1 目录

2 Vimrc

```
e $VIM\.vimrc
set nu
set cin sw=2
map <F9> :w<return> :!g++ -Wall % -o %< <return>:!./%<
<return>
map <M-a> i//<ESC>h
map <M-d> xx
```

3 MinCostMaxFlow

```
struct Arc
{
  int dest,cost,rest;
  Arc *rev,*next;
  Arc(){};
  Arc(int dest,int cost,int rest,Arc *next):
    dest(dest),cost(cost),rest(rest),next(next){};
}Npool[maxE],*Nptr(Npool);
Arc *adj[maxV],*preE[maxV];
Queue<int,maxV>q;
int dis[maxV],preV[maxV],mincost(0),maxflow(0);
bool v[maxV];
bool spfa()
{
  memset(dis,inf,sizeof(dis));
  memset(v,0,sizeof(v));
  while(!q.empty()) q.pop();
  dis[S]=0; v[S]=1; q.push(S);
  while(!q.empty())
  {
    int id=q.front(); q.pop(); v[id]=0;
    for(Arc *p=adj[id];p;p=p->next)
    {
      if(p->rest>0)
      {
        int temp=dis[id]+p->cost;
        if(dis[p->dest]>temp)
        {
          dis[p->dest]=temp;
          preV[p->dest]=id; preE[p->dest]=p;
          if(!v[p->dest]) v[p->dest]=1, q.push(p->dest);
        }}}}
  return dis[T]!=inf;
}
void aug()
{
  int tflow=inf;
  for(int i=T;i!=S;i=preV[i])
    upd_min(tflow,preE[i]->rest);
  for(int i=T;i!=S;i=preV[i])
     preE[i]->rest-=tflow, preE[i]->rev->rest+=tflow;
  maxflow+=tflow;
  mincost+=dis[T]*tflow;
  return ;
}
void MinCostMaxFlow(){while(spfa())aug();}
```

4 Dinic

```
inline void add_edge(int s,int t,int r)
{
  adj[s]=new (nextArc()) Arc(t,r,adj[s]);
  adj[t]=new (nextArc()) Arc(s,0,adj[t]);
  adj[s]->rev=adj[t];
```

```
  adj[t]->rev=adj[s];
}
bool bfs()
{
  memset(v,0,sizeof(v)); memset(dis,0xff,sizeof(dis));
  dis[S]=13; q.clear(); q.push(S);
  while(!q.empty())
  {
    int id=q.front(); q.pop();
    for(Arc *p=adj[id];p;p=p->next)
    {
      int j=p->dest;
      if(p->rest && dis[j]==-1)
      {
        dis[j]=dis[id]+1; q.push(j);
        if(j==T)return true; } } }
  return false;
}
int aug(int i,int m=inf)
{
  if(i==T)return m;
  int ret(0);
  for(Arc *&p=cur[i];p;p=p->next)
  {
    int j=p->dest;
    if(p->rest && !v[j] && dis[i]==dis[j]-1)
    {
      if(int a=aug(j,min(m-ret,p->rest)))
      {
        ret+=a;  p->rest-=a; p->rev->rest+=a;
        if(ret==m)return ret;
      }
    }
  }
  if(ret==0)dis[i]=-1;
  v[i]=1;
  return ret;
}
void MaxFlow()
{
  flow=0;
  while(bfs())
  {
    for(int i=0;i<maxV;i++)
      cur[i]=adj[i];
    flow+=aug(S);
  }
}
```

5 SAP

```
struct Arc
{
  int dest,rest;
  Arc *next,*rev;
  Arc(){};
  Arc(int dest,int rest,Arc *next):
    dest(dest),rest(rest),next(next){};
}Npool[maxE],*Nptr(Npool);

int N,M,S,T;
int maxflow;
int h[maxN],hv[maxN];
Arc* adj[maxN],cur[maxN];

int aug(int id,int m)
{
```

```
  if(id==T) return m;
  int tt=m;
  for(Arc* p=cur[id];p;p=p->next)
  {
    if(p->rest && h[p->ed]+1==h[id])
    {
      int d=aug(p->ed,min(tt,p->rest));
      p->rest-=d;p->rev->rest+=d;tt-=d;
      if(h[1]==N || tt==0) return m-tt;
    }
  }
  int minh=N;
  for(Arc* p=cur[id]=adj[id];p;p=p->next)
    if(h[p->ed]+1<minh)minh=h[p->ed]+1;
  if(!--vh[h[id]])h[1]=n;
  else ++vh[h[id]=minh];
  return m-tt;
}

inline void add_edge(int st,int ed,int r)
{
  adj[st]=new Arc(ed,r,adj[st]);
  adj[ed]=new Arc(st,0,adj[ed]);
  adj[st]->rev=adj[ed];
  adj[ed]->rev=adj[st];
}
```

6 HUNGARY

```
bool find(int id)
{
  for(Arc *p=adj[id];p;p=p->next)
  {
    int j=p->dest;
    if(!used[j])
    {
      used[j]=1;
      if(ilink[j]==-1 || find(ilink[j]))
      {
        ilink[j]=i;
        return true;
      } } }
  return false;
}
```

7 KM

```
int N,nx,ny;
int ilink[maxN],lx[maxN],ly[maxN],slack[maxN];
int vix[maxN],viy[maxN],w[maxN][maxN];

int dfs(int id)
{
  vix[id]=1;
  for(int i=nx;i<=ny;i++)
  {
    if(viy[i]) continue;
    int tt=lx[id]+ly[i]-w[id][i];
    if(tt==0)
    {
      viy[i]=1;
      if(ilink[i]==-1 || dfs(ilink[i]))
      {
        ilink[i]=id;
        return 1;
```

```
      }
    }
    else if(slack[i]>tt)
      slack[i]=tt;
  }
  return 0;
}
void KM()
{
  memset(ilink,-1,sizeof(ilink));
  memset(ly,0,sizeof(ly));
  for(int i=1,j;i<=N;i++)
    for(j=nx,lx[i]=-inf;j<=ny;j++)
      if(w[i][j]>lx[i])
        lx[i]=w[i][j];
  for(int x=1;x<=N;x++)
  {
    for(int i=nx;i<=ny;i++)
      slack[i]=inf;
    while(1)
    {
      memset(vix,0,sizeof(vix));
      memset(viy,0,sizeof(viy));
      if(dfs(x))break;
      int d=inf;
      for(int i=nx;i<=ny;i++)
        if(!viy[i] && d>slack[i])
          d=slack[i];
      for(int i=1;i<=N;i++)
        if(vix[i])
          lx[i]-=d;
      for(int i=nx;i<=ny;i++)
        if(viy[i]) ly[i]+=d;
        else  slack[i]-=d;
    }
  }
}
```

8 TARJAN

```
int low[maxN],dfn[maxN],Belong[maxN];
bool v[maxN];
void tarjan(int id)
{
  dfn[id]=low[id]=++ind; v[id]=true;  s.push(id);
  int tt;
  for(Arc* p=adj[id];p;p=p->next)
  {
    tt=p->dest;
    if(!dfn[tt])
    {
      tarjan(tt);
      if(low[tt]<low[id]) low[id]=low[tt];
    }
    else if(v[tt] && dfn[tt]<low[id]) low[id]=dfn[tt];
  }
  if(dfn[id]==low[id])
  {
    Bcnt++; tt=0;
    while(tt!=id)
    {
      tt=s.top(); s.pop(); v[tt]=false;
      Belong[tt]=Bcnt;
    }
  }
  return ;
```

```
}
void solve()
{
  for(int i=1;i<=N;i++) if(!dfn[i]) tarjan(i);
  return ;
}
```

9 Inverse element

```
void inverse()  // Mod 素数
{
  inv[0]=inv[1]=1;
  for(int i=2;i<Mod;i++)
    inv[i]= ((Mod-Mod/i) * inv[Mod%i]) %Mod;
}
```

10 RMQ

```
const int maxN=50000+13;
const int maxB=16;//log2(maxN)
int N,Q,a[maxN],amax[maxB][maxN];
void RMQinit()
{
  for(int i=1;i<=N;i++) amax[0][i]=a[i];
  for(int j=1;(1<<j)<=N;j++)
  for(int i=1;i+(1<<j)-1<=N;i++)
    amax[j][i]=max(amax[j-1][i],amax[j-1][i+(1<<(j-1))]);
}
int RMQmax(int l,int r)
{
  int m=(int)floor(log(r-l+1)/log(2.0));
  return max(amax[m][l],amax[m][r-(1<<m)+1]);
}
```

11 KMP

```
int fail[maxN],m;
char s[maxN];
void init()
{
  m=strlen(s);
  int crt=fail[0]=-1;
  for(int i=1;i<=m;i++)
  {
    while(crt>=0 && s[crt]!=s[i-1]) crt=fail[crt];
    fail[i]=++crt;
  }
}
int kmp(char* t)
{
  int n=strlen(t),ret=0;
  for(int i=0,j=0;i<n;i++)
  {
    while(j>=0 && t[i]!=s[j]) j=fail[j];
    if(++j==m)
    {
      ret++;
      j=fail[j];
    }
  }
  return ret;
}
```

12 Manacher

```
int L=strlen(s);
int N=2*L+2;
t[0]='$';t[1]='#';
for(int i=L-1;i>=0;i--)
  t[i*2+2]=s[i] , t[i*2+3]='#';
t[L*2+2]=0;
int ind=0;
for(int i=1;i<N;i++)
{
  if(i>=ind+p[ind]) p[i]=1;
  else p[i]=min(p[ind*2-i],p[ind]-i+ind);
  for(;t[i+p[i]]==t[i-p[i]];p[i]++);
  if(p[i]+i>ind+p[ind])ind=i;
}
int ans=0;
for(int i=2;i<=2*L;i++)
  upd_max(ans,p[i]);
printf("%d\n",ans-1);
```

13 Hash

```
unsigned int BKDRHash(char *str)
{
    unsigned int seed = 131; // 31 131 1313 13131 131313
etc..
    unsigned int hash = 0;
    while (*str) hash = hash * seed + (*str++);
    return (hash & 0x7FFFFFFF);
}
unsigned int APHash(char *str)
{
    unsigned int hash = 0;
    for (int I;i=0; *str; i++)
    {
       if ((i & 1) == 0) hash ^= ((hash << 7) ^ (*str++)
^ (hash >> 3));
       else hash ^= (~((hash << 11) ^ (*str++) ^ (hash >>
5)));
    }
    return (hash & 0x7FFFFFFF);
}
unsigned int DJBHash(char *str)
{
    unsigned int hash = 5381;
    while (*str) hash += (hash << 5) + (*str++);
    return (hash & 0x7FFFFFFF);
}
unsigned int JSHash(char *str)
{
    unsigned int hash = 1315423911;
    while (*str)  hash ^= ((hash << 5) + (*str++) +
(hash >> 2));
    return (hash & 0x7FFFFFFF);
}
```

14 Suffix array

```
int SA[maxL],R[maxL],tmp[maxL],freq[maxL],height[maxL];
int cmp_S(const void* a, const void* b){return
S[*(int*)a]-S[*(int*)b];}
void get_SA()
{
  int tie_n;
  for(int i=0;i<L;++i) SA[i]=i;
  qsort(SA,L,sizeof(int),cmp_S);
```

```
  tie_n=0;
  for(int i=0;i<L;++i)
  {
    int a=SA[i],pa=SA[i-1];
    if(!i || S[a]!=S[pa])R[a]=i+1;
    else R[a]=R[pa], ++tie_n;
  }
  for(int k=1;k<L && tie_n;k<<=1)
  {
    int p=0;
    for(int i=L-k;i<L;++i)tmp[p++]=i;
    for(int i=0;i<L;++i) if(SA[i]>=k) tmp[p++]=SA[i]-k;
    memset(freq, 0, (L+1)*sizeof(int));
    for(int i=0;i<L;++i)++freq[R[i]];
    for(int i=1;i<=L;++i)freq[i]+=freq[i-1];
    for(int i=L-1;i>=0;--i)SA[--freq[R[tmp[i]]]]=tmp[i];
    memcpy(tmp, R, L*sizeof(int));
    tie_n=0;
    for(int i=0;i<L;++i)
    {
      int a=SA[i], pa=SA[i-1];
      int b=a+k, pb=pa+k;
      if(!i || tmp[a]!=tmp[pa] || b>=L || pb>=L ||
tmp[b]!=tmp[pb]) R[a]=i+1;
      else R[a]=R[pa], ++tie_n;
    }
  }
}
void get_height()
{
  int h=0;
  for(int i=0;i<L;++i)
  {
    if(R[i]==1)continue;
    int pi=SA[R[i]-2];
    for(h>0?--h:0;S[i+h]==S[pi+h];++h);
    height[R[i]]=h;
  }
  height[1]=0;
}
```

15 AC-automation

```
struct Node
{
  bool hit;
  Node *child[26], *sfx;
  Node(): hit(0), sfx(0) { memset(child, 0,
sizeof(child)); }
  Node* get_child(int i)
  {
    Node* p;
    for(p=this;p!=root && !p->child[i];p=p->sfx);
    return p->child[i]?p->child[i]:root;
  }
}Npool[maxNode], *Nptr, *root;
void dict_insert(char* str)
{
  Node* p=root;

  for(;*str;++str)
  {
    int i=c2i(*str);
    if(!p->child[i]) p->child[i] = new (Nptr++) Node();
    p=p->child[i];
  }
```

```
  p->hit=true;
}
void dict_get_sfx()
{
  static queue<Node*> Q;
  while(!Q.empty())Q.pop(); root->sfx=root;
  for(int i=0;i<26;++i) if(root->child[i])
  {
    root->child[i]->sfx=root;
    Q.push(root->child[i]);
  }
  while(!Q.empty())
  {
    Node* p=Q.front(); Q.pop();
    p->hit|=p->sfx->hit;
    for(int i=0;i<26;++i) if(p->child[i])
    {
      p->child[i]->sfx=p->sfx->get_child(i);
      Q.push(p->child[i]);
    }
  }
}
```

16 Geometry

```
const double Eps=1e-6;
const double PI=acos(-1.0);
inline int sig(double x, double eps = Eps)
{  return x<-eps?-1:x>eps; }
inline double deg2rad(double d)
{  return d*PI/180.0; }
inline double rad2deg(double r)
{  return r*180.0/PI; }
```

```
struct Point
{
  double x,y;
  Point(){};
  Point(double x,double y):x(x),y(y){};
  Point operator + (const Point& b)
  {    return Point(x+b.x,y+b.y);  }
  Point operator - (const Point& b)
  {    return Point(x-b.x,y-b.y);  }
  Point operator * (const double& a)
  {    return Point(x*a,y*a);  }
  Point operator / (const double& a)
  {    return Point(x/a,y/a);  }
  double operator * (const Point& b)
  {    return x*b.x+y*b.y;  }
  double operator % (const Point& b)
  {    return x*b.y-y*b.x;  }
  double dis()
  {    return hypot(x,y);  }
  double dis2()
  {    return sqr(x)+sqr(y);  }
  double alpha()
  {    return atan2(y,x);  }
  double disTo(const Point& a)
  {
    double dx=x-a.x,dy=y-a.y;
    return hypot(dx,dy);
  }
  double alphaTo(const Point& a)
  {
    double dx=x-a.x,dy=y-a.y;
    return atan2(dy,dx);
  }
```

```
  Point rot90()
  {    return Point(y,-x);  }
  Point rot(double al)//radian measure---- counter-
clockwise
  {
    return Point(x*cos(al)-
y*sin(al),x*sin(al)+y*cos(al));
  }
};
const Point O=Point(0,0);
typedef vector<Point> vP;
//三角形的外心，重心
Point ccenter(Point p1,Point p2,Point p3)
{
  Point ret;
  ret=(p3/p3.dis())*((p3-p2)*(p1-p3))/((p2-p1)%(p3-p2));
  ret=p1+p2-ret;
  ret=ret/2;
  return ret;
}
//点到直线的距离
double disLP(Point p1,Point p2,Point q)
{
  return abs((p2-p1)%(q-p1))/(p2-p1).dis();
}

//点到线段的距离
double disSP(Point p1,Point p2,Point q)
{
  if(((p2-p1)*(q-p1))<Eps)return (q-p1).dis();
  if(((p1-p2)*(q-p2))<Eps)return (q-p2).dis();
    return disLP(p1,p2,q);
}
```

```
//线段与线段相交
bool crsSS(Point p1,Point p2,Point q1,Point q2)
{
  if (max(p1.x,p2.x)+Eps<min(q1.x, q2.x)) return false;
  if (max(q1.x,q2.x)+Eps<min(p1.x, p2.x)) return false;
  if (max(p1.y,p2.y)+Eps<min(q1.y, q2.y)) return false;
  if (max(q1.y,q2.y)+Eps<min(p1.y, p2.y)) return false;
  return sig((p2-p1)%(q1-p1))*sig((p2-p1)%(q2-p1))<Eps
     && sig((q2-q1)%(p1-q1))*sig((q2-q1)%(p2-q1))<Eps;
}
//线段与圆相交
bool crsCS(Point c,double r,Point p1,Point p2)
{
  return disSP(p1,p2,c)<r+Eps &&
     (r<(c-p1).dis()+Eps||r<(c-p2).dis()+Eps);
}
//圆与圆相交
bool crsCC(Point c1,double r1,Point c2,double r2)
{
  double dis=(c1-c2).dis();
  return dis<r1+r2+Eps && abs(r1-r2)<dis+Eps;
}
//点与直线的垂足
Point proj(Point p1,Point p2,Point q)
{
  return p1+((p2-p1)*((p2-p1)*((q-p1))/(p2-p1).dis2()));
}
//直线与直线平行
bool isLLP(Point p1,Point p2,Point q1,Point q2)
{
  return sig((q2-q1)%(p2-p1))==0;
}
```

```
//直线与直线的交点(先判平行）
Point isLL(Point p1,Point p2,Point q1,Point q2)
{
  double d=(q2-q1)%(p2-p1);
  return p1+((p2-p1)*((q2-q1)%(q1-p1))/d);
}
//直线与圆的交点 (按照 p1 的远近顺序排列)
vP isCL(Point c,double r,Point p1,Point p2)
{
  vP ret;
  ret.clear();
  double x=(p1-c)*(p2-p1);
  double y=(p2-p1).dis2();
  double d=x*x-y*((p1-c).dis2()-r*r);
  if(d<-Eps)return ret;
  if(d<0)d=0;
  Point q1=p1-((p2-p1)*(x/y));
  Point q2=p2-((p1)*(sqrt(d)/y));
  ret.push_back(q1);
  ret.push_back(q2);
  return ret;
}
//两圆的交点
vP isCC(Point c1,double r1,Point c2,double r2)
{
  vP ret;
  ret.clear();
  double x=(c1-c2).dis2();
  double y=((r1*r1-r2*r2)/x+1)/2;
  double d=r1*r1/x-y*y;
  if(d<-Eps)return ret;
  if(d<0)d=0;
  Point q1=c1+((c2-c1)*y);
  Point q2=((c2-c1)*sqrt(d)).rot90();
  ret.push_back(q1);
  ret.push_back(q2);
  return ret;
}
//点 P 与圆的切点
vP tanCP(Point c,double r,Point p)
{
  vP ret;ret.clear();
  double x=(p-c).dis2();
  double d=x-r*r;
  if(d<-Eps) return ret;
  if(d<0)d=0;
  Point q1=(p-c)*(r*r/x);
  Point q2=((p-c)*(-r*sqrt(d)/x)).rot90();
  ret.push_back(q1);
  ret.push_back(q2);
  return ret;
}

//凸包 逆时针
bool cmp_cH(const Point& a,const Point& b)
{
  if(a.x==b.x)return a.y<b.y;
  return a.x<b.x;
}
vP convexHull(vP ps)
{
  int n=ps.size(),k=0;
  if(n<=1)return ps;
  sort(ps.begin(),ps.end(),cmp_cH);
  vP qs;
  for(int i=0;i<n;qs[k++]=ps[i++])
```

```
    while(k>1 && (qs[k-1]-qs[k-1])%(ps[i]-qs[k-1])<Eps)k--;
  for(int i=n-2,t=k;i>=0;qs[k++]=ps[i--])
    while(k>t && (qs[k-1]-qs[k-2])%(ps[i]-qs[k-1])<Eps)k--;
  vP ret;
  for(int i=0;i<k-1;i++)
    ret.push_back(qs[i]);
  return ret;
}

//点在多边形内部判定
//内部 1       边上 0      外部 -1
int contains(vP ps,Point q)
{
  int n=ps.size();
  int ret=-1;
  for(int i=0;i<n;i++)
  {
    Point a=ps[i]-q,b=ps[(i+1)%n]-q;
    if(a.y>b.y) swap(a,b);
    if(a.y<Eps && b.y>Eps && a%b>Eps) ret=-ret;
    if(abs(a%b)<Eps && a*b<Eps) return 0;
  }
  return ret;
}

vP convexCut(vP ps,Point p1,Point p2)
{
  int n=ps.size();
  vP ret;
  for(int i=0;i<n;i++)
  {
    int d1=sig((p2-p1)%(ps[i]-p1));
    int d2=sig((p2-p1)%(ps[(i+1)%n]-p1));
    if(d1>=0) ret.push_back(ps[i]);

if(d1*d2<0)ret.push_back(isLL(p1,p2,ps[i],ps[(i+1)%n]));
  }
  return ret;
}

double convexDiameter(vP ps)
{
  int n=ps.size();
  int is=0,js=0;
  for(int i=1;i<n;i++)
  {
    if(ps[i].x>ps[is].x)is=i;
    if(ps[i].x<ps[is].x)js=i;
  }
  double maxd=(ps[is]-ps[js]).dis();
  int i=is,j=js;
  do
  {
    if((ps[(i+1)%n]-ps[i])%(ps[(j+1)%n]-ps[j])>=0)
      j=(j+1)%n;
    else
      i=(i+1)%n;
    upd_max(maxd,(ps[i]-ps[j]).dis());
  } while(i!=is || j!=js);
  return maxd;
}
```

```
//左侧区域
const int pi=acos((double)-1.0);
const double Eps=1e-6;
const int maxN=40000+13;
const int inf=10000;

inline int sig(double x)
{
  if(x<-Eps) return -1;
  else return x>Eps;
}

struct Point
{
  double x,y;
  Point(){};
  Point(double x,double y):x(x),y(y){};
  Point operator - (const Point &b){return Point(x-b.x,y-
b.y);}
  Point operator + (const Point &b){return
Point(x+b.x,y+b.y);}
  Point operator * (const double &b){return
Point(x*b,y*b);}
  Point operator / (const double &b){return
Point(x/b,y/b);}
  double operator * (const Point &b)
  {    return x*b.y-y*b.x;  }
  double operator % (const Point &b)
  {
    return x*b.x+y*b.y;
  }
  double ang()
  {
    double ret=atan2(y,x);
    if(sig(ret)<0)ret+=pi*2;
    return ret;
  }
};
inline double xmul(Point a,Point b,Point c)
{
  return (b-a)*(c-a);
}
struct Segment
{
  Point s,e;
  double angle;
  Segment(){}
  Segment(Point _s,Point _e)
  {
    s=_s;
    e=_e;
    angle=atan2(e.y-s.y,e.x-s.x);
  }
};
Point get_intersect(Segment s1,Segment s2)
{
  double u=xmul(s1.s,s1.e,s2.s);
  double v=xmul(s1.e,s1.s,s2.e);
  Point t;
  t.x=(s2.s.x*v+s2.e.x*u)/(u+v);
  t.y=(s2.s.y*v+s2.e.y*u)/(u+v);
  return t;
}
```

```
bool cmp(Segment a,Segment b)
{
  if(sig(a.angle-b.angle)==0) return
sig(xmul(a.s,a.e,b.s))<0;
  return sig(a.angle-b.angle)<0;
}
bool IsParallel(Segment P,Segment Q)
{
  return sig((P.e-P.s)*(Q.e-Q.s))==0;
}
Segment deq[maxN];
Point hull[maxN];

int HalfPlaneIntersect(Segment seg[],int n)
{
  sort(seg,seg+n,cmp);
  int temp=1;
  for(int i=1;i<n;i++)
    if(sig(seg[i].angle-seg[temp-1].angle)!=0)
      seg[temp++]=seg[i];
  n=temp;
  deq[0]=seg[0];
  deq[1]=seg[1];
  int front=0,tail=1;
  for(int i=2;i<n;i++)
  {
    if(IsParallel(deq[tail],deq[tail-1]) ||
IsParallel(deq[front],deq[front+1])) return 0;
    while(front<tail &&
sig(xmul(seg[i].s,seg[i].e,get_intersect(deq[tail],deq[t
ail-1])))<0)--tail;
    while(front<tail &&
sig(xmul(seg[i].s,seg[i].e,get_intersect(deq[front],deq[
front+1])))<0)++front;
    deq[++tail]=seg[i];
  }
  while(front<tail &&
xmul(deq[front].s,deq[front].e,get_intersect(deq[tail],d
eq[tail-1]))<-Eps)tail--;
  while(front<tail &&
xmul(deq[tail].s,deq[tail].e,get_intersect(deq[front],de
q[front+1]))<-Eps)front++;
  int ret=0;
  deq[++tail]=deq[front];
  for(int i=front;i<tail;i++)
hull[ret++]=get_intersect(deq[i],deq[i+1]);
  return ret;
}
```

18 BigInt

```
const int BI_n=100;
struct BigInt
{
  private:
    int a[BI_n];
    mutable int n;
    inline void trunc()const{for(--n;n>=0 && !a[n];--
n);++n;}
    void carry(int start=0)
    {
      int i, tmp=0;
      for(i=start;i<n || tmp;++i)
      {
        tmp+=a[i]; a[i]=tmp%10000; tmp/=10000;
```

```
    }
    if(i>=n) n=i;
  }
public:
  BigInt(int x=0)
  {
    memset(a, 0, sizeof(a));
    for(n=0;x;++n) a[n]=x%10000, x/=10000;
  }
  BigInt(const string& x)
  {
    memset(a, 0, sizeof(a)); n=0;
    int block=0, p10=1;
    for(int l=(int)x.size(), i=l-1;i>=0;--i)
    {
      block+=(x[i]-'0')*p10;  p10*=10;
      if(p10==10000)
      {
        a[n++]=block;
        block=0; p10=1;
      }
    }
    if(block) a[n++]=block;
  }
  void output()
  {
    printf("%d",a[n-1]);
    for(int i=n-2;i>=0;--i) printf("%04d",a[i]);
    printf("\n");
    return ;
  }
  friend ostream& operator<<(ostream& out, BigInt x)
  {
    if(x.n==0) return out<<"0";
    out<<x.a[x.n-1];
    for(int i=x.n-2;i>=0;--i)
      out.fill('0'), out.width(4), out<<x.a[i];
    return out;
  }

  friend BigInt operator+(const BigInt& x, const
BigInt& y)
  {
    BigInt ret;
    ret.n=max(x.n, y.n)+1;
    int tmp=0;
    for(int i=0;i<ret.n;++i)
    {
      if(i<x.n) tmp+=x.a[i];
      if(i<y.n) tmp+=y.a[i];
      ret.a[i]=tmp%10000;  tmp/=10000;
    }
    ret.trunc();
    return ret;
  }
  friend BigInt operator-(const BigInt& x, const
BigInt& y)
  {
    BigInt ret;  ret.n=x.n;
    int tmp=0, borrow=0;
    for(int i=0;i<x.n;++i)
    {
      tmp=x.a[i]-y.a[i]-borrow;
      if(tmp<0) ret.a[i]=tmp+10000, borrow=1;
      else     ret.a[i]=tmp,  borrow=0;
    }
```

```
    ret.trunc();
    return ret;
  }
  friend BigInt operator*(const BigInt& x, const
BigInt& y)
  {
    BigInt ret;
    ret.n=x.n+y.n;
    for(int i=0;i<x.n;++i)
      for(int j=0;j<y.n;++j)
      {
        ret.a[i+j]+=x.a[i]*y.a[j];
        if(ret.a[i+j]>100000000) ret.carry(i+j);
      }
    ret.carry(0);
    ret.trunc();
    return ret;
  }
}
```

19 Segment-tree

```
struct Node
{
  int st,ed;
  long long col;
  int label;
  Node *left,*right;
  Node(){};
  Node(int st,int
ed):st(st),ed(ed),label(0),left(NULL),right(NULL){};
  void update()
  {
    col=0ll;
    if(left)  col=left->col;
    if(right) col|=right->col;
    return ;
  }
  void downlabel()
  {
    if(left)  left->label=label, left->col=col;
    if(right) right->label=label, right->col=col;
    label=0;
  }
}Npool[maxNode],*Nptr,*root;
void Build(int st,int ed,Node *&p=root)
{
  p=new (Nptr++) Node(st,ed);
  p->col=2;
  if(ed-st==1)return ;
  int mid=(st+ed)>>1;
  Build(st,mid,p->left),Build(mid,ed,p->right);
  return ;
}
void Modify(int st,int ed,int c,Node *&p=root)
{
  if(p->label)p->downlabel();
  if(st==p->st && ed==p->ed)
  {
    p->label=c;
    p->col=(1ll)<<(c-1);
    return;
  }
  int mid=(p->st+p->ed)>>1;
  if(mid<=st) Modify(st,ed,c,p->right);
  else if(mid>=ed) Modify(st,ed,c,p->left);
```

```
  else
Modify(st,mid,c,p->left),Modify(mid,ed,c,p->right);
  p->update();
}

long long Query(int st,int ed,Node *&p=root)
{
  if(p->label) p->downlabel();
  if(p->st==st && p->ed==ed) return p->col;
  int mid=(p->st+p->ed)>>1;
  if(mid<=st) return Query(st,ed,p->right);
  else if(mid>=ed) return Query(st,ed,p->left);
  else return
Query(st,mid,p->left)|Query(mid,ed,p->right);
}
```

20 Segment-tree-2D

```
truct NodeY
{
  int st,ed;
  int imin,imax;
};
int locx[maxN],locy[maxN];
int loc[maxN];

struct NodeX
{
  int st,ed;
  NodeY node[maxNode];
  void Build(int id,int _st,int _ed)
  {
    node[id].st=_st;
    node[id].ed=_ed;
    node[id].imin=inf;
    node[id].imax=-inf;
    if(_ed-_st==1) return ;
    int mid=(_st+_ed)>>1;
    Build(id<<1,_st,mid);
    Build((id<<1)|1,mid,_ed);
  }
  int queryMin(int id,int _st,int _ed)
  {
    if(node[id].st==_st && node[id].ed==_ed)
      return node[id].imin;
    int mid=(node[id].st+node[id].ed)>>1;
    if(_ed<=mid) return queryMin(id<<1,_st,_ed);
    else if(_st>=mid) return queryMin((id<<1)|1,_st,_ed);
    else return
min(queryMin((id<<1),_st,mid),queryMin((id<<1)|1,mid,_ed
));
  }
  int queryMax(int id,int _st,int _ed)
  {
    if(node[id].st==_st && node[id].ed==_ed)
      return node[id].imax;
    int mid=(node[id].st+node[id].ed)>>1;
    if(_ed<=mid) return queryMax(id<<1,_st,_ed);
    else if(_st>=mid) return queryMax((id<<1)|1,_st,_ed);
    else return
max(queryMax((id<<1),_st,mid),queryMax((id<<1)|1,mid,_ed
));
  }
}nodex[maxNode];

int N;
```

```
void Build(int id,int st,int ed)
{
  nodex[id].st=st;
  nodex[id].ed=ed;
  nodex[id].Build(1,1,N+1);
  if(ed-st==1) return ;
  int mid=(st+ed)>>1;
  Build(id<<1,st,mid);
  Build((id<<1)|1,mid,ed);
  return ;
}

void Modify(int x,int y,int val)
{
  int tx=locx[x];
  int ty=locy[y];
  nodex[tx].node[ty].imin=val;
  nodex[tx].node[ty].imax=val;
  for(int i=tx;i;i>>=1)
    for(int j=ty;j;j>>=1)
    {
      if(i==tx && j==ty) continue;
      if(j==ty)
      {

nodex[i].node[j].imin=min(nodex[i<<1].node[j].imin,nodex
[(i<<1)|1].node[j].imin);

nodex[i].node[j].imax=max(nodex[i<<1].node[j].imax,nodex
[(i<<1)|1].node[j].imax);
      }
      else
      {

nodex[i].node[j].imin=min(nodex[i].node[j<<1].imin,nodex
[i].node[(j<<1)|1].imin);

nodex[i].node[j].imax=max(nodex[i].node[j<<1].imax,nodex
[i].node[(j<<1)|1].imax);
      }
    }
}

int queryMin(int id,int x1,int x2,int y1,int y2)
{
  if(nodex[id].st==x1 && nodex[id].ed==x2)
    return nodex[id].queryMin(1,y1,y2);
  int mid=(nodex[id].st+nodex[id].ed)>>1;
  if(x2<=mid) return queryMin(id<<1,x1,x2,y1,y2);
  else if(mid<=x1) return
queryMin((id<<1)|1,x1,x2,y1,y2);
  else return
min(queryMin(id<<1,x1,mid,y1,y2),queryMin((id<<1)|1,mid,
x2,y1,y2));
}

int queryMax(int id,int x1,int x2,int y1,int y2)
{
  if(nodex[id].st==x1 && nodex[id].ed==x2)
    return nodex[id].queryMax(1,y1,y2);
  int mid=(nodex[id].st+nodex[id].ed)>>1;
  if(x2<=mid) return queryMax(id<<1,x1,x2,y1,y2);
  else if(mid<=x1) return
queryMax((id<<1)|1,x1,x2,y1,y2);
```

```
  else return
max(queryMax(id<<1,x1,mid,y1,y2),queryMax((id<<1)|1,mid,
x2,y1,y2));
}

void init(int id,int st,int ed)
{
  if(ed-st==1)
  {
    loc[st]=locx[st]=locy[st]=id;
    return ;
  }
  int mid=(st+ed)>>1;
  init(id<<1,st,mid);
  init((id<<1)|1,mid,ed);
}

int main()
{
   init(1,1,N+1);
   Build(1,1,N+1);
}
```

21 Dancing Links

```
const int maxN=100+13;
const int maxNode=10000+13;
const int inf=0x7f7f7f7f;

int K;

struct DLX
{
  int n,m,ind;
  int
U[maxNode],D[maxNode],R[maxNode],L[maxNode],row[maxNode]
,col[maxNode];
  int Head[maxN],Size[maxN];//注意大小，一个是行，一个是列
  int ansd;
  void init(int a,int b)
  {
    n=a,m=b;
    ind=m;
    ansd=inf;
    for(int i=0;i<=m;i++)
    {
      Size[i]=0;
      U[i]=D[i]=i;
      L[i]=i-1; R[i]=i+1;
    }
    R[m]=0; L[0]=m;
    for(int i=1;i<=n;i++) Head[i]=-1;
    return ;
  }
  void link(int r,int c)
  {
    col[++ind]=c;
    ++Size[c];
    row[ind]=r;
    D[ind]=D[c]; U[D[c]]=ind;
    U[ind]=c; D[c]=ind;
    if(Head[r]<0) Head[r]=L[ind]=R[ind]=ind;
    else
    {
      R[ind]=R[Head[r]];
      L[R[Head[r]]]=ind;
```

```
    L[ind]=Head[r];
    R[Head[r]]=ind;
  }
}
void remove(int c)
{
  for(int i=D[c];i!=c;i=D[i])
    L[R[i]]=L[i],R[L[i]]=R[i];
}
void resume(int c)
{
  for(int i=U[c];i!=c;i=U[i])
    L[R[i]]=R[L[i]]=i;
}
bool v[maxNode];
int cal()
{
  int ret=0;
  for(int i=R[0];i!=0;i=R[i]) v[i]=true;
  for(int i=R[0];i!=0;i=R[i]) if(v[i])
  {
    v[i]=false;
    ret++;
    for(int j=D[i];j!=i;j=D[j])
      for(int k=R[j];k!=j;k=R[k])
        v[col[k]]=false;
  }
  return ret;
}
bool dance(int d)
{
  if(d+cal()>ansd) return false;
  if(R[0]==0) return d<=K;
    int c=R[0];
    for(int i=R[0];i!=0;i=R[i])
      if(Size[i]<Size[c]) c=i;
    for(int i=D[c];i!=c;i=D[i])
    {
      remove(i);
      for(int j=R[i];j!=i;j=R[j]) remove(j);
      if(dance(d+1))return true;
      for(int j=L[i];j!=i;j=L[j]) resume(j);
      resume(i);
    }
    return false;
  }
};
xx.init(N,N);
xx.ansd=K;
xx.link(i,j);
xx.dance(0);
```

22 Simulated annealing

```
double simulated_annealing()
{
  double x=0,y=0,z=func(x,y);
  double step=1.0,rate=0.99;
  while(step>eps)
  {
    for(int k=0;k<8;k++)
    {
      double tx,ty,tz;
      tx=x+step*dx[k];
      ty=y+step*dy[k];
      tz=func(tx,ty);
```

```
      if(tz>1e30)continue;
      if(dis(tx,ty,tz)<dis(x,y,z))
      {
        x=tx;y=ty;z=tz;
      }
    }
    step*=rate;
  }
  return dis(x,y,z);
}
```

23 Lucas

```
long long f[maxN];
void init(long long p)
{
  f[0]=1;
  for(int i=1;i<=p;i++)
    f[i]=f[i-1]*i%p;
}
long long inv(long long a,long long m)
{
  if(a==1) return 1;
  return inv(m%a,m)*(m-m/a)%m;
}
long long Lucas(long long n,long long m,long long p)
{
  long long ans=1;
  while(n&&m)
  {
    long long a=n%p;
    long long b=m%p;
    if(a<b) return 0;
    ans=ans*f[a]%p*inv(f[b]*f[a-b]%p,p)%p;
    n/=p;
    m/=p;
  }
  return ans;
}
```

24 Heavy Light Decomposition of Tree

```
int num[maxN],top[maxN],son[maxN];
int dep[maxN],p[maxN],fp[maxN],fa[maxN];
//p--在树上的位置，fp--p 的反函数
void dfs1(int id,int deep,int pre)
{
  dep[id]=deep;   fa[id]=pre;
  num[id]=1;
  for(Arc* p=adj[id];p;p=p->next)
  {
    int j=p->dest;
    if(j!=pre)
    {
      dfs1(j,deep+1,id);
      num[id]+=num[j];
      if(son[id]==-1 || num[id]>num[son[id]])  son[id]=j;
    }
  }
  return ;
}
int ind=0;
void dfs2(int id,int sp)
{
  top[id]=sp;
  p[id]=ind++;
  fp[p[id]]=id;
```

```
  if(son[id]==-1) return ;
  dfs2(son[id],sp);
  for(Arc* p=adj[id];p;p=p->next)
  {
    int j=p->dest;
    if(j!=son[id] && j!=fa[id]) dfs2(j,j);
  }
  return ;
}

void change(int st,int ed,int tn)
{
  int f1=top[st],f2=top[ed];
  while(f1!=f2)
  {
    if(dep[f1]<dep[f2])
    {
      swap(st,ed);  swap(f1,f2);
    }
    add(p[f1],tn);  add(p[st]+1,-tn);
    st=fa[f1];
    f1=top[st];
  }
  if(dep[st]>dep[ed]) swap(st,ed);
  add(p[st],tn); add(p[ed]+1,-tn);
}

void init()
{
  ind=1;
  memset(son,-1,sizeof(son));
  return ;
}
```

25 Mo

```
struct Query
{
  int st,ed,id;
  Query(){};
}q[maxN];
bool cmp(Query a,Query b)
{
  if(a.st/unit != b.st/unit)
    return a.st/unit < b.st/unit;
  return a.ed<b.ed;
}
void bfs()
{
  long long temp=0;
  memset(n,0,sizeof(n));
  int L=1,R=0;
  for(int i=1;i<=M;i++)
  {
    while(R<q[i].ed)
    {
      R++;
      temp-=(long long)n[a[R]]*n[a[R]];
      n[a[R]]++;
      temp+=(long long)n[a[R]]*n[a[R]];
    }
    while(R>q[i].ed)
    {
      temp-=(long long)n[a[R]]*n[a[R]];
      n[a[R]]--;
      temp+=(long long)n[a[R]]*n[a[R]];
      R--;
    }
```

```
    while(L<q[i].st)
    {
      temp-=(long long)n[a[L]]*n[a[L]];
      n[a[L]]--;
      temp+=(long long)n[a[L]]*n[a[L]];
      L++;
    }
    while(L>q[i].st)
    {
      L--;
      temp-=(long long)n[a[L]]*n[a[L]];
      n[a[L]]++;
      temp+=(long long)n[a[L]]*n[a[L]];
    }
    ans[q[i].id].a=temp-(R-L+1);
    ans[q[i].id].b=(long long)(R-L+1)*(R-L);
    ans[q[i].id].simple();
  }
}
```

26 Locale

isspace	Check if character is a white-space
isprint	Check if character is printable
iscntrl	Check if character is a control character
isupper	Check if character is uppercase letter
islower	Check if character is lowercase letter
isalpha	Check if character is alphabetic
isdigit	Check if character is decimal digit
ispunct	Check if character is a punctuation character
isxdigit	Check if character is hexadecimal digit
isalnum	Check if character is alphanumeric
isgraph	Check if character has graphical representation
isblank	Check if character is blank

27 Math

cbrt(x)	Returns the *cubic root* of *x*.
hypot(x,y)	Compute hypotenuse—(x^2+y^2)
ceil	Round up value
floor	Round down value
round	Round to nearest
lround	Round to nearest and cast to long integer
llround	Round to nearest and cast to long long integer
rint	Round to integral value
lrint	Round and cast to long integer
llrint	Round and cast to long long integer
nearbyint	Round to nearby integral value
trunc	Rounds *x* toward zero, returning the nearest integral value that is not larger in magnitude than *x*.