1.bell_manford.cpp

```
#include <cstdio>

#define MAXN 200
#define inf  1000000000
typedef int  elem_i;

//最短路径 (单源 bellman_ford 邻接阵)
//单源最短路径,bellman_ford 算法,邻接阵形式,复杂度 O(n^3)
//求出源 s 到所有点的最短路经,传入图的大小 n 和邻接阵 mat
//返回到各点最短距离 min[]和路径 pre[],pre[i]记录 s 到 i 路径上 i 的父结点,pre[s]=-1
//可更改路权类型,路权可为负,若图包含负环则求解失败,返回 0
//优化:先删去负边使用 dijkstra 求出上界,加速迭代过程
int bellman_ford(int n,elem_i mat[][MAXN],int s,elem_i *min,int *pre)
{
  int v[MAXN],i,j,k,tag;
```

```
    for(i=0; i<n; i++)  min[i]=inf,v[i]=0,pre[i]=-1;
    for(min[s]=0,j=0; j<n; j++)     //dijkstra
    {
        for(k=-1,i=0; i<n; i++)
            if(!v[i] && (k==-1||min[i]<min[k]))
                k=i;
        for(v[k]=1,i=0; i<n; i++)
            if(!v[i] && mat[k][i]>=0 && min[k]+mat[k][i]<min[i])
                min[i]=min[k]+mat[pre[i]=k][i];
    }
    for(tag=1,j=0; tag&&j<=n; j++)
        for(tag=i=0; i<n; i++)
            for(k=0; k<n; k++)
                if(min[k]+mat[k][i]<min[i])
                    min[i]=min[k]+mat[pre[i]=k][i],tag=1;
    return j<=n;
}

//bellman_ford 邻接表 O(m*n)       ac poj 3259
//可自行添加 pre 数组记录前一个点
#include <cstdio>
#define inf 1000000
#define MAXS 30000
#define MAXN 500
typedef int elem_t;

struct edge{ int f,t,n; elem_t l; }e[MAXS];
int list[MAXN],top;
elem_t min[MAXN];

void insert(int f,int t,elem_t l)
{
    e[top].f=f,e[top].t=t,e[top].l=l;
    e[top].n=list[f],list[f]=top++;
}

int bellman_ford(int n,int s)
{
    int i,tag,j;
    for(i=0; i<n; i++)      min[i]=inf;
    for(min[s]=i=0,tag=1; i<=n&&tag; i++)
    {
        for(tag=0,j=0; j<top; j++)
            if(min[e[j].f]+e[j].l<min[e[j].t])
                min[e[j].t]=min[e[j].f]+e[j].l,tag=1;
    }
    return i<=n;
}
```

2.big_integer.cpp

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

#define DIGIT 4
#define DEPTH 10000
```

```
#define MAXN  2502
#define SGN(x) ((x)>0?1:((x)<0?-1:0))
#define ABS(x) ((x)>0?(x):-(x))
typedef int big_num[MAXN+1];

int read(big_num a)
{
   char buf[MAXN*DIGIT+1],ch; int i,j;
   memset(a,0,sizeof(int)*(MAXN+1));
   if( scanf("%s",&buf)==EOF ) return 0;
   for(a[0]=strlen(buf),i=a[0]/2-1; i>=0; i--)
       ch=buf[i],buf[i]=buf[a[0]-1-i],buf[a[0]-1-i]=ch;
   for(j=a[0],a[0]=(a[0]+DIGIT-1)/DIGIT; j<a[0]*DIGIT; buf[j++]='0');
   for(i=1; i<=a[0]; i++)
       for(a[i]=0,j=0; j<DIGIT; j++)
           a[i]=a[i]*10+buf[i*DIGIT-1-j]-'0';
   for(;!a[a[0]]&&a[0]>1;a[0]--);
   return 1;
}

int read(big_num a,int &sgn)                    //read big_num a which
is lower than zero
{
   char str[MAXN*DIGIT+1],ch,*buf; int i,j;
   memset(a,0,sizeof(int)*(MAXN+1));
   if( scanf("%s",&str)==EOF ) return 0;
   buf=str,sgn=1;
   if(*buf=='-')   sgn=-1,buf++;
   for(a[0]=strlen(buf),i=a[0]/2-1; i>=0; i--)
       ch=buf[i],buf[i]=buf[a[0]-1-i],buf[a[0]-1-i]=ch;
   for(j=a[0],a[0]=(a[0]+DIGIT-1)/DIGIT; j<a[0]*DIGIT; buf[j++]='0');
   for(i=1; i<=a[0]; i++)
       for(a[i]=0,j=0; j<DIGIT; j++)
           a[i]=a[i]*10+buf[i*DIGIT-1-j]-'0';
   for(;!a[a[0]]&&a[0]>1;a[0]--);
   if(a[0]==1 && !a[1])    sgn=0;
   return 1;
}

void write(const big_num a)
{
   int i,j;
   for(printf("%d",a[i=a[0]]),i--; i; i--)
       for(j=DEPTH/10; j; j/=10)
           printf("%d",a[i]/j%10);
}

int comp(big_num a,big_num b)
{
   int i;
   if(a[0]!=b[0])  return a[0]-b[0];
   for(i=a[0]; i; i--)
       if(a[i]!=b[i])
           return a[i]-b[i];
   return 0;
}
```

```
int comp(big_num a,const int b)
{
  int c[12]={1};
  for(c[1]=b; c[c[0]]>=DEPTH; c[c[0]+1]=c[c[0]]/DEPTH,c[c[0]]%=DEPTH,c[0]++);
  return comp(a,c);
}

int comp(big_num a,int c,int d,big_num b)      //compete a*c before
dth bit with b...  return a*c>b;
{
  int i,t=0,O=-DEPTH*2;
  if(b[0]-a[0]<d && c)
      return 1;
  for(i=b[0]; i>d; i--)
  {
      t=t*DEPTH+a[i-d]*c-b[i];
      if(t>0) return 1;
      if(t<O) return 0;
  }
  for(i=d; i; i--)
  {
      t=t*DEPTH-b[i];
      if(t>0) return 1;
      if(t<O) return 0;
  }
  return t>0;
}

void add(big_num a,big_num b)
{
  int i;
  for(i=1; i<=b[0]; i++)
      if((a[i]+=b[i])>=DEPTH)
          a[i]-=DEPTH,a[i+1]++;
  if(b[0]>=a[0])      a[0]=b[0];
  else                for(; a[i]>=DEPTH && i<=a[0]; a[i]-=DEPTH,i++,a[i]++ );
  a[0]+=(a[a[0]+1]>0);
}

void add(big_num a,int b)
{
  int i=1;
  for(a[1]+=b; a[i]>=DEPTH && i<a[0]; a[i+1]+=a[i]/DEPTH,a[i]%=DEPTH,i++);
  for(; a[a[0]]>=DEPTH; a[a[0]+1]=a[a[0]]/DEPTH,a[a[0]]%=DEPTH,a[0]++);
}

void sub(big_num a,big_num b)   //a>b;
{
  int i;
  for(i=1; i<=b[0]; i++)
      if((a[i]-=b[i])<0)
          a[i+1]--,a[i]+=DEPTH;
  for(; a[i]<0; a[i]+=DEPTH,i++,a[i]--);
  for(; !a[a[0]]&&a[0]>1; a[0]--);
}
```

```
void sub(big_num a,int b)
{
   int i=1;
   for(a[1]-=b; a[i]<0;
a[i+1]+=(a[i]-DEPTH+1)/DEPTH,a[i]-=(a[i]-DEPTH+1)/DEPTH*DEPTH,i++);
   for(; !a[a[0]]&&a[0]>1; a[0]--);
}

void sub(big_num a,big_num b,int c,int d)       //a-=b*c...before dth bit of a..
{
   int i,O=b[0]+d;
   for(i=1+d; i<=O; i++)
       if((a[i]-=b[i-d]*c)<0)
           a[i+1]+=(a[i]-DEPTH+1)/DEPTH,a[i]-=(a[i]-DEPTH+1)/DEPTH*DEPTH;
   for(; a[i]<0;
a[i+1]+=(a[i]-DEPTH+1)/DEPTH,a[i]-=(a[i]-DEPTH+1)/DEPTH*DEPTH,i++);
   for(; !a[a[0]]&&a[0]>1; a[0]--);
}

void mul(big_num a,big_num b,big_num c)
{
   int i,j;
   memset(c,0,sizeof(int)*(MAXN+1));
   for(c[0]=a[0]+b[0]-1,i=1; i<=a[0]; i++)
       for(j=1;j<=b[0];j++)
           if((c[i+j-1]+=a[i]*b[j])>=DEPTH)
               c[i+j]+=c[i+j-1]/DEPTH,c[i+j-1]%=DEPTH;
   for(c[0]+=(c[c[0]+1]>0); !c[c[0]]&&c[0]>1; c[0]--);
}

void mul(big_num a,const int b)     // AC
{
   int i;
   for(a[1]*=b,i=2; i<=a[0]; i++)
   {
       a[i]*=b;
       if(a[i-1]>=DEPTH)
           a[i]+=a[i-1]/DEPTH,a[i-1]%=DEPTH;
   }
   for(; a[a[0]]>=DEPTH; a[a[0]+1]=a[a[0]]/DEPTH,a[a[0]]%=DEPTH,a[0]++);
   for(; !a[a[0]] && a[0]>1; a[0]--);
}

void div(big_num c,big_num a,big_num b)
{
   int h,l,m,i;
   memset(c,0,sizeof(int)*(MAXN+1));
   c[0]=(b[0]<a[0]+1)?(a[0]-b[0]+2):1;
   for(i=c[0]; i; sub(a,b,c[i]=m,i-1),i--)
       for(h=DEPTH-1,l=0,m=(h+l+1)>>1; h>l; m=(h+l+1)>>1)
           if( comp(b,m,i-1,a) )   h=m-1;
           else                    l=m;
   for(; !c[c[0]]&&c[0]>1; c[0]--);
   c[0]=c[0]>1?c[0]:1;
}
```

```
void div(big_num a,const int b,int& c)      //c=a%b;    a=a/b;
{
   int i;
   for(c=0,i=a[0]; i; c=c*DEPTH+a[i],a[i]=c/b,c%=b,i--);
   for(; !a[a[0]]&&a[0]>1; a[0]--);
}

void sqrt(big_num b,big_num a)      //b=sqrt(a)
{
   int h,l,m,i;
   memset(b,0,sizeof(int)*(MAXN+1));
   for(i=b[0]=(a[0]+1)>>1; i; sub(a,b,m,i-1),b[i]+=m,i--)
       for(h=DEPTH-1,l=0,b[i]=m=(h+l+1)>>1; h>l; b[i]=m=(h+l+1)>>1)
           if(comp(b,m,i-1,a)) h=m-1;
           else                l=m;
   for(; !b[b[0]]&&b[0]>1; b[0]--);
   for(i=1; i<=b[0]; b[i++]>>=1);
}

int length(big_num a)
{
   int t,ret;
   for(ret=(a[0]-1)*DIGIT,t=a[a[0]]; t; t/=10,ret++);
   return ret>0?ret:1;
}

int digit(big_num a,int b)      //return bth digit of a;
{
   int i,ret;
   for(ret=a[(b-1)/DIGIT+1],i=(b-1)%DIGIT; i; ret/=10,i--);
   return ret%10;
}

int zeronum(big_num a)          //return the first one digit wihch is zero...
{
   int ret,t;
   for(ret=0; !a[ret+1]; ret++);
   for(t=a[ret+1],ret*=DIGIT; !(t%10); t/=10,ret++);
   return ret;
}

void comp(int t[],int l,int h,int d)        //divide l*(l+1)*...*(h)
to t[]...; d=1 or -1
{
   int i,j,tmp;
   for(i=l; i<=h; i++)
       for(tmp=i,j=2; tmp>1; j++)
           while(!(tmp%j))
               t[j]+=d,tmp/=j;
}

void convert(int t[],int h,big_num a)       //conver C(m,n) or A(m,n)
 from t[]    to ret;
{
   int i,j,tmp=1;
   memset(a,0,sizeof(int)*(MAXN+1));
   for(a[0]=a[1]=1,i=2; i<=h; i++)
       if(t[i])
           for(j=t[i]; j; tmp*=i,j--)
               if(tmp*i>DEPTH)
                   mul(a,tmp),tmp=1;
   mul(a,tmp);
}

const int MMM=10000;
void combination(big_num a,int m,int n)     //return a=C(m,n)
{
   int t[MMM]={0};
   comp(t,n+1,m,1);
   comp(t,2,m-n,-1);
 //  for(int i=0;i<m;i++)    printf("%d ",t[i]);
 //  printf("\n");
   convert(t,m,a);
}

void permutation(big_num a,int m,int n) //return a=A(m,n)
{
   int i,t=1;
   memset(a,0,sizeof(int)*(MAXN+1));
   a[0]=a[1]=1;
   for(i=m-n+1; i<=m; t*=i++)
       if(t*i>DEPTH)
           mul(a,t),t=1;
   mul(a,t);
}
```

3.cmp_geometry.cpp

```
//求两直线交点，面积法
//须先判 u1u2 与 v1v2 是否平行
#include <cstdio>
#include <cmath>

typedef double elem_t;
struct point{ elem_t x,y; };

inline elem_t xmult(point a,point b,point c,point d)
{
       return (b.x-a.x)*(d.y-c.y)-(b.y-a.y)*(d.x-c.x);
}
point intersection(point a,point b,point c,point d)
{
       point ret=a;
       elem_t up,dw,t;
       up=xmult(c,d,c,a);
       dw=xmult(a,b,c,d);
       t=up/dw;
       ret.x+=(b.x-a.x)*t;
       ret.y+=(b.y-a.y)*t;
       return ret;
}
```

```
//判断点与任意多边形的关系
#include <cstdio>
#include <cmath>
#define MAXN 10001
#define eps 1e-8
typedef double elem_t;

struct point{ elem_t x,y; }p[MAXN];

inline int min(elem_t a,elem_t b){ return a<b?a:b; }
inline int max(elem_t a,elem_t b){ return a<b?b:a; }
inline double xmult(point a,point b,point c,point d)
{
    return (b.x-a.x)*(d.y-c.y)-(b.y-a.y)*(d.x-c.x);
}
inline int zero(double x){ return x>=-eps&&x<=eps;
}

int in_polygon(point a,int n,point p[],int on_edge)
{
    point low_p,high_p;
    int i,cnt=0;
    for(i=0; i<n; i++)
    {
        if(p[i].y<p[(i+1)%n].y) low_p=p[i],high_p=p[(i+1)%n];
        else    high_p=p[i],low_p=p[(i+1)%n];
        if(xmult(a,low_p,a,high_p)>eps&&a.y>=low_p.y&&a.y<high_p.y)
            cnt++;
        else
if(zero(xmult(a,low_p,a,high_p))&&a.y>=low_p.y&&a.y<=high_p.y&&a.x>=min(low_p.x,high_p.x)&&a.x<=max(low_p.
x,high_p.x))
            return on_edge;
    }
    return cnt&1;
}
```

4.comb2num.cpp

```
//字典序组合与序号的转换  求 c 是所有组合中的第 ret 个,第一个组合 ret 为 1.
//c={1,.2,3...n}
#include <cstdio>
typedef long long LL;

LL comb(int n,int m)
{
    LL ret=1; int i=n;
    for(; i>n-m; i--)       ret*=i,ret/=n-i+1;
    return ret;
}

LL comb2num(int n,int m,int c[])
{
    LL ret=comb(n,m); int i,j;
    for(i=0; i<m; i++)
```

```
        ret-=comb(n-c[i],m-i);
    return ret;
}
```

5.converx_hull.cpp

```
#include <cstdio>
#include <cmath>
#include <cstring>
#include <algorithm>
#define eps 1e-8
#define inf 100000000
#define MAXN 50100
using std::sort;
typedef double elem_t;

struct point{ elem_t x,y; }p1,p[MAXN],converx[MAXN];
inline elem_t max(elem_t a,elem_t b){ return a>b?a:b; }
inline int zero(elem_t x){ return x>=-eps&&x<=eps; }

inline elem_t len(point a,point b)
{
    return sqrt((b.x-a.x)*(b.x-a.x)+(b.y-a.y)*(b.y-a.y));
}
inline elem_t xmult(point a,point b,point c)
{
    return (a.x-c.x)*(b.y-c.y)-(a.y-c.y)*(b.x-c.x);
}

int cmp(point a,point b)
{
    elem_t x;
    x=xmult(a,b,p1);
    if(zero(x))
        return len(a,p1)<len(b,p1);
    return x>0;
}

int _converx(int n,point p[],point con[])
{
    int i,s;
    p1.x=p1.y=inf;
    for(i=0; i<n; i++)
    {
        if(p[i].y<p1.y)
            p1=p[i];
        else if(p[i].y==p1.y&&p[i].x<p1.x)
            p1=p[i];
    }
    sort(p,p+n,cmp);
    con[0]=p[0],con[1]=p[1],s=2;
    for(i=0; i<n; con[s++]=p[i++])
    for(; s>1&&xmult(con[s-1],p[i],con[s-2])<=eps; s--);
    //改为<-eps 可保留共线点
    return s;
}
```

6.date.cpp

```
//日期模板 判断合法性/ 比较大小/ 判闰年/ 判星期几/ 日期转天数/ 天数转日期/
#include <cstdio>

int days[12]={31,28,31,30,31,30,31,31,30,31,30,31};
struct date{ int y,m,d; };

//判闰年
inline int leap(int year)
{
  return (year%4==0 && year%100!=0)||0==year%400;
}

//判合法性
inline int legal(date a)
{
  if(a.m<0 || a.m>12) return 0;
  if(a.m==2)      return a.d>0 && a.d<=28+leap(a.y);
  return a.d>0 && a.d<=days[a.m-1];
}

//比较日期大小
inline int datecmp(date a,date b)
{
  if(a.y != b.y)  return a.y-b.y;
  if(a.m != b.m)  return a.m-b.m;
  return a.d-b.d;
}

//返回指定日期是星期几
//蔡勒公式  适合于 1582 年 10 月 15 日之后的情形
//0 表示星期日
int weekday(date a)
{
  int tm=a.m>=3? (a.m-2):(a.m+10);
  int ty=a.m>=3? a.y:(a.y-1);
  return (ty+ty/4-ty/100+ty/400+(int)(2.6*tm-0.2)+a.d)%7;
}

//日期转天数偏移
//1 1 1 认为是第 366 天
int date2int(date a)
{
  int ret=a.y*365+(a.y-1)/4-(a.y-1)/100+(a.y-1)/400,i;
  days[1]+=leap(a.y);
  for(i=0; i<a.m-1; ret+=days[i++]);
  days[1]=28;
  return ret+a.d;
}

//天数偏移转日期
//400 年=146097 天
//1 1 1 认为是第 366 天  400 1 1 则第 146097 天
date int2date(int a)
{
  date ret;
  ret.y=a/146097*400;
  for(a%=146097; a>=365+leap(ret.y); a-=365+leap(ret.y),ret.y++);
  days[1]+=leap(ret.y);
  for(ret.m=1; a>=days[ret.m-1]; a-=days[ret.m-1],ret.m++);
  days[1]=28;
  ret.d=a+1;
  return ret;
}
```

7.dijkstra.cpp

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

const int MAXN=1001;
const int inf=1000000000;
typedef int elem_i;
int min1[MAXN],min2[MAXN],mat[MAXN][MAXN];
//最短路径(单源 dijkstra 邻接阵)
//单源最短路径,dijkstra 算法,邻接阵形式,复杂度 O(n^2)
//求出源 s 到所有点的最短路经,传入图的顶点数 n,(有向)邻接矩阵 mat
//返回到各点最短距离 min[]和路径 pre[],pre[i]记录 s 到 i 路径上 i 的父结点,pre[s]=-1
//可更改路权类型,但必须非负!
void dijkstra(int n,elem_i mat[][MAXN],int s,elem_i min[],int pre[])
 //pre[] to record the path
{
      int v[MAXN],i,j,k;
      for(i=0;i<n;i++)        v[i]=0,min[i]=inf,pre[i]=-1;                       //pre[i]
records: pre[i] -> i ;
      for(min[s]=0,i=0;i<n;i++){
      for(k=-1,j=0;j<n;j++)
      if (!v[j]&&((k==-1)||min[j]<min[k]))
             k=j;
      for(v[k]=1,j=0;j<n;j++)
      if(!v[j] && min[k]+mat[k][j]<min[j])
             min[j]=min[k]+mat[pre[j]=k][j];
      }
}

//dijkstra 优先队列
#include <cstdio>
#include <queue>
#define MAXN 1000
#define MAXS 2000000
#define inf 100000000
using namespace std;
typedef int elem_t;
struct edge{ int f,t,n; elem_t l; }e[MAXS];
struct heap_t
{
      int v; elem_t l;
      friend bool operator<(heap_t a,heap_t b){ return a.l>b.l; }
}h_t;
int list[MAXN],top=0,v[MAXN],pre[MAXN];
```

```
elem_t min_l[MAXN];
priority_queue<heap_t> q;

void insert(int f,int t,elem_t l)
{
    e[top].f=f,e[top].t=t,e[top].l=l;
    e[top].n=list[f],list[f]=top++;
}

void dijkstra(int n,int s)
{
    int i;
    for(i=0; i<n; i++)      v[i]=0,min_l[i]=inf,pre[i]=-1;
    min_l[h_t.v=s]=h_t.l=0; q.push(h_t);
    for(; !q.empty();)
    {
        h_t=q.top(),q.pop();
        if(!v[h_t.v])
            for(v[h_t.v]=1,i=list[h_t.v]; i!=-1; i=e[i].n)
                if(!v[e[i].t]&&min_l[e[i].f]+e[i].l<min_l[e[i].t])
                    pre[e[i].t]=e[i].f,min_l[h_t.v=e[i].t]=h_t.l=min_l[e[i].f]+e[i].l,q.push(h_t);
    }
}
```

8.find_component.cpp

```
//有向图强连通分支(tarjan 算法),dfs 邻接表形式,O(n+m)
//返回分支数,id[i]=j,点 i 所在分支为 j   (j=1,2,3...)
//传入图的大小 n,不相邻点边权 l（可为 0）
#include <cstdio>
#include <cstring>
#define MAXN 5010
#define MAXS 100000
typedef int elem_t;

struct edge{ int f,t,n; elem_t l; }e[MAXS];
int v[MAXN],list[MAXN],id[MAXN],st[MAXN],dfn[MAXN],low[MAXN],cnt,top;

void insert(int f,int t,elem_t l)
{
    e[top].f=f,e[top].t=t,e[top].l=l;
    e[top].n=list[f],list[f]=top++;
}

void search(int n,int now,int &ret)
{
    int i;
    dfn[st[st[0]++]=now]=low[now]=++cnt; v[now]=1;
    for(i=list[now]; i!=-1; i=e[i].n)
        if(!dfn[e[i].t])
        {
            search(n,e[i].t,ret);
            if(low[e[i].t]<low[now])        low[now]=low[e[i].t];
        }
        else if(dfn[e[i].t]<dfn[now])
        {
                if(v[e[i].t]&&dfn[e[i].t]<low[now])
                    low[now]=dfn[e[i].t];
            }
    if(low[now]==dfn[now])
        for(ret++; st[st[0]]!=now; id[st[--st[0]]]=ret,v[st[st[0]]]=0);
}

int find_components(int n)
{
    int ret=0,i;
    for(i=0; i<=n+1; i++)   st[i]=-1,dfn[i]=0;
    for(st[0]=1,cnt=i=0; i<n; i++)
        if(!dfn[i])
            search(n,i,ret);
    return ret;
}
```

9.floyd_warshall.cpp

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

typedef int elem_i;
#define MAXN 200
#define inf 1000000000
elem_i mat[MAXN][MAXN],min[MAXN][MAXN];
int pre[MAXN][MAXN];
//最短路径(多源 floyd_warshall 邻接阵)
//多源最短路径,floyd_warshall 算法,复杂度 O(n^3)
//求出所有点对之间的最短路经,传入图的大小和邻接阵
//返回各点间最短距离 min[]和路径 pre[],pre[i][j]记录 i 到 j 最短路径上 j 的父结点
//可更改路权类型,路权必须非负!
void floyd_warshall(int n,elem_i mat[][MAXN],elem_i min[][MAXN],int
pre[][MAXN])    //假设无权值为负的回路
{
    int i,j,k;
    for(i=0;i<n;i++)
    for(j=0;j<n;j++)
        min[i][j]=mat[i][j],pre[i][j]=(i==j)?-1:i;
    for(k=0;k<n;k++)
    for(i=0;i<n;i++)
    for(j=0;j<n;j++)
    if(min[i][k]+min[k][j]<min[i][j])
        min[i][j]=min[i][k]+min[k][j],pre[i][j]=pre[k][j];
}
```

10.gen_comb.cpp

```
//生成组合数
#include <cstdio>
#define MAXN 1000

int dat[MAXN],tmp[MAXN];

void show(int n,int tmp[])
{
    int i;
    for(i=0; i<n; i++)    printf("%d ",tmp[i]);
    puts("");
}

void gencomb(int s,int e,int m,int cnt)
{
    int i;
    if(cnt==m)    show(m,tmp);
    else
        for(i=s; i+m-cnt<=e; i++)
            tmp[cnt++]=dat[i],gencomb(i+1,e,m,cnt),cnt--;
}

void gen_comb(int n,int m)
{
    int i;
    for(i=0; i<n; i++)    dat[i]=i+1;
    gencomb(0,n,m,0);
}
```

11.gen_perm.cpp

```
//排列数生成
#include <cstdio>
#define MAXN 1000

int tag[MAXN],tmp[MAXN],a[MAXN];

void show(int m,int tmp[])
{
    int i;
    for(i=0; i<m; i++)    printf("%d ",tmp[i]);
    puts("");
}

void genperm(int n,int m,int cnt)
{
    int i;
    if(cnt==m)    show(m,tmp);
    else
        for(i=0; i<n; i++)
            if(!tag[i])
                tag[i]=1,tmp[cnt]=a[i],genperm(n,m,cnt+1),tag[i]=0;
}

void gen_perm(int n,int m)
{
    int i;
    for(i=0; i<n; i++)    a[i]=i+1,tag[i]=0;
    genperm(n,m,0);
}
```

12.gen_perm_swap.cpp

```
//产生邻位交换全排列 O(n!)
#include <cstdio>
#include <cstring>
#define MAXN 1000

int pos[MAXN],dir[MAXN],a[MAXN];

void show(int n,int tmp[])
{
    int i;
    for(i=0; i<n; i++)     printf("%d ",tmp[i]);
    puts("");
}

void _gen_perm_swap(int n,int l)
{
    int p1,p2,i,t;
    if(l==n)       show(n,a);
    else
    {
        _gen_perm_swap(n,l+1);
        for(i=0; i<l; i++)
        {
            p2=(p1=pos[l])+dir[l];
            t=a[p2],a[p2]=a[p1],a[p1]=t;
            pos[a[p1]-1]=p1,pos[a[p2]-1]=p2;
            _gen_perm_swap(n,l+1);
        }
    }
    dir[l]=-dir[l];
}

void gen_perm_swap(int n)
{
    int i;
    for(i=0; i<n; i++)     a[i]=i+1,dir[i]=-1,pos[i]=i;
    _gen_perm_swap(n,0);
}
```

13.hungary.cpp

```
//二分图最大匹配,hungary 算法,邻接表形式,复杂度 O(m*e)
//返回最大匹配数,传入二分图大小 m,n 和邻接表 list(只需一边)__此模板为 m 个点这边
//match1,match2 返回一个最大匹配,未匹配顶点 match 值为-1
#include <cstdio>
#define MAXN 220
#define MAXS 100000
typedef int elem_t;

struct edge{ int f,t,n; elem_t l; }e[MAXS];
int top,match1[MAXN],q[MAXS],match2[MAXN],list[MAXN],tmp2[MAXN];

void insert(int f,int t,elem_t l)
{
    e[top].f=f,e[top].t=t,e[top].l=l;
    e[top].n=list[f],list[f]=top++;
}

int hungary(int m,int n)
{
    int i,j,k,l,r,mx=(m>n)?m:n,ret=0,x,y,t;
    for(i=0; i<mx; i++)     match1[i]=match2[i]=-1;
    for(i=0; i<m; ret+=(match1[i++]>-1))
    {
        for(j=0; j<n; j++)      tmp2[j]=-1;
        for(q[l=r=0]=i; l<=r&&match1[i]<0; l++)
            for(k=q[l],j=list[k]; j!=-1&&match1[i]<0; j=e[j].n)
                if(tmp2[e[j].t]<0)
                {
                    q[++r]=match2[e[j].t],tmp2[e[j].t]=k;
                    if(q[r]<0)
                        for(y=e[j].t; y>=0; y=t)
                            match2[y]=x=tmp2[y],t=match1[x],match1[x]=y;
                }
    }
    return ret;
}
```

14.KMP_Match.cpp

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

void kmp_fail(char s[],int fail[])
{
    int i,p,l=strlen(s);
    for(fail[0]=-1,i=1; i<l; i++)
    {
        p=fail[i-1];
        while(p>=0&&s[p+1]!=s[i])       p=fail[p];
        if(s[p+1]==s[i])        fail[i]=p+1;
        else    fail[i]=-1;
    }
}

int kmp_match(char s[],char t[],int fail_t[])
{
    int SL=strlen(s),TL=strlen(t),s_i,t_i;
    kmp_fail(t,fail_t);
    for(s_i=t_i=0; s_i<SL&&t_i<TL;)
    {
        if(s[s_i]==t[t_i])      s_i++,t_i++;
        else if(t_i==0) s_i++;
        else    t_i=fail_t[t_i-1]+1;
    }
    return t_i<TL?-1:s_i-t_i;
}
```

15.max_flow_dicnic.cpp

```
//anti(int ind) 求 ind 的反向边
#include <cstdio>
#define MAXN 20100
#define MAXS 880100
#define inf 0x7fffffff
typedef int elem_t;

struct edge{ int f,t,n; elem_t w; }e[MAXS];
int list[MAXN],top,q[MAXS],lay[MAXN];

void insert(int f,int t,elem_t w)
{
  e[top].f=f,e[top].t=t,e[top].w=w;
  e[top].n=list[f],list[f]=top++;
}

inline int anti(int ind){ return ind^1; }
inline int min(int a,int b){ return a<b?a:b; }

int bfs(int s,int t,int n)
{
  int l,r,i,j;
  for(i=0; i<n; i++)  lay[i]=-1;
  q[l=r=0]=s; lay[s]=0;
  for(; l<=r; )
  {
     i=q[l++];
     for(j=list[i]; j!=-1; j=e[j].n)
     {
        if(lay[e[j].t]<0&&e[j].w>0)
           lay[e[j].t]=lay[i]+1,q[++r]=e[j].t;
     }
  }
  return lay[t]>=0;
}

elem_t dfs(int now,int t,elem_t now_flow)
{
  int ttl=0,single,i,j;
  if(now==t)  return now_flow;
  for(i=list[now]; i!=-1&&ttl<now_flow; i=e[i].n)
     if(lay[e[i].t]==lay[now]+1&&e[i].w>0&&(single=dfs(e[i].t,t,min(e[i].w,now_flow-ttl))))
     {
        e[i].w-=single;
        e[anti(i)].w+=single;
        ttl+=single;
     }
  if(ttl==0)  lay[now]=-1;
  return ttl;
}

elem_t dicnic(int s,int t,int n)
{
  elem_t ret=0;
  while(bfs(s,t,n))
```

```
        ret+=dfs(s,t,inf);
    return ret;
}
```

16.mcmf.cpp

```
#include <cstdio>
#define inf 1000000000
#define MAXN 200
#define MAXS 160000

struct edge{ int f,t,c,flow,n; }e[MAXS];
int list[MAXN],top,q[MAXS],v[MAXN],pre[MAXN],min[MAXN],max_flow;

void s_insert(int f,int t,int c,int flow)
{
        e[top].f=f,e[top].t=t,e[top].c=c,e[top].flow=flow;
        e[top].n=list[f],list[f]=top++;
}

void insert(int f,int t,int c,int flow)
{
        s_insert(f,t,c,flow);
        s_insert(t,f,-c,0);
}

int spfa(int n,int s,int t)
{
        int i,l,r,j;
        for(i=0; i<n; i++)      pre[i]=-1,min[i]=inf,v[i]=0;
        q[l=r=0]=s;
        min[s]=0,v[s]=1;
        for(; l<=r; )
        {
                i=q[l++];
                v[i]=0;
                for(j=list[i]; j!=-1; j=e[j].n)
                        if(e[j].flow>0&&min[i]+e[j].c<min[e[j].t])
                        {
                                pre[e[j].t]=j;
                                min[e[j].t]=min[i]+e[j].c;
                                if(!v[e[j].t])
                                        v[e[j].t]=1,q[++r]=e[j].t;
                        }
        }
        return min[t]!=inf;
}

int mcmf(int n,int s,int t)
{
        int ret=0,i,now_flow;
        max_flow=0;
        for(; spfa(n,s,t); )
        {
                now_flow=inf;
                for(i=pre[t]; i!=-1; i=pre[e[i].f])
```

```
            if(now_flow>e[i].flow)
                now_flow=e[i].flow;
        for(i=pre[t]; i!=-1; i=pre[e[i].f])
            e[i].flow-=now_flow,e[i^1].flow+=now_flow;
        max_flow+=now_flow;
        ret+=now_flow*min[t];
    }
    return ret;
}
```

17.modular_exponent.cpp

```
#include <cstdio>
typedef long long LL;

//计算 m^a, O(loga), 本身没什么用, 注意这个按位处理的方法 :-P
LL exponent(LL m,LL a)
{
    LL ret=1;
    for(; a; a>>=1,m*=m)
        if(a&1)
            ret*=m;
    return ret;
}

//计算幂取模 a^b mod n, O(logb)
LL modular_exponent(LL a,LL b,LL m)
{
    LL ret=1;
    for(; b; b>>=1,a=(a%m*(a%m))%m)
        if(b&1)
            ret=ret%m*(a%m)%m;
    return ret;
}
```

18.modular_linear_system.cpp

```
#include <cstdio>
#define MAXN 100
typedef long long LL;

LL gcd(LL a,LL b){ return b?gcd(b,a%b):a; }
LL lcm(LL a,LL b){ return a*b/gcd(a,b); }

//扩展 Euclid 求解 gcd(a,b)=ax+by
LL ext_gcd(LL a,LL b,LL &x,LL &y)
{
    LL ret,t;
    if(!b)
    {
        x=1,y=0;
        return a;
    }
    ret=ext_gcd(b,a%b,x,y);
    t=x,x=y,y=t-a/b*y;
    return ret;
}
```

```
//求解模线性方程 ax=b (mod m)
//返回解的个数,解保存在 sol[]中
LL modular_linear(LL a,LL b,LL m,LL sol[])
{
    LL d,e,x,y,i;
    d=ext_gcd(a,m,x,y);
    if(b%d) return 0;
    e=(x*(b/d)%m+m)%m;
    for(i=0; i<d; i++)
        sol[i]=(e+i*(m/d))%m;
    return d;
}

//求解模线性方程组(中国剩余定理)
//  x = b[0] (mod w[0])
//  x = b[1] (mod w[1])
//  ...
//  x = b[k-1] (mod w[k-1])
//要求 w[i]>0,w[i]与 w[j]互质,解的范围 1..n,n=w[0]*w[1]*...*w[k-1]
LL modular_linear_system(LL b[],LL w[],LL k)
{
    LL d,x,y,ret=0,m,n=1,i;
    for(i=0; i<k; i++)      n*=w[i];
    for(i=0; i<k; i++)
    {
        m=n/w[i];
        d=ext_gcd(w[i],m,x,y);
        ret=(ret+y*m*b[i])%n;
    }
    return (ret+n)%n;
}

//求解模线性方程组(中国剩余定理)
//  x = b[0] (mod w[0])
//  x = b[1] (mod w[1])
//  ...
//  x = b[k-1] (mod w[k-1])
//要求 w[i]>0,w[i]与 w[j]可以不互质,解的范围 0,1..m-1,m=w[i]...的最小公倍数
LL modular_linear_system(LL b[],LL w[],LL k)
{
    LL t,d,x,y,ret=b[0],m=w[0],tt,fac,nextm,i;
    for(i=1; i<k; i++)
    {
        d=ext_gcd(m,w[i],x,y);
        t=b[i]-ret;
        if(t%d) return -1;
        tt=w[i]/d;
        fac=(x*(t/d)%tt+tt)%tt;
        nextm=m/d*w[i];
        ret=((ret+m*fac)%nextm+nextm)%nextm;
        m=nextm;
    }
    return (ret%m+m)%m;
}
```

19.mst_kruskal.cpp

```
//kruskal 优先队列＋并查集
//nkoj 2184 ac
#include <cstdio>
#include <cstring>
#include <queue>
#define MAXN 10010
#define MAXS 200020
#define inf 1000000000
typedef int elem_t;
using namespace std;

struct edge{ int f,t,n; elem_t l;
	friend bool operator<(edge a,edge b){ return a.l>b.l; }
}e[MAXS],e_t;
struct ufind
{
	int p[MAXN],t;
	void ini(){ memset(p,0,sizeof(p)); }
	void run(int &x){ for(; p[t=x]; x=p[x],p[t]=(p[x]?p[x]:x));}
	int isfriend(int i,int j){ run(i); run(j); return i==j&&i; }
	void setfriend(int i,int j){ run(i); run(j); p[i]=(i==j)?0:j; }
}uf;
int top,list[MAXN];

void insert(int f,int t,elem_t l)
{
	e[top].f=f,e[top].t=t,e[top].l=l;
	e[top].n=list[f],list[f]=top++;
}

elem_t kruskal(int n)
{
	priority_queue<edge> q;
	uf.ini(); elem_t ret=0;
	int i,j;
	for(i=0; i<n; i++)
		for(j=list[i]; j!=-1; j=e[j].n)
			q.push(e[j]);
	for(i=0; i<n-1&&!q.empty(); )
	{
		e_t=q.top(); q.pop();
		if(!uf.isfriend(e_t.f+1,e_t.t+1))
			ret+=e_t.l,uf.setfriend(e_t.f+1,e_t.t+1),i++;
	}
	return ret;
}
```

20.mst_prim.cpp

```
//prim 优先队列
//经 nkoj 2184 ac
#include <cstdio>
#include <queue>
```

```
using namespace std;
#define MAXN 10010
#define MAXS 200020
#define inf 1000000000
typedef int elem_t;

struct edge{ int f,t,n; elem_t l; }e[MAXS];
struct heap_t{ int v; elem_t l;
    friend bool operator<(heap_t a,heap_t b){ return a.l>b.l; }
}h_t;
int top,pre[MAXN],v[MAXN],list[MAXN];
elem_t min_l[MAXN];

void insert(int f,int t,elem_t l)
{
    e[top].f=f,e[top].t=t,e[top].l=l;
    e[top].n=list[f],list[f]=top++;
}

elem_t prim(int n)
{
    int i,j; elem_t ret=0;
    priority_queue<heap_t> q;
    for(i=0; i<n; i++)      min_l[i]=inf,v[i]=0,pre[i]=-1;
    h_t.v=h_t.l=0; q.push(h_t);
    for(; !q.empty(); )
    {
        h_t=q.top(),q.pop(); i=h_t.v;
        if(!v[i])
            for(v[i]=1,ret+=h_t.l,j=list[i]; j!=-1; j=e[j].n)
                if(!v[e[j].t]&&min_l[e[j].t]>e[j].l)
                    pre[e[j].t]=i,min_l[h_t.v=e[j].t]=h_t.l=e[j].l,q.push(h_t);
    }
    return ret;
}
```

21.num2comb.cpp

```
#include <cstdio>
typedef long long LL;

LL comb(int n,int m)
{
    LL ret=1; int i=n;
    for(; i>n-m; i--)       ret*=i,ret/=n-i+1;
    return ret;
}

//求出 c 的第 t 个组合,t=1 时为第 1 个排列...c={ 1,2,3....n }
void num2comb(int n,int m,LL t,int c[])
{
    int i,j=1,p;
    for(i=0; i<m; c[i++]=j++)
        for(; t>(p=comb(n-j,m-i-1)); t-=p,j++);
}
```

22.num2perm.cpp

```
#include <cstdio>

//求出 p 的第(t+1)%(n!)个排列,t=0 时为第 1 个排列...p={ 0,1,....n-1 }
void num2perm(int n,int t,int p[])
{
	int i,j;
	for(i=n-1; i>=0; i--)
		p[i]=t%(n-i),t/=n-i;
	for(i=n-1; i>=0; i--)
		for(j=i-1; j>=0; j--)
			if(p[j]<=p[i])
				p[i]++;
}
```

23.perm2num.cpp

```
//perm2num.cpp
#include <cstdio>

int perm2num(int n,int p[])
{
	int ret=1,k=1,i,j;
	for(i=n-2; i>=0; k*=(n-(i--)))
		for(j=i+1; j<n; j++)
			if(p[j]<p[i])
				ret+=k;
	return ret;
}
```

24.point_binary_connect_component.cpp

```
//tarjan
#include <cstdio>
#include <cstring>
#define MAXS 8000800
#define MAXN 1010

struct edge{ int f,t,n,v; }e[MAXS];
int list[MAXN],top,st[MAXS],s_top,dfn[MAXN],low[MAXN],id,comp[MAXN];

void s_insert(int f,int t,int v)
{
	e[top].f=f,e[top].t=t,e[top].v=v;
	e[top].n=list[f],list[f]=top++;
}

void insert(int f,int t,int v)
{
	s_insert(f,t,0);
	s_insert(t,f,0);
}

inline int min(int a,int b){ return a<b?a:b; }


void record(int now,int n)
```

```
{
	int i;
	for(id++; ;)
	{
		i=st[s_top--];
		comp[e[i].f]=comp[e[i].t]=id;
		if(e[i].f==now) break;
	}
}

void dfs(int now,int &cnt,int n)
{
	int i,t;
	dfn[now]=low[now]=++cnt;
	for(i=list[now]; i!=-1; i=e[i].n)
		if(!e[i].v)
		{
			e[i].v=e[i^1].v=1;
			t=e[i].t;
			st[++s_top]=i;
			if(!dfn[t])
			{
				dfs(t,cnt,n);
				low[now]=min(low[now],low[t]);
				if(low[t]>=dfn[now])
					record(now,n);
			}
			else low[now]=min(low[now],dfn[t]);
		}
}

void p_b_component(int n)
{
	int i,cnt=0; s_top=-1,id=0;
	memset(st,0,sizeof(st));
	memset(dfn,0,sizeof(dfn));
	for(i=0; i<n; i++)
		if(!dfn[i])
			dfs(i,cnt,n);
}
```

25.segment_tree.cpp

```
//线段树
//可以计算长度和线段数
//可以处理加入边和删除边不同的情况
//insert_seg 和 del_seg 用于加入边删除边
//seg_len 求长度,seg_cnt 求线段数
//id 传根节点(一律为 1)
//ll,rr 传线段(端点)
#include <cstdio>
#include <cmath>
#include <cstring>
#include <algorithm>
using std::sort;
#define N 10110
```

```
struct line{ int x,y1,y2,tag; }l[N<<1];
struct seg_tree
{
	int l,r,col,len,cnt,bl,br;
}segt[N<<2];
//col 重叠线段数, 0 表示无线段覆盖
//len 该段上的已有线段总长度
//cnt 该段上不连续不重叠的线段数
//bl,br 该段上左右端是否有线段覆盖

inline int L(int i){ return i<<1; }
inline int R(int i){ return (i<<1)|1; }
inline int min(int a,int b){ return a<b?a:b; }
inline int max(int a,int b){ return a>b?a:b; }
inline int length(int l,int r){ return coor[r]-coor[l]; }

int coor[N<<1],ttl,top;
//coor 离散化坐标数组

int get_id(int x)
{
	int l,r,m;
	for(l=1,r=top; l<=r; )
	{
		m=(l+r)>>1;
		if(coor[m]<x)
			l=m+1;
		else if(coor[m]>x)
			r=m-1;
		else
			return m;
	}
	return -1; //impossible
}

void create_tree(int id,int l,int r)
{
	int m=(l+r)>>1;
	segt[id].l=l,segt[id].r=r;
	segt[id].col=segt[id].len=0;
	segt[id].bl=segt[id].br=segt[id].cnt=0;
	if(r-l<=1)
		return ;
	create_tree(L(id),l,m);
	create_tree(R(id),m,r);
}

void update(int id,int l,int r)
{
	if(segt[id].col||r==l)
	{
		segt[id].len=length(l,r);
		segt[id].cnt=segt[id].bl=segt[id].br=1;
	}
	else
	{
```

```
            segt[id].len=segt[L(id)].len+segt[R(id)].len;
            segt[id].cnt=segt[L(id)].cnt+segt[R(id)].cnt;
            if(segt[L(id)].br&&segt[R(id)].bl)
                segt[id].cnt--;
            segt[id].bl=segt[L(id)].bl;
            segt[id].br=segt[R(id)].br;
        }
}

//往 id 上加[ll,rr]的线段
void insert_seg(int id,int ll,int rr)
{
    int l=segt[id].l,r=segt[id].r,m=(l+r)>>1;
    if(ll==l&&rr==r)
        segt[id].col++;
    else
    {
        if(ll<m)
            insert_seg(L(id),ll,min(m,rr));
        if(rr>m)
            insert_seg(R(id),max(ll,m),rr);
        if(segt[L(id)].col&&segt[R(id)].col)
        {
            segt[L(id)].col--;
            update(L(id),l,m);
            segt[R(id)].col--;
            update(R(id),m,r);
            segt[id].col++;
        }
    }
    update(id,l,r);
}

//往 id 上减[ll,rr]的线段
void del_seg(int id,int ll,int rr)
{
    int l=segt[id].l,r=segt[id].r,m=(l+r)>>1;
    if(ll==l&&rr==r)
        segt[id].col--;
    else if(segt[id].col)
    {
        segt[id].col--;
        if(ll>l)
            insert_seg(id,l,ll);
        if(rr<r)
            insert_seg(id,rr,r);
    }
    else
    {
        if(ll<m)
            del_seg(L(id),ll,min(m,rr));
        if(rr>m)
            del_seg(R(id),max(ll,m),rr);
    }
    update(id,l,r);
}
```

```
//求 id 上[ll,rr]范围内覆盖的线段长度
int seg_len(int id,int ll,int rr)
{
    int l=segt[id].l,r=segt[id].r,m=(l+r)>>1,ret=0;
    if(segt[id].col||(ll==l&&rr==r))
        return segt[id].len;
    if(ll<m)
        ret+=seg_len(L(id),ll,min(rr,m));
    if(rr>m)
        ret+=seg_len(R(id),max(ll,m),rr);
    return ret;
}

//求 id 上[ll,rr]范围内线段的段数
int seg_cnt(int id,int ll,int rr)
{
    int l=segt[id].l,r=segt[id].r,m=(l+r)>>1,ret=0;
    if(segt[id].col)
        return 1;
    if(l==ll&&r==rr)
        return segt[id].cnt;
    if(ll<m)
        ret+=seg_cnt(L(id),ll,min(rr,m));
    if(rr>m)
        ret+=seg_cnt(R(id),max(ll,m),rr);
    return ret;
}
```

26.spfa.cpp

```
#include <cstdio>
#include <cstring>
#include <cmath>
#include <queue>
#define inf 1e8
#define MAXS 2000001
#define MAXN 1001

typedef double elem_t;
struct edge{ int f,t,n; elem_t l; }e[MAXS];
int list[MAXN],top=0,v[MAXN],in[MAXN],q[MAXS];
elem_t min[MAXN];

void insert(int f,int t,elem_t l)
{
    e[top].f=f,e[top].t=t,e[top].l=l;
    e[top].n=list[f],list[f]=top++;
}

bool spfa(int n,int s)
{
    int i,j,l,r;
    for(i=0; i<n; i++)      v[i]=0,min[i]=inf,in[i]=0;
    l=r=min[s]=0,q[l]=s,in[s]++;
    for(; l<=r; )
    {
        i=q[l++],v[i]=0;
        if(in[i]>n)     return 0;
        for(j=list[i]; j!=-1; j=e[j].n)
            if(min[i]+e[j].l<min[e[j].t])
            {
                min[e[j].t]=min[i]+e[j].l;
                if(!v[e[j].t])
                {
                    v[e[j].t]=1;
                    in[e[j].t]++;
                    q[++r]=e[j].t;
                }
            }
    }
    return 1;
}
```

27.trie_tree.cpp

```
//untest
#include <cstdio>
#include <cstring>
#define MAXN 100011
#define MAXC 11
int top;

int get_ind(char x){ return x-'0'; }
struct trie_node{ bool is; int n[MAXC]; }no[MAXN];

struct trie_tree
{
    trie_node r;
    trie_tree(){ r=no[0]; }
    void insert(char s[])
    {
        int i,l=strlen(s),p=0;
        for(i=0; i<l; i++)
        {
            if(no[p].n[get_ind(s[i])]<0)
            {
                no[top].is=0;
                no[p].n[get_ind(s[i])]=top++;
            }
            p=no[p].n[get_ind(s[i])];
        }
        no[p].is=1;
    }

    int search(char s[])
    {
        int i,l=strlen(s),p=0;
        for(i=0; i<l&&p!=-1; i++)
            p=no[p].n[get_ind(s[i])];
        return p!=-1&&no[p].is==1;
```

```
	}
}trie;

void init()
{
	int i,j;
	for(i=0; i<MAXN; i++)
		for(no[i].is=j=0; j<MAXC; j++)
			no[i].n[j]=-1;
	top=1;
}
```

28.union_find_set.cpp

```
#include <cstdio>
#include <cstring>
#define MAXN 100011

//1st_kind
struct ufind
{
	int p[MAXN],t;
	void ini(){ memset(p,0,sizeof(p)); }
	void run(int &x){ for(; p[t=x]; x=p[x],p[t]=(p[x]?p[x]:x));}
	int isfriend(int i,int j){ run(i); run(j); return i==j&&i; }
	void setfriend(int i,int j){ run(i); run(j); p[i]=(i==j)?0:j; }
}uf;

//2nd_kind
struct ufind
{
	int p[N],t;
	void init()
	{
		for(int i=0; i<=N; i++)
			p[i]=i;
	}
	void run(int &x)
	{
		for(; p[t=x]!=x; x=p[x],p[t]=p[x]);
	}
	int isfriend(int i,int j)
	{
		run(i); run(j);
		return i==j;
	}
	void setfriend(int i,int j)
	{
		run(i); run(j);
		p[i]=p[j];
	}
}uf;


inline int sig(int x){ return x>0?1:-1; }
inline int abs(int x){ return x>0?x:-x; }
```

```
struct ufind
{
	int p[MAXN],t;

	void init(){ memset(p,0,sizeof(p)); }

	void _run(int &x)
	{
		for( ; p[t=abs(x)];
x=sig(x)*p[abs(x)],p[t]=sig(p[t])*(p[abs(x)]?p[abs(x)]:abs(p[t])));
	}

	void _run_both(int &i,int &j)
	{
		_run(i); _run(j);
	}

	void _set_side(int x,int i,int j)
	{
		p[abs(i)]=sig(i)*(abs(i)==abs(j)?0:(x*j));

	}

	int _judge_side(int x,int i,int j)
	{
		return ((i==x*j)&&i);
	}

	int set_friend(int i,int j)
	{
		_run_both(i,j);
		_set_side(1,i,j);
		return !_judge_side(-1,i,j);
	}

	int set_enemy(int i,int j)
	{
		_run_both(i,j);
		_set_side(-1,i,j);
		return !_judge_side(1,i,j);
	}

	int is_friend(int i,int j)
	{
		_run_both(i,j);
		return _judge_side(1,i,j);
	}

	int is_enemy(int i,int j)
	{
		_run_both(i,j);
		return _judge_side(-1,i,j);
	}
}uf;
```

29.KM_match.cpp

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#define inf 0x3fffffff
#define N 1001
#define S 1000100

using namespace std;

struct edge{ int f,t,n; }e[S];

int slack[N],valx[N],valy[N],mat[N][N],m2y[N],vx[N],vy[N],list[N],top;

void insert(int f,int t)
{
	e[top].f=f,e[top].t=t;
	e[top].n=list[f],list[f]=top++;
}

int find_path(int now)
{
	int t,i,v;
	vx[now]=1;
	for(i=list[now]; i!=-1; i=e[i].n)
	{
		t=e[i].t;
		v=valx[now]+valy[t]-mat[now][t];
		if(!vy[t]&&v==0)
		{
			vy[t]=1;
			if(m2y[t]==-1||find_path(m2y[t]))
			{
				m2y[t]=now;
				return 1;
			}
		}
		else if(slack[t]>v)
			slack[t]=v;
	}
	return 0;
}

int km_match(int n)
{
	int i,j,k,ret=0,d;
	for(i=0; i<n; i++)
	{
		m2y[i]=-1;
		valx[i]=-inf,valy[i]=0;
		for(j=0; j<n; j++)
			valx[i]=max(valx[i],mat[i][j]);
	}
	for(i=0; i<n; i++)
	{
		memset(vx,0,sizeof(vx));
		memset(vy,0,sizeof(vy));
```

```
        for(k=0; k<n; k++)
            slack[k]=inf;
        for(; !find_path(i); )
        {
            d=inf;
            for(k=0; k<n; k++)
                if(!vy[k]&&slack[k]<d)
                    d=slack[k];
            for(k=0; k<n; k++)
            {
                if(vx[k])
                    valx[k]-=d,vx[k]=0;
                if(vy[k])
                    valy[k]+=d,vy[k]=0;
            }
        }
    }
    for(i=0; i<n; i++)
        ret+=valx[i]+valy[i];
    return ret;
}
```

bdep__