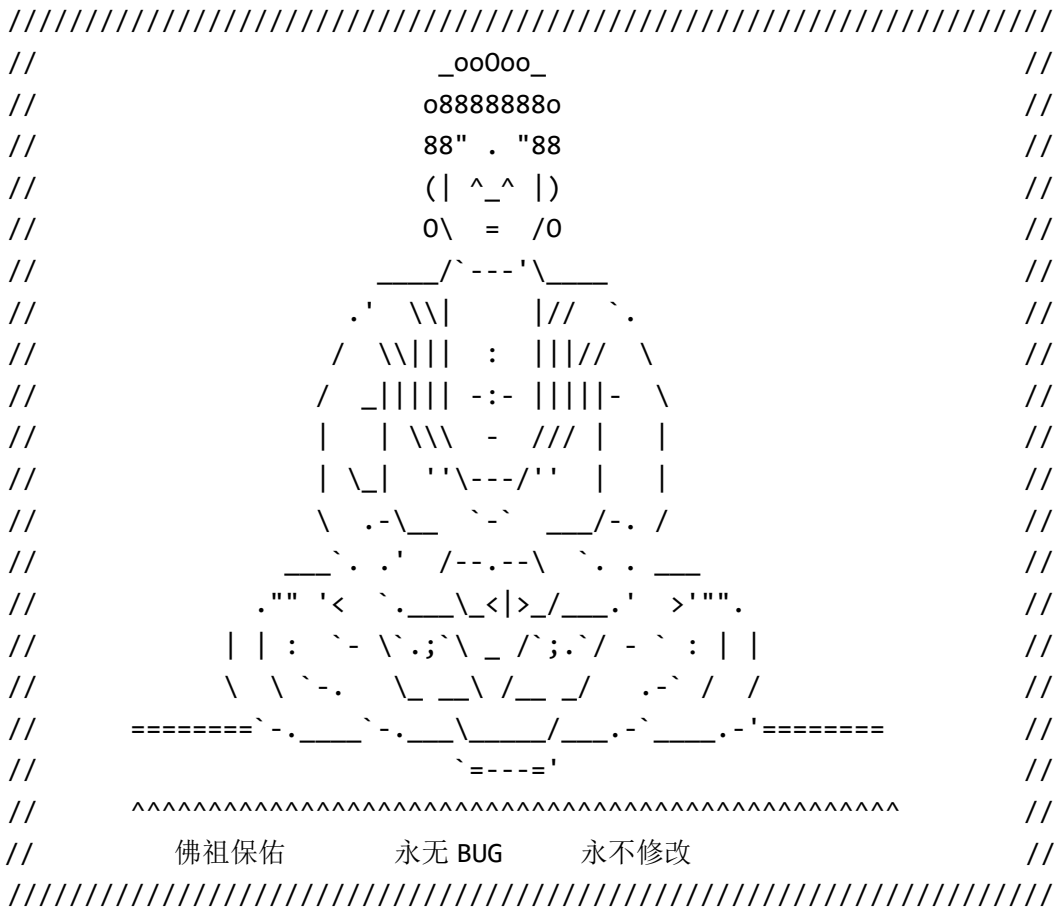

laonianrencaozuo

2018.8.30

1.fread 输入挂

2.手写 hash_table

3.次短路

4.网络流

5. 01 分数规划

6. 1 到 x 范围内，因子个数的和

7.中国剩余定理

8.LGV

9.x+y==n 的解的个数

10.矩阵快速幂

11.卡特兰数

12.求因子个数

13.三角形内心

14.三角形外心

15.辛普森积分

16.常见 STL

17.单调队列

18.splay

19.二维树状数组

20.权值线段树

21.矩形面积交

22.矩形周长并

23.ac 自动机

24.hash 字符串

25.失配优化的 kmp

26.SA

27.SAM 区间不同子串个数 DP

28.SAM 求 LCS1

29.SAM 求 LCS2

30.SAM 求 right 集合大小

31.trie

32.最长回文子序列，回文序列个数

33.二维线段树

34.CDQ 分治求三维 LIS

35.CDQ 分治求矩阵内数字和

36.多边形面积交与并

37.模拟退火求最小费马点

38.最小圆覆盖

39.最小矩形覆盖

40.凸包上最大三角形面积

41.凸包直径

42.pb_ds

1. fread 输入挂

```
inline char nc(){
    static char buf[100000],*p1=buf,*p2=buf;
    return
p1==p2&&(p2=(p1=buf)+fread(buf,1,100000,stdin),p1==p2)?EOF:*p1
++;
}
template<class T>
inline void read(T &sum){
    char ch=nc();sum=0;
    while(!(ch>='0'&&ch<='9'))ch=nc();
    while(ch>='0'&&ch<='9')sum=sum*10+ch-48,ch=nc();
}
template<class T>
inline void print(T x){
    if(x>9)print(x/10);putchar(x%10+'0');
```

2.手写 hash_table

```
const int maxsize = 1000007;//如果内存允许并且单样例，随便开，否则
maxc/10 ~ maxc
const int maxc = 1e6+5;//要大于最大状态个数
struct e{
    int to,nxt,val,cnt;
};
struct hash_table_for_map_int_int{
    int cnt=0;
    int head[maxsize];
    e edge[maxc+10];
    int h(int x){
        return abs(((x)^(x<<1)^(x>>1))%maxsize);
    }
    void Insert(int id,int val){
        for(int i=head[id];i!=0;i=edge[i].nxt){
            if(edge[i].to==val)
                edge[i].cnt++;return;
        }
        edge[++cnt].to=val;
        edge[cnt].cnt=1;
        edge[cnt].nxt=head[id];
        head[id]=cnt;
    }
    void drop(int id,int val,int num){
        for(int i=head[id];i!=0;i=edge[i].nxt){
            if(edge[i].to==val){
                edge[i].cnt=num;return;
            }
        }
    }
    int query(int id,int val){
        for(int i=head[id];i!=0;i=edge[i].nxt){
            if(edge[i].to==val)
                return edge[i].cnt;
        }return 0;
    }
    void init(){
        cnt=0;
        memset(head,0,sizeof head);
    }
};
hash_table_for_map_int_int mp;
```

3.次短路

```
#include <bits/stdc++.h>
#define INF 1e16
#define LL long long
using namespace std;
struct edge{
    int to;
    LL cost;
    edge(int tv = 0,LL tc = 0):to(tv),cost(tc){}
};
int N,R;
typedef pair<LL ,LL> P;
vector<edge> graph[100010];
LL dist[100010],dist2[100010];
LL solve()
{
    fill(dist,dist+N,INF);
    fill(dist2,dist2+N,INF);
    priority_queue<P, vector<P> ,greater<P> > q;
    dist[0] =0;
    q.push(P(0,0));
    while(!q.empty())
    {
        P p =q.top();

        q.pop();
        LL v=p.second,d=p.first;
        if(dist2[v] < d) continue;
        for(int i=0 ; i<graph[v].size();i++)
        {
            edge &e=graph[v][i];
            LL d2=d+e.cost;
            if(dist[e.to]>d2)
            {
                swap(dist[e.to],d2);
                q.push(P(dist[e.to],e.to));
            }
            if(dist2[e.to]>d2&&dist[v]<d2){
                dist2[e.to]=d2;
                q.push(P(dist2[e.to],e.to));
            }
        }
    }
```

```
    return dist2[N-1];
}
int main(){
    int t,n,m;
    cin>>t;
    while(t--)
    {
        cin>>N>>R;
        for(int i=0;i<N;i++) graph[i].clear();
        for(int i=0;i<R;i++)
        {
            LL a,b,c;
            scanf("%lld %lld %lld",&a,&b,&c);
            graph[a-1].push_back(edge(b-1,c));
            graph[b-1].push_back(edge(a-1,c));
        }
        printf("%lld\n",solve());
    }
    return 0;
}
```

4. 网络流

```
#include<bits/stdc++.h>
using namespace std;
int
level[80010],n,m,cnt=-1,head[80010],ans,s,t,k,mp[110][110],hed
[81000];
struct e{
    int to,next,c;
}edge[800010];
void add(int a,int b,int c)
{
    edge[++cnt].to=b;
    edge[cnt].next=head[a];
    head[a]=cnt;
    edge[cnt].c=c;
    edge[++cnt].to=a;
    edge[cnt].next=head[b];
    head[b]=cnt;
    edge[cnt].c=0;
}
int spfa()
{
    memset(level,-1,sizeof(level));
    queue<int> q;
    q.push(s);
    while(!q.empty())
    {
        int u=q.front();
        for(int i=head[u];i!=-1;i=edge[i].next)
        {
            int v=edge[i].to;
            if(level[v]==-1&&edge[i].c!=0&&v!=s)
            {
                level[v]=level[u]+1;
                q.push(v);
                if(v==t) return 1;
            }
        }
        q.pop();
    }
    return 0;
}
int dfs()
```

```
{
    int stac1[20100],stac2[21010],top=0,res=0;
    stac1[top++]=s;int tr=10000000,pos=-1;
    while(1)
    {
        int trr=10000000;
        if(stac1[top-1]==t)
        {
            for(int i=0;i<top-1;i++)
            {
                if(tr>edge[stac2[i]].c)
                {
                    tr=edge[stac2[i]].c;
                    pos=i;
                }
            }
            for(int i=0;i<top-1;i++)
            {
                edge[stac2[i]].c-=tr;
                edge[stac2[i]^1].c+=tr;
                if(i<pos)
                {
                    if(trr>edge[stac2[i]].c)
                    {
                        trr=edge[stac2[i]].c;
                        pos=i;
                    }
                }
            }
            res+=tr;
            tr=trr;
            top=pos+1;
        }
        int j;
        for(j=head[stac1[top-1]];j!=-1;j=edge[j].next)
        {
            int u=stac1[top-1],v=edge[j].to;
            if(level[v]==level[u]+1&&edge[j].c)
            {
                stac2[top-1]=j;
                stac1[top++]=v;
                break;
            }
        }
```

```
        if(j==-1)
        {
            if(stac1[top-1]==s) break;
            else{
                level[stac1[top-1]]=-1;
                top--;
            }
        }

    }
    return res;
}
int maxflow()
{
    int res=0;
    while(spfa()) res+=dfs();
    return res;
}
int main()
{
    int T,cas=0;
    cin>>T;
    while(T--)
    {
        memset(head,-1,sizeof(head));
        cnt=-1;
        cin>>n;
        char str[120];
        int ca[11],cb[11],sum=0;
        scanf("%s",str);
        for(int i=0;i<=9;i++)
        {
            scanf("%d %d",&ca[i],&cb[i]);
        }
        for(int i=1;i<=n;i++)
        {
            for(int j=1;j<=n;j++)
            {
                scanf("%d",&mp[i][j]);
            }
        }
        s=0;
        t=n*n+n+11;
        for(int i=0;i<n;i++)
```

```
        {
            hed[i+1]=-1*ca[str[i]-'0'];
            add(i+1,n*n+n+1+str[i]-'0',1000);
        }
        for(int i=1;i<=n;i++)
        {
            for(int j=1;j<=n;j++)
            {
                hed[j+i*n]=mp[i][j];
                add(j+i*n,i,1000);
                add(j+i*n,j,1000);
            }
        }
        for(int i=0;i<10;i++)
        {
            hed[n*n+n+1+i]=-1*(cb[i]-ca[i]);
        }
        for(int i=s+1;i<t;i++)
        {
            sum+=max(hed[i],0);
            if(hed[i]<0) add(i,t,-1*hed[i]);
            if(hed[i]>0) add(s,i,hed[i]);
        }
        printf("Case #%d: %d\n",++cas,sum-maxflow());
    }
}
```

5. 01 分数规划

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=100010;
const double eps=1e-6;
int n,k,Max;
double m;
struct Node{
    int s,c;
    bool operator<(const Node&p)const{
        return s*(c-m)>p.s*(p.c-m);
    }
}arr[maxn];
inline bool check(){
    nth_element(arr,arr+k,arr+n);
    double ans=0;
    for(int i=0;i<k;i++) ans+=arr[i].s*(arr[i].c-m);
    return !(ans<-eps);
}
inline double solve(){
    double l=0,r=Max+2;
    while(r-l>eps){
        double mid=(l+r)/2;
        m=mid;
        if(check()) l=mid;
        else r=mid;
    }
    return (l+r)/2;
}
int main(){
    while(~scanf("%d%d",&n,&k)){
        k=n-k;
        Max=0;
        for(int i=0;i<n;i++) scanf("%d",&arr[i].s);
        for(int i=0;i<n;i++){
            scanf("%d",&arr[i].c);
            Max=max(Max,arr[i].c);
        }
        printf("%.10lf\n",solve());
    }
    return 0;
}
```

6. 1 到 x 范围内，因子个数的和

```
int n;cin>>n;
ll ans=0;
for(int i=1;i*i<=n;i++){
    ans += (n/i)*2;
}
cout<<ans-(int)sqrt(n)*(int)sqrt(n)<<endl;
```

7. 中国剩余定理

```
ll n,m,b[10009],a[10009];
ll gcd(ll a,ll b){
   return b? gcd(b,a%b):a;
}
void exgcd(ll a,ll b,ll &d,ll &x,ll &y){
   if (b==0){d=a;x=1;y=0;return ;}
   exgcd(b,a%b,d,y,x);y-=x*(a/b);
}
int main(){
   int T;
   scanf("%d",&T);
   for (int kk=1;kk<=T;kk++){
      scanf("%lld%lld",&n,&m);
      bool o=true;
      ll m1,m2,a1,a2;
      for (int i=1;i<=m;i++) scanf("%lld",&b[i]);
      for (int i=1;i<=m;i++) scanf("%lld",&a[i]);
      a1=a[1];m1=b[1];
      for (int i=2;i<=m;i++){
         ll d,x,y;
         m2=b[i];a2=a[i];//还是按这样写中国剩余定理吧!
         exgcd(m1,m2,d,x,y);
         if ((a2-a1)%d) o=false;
         x*=(a2-a1)/d;
         ll k=m2/d;
         x=(x%k+k)%k;
         a1=m1*x+a1;
         m1=m1/d*m2;
         a1=(a1%m1+m1)%m1;
      }
      if (a1==0){
         a1=1;
```

```
            for (int i=1;i<=m;i++) a1=a1/gcd(a1,b[i])*b[i];
        }//特殊情况，如果余数都是 0
        ll ans=0;
        ans=n/m1;
        n=n%m1;
        if(n>=a1) ans++;//计算答案
        if (o) printf("%lld\n",ans);else printf("0\n");
    }
    return 0;
}
//注意 b[i]和 a[i]都不能为 0，不然会石雕
//适用与 b 不是两两互质的情况下
```

8. LGV

起点(n,0)和(n-1,-1)，终点(0,m)和(-1,m-1)的严格不相交路径

```
C(n+m,n) * C(n+m,n) - C(n+m,m-1) * C(n+m,n-1)
```

9. x+y==n 的解的个数

```
int n,p=0;
memset(num,0,sizeof(num));
scanf("%d",&n);
for(int i=2;i*i<=n;++i){
   if(n%i==0){
   while(n%i==0){
      ++num[p];
      n/=i;
   }
   ++p;
   }
}
if(n!=1){
   ++num[p];
   ++p;
}
for(int i=0;i<p;++i)
   num[i]*=2;
int ans=1;
for(int i=0;i<p;++i)
   ans*=(num[i]+1);
printf("%d\n",(ans+1)/2);
```

10. 矩阵快速幂

```
#define ll long long
#define mod 1000000007
struct mat{
    ll a[2][2];
    mat(){memset(a,0,sizeof(a));}
    mat operator * (const mat& b)const{
        mat ret=mat();
        for(int i=0;i<2;i++){
            for(int j=0;j<2;j++){
                for(int k=0;k<2;k++){
            ret.a[i][j]=(ret.a[i][j]+a[i][k]*b.a[k][j]+mod)%mod;
                }
            }
        }return ret;
    }
    mat operator *= (const mat& b){
        return *this = *this*b;
    }
};
mat qpow(mat a,ll n){
    mat ret=mat();
    for(int i=0;i<2;i++)ret.a[i][i]=1;
    while(n){
        if(n&1){
            ret*=a;
        }a*=a;
        n>>=1;
    }return ret;
}
```

11. 卡特兰数

```
fac[2*n]*qmod(fac[n+1], mod-2)%mod*qmod(fac[n], mod-2)%mod;
```

12. 求因子个数

```
#define M 1000010
int prime[M],e[M],div_num[M];
int d[M];
bool flag[M];
void init()
{
    int i,j,k;
    memset(flag,false,sizeof(flag));
    k=0;
    div_num[1]=1;
    for(i=2; i<M; i++)
    {
        if(!flag[i])
        {
            prime[k++]=i;
            e[i]=1;
            div_num[i]=2;
        }
        for(j=0; j<k&&i*prime[j]<M; j++)
        {
            flag[i*prime[j]]=true;
            if(i%prime[j]==0)
            {
               div_num[i*prime[j]]=div_num[i]/(e[i]+1)*(e[i]+2);
                e[i*prime[j]]=e[i]+1;
                break;
            }
            else
            {
               div_num[i*prime[j]]=div_num[i]*div_num[prime[j]];
                e[i*prime[j]]=1;
            }
        }
    }
    for(int i=1;i<M;i++)
        for(int j=i;j<M;j+=i)
            d[j]++;
    for(int i=1;i<M;i++)
        if(d[i]!=div_num[i])cout<<i<<endl;
}
```

13.三角形内心

内心是角平分线的交点,到三边距离相等.
设:在三角形 ABC 中,三顶点的坐标为: A(x1,y1),B(x2,y2),C(x3,y3)
BC=a,CA=b,AB=c
内心为 M （X,Y）则有 aMA+bMB+cMC=0（三个向量）
MA=(X1-X,Y1-Y)
MB=(X2-X,Y2-Y)
MC=(X3-X,Y3-Y)
则：a(X1-X)+b(X2-X)+c(X3-X)=0,a(Y1-Y)+b(Y2-Y)+c(Y3-Y)=0
∴X=(aX1+bX2+cX3)/(a+b+c),Y=(aY1+bY2+cY3)/(a+b+c)
∴M((aX1+bX2+cX3)/(a+b+c),(aY1+bY2+cY3)/(a+b+c))

14.三角形外心

```
x=((C1*B2)-(C2*B1))/((A1*B2)-(A2*B1));
y=((A1*C2)-(A2*C1))/((A1*B2)-(A2*B1));
```

15.辛普森积分

```
#define eps 0.0000001
double r1,r2;
double F(double x){
    return 8*sqrt((r1*r1-x*x)*(r2*r2-x*x));
}
double cal(double l, double r)
{
    return (r-l)/6.0*(F(r)+4.0*F((r+l)/2.0)+F(l));
}
double simpson(double l,double r)
{
    double m=(l+r)/2.0;
    double fl=cal(l,m),fr=cal(m,r);
    if(fabs(fl+fr-cal(l,r))<eps) return fr+fl;
    else return simpson(l,m)+simpson(m,r);
}
int main()
{
    scanf("%lf%lf",&r1,&r2);
    double minr;
    if(r1<r2) minr=r1;
    else minr=r2;
    printf("%.6f\n",simpson(0,minr));
}
```

16. 常见 STL

```
nth_element(a+1,a+1+n-m,a+1+n);
rope:
   using namespace __gnu_cxx;
   rope<int> T;
   for(int i=1;i<=n;i++) T.push_back(i);
   T = T.substr(p,s) + T.substr(0,p)+T.substr(p+s,n-p-s);
   for(int i=1;i<=n;i++) printf("%d ",T.at(i));
   可持久化:
   rope<char> *his[maxn];
   his[0] = new rope<char>();
   his[i] = new rope<char>(*his[i-1]);
   O(1)copy 历史版本
   insert(位置,值)
   erase(位置，大小)
   substr(位置，大小)

map<int,deque<int> > mp;
/*合并两个队列*/
if (w == 0) {
    mq[u].insert(mq[u].end(), mq[v].begin(), mq[v].end());
    mq[v].clear();
}
else {
    mq[u].insert(mq[u].end(), mq[v].rbegin(), mq[v].rend());
    mq[v].clear();
}

random_shuffle(p+1,p+1+n);
```

17.单调队列

```
#define N 1000050
#define INF 2147483647
int n,k;
int a[N];
struct Elem{
   int k,num;
}Queue[N];
int l=1,r=1;
inline void GetMin(){
   memset(Queue,0,sizeof Queue);
   Queue[0].k=-INF;
   l=1,r=1;
   for(int i=1;i<=k;i++){
      while(Queue[r].k>=a[i] && r>=l) r--;
      Queue[++r].k=a[i];
      Queue[r].num=i;
    }
   for(int i=k;i<=n;i++){
       while(Queue[r].k>=a[i] && r>=l) r--; //维护单调性
       Queue[++r].k=a[i];
       Queue[r].num=i;
       while(Queue[l].num<=i-k) l++;  //维护队列下标范围 k 以内
       printf("%d ",Queue[l].k);
    }
}
inline void GetMax(){
   memset(Queue,0,sizeof Queue);
   Queue[0].k=INF;
   l=1,r=1;
   for(int i=1;i<=k;i++){
      while(Queue[r].k<=a[i] && r>=l) r--;
      Queue[++r].k=a[i];
      Queue[r].num=i;
    }
   for(int i=k;i<=n;i++){
       while(Queue[r].k<=a[i] && r>=l) r--;
       Queue[++r].k=a[i];
       Queue[r].num=i;
       while(Queue[l].num<=i-k) l++;
       printf("%d ",Queue[l].k);
    }
}
```

18. splay

```
/*
 * 给定一个数列 a1,a2,...an
 * 进行以下 6 种操作
 * ADD x y D :给第 x 个数到第 y 个数加 D(增加一个 add 进行延迟标记)
 * REVERSE x y :反转[x,y]之间的数(伸展树经典操作)
 * REVOLVE x y T:循环右移 T 次(先把 T 对长度进行取模，然后就相当于把[y-T+1,y]放在[x,y-T]前面)
 * INSERT x P:在第 x 个数后面插入 P （经典的插入）
 * DELETE x:删除第 x 个数（删除操作）
 * MIN x y:查询[x,y]之间最小的数(标记)
 * CUT x y z 把区间[l,r]切断，贴到第 z 个元素后面
 * REMOVE 移除根结点
 *
 * 需要的操作:反转、删除、插入、查询区间最小值、成段更新、区间搬移(循环右移转化为区间搬移)
 * 需要的变量:pre,ch,key,size,add,rev,m(最小值)
 */

#include <iostream>
#include <stdio.h>
#include <string.h>
#include <algorithm>
using namespace std;
#define Key_value ch[ch[root][1]][0]
#define Key_Value ch[ch[root][1]][0]
const int MAXN=200010;
const int INF=0x3f3f3f3f;
int
pre[MAXN],ch[MAXN][2],key[MAXN],size[MAXN],add[MAXN],rev[MAXN],
m[MAXN];
int root,tot1;
int s[MAXN],tot2;//内存池、内存池容量
int a[MAXN];
int n,q;

void NewNode(int &r,int father,int k)
{
    if(tot2)r=s[tot2--];
    else r=++tot1;
    ch[r][0]=ch[r][1]=0;
    pre[r]=father;
    size[r]=1;
```

```
    add[r]=rev[r]=0;
    key[r]=m[r]=k;
}
void Update_Rev(int r)
{
    if(!r)return;
    swap(ch[r][0],ch[r][1]);
    rev[r]^=1;
}
void Update_Add(int r,int ADD)
{
    if(!r)return;
    add[r]+=ADD;
    key[r]+=ADD;
    m[r]+=ADD;
}
void Push_Up(int r)
{
    size[r]=size[ch[r][0]]+size[ch[r][1]]+1;
    m[r]=key[r];
    if(ch[r][0])m[r]=min(m[r],m[ch[r][0]]);
    if(ch[r][1])m[r]=min(m[r],m[ch[r][1]]);
}
void Push_Down(int r)
{
    if(rev[r])
    {
        Update_Rev(ch[r][0]);
        Update_Rev(ch[r][1]);
        rev[r]=0;
    }
    if(add[r])
    {
        Update_Add(ch[r][0],add[r]);
        Update_Add(ch[r][1],add[r]);
        add[r]=0;
    }
}
void Build(int &x,int l,int r,int father)
{
    if(l>r)return;
    int mid=(l+r)/2;
    NewNode(x,father,a[mid]);
    Build(ch[x][0],l,mid-1,x);
```

```
    Build(ch[x][1],mid+1,r,x);
    Push_Up(x);
}
void Init(){
    root=tot1=tot2=0;
    ch[root][0]=ch[root][1]=size[root]=
    add[root]=rev[root]=pre[root]=0;
    m[root]=INF;//这个不用也可以，如果在 push_up 那判断了的话，否则需
要
    NewNode(root,0,INF);
    NewNode(ch[root][1],root,INF);
    Build(Key_value,1,n,ch[root][1]);
    Push_Up(ch[root][1]);
    Push_Up(root);
}
//旋转
void Rotate(int x,int kind)
{
    int y=pre[x];
    Push_Down(y);
    Push_Down(x);
    ch[y][!kind]=ch[x][kind];
    pre[ch[x][kind]]=y;
    if(pre[y])
        ch[pre[y]][ch[pre[y]][1]==y]=x;
    pre[x]=pre[y];
    ch[x][kind]=y;
    pre[y]=x;
    Push_Up(y);
}
//Splay 调整
void Splay(int r,int goal)
{
    Push_Down(r);
    while(pre[r]!=goal)
    {
        if(pre[pre[r]]==goal)
        {
            //这题有反转操作，需要先 push_down,在判断左右孩子
            Push_Down(pre[r]);
            Push_Down(r);
            Rotate(r,ch[pre[r]][0]==r);
        }
```

```
        else
        {
            //这题有反转操作，需要先 push_down,在判断左右孩子
            Push_Down(pre[pre[r]]);
            Push_Down(pre[r]);
            Push_Down(r);
            int y=pre[r];
            int kind=(ch[pre[y]][0]==y);
            //两个方向不同，则先左旋再右旋
            if(ch[y][kind]==r)
            {
                Rotate(r,!kind);
                Rotate(r,kind);
            }
            //两个方向相同，相同方向连续两次
            else
            {
                Rotate(y,kind);
                Rotate(r,kind);
            }
        }
    }
    Push_Up(r);
    if(goal==0)root=r;
}
int Get_Kth(int r,int k)
{
    Push_Down(r);
    int t=size[ch[r][0]]+1;
    if(t==k)return r;
    if(t>k)return Get_Kth(ch[r][0],k);
    else return Get_Kth(ch[r][1],k-t);
}
int Get_Min(int r)
{
    Push_Down(r);
    while(ch[r][0])
    {
        r=ch[r][0];
        Push_Down(r);
    }
    return r;
}
int Get_Max(int r)
```

```
{
    Push_Down(r);
    while(ch[r][1])
    {
        r=ch[r][1];
        Push_Down(r);
    }
    return r;
}
//下面是操作了

void CUT(int l,int r,int c)
{
    Splay(Get_Kth(root,l),0);
    Splay(Get_Kth(root,r+2),root);
    int tmp=Key_value;
    Key_value=0;
    Push_Up(ch[root][1]);
    Push_Up(root);
    Splay(Get_Kth(root,c+1),0);
    Splay(Get_Kth(root,c+2),root);
    Key_value=tmp;
    pre[Key_value]=ch[root][1];
    Push_Up(ch[root][1]);
    Push_Up(root);
}

void Remove()//移除根结点
{
    if(ch[root][0]==0)//没有左孩子
    {
        root=ch[root][1];
        pre[root]=0;
    }
    else
    {
        int m=Get_Max(ch[root][0]);
        Splay(m,root);
        ch[m][1]=ch[root][1];
        pre[ch[root][1]]=m;
        root=m;
        pre[root]=0;
        Push_Up(root);//要更新
    }
```

```
}

void ADD(int l,int r,int D)
{
    Splay(Get_Kth(root,l),0);
    Splay(Get_Kth(root,r+2),root);
    Update_Add(Key_value,D);
    Push_Up(ch[root][1]);
    Push_Up(root);
}
void Reverse(int l,int r)
{
    Splay(Get_Kth(root,l),0);
    Splay(Get_Kth(root,r+2),root);
    Update_Rev(Key_value);
    Push_Up(ch[root][1]);
    Push_Up(root);
}
void Revolve(int l,int r,int T)//循环右移
{
    int len=r-l+1;
    T=(T%len+len)%len;
    if(T==0)return;
    int c=r-T+1;//将区间[c,r]放在[l,c-1]前面
    Splay(Get_Kth(root,c),0);
    Splay(Get_Kth(root,r+2),root);
    int tmp=Key_value;
    Key_value=0;
    Push_Up(ch[root][1]);
    Push_Up(root);
    Splay(Get_Kth(root,l),0);
    Splay(Get_Kth(root,l+1),root);
    Key_value=tmp;
    pre[Key_value]=ch[root][1];//这个不用忘记
    Push_Up(ch[root][1]);
    Push_Up(root);
}
void Insert(int x,int P)//在第 x 个数后面插入 P
{
    Splay(Get_Kth(root,x+1),0);
    Splay(Get_Kth(root,x+2),root);
    NewNode(Key_value,ch[root][1],P);
    Push_Up(ch[root][1]);
    Push_Up(root);
```

```
}
void erase(int r)//回收内存
{
    if(r)
    {
        s[++tot2]=r;
        erase(ch[r][0]);
        erase(ch[r][1]);
    }
}
void Delete(int x)//删除第 x 个数
{
    Splay(Get_Kth(root,x),0);
    Splay(Get_Kth(root,x+2),root);
    erase(Key_value);
    pre[Key_value]=0;
    Key_value=0;
    Push_Up(ch[root][1]);
    Push_Up(root);
}
int Query_Min(int l,int r)
{
    Splay(Get_Kth(root,l),0);
    Splay(Get_Kth(root,r+2),root);
    return m[Key_value];
}

void Inorder(int r)
{
    if(!r)return;
    Push_Down(r);
    Inorder(ch[r][0]);
    if(cnt>=1&&cnt<=n)
    {
        printf("%d",key[r]);
        if(cnt<n)printf(" ");
        else printf("\n");
    }
    cnt++;
    Inorder(ch[r][1]);
}

int main()
{
```

```
    char op[20];
    int x,y,z;
    while(scanf("%d",&n)==1)
    {
        for(int i=1;i<=n;i++)scanf("%d",&a[i]);
        Init();
        scanf("%d",&q);
        while(q--)
        {
            scanf("%s",op);
            if(strcmp(op,"ADD")==0)
            {
                scanf("%d%d%d",&x,&y,&z);
                ADD(x,y,z);
            }
            else if(strcmp(op,"REVERSE")==0)
            {
                scanf("%d%d",&x,&y);
                Reverse(x,y);
            }
            else if(strcmp(op,"REVOLVE")==0)
            {
                scanf("%d%d%d",&x,&y,&z);
                Revolve(x,y,z);
            }
            else if(strcmp(op,"INSERT")==0)
            {
                scanf("%d%d",&x,&y);
                Insert(x,y);
            }
            else if(strcmp(op,"DELETE")==0)
            {
                scanf("%d",&x);
                Delete(x);
            }
            else
            {
                scanf("%d%d",&x,&y);
                printf("%d\n",Query_Min(x,y));
            }
        }
    }
    return 0;
}
```

```
/*
 * 维修数列
 * 经典的伸展树的题目。
 * 题目首先给出一个数列，然后进行下列 6 种操作
 * 1:INSERT post tot c1,c2,...ctot :在当前数列的第 pos 个数字后插入 tot 个数字
 * 2:DELETE pos tot : 从当前数列的第 pos 个数字开始连续 删除 tot 个数字
 * 3:MAKE-SAME pos tot c :将当前数列的第 pos 个数字开始连续的 tot 个数字统一修改为 c
 * 4:REVERSE pos tot : 翻转当前数列的第 pos 个数字来说的连续的 tot 个数字
 * 5:GET-SUM pos tot :计算当前数列的第 pos 个数字来说的连续的 tot 个数字的和并输出
 * 6：MAX-SUM : 求出当前数列中和最大的一段序列，输出最大和
 */
#include <iostream>
#include <string.h>
#include <stdio.h>
#include <algorithm>
using namespace std;
#define Key_value ch[ch[root][1]][0]
const int MAXN=500010;
const int INF=0x3f3f3f3f;
int pre[MAXN],ch[MAXN][2],key[MAXN],size[MAXN];
int sum[MAXN],rev[MAXN],same[MAXN];
int lx[MAXN],rx[MAXN],mx[MAXN];
int root,tot1;
int s[MAXN],tot2;
int a[MAXN];
int n,q;
//debug 部分
void Treavel(int x)
{
    if(x)
    {
        Treavel(ch[x][0]);
        printf("结点%2d:左儿子 %2d 右儿子 %2d 父结点 %2d key=%2d, size=  %2d,  sum=%2d,rev=%2d  same=%2d  lx=%2d  rx=%2d mx=%2d\n",x,ch[x][0],ch[x][1],pre[x],key[x],size[x],sum[x],rev[x],same[x],lx[x],rx[x],mx[x]);
        Treavel(ch[x][1]);
```

```
    }
}
void debug()
{
    printf("root%d\n",root);
    Treavel(root);
}
void NewNode(int &r,int father,int k)
{
    if(tot2)r=s[tot2--];
    else r=++tot1;
    pre[r]=father;
    ch[r][0]=ch[r][1]=0;
    key[r]=k;
    sum[r]=k;
    rev[r]=same[r]=0;
    lx[r]=rx[r]=mx[r]=k;
    size[r]=1;
}
void Update_Same(int r,int v)
{
    if(!r)return;
    key[r]=v;
    sum[r]=v*size[r];
    lx[r]=rx[r]=mx[r]=max(v,v*size[r]);
    same[r]=1;
}
void Update_Rev(int r)
{
    if(!r)return;
    swap(ch[r][0],ch[r][1]);
    swap(lx[r],rx[r]);
    rev[r]^=1;//这里要注意，一定是异或 1
}
void Push_Up(int r)
{
    int lson=ch[r][0],rson=ch[r][1];
    size[r]=size[lson]+size[rson]+1;
    sum[r]=sum[lson]+sum[rson]+key[r];
    lx[r]=max(lx[lson],sum[lson]+key[r]+max(0,lx[rson]));
    rx[r]=max(rx[rson],sum[rson]+key[r]+max(0,rx[lson]));
    mx[r]=max(0,rx[lson])+key[r]+max(0,lx[rson]);
    mx[r]=max(mx[r],max(mx[lson],mx[rson]));
}
```

```
void Push_Down(int r)
{
    if(same[r])
    {
        Update_Same(ch[r][0],key[r]);
        Update_Same(ch[r][1],key[r]);
        same[r]=0;
    }
    if(rev[r])
    {
        Update_Rev(ch[r][0]);
        Update_Rev(ch[r][1]);
        rev[r]=0;
    }
}
void Build(int &x,int l,int r,int father)
{
    if(l>r)return;
    int mid=(l+r)/2;
    NewNode(x,father,a[mid]);
    Build(ch[x][0],l,mid-1,x);
    Build(ch[x][1],mid+1,r,x);
    Push_Up(x);
}
void Init()
{
    root=tot1=tot2=0;

ch[root][0]=ch[root][1]=pre[root]=size[root]=same[root]=rev[ro
ot]=sum[root]=key[root]=0;
    lx[root]=rx[root]=mx[root]=-INF;
    NewNode(root,0,-1);
    NewNode(ch[root][1],root,-1);
    for(int i=0;i<n;i++)scanf("%d",&a[i]);
    Build(Key_value,0,n-1,ch[root][1]);
    Push_Up(ch[root][1]);
    Push_Up(root);
}
void Rotate(int x,int kind)
{
    int y=pre[x];
    Push_Down(y);
    Push_Down(x);
    ch[y][!kind]=ch[x][kind];
```

```
    pre[ch[x][kind]]=y;
    if(pre[y])
        ch[pre[y]][ch[pre[y]][1]==y]=x;
    pre[x]=pre[y];
    ch[x][kind]=y;
    pre[y]=x;
    Push_Up(y);
}
void Splay(int r,int goal)
{
    Push_Down(r);
    while(pre[r]!=goal)
    {
        if(pre[pre[r]]==goal)
        {
            Push_Down(pre[r]);
            Push_Down(r);
            Rotate(r,ch[pre[r]][0]==r);
        }
        else
        {
            Push_Down(pre[pre[r]]);
            Push_Down(pre[r]);
            Push_Down(r);
            int y=pre[r];
            int kind=ch[pre[y]][0]==y;
            if(ch[y][kind]==r)
            {
                Rotate(r,!kind);
                Rotate(r,kind);
            }
            else
            {
                Rotate(y,kind);
                Rotate(r,kind);
            }
        }
    }
    Push_Up(r);
    if(goal==0)root=r;
}
int Get_Kth(int r,int k)
{
    Push_Down(r);
```

```
    int t=size[ch[r][0]]+1;
    if(t==k)return r;
    if(t>k)return Get_Kth(ch[r][0],k);
    else return Get_Kth(ch[r][1],k-t);
}

//在第 pos 个数后插入 tot 个数
void Insert(int pos,int tot)
{
    for(int i=0;i<tot;i++)scanf("%d",&a[i]);
    Splay(Get_Kth(root,pos+1),0);
    Splay(Get_Kth(root,pos+2),root);
    Build(Key_value,0,tot-1,ch[root][1]);
    Push_Up(ch[root][1]);
    Push_Up(root);
}
void erase(int r)
{
    if(!r)return;
    s[++tot2]=r;
    erase(ch[r][0]);
    erase(ch[r][1]);
}
//从第 pos 个数开始连续删除 tot 个数
void Delete(int pos,int tot)
{
    Splay(Get_Kth(root,pos),0);
    Splay(Get_Kth(root,pos+tot+1),root);
    erase(Key_value);
    pre[Key_value]=0;
    Key_value=0;
    Push_Up(ch[root][1]);
    Push_Up(root);
}
//从第 pos 个数连续开始的 tot 个数修改为 c
void Make_Same(int pos,int tot,int c)
{
    Splay(Get_Kth(root,pos),0);
    Splay(Get_Kth(root,pos+tot+1),root);
    Update_Same(Key_value,c);
    Push_Up(ch[root][1]);
    Push_Up(root);
}
//反转
```

```
void Reverse(int pos,int tot)
{
    Splay(Get_Kth(root,pos),0);
    Splay(Get_Kth(root,pos+tot+1),root);
    Update_Rev(Key_value);
    Push_Up(ch[root][1]);
    Push_Up(root);
}
//求和
int Get_Sum(int pos,int tot)
{
    Splay(Get_Kth(root,pos),0);
    Splay(Get_Kth(root,pos+tot+1),root);
    return sum[Key_value];
}
//得到最大和
int Get_MaxSum(int pos,int tot)
{
    Splay(Get_Kth(root,pos),0);
    Splay(Get_Kth(root,pos+tot+1),root);
    return mx[Key_value];
}
void Inorder(int r)
{
    if(!r)return;
    Push_Down(r);
    Inorder(ch[r][0]);
    printf("%d ",key[r]);
    Inorder(ch[r][1]);
}
int main()
{
    while(scanf("%d%d",&n,&q)==2)
    {
        Init();
        char op[20];
        int x,y,z;
        while(q--)
        {
            scanf("%s",op);
            if(op[0]=='I')
            {
                scanf("%d%d",&x,&y);
                Insert(x,y);
```

```
            }
            else if(op[0]=='D')
            {
                scanf("%d%d",&x,&y);
                Delete(x,y);
            }
            else if(op[0]=='M'&&op[2]=='K')
            {
                scanf("%d%d%d",&x,&y,&z);
                Make_Same(x,y,z);
            }
            else if(op[0]=='R')
            {
                scanf("%d%d",&x,&y);
                Reverse(x,y);
            }
            else if(op[0]=='G')
            {
                scanf("%d%d",&x,&y);
                printf("%d\n",Get_Sum(x,y));
            }
            else
            {
                printf("%d\n",Get_MaxSum(1,size[root]-2));
            }
        }
    }
    return 0;
}
```

19. 二维树状数组

```
int c[maxn][maxn];

int sum(int x,int y)
{
    int s=0;
    for(int i=x;i;i-=lowbit(i))
    {
        for(int j=y;j;j-=lowbit(j))
        {
            s+=c[i][j];
        }
    }
    return s;
}

void add(int x,int y,int v)
{
    for(int i=x;i<=n;i+=lowbit(i))
    {
        for(int j=y;j<=n;j+=lowbit(j))
        {
            c[i][j]+=v;
        }
    }
}
```

20.权值线段树

```
/******权值线段树模板******/
/**
权值线段树维护全局值域信息，每个结点记录值域的值出现次数
如果值域太大，需要离线操作
**/
typedef long long ll;
#define maxn 3e5+5
#define lson l,mid,rt<<1
#define rson mid+1,r,rt<<1
#define intmid int mid = (l+r)>>1

int seg[maxn<<2];
void update(int p,int v,int l,int r,int rt){
    //单点修改 所到的值域都需要修改 复杂度 logn
    t[rt]+=v;
    if(l==r)return ;
    intmid;
    if(p<=mid)update(p,v,lson);
    else update(p,v,rson);
}
int kth(int k,int l,int r,int rt){
    //如果左边多于 k 个，那么去左边找，否则找到右边
    if(l==r)return l;
    intmid;
    if(t[rt<<1]>=k)return kth(k,lson);
    return kth(k-t[rt<<1],rson);
}
int Rank(int p,int l,int r,int rt){
    //查询小于 p 的数出现的总次数，即区间[1,p-1]的数字个数，即 rank
    if(r<p)return t[rt];
    intmid;
    //[l,mid] + [mid+1,r]
    int ret=Rank(p,lson);
    if(mid+1<=p-1)ret+=Rank(p,rson);
    return ret;
}
//查询小于 p 的最大的数，即前驱
int findpre(int l,int r,int rt){
    if(l==r)return l;
    intmid;
    if(t[rt<<1|1])return findpre(rson);
```

```
    return findpre(lson);
}
int pre(int p,int l,int r,int rt){
    if(r<p){
        if(t[rt])return findpre(l,r,rt);
        return 0;
    }
    intmid;
    int re;
    if(mid+1<=p-1 && t[rt<<1|1] && (re=pre(p,rson))){
        return re;
    }
    return pre(p,lson);
}
//查询大于 p 的最小的数，即后继
int findnxt(int l,int r,int rt){
    if(l==r)return l;
    intmid;
    if(t[rt<<1])return findnxt(lson);
    return findnxt(rson);
}
int nxt(int p,int l,int r,int rt){
    if(p<l){
        if(t[rt])return findnxt(l,r,rt);
        return 0;
    }
    intmid;
    int re;
    if(p<mid && t[rt<<1] && (re=nxt(p,lson))){
        return re;
    }
    return nxt(p,rson);
}
```

21.矩形面积交

```
#include <bits/stdc++.h>
using namespace std;
#define maxn 100005
#define lson l,mid,rt<<1
#define rson mid+1,r,rt<<1|1
#define intmid int mid=(l+r)>>1
typedef long long ll;
struct seg
{
    int v;
    double l,r,h;
    seg(){}
    seg(int v,double l,double r,double h):v(v),l(l),r(r),h(h){}
    friend bool operator < (seg a,seg b){
        return a.h<b.h;
    }
}s[maxn];
int n;
int cnt[maxn<<2];
double x[maxn];
double one[maxn<<2],two[maxn<<2];
void push_up(int l,int r,int rt)
{
    if(cnt[rt]>=2)
        two[rt]=one[rt]=x[r+1]-x[l];//覆盖 2 次以上，直接算长度
    else if(cnt[rt])
    {
        one[rt]=x[r+1]-x[l];//覆盖 1 次以上，直接算长度
        if(l==r)two[rt]=0;
        else
            two[rt]=one[rt<<1]+one[rt<<1|1];//计算子区间的 1 次，加
上这一次
    }
    else
    {
        one[rt]=one[rt<<1]+one[rt<<1|1];
        two[rt]=two[rt<<1]+two[rt<<1|1];
    }
}
void update(int L,int R,int v,int l,int r,int rt)
{
    if(L<=l&&r<=R)
```

```
    {
        cnt[rt]+=v;
        push_up(l,r,rt);
        return;
    }
    intmid;
    if(L<=mid)update(L,R,v,lson);
    if(R>mid)update(L,R,v,rson);
    push_up(l,r,rt);
}
int main()
{
    int t;scanf("%d",&t);
    while(t--)
    {
        memset(cnt,0,sizeof(cnt));
        memset(one,0,sizeof(one));
        memset(two,0,sizeof(two));
        double x1,y1,x2,y2;
        scanf("%d",&n);
        for(int i=1;i<=n;i++)
        {
            scanf("%lf%lf%lf%lf",&x1,&y1,&x2,&y2);
            x[i]=x1;x[i+n]=x2;
            s[i]=seg(1,x1,x2,y1);
            s[i+n]=seg(-1,x1,x2,y2);
        }
        n<<=1;
        sort(x+1,x+1+n);
        sort(s+1,s+1+n);
        int len=unique(x+1,x+1+n)-(x+1);
        double ans=0;
        for(int i=1;i<n;i++)//数字代表点改为数字代表边，那么 r 值-1，
即左闭右开
        {
            int l=lower_bound(x+1,x+1+len,s[i].l)-x;
            int r=lower_bound(x+1,x+1+len,s[i].r)-x;
            update(l,r-1,s[i].v,1,len,1);
            ans+=two[1]*(s[i+1].h-s[i].h);
        }
        printf("%.2f\n",ans);
    }
    return 0;
}
```

22.矩形周长并

```
const int N = 500010;
const int INF = 1e8;
struct line//线段树节点{
    int l, r;//左右端点
    int lp, rp;//判断左右端点是否存在，存在为1，不存在未0
    int cnt, len;//cnt代表是否被覆盖，0代表未被完全覆盖，1代表被完全覆盖
    int num;//记录区间内的线段数目
}tree[4*N];
struct node//保存线段{
    int l, r, h;
    int f;
    bool operator < (const struct node & tmp) const{
        return h < tmp.h;
    }
}seg[4*N];
int x[N];
void build(int i, int l, int r){
    tree[i].l = l;
    tree[i].r = r;
    tree[i].len = tree[i].cnt = tree[i].num = 0;
    tree[i].lp = tree[i].rp = 0;
    if(l == r){
        return;
    }
    int mid = (l+r) >> 1;
    build(i*2, l, mid);
    build(i*2+1, mid+1, r);
}
int binsearch(int key, int k){
    int high = k;
    int low = 1;
    while(high >= low){
        int mid = (high+low) >> 1;
        if(x[mid] == key){
            return mid;
        }
        else if(x[mid] < key){
            low = mid+1;
        }
        else high = mid-1;
    }
```

```
    return -1;
}
void maintain(int i){
    if(tree[i].cnt)
    {
        tree[i].len = x[tree[i].r+1]-x[tree[i].l];
        tree[i].lp = tree[i].rp = tree[i].num = 1;
        return;
    }
    if(tree[i].l == tree[i].r)
    {
        tree[i].len = tree[i].lp = tree[i].rp = tree[i].num = 0;
        return;
    }
    tree[i].len = tree[i*2].len+tree[i*2+1].len;
    tree[i].lp = tree[i*2].lp;
    tree[i].rp = tree[i*2+1].rp;
    tree[i].num= (tree[i*2].num+tree[i*2+1].num-
    (tree[i*2].rp&&tree[i*2+1].lp));
}
void update(int i, int l, int r, int f)
{
    if(tree[i].l == l && tree[i].r == r)
    {
        tree[i].cnt += f;
        maintain(i);
        return;
    }
    int mid = (tree[i].l+tree[i].r) >> 1;
    if(mid >= r)
        update(i*2, l, r, f);
    else if(mid < l)
        update(i*2+1, l, r, f);
    else
    {
        update(i*2, l, mid, f);
        update(i*2+1, mid+1, r, f);
    }
    maintain(i);
}

int main()
{
     int n, x1, y1, x2, y2;
```

```
    while(~scanf("%d", &n))
    {
        int num = 1;
        for(int i = 1; i <= n; i++)
        {
            scanf("%d%d%d%d", &x1, &y1, &x2, &y2);
            seg[num] = (struct node){x1, x2, y1, 1};
            x[num++] = x1;

            seg[num] = (struct node){x1, x2, y2, -1};
            x[num++] = x2;
        }
        sort(seg+1, seg+num);
        sort(x+1, x+num);
        int k = 1;
        for(int i = 2; i < num; i++)
        {
            if(x[i-1] != x[i])
            {
                x[++k] = x[i];
            }
        }
        build(1, 1, k);
        int ans = 0;
        int pre = 0;
        for(int i = 1; i < num; i++)
        {
            int l = binsearch(seg[i].l, k);
            int r = binsearch(seg[i].r, k)-1;
            update(1, l, r, seg[i].f);
            int t = abs(tree[1].len-pre);
            pre = tree[1].len;
            ans += t;
            if(i<num-1)ans+=(seg[i+1].h-seg[i].h)*2*tree[1].num;
        }
        printf("%d\n", ans);
    }
    return 0;
}
```

23.ac 自动机

```
#define maxn 100005
#define maxm 100005

struct trie {
    int next[maxn][26], fail[maxn], end[maxn];
    int root, cnt;
    int new_node () {
        memset (next[cnt], -1, sizeof next[cnt]);
        end[cnt++] = 0;
        return cnt-1;
    }
    void init () {
        cnt = 0;
        root = new_node ();
    }
    void insert (char *buf) {//字典树插入一个单词
        int len = strlen (buf);
        int now = root;
        for (int i = 0; i < len; i++) {
            int id = buf[i]-'a';
            if (next[now][id] == -1) {
                next[now][id] = new_node ();
            }
            now = next[now][id];
        }
        end[now]++;
    }
    void build () {//构建 fail 指针
        queue <int> q;
        fail[root] = root;
        for (int i = 0; i < 26; i++) {
            if (next[root][i] == -1) {
                next[root][i] = root;
            }
            else {
                fail[next[root][i]] = root;
                q.push (next[root][i]);
            }
        }
        while (!q.empty ()) {
```

```
            int now = q.front (); q.pop ();
            for (int i = 0; i < 26; i++) {
                if (next[now][i] == -1) {
                    next[now][i] = next[fail[now]][i];
                }
                else {
                    fail[next[now][i]] = next[fail[now]][i];
                    q.push (next[now][i]);
                }
            }
        }
    }
    int query (char *buf) {
        int len = strlen (buf);
        int now = root;
        int res = 0;
        for (int i = 0; i < len; i++) {
            int id = buf[i]-'a';
            now = next[now][id];
            int tmp = now;
            while (tmp != root) {
                if (end[tmp]) {
                    res += end[tmp];
                    end[tmp] = 0;
                }
                tmp = fail[tmp];//沿着失配边走
            }
        }
        return res;
    }
}ac;
```

24. hash 字符串

```
typedef unsigned long long ull;
int n,m;
char s[100005],t[5005];
int lens,lent;ull base = 131;
ull po[100105],hs[100005];
ull geth(int l,int r){
    return (ull)hs[r]-po[r-l+1]*hs[l-1];
}
po[0]=1;
for(int i=1;i<=100000;i++){
po[i]=po[i-1]*base;
}
for(int i=1;i<=lens;i++){
    hs[i]=hs[i-1]*base+s[i];
}
```

25. 失配优化的 kmp

```
void getfail(char *p,int len){
    f[0]=f[1]=f2[0]=f2[1]=0;
    for(int i=1;i<len;i++){
        int j=f2[i];
        while(j&&p[i]!=p[j])j=f2[j];
        f2[i+1] = f[i+1] = (p[i]==p[j])?j+1:0;
        if(f[i+1]==j+1 && p[i+1]==p[j+1])f[i+1]=f[j+1];
    }
    /**
    带退格的优化 kmp：
    当在 p[i+1]的位置失配的时候，如果普通 kmp 的 p[i+1]指向 p[j+1]
    并且他们相同，那么下次匹配依旧还是失配的，就会继续跳到 p[f[j+1]]
    如此下来失配的时候刚好会跳到本该在的位置
    **/
}
int find(string a,string s){
    getfill(s);int j=0;
    for(int i=0;i<a.size();i++){
        while(j && a[i]!=s[j])j=f[j];
        if(a[i]==s[j])j++;
        if(j==s.size()){
            return i-s.size()+1;
        }
    }
}
```

26. SA

```
/**
sa[i]表示排名为 i 的后缀的开头位置；排名第 i 的是谁？
rank[i]表示后缀 i 在所有后缀中的名次；i 排第几？
height[i]表示排名 i-1 的后缀和排名 i 的后缀的最长公共前缀
**/
#include <bits/stdc++.h>
using namespace std;
const int maxn = 2e6+10;

char s[maxn],s2[maxn];
int len;
int sa[maxn],t1[maxn],t2[maxn],c[maxn],rk[maxn],height[maxn],n;

void build_sa(int m){
    int *x=t1,*y=t2;
    for(int i=0;i<m;i++)c[i]=0;
    for(int i=0;i<n;i++)c[x[i]=s[i]]++;
    for(int i=1;i<m;i++)c[i]+=c[i-1];
    for(int i=n-1;i>=0;i--)sa[--c[x[i]]]=i;

    for(int k=1;k<n;k<<=1){
        int p=0;
        for(int i=n-k;i<n;i++)y[p++]=i;
        for(int i=0;i<n;i++)if(sa[i]>=k)y[p++]=sa[i]-k;
        for(int i=0;i<m;i++)c[i]=0;
        for(int i=0;i<n;i++)c[x[i]]++;
        for(int i=1;i<m;i++)c[i]+=c[i-1];
        for(int i=n-1;i>=0;i--)sa[--c[x[y[i]]]]=y[i];
        swap(x,y);
        p=1;
        x[sa[0]]=0;
        for(int i=1;i<n;i++)
            x[sa[i]]=y[sa[i-1]]==y[sa[i]]&&
            y[sa[i-1]+k]==y[sa[i]+k] ? p-1:p++;
        if(p>=n)break;
        m=p;
    }
}

void get_height(){
```

```
    for(int i=0;i<n;i++)rk[sa[i]]=i;
    height[0]=0;
    int k=0;
    for(int i=0;i<n;i++){
        if(!rk[i])continue;
        if(k)k--;
        int j=sa[rk[i]-1];
        while(s[i+k]==s[j+k])k++;
        height[rk[i]]=k;
    }
}

int RMQ[maxn];
int mm[maxn];
int best[20][maxn];
void initRMQ(int n)
{
    mm[0]=-1;
    for(int i=1; i<=n; i++)
        mm[i]=((i&(i-1))==0)?mm[i-1]+1:mm[i-1];
    for(int i=1; i<=n; i++)best[0][i]=i;
    for(int i=1; i<=mm[n]; i++)
        for(int j=1; j+(1<<i)-1<=n; j++)
        {
            int a=best[i-1][j];
            int b=best[i-1][j+(1<<(i-1))];
            if(RMQ[a]<RMQ[b])best[i][j]=a;
            else best[i][j]=b;
        }
}
int askRMQ(int a,int b)
{
    int t;
    t=mm[b-a+1];
    b-=(1<<t)-1;
    a=best[t][a];
    b=best[t][b];
    return RMQ[a]<RMQ[b]?a:b;
}
int lcp(int a,int b)
{
    a=rk[a];
    b=rk[b];
    if(a>b)swap(a,b);
```

```
    return height[askRMQ(a+1,b)];
}



void solve(){
    int ans=-1;
    for(int i=1;i<n;i++){
        if(1ll*(sa[i]-len)*(sa[i-1]-len)<0)
            ans=max(ans,height[i]);
    }
    //printf("%d\n",ans);
}

int main(){
    while(scanf("%s%s",s,s2)!=EOF){
        len = strlen(s);
        s[len] = '$';
        s[len+1] = '\0';
        strcat(s,s2);
        n = strlen(s);n++;
        build_sa(128);get_height();n--;
        for(int i=1; i<=n; i++)RMQ[i]=height[i];
        initRMQ(n);

        while(1){
            int a,b;cin>>a>>b;
            cout<<lcp(sa[a],sa[b])<<endl;
        }

        solve();
    }
    return 0;
}
```

27.SAM 区间不同子串个数 DP

```
/**SAM 最长公共子串问题**/
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

const int maxn = 4010;
char s[maxn];
int pa[maxn],son[maxn][27];
int deep[maxn],cnt,root,last;
int ans[maxn][maxn];
int tot;

inline int newnode(int _deep){
    deep[++cnt]=_deep;
    return cnt;
}

inline void sam(int alp){
    int np=newnode(deep[last]+1);
    int u=last;
    memset(son[np],0,sizeof son[np]);
    while(u && !son[u][alp])son[u][alp]=np,u=pa[u];
    if(!u)pa[np]=root;
    else{
        int v=son[u][alp];
        if(deep[v]==deep[u]+1)pa[np]=v;
        else{
            int nv=newnode(deep[u]+1);
            memcpy(son[nv],son[v],sizeof son[v]);
            pa[nv]=pa[v],pa[v]=pa[np]=nv;
            while(u&&son[u][alp]==v)son[u][alp]=nv,u=pa[u];
        }
    }
    last=np;
    tot+=deep[np]-deep[pa[np]];
}

inline void pre(){
    cnt=0;
    memset(son[1],0,sizeof son[1]);
    root=last=newnode(0);
```

```
    tot=0;
}

int main(){
    int t;cin>>t;
    while(t--){
        memset(ans,0,sizeof ans);
        scanf("%s",s);
        int len=strlen(s);
        for(int i=0;i<len;i++){
            pre();
            for(int j=i;j<len;j++){
                sam(s[j]-'a');
                ans[i][j]=tot;
            }
        }
        int q;scanf("%d",&q);
        while(q--){
            int l,r;
            scanf("%d%d",&l,&r);
            printf("%d\n",ans[l-1][r-1]);
        }
    }
    return 0;
}
```

28.SAM 求 LCS1

```
/**SAM 最长公共子串问题**/
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 3e5+10;
char s[maxn],t[maxn];
int pa[maxn<<1],son[maxn<<1][27];
int deep[maxn<<1],cnt,root,last;
inline int newnode(int _deep){
    deep[++cnt]=_deep;
    return cnt;
}
inline void sam(int alp){
    int np=newnode(deep[last]+1);
    int u=last;
    memset(son[np],0,sizeof son[np]);
```

```
    while(u && !son[u][alp])son[u][alp]=np,u=pa[u];
    if(!u)pa[np]=root;
    else{
        int v=son[u][alp];
        if(deep[v]==deep[u]+1)pa[np]=v;
        else{
            int nv=newnode(deep[u]+1);
            memcpy(son[nv],son[v],sizeof son[v]);
            pa[nv]=pa[v],pa[v]=pa[np]=nv;
            while(u&&son[u][alp]==v)son[u][alp]=nv,u=pa[u];
        }
    }
    last=np;
}
inline void pre(){
    root=last=newnode(0);
}
int main(){
    while(scanf("%s%s",s,t+1)!=EOF){
        cnt=0;
        memset(son[1],0,sizeof son[1]);
        pre();
        int len=strlen(s);
        int m=strlen(t+1);
        int ans=0;
        for(int a=0;a<len;a++)sam(s[a]-'a');
        for(int a=1,l=0,p=1;a<=m;a++){
            int x=t[a]-'a';
            if(son[p][x])p=son[p][x],l++;
            else{
                for(;p&&!son[p][x];p=pa[p]);
                if(!p)l=0,p=1;
                else l=deep[p]+1,p=son[p][x];
            }
            ans=max(ans,l);
        }
        printf("%d\n",ans);
    }
    return 0;
}
//理解：为什么失配之后要转向 par 呢，因为在状态 p 失配
//说明该状态的[min,max]都不满足要求的子串，但是比 p 中短的后缀仍然可能是满足的
```

29.SAM 求 LCS2

SAM 的状态要多维护两个信息：
lcs，当多个串的最长公共子串的最后一个字符落在该状态上的长度；
nlcs，当前串的最长公共子串的最后一个字符落在该状态上的长度。
我们对每个串的匹配之后，要对每个状态的 lcs 进行维护，
显然 lcs=min(lcs, nlcs)，而我们最后所求的就是所有状态中 lcs 的最大值。
匹配的过程与上一题相同，但是在匹配过程中，
到达状态 p 时得到的 nlcs 未必就是该状态能表示的最长公共子串长，
因为如果一个子串出现了 n 次，那么子串的所有后缀也至少出现了 n 次。
因此在每个串匹配之后求要按照拓扑序的逆序维护每个状态的 nlcs，
使 p->par->nlcs=max(p->nlcs, p->par->nlcs)。

```
/**SAM 最长公共子串问题**/
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 1e5+10;
char s[maxn];
int pa[maxn<<1],son[maxn<<1][27];
int deep[maxn<<1],cnt,root,last;
int sum[maxn<<1],topo[maxn<<1];
int arr[maxn<<1],ned[maxn<<1];
inline int newnode(int _deep){
    deep[++cnt]=_deep;
    return cnt;
}
inline void sam(int alp){
    int np=newnode(deep[last]+1);
    int u=last;
    memset(son[np],0,sizeof son[np]);
    while(u && !son[u][alp])son[u][alp]=np,u=pa[u];
    if(!u)pa[np]=root;
    else{
        int v=son[u][alp];
        if(deep[v]==deep[u]+1)pa[np]=v;
        else{
            int nv=newnode(deep[u]+1);
            memcpy(son[nv],son[v],sizeof son[v]);
            pa[nv]=pa[v],pa[v]=pa[np]=nv;
            while(u&&son[u][alp]==v)son[u][alp]=nv,u=pa[u];
        }
```

```
    }
    last=np;
}
inline void toposort(){
    for(int a=1;a<=cnt;a++)sum[deep[a]]++;
    for(int a=1;a<=deep[last];a++)sum[a]+=sum[a-1];
    for(int a=1;a<=cnt;a++)topo[sum[deep[a]]--]=a;
}
inline void pre(){
    root=last=newnode(0);
}
int main(){
    scanf("%s",s);
    pre();
    memset(son[1],0,sizeof son[1]);
    int len=strlen(s);
    for(int a=0;a<len;a++)sam(s[a]-'a');
    toposort();
    for(int a=1;a<=cnt;a++)ned[a]=deep[a];
    while(scanf("%s",s+1)!=EOF){
        int m=strlen(s+1);
        for(int a=1,l=0,p=1;a<=m;a++){
            int x=s[a]-'a';
            if(son[p][x])p=son[p][x],l++;
            else{
                for(;p && !son[p][x];p=pa[p]);
                if(!p)l=0,p=1;
                else l=deep[p]+1,p=son[p][x];
            }
            arr[p]=max(arr[p],l);
        }
        for(int a=cnt;a>=1;a--){
            int p=topo[a];
            ned[p]=min(ned[p],arr[p]);
            if(arr[p]&&pa[p])arr[pa[p]]=deep[pa[p]];
            arr[p]=0;
        }
    }
    int ans=0;
    for(int a=1;a<=cnt;a++)ans=max(ans,ned[a]);
    printf("%d\n",ans);
    return 0;
}
```

30.SAM 求 right 集合大小

```
/**
统计长度为[1..l]的子串最多出现了多少次
关键问题在于统计某个串出现了多少次，
在 SAM 中，答案就是包含这个串的状态的 right 集合的大小
从 trans DAG 出发，right 集合的大小就是从该状态走到 end 的方案数
从 parent 树出发，每个结点的长度都是一个区间，只考虑最长串，
用最长串去更新短串。
**/
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 3e5+10;
char s[maxn];
int pa[maxn<<1],son[maxn<<1][27];
int deep[maxn<<1],cnt,root,last;
int sum[maxn<<1],topo[maxn<<1];
int arr[maxn<<1],ned[maxn<<1];
int r[maxn<<1],f[maxn<<1];
inline int newnode(int _deep){
    deep[++cnt]=_deep;
    return cnt;
}
inline void sam(int alp){
    int np=newnode(deep[last]+1);
    int u=last;
    memset(son[np],0,sizeof son[np]);
    while(u && !son[u][alp])son[u][alp]=np,u=pa[u];
    if(!u)pa[np]=root;
    else{
        int v=son[u][alp];
        if(deep[v]==deep[u]+1)pa[np]=v;
        else{
            int nv=newnode(deep[u]+1);
            memcpy(son[nv],son[v],sizeof son[v]);
            pa[nv]=pa[v],pa[v]=pa[np]=nv;
            while(u&&son[u][alp]==v)son[u][alp]=nv,u=pa[u];
        }
    }
    last=np;
}
```

```
inline void toposort(){
    for(int a=1;a<=cnt;a++)sum[deep[a]]++;
    for(int a=1;a<=deep[last];a++)sum[a]+=sum[a-1];
    for(int a=1;a<=cnt;a++)topo[sum[deep[a]]--]=a;
}
inline void pre(){
    root=last=newnode(0);
}
int main(){
    scanf("%s",s);
    pre();
    memset(son[1],0,sizeof son[1]);
    int len=strlen(s);
    for(int a=0;a<len;a++)sam(s[a]-'a');

    int tmp=root;
    for(int a=0;a<len;a++){
        tmp=son[tmp][s[a]-'a'];
        r[tmp]=1;
    }
    /***将原串前缀状态的 right 集合大小设为 1，
    相当于在后缀树中将后缀结点标记为 1**/
    toposort();
    /**deep[a] > deep[pa[a]]，所以排序，
    用 a 状态更新 pa[a]的状态，只考虑结点的 deep 便可
    **/
    for(int a=cnt;a>=1;a--)r[pa[topo[a]]] += r[topo[a]];/**right
表示状态在整个串出现次数**/
    for(int a=1;a<=cnt;a++)f[deep[a]] = max(f[deep[a]] , r[a]);/**
只更新最长串**/
    for(int a=len;a>=1;a--)f[a-1]=max(f[a],f[a-1]);/**长串更新短
串：短串包含在长串里面**/
    for(int i=1;i<=len;i++)printf("%d\n",f[i]);
    return 0;
}
```

31.trie

```
const int maxnode = 20000 * 1000 + 10;
const int sigma_size = 2;
int ans;
struct Trie
{
    int ch[maxnode][sigma_size];
    int val[maxnode];
    int sz; // 结点总数
    void clear()
    {
        sz = 1;    // 初始时只有一个根结点
        memset(ch[0], 0, sizeof(ch[0]));
    }
    int idx(char c)
    {
        return c - '0';    // 字符 c 的编号
    }
    // 插入字符串 s，附加信息为 v。注意 v 必须非 0，因为 0 代表“本结点不是单词结点”
    void insert(string s, int v)
    {
        int u = 0, n = s.size();
        for(int i = 0; i < n; i++)
        {
            int c = idx(s[i]);
            if(!ch[u][c])   // 结点不存在
            {
                memset(ch[sz], 0, sizeof(ch[sz]));
                val[sz] = 0;  // 中间结点的附加信息为 0
                ch[u][c] = sz++; // 新建结点
            }
            else ans=max(ans,i);
            u = ch[u][c]; // 往下走
        }
        val[u] = v; // 字符串的最后一个字符的附加信息为 v
    }
};
Trie trie;
```

32.最长回文子序列

```
int longestPalindromeSubSequence2(string str){
    int n=str.length();
    vector<vector<int> > dp(n,vector<int>(n));

    for(int i=n-1;i>=0;i--){
        dp[i][i]=1;
        for(int j=i+1;j<n;j++){
            if(str[i]==str[j])
                dp[i][j]=dp[i+1][j-1]+2;
            else
                dp[i][j]=max(dp[i+1][j],dp[i][j-1]);
        }
    }
    return dp[0][n-1];
}
```

回文序列个数

```
int NumOfPalindromeSubSequence(string str){
    int len=str.length();
    vector<vector<int> > dp(len,vector<int>(len));

    for(int j=0;j<len;j++){
        dp[j][j]=1;
        for(int i=j-1;i>=0;i--){
            dp[i][j]=dp[i+1][j]+dp[i][j-1]-dp[i+1][j-1];
            if(str[i]==str[j])
                dp[i][j]+=1+dp[i+1][j-1];
        }
    }
    return dp[0][len-1];
}
```

33.二维线段树

```
/*给定一个 n*n 的矩阵,每次给定一个子矩阵区域(x,y,l),
求出该区域内的最大值(A)和最小值(B),输出(A+B)/2,并用这个值更新矩阵
[x,y]的值*/
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
#define N 810
#define lx x<<1
#define rx x<<1|1
#define lsx lx,s,mid
#define rsx rx,mid+1,t
#define ly y<<1
#define ry y<<1|1
#define lsy ly,s,mid
#define rsy ry,mid+1,t
int maxv[N*4][N*4],minv[N*4][N*4],a[N][N];
int n,xL,xR,yL,yR,ans_max,ans_min,ans,idx,idy;
void pushup(int x,int y){
    maxv[x][y]=max(maxv[x][ly],maxv[x][ry]);
    minv[x][y]=min(minv[x][ly],minv[x][ry]);
}
void buildY(int x,int id,int y,int s,int t){
    if(s==t)
    {
        if(id!=-1)
        {
            maxv[x][y]=minv[x][y]=a[id][s];
        }
        else
        {
            maxv[x][y]=max(maxv[lx][y],maxv[rx][y]);
            minv[x][y]=min(minv[lx][y],minv[rx][y]);
        }
        return;
    }
    int mid=s+t>>1;
    buildY(x,id,lsy);buildY(x,id,rsy);
    pushup(x,y);
}
void buildX(int x,int s,int t)
{
```

```
    if(s==t)
    {
        buildY(x,s,1,1,n);
        return;
    }
    int mid=s+t>>1;
    buildX(lsx);buildX(rsx);
    buildY(x,-1,1,1,n);
}
void queryY(int x,int y,int s,int t)
{
    if(yL<=s&&t<=yR)
    {
        ans_min=min(ans_min,minv[x][y]);
        ans_max=max(ans_max,maxv[x][y]);
        return;
    }
    int mid=s+t>>1;
    if(yL<=mid) queryY(x,lsy);
    if(mid<yR) queryY(x,rsy);
}
void queryX(int x,int s,int t)
{
    if(xL<=s&&t<=xR)
    {
        queryY(x,1,1,n);
        return;
    }
    int mid=s+t>>1;
    if(xL<=mid) queryX(lsx);
    if(mid<xR) queryX(rsx);
}
void updataY(int x,int id,int y,int s,int t)
{
    if(s==t)
    {
        if(id!=-1) maxv[x][y]=minv[x][y]=ans;
        else
        {
            maxv[x][y]=max(maxv[lx][y],maxv[rx][y]);
            minv[x][y]=min(minv[lx][y],minv[rx][y]);
        }
        return;
    }
```

```
    int mid=s+t>>1;
    if(idy<=mid) updataY(x,id,lsy);
    else updataY(x,id,rsy);
    pushup(x,y);
}
void updataX(int x,int s,int t)
{
    if(s==t)
    {
        updataY(x,s,1,1,n);return;
    }
    int mid=s+t>>1;
    if(idx<=mid) updataX(lsx);
    else updataX(rsx);
    updataY(x,-1,1,1,n);
}
int main()
{
    int ca;
    scanf("%d",&ca);
    for(int cas=1;cas<=ca;++cas)
    {
        scanf("%d",&n);
        for(int i=1;i<=n;++i)
            for(int j=1;j<=n;++j)
                scanf("%d",&a[i][j]);
        buildX(1,1,n);
        int m,l,x1,y1,x2,y2;
        scanf("%d",&m);
        printf("Case #%d:\n",cas);
        for(int i=1;i<=m;++i)
        {
            scanf("%d%d%d",&idx,&idy,&l);
            l>>=1;
            xL=max(1,idx-l);xR=min(n,idx+l);
            yL=max(1,idy-l);yR=min(n,idy+l);
            ans_min=1e9;ans_max=0;
            queryX(1,1,n);
            ans=ans_min+ans_max>>1;
            updataX(1,1,n);
            printf("%d\n",ans);
        }
    }
}
```

34.CDQ 分治求三维 LIS

```
/*给定 n 个三维坐标，求其 LIS 及其数量，
当 x1<=x2,y1<=y2,z1<=z2 是，p1<=p2。
先排序掉一维，然后剩下两维分治，
用一个带长度和方案数的结构体树状数组维护。*/
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
#define N 100010
const int mod=(1<<30)-1;
struct Point{
    int x,y,z,id;
}q[N],tmp[N];
struct rec{
    int len,num;
    void init(){
        len=num=0;
    }
}dp[N],c[N];
int a[N];
int n,tot;
void updata(rec &a,rec b){
    if(a.len<b.len) a=b;
    else if(a.len==b.len) a.num+=b.num,a.num&=mod;
}
void add(int x,const rec &a){
    for(;x<=tot;x+=x&-x) updata(c[x],a);
}
rec getmax(int x){
    rec ans;ans.init();
    for(;x;x-=x&-x) updata(ans,c[x]);
    return ans;
}
void clr(int x){
    for(;x<=tot;x+=x&-x) c[x].init();
}
bool cmp1(Point &p1,Point &p2){
    return p1.x<p2.x||p1.x==p2.x&&
           p1.y<p2.y||p1.x==p2.x&
           &p1.y==p2.y&&
           p1.z<p2.z;
}
```

```
bool cmp2(Point &p1,Point &p2){
    return p1.y<p2.y||p1.y==p2.y&&p1.id<p2.id;
}
void solve(int s,int t){
    if(s==t) return;
    int mid=s+t>>1;
    solve(s,mid);
    for(int i=s;i<=t;++i) tmp[i]=q[i];
    sort(tmp+s,tmp+t+1,cmp2);
    rec ans;
    for(int i=s;i<=t;++i)
        if(tmp[i].id<=mid)
            add(tmp[i].z,dp[tmp[i].id]);
        else
        {
            ans=getmax(tmp[i].z);++ans.len;
            updata(dp[tmp[i].id],ans);
        }
    for(int i=s;i<=t;++i)
        if(tmp[i].id<=mid) clr(tmp[i].z);
    solve(mid+1,t);
}
int main(){
    int ca;
    scanf("%d",&ca);
    while(ca--){
        scanf("%d",&n);
        for(int i=1;i<=n;++i){
            scanf("%d%d%d",&q[i].x,&q[i].y,&q[i].z);
            a[i]=q[i].z;dp[i].len=dp[i].num=1;
        }
        sort(a+1,a+n+1);
        tot=unique(a+1,a+n+1)-a-1;
        for(int i=1;i<=n;++i)
            q[i].z=lower_bound(a+1,a+tot+1,q[i].z)-a;
        sort(q+1,q+n+1,cmp1);
        for(int i=1;i<=n;++i) q[i].id=i;
        solve(1,n);
        rec ans;ans.init();
        for(int i=1;i<=n;++i) updata(ans,dp[i]);
        printf("%d %d\n",ans.len,ans.num);
    }
    return 0;
}
```

35.CDQ 分治求矩阵内数字和

```
/*
你有一个 N*N 的棋盘，每个格子内有一个整数，初始时的时候全部为 0，现在需要维护两种操作：
1 x y A 1<=x,y<=N，A 是正整数，将格子 x,y 里的数字加上 A
2 x1 y1 x2 y2，1<=x1<= x2<=N，1<=y1<= y2<=N，输出 x1 y1 x2 y2 这个矩形内的数字和
3 无 终止程序
*/
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
#define N 500010
#define M 200010
struct rec{
    int x,y,t,k,w,be;
}a[M*4],tmp[M*4];
int c[N],ans[M];
int n,m;
void add(int x,int w)
{
    for(;x<=n;x+=x&-x) c[x]+=w;
}
int getsum(int x)
{
    int ans=0;
    for(;x;x-=x&-x) ans+=c[x];
    return ans;
}
bool cmp(const rec &a,const rec &b)
{
    return a.y<b.y||a.y==b.y&&a.k<b.k;
}
void solve(int l,int r)
{
    if(l==r) return;
    int mid=l+r>>1;
    for(int i=l;i<=r;++i)
        if(a[i].t<=mid&&a[i].k==1) add(a[i].x,a[i].w);
        else if(a[i].t>mid&&a[i].k==2)
            ans[a[i].be]+=getsum(a[i].x)*a[i].w;
    for(int i=l;i<=r;++i)
```

```
        if(a[i].t<=mid&&a[i].k==1) add(a[i].x,-a[i].w);
    int ll=l,rr=mid+1;
    for(int i=l;i<=r;++i)
        if(a[i].t<=mid) tmp[ll++]=a[i];
        else tmp[rr++]=a[i];
    for(int i=l;i<=r;++i) a[i]=tmp[i];
    solve(l,mid);solve(mid+1,r);
}
int main(){
    scanf("%d",&n);
    int tot=0,k,x1,y1,x2,y2,w;
    while(true){
        scanf("%d",&k);
        if(k==1)
        {
            scanf("%d%d%d",&x1,&y1,&w);

a[++tot].x=x1;a[tot].y=y1;a[tot].t=tot;a[tot].k=k;a[tot].w=w;
        }
        else if(k==2)
        {
            scanf("%d%d%d%d",&x1,&y1,&x2,&y2);

a[++tot].x=x2;a[tot].y=y2;a[tot].t=tot;a[tot].k=2;a[tot].w=1;a
[tot].be=++m;

a[++tot].x=x2;a[tot].y=y1-1;a[tot].t=tot;a[tot].k=2;a[tot].w=-
1;a[tot].be=m;

a[++tot].x=x1-1;a[tot].y=y2;a[tot].t=tot;a[tot].k=2;a[tot].w=-
1;a[tot].be=m;

a[++tot].x=x1-1;a[tot].y=y1-1;a[tot].t=tot;a[tot].k=2;a[tot].w
=1;a[tot].be=m;
        }
        else break;
    }
    sort(a+1,a+tot+1,cmp);
    solve(1,tot);
    for(int i=1;i<=m;++i) printf("%d\n",ans[i]);
    return 0;
}
```

36.多边形面积交与并

```
/*
 * 多边形的交，多边形的边一定是要按逆时针方向给出
 * 还要判断是凸包还是凹包，调用相应的函数
 * 面积并，只要和面积减去交即可
 */
#include <bits/stdc++.h>
using namespace std;
const int maxn = //300;
const double eps = 1e-8;
int dcmp(double x)  {
    if(x > eps) return 1;
    return x < -eps ? -1 : 0;
}
struct Point  {
    double x, y;
};
double cross(Point a,Point b,Point c) ///叉积  {
    return (a.x-c.x)*(b.y-c.y)-(b.x-c.x)*(a.y-c.y);
}
Point intersection(Point a,Point b,Point c,Point d)  {
    Point p = a;
    double t
    =((a.x-c.x)*(c.y-d.y)-(a.y-c.y)*(c.x-d.x))
    /((a.x-b.x)*(c.y-d.y)-(a.y-b.y)*(c.x-d.x));
    p.x +=(b.x-a.x)*t;
    p.y +=(b.y-a.y)*t;
    return p;
}
//计算多边形面积
double PolygonArea(Point p[], int n)  {
    if(n < 3) return 0.0;
    double s = p[0].y * (p[n - 1].x - p[1].x);
    p[n] = p[0];
    for(int i = 1; i < n; ++ i)
        s += p[i].y * (p[i - 1].x - p[i + 1].x);
    return fabs(s * 0.5);
}
double CPIA(Point a[], Point b[], int na, int nb){
    Point p[20], tmp[20];
    int tn, sflag, eflag;
    a[na] = a[0], b[nb] = b[0];
    memcpy(p,b,sizeof(Point)*(nb + 1));
```

```
    for(int i = 0; i < na && nb > 2; i++)
    {
        sflag = dcmp(cross(a[i + 1], p[0],a[i]));
        for(int j = tn = 0; j < nb; j++, sflag = eflag)
        {
            if(sflag>=0) tmp[tn++] = p[j];
            eflag = dcmp(cross(a[i + 1], p[j + 1],a[i]));
            if((sflag ^ eflag) == -2)
                tmp[tn++] = intersection(a[i], a[i + 1], p[j], p[j
+ 1]); ///求交点
        }
        memcpy(p, tmp, sizeof(Point) * tn);
        nb = tn, p[nb] = p[0];
    }
    if(nb < 3) return 0.0;
    return PolygonArea(p, nb);
}
double SPIA(Point a[], Point b[], int na, int nb){
    int i, j;
    Point t1[4], t2[4];
    double res = 0, num1, num2;
    a[na] = t1[0] = a[0], b[nb] = t2[0] = b[0];
    for(i = 2; i < na; i++)
    {
        t1[1] = a[i-1], t1[2] = a[i];
        num1 = dcmp(cross(t1[1], t1[2],t1[0]));
        if(num1 < 0) swap(t1[1], t1[2]);
        for(j = 2; j < nb; j++)
        {
            t2[1] = b[j - 1], t2[2] = b[j];
            num2 = dcmp(cross(t2[1], t2[2],t2[0]));
            if(num2 < 0) swap(t2[1], t2[2]);
            res += CPIA(t1, t2, 3, 3) * num1 * num2;
        }
    }
    return res;
}
Point p1[maxn], p2[maxn];
int n1, n2;
int main()
{
    while(cin>>n1>>n2)
    {
        for(int i = 0; i < n1; i++) scanf("%lf%lf", &p1[i].x,
```

```
&p1[i].y);
        for(int i = 0; i < n2; i++) scanf("%lf%lf", &p2[i].x,
&p2[i].y);
        double Area = SPIA(p1, p2, n1, n2);
    }
    return 0;
}
```

37.模拟退火求最小费马点

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;
#define N 110
#define inf 10010
struct Point{
    double x,y;
}a[N];
const int fx[4][2]={{1,0},{-1,0},{0,1},{0,-1}};
int n;
double dis(Point p1,Point p2)
{
    return
sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
}
double get_sum(Point p)
{
    double ans=0;
    for(int i=1;i<=n;++i)
        ans+=dis(a[i],p);
    return ans;
}
double solve()
{
    Point p,p1,p2;
    p.x=p.y=0;
    double ans=get_sum(p);
    int x1=inf,y1=inf,x2=-inf,y2=-inf;
    for(int i=1;i<=n;++i)
    {
        x1=min(x1,(int)a[i].x);
        y1=min(y1,(int)a[i].y);
        x2=max(x2,(int)a[i].x);
        y2=max(y2,(int)a[i].y);
    }
    double diff,minn,d;
    int i;
    srand(0);
    for(int z=1;z<=20;++z)
```

```
    {
        p1.x=x1+rand()%(x2-x1+1);
        p1.y=y1+rand()%(y2-y1+1);
        minn=get_sum(p1);
        diff=10000;
        while(diff>0.001)
        {
            for(i=0;i<4;++i)
            {
                p2.x=p1.x+fx[i][0]*diff;
                p2.y=p1.y+fx[i][1]*diff;
                d=get_sum(p2);
                if(d<minn)
                {
                    minn=d;p1=p2;break;
                }
            }
            if(i==4) diff*=0.9;
        }
        if(minn<ans) ans=minn;
    }
    return ans;
}
int main()
{
    while(scanf("%d",&n)!=EOF)
    {
        for(int i=1;i<=n;++i)
            scanf("%lf%lf",&a[i].x,&a[i].y);
        printf("%.0f\n",solve());
    }
    return 0;
}
```

38.最小圆覆盖

```
#include <cstdio>
#include <cmath>
#include <cstring>
#include <algorithm>
using namespace std;
#define N 510
const double eps=1e-8;
struct Point{
    double x,y;
}a[N];
int n;
double dis(Point p1,Point p2)
{
    return
sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
}
Point circumcenter(Point p1,Point p2,Point p3)//求三角形外心
{
    Point p;
    double a1=p2.x-p1.x,b1=p2.y-p1.y,c1=(a1*a1+b1*b1)/2;
    double a2=p3.x-p1.x,b2=p3.y-p1.y,c2=(a2*a2+b2*b2)/2;
    double d=a1*b2-a2*b1;
    p.x=p1.x+(c1*b2-c2*b1)/d;
    p.y=p1.y+(a1*c2-a2*c1)/d;
    return p;
}
void min_cover_circle(Point p[],int n,Point &c,double &r)
{
    random_shuffle(a+1,a+n+1);//随机化序列
    c=a[1];r=0;
    for(int i=2;i<=n;++i)
        if(dis(p[i],c)>r+eps)//第一个点
        {
            c=p[i];r=0;
            for(int j=1;j<i;++j)
                if(dis(p[j],c)>r+eps)//第二个点
                {
                    c.x=(p[i].x+p[j].x)/2;
                    c.y=(p[i].y+p[j].y)/2;
                    r=dis(p[i],c);
                    for(int k=1;k<j;++k)
```

```
                    if(dis(p[k],c)>r+eps)//第三个点
                    {
                        c=circumcenter(p[i],p[j],p[k]);
                        r=dis(p[i],c);
                    }
                }
            }
        }
        int main()
        {
            Point c;
            double r;
            while(scanf("%d",&n),n)
            {
                for(int i=1;i<=n;++i)
                    scanf("%lf%lf",&a[i].x,&a[i].y);
                min_cover_circle(a,n,c,r);
                printf("%.2f %.2f %.2f\n",c.x,c.y,r);
            }
        }
```

39.最小矩形覆盖

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;
#define N 50010
const double eps=1e-12;
struct Point{
    double x,y,angle;
    Point(){}
    Point(double _x,double _y):x(_x),y(_y){}
    void input()
    {
        scanf("%lf%lf",&x,&y);
    }
    void output()
    {
        printf("%.5f %.5f\n",x+eps,y+eps);
    }
    void f(double &a,double &b)
    {
        a=x;b=y;
    }
    bool operator<(const Point &a)const
    {
        return y<a.y||y==a.y&&x<a.x;
    }
    Point operator-(const Point &a)const
    {
        return Point(x-a.x,y-a.y);
    }
    double operator*(const Point &a)const
    {
        return x*a.x+y*a.y;
    }
    bool operator!=(const Point &a)const
    {
        return x!=a.x||y!=a.y;
    }
    Point operator+(const Point &a)const
    {
```

```
        return Point(x+a.x,y+a.y);
    }
}P[N],sta[N],Q[5];
//Q 数组记录最小矩形的四个点
int n;
const double inf=1e20;
double cross(Point p0,Point p1,Point p2)
{
    return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}
void Graham()
{
    sort(P,P+n);
    int cnt=0;
    for(int i=1;i<n;++i)
        if(P[i]!=P[i-1]) P[++cnt]=P[i];
    int top=1;
    sta[0]=P[0];sta[1]=P[1];
    for(int i=2;i<=cnt;++i)
    {
        while(top&&cross(sta[top-1],sta[top],P[i])>-eps) --top;
        sta[++top]=P[i];
    }
    int k=top;
    for(int i=cnt-1;i>=0;--i)
    {
        while(top>k&&cross(sta[top-1],sta[top],P[i])>-eps)
--top;
        sta[++top]=P[i];
    }
    if(top) --top;
    for(int i=0;i<=top;++i) P[i]=sta[i];
    n=top+1;
}
double dis(Point p1,Point p2)
{
    return
sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
}
double dis_lp(Point p1,Point p2,Point p3)
{
    return fabs(cross(p1,p2,p3))/dis(p1,p2);
}
Point intersect(Point p1,Point p2,Point p3)//过点 p3 与向量 p1p2
```

```
垂直的向量与直线 p1p2 的交点
{
    double x1,x2,x3,y1,y2,y3,d,dx,dy,a1,b1,c1,a2,b2,c2;
    p1.f(x1,y1);p2.f(x2,y2);p3.f(x3,y3);
    a1=x2-x1;b1=y2-y1;c1=x3*(x2-x1)+y3*(y2-y1);
    a2=y2-y1;b2=x1-x2;c2=x1*y2-x2*y1;
    d=a1*b2-a2*b1;dx=c1*b2-c2*b1;dy=a1*c2-a2*c1;
    return Point(dx/d,dy/d);
}
double solve()
{
    P[n]=P[0];
    int i1=0,i2=0;
    for(int i=1;i<n;++i)
    {
        if(P[i].y<P[i1].y) i1=i;
        if(P[i].y>P[i2].y) i2=i;
    }
    int i3=i1+1,i4=i2+1;
    double ans=inf,a,h;
    for(int i=0;i<n;++i)
    {

while(cross(P[i1],P[i2],P[i1+1])-cross(P[i1],P[i2+1],P[i1+1])<
-eps)
            i2=(i2+1)%n;
        h=dis_lp(P[i1],P[i1+1],P[i2]);
        while((P[i1+1]-P[i1])*(P[i3+1]-P[i3])>eps) i3=(i3+1)%n;
        while((P[i1+1]-P[i1])*(P[i4+1]-P[i4])<-eps) i4=(i4+1)%n;

a=fabs((P[i1+1]-P[i1])*(P[i3]-P[i4]))/dis(P[i1+1],P[i1]);
        if(ans>a*h)
        {
            ans=a*h;
            Q[0]=intersect(P[i1],P[i1+1],P[i3]);
            Q[1]=intersect(P[i1],P[i1+1],P[i4]);
            Point p=P[i2]+(P[i1]-P[i1+1]);
            Q[2]=intersect(P[i2],p,P[i3]);
            Q[3]=intersect(P[i2],p,P[i4]);
        }
        i1=(i1+1)%n;
    }
    return ans;
}
```

```
bool cmp(Point p1,Point p2)
{
    return p1.angle<p2.angle;
}
int main()
{
    scanf("%d",&n);
    for(int i=0;i<n;++i) P[i].input();
    Graham();
    if(n==1)
    {
        printf("%.5f\n",0);
        for(int i=1;i<=4;++i)
            P[0].output();
    }
    else if(n==2)
    {
        printf("%.5f\n",0);
        sort(P,P+1);
        P[0].output();P[0].output();
        P[1].output();P[1].output();
    }
    else
    {
        printf("%.5f\n",solve()+eps);
        sort(Q,Q+4);
        for(int i=1;i<4;++i)
            Q[i].angle=atan2(Q[i].y-Q[0].y,Q[i].x-Q[0].x);
        sort(Q+1,Q+4,cmp);
        for(int i=0;i<4;++i)
            Q[i].output();
    }
}
```

40. 凸包上最大三角形面积

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
#include <set>
#include <utility>
using namespace std;
#define N 50010
struct Point{
    double x,y;
    Point(){}
    Point(double _x,double _y):x(_x),y(_y){}
    bool operator<(const Point &a)const
    {
        return x<a.x||x==a.x&&y<a.y;
    }
    void input()
    {
        scanf("%lf%lf",&x,&y);
    }
}P[N],sta[N];
int n;
bool f[N];
const double eps=1e-8;
double cross(Point p0,Point p1,Point p2)
{
    return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}
double solve()
{
    if(n<3) return 0;
    double ans=0;
    int i=0,j,k;
    P[n]=P[0];
    for(int i=0;i<n;++i)
    {
        j=(i+1)%n;k=(j+1)%n;
        while(i!=k)
        {

while(j!=k&&cross(P[i],P[k],P[j])-cross(P[i],P[k],P[j+1])<-eps
```

```
)
                j=(j+1)%n;
            ans=max(ans,cross(P[i],P[k],P[j])/2);
            k=(k+1)%n;
        }
    }
    return ans;
}
void Graham()
{
    sort(P,P+n);
    int top=1;
    sta[0]=P[0];sta[1]=P[1];
    for(int i=2;i<n;++i)
    {
        while(top&&cross(sta[top-1],sta[top],P[i])>-eps) --top;
        sta[++top]=P[i];
    }
    int k=top;
    for(int i=n-2;i>=0;--i)
    {
        while(top>k&&cross(sta[top-1],sta[top],P[i])>-eps)
--top;
        sta[++top]=P[i];
    }
    if(top) --top;
    for(int i=0;i<=top;++i) P[i]=sta[i];
    n=top+1;
}
int main()
{
    while(scanf("%d",&n),n!=-1)
    {
        for(int i=0;i<n;++i) P[i].input();
        Graham();
        printf("%.2f\n",solve());
    }
    return 0;
}
```

41. 凸包直径

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;
#define N 50010
struct Point{
    double x,y;
    Point(){}
    Point(double _x,double _y):x(_x),y(_y){}
    bool operator<(const Point &a)const{
        return x<a.x||x==a.x&&y<a.y;
    }
    void input(){
        scanf("%lf%lf",&x,&y);
    }
}P[N],sta[N];
int n;
const double eps=1e-8;
double cross(Point p0,Point p1,Point p2){
    return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}
double area(Point P[],int n){
    double ans=0;
    P[n]=P[0];
    for(int i=0;i<n;++i)
        ans+=cross(P[0],P[i],P[i+1]);
    return -ans/2;
}
void Reverse(Point P[],int n){
    for(int i=0;i<(n+1)/2;++i)
        swap(P[i],P[n-i-1]);
}
double dis2(Point p1,Point p2){
    return (p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y);
}
double solve(Point P[],int n){
    int i1=0,i2=0;
    P[n]=P[0];
    for(int i=1;i<n;++i)
    {
```

```
        if(P[i].y>P[i1].y) i1=i;
        if(P[i].y<P[i2].y) i2=i;
    }
    double ans=0,tmp;
    for(int i=0;i<n;++i)
    {

while(tmp=cross(P[i1],P[i2],P[i1+1])-cross(P[i1],P[i2+1],P[i1+
1])<-eps)
           i2=(i2+1)%n;

ans=max(ans,max(dis2(P[i1],P[i2]),dis2(P[i1+1],P[i2+1])));
        i1=(i1+1)%n;
    }
    return ans;
}
void Graham(){
    sort(P,P+n);
    int top=1;
    sta[0]=P[0];sta[1]=P[1];
    for(int i=2;i<n;++i){
        while(top&&cross(sta[top-1],sta[top],P[i])>-eps) --top;
        sta[++top]=P[i];
    }
    int k=top;
    for(int i=n-2;i>=0;--i) {
        while(top>k&&cross(sta[top-1],sta[top],P[i])>-eps)
--top;
        sta[++top]=P[i];
    }
    if(top) top--;
    for(int i=0;i<=top;++i) P[i]=sta[i];
    n=top+1;
}
int main(){
    while(scanf("%d",&n)!=EOF){
        for(int i=0;i<n;++i) P[i].input();
        if(area(P,n)<0) Reverse(P,n);
        Graham();
        printf("%.0f\n",solve(P,n));
    }
    return 0;
}
```

42. pb_ds

```
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
typedef                                    tree<pt,null_type,less<
pt >,rb_tree_tag,tree_order_statistics_node_update> rbtree;

/*
定义一颗红黑树
int 关键字类型
null_type 无映射(低版本 g++为 null_mapped_type)
less<int>从小到大排序
rb_tree_tag 红黑树（splay_tree_tag）
tree_order_statistics_node_update 结点更新
插入 t.insert();
删除 t.erase();
Rank:t.order_of_key();
第 K 值:t.find_by_order();
前驱:t.lower_bound();
后继 t.upper_bound();
a.join(b)b 并入 a 前提是两棵树的 key 的取值范围不相交
a.split(v,b)key 小于等于 v 的元素属于 a，其余的属于 b
T.lower_bound(x)  >=x 的 min 的迭代器
T.upper_bound((x)  >x 的 min 的迭代器
T.find_by_order(k) 有 k 个数比它小的数
*/

struct pt{
    int first,second;
    pt(int x,int y) :first(x),second(y) {}
    bool operator<(const pt h)const{
        return                  first<h.first                   ||
(first==h.first&&second<h.second);
    }
    bool operator==(const pt h)const{
        return first==h.first&&second==h.second;
    }
};
```

```
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/hash_policy.hpp>
__gnu_pbds::gp_hash_table<int,bool> h;
```
支持 find 和[]

```
#include<ext/pb_ds/priority_queue.hpp>
typedef
__gnu_pbds::priority_queue<node,less<node>,pairing_heap_tag>
heap;
heap::point_iterator hit[M];
heap pq;
```
pb_ds 库的 push 操作是有返回值的（与 STL 不同），返回的类型就是迭代器，这样用一个迭代器数组保存所有 push 进优先队列的元素的迭代器，就可以随时修改优先队列内部元素了。
```
pq.modify(hit[e[i].to],node(e[i].to,d[e[i].to]));

a.join(b);
```
此时优先队列 b 内所有元素就被合并进优先队列 a 中，且优先队列 b 被清空。
```
hit[i]=pq.push(i);
if(hit[i]==0){
    pq.modify(hit[i],1);
}
pq.erase(hit[i]);

rope
```
二叉树实现，存储字符串，时间复杂度 O(logN)
```
#include<ext/rope>
using namespace __gnu_cxx;
rope<int> T;
for(int i=1;i<=n;i++) T.push_back(i);
T = T.substr(p,s) + T.substr(0,p)+T.substr(p+s,n-p-s);
for(int i=1;i<=n;i++) printf("%d ",T.at(i));
```
可持久化：
```
rope<char> *his[maxn];

his[0] = new rope<char>();
his[i] = new rope<char>(*his[i-1]);
```
O(1)copy 历史版本

insert(位置,值)
erase(位置，大小)
substr(位置，大小)