Codebook of Team whatever

Contents

一、 Note

1. Formula
   - 1-1. 次方和 1
   - 1-2. Pick 定理 1
   - 1-3. 尤拉公式 1
   - 1-4. Harmonic Number 1
   - 1-5. Fibonacci Number 1
   - 1-6. Generating Function 1
   - 1-7. Catalan Number 1

二、 Combination

1. Polya 定理 1
2. 2-SAT 2

三、 Geometry

1. Header 2
2. 平面上點與線 2
3. 多邊形 3
4. 三角形 3
5. 圓 4
6. 公式 4

四、 Mathematics

1. Euler Phi Function 4
2. ax+by=gcd(a,b) 4
3. ax=b(modn) 4
4. 中國剩餘定理 4
5. Modular multiplicative inverse 5
6. Determinant 5
7. 質數判定(Miller-Rabin) 5

五、 Graph

1. 二分圖匹配（Hopcroft-Karp） 5
2. 二分圖匹配（Hungry Method） 5
3. Maximum Flow 5
4. Minimum cost Maximum Flow 5
5. 二分圖帶權匹配（Hungry Method） 6
6. All Pairs Maximum Flow（Gomory-Hu Tree） 6
7. Minimum Cut(Stoer-Wagner) 7
8. Strongly Connected Component(Kosaraju) 7
9. Bi-Connected Component(DFN & LOW) 6
10. Lowest Common Ancestor 7
11. General Graph Matching 7
12. Directed MST 8

六、 Data Structure

1. RMQ 8
2. Binary Indexed Tree 8

七、 Strings

1. KMP 9
2. Suffix Array 9

一、Note

1. Formula

1-1. 次方和

$\sum_{k=1}^{n} k^3 = \frac{n(n+1)(2n+1)}{6}$, $\sum_{k=1}^{n} k^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$

$\sum_{k=1}^{n} k^5 = \frac{1}{3}\left[\frac{n(n+1)}{2}\right]^2 [2x^2 + 2n - 1].$

1-2. pick 定理

簡單多邊形面積 ＝ 內部格子點數 ＋ 邊上格子點數 -1

1-3. 尤拉公式

點 − 線 + 面 = 1 + 連通塊個數

1-4. Harmonic Number

$$H_n = \sum_{k=1}^{n} \frac{1}{k} = \int_0^1 \frac{1-x^n}{1-x} dx = \sum_{k=1}^{n} (-1)^{k-1} \frac{1}{k} \binom{n}{k}$$

1-5. Fibonacci number

$F_n = \frac{\phi^n - (1-\phi)^n}{\sqrt{5}}, \phi = \frac{1+\sqrt{5}}{2} \approx 1.6180339887$

$\lim_{n\to\infty} \frac{F_{n+1}}{F_n} = \phi, \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n = \begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix},$

$F_{2n-1} = {F_n}^2 + {F_{n-1}}^2, F_{2n} = (2F_{n-1} + F_n)F_n$

1-6. Generating function

$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}, \sum_{k=0}^{\infty} \binom{n+k}{k} x^k = \frac{1}{(1-x)^k}$

$\sum_{k=0}^{\infty} \frac{x^k}{k!} = e^x, \sum_{k=0}^{\infty} \frac{x^k}{2k!} = \frac{e^x + e^{-x}}{2}, \sum_{k=0}^{\infty} \frac{x^k}{(2k+1)!} = \frac{e^x - e^{-x}}{2}$

解遞迴式可設 $A(x), B(x) \ldots$ 後乘 x^n 並取 $\sum a_k x^k$ 至無限大求之。

1-7. Catalan number

N 點二元樹數、三角分割正 N 邊形方法數、N 對括號匹配數、N 物品分群數、N 個長方形填充高度為 n 的階梯方法數……

$F_n = \binom{2n}{n} - \binom{2n}{n-1} = \frac{F_{n-1} * (4n-2)}{n+1} = \frac{1}{n+1}\binom{2n}{n}, F_0 = 1$

當 $n = 2^k - 1$，F_n 為奇數，否則為偶數。

二、Combination

1. Polya 定理

著色方案為

k 色對 n 點著色數 $= \frac{\sum_{i=1}^{p} k^{\text{循環節 i 的方法數}}}{p}$,若有 p 個循環節

```
int polya(int* perm,int n,int& num){//num 循環節個數
    int tmp,v[n]={},ret=1;
    num = 0;
    for(int i=0;i<n;i++){
        if(!v[i]){
            num++;
            tmp=0
            for(int p=i;!v[p=perm[p]];tmp++){
                v[p]=1;
            }
            ret*=tmp/gcd(ret,tmp);
        }
    }
    return ret;//置換群最小週期
}
```

2. 2-SAT

Notice:互斥對稱性

構圖：ab 衝突→建立(a,!b)及(b,!a)兩邊（對稱）

判斷：若存在 a 及!a 存在同一強連通塊，則為 false

求解：拓樸排序後逆向挑點即為一解

```
int n,m,idx[1010],ncnt,ans[1010];
vector<int> conn[1010],bconn[1010],stamp;
bool v[1010];
void DFS(int np){
	if(v[np]) return;
	v[np]=1;
	int Size = conn[np].size();
	for(int i=0;i<Size;i++){
	   DFS(conn[np][i]);
	}
	stamp.push_back( np );
}
void KOSA(int np){
	if(v[np]) return;
	v[np]=1;
	idx[np]=ncnt;
	int Size = bconn[np].size();
	for(int i=0;i<Size;i++){
	   KOSA(bconn[np][i]);
	}

}
void SCC(){
	stamp.clear();
	memset(v,0,sizeof(v));
	for(int i=0;i<2*n;i++){
	   DFS(i);
	}
	memset(v,0,sizeof(v));
	ncnt=0;
	for(int i=2*n-1;i>=0;i--){
	   if(v[stamp[i]]==0){
		ncnt++;
		KOSA(stamp[i]);
	   }
	}
}
bool chk(){
	for(int i=0;i<n;i++){
	   if(idx[2*i] == idx[2*i+1]){
		return 0;
	   }
	}
	return 1;
}
void getans(){//ans[i]為 1 的為一組解
	int np;
	memset(ans,-1,sizeof(ans));
	for(int i=0;i<2*n;i++){
	   np = stamp[ i ] / 2;
	   if(ans[stamp[ i ]]!=-1){
		continue;
	   }else if(stamp[i] == 2*np){
		ans[2*np]=0;
		ans[2*np+1]=1;
	   }else{
		ans[2*np+1]=0;
		ans[2*np]=1;
	   }
	}
}
```

三、Geometry

1. Header

```
#define EPS 1e-8
#define offset 10000
#define zero(x) (((x)>0?(x):-(x))<eps)
#define _sign(x) ((x)>eps?1:((x)<-eps?2:0))
struct point{double x,y;};
struct line{point a,b;};
//cross product
double cross(point p1,point p2,point p0){
	return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}

double cross (x1, y1, x2, y2, x0, y0){//all double
	return (x1-x0)*(y2-y0)-(x2-x0)*(y1-y0);
}
//dot product
double dot(point p1,point p2,point p0){
	return (p1.x-p0.x)*(p2.x-p0.x)+(p1.y-p0.y)*(p2.y-p0.y);
}
double dot (x1, y1, x2, y2, x0, y0){//all double
	return (x1-x0)*(x2-x0)+(y1-y0)*(y2-y0);
}
```

2. 平面上點與線

2-1. 三點共線

```
int dots_inline(point p1,point p2,point p3){
	return zero(cross(p1,p2,p3));
}
int dots_inline(x1, y1, x2, y2, x3, y3){//all double
	return zero(cross(x1,y1,x2,y2,x3,y3));
}
```

2-2. 判斷是否在線段上

```
int dot_online_in(point p,line l){
	return
zero(cross(p,l.a,l.b))&&
(l.a.x-p.x)*(l.b.x-p.x)<eps&&
(l.a.y-p.y)*(l.b.y-p.y)<eps;
}
```

2-2. 兩點是否同側

```
int same_side(point p1,point p2,line l){
   return cross(l.a,p1,l.b)*cross(l.a,p2,l.b)>eps;
}//回傳 0 同側
int opposite_side(point p1,point p2,line l){
   return cross(l.a,p1,l.b)*cross(l.a,p2,l.b)<-eps;
}//回傳 0 異側
```

2-3. 線段相交

```
int intersect_in(line u,line v){
if (!dots_inline(u.a,u.b,v.a)|| !dots_inline(u.a,u.b,v.b))
   return !same_side(u.a,u.b,v)&&!same_side(v.a,v.b,u);
return dot_online_in(u.a,v)||dot_online_in(u.b,v)||
	dot_online_in(v.a,u)||dot_online_in(v.b,u);
}
```

2-4. 直線交點(先判斷平行，線段則先判斷相交)

```
point intersection(line u,line v){
	point ret=u.a;
	double t=((u.a.x-v.a.x)*(v.a.y-v.b.y)-(u.a.y-v.a.y)*(v.a.x-v.b.x))
		/((u.a.x-u.b.x)*(v.a.y-v.b.y)-(u.a.y-u.b.y)*(v.a.x-v.b.x));
	ret.x+=(u.b.x-u.a.x)*t;		ret.y+=(u.b.y-u.a.y)*t;
	return ret;
}
```

2-5. 點到直線最近點

```
point ptoline(point p,line l){
	point t=p;	t.x+=l.a.y-l.b.y,t.y+=l.b.x-l.a.x;
	return intersection(p,t,l.a,l.b);
}
```

2-6. 點到直線最近距離

```
double disptoline(point p,line l){
	return fabs(cross(p,l.a,l.b))/distance(l.a,l.b);
}
```

2-7. 點到線段最近點

```
point ptoseg(point p,line l){
   point t=p;
   t.x+=l.a.y-l.b.y,t.y+=l.b.x-l.a.x;
   if (cross(l.a,t,p)* cross (l.b,t,p)>eps)
      return distance(p,l.a)<distance(p,l.b)?l.a:l.b;
   return intersection(p,t,l.a,l.b);
}
```

2-7. 點到線段最近距離

```
double disptoseg(point p,line l){
   point t=p;
   t.x+=l.a.y-l.b.y,t.y+=l.b.x-l.a.x;
   if (xmult(l.a,t,p)*xmult(l.b,t,p)>eps)
      return distance(p,l.a)<distance(p,l.b)?distance(p,l.a):distance(p,l.b);
   return fabs(xmult(p,l.a,l.b))/distance(l.a,l.b);
}
```

3. 多邊形

3-1. 判定凸多邊形

```
int is_convex(int n,point* p){//按順序
   int i,s[3]={1,1,1};
      for (i=0;i<n&&s[1]|s[2];i++)
         s[_sign(cross(p[(i+1)%n],p[(i+2)%n],p[i]))]=0;
   return s[1]|s[2];
}
```

3-2. 判定點在任意多邊形內

```
int inside_polygon(point q,int n,point* p,int on_edge=1){
   point q2;
   int i=0,count;
   while (i<n)
      for (count=i=0,q2.x=rand()+offset,q2.y=rand()+offset;i<n;i++)
         if (zero(xmult(q,p[i],p[(i+1)%n]))&&
            (p[i].x-q.x)*(p[(i+1)%n].x-q.x)<eps&&
            (p[i].y-q.y)*(p[(i+1)%n].y-q.y)<eps)    return on_edge;
         else if (zero(xmult(q,q2,p[i])))
               break;
         else if (xmult(q,p[i],q2)*xmult(q,p[(i+1)%n],q2)<-eps&&
                  xmult(p[i],q,p[(i+1)%n])*xmult(p[i],q2,p[(i+1)%n])<-eps)
               count++;
   return count&1;
}
```

3-3. 判定線段在任意多邊形相交

```
int inside_polygon(point l1,point l2,int n,point* p){
   point t[MAXN],tt;
   int i,j,k=0;
   if (!inside_polygon(l1,n,p)||!inside_polygon(l2,n,p))    return 0;
   for (i=0;i<n;i++)
      if (opposite_side(l1,l2,p[i],p[(i+1)%n])&&
           opposite_side(p[i],p[(i+1)%n],l1,l2)) return 0;
      else if (dot_online_in(l1,p[i],p[(i+1)%n])) t[k++]=l1;
      else if (dot_online_in(l2,p[i],p[(i+1)%n])) t[k++]=l2;
      else if (dot_online_in(p[i],l1,l2)) t[k++]=p[i];
      for (i=0;i<k;i++)for (j=i+1;j<k;j++){
         tt.x=(t[i].x+t[j].x)/2;
         tt.y=(t[i].y+t[j].y)/2;
         if (!inside_polygon(tt,n,p)) return 0;
      }
   return 1;
}
```

3-4. 三角形重心

```
point tri_barycenter(point a,point b,point c){
         line u,v;
         u.a.x=(a.x+b.x)/2;
         u.a.y=(a.y+b.y)/2;
         u.b=c;
         v.a.x=(a.x+c.x)/2;
         v.a.y=(a.y+c.y)/2;
         v.b=b;
         return intersection(u,v);
}
```

3-5. 多邊形重心

```
point barycenter(int n,point* p){
   point ret,t;
   double t1=0,t2;
   int i;          ret.x=ret.y=0;
   for (i=1;i<n-1;i++)if (fabs(t2=cross(p[0],p[i],p[i+1]))>eps){
      t=tri_barycenter(p[0],p[i],p[i+1]);
      ret.x+=t.x*t2;    ret.y+=t.y*t2;     t1+=t2;
   }
   if (fabs(t1)>eps) ret.x/=t1,ret.y/=t1;
   return ret;
}
```

3-6. 沿 line(l1,l2)切割多邊形於點 side 側

```
void polygon_cut(int& n,point* p,point l1,point l2,point side){
   point pp[100];
   int m=0,i;
   for (i=0;i<n;i++){
      if (same_side(p[i],side,l1,l2))    pp[m++]=p[i];
      if (!same_side(p[i],p[(i+1)%n],l1,l2)&&
         !(zero(xmult(p[i],l1,l2))&&zero(xmult(p[(i+1)%n],l1,l2))))
            pp[m++]=intersection(p[i],p[(i+1)%n],l1,l2);
   }
   for (n=i=0;i<m;i++)
      if (!i||!zero(pp[i].x-pp[i-1].x)||!zero(pp[i].y-pp[i-1].y))
         p[n++]=pp[i];
   if (zero(p[n-1].x-p[0].x)&&zero(p[n-1].y-p[0].y))    n--;
   if (n<3) n=0;
}
```

3-7. 多邊形面積

```
double area_polygon(int n,point* p){
         double s1=0,s2=0;
         int i;
         for (i=0;i<n;i++)
                  s1+=p[(i+1)%n].y*p[i].x,s2+=p[(i+1)%n].y*p[(i+2)%n].x;
         return fabs(s1-s2)/2;
}
```

3-8. 凸包

```
int p_comp(const void *p, const void *q){
   if(((point*)p)->x != ((point*)q)->x)
     return ((point*)p)->x > ((point*)q)->x ? 1 : -1;
   return ((point*)p)->y > ((point*)q)->y ? 1 : -1;
}
void convex_hull(point *a, int n, point *c, int &m){
   int i, j;
   qsort(a, n, sizeof(point), p_comp);
   for(i=0, m=0; i<n; i++){
      for(; m>=2&&cross(c[m-2], c[m-1], a[i])<=0; m--);
      c[m++] = a[i];
   }
   for(i=n-2, j=m+1; i>=0; i--){
      for(; m>=j&&cross(c[m-2], c[m-1], a[i])<=0; m--);
      c[m++] = a[i];
   }
}
```

4. 三角形

4-1. 外心

```
point circumcenter(point a,point b,point c){
         line u,v;
         u.a.x=(a.x+b.x)/2;       u.a.y=(a.y+b.y)/2;
         u.b.x=u.a.x-a.y+b.y;     u.b.y=u.a.y+a.x-b.x;
         v.a.x=(a.x+c.x)/2;       v.a.y=(a.y+c.y)/2;
         v.b.x=v.a.x-a.y+c.y;     v.b.y=v.a.y+a.x-c.x;
         return intersection(u,v);
}
```

4-2. 內心

```
point incenter(point a,point b,point c){
         line u,v;
         double m,n;
         u.a=a;  m=atan2(b.y-a.y,b.x-a.x);       n=atan2(c.y-a.y,c.x-a.x);
         u.b.x=u.a.x+cos((m+n)/2);               u.b.y=u.a.y+sin((m+n)/2);
         v.a=b;  m=atan2(a.y-b.y,a.x-b.x);       n=atan2(c.y-b.y,c.x-b.x);
         v.b.x=v.a.x+cos((m+n)/2);               v.b.y=v.a.y+sin((m+n)/2);
         return intersection(u,v);
}
```

4-3. 垂心

```
point perpencenter(point a,point b,point c){
	line u,v;
	u.a=c;  u.b.x=u.a.x-a.y+b.y;     u.b.y=u.a.y+a.x-b.x;
	v.a=b;  v.b.x=v.a.x-a.y+c.y;     v.b.y=v.a.y+a.x-c.x;
	return intersection(u,v);
}
```

5. 圓

5-1. 直線與圓相交

```
int intersect_seg_circle(point c,double r,point l1,point l2){
	double t1=distance(c,l1)-r,t2=distance(c,l2)-r;
	point t=c;
	if (t1<eps||t2<eps) return t1>-eps||t2>-eps;
	t.x+=l1.y-l2.y;  t.y+=l2.x-l1.x;
	return cross(l1,c,t)* cross (l2,c,t)<eps&& cross (c,l1,l2)-r<eps;
}
```

5-2. 圓與圓相交

```
int intersect_circle_circle(point c1,double r1,point c2,double r2){
	return distance(c1,c2)<r1+r2+eps&&distance(c1,c2)>fabs(r1-r2)-eps;
}
```

5-3. 圓上最近點

```
point dot_to_circle(point c,double r,point p){
	point u,v;
	if (distance(p,c)<eps)  return p;
	u.x=c.x+r*fabs(c.x-p.x)/distance(c,p);
	u.y=c.y+r*fabs(c.y-p.y)/distance(c,p)*((c.x-p.x)*(c.y-p.y)<0?-1:1);
	v.x=c.x-r*fabs(c.x-p.x)/distance(c,p);
	v.y=c.y-r*fabs(c.y-p.y)/distance(c,p)*((c.x-p.x)*(c.y-p.y)<0?-1:1);
	return distance(u,p)<distance(v,p)?u:v;
}
```

5-4. 直線與圓交點

```
void intersection_line_circle
		(point c,double r,point l1,point l2,point& p1,point& p2){
	point p=c;       double t;
	p.x+=l1.y-l2.y;  p.y+=l2.x-l1.x;
	p=intersection(p,c,l1,l2);
	t=sqrt(r*r-distance(p,c)*distance(p,c))/distance(l1,l2);
	p1.x=p.x+(l2.x-l1.x)*t; p1.y=p.y+(l2.y-l1.y)*t;
	p2.x=p.x-(l2.x-l1.x)*t;  p2.y=p.y-(l2.y-l1.y)*t;
}
```

5-5. 圓與圓交點

```
void intersection_circle_circle
	(point c1,double r1,point c2,double r2,point& p1,point& p2){
	point u,v;       double t;
	t=(1+(r1*r1-r2*r2)/distance(c1,c2)/distance(c1,c2))/2;
	u.x=c1.x+(c2.x-c1.x)*t;          u.y=c1.y+(c2.y-c1.y)*t;
	v.x=u.x+c1.y-c2.y;        v.y=u.y-c1.x+c2.x;
	intersection_line_circle(c1,r1,u,v,p1,p2);
}
```

6. 公式

6-1. 三角形

1. 半周長 P=(a+b+c)/2
2. 面積 S=aHa/2=absin(C)/2=sqrt(P(P-a)(P-b)(P-c))
3. 中線 Ma=sqrt(2(b^2+c^2)-a^2)/2=sqrt(b^2+c^2+2bccos(A))/2
4. 角平分線 Ta=sqrt(bc((b+c)^2-a^2))/(b+c)=2bccos(A/2)/(b+c)
5. 垂線 Ha=bsin(C)=csin(B)=sqrt(b^2-((a^2+b^2-c^2)/(2a))^2)
6. 內切圓半徑 r=S/P=asin(B/2)sin(C/2)/sin((B+C)/2)
   =4Rsin(A/2)sin(B/2)sin(C/2)=sqrt((P-a)(P-b)(P-c)/P)
   =Ptan(A/2)tan(B/2)tan(C/2)
7. 外接圓半徑 R=abc/(4S)=a/(2sin(A))=b/(2sin(B))=c/(2sin(C))

6-2. 四邊形

D1,D2 為對角線長,M 對角線中點連線長,A 為對角線夾角

1. a^2+b^2+c^2+d^2=D1^2+D2^2+4M^2
2. S=D1D2sin(A)/2

6-2. 圓內接四邊形

1. ac+bd=D1D2
2. S=sqrt((P-a)(P-b)(P-c)(P-d)), P 為半周長

6-3. 正 n 邊形

R 為外接圓半徑,r 為內切圓半徑

1. 中心角 A=2PI/n
2. 內角 C=(n-2)PI/n
3. 邊長 a=2sqrt(R^2-r^2)=2Rsin(A/2)=2rtan(A/2)
4. 面積 S=nar/2=nr^2tan(A/2)=nR^2sin(A)/2=na^2/(4tan(A/2))

6-4. 圓

1. 弧長 l=rA
2. 弦長 a=2sqrt(2hr-h^2)=2rsin(A/2)
3. 弓形高 h=r-sqrt(r^2-a^2/4)=r(1-cos(A/2))=atan(A/4)/2
4. 扇形面積 S1=rl/2=r^2A/2
5. 弓形面積 S2=(rl-a(r-h))/2=r^2(A-sin(A))/2

6-5. 角錐

1. 體積 V=Ah/3,A 為底面積,h 為高
2. 若為正角錐，側面積 S=lp/2,l 為斜高,p 為底面周長
3. 若為正角錐，表面積 T=S+A

6-6. 圓錐

1. 母線 l=sqrt(h^2+r^2)
2. 側面積 S=PIrl
3. 表面積 T=PIr(l+r)
4. 體積 V=PIr^2h/3

6-7. 球體

1. 表面積 T=4PIr^2
2. 體積 V=4PIr^3/3

四、Math

1. Euler Phi function

P[i]為 i 以下與 i 互質數的個數，pp[]與 pc 為質數表

```
int make_phi(int n){
   if((n&1) && s[(n-1)>>1]) return n-1;
   int nn=n,nnn=n;
   for(int i=0;p[i]<=n && i<pc;i++){
      if(n%p[i]==0){
         if(n/p[i]!=1){
            if((n/p[i])%p[i]==0) return phi[n/p[i]]*p[i];
            if((n/p[i])%p[i]!=0) return phi[n/p[i]]*(p[i]-1);
         }
         while(n%p[i]==0)n/=p[i];
         nn=nn/p[i]*(p[i]-1);
      }
   }
   return nn;
}
```

2. ax+by=gcd(a,b)

```
int ext_gcd(int a,int b,int& x,int& y){
   int t,ret;
   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;
}
```

3. ax=b(mod n)

```
int modular_linear(int a,int b,int n,int* sol){
	int d,e,x,y,i;
	d=ext_gcd(a,n,x,y);
	if (b%d) return 0;
	e=(x*(b/d)%n+n)%n;
	for (i=0;i<d;i++) sol[i]=(e+i*(n/d))%n;
	return d;
}
```

4. 中國剩餘定理

```
int modular_linear_system(int b[],int w[],int k){
   int d,x,y,a=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);
      a=(a+y*m*b[i])%n;
   }
   return (a+n)%n;
}
```

5. Modular multiplicative inverse

Notice:gcd(x,mod) must be 1.

```
long long inv(x, y, p, q, r, s){//all long long
        if(y==0) return p;
        return inv(y, x%y, r, s, p-r*(x/y), q-s*(x/y));
}
long long get_inv(long long x, long long mod) {
        long long r = inv(x, mod, 1, 0, 0, 1);
        return (r%mod+mod)%mod;
}
```

6. Determinant

```
double d(){
    double c,a[330][330],k=1;
    for(int i=0;i<N;i++)
        for(int j=0;j<N;j++)
            a[i][j]=(double)maze[i][j];
    for(int i=0;i<N-1;i++){
        int kk;
        double mx=0;
        for(int j=i;j<N;j++) if(ABS(a[j][i])>mx){kk=j;mx=ABS(a[j][i]);}
        if(kk!=i){
            for(int n=0;n<N;n++){c=a[i][n]; a[i][n]=a[kk][n]; a[kk][n]=c;}
            k*=(-1);
        }
        for(int s=i+1;s<N;s++){
            a[s][i]/=a[i][i];
            for(int t=i+1;t<N;t++) a[s][t]-=a[i][t]*a[s][i];
        }
    }
    for (int i=0;i<N;i++) k*=a[i][i];
    return k;
}
```

7. 質數判定（Miller_Rabin）

```
int miller_rabin(int n,int time=10){
   if (n==1||(n!=2&&!(n%2))||(n!=3&&!(n%3))||(n!=5&&!(n%5))||(n!=7&&!(n%7)))
      return 0;
   while (time--)
      if (modular_exponent(((rand()&0x7fff<<16)+rand()&0x7fff
         +rand()&0x7fff)%(n-1)+1,n-1,n)!=1) return 0;
   return 1;
}
```

五、Graph

1. 二分圖匹配（Hopcroft-Karp）$O(m\sqrt{n})$

```
int n,m,conn[1010][1010],Size[1010]={},par[1010];//pair
int bfs[100000],w,l,dist[1010]; //G1=1~n, G2=n+1~2*n Nil=0 INF=2^31-1
bool BFS(){
    l=0; //Check Points in G1
    for(int i=1;i<=n;i++){if(par[i]==NIL){dist[i]=0;bfs[l++]=i;}else{dist[i]=INF;}}
    dist[NIL]=INF;    int nxp,np;
    for(w=0;w!=l;w++){np=bfs[w];
        for(int i=0;i<Size[np];i++){nxp=conn[np][i];
            if(dist[par[nxp]]==INF)
                dist[par[nxp]]=dist[np]+1,bfs[l++]=par[nxp];
        }
    }return dist[NIL]!=INF;
}
bool DFS(int np){ int nxp;
        if(np!=NIL){
            for(int i=0;i<Size[np];i++){nxp=conn[np][i];
                    if(dist[par[nxp]]==dist[np]+1){
                        if(DFS(par[nxp])){par[nxp]=np; par[np]=nxp; return 1;}
                    }
            }
        }else return 1;
        dist[np]=INF; return 0;
}
int Hopcroft_Karp(){
        for(int i=0;i<=2*n;i++) par[i]=NIL;
        int ans=0;
        while(BFS()){
            for(int i=1;i<=n;i++) if(par[i]==NIL) if(DFS(i))ans++;
        } return ans;
}
```

2. 二分圖匹配（Hungry Method）$O(mn)$

```
int maze[502][502]={0},my[502]={0},v[502]={0},n,k,xx,yy,cnt;
bool chk(int x){
    if(v[x]) return 0;
    v[x]=1;
    for(int i=1;i<=n;i++){
        if(maze[x][i]&&(!my[i]||chk(my[i]))){
            my[i]=x;
            return 1;
        }
    }
    return 0;
}
int main(){
    cnt=0;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++) v[j]=0;
         if(chk(i)) cnt++;
    }
}
```

3. Maximum Flow（EK）

```
int max_flow(){
    int ans=0;
    int bfs[1000000],w,l,path[5050];
    while(1){
        memset(path,-1,sizeof(path));
        bfs[0]=S; path[S]=-2;
        for(w=0,l=1;w!=l;w++){
            for(int i=1;i<=n;i++){
                if(cost[bfs[w]][i] && path[i]==-1){
                    path[i]=bfs[w];    bfs[l++]=i;
                }
            }
            if(path[T]!=-1) break;
        }
        if(w==l) break;
        int mn=2147483647;
        for(int find=T;find!=S;find=path[find]) mn<?=cost[path[find]][find];
        ans+=mn;
        for(int find=T;find!=S;find=path[find]){
            cost[path[find]][find]-=mn;
            cost[find][path[find]]+=mn;
        }
    }
            return ans;
}
```

4. Minimum cost Maximum Flow

```
int min_cost(){//S-->0, T-->bn+tn+1;
    int mn=0,dist[110];
    int bfs[100000],w,l,path[110],aug[110];
    int S=0,T=bn+tn+1;
    while(1){
        for(int i=0;i<=T;i++) dist[i]=9999999,path[i]=-1,aug[i]=0;
        bfs[0]=0; aug[0]=1; path[0]=-1; dist[0]=0;
        for(w=0,l=1;w!=l;w++){
            aug[bfs[w]]=0;
            for(int np=S;np<=T;np++){
                if(flow[bfs[w]][np] && dist[np]>dist[bfs[w]]+cost[bfs[w]][np]){
                    dist[np]=dist[bfs[w]]+cost[bfs[w]][np];   path[np]=bfs[w];
                    if(!aug[np]){ aug[np]=1; bfs[l++]=np; }
                }
            }
        }
        if(path[T]==-1) break;
        int find=T;
        while(find!=S){
            mn+=cost[path[find]][find];
            flow[path[find]][find]=0; flow[find][path[find]]=1;
            find=path[find];
        }
    }
    return mn;
}
```

3. 二分圖帶權匹配（Hungry Method）

```
#define INF 2147483647
int m,n;
int conn[220][220],Size[220];
int ori[220][220],dist[220][220],isconn[220][220],mx[220],my[220];
bool cho[220],match[220][220],v[220],cx[220],cy[220];
int ans,anscnt;//解,匹配數
bool Hungry(int x){
    if(v[x]) return 0;
    v[x]=1; int nxp;
    for(int i=0;i<Size[x];i++){
        nxp = conn[x][i];
        if(my[nxp]==-1 || Hungry( my[nxp] )){
            my[nxp] = x; mx[x] = nxp; return 1;
        }
    }
    return 0;
}
void Trace(int x,bool way){
    if(v[x]) return;
    v[x] = 1; cho[x]=1;    int nxp;
    for(int i=0;i<Size[x];i++){
        nxp = conn[x][i];
        if(match[x][nxp] == way)
            Trace(nxp,!way);
    }
}
void Adjust(){
    int tmp;
    for(int i=0;i<m;i++){
        tmp = INF;
        for(int j=0;j<n;j++) tmp<?=dist[i][j];
        for(int j=0;j<n;j++) dist[i][j]-=tmp;
    }
    for(int j=0;j<n;j++){
        tmp = INF;
        for(int i=0;i<m;i++) tmp<?=dist[i][j];
        for(int i=0;i<m;i++) dist[i][j]-=tmp;
    }
}
void init(){for(int i=0;i<m;i++)for(int j=0;j<n;j++) dist[i][j]=INF;}
inline void Add(int a,int b){conn[a][Size[a]++]=b;conn[b][Size[b]++]=a;}
void Solve(){
    int cnt,mn;
    Adjust();
    while(1){
        for(int i=0;i<m+n;i++)  Size[i]=0,mx[i]=my[i]=-1,cho[i]=cx[i]=cy[i]=0;
        for(int i=0;i<m;i++) for(int j=0;j<n;j++)if(dist[i][j]==0)Add(i,j+m);
        cnt = 0;
        for(int i=0;i<m;i++){
            memset(v,0,sizeof(v));
            if(Hungry(i))cnt++;
        }
        if(cnt==n) break;
        memset(match,0,sizeof(match));
        for(int i=0;i<m;i++)if(mx[i]!=-1)match[i][mx[i]]=match[mx[i]][i]=1;
        for(int i=0;i<m;i++){
            if(mx[i]==-1){
                memset(v,0,sizeof(v));
                Trace(i,0);
            }
        }
        for(int i=0;i<m;i++) if(cho[i]==0)    cx[i]=1;
        for(int i=0;i<n;i++) if(cho[i+m]==1) cy[i]=1;
        mn = INF;
        for(int i=0;i<m;i++)
            for(int j=0;j<n;j++)
                if(cx[i]==0 && cy[j]==0 && isconn[i][j])
                    mn<?=dist[i][j];
        for(int i=0;i<m;i++)
            for(int j=0;j<n;j++)
                if(cx[i]==0 && cy[j]==0) cost[i][j]-=mn;
                else if(cx[i]==1 && cy[j]==1) cost[i][j]+=mn;
    }
    ans = 0;
    for(int i=0;i<m;i++)if(ori[i][mx[i]-m] != INF) ans+=ori[i][mx[i]-m];
}
```

6. All Pairs Maximum Flow（Gomory-Hu Tree）

```
vector<int> Tree[220],Weight[220];
int T,n,ori[220][220],cost[220][220],Sink[220],f=0;
int bfs[400000],pre[220],w,l,np,wei[220],ans[220][220]={};
void MaxFlow_to_Gomory_HU_Tree(int s){
  for(int i=0;i<n;i++) for(int j=0;j<n;j++) cost[i][j]=ori[i][j];
  int t=Sink[s],mn,mxflow=0;
  while(1){
    memset(pre,-1,sizeof(pre));     bfs[0]=s;   pre[s]=-2;
    for(w=0,l=1;w!=l;w++){
      np=bfs[w];
      for(int i=0;i<n;i++){if(pre[i]==-1 && cost[np][i]){pre[i]=np;bfs[l++]=i;}}
      if(pre[t]!=-1) break;
    }
    if(w==l) break;
    mn=2147483647;
    for(int find=t;find!=s;find=pre[find]) mn<?=cost[pre[find]][find];
    mxflow+=mn;
    for(int find=t;find!=s;find=pre[find]){
      cost[pre[find]][find]-=mn;   cost[find][pre[find]]+=mn;
    }
  }
  wei[s]=mxflow;
  for(int i=0;i<n;i++) if(i!=s && pre[i]!=-1 && Sink[i]==t) Sink[i]=s;
  if(pre[Sink[t]]!=-1){
    Sink[s]=Sink[t]; Sink[t]=s; wei[s]=wei[t]; wei[t]=mxflow;
  }
}
void Make_Tree(){
  memset(Sink,0,sizeof(Sink)); memset(wei,0,sizeof(wei));
  for(int i=1;i<n;i++) MaxFlow_to_Gomory_HU_Tree(i);
  for(int i=0;i<n;i++){
    if(i!=Sink[i]){
      Tree[i].push_back(Sink[i]); Weight[i].push_back(wei[i]);
      Tree[Sink[i]].push_back(i); Weight[Sink[i]].push_back(wei[i]);
    }
  }
}
int MaxFlow(int s,int t,int P,int W){
  if(s==t) return W;
  int mn=2147483647;   int Size=Tree[s].size();
  for(int i=0;i<Size;i++)if(Tree[s][i]!=P)
    mn<?=MaxFlow(Tree[s][i],t,s,MIN(W,Weight[s][i]));
  return mn;
}
```

9. Bi-Connected Component to Find AP(DFS number and low value)

```
void dfnlow(int u,int v){
  vector<int>::iterator ptr;
  dfn[u]=low[u]=cnt++;
  for(ptr=conn[u].begin();ptr!=conn[u].end();ptr++){
    if(dfn[*ptr]<0){
      dfnlow(*ptr,u);
        if(low[*ptr]>=dfn[u]) cc[u]++;
        low[u]=MIN(low[u],low[*ptr]);
      }else if((*ptr)!=v)   low[u]=MIN(low[u],low[*ptr]);
   }
   if(u==root && cc[u]>1) ans.push_back(u);
   else if(u!=root && cc[u]) ans.push_back(u);
}
```

9. Bi-Connected Component to Find bridge

```
void dfnlow(int u,int v){
  vector<int>::iterator ptr;
  dfn[u]=low[u]=cnt++;
  char chk=0;
  int Size=conn[u].size(),np;
  for(int i=0;i<Size;i++){ np=conn[u][i];
    if(dfn[np]<0){
      dfnlow(np,u);
      if(low[np]>=dfn[u]) chk++;
      low[u]=MIN(low[u],low[np]);
      if(low[np]>dfn[u]) ans.push_back(id[u][i]);
     }else if(np!=v) low[u]=MIN(low[u],low[np]);
   }
   if(u==root && chk>1) is[u]=1;
   else if(u!=root && chk) is[u]=1;
}
```

7. Minimum Cut(Stoer-Wagner Algorithm)

```
int T,m,last,f=0,W,mx,mxp,s,t, tra[151],ans;
bool v[151];
vector<int> set;
struct Graph{ int n,w[151][151];}G[2];
void Find(int p){
   memset(v,0,sizeof(v));    v[0]=1;
   set.clear();       set.push_back(0);       t=0;
   for(int cnt=1;cnt<G[p].n;cnt++){
      mx=-2147483647; mxp=-1;
      for(int i=0;i<G[p].n;i++){
         if(v[i]) continue;
         W=0;
         for(int j=0;j<cnt;j++) W+=G[p].w[i][set[j]];
         if(W>mx){mx=W;mxp=i; }
      }
      v[mxp]=1; set.push_back(mxp); s=t; t=mxp;
   }
   W=0;
   for(int i=0;i<G[p].n;i++)
      if(i!=t)    W+=G[p].w[t][i];
   ans<?=W;
}
int main(){
   int a,b,c,nxt;
   scanf("%d",&T);
   while(T--){
      ans=2147483647;
      scanf("%d%d",&G[0].n,&m);
      memset(G[0].w,0,sizeof(G[0].w));
      while(m--){
         scanf("%d%d%d",&a,&b,&c); a--; b--;
         G[0].w[a][b]+=c;     G[0].w[b][a]+=c;
      }
      for(int now=0;G[now].n>1;now=(now+1)&1){
         Find(now);            nxt=(now+1)&1;        G[nxt].n=0;
         for(int i=0;i<G[now].n;i++) if(i!=s && i!=t)    tra[i]=G[nxt].n++;
         for(int i=0;i<G[now].n;i++) if(i!=s && i!=t)    for(int j=0;j<G[now].n;j++)
                             if(j!=s && j!=t) G[nxt].w[tra[i]][tra[j]]=G[now].w[i][j];
         c=G[nxt].n;
         for(int i=0;i<G[now].n;i++){
            if(i==s || i==t) continue;
            G[nxt].w[c][tra[i]]=G[nxt].w[tra[i]][c]=G[now].w[s][i]+G[now].w[t][i];
         }
         G[nxt].n++;
      }
      printf("Case #%d: %d\n",++f,ans);
   }
}
```

8. Strongly Connected Component(Kosaraju)

```
inline void DFS(int np){
   if(v[np]) return; v[np]=1;
   for(int i=0;i<ed[np].size();i++) DFS(ed[np][i]);
   sta[cnt++]=np;
}
inline void Kosa(int np){
   if(v[np]) return; v[np]=1;
   color[np]=++cnt;
   SCC[cSCC++]=np;
   for(int i=0;i<back_ed[np].size();i++)    Kosa(back_ed[np][i]);
}
```

10. Lowest Common Ancestor

```
Int n,dep[10010],cnt;
int L[20010],R[20010],E[20010],P[10010],d[20010][15],PP[10010][15];//RMQ
vector<int> conn[10010];bool v[10010];
void DFS(int np,int depth){
   if(v[np]) return;
   dep[np]=depth;
   v[np]=1;     R[np]=cnt;    L[cnt]=depth;    E[cnt++]=np;
   for(int i=0;i<conn[np].size();i++)
      if(v[conn[np][i]]==0){
         P[conn[np][i]]=np;
         DFS(conn[np][i],depth+1);
         L[cnt]=depth;  E[cnt++]=np;
      }
}
void RMQ_ST(){//RMQ
   for(int i=0;(1<<i)<=cnt;i++)
      for(int j=0;j+(1<<i)<=cnt;j++)
         if(!i) d[j][i]=j;
         else if(L[d[j][i-1]]<L[d[j+(1<<(i-1))][i-1]])d[j][i]=d[j][i-1];
         else d[j][i]=d[j+(1<<(i-1))][i-1];
}
int RMQ(int a,int b){
         int k=0;
         while((1<<(k+1))<(b-a+1)) k++;
         if(L[d[a][k]]<L[d[b-(1<<k)+1][k]]) return E[d[a][k]];
         else return E[d[b-(1<<k)+1][k]];
}
int LCA(int a,int b){ //LCA
         if(R[a]<R[b]) return RMQ(R[a],R[b]);
         else return RMQ(R[b],R[a]);
}
void init(){ cnt=0; DFS(0,1); RMQ_ST();}
```

11. General Graph Matching

```
bool conn[110][110],flag[110],in[110];
int block[110],mate[110],path[110];
int que[10000],w,l;
void Modify(int u,int LCA){ int v;
      while(block[u]!=LCA){
         v = mate[u];
         flag[block[u]]=flag[block[v]]=1;
         u = path[v];
         if(block[u]!=LCA) path[u]=v;
      }
}
void Contract(int u,int v,int s){
      int LCA; memset(flag,0,sizeof(flag));
      for(LCA = u;    1    ;LCA = path[mate[LCA]]){
         LCA = block[LCA];flag[LCA]=1;
         if(LCA==s) break;
      }
      for(LCA = v; 1 ;LCA = path[mate[LCA]]){
         LCA = block[LCA];
         if(flag[LCA]) break;
      }
      memset(flag,0,sizeof(flag));
      Modify(u,LCA);       Modify(v,LCA);
      if(block[u]!=LCA) path[u] = v;
      if(block[v]!=LCA) path[v] = u;
      for(int i=0;i<n;i++){
         if(flag[block[i]]){
               block[i] = LCA;
               if(!in[i]){in[i] = 1; que[l++]=i; }
         }
      }
}
int Find(int s){
   int now,nxt,tmp;
   memset(in,0,sizeof(in));       memset(path,-1,sizeof(path));
   for(int i=0;i<n;i++) block[i] = i;
   que[0] = s; in[s] =1;
   for(w=0,l=1;w!=l;w++){    now = que[w]; in[now] = 0;
      for(nxt=0;nxt<n;nxt++){
         if(conn[now][nxt]&&block[now]!=block[nxt]&&mate[now]!=nxt){
            if(nxt==s || (mate[nxt]!=-1 && path[mate[nxt]]!=-1)){
               Contract(now,nxt,s);
            }else if(path[nxt]==-1){
               path[nxt] = now;
               if(mate[nxt]==-1){
                  while(nxt!=-1){
                     now = path[nxt];     tmp = mate[now];
                     mate[now] = nxt;    mate[nxt] = now;         nxt = tmp;
                  }
                  return 1;
               }
               in[mate[nxt]]=1;    que[l++]=mate[nxt];
            }
         }
      }
   }
      return 0;
}
```

```
int Matching(){
    int ans=0;
    memset(mate,-1,sizeof(mate));
    for(int i=0;i<n;i++){
        if(mate[i]==-1 && Find(i)){
            ans++;
        }
    }
    return ans;
}
```

12. Directed MST

```
int T,n,m,f=0;
bool v[1010];
struct EDGE{int st,ed,val;}E[40010];
vector<int> conn[1010];
void DFS(int np){
  v[np]=1; int Size=conn[np].size();
  for(int i=0;i<Size;i++) if(!v[conn[np][i]]) DFS(conn[np][i]);
}//  連入最小 cost,最小 cost 來源,造訪判重,水母
int mcost[1010],pre[1010],visit[1010],jelly[1010];
bool con[1010];//是否收縮
int DMST(){
  memset(v,0,sizeof(v));
  DFS(0);
  for(int i=0;i<n;i++) if(!v[i]) return -1;
  bool isc;//是否有環
  int w1=0,w2=0,np,np2;     memset(con,0,sizeof(con));
  while(1){   w1=0;
    for(int i=0;i<n;i++){
      mcost[i]=2147483647;     pre[i]=-1; visit[i]=-1;     jelly[i]=-1;
    }
    for(int i=0;i<m;i++){
      if(E[i].st!=E[i].ed && E[i].ed!=0 && mcost[E[i].ed]>E[i].val){
        mcost[E[i].ed]=E[i].val;
        pre[E[i].ed]=E[i].st;
      }
    }
    isc=0;
    for(int i=0;i<n;i++){
      if(con[i]) continue;
      if(i!=0 && pre[i]==-1) return -1;
      if(pre[i]>=0) w1+=mcost[i];
      if(visit[i]!=-1) continue;
      for(np=i;np!=-1 && visit[np]==-1;np=pre[np]) visit[np]=i;
      if(np!=-1 && visit[np]==i){
        isc=1;        np2=np;
        while(1){
          jelly[np2]=np;           con[np2]=1;
          w2+=mcost[np2];        np2=pre[np2];
          if(np2==np) break;
        }
        con[np]=0;
      }
    }
    if(!isc) break;
    for(int i=0;i<m;i++){
      if(jelly[E[i].ed]!=-1)    E[i].val-=mcost[E[i].ed];
      if(jelly[E[i].st]!=-1)    E[i].st=jelly[E[i].st];
      if(jelly[E[i].ed]!=-1)    E[i].ed=jelly[E[i].ed];
      if(E[i].st==E[i].ed)      E[i--]=E[--m];
    }
  }
  return w1+w2;
}
```

六、Data Structure

1. RMQ

```
int num[50010],RMQ_MX[50010][16],RMQ_MN[50010][16],n;
inline void RMQ(){
  int m=(int)floor(log2(n));
  for(int i=1;i<=n;i++) RMQ_MX[i][0]=RMQ_MN[i][0]=num[i];
  for(int j=1;j<=m;j++){
    int k=(1<<(j-1));
    for(int i=1;i<=n-k;i++){
      RMQ_MX[i][j]=MAX(RMQ_MX[i][j-1],RMQ_MX[i+k][j-1]);
      RMQ_MN[i][j]=MIN(RMQ_MN[i][j-1],RMQ_MN[i+k][j-1]);
    }
  }
}
inline int Query(int a,int b){
 int k=(int)log2(b-a+1);
 return MAX(
  RMQ_MX[a][k],
  RMQ_MX[b-(1<<k)+1][k])-MIN(RMQ_MN[a][k],RMQ_MN[b-(1<<k)+1][k]);
}
```

2. Binary Indexed Tree

```
#define LOWBIT(x) ((x)&(-(x)))
int B[10000],C[10000];
void bit_update(int *a, int p, int d) {
  for ( ; p && p < MAXN ; p += LOWBIT(p))
    a[p] += d;
}
int bit_query(int *a, int p) {
  int s = 0;
  for ( ; p ; p -= LOWBIT(p))   s += a[p];
  return s;
}
void bit_update2(int *a, int p, int d) {
  for ( ; p ; p -= LOWBIT(p))   a[p] += d;
}
int bit_query2(int *a, int p) {
  int s = 0;
  for ( ; p && p < MAXN ; p += LOWBIT(p)) s += a[p];
  return s;
}
void _insert(int p, int d) {
  bit_update(B, p, p*d);   bit_update2(C, p-1, d);
}
int _query(int p) {
  return bit_query(B, p) + bit_query2(C, p) * p;
}
inline void insert_seg(int a, int b, int d) {
  _insert(a-1, -d);   _insert(b, d);
}
inline int query_seg(int a, int b) {
  return _query(b) - _query(a-1);
}
```

七、Strings

1. KMP

```
inline void PreProcess(int L,int R){
	alen=0;
	for(int i=L;i<=R;i++) ss[++alen]=s[0][i];
	P[1]=ptr=0;
	for(int i=2;i<=alen;i++){
		while(ptr>0 && ss[ptr+1]!=ss[i])	ptr=P[ptr];
		if(ss[ptr+1]==ss[i])ptr++;
		P[i]=ptr;
	}
}
inline bool KMP(int m){
	ptr=0; len=strlen(s[m]+1);
	for(int i=1;i<=len;i++){
		while(ptr>0 && ss[ptr+1]!=s[m][i])	ptr=P[ptr];
		if(ss[ptr+1]==s[m][i]) ptr++;
		if(ptr==alen) return 1;
	}
	return 0;
}
```

2. Suffix Array

```
char S[100010];    int len,SA1[100010],SA2[100010],
Rank1[100010],Rank2[100010],Bucket[100010]={},tmp;
int main(){
    for(int i=0;i<len;i++) Bucket[S[i]+128]++; //Find SA1 by Counting Sort
    for(int i=1;i<256;i++) Bucket[i]+=Bucket[i-1];
    for(int i=0;i<len;i++) SA1[--Bucket[S[i]+128]]=i;
    Rank1[SA1[0]]=0; //Calc Rank(1) Side by Side
    for(int i=1;i<len;i++)Rank1[SA1[i]]=Rank1[SA1[i-1]]+(S[SA1[i]]!=S[SA1[i-1]]);
    for(int k=1,k0;k<len;k*=2){ k0=(k>>1);
        for(int i=0;i<len;i++) Bucket[Rank1[SA1[i]]]=i;
        for(int i=len-1;i>=0;i--)
            if(k<=SA1[i]) SA2[Bucket[Rank1[SA1[i]-k]]--]=SA1[i]-k;
        for(int i=len-k;i<len-k0;i++) SA2[Bucket[Rank1[i]]--]=i;
        Rank2[SA2[0]]=0;
        for(int i=1;i<len;i++)
            Rank2[SA2[i]]=Rank2[SA2[i-1]]+
      (Rank1[SA2[i]]!=Rank1[SA2[i-1]] || Rank1[SA2[i]+k]!=Rank1[SA2[i-1]+k]);
        for(int i=0;i<len;i++){SA1[i]=SA2[i]; Rank1[i]=Rank2[i];}
    }
}
```