d[a[e].x]+a[e].d;
        sum+=d[a[e].y];
        if (!p[a[e].y])
        {
          if (num && d[a[e].y]<d[b[h]])
          {
            h=(h+n-1)%n;
            b[h]=a[e].y;
          } else
          {
            t=(t+1)%n;
            b[t]=a[e].y;
          }
          p[a[e].y]=1;
          num++;
        }
      }
  }
  printf("%d\n",d[n]);
}
```

## 7.4 二分图最大匹配-匈牙利

$O(VE)$

```
struct edge{int x,y,pre;} a[maxm];
int nx,ny,m,last[maxn],my[maxn];
bool p[maxn];
int dfs(int x)
{
  for (int e=last[x];e>-1;e=a[e].pre)
    if (!p[a[e].y])
    {
      int y=a[e].y;
      p[y]=1;
      if (!my[y]||dfs(my[y])) return my[y]=x;
    }
  return 0;
}
void hungary()
{
  int i,ans=0;
  memset(my,0,sizeof(my));
  tr(i,1,nx)
  {
    memset(p,0,sizeof(p));
    if (dfs(i)) ans++;
  }
  printf("%d\n",ans);
}
```

## 7.5 有向图极大强连通分量-Tarjan 强连通

$O(V+E)$

```
//ds，ss，gs分别是dfn，sta，group计数器；group记所属分量号码，size记分量大小；
    insta记是否在栈中
```

```
struct edge{int x,y,pre;} a[maxm];
int n,m,ah[maxn],ds,dfn[maxn],low[maxn],ss,sta[maxn],gs,group[maxn],size[maxn
    ];
bool insta[maxn];
void tarjan(int x)
{
    int e,y,t;
    dfn[x]=low[x]=++ds;
    sta[++ss]=x; insta[x]=1;
    for(e=ah[x];e>-1;e=a[e].pre)
    {
        if (!dfn[y=a[e].y]) tarjan(y);
        if (insta[y]) low[x]=min(low[x],low[y]);
    }
    if (low[x]==dfn[x])
        for(gs++,t=0;t!=x;t=sta[ss--]) {group[sta[ss]]=gs; size[gs]++;}
}
void work()
{
    ds=ss=gs=0;
    int i; tr(i,1,n) if (!dfn[i]) tarjan(i);
}
```

### 7.6 最大流-iSAP

简版（无 BFS，递归，gap，cur），$O(V^2 * E)$

```
struct edge{int x,y,c,f,pre;} a[2*maxm];
int n,mm,m,last[maxn],d[maxn],gap[maxn],cur[maxn],ans;
void newedge(int x,int y,int c,int f)
{
  m++;
  a[m].x=x; a[m].y=y; a[m].c=c; a[m].f=f;
  a[m].pre=last[x]; last[x]=m;
}
void init()
{
  int i,x,y,c;
  m=-1;
  memset(last,-1,sizeof(last));
  tr(i,1,mm)
  {
    x=read(); y=read(); c=read();
    newedge(x,y,c,0);
    newedge(y,x,c,c);
  }
  tr(i,1,n) cur[i]=last[i];
    memset(d,0,sizeof(d));
  memset(gap,0,sizeof(gap));
  gap[0]=n;
  ans=0;
}
int sap(int x,int flow)
{
  int e,t;
  if (x==n) return flow;
  for (e=cur[x];e!=-1;e=a[e].pre)
    if (a[e].f<a[e].c && d[a[e].y]+1==d[x])
    {
      cur[x]=e;
      if (t=sap(a[e].y,min(flow,a[e].c-a[e].f)))
      {
        a[e].f+=t; a[e^1].f-=t; return t;
      }
    }
  if (--gap[d[x]]==0) d[n]=n;
  d[x]=n;
  for (e=last[x];e!=-1;e=a[e].pre)
    if (a[e].f<a[e].c) d[x]=min(d[x],d[a[e].y]+1);
  cur[x]=last[x];
  ++gap[d[x]];
  return 0;
}
int work()
{
    while (d[n]<n) ans+=sap(1,oo);
}
```

完全版（有 BFS，非递归，gap，cur），$O(V^2 * E)$

```
int n,mm,m,ans,last[maxn],cur[maxn],pre[maxn],d[maxn],gap[maxn],b[maxn];
bool p[maxn];
struct edge{int x,y,c,f,pre;} a[2*maxm];
void newedge(int x,int y,int c,int f)
{
  m++;
  a[m].x=x; a[m].y=y; a[m].c=c; a[m].f=f;
  a[m].pre=last[x]; last[x]=m;
}
void init()
{
  int i,x,y,c;
  m=-1;
  memset(last,-1,sizeof(last));
  tr(i,1,mm)
  {
    x=read(); y=read(); c=read();
    newedge(x,y,c,0);
    newedge(y,x,c,c);
  }
}
int aug()
{
  int x,flow=a[cur[1]].c-a[cur[1]].f;
  for (x=pre[n];x>1;x=pre[x]) flow=min(flow,a[cur[x]].c-a[cur[x]].f);
  return flow;
}
void bfs()
{
  int h,t,e;
  memset(p,0,sizeof(p));
  b[1]=n; p[n]=1;
  h=0; t=1;
  while (h<t)
  {
    h++;
    for (e=last[b[h]];e!=-1;e=a[e].pre)
      if (a[e].c==a[e].f && !p[a[e].y])
      {
        b[++t]=a[e].y;
        p[a[e].y]=1;
        d[a[e].y]=d[a[e].x]+1;
      }
  }
}
void sap()
{
  int x,e,flow;
  memset(d,0,sizeof(d));
  memset(gap,0,sizeof(gap));
  bfs();
  tr(x,1,n) gap[d[x]]++;
  ans=0;
  tr(x,1,n) cur[x]=last[x];
  x=1; pre[1]=1;
  while (d[1]<n)
  {
    for (e=cur[x];e!=-1;e=a[e].pre)
      if (d[x]==d[a[e].y]+1 && a[e].f<a[e].c)
      {
        cur[x]=e;
        pre[a[e].y]=x;
        x=a[e].y;
        break;
      }
    if (e==-1)
    {
      if (!(--gap[d[x]])) return;
      cur[x]=last[x];
      d[x]=n;
      for (e=last[x];e!=-1;e=a[e].pre)
        if (a[e].f<a[e].c) d[x]=min(d[x],d[a[e].y]+1);
      gap[d[x]]++;
      x=pre[x];
    }
    if (x==n){
      flow=aug();
      for (x=pre[x];x>1;x=pre[x])
      {
        a[cur[x]].f+=flow; a[cur[x]^1].f-=flow;
      }
      a[cur[x]].f+=flow; a[cur[x]^1].f-=flow;
      ans+=flow;
      x=1;
    }
  }
}
```

## 7.7 最小生成树-Prim

不加堆，$O(V+E)$

```
struct edge{int x,y,d,pre;} a[maxm];
int n,m,ah[maxn],d[maxn];
bool p[maxn];
void prim()
{
  int i,j,x,y,e,ans=0;
  memset(d,0x7f,sizeof(d)); d[1]=0;
  memset(p,0,sizeof(p));
  tr(i,1,n)
  {
    x=0;
    tr(j,1,n) if (!p[j]&&d[j]<d[x]) x=j;
    ans+=d[x];
    p[x]=1;
    for(e=ah[x];e>-1;e=a[e].pre)
      if (!p[y=a[e].y]) d[y]=min(d[y],a[e].d);
  }
  printf("%d\n",ans);
}
```

加堆，$O(V+E)$

```
struct edge{int x,y,d,pre;} a[maxm];
typedef pair<int,int> pa;
priority_queue<pa,vector<pa>,greater<pa> >d;
int n,m,ah[maxn];
bool p[maxn];
void prim()
{
  int i,x,y,e,ans=0;
  pa t;
  while (!d.empty()) d.pop();
  d.push(make_pair(0,1));
  memset(p,0,sizeof(p));
  tr(i,1,n)
  {
        while (!d.empty()&&p[d.top().second]) d.pop();
    t=d.top();
    ans+=t.first;
    p[x=t.second]=1;
    for(e=ah[x];e>-1;e=a[e].pre)
      if (!p[y=a[e].y]) d.push(make_pair(a[e].d,y));
  }
  printf("%d\n",ans);
}
```

## 7.8 最小生成树-Kruskal

$O(ElogE+E)$

```
//a从1开始！
```

```
struct edge{int x,y,d;} a[maxm];
bool cmp(edge a,edge b){return a.d<b.d;}
int n,i,j,m,fa[maxn];
int gfa(int x){return x==fa[x]?x:fa[x]=gfa(fa[x]);}
void kruskal()
{
  int ans,fx,fy;
  sort(a+1,a+m+1,cmp);
  tr(i,1,n) fa[i]=i;
  ans=0;
  tr(i,1,m)
    if ((fx=gfa(a[i].x))!=(fy=gfa(a[i].y)))
    {
      fa[fx]=fy;
      ans+=a[i].d;
    }
  printf("%d\n",ans);
}
```

### 7.9 树的直径-BFS

$O(N)$

```
struct edge{int x,y,d,pre;} a[2*maxn];
int n,m,ah[maxn],d0[maxn],d1[maxn],b[maxn];
bool p[maxn];
void bfs(int root,int *d)
{
  int h,t,e,y;
  memset(p,0,sizeof(p));
  h=0; t=1;
  b[1]=root;
  p[root]=1;
  while (h<t)
  {
    h++;
    for (e=ah[b[h]];e>-1;e=a[e].pre)
      if (!p[y=a[e].y])
      {
        b[++t]=y;
        p[y]=1;
        d[y]=d[a[e].x]+a[x].d;
      }
  }
}
void work()
{
  int i,s1,s2;
    memset(d0,0,sizeof(d0));
  memset(d1,0,sizeof(d1));
  bfs(1,d0); s1=1; tr(i,1,n) if (d0[i]>d0[s1]) s1=i;
  bfs(s1,d1); s2=1; tr(i,1,n) if (d1[i]>d1[s2]) s2=i;
    printf("%d %d %d\n",s1,s2,d1[s2]);
}
```

### 7.10 LCA-TarjanLCA

$O(N+Q)$

```
struct query{int x,y,pre,lca;} b[2*maxq];
struct edge{int x,y,pre,d;} a[2*maxn];
int n,q,am,bm,ah[maxn],bh[maxn],fa[maxn],dep[maxn];
bool p[maxn];
int gfa(int x){return fa[x]==x?x:fa[x]=gfa(fa[x]);}
void tarjan(int x,int depth)
{
    int tmp,y;
    p[x]=1;
    dep[x]=depth;
    for(tmp=ah[x];tmp>-1;tmp=a[tmp].pre)
        if (!p[y=a[tmp].y])
        {
            tarjan(y,depth+a[tmp].d);
            fa[y]=x;
        }
    for(tmp=bh[x];tmp>-1;tmp=b[tmp].pre)
        if (p[y=b[tmp].y]) b[tmp].lca=b[tmp^1].lca=gfa(y);
}
void work()
{
    memset(dep,0,sizeof(dep));
    memset(p,0,sizeof(p));
    tarjan(1,0);
    int i; tr(i,0,q-1) writeln(dep[b[2*i].x]+dep[b[2*i].y]-2*dep[b[2*i].lca]);
}
```

# 8 数据结构

## 8.1 并查集

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

## 8.2 区间和 _ 单点修改区间查询-树状数组

$O(NlogN+QlogN)$

```
int n,a[maxn],f[maxn];
char tc;
void modify(int x,int y)
{
    while (x<=n) {f[x]+=y; x+=x&-x;}
}
int sum(int x)
{
    int res=0;
    while (x) {res+=f[x]; x-=x&-x;}
    return res;
}
void work()
{
    int q,i,tx,ty;
    n=read(); q=read();
    memset(f,0,sizeof(f));
    tr(i,1,n) modify(i,a[i]=read());
    tr(i,1,q)
    {
        tc=getchar(); tx=read(); ty=read();
        if (tc=='M') {modify(tx,ty-a[tx]); a[tx]=ty;}
        else writeln(sum(ty)-sum(tx-1));
    }
}
```

### 8.3 区间和 _ 区间修改单点查询-树状数组

$O(NlogN + QlogN)$

```
int n,i,f[maxn];
void modify(int x,int y)
{
    while (x) {f[x]+=y; x-=x&-x;}
}
int sum(int x)
{
    int res=0;
    while (x<=n) {res+=f[x]; x+=x&-x;}
    return res;
}
void work()
{
    int q,i;
    n=read(); q=read();
    memset(f,0,sizeof(f));
    tr(i,1,q)
    {
        tc=getchar();
        if (tc=='M') {modify(read()-1,-1); modify(read(),1);}
        else writeln(sum(read()));
    }
}
```

### 8.4 区间和-线段树

$O(NlogN + QlogN)$

```
struct node{int s,tag;} a[4*maxn];
int n;
void update(int t,int l,int r)
{
    if (l!=r)
    {
        a[t<<1].tag+=a[t].tag;
        a[t<<1|1].tag+=a[t].tag;
    }
    a[t].s+=(int)(r-l+1)*a[t].tag;
    a[t].tag=0;
}
void add(int t,int l,int r,int x,int y,int z)
{
    if (x<=l&&r<=y) {a[t].tag+=z; return ;}
    a[t].s+=(int)(min(r,y)-max(l,x)+1)*z;
    update(t,l,r);
    int mid=(l+r)>>1;
    if (x<=mid) add(t<<1,l,mid,x,y,z);
    if (y>mid) add(t<<1|1,mid+1,r,x,y,z);
}
int sum(int t,int l,int r,int x,int y)
{
    int res=0;
    update(t,l,r);
    if (x<=l&&r<=y) return a[t].s;
    int mid=(l+r)>>1;
    if (x<=mid) res+=sum(t<<1,l,mid,x,y);
    if (y>mid) res+=sum(t<<1|1,mid+1,r,x,y);
    return res;
}
void work()
{
    int q,i,tx,ty; char tc;
    n=read(); q=read();
    tr(i,1,n) add(1,1,n,i,i,read());
    tr(i,1,q)
    {
        tc=getchar(); tx=read(); ty=read();
        if (tc=='A') add(1,1,n,tx,ty,read());
        else writeln(sum(1,1,n,tx,ty));
    }
}
```

### 8.5 区间第 k 大 _ 无修改-主席树

$O(NlogN + QlogN)$

```
struct node{int l,r,size;} a[maxm];
int n,q,m,num,b[maxn],dc[maxn],root[maxn];
int rdc(int x){return lower_bound(dc+1,dc+num+1,x)-dc;}
void init()
{
    int i;
    n=read(); q=read();
    tr(i,1,n) b[i]=read();
    memcpy(dc,b,(n+1)*sizeof(int));
    sort(dc+1,dc+n+1);
    num=unique(dc+1,dc+n+1)-(dc+1);
}
int insert(int tx,int l,int r,int x)
{
    int t,mid=(l+r)>>1;
    a[t=++m]=a[tx]; a[t].size++;
    if (l==r) return t;
    if (x<=mid) a[t].l=insert(a[tx].l,l,mid,x);
    else a[t].r=insert(a[tx].r,mid+1,r,x);
    return t;
}
int kth(int tx,int ty,int l,int r,int k)
{
    int ds,mid=(l+r)>>1;
    if (l==r) return l;
    if (k<=(ds=a[a[ty].l].size-a[a[tx].l].size))
        return kth(a[tx].l,a[ty].l,l,mid,k);
    else return kth(a[tx].r,a[ty].r,mid+1,r,k-ds);
}
void work()
{
    int i,x,y,z;
    tr(i,1,n) root[i]=insert(root[i-1],1,num,rdc(b[i]));
    tr(i,1,q)
    {
        x=read(); y=read(); z=read();
        writeln(dc[kth(root[x-1],root[y],1,num,z)]);
    }
}
```

### 8.6 RMQ-ST

$O(NlogN)$ $O(1)$

```
//！！注意！！__builtin_clz只有g++能用
//x为int时，31-__builtin_clz(x) 等价于 int(log(x)/log(2))
//x为ll时，63-__builtin_clzll(x) 等价于 (ll)(log(x)/log(2))
int n,q,mn[maxn][maxln];
void init()
{
    int i;
    n=read(); q=read();
    tr(i,1,n) mn[i][0]=read();
}
void st()
{
    int i,j,ln;
    ln=31-__builtin_clz(n);
    tr(i,1,ln) tr(j,1,n-(1<<i)+1)
        mn[j][i]=min(mn[j][i-1],mn[j+(1<<(i-1))][i-1]);
}
void work()
{
    int i,x,y,t;
    st();
    tr(i,1,q)
    {
        x=read(); y=read();
        t=31-__builtin_clz(y-x+1);
        writeln(min(mn[x][t],mn[y-(1<<t)+1][t]));
    }
}
```

## 9 其它