1 Computational Geometry

1.1 Geometry

```cpp
const double PI=atan2(0.0,-1.0);
template<typename T>
struct point{
  T x,y;
  point(){}
  point(const T&x,const T&y):x(x),y(y){}
  point operator+(const point &b)const{
    return point(x+b.x,y+b.y); }
  point operator-(const point &b)const{
    return point(x-b.x,y-b.y); }
  point operator*(const T &b)const{
    return point(x*b,y*b); }
  point operator/(const T &b)const{
    return point(x/b,y/b); }
  bool operator==(const point &b)const{
    return x==b.x&&y==b.y; }
  T dot(const point &b)const{
    return x*b.x+y*b.y; }
  T cross(const point &b)const{
    return x*b.y-y*b.x; }
  point normal()const{//求法向量
    return point(-y,x); }
  T abs2()const{//向量長度的平方
    return dot(*this); }
  T rad(const point &b)const{//兩向量的弧度
return fabs(atan2(fabs(cross(b)),dot(b))); }
  T getA()const{//對x軸的弧度
    T A=atan2(y,x);//超過180度會變負的
    if(A<=-PI/2)A+=PI*2;
    return A;
  }
};
template<typename T>
struct line{
  line(){}
  point<T> p1,p2;
  T a,b,c;//ax+by+c=0
  line(const point<T>&x,const point<T>&y):p1(x),p2(y){}
  void pton(){//轉成一般式
    a=p1.y-p2.y;
    b=p2.x-p1.x;
    c=-a*p1.x-b*p1.y;
  }
  T ori(const point<T> &p)const{//點和有向直線的關係，>0左邊、=0在線上<0右邊
    return (p2-p1).cross(p-p1);
  }
  T btw(const point<T> &p)const{//點投影落在線段上<=0
    return (p1-p).dot(p2-p);
  }
  bool point_on_segment(const point<T>&p)const{//點是否在線段上
    return ori(p)==0&&btw(p)<=0;
  }
  T dis2(const point<T> &p,bool is_segment=0)const{//點跟直線/線段的距離平方
    point<T> v=p2-p1,v1=p-p1;
    if(is_segment){
      point<T> v2=p-p2;
      if(v.dot(v1)<=0)return v1.abs2();
      if(v.dot(v2)>=0)return v2.abs2();
    }
    T tmp=v.cross(v1);
    return tmp*tmp/v.abs2();
  }
  T seg_dis2(const line<T> &l)const{//兩線段距離平方
    return min({dis2(l.p1,1),dis2(l.p2,1),l.dis2(p1,1),l.dis2(p2,1)});
  }
  point<T> projection(const point<T> &p)const{//點對直線的投影
    point<T> n=(p2-p1).normal();
    return p-n*(p-p1).dot(n)/n.abs2();
  }
  point<T> mirror(const point<T> &p)const{
    //點對直線的鏡射，要先呼叫pton轉成一般式
    point<T> R;
    T d=a*a+b*b;
    R.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/d;
    R.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/d;
    return R;
  }
  bool equal(const line &l)const{//直線相等
    return ori(l.p1)==0&&ori(l.p2)==0;
  }
  bool parallel(const line &l)const{
    return (p1-p2).cross(l.p1-l.p2)==0;
  }
  bool cross_seg(const line &l)const{
    return (p2-p1).cross(l.p1-p1)*(p2-p1).cross(l.p2-p1)<=0;//直線是否交線段
  }
  int line_intersect(const line &l)const{//直線相交情況，-1無限多點、1交於一點、0不相交
    return parallel(l)?(ori(l.p1)==0?-1:0):1;
  }
  int seg_intersect(const line &l)const{
    T c1=ori(l.p1), c2=ori(l.p2);
    T c3=l.ori(p1), c4=l.ori(p2);
    if(c1==0&&c2==0){//共線
      bool b1=btw(l.p1)>=0,b2=btw(l.p2)>=0;
      T a3=l.btw(p1),a4=l.btw(p2);
      if(b1&&b2&&a3==0&&a4>=0) return 2;
      if(b1&&b2&&a3>=0&&a4==0) return 3;
      if(b1&&b2&&a3>=0&&a4>=0) return 0;
      return -1;//無限交點
    }else if(c1*c2<=0&&c3*c4<=0)return 1;
    return 0;//不相交
  }
  point<T> line_intersection(const line &l)const{/*直線交點*/
    point<T> a=p2-p1,b=l.p2-l.p1,s=l.p1-p1;
    //if(a.cross(b)==0)return INF;
    return p1+a*(s.cross(b)/a.cross(b));
  }
  point<T> seg_intersection(const line &l)const{//線段交點
    int res=seg_intersect(l);
    if(res<=0) assert(0);
    if(res==2) return p1;
    if(res==3) return p2;
    return line_intersection(l);
  }
};
template<typename T>
struct polygon{
  polygon(){}
  vector<point<T> > p;//逆時針順序
  T area()const{//面積
    T ans=0;
    for(int i=p.size()-1,j=0;j<(int)p.size();i=j++)
      ans+=p[i].cross(p[j]);
    return ans/2;
  }
  point<T> center_of_mass()const{//重心
    T cx=0,cy=0,w=0;
    for(int i=p.size()-1,j=0;j<(int)p.size();i=j++){
      T a=p[i].cross(p[j]);
      cx+=(p[i].x+p[j].x)*a;
      cy+=(p[i].y+p[j].y)*a;
      w+=a;
    }
    return point<T>(cx/3/w,cy/3/w);
  }
  char ahas(const point<T>& t)const{//點是否在簡單多邊形內，是的話回傳1、在邊上回傳-1、否則回傳0
    bool c=0;
    for(int i=0,j=p.size()-1;i<p.size();j=i++)
      if(line<T>(p[i],p[j]).point_on_segment(t))return -1;
      else if((p[i].y>t.y)!=(p[j].y>t.y)&&
      t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j].y-p[i].y)+p[i].x)
        c=!c;
    return c;
  }
  char point_in_convex(const point<T>&x)const{
    int l=1,r=(int)p.size()-2;
    while(l<=r){//點是否在凸多邊形內，是的話回傳1、在邊上回傳-1、否則回傳0
      int mid=(l+r)/2;
      T a1=(p[mid]-p[0]).cross(x-p[0]);
      T a2=(p[mid+1]-p[0]).cross(x-p[0]);
      if(a1>=0&&a2<=0){
        T res=(p[mid+1]-p[mid]).cross(x-p[mid]);
        return res>0?1:(res>=0?-1:0);
      }else if(a1<0)r=mid-1;
      else l=mid+1;
    }
    return 0;
  }
  vector<T> getA()const{//凸包邊對x軸的夾角
    vector<T>res;//一定是遞增的
    for(size_t i=0;i<p.size();++i)
      res.push_back((p[(i+1)%p.size()]-p[i]).getA());
    return res;
  }
  bool line_intersect(const vector<T>&A,const line<T> &l)const{//O(logN)
    int f1=upper_bound(A.begin(),A.end(),(l.p1-l.p2).getA())-A.begin();
    int f2=upper_bound(A.begin(),A.end(),(l.p2-l.p1).getA())-A.begin();
    return l.cross_seg(line<T>(p[f1],p[f2]));
  }
  polygon cut(const line<T> &l)const{//凸包對直線切割，得到直線l左側的凸包
    polygon ans;
    for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
      if(l.ori(p[i])>=0){
        ans.p.push_back(p[i]);
        if(l.ori(p[j])<0)
          ans.p.push_back(l.line_intersection(line<T>(p[i],p[j])));
      }else if(l.ori(p[j])>0)
        ans.p.push_back(l.line_intersection(line<T>(p[i],p[j])));
    }
    return ans;
  }
  static bool monotone_chain_cmp(const point<T>& a,const point<T>& b){//凸包排序函數
    return (a.x<b.x)||(a.x==b.x&&a.y<b.y);
  }
  void monotone_chain(vector<point<T> > &s){//凸包
    sort(s.begin(),s.end(),monotone_chain_cmp);
    p.resize(s.size()+1);
    int m=0;
    for(size_t i=0;i<s.size();++i){
      while(m>=2&&(p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0)--m;
      p[m++]=s[i];
    }
    for(int i=s.size()-2,t=m+1;i>=0;--i){
      while(m>=t&&(p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0)--m;
      p[m++]=s[i];
    }
    if(s.size()>1)--m;
    p.resize(m);
  }
  T diam(){//直徑
    int n=p.size(),t=1;
    T ans=0;p.push_back(p[0]);
    for(int i=0;i<n;i++){
      point<T> now=p[i+1]-p[i];
      while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i]))t=(t+1)%n;
      ans=max(ans,(p[i]-p[t]).abs2());
    }
    return p.pop_back(),ans;
  }
  T min_cover_rectangle(){//最小覆蓋矩形
```

```cpp
    int n=p.size(),t=1,r=1,l;
    if(n<3)return 0;//也可以做最小周長矩形
    T ans=1e99;p.push_back(p[0]);
    for(int i=0;i<n;i++){
      point<T> now=p[i+1]-p[i];
      while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i]))t=(t+1)%n;
      while(now.dot(p[r+1]-p[i])>now.dot(p[r]-p[i]))r=(r+1)%n;
      if(!i)l=r;
      while(now.dot(p[l+1]-p[i])<=now.dot(p[l]-p[i]))l=(l+1)%n;
      T d=now.abs2();
      T tmp=now.cross(p[t]-p[i])*(now.dot(p[r]-p[i])-now.dot(p[l]-p[i]))/d;
      ans=min(ans,tmp);
    }
    return p.pop_back(),ans;
  }
  T dis2(polygon &pl){//凸包最近距離平方
    vector<point<T> > &P=p,&Q=pl.p;
    int n=P.size(),m=Q.size(),l=0,r=0;
  for(int i=0;i<n;++i)if(P[i].y<P[l].y)l=i;
  for(int i=0;i<m;++i)if(Q[i].y<Q[r].y)r=i;
    P.push_back(P[0]),Q.push_back(Q[0]);
    T ans=1e99;
    for(int i=0;i<n;++i){
      while((P[l]-P[l+1]).cross(Q[r+1]-Q[r])<0)r=(r+1)%m;
      ans=min(ans,line<T>(P[l],P[l+1]).seg_dis2(line<T>(Q[r],Q[r+1])));
      l=(l+1)%n;
    }
    return P.pop_back(),Q.pop_back(),ans;
  }
  static char sign(const point<T>&t){
    return (t.y==0?t.x:t.y)<0;
  }
  static bool angle_cmp(const line<T>& A,const line<T>& B){
    point<T> a=A.p2-A.p1,b=B.p2-B.p1;
    return sign(a)<sign(b)||(sign(a)==sign(b)&&a.cross(b)>0);
  }
  int halfplane_intersection(vector<line<T> > &s){//半平面交
    sort(s.begin(),s.end(),angle_cmp);//線段左側為該線段半平面
    int L,R,n=s.size();
    vector<point<T> > px(n);
    vector<line<T> > q(n);
    q[L=R=0]=s[0];
    for(int i=1;i<n;++i){
      while(L<R&&s[i].ori(px[R-1])<=0)--R;
      while(L<R&&s[i].ori(px[L])<=0)++L;
      q[++R]=s[i];
      if(q[R].parallel(q[R-1])){
        --R;
        if(q[R].ori(s[i].p1)>0)q[R]=s[i];
      }
      if(L<R)px[R-1]=q[R-1].line_intersection(q[R]);
    }
    while(L<R&&q[L].ori(px[R-1])<=0)--R;
    p.clear();
    if(R-L<=1)return 0;
    px[R]=q[R].line_intersection(q[L]);
    for(int i=L;i<=R;++i)p.push_back(px[i]);
    return R-L+1;
  }
};
template<typename T>
struct triangle{
  point<T> a,b,c;
  triangle(){}
  triangle(const point<T> &a,const point<T> &b,const point<T> &c):a(a),b(b),c(c){}
  T area()const{
    T t=(b-a).cross(c-a)/2;
    return t>0?t:-t;
  }
  point<T> barycenter()const{//重心
    return (a+b+c)/3;
  }
  point<T> circumcenter()const{//外心
    static line<T> u,v;
    u.p1=(a+b)/2;
    u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-b.x);
    v.p1=(a+c)/2;
    v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-c.x);
    return u.line_intersection(v);
  }
  point<T> incenter()const{//內心
    T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2()),C=sqrt((a-b).abs2());
    return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+B*b.y+C*c.y)/(A+B+C);
  }
  point<T> perpencenter()const{//垂心
    return barycenter()*3-circumcenter()*2;
  }
};
template<typename T>
struct point3D{
  T x,y,z;
  point3D(){}
  point3D(const T&x,const T&y,const T&z):x(x),y(y),z(z){}
  point3D operator+(const point3D &b)const{
    return point3D(x+b.x,y+b.y,z+b.z);}
  point3D operator-(const point3D &b)const{
    return point3D(x-b.x,y-b.y,z-b.z);}
  point3D operator*(const T &b)const{
    return point3D(x*b,y*b,z*b);}
  point3D operator/(const T &b)const{
    return point3D(x/b,y/b,z/b);}
  bool operator==(const point3D &b)const{
    return x==b.x&&y==b.y&&z==b.z;}
  T dot(const point3D &b)const{
    return x*b.x+y*b.y+z*b.z;}
  point3D cross(const point3D &b)const{
    return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);}
  T abs2()const{//向量長度的平方
    return dot(*this);}
  T area2(const point3D &b)const{//和b、原點圍成面積的平方
    return cross(b).abs2()/4;}
};
template<typename T>
struct line3D{
  point3D<T> p1,p2;
  line3D(){}
  line3D(const point3D<T> &p1,const point3D<T> &p2):p1(p1),p2(p2){}
  T dis2(const point3D<T> &p,bool is_segment=0)const{//點跟直線/線段的距離平方
    point3D<T> v=p2-p1,v1=p-p1;
    if(is_segment){
      point3D<T> v2=p-p2;
      if(v.dot(v1)<=0)return v1.abs2();
      if(v.dot(v2)>=0)return v2.abs2();
    }
    point3D<T> tmp=v.cross(v1);
    return tmp.abs2()/v.abs2();
  }
  pair<point3D<T>,point3D<T> > closest_pair(const line3D<T> &l)const{
    point3D<T> v1=(p1-p2),v2=(l.p1-l.p2);
    point3D<T> N=v1.cross(v2),ab(p1-l.p1);
    //if(N.abs2()==0)return NULL;平行或重合
    T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//最近點對距離
    point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.cross(d2),G=l.p1-p1;
    T t1=(G.cross(d2)).dot(D)/D.abs2();
    T t2=(G.cross(d1)).dot(D)/D.abs2();
    return make_pair(p1+d1*t1,l.p1+d2*t2);
  }
  bool same_side(const point3D<T> &a,const point3D<T> &b)const{
    return (p2-p1).cross(a-p1).dot((p2-p1).cross(b-p1))>0;
  }
};
template<typename T>
struct plane{
  point3D<T> p0,n;//平面上的點和法向量
  plane(){}
  plane(const point3D<T> &p0,const point3D<T> &n):p0(p0),n(n){}
  T dis2(const point3D<T> &p)const{//點到平面距離的平方
    T tmp=(p-p0).dot(n);
    return tmp*tmp/n.abs2();
  }
  point3D<T> projection(const point3D<T> &p)const{
    return p-n*(p-p0).dot(n)/n.abs2();
  }
  point3D<T> line_intersection(const line3D<T> &l)const{
    T tmp=n.dot(l.p2-l.p1);//等於0表示平行或重合該平面
    return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/tmp);
  }
  line3D<T> plane_intersection(const plane &pl)const{
    point3D<T> e=n.cross(pl.n),v=n.cross(e);
    T tmp=pl.n.dot(v);//等於0表示平行或重合該平面
    point3D<T> q=p0+(v*(pl.n.dot(pl.p0-p0))/tmp);
    return line3D<T>(q,q+e);
  }
};
```

1.2 MinCircleCover

```cpp
const double eps = 1e-10;
int sign(double a){
  return fabs(a)<eps?0:a>0?1:-1;
}
template<typename T>
T len(point<T> p){
  return sqrt(p.dot(p));
}
template<typename T>
point<T> findCircumcenter(point<T> A,point<T> B,point<T> C){
  point<T> AB = B-A;
  point<T> AC = C-A;
  T AB_len_sq = AB.x*AB.x+AB.y*AB.y;
  T AC_len_sq = AC.x*AC.x+AC.y*AC.y;
  T D = AB.x*AC.y-AB.y*AC.x;
  T X = A.x+(AC.y*AB_len_sq-AB.y*AC_len_sq)/(2*D);
  T Y = A.y+(AB.x*AC_len_sq-AC.x*AB_len_sq)/(2*D);
  return point<T>(X,Y);
}
template<typename T>
pair<T, point<T>> MinCircleCover(vector<point<T>> &p){
// 回傳最小覆蓋圓{半徑,中心}
  random_shuffle(p.begin(),p.end());
  int n = p.size();
  point<T> c = p[0]; T r = 0;
  for(int i=1;i<n;i++){
    if(sign(len(c-p[i])-r) > 0){ // 不在圓內
      c = p[i], r = 0;
      for(int j=0;j<i;j++){
        if(sign(len(c-p[j])-r) > 0) {
          c = (p[i]+p[j])/2.0;
          r = len(c-p[i]);
          for(int k=0;k<j;k++) {
            if(sign(len(c-p[k])-r) > 0){
  //c=triangle<T>(p[i],p[j],p[k]).circumcenter();
              c = findCircumcenter(p[i],p[j],p[k]);
              r = len(c-p[i]);
            }
          }
        }
      }
    }
  }
  return make_pair(r,c);
}
```

1.3 最近點對

```
template<typename _IT=point<T>* >
T cloest_pair(_IT L, _IT R){
  if(R-L <= 1) return INF;
  _IT mid = L+(R-L)/2;
  T x = mid->x;
  T d = min(cloest_pair(L,mid),cloest_pair(mid,R));
  inplace_merge(L, mid, R, ycmp);
  static vector<point> b; b.clear();
  for(auto u=L;u<R;++u){
    if((u->x-x)*(u->x-x)>=d) continue;
    for(auto v=b.rbegin();v!=b.rend();++v){
      T dx=u->x-v->x, dy=u->y-v->y;
      if(dy*dy>=d) break;
      d=min(d,dx*dx+dy*dy);
    }
    b.push_back(*u);
  }
  return d;
}
T closest_pair(vector<point<T>> &v){
  sort(v.begin(),v.end(),xcmp);
  return closest_pair(v.begin(),v.end());
}
```

2 DP

2.1 basic DP

```
// 0/1背包問題
for(int i=0;i<n;i++) {
    for(int k = W; k >= w[i]; k--) {
        dp[k] = max(dp[k],dp[k-w[i]]+v[i]);
    }
    //因為不能重複拿，所以要倒回來
}
//無限背包問題
dp[0] = 1;
for(int i=0;i<n;i++) {
    int a;cin>>a;
    for(int k=a;k<=m;k++) {
        dp[k] += dp[k-a];
        if(dp[k]>=mod) dp[k] -= mod;
    }
}
//LIS問題
for(int i=0;i<n;i++) {
    cin>>x;
    auto it = lower_bound(dp.begin(),dp.end(),x);
    if(it == dp.end()) {
        dp.emplace_back(x);
    }
    else {
        *it = x;
    }
}
cout<<dp.size();
//LCS問題
#include<bits/stdc++.h>
using namespace std;
signed main() {
    string a,b;
    cin>>a>>b;
    vector<vector<int>> dp(a.size()+1,vector<int> (b.size()+1,0));
    vector<vector<pair<int,int>>> pre(a.size()+1,vector<pair<int,int>> (b.size()+1));
    for(int i=0;i<a.size();i++) {
        for(int j=0;j<b.size();j++) {
            if(a[i] == b[j]) {
                dp[i+1][j+1] = dp[i][j] + 1;
                pre[i+1][j+1] = {i,j};
            }
            else if(dp[i+1][j] >= dp[i][j+1]) {
                dp[i+1][j+1] = dp[i+1][j];
                pre[i+1][j+1] = {i+1,j};
            }
            else {
                dp[i+1][j+1] = dp[i][j+1];
                pre[i+1][j+1] = {i,j+1};
            }
        }
    }
    int index1 = a.size(), index2 = b.size();
    string ans;
    while(index1>0&&index2>0) {
        if(pre[index1][index2] == make_pair(index1-1,index2-1)) {
            ans+=a[index1-1];
        }

        pair<int,int> u = pre[index1][index2];
        index1= u.first;
        index2= u.second;
    }
    for(int i=ans.size()-1;i>=0;i--)cout<<ans[i];
    return 0;
}
```

2.2 DP on Graph

```
//G.Longest Path
vector<vector<int>> G;
vector<int> in;
int n, m;
cin >> n >> m;
G.assign(n + 1, {});
in.assign(n + 1, 0);
while (m--) {
  int u, v;
  cin >> u >> v;
  G[u].emplace_back(v);
  ++in[v];
}
int solve(int n) {
  vector<int> DP(G.size(), 0);
  vector<int> Q;
  for (int u = 1; u <= n; ++u)
    if (in[u] == 0)
      Q.emplace_back(u);
  for (size_t i = 0; i < Q.size(); ++i) {
    int u = Q[i];
    for (auto v : G[u]) {
      DP[v] = max(DP[v], DP[u] + 1);
      if (--in[v] == 0)
        Q.emplace_back(v);
    }
  }
  return *max_element(DP.begin(), DP.end());
}
//max_indepent_set on tree
vector<int> DP[2];
int dfs(int u, int pick, int parent = -1) {
  if (u == parent) return 0;
  if (DP[pick][u]) return DP[pick][u];
  if (Tree[u].size() == 1) return pick; //葉子
  for (auto v : Tree[u]) {
    if (pick == 0) {
      DP[pick][u] += max(dfs(v, 0, u), dfs(v, 1, u));
    } else {
      DP[pick][u] += dfs(v, 0, u);
    }
  }
  return DP[pick][u] += pick;
}
int solve(int n) {
  DP[0] = DP[1] = vector<int>(n + 1, 0);
  return max(dfs(1, 0), dfs(1, 1));
}
//Traveling Salesman // AtCoder
#include<bits/stdc++.h>
using namespace std;

const int INF = 1e9;
int cost(vector<tuple<int,int,int>> &point, int from, int to) {
  auto [x,y,z] = point[from];
  auto [X,Y,Z] = point[to];
  return abs(X-x)+abs(Y-y)+max(0,Z-z);
}//從一個點走到另一個點的花費

signed main() {
    int n;cin>>n;
    vector<tuple<int,int,int>> point(n);
    for(auto &[x,y,z]:point) {
        cin>>x>>y>>z;
    }
    vector<vector<int>> dp(1<<n,vector<int>(n,INF));
    //1<<n(2^n)代表1~n的所有子集，代表走過的點
    //n代表走到的最後一個點
    dp[0][0] = 0;
    for(int i=1;i<(1<<n);i++) {
        for(int j=0;j<n;j++) {
            if(i & (1<<j)) {
                //j是走到的最後一個點，必須要在i裡面
                for(int k=0;k<n;k++) {
                    dp[i][j] = min(dp[i][j],dp[i-(1<<j)][k]+cost(point,k,j));
                    //i集合裡面走到j = i/{j}集合裡走到k，再從k走到j
                }
            }
            //cout<<dp[i][j]<<' ';
        }
        //cout<<endl;
    }
    cout<<dp[(1<<n)-1][0];//每個都要走到，要走回1
    return 0;
}
```

2.3 LineContainer

```
// Usually used for DP 斜率優化
template<class T>
T floor_div(T a, T b) {
  return a / b - ((a ^ b) < 0 && a % b != 0);
}

template<class T>
T ceil_div(T a, T b) {
  return a / b + ((a ^ b) > 0 && a % b != 0);
}

namespace line_container_internal {

struct line_t {
  mutable long long k, m, p;

  inline bool operator<(const line_t& o) const { return k < o.k; }
  inline bool operator<(long long x) const { return p < x; }
};

} // line_container_internal

template<bool MAX>
struct line_container : std::multiset<line_container_internal::line_t, std::less<>> {
  static const long long INF = std::numeric_limits<long long>::max();

  bool isect(iterator x, iterator y) {
    if(y == end()) {
      x->p = INF;
      return 0;
    }
    if(x->k == y->k) {
      x->p = (x->m > y->m ? INF : -INF);
```

```cpp
    } else {
      x->p = floor_div(y->m - x->m, x->k - y->k);
    }
    return x->p >= y->p;
  }

  void add_line(long long k, long long m) {
    if(!MAX) {
      k = -k;
      m = -m;
    }
    auto z = insert({k, m, 0}), y = z++, x = y;
    while(isect(y, z)) {
      z = erase(z);
    }
    if(x != begin() && isect(--x, y)) {
      isect(x, y = erase(y));
    }
    while((y = x) != begin() && (--x)->p >= y->p) {
      isect(x, erase(y));
    }
  }

  long long get(long long x) {
    assert(!empty());
    auto l = *lower_bound(x);
    return (l.k * x + l.m) * (MAX ? +1 : -1);
  }
};
```

2.4 單調隊列優化

```cpp
long long solve(vector<int> a, int N, int K) {
  vector<long long> DP(N + 1);
  deque<int> dq(1);
  for (int i = 1; i <= N; ++i) {
    while (dq.front() < i - K)
      dq.pop_front();
    DP[i] = DP[dq.front()] + a[i];
    while (dq.size() && DP[dq.back()] > DP[i])
      dq.pop_back();
    dq.push_back(i);
  }
  long long ans = INF;
  for (int i = N - K + 1; i <= N; ++i)
    ans = min(ans, DP[i]);
  return ans;
}
```

2.5 整體二分

```cpp
void compute(int L, int R, int optL, int optR) {
  if (L > R)
    return;
  int mid = L + (R - L) / 2;
  DP[mid] = INF;
  int opt = -1;
  for (int k = optL; k <= min(mid - 1, optR); k++) {
    if (DP[mid] > f(k) + w(k, mid)) {
      DP[mid] = f(k) + w(k, mid);
      opt = k;
    }
  }
  compute(L, mid - 1, optL, opt);
  compute(mid + 1, R, opt, optR);
}
// compute(1, n, 0, n);
```

2.6 斜率優化-動態凸包

```cpp
struct Line
{
    mutable ll a, b, l;
    Line(ll _a, ll _b, ll _l) : a(_a), b(_b), l(_l) {}
    bool operator<(const Line &rhs) const
    {
        return make_pair(-a, -b) < make_pair(-rhs.a, -rhs.b);
    }
    bool operator<(ll rhs_l) const
    {
        return l < rhs_l;
    }
};

struct ConvexHullMin : std::multiset<Line, std::less<>>
{
    static const ll INF = (1ll << 60);
    static ll DivCeil(ll a, ll b)
    {
        return a / b - ((a ^ b) < 0 && a % b);
    }
    bool Intersect(iterator x, iterator y)
    {
        if (y == end())
        {
            x->l = INF;
            return false;
        }
        if (x->a == y->a)
        {
            x->l = x->b < y->b ? INF : -INF;
        }
        else
        {
            x->l = DivCeil(y->b - x->b, x->a - y->a);
        }
        return x->l >= y->l;
    }
    void Insert(ll a, ll b)
    {
        auto z = insert(Line(a, b, 0)), y = z++, x = y;
        while (Intersect(y, z))
            z = erase(z);
        if (x != begin() && Intersect(--x, y))
            Intersect(x, y = erase(y));
        while ((y = x) != begin() && (--x)->l >= y->l)
            Intersect(x, erase(y));
    }
    ll query(ll x) const
    {
        auto l = *lower_bound(x);
        return l.a * x + l.b;
    }
} convexhull;

const ll maxn = 200005;
ll s[maxn];
ll f[maxn];
ll dp[maxn];
// CSES monster game2
int main()
{   Crbubble
    ll n,m,i,k,t;
    cin >> n >> f[0];
    for(i=1;i<=n;i++) cin >> s[i];
    for(i=1;i<=n;i++) cin >> f[i];
    convexhull.Insert(f[0],0);
    for(i=1;i<=n;i++)
    {
        dp[i] = convexhull.query(s[i]);
        convexhull.Insert(f[i],dp[i]);
    }
    cout << dp[n] << endl;
    return 0;
}
```

3 Data Structure

3.1 2D BIT

```cpp

//2維BIT
#define lowbit(x) (x&-x)

class BIT {
    int n;
    vector<int> bit;

public:
    void init(int _n) {
        n = _n;
        bit.resize(n);
        for(auto &b : bit) b = 0;
    }
    int query(int x) const {
        int sum = 0;
        for(; x; x -= lowbit(x))
            sum += bit[x];
        return sum;
    }
    void modify(int x, int val) {
        for(; x <= n; x += lowbit(x))
            bit[x] += val;
    }
};

class BIT2D {
    int m;
    vector<BIT> bit1D;

public:
    void init(int _m, int _n) {
        m = _m;
        bit1D.resize(m);
        for(auto &b : bit1D) b.init(_n);
    }
    int query(int x, int y) const {
        int sum = 0;
        for(; x; x-= lowbit(x))
            sum += bit1D[x].query(y);
        return sum;
    }
    void modify(int x, int y, int val) {
        for(; x <= m; x += lowbit(x))
            bit1D[x].modify(y,val);
    }
};
```

3.2 BinaryTrie

```cpp
template<class T>
struct binary_trie {
public:
  binary_trie() {
    new_node();
  }

  void clear() {
    trie.clear();
    new_node();
  }

  void insert(T x) {
    for(int i = B - 1, p = 0; i >= 0; i--) {
      int y = x >> i & 1;
      if(trie[p].go[y] == 0) {
        trie[p].go[y] = new_node();
      }
      p = trie[p].go[y];
      trie[p].cnt += 1;
    }
  }

  void erase(T x) {
    for(int i = B - 1, p = 0; i >= 0; i--) {
      p = trie[p].go[x >> i & 1];
      trie[p].cnt -= 1;
    }
  }

  bool contains(T x) {
    for(int i = B - 1, p = 0; i >= 0; i--) {
      p = trie[p].go[x >> i & 1];
      if(trie[p].cnt == 0) {
        return false;
      }
    }
    return true;
  }

  T get_min() {
    return get_xor_min(0);
  }

  T get_max() {
    return get_xor_max(0);
  }

  T get_xor_min(T x) {
    T ans = 0;
    for(int i = B - 1, p = 0; i >= 0; i--) {
      int y = x >> i & 1;
      int z = trie[p].go[y];
      if(z > 0 && trie[z].cnt > 0) {
        p = z;
      } else {
        ans |= T(1) << i;
        p = trie[p].go[y ^ 1];
      }
    }
    return ans;
  }

  T get_xor_max(T x) {
    T ans = 0;
    for(int i = B - 1, p = 0; i >= 0; i--) {
      int y = x >> i & 1;
      int z = trie[p].go[y ^ 1];
      if(z > 0 && trie[z].cnt > 0) {
        ans |= T(1) << i;
        p = z;
      } else {
        p = trie[p].go[y];
      }
    }
    return ans;
  }

private:
  static constexpr int B = sizeof(T) * 8;

  struct Node {
    std::array<int, 2> go = {};
    int cnt = 0;
  };

  std::vector<Node> trie;

  int new_node() {
    trie.emplace_back();
    return (int) trie.size() - 1;
  }
};
```

3.3 BIT

```
#define lowbit(x) x & -x

void modify(vector<int> &bit, int idx, int val) {
  for(int i = idx; i <= bit.size(); i+= lowbit(i)) bit[i] += val;
}

int query(vector<int> &bit, int idx) {
  int ans = 0;
  for(int i = idx; i > 0; i-= lowbit(i)) ans += bit[i];
  return ans;
}

int findK(vector<int> &bit, int k) {
  int idx = 0, res = 0;
  int mx = __lg(bit.size()) + 1;
  for(int i = mx; i >= 0; i--) {
    if((idx | (1<<i)) > bit.size()) continue;
    if(res + bit[idx | (1<<i)] < k) {
      idx = (idx | (1<<i));
      res += bit[idx];
    }
  }
  return idx + 1;
}
//O(n)建bit
for (int i = 1; i <= n; ++i) {
    bit[i] += a[i];
    int j = i + lowbit(i);
    if (j <= n) bit[j] += bit[i];
}
```

3.4 DSU

```
struct DSU {
  vector<int> dsu, sz;
  DSU(int n) {
    dsu.resize(n + 1);
    sz.resize(n + 1, 1);
    for (int i = 0; i <= n; i++) dsu[i] = i;
  }
  int find(int x) {
    return (dsu[x] == x ? x : dsu[x] = find(dsu[x]));
  }
  int unite(int a, int b) {
    a = find(a), b = find(b);
    if(a == b) return 0;
    if(sz[a] > sz[b]) swap(a, b);
    dsu[a] = b;
    sz[b] += sz[a];
          return 1;
  }
};
```

3.5 Dynamic Segment Tree

```
using ll = long long;
struct node {
  node *l, *r; ll sum;
  void pull() {
    sum = 0;
    for(auto x : {l, r}) if(x) sum += x->sum;
  }
  node(int v = 0): sum(v) {l = r = nullptr;}
};

void upd(node*& o, int x, ll v, int l, int r) {
  if(!o) o = new node;
  if(l == r) return o->sum += v, void();
  int m = (l + r) / 2;
  if(x <= m) upd(o->l, x, v, l, m);
  else upd(o->r, x, v, m+1, r);
  o->pull();
}

ll qry(node* o, int ql, int qr, int l, int r) {
  if(!o) return 0;
  if(ql <= l && r <= qr) return o->sum;
  int m = (l + r) / 2; ll ret = 0;
  if(ql <= m) ret += qry(o->l, ql, qr, l, m);
  if(qr > m) ret += qry(o->r, ql, qr, m+1, r);
  return ret;
}
```

3.6 Kruskal

```
vector<tuple<int,int,int>> Edges;
int kruskal(int N) {
    int cost = 0;
    sort(Edges.begin(), Edges.end());

    DisjointSet ds(N);

    sort(Edges.begin(), Edges.end());
    for(auto [w, s, t] : Edges) {
        if (!ds.same(s, t)) {
            cost += w;
            ds.unit(s, t);
        }
    }
    return cost;
}
```

3.7 Lazytag Segment Tree

```
using ll = long long;
const int N = 2e5 + 5;
#define lc(x) (x << 1)
#define rc(x) (x << 1 | 1)
ll seg[N << 2], tag[N << 2];
int n;

void pull(int id) {
  seg[id] = seg[lc(id)] + seg[rc(id)];
}

void push(int id, int l, int r) {
  if (tag[id]) {
    int m = (l + r) >> 1;
    tag[lc(id)] += tag[id], tag[rc(id)] += tag[id];
    seg[lc(id)] += (m - l + 1) * tag[id], seg[rc(id)] += (r - m) * tag[id];
    tag[id] = 0;
  }
}

void upd(int ql, int qr, ll v, int l = 1, int r = n, int id = 1) {
  if (ql <= l && r <= qr) return tag[id] += v, seg[id] += (r - l + 1) * v, void();
  push(id, l, r);
  int m = (l + r) >> 1;
  if (ql <= m) upd(ql, qr, v, l, m, lc(id));
  if (qr > m) upd(ql, qr, v, m + 1, r, rc(id));
  pull(id);
}

ll qry(int ql, int qr, int l = 1, int r = n, int id = 1) {
  if (ql <= l && r <= qr) return seg[id];
  push(id, l, r);
  int m = (l + r) >> 1; ll ret = 0;
  if (ql <= m)  ret += qry(ql, qr, l, m, lc(id));
  if (qr > m) ret += qry(ql, qr, m + 1, r, rc(id));
  return ret;
}
```

3.8 monotonic queue

```

vector<int> maxSlidingWindow(vector<int> &num, int k) {
    deque<int> dq;
    vector<int> ans;
    for(int i = 0; i < num.size(); i++) {
        while(dq.size() && dq.front() <= i - k) dq.pop_front();
        while(dq.size() && num[dq.back()] < num[i]) dq.pop_back();
        dq.emplace_back(i);
        if(i >= k - 1) ans.emplace_back(num[dq.front()]);
    }
    return ans;
}
```

3.9 monotonic stack

```cpp

long long maxRectangle(vector<int> &h) {
    h.emplace_back(0);
    stack<pair<int,int>> stick;
    long long ans = 0;
    for(int i = 0; i < h.size(); i++) {
        int corner = i;
        while(stick.size() && stick.top().first >= h[i]) {
            corner = stick.top().second;
            ans = max(ans, 1LL * (i - corner) * stick.top().first);
            stick.pop();
        }
        stick.emplace(h[i],corner);
    }
    return ans;
}
```

3.10 pbds

```cpp
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;

template <class T>
using ordered_set = tree<T, null_type, less<T>, rb_tree_tag, tree_order_statistics_node_update>;

template <class T>
// ordered_multiset: do not use erase method, use myerase() instead
using ordered_multiset = tree<T, null_type, less_equal<T>, rb_tree_tag, tree_order_statistics_node_update>;

template<class T>
void myerase(ordered_multiset<T> &ss, T v)
{
    T rank = ss.order_of_key(v);    // Number of elements that are less than v in ss
    auto it = ss.find_by_order(rank); // Iterator that points to the element which index = rank
    ss.erase(it);
}
```

3.11 Persistent DSU

```cpp
int rk[200001] = {};
struct Persistent_DSU{
  rope<int>*p;
  int n;
  Persistent_DSU(int _n = 0):n(_n){
    if(n==0)return;
    p = new rope<int>;
    int tmp[n+1] = {};
    for(int i = 1;i<=n;++i)tmp[i] = i;
    p->append(tmp,n+1);
  }
  Persistent_DSU(const Persistent_DSU &tmp){
    p = new rope<int>(*tmp.p);
    n = tmp.n;
  }
  int Find(int x){
    int px = p->at(x);
    return px==x?x:Find(px);
  }
  bool Union(int a,int b){
    int pa = Find(a),pb = Find(b);
    if(pa==pb)return 0;
    if(rk[pa]<rk[pb])swap(pa,pb);
    p->replace(pb,pa);
    if(rk[pa]==rk[pb])rk[pa]++;
    return 1;
  }
};
```

3.12 Persistent Segment Tree

```cpp
using ll = long long;
int n;

struct node {
  node *l, *r; ll sum;
  void pull() {
    sum = 0;
    for (auto x : {l, r})
      if (x) sum += x->sum;
  }
  node(int v = 0): sum(v) {l = r = nullptr;}
} *root = nullptr;

void upd(node *prv, node* cur, int x, int v, int l = 1, int r = n) {
  if (l == r) return cur->sum = v, void();
  int m = (l + r) >> 1;
  if (x <= m) cur->r = prv->r, upd(prv->l, cur->l = new node, x, v, l, m);
  else cur->l = prv->l, upd(prv->r, cur->r = new node, x, v, m + 1, r);
  cur->pull();
}

ll qry(node* a, node* b, int ql, int qr, int l = 1, int r = n) {
  if (ql <= l && r <= qr) return b->sum - a->sum;
  int m = (l + r) >> 1; ll ret = 0;
  if (ql <= m) ret += qry(a->l, b->l, ql, qr, l, m);
  if (qr > m) ret += qry(a->r, b->r, ql, qr, m + 1, r);
  return ret;
}
```

3.13 Prim

```cpp
int cost[MAX_V][MAX_V];//Edge的權重(不存在時為INF)
int mincost[MAX_V];//來自集合X的邊的最小權重
bool used[MAX_V];//頂點i是否包含在X之中
int V;//頂點數

int prim() {
    for(int i = 0; i < v; i++) {
        mincost[i] = INF;
        used[i] = false;
    }
    mincost[0] = 0;
    int res = 0;
    while(true) {
        int v = -1;
        //從不屬於X的頂點中尋找會讓來自X的邊之權重最小的頂點
        for(int u = 0; u < V; u++) {
            if(!used[u] && (v==-1 || mincost[u] < mincost[v])) v = u;
        }
        if(v == -1) break;
        used[v] = true;//將頂點v追加至X
        res += mincost[v];//加上邊的權重
        for(int u = 0; u < V; u++) {
            mincost[u] = min(mincost[u],cost[v][u]);
        }
    }
    return res;
}
```

3.14 SegmentTree

```cpp

//build
const int N = 100000 + 9;
int a[N];//葉
int seg[4 * N];
void bulid(int id, int l, int r) { // 編號為id 的節點，存的區間為[l, r]
    if (l == r) {
        seg[id] = a[l]; // 葉節點的值
        return;
    }
    int mid = (l + r) / 2; // 將區間切成兩半
    build(id * 2, l, mid); // 左子節點
    build(id * 2 + 1, mid + 1, r); // 右子節點
    seg[id] = seg[id * 2] + seg[id * 2 + 1]
}

//區間查詢

int query(int id, int l, int r, int ql, int qr) {
    if (r < ql || qr < l) return 0;//若目前的區間與詢問的區間的交集為空的話，return 0
    if (ql <= l && r <= qr) return seg[id];//若目前的區間是詢問的區間的子集的話，則終止，並回傳當前節點的答案
    int mid = (l + r) / 2;
    return query(id * 2, l, mid, ql, qr) //左
        + query(id * 2 + 1, mid + 1, r, ql, qr);//右
    //否則，往左、右進行遞迴
}


//單點修改

void modify(int id, int l, int r, int i, int x) {
    if (l == r) {
        seg[id] = x; // 將a[i]改成x
        //seg[id] += x; // 將a[i]加上x
        return;
    }
    int mid = (l + r) / 2;
    // 根據修改的點在哪裡，來決定要往哪個子樹進行DFS
    if (i <= mid) modify(id * 2, l, mid, i, x);//左
    else modify(id * 2 + 1, mid + 1, r, i, x);//右
    seg[id] = seg[id * 2] + seg[id * 2 + 1];
}
```

3.15 sparse table

```cpp
//CSES Static Range Minimum Queries
#include<bits/stdc++.h>
using namespace std;
#define inf 1e9
vector<vector<int>> st;

void build_sparse_table(int n) {
  st.assign(__lg(n)+1,vector<int> (n+1,inf));
  for(int i=1;i<=n;i++) cin>>st[0][i];
  for(int i=1;(1<<i)<=n;i++) {
    for(int j=1;j + (1<<i) - 1 <= n;j++) {
      st[i][j] = min(st[i-1][j],st[i-1][j+(1<<(i-1))]);
    }
  }
}

int query(int l, int r) {
  int k = __lg(r - l + 1);
  return min(st[k][l],st[k][r-(1<<k)+1]);
}

signed main() {
  int n,q;cin>>n>>q;
  build_sparse_table(n);
  while(q--) {
    int l,r;cin>>l>>r;
    cout<<query(l,r)<<'\n';
  }
}
```

3.16 TimingSegmentTree

```cpp
template<class T,class D>struct timing_segment_tree{
  struct node{
    int l,r;
    vector<T>opt;
  };
  vector<node>arr;
  void build(int l,int r,int idx = 1){
    if(idx==1)arr.resize((r-l+1)<<2);
    if(l==r){
      arr[idx].l = arr[idx].r = l;
      arr[idx].opt.clear();
      return;
    }
    int m = (l+r)>>1;
    build(l,m,idx<<1);
    build(m+1,r,idx<<1|1);
    arr[idx].l = l,arr[idx].r = r;
    arr[idx].opt.clear();
  }
  void update(int ql,int qr,T k,int idx = 1){
    if(ql<=arr[idx].l and arr[idx].r<=qr){
      arr[idx].opt.push_back(k);
      return;
    }
    int m = (arr[idx].l+arr[idx].r)>>1;
    if(ql<=m)update(ql,qr,k,idx<<1);
    if(qr>m)update(ql,qr,k,idx<<1|1);
  }
  void dfs(D &d,vector<int>&ans,int idx = 1){
    int cnt = 0;
    for(auto [a,b]:arr[idx].opt){
      if(d.Union(a,b))cnt++;
    }
    if(arr[idx].l==arr[idx].r)ans[arr[idx].l] = d.comps;
    else{
      dfs(d,ans,idx<<1);
      dfs(d,ans,idx<<1|1);
    }
    while(cnt--)d.undo();
  }
};
```

3.17 回滾並查集

```cpp
struct dsu_undo{
  vector<int>sz,p;
  int comps;
  dsu_undo(int n){
    sz.assign(n+5,1);
    p.resize(n+5);
    for(int i = 1;i<=n;++i)p[i] = i;
    comps = n;
  }
  vector<pair<int,int>>opt;
  int Find(int x){
    return x==p[x]?x:Find(p[x]);
  }
  bool Union(int a,int b){
    int pa = Find(a),pb = Find(b);
    if(pa==pb)return 0;
    if(sz[pa]<sz[pb])swap(pa,pb);
    sz[pa]+=sz[pb];
    p[pb] = pa;
    opt.push_back({pa,pb});
    comps--;
    return 1;
  }
  void undo(){
        auto [pa,pb] = opt.back();
        opt.pop_back();
        p[pb] = pb;
        sz[pa]-=sz[pb];
        comps++;
  }
};
```

3.18 掃描線 + 線段樹

```cpp
//CSES Area of Rectangle
#include <bits/stdc++.h>
#define pb push_back
#define int long long
#define mid ((l + r) >> 1)
#define lc (p << 1)
#define rc ((p << 1) | 1)
using namespace std;
struct ooo{
    int x, l, r, v;
};
const int inf = 1e6;
array<int, 8000004> man, tag, cnt;
vector<ooo> Q;
bool cmp(ooo a, ooo b){
    return a.x < b.x;
}
void pull(int p){
    man[p] = min(man[lc], man[rc]);
    if(man[lc] < man[rc]) cnt[p] = cnt[lc];
    else if(man[rc] < man[lc]) cnt[p] = cnt[rc];
    else cnt[p] = cnt[lc] + cnt[rc];
}
void push(int p){
    man[lc] += tag[p];
    man[rc] += tag[p];
    tag[lc] += tag[p];
    tag[rc] += tag[p];
    tag[p] = 0;
}
void build(int p, int l, int r){
    if(l == r){
        cnt[p] = 1;
        return;
    }
    build(lc, l, mid);
    build(rc, mid + 1, r);
    pull(p);
}
void update(int p, int l, int r, int ql, int qr, int x){
    if(ql > r || qr < l) return;
    if(ql <= l && qr >= r){
        man[p] += x;
        tag[p] += x;
        return;
    }
    push(p);
    update(lc, l, mid, ql, qr, x);
    update(rc, mid + 1, r, ql, qr, x);
    pull(p);
}
signed main(){
    int n, x1, y1, x2, y2, p = 0, sum = 0;
    cin >> n;
    for(int i = 1; i <= n; i++){
        cin >> x1 >> y1 >> x2 >> y2;
        Q.pb({x1, y1, y2 - 1, 1});
        Q.pb({x2, y1, y2 - 1, -1});
    }
    sort(Q.begin(), Q.end(), cmp);
    build(1, -inf, inf);
    for(int i = -inf; i < inf; i++){
        while(p < Q.size() && Q[p].x == i){
            auto [x, l, r, v] = Q[p++];
            update(1, -inf, inf, l, r, v);
        }
        sum += 2 * inf + 1 - cnt[1];
    }
    cout << sum << "\n";
    return 0;
}
//長方形面積
long long AreaOfRectangles(vector<tuple<int,int,int,int>>v){
  vector<tuple<int,int,int,int>>tmp;
  int L = INT_MAX,R = INT_MIN;
  for(auto [x1,y1,x2,y2]:v){
    tmp.push_back({x1,y1+1,y2,1});
    tmp.push_back({x2,y1+1,y2,-1});
    R = max(R,y2);
    L = min(L,y1);
  }
  vector<long long>seg((R-L+1)<<2),tag((R-L+1)<<2);
  sort(tmp.begin(),tmp.end());
  function<void(int,int,int,int,int,int)> update = [&](int ql,int qr,int val,int l,int r,int idx){
    if(ql<=l and r<=qr){
      tag[idx]+=val;
      if(tag[idx])seg[idx] = r-l+1;
      else if(l==r)seg[idx] = 0;
      else seg[idx] = seg[idx<<1]+seg[idx<<1|1];
      return;
    }
    int m = (l+r)>>1;
    if(ql<=m)update(ql,qr,val,l,m,idx<<1);
    if(qr>m)update(ql,qr,val,m+1,r,idx<<1|1);
    if(tag[idx])seg[idx] = r-l+1;
    else seg[idx] = seg[idx<<1]+seg[idx<<1|1];
  };
  long long last_pos = 0,ans = 0;
  for(auto [pos,l,r,val]:tmp){
    ans+=(pos-last_pos)*seg[1];
    update(l,r,val,L,R,1);
    last_pos = pos;
  }
  return ans;
}

// CSES Intersection Points
#include <bits/stdc++.h>
#define int long long
#define pb push_back
using namespace std;
struct line{
    int p, l, r;
};
const int inf = 1e6 + 1;
array<int, 2000004> BIT;
vector<line> A, Q;
bool cmp(line a, line b){
    return a.p < b.p;
}
void update(int p, int x){
    for(; p < 2000004; p += p & -p) BIT[p] += x;
}
int query(int p){
    int sum = 0;
    for(; p; p -= p & -p) sum += BIT[p];
    return sum;
}
int run(){
    int ans = 0, p = 0;
    for(auto [t, l, r] : Q){
        while(p < A.size()){
            auto [x, y, v] = A[p];
            if(x > t) break;
            update(y, v);
            p++;
        }
        ans += query(r) - query(l - 1);
    }
    return ans;
}
signed main(){
    int n, x1, x2, y1, y2;
    cin >> n;
    for(int i = 0; i < n; i++){
        cin >> x1 >> y1 >> x2 >> y2;
        x1 += inf, x2 += inf, y1 += inf, y2 += inf;
        if(x1 == x2) Q.pb({x1, y1, y2});
        else A.pb({x1, y1, 1}), A.pb({x2 + 1, y2, -1});
    }
    sort(Q.begin(), Q.end(), cmp);
  sort(A.begin(), A.end(), cmp);
  cout << run() << "\n";
  return 0;
}
```

3.19 陣列上 Treap

```cpp


struct Treap {
  Treap *lc = nullptr, *rc = nullptr;
  unsigned pri, sz;
  long long Val, Sum;
  Treap(int Val):pri(rand()),sz(1),Val(Val),Sum(Val),Tag(false) {}
  void pull();
  bool Tag;
  void push();
} *root;

inline unsigned sz(Treap *x) {
  return x ? x->sz:0;
}

inline void Treap::push() {
  if(!Tag) return ;
  swap(lc,rc);
  if(lc) lc->Tag ^= Tag;
  if(rc) rc->Tag ^= Tag;
  Tag = false;
}

inline void Treap::pull() {
  sz = 1;
  Sum = Val;
  if(lc) {
    sz += lc->sz;
    Sum += lc->Sum;
  }
  if(rc) {
    sz += rc->sz;
    Sum += rc->Sum;
  }
}

Treap *merge(Treap *a, Treap *b) {
  if(!a || !b) return a ? a : b;
  if(a->pri < b->pri) {
    a->push();
    a->rc = merge(a->rc,b);
    a->pull();
    return a;
  }
  else {
    b->push();
    b->lc = merge(a,b->lc);
    b->pull();
    return b;
  }
}

pair<Treap *,Treap *> splitK(Treap *x, unsigned K) {
  Treap *a = nullptr, *b = nullptr;
  if(!x) return {a,b};
  x->push();
  unsigned leftSize = sz(x->lc) + 1;
  if(K >= leftSize) {
    a = x;
    tie(a->rc,b) = splitK(x->rc, K - leftSize);
  }
  else {
    b = x;
    tie(a, b->lc) = splitK(x->lc, K);
  }
  x->pull();
  return {a,b};
}

Treap *init(const vector<int> &a) {
  Treap *root = nullptr;
  for(size_t i = 0;i < a.size(); i++) {
    root = merge(root,new Treap(a[i]));
  }
  return root;
}

long long query(Treap *&root, unsigned ql, unsigned qr) {
  auto [a,b] = splitK(root,ql);
  auto [c,d] = splitK(b,qr-ql+1);
  c->push();
  long long Sum = c->Sum;
  root = merge(a,merge(c,d));
  return Sum;
}

void Reverse(Treap *&root, unsigned ql, unsigned qr) {
  auto [a,b] = splitK(root,ql);
  auto [c,d] = splitK(b,qr-ql+1);
  c->Tag ^= true;
  root = merge(a, merge(c,d));
}
```

4 Flow

4.1 dinic

```cpp
template<class T>
struct Dinic{
  struct edge{
    int from, to;
    T cap;
    edge(int _from, int _to, T _cap) : from(_from), to(_to), cap(_cap) {}
  };
  int n;
  vector<edge> edges;
  vector<vector<int>> g;
  vector<int> cur, h;
  Dinic(int _n) : n(_n+1), g(_n+1) {}
  void add_edge(int u, int v, T cap){
    g[u].push_back(edges.size());
    edges.push_back(edge(u, v, cap));
    g[v].push_back(edges.size());
    edges.push_back(edge(v, u, 0));
  }
  bool bfs(int s,int t){
    h.assign(n, -1);
    h[s] = 0;
    queue<int> que;
    que.push(s);
    while(!que.empty()) {
      int u = que.front();
      que.pop();
      for(auto id : g[u]) {
        const edge& e = edges[id];
        int v = e.to;
        if(e.cap > 0 && h[v] == -1) {
          h[v] = h[u] + 1;
          if(v == t) {
            return 1;
          }
          que.push(v);
        }
      }
    }
    return 0;
  }
  T dfs(int u, int t, T f) {
    if(u == t) {
      return f;
    }
    T r = f;
    for(int& i = cur[u]; i < (int) g[u].size(); ++i) {
      int id = g[u][i];
      const edge& e = edges[id];
      int v = e.to;
      if(e.cap > 0 && h[v] == h[u] + 1) {
        T send = dfs(v, t, min(r, e.cap));
        edges[id].cap -= send;
        edges[id ^ 1].cap += send;
        r -= send;
        if(r == 0) {
          return f;
        }
      }
    }
    return f - r;
  }
  T flow(int s, int t, T f = numeric_limits<T>::max()) {
    T ans = 0;
    while(f > 0 && bfs(s, t)) {
      cur.assign(n, 0);
      T send = dfs(s, t, f);
      ans += send;
      f -= send;
    }
    return ans;
  }
  vector<pair<int,int>> min_cut(int s) {
    vector<bool> vis(n);
    vis[s] = true;
    queue<int> que;
    que.push(s);
    while(!que.empty()) {
      int u = que.front();
      que.pop();
      for(auto id : g[u]) {
        const auto& e = edges[id];
        int v = e.to;
        if(e.cap > 0 && !vis[v]) {
          vis[v] = true;
          que.push(v);
        }
      }
    }
    vector<pair<int,int>> cut;
    for(int i = 0; i < (int) edges.size(); i += 2) {
      const auto& e = edges[i];
      if(vis[e.from] && !vis[e.to]) {
        cut.push_back(make_pair(e.from, e.to));
      }
    }
    return cut;
  }
};

//CSES Distinct Routes
#include <bits/stdc++.h>

using namespace std;

struct FlowEdge {
    int v, u;
    long long cap, flow = 0;
    FlowEdge(int v, int u, long long cap) : v(v), u(u), cap(cap) {}
};

struct Dinic {
    const long long flow_inf = 1e18;
    vector<FlowEdge> edges;
    vector<vector<int>> adj;
    int n, m = 0;
    int s, t;
    vector<int> level, ptr, path;
    vector< vector<int> > paths;
    queue<int> q;

    Dinic(int n, int s, int t) : n(n), s(s), t(t) {
        adj.resize(n);
        level.resize(n);
        ptr.resize(n);
    }

    void add_edge(int v, int u, long long cap) {
        edges.emplace_back(v, u, cap);
        edges.emplace_back(u, v, 0);
        adj[v].push_back(m);
        adj[u].push_back(m + 1);
        m += 2;
    }

    bool bfs() {
        while (!q.empty()) {
            int v = q.front();
            q.pop();
            for (int id : adj[v]) {
                if (edges[id].cap - edges[id].flow < 1)
                    continue;
                if (level[edges[id].u] != -1)
                    continue;
                level[edges[id].u] = level[v] + 1;
                q.push(edges[id].u);
            }
        }
        return level[t] != -1;
    }

    long long dfs(int v, long long pushed) {
        if (pushed == 0)
            return 0;
        path.push_back(v);
        if (v == t) {
            for (int iiddxx = 0; iiddxx < pushed; ++iiddxx)
                paths.push_back(path);
            path.pop_back();
            return pushed;
        }
        for (int& cid = ptr[v]; cid < (int)adj[v].size(); cid++) {
            int id = adj[v][cid];
            int u = edges[id].u;
            if (level[v] + 1 != level[u] || edges[id].cap - edges[id].flow < 1)
                continue;
            long long tr = dfs(u, min(pushed, edges[id].cap - edges[id].flow));
            if (tr == 0)
                continue;
            edges[id].flow += tr;
            edges[id ^ 1].flow -= tr;
            path.pop_back();
            return tr;
        }
        path.pop_back();
        return 0;
    }

    long long flow() {
        long long f = 0;
        while (true) {
            fill(level.begin(), level.end(), -1);
            level[s] = 0;
            q.push(s);
            if (!bfs())
                break;
            fill(ptr.begin(), ptr.end(), 0);
            while (long long pushed = dfs(s, flow_inf)) {
                f += pushed;
            }
        }
        return f;
    }
};

int main() {
    int n, m, v, u;
    cin >> n >> m;
    Dinic D(n+1, 1, n);
    for (int i = 0; i < m; ++i) {
        cin >> v >> u;
        D.add_edge(v, u, 1);
    }
    D.flow();
    Dinic FLOW(n+1, 1, n);
    for (auto e: D.edges) {
        if (e.flow > 0) {
            FLOW.add_edge(e.v, e.u, 1);
        }
    }
    cout << FLOW.flow() << "\n";
    for (auto p: FLOW.paths) {
        cout << p.size() << "\n";
        for (auto verti: p)
            cout << verti << " ";
        cout << "\n";
    }

    return 0;
}
```

4.2 Gomory Hu

```cpp
//最小割樹+求任兩點間最小割
//0-base, root=0
LL e[MAXN][MAXN]; //任兩點間最小割
int p[MAXN]; //parent
ISAP D; // original graph
void gomory_hu(){
  fill(p, p+n, 0);
  fill(e[0], e[n], INF);
  for( int s = 1; s < n; ++s ) {
    int t = p[s];
    ISAP F = D;
    LL tmp = F.min_cut(s, t);
    for( int i = 1; i < s; ++i )
      e[s][i] = e[i][s] = min(tmp, e[t][i]);
    for( int i = s+1; i <= n; ++i )
      if( p[i] == t && F.vis[i] ) p[i] = s;
  }
}
```

4.3 ISAP with cut

```cpp
template<typename T>
struct ISAP{
  static const int MAXN=105;
  static const T INF=INT_MAX;
  int n;//點數
  int d[MAXN],gap[MAXN],cur[MAXN];
  struct edge{
    int v,pre;
    T cap,r;
    edge(int v,int pre,T cap):v(v),pre(pre),cap(cap),r(cap){}
  };
  int g[MAXN];
  vector<edge> e;
  void init(int _n){
    memset(g,-1,sizeof(int)*((n=_n)+1));
    e.clear();
  }
  void add_edge(int u,int v,T cap,bool directed=false){
    e.push_back(edge(v,g[u],cap));
    g[u]=e.size()-1;
    e.push_back(edge(u,g[v],directed?0:cap));
    g[v]=e.size()-1;
  }
  T dfs(int u,int s,int t,T CF=INF){
    if(u==t)return CF;
    T tf=CF,df;
    for(int &i=cur[u];~i;i=e[i].pre){
      if(e[i].r&&d[u]==d[e[i].v]+1){
        df=dfs(e[i].v,s,t,min(tf,e[i].r));
        e[i].r-=df;
        e[i^1].r+=df;
        if(!(tf-=df)||d[s]==n)return CF-tf;
      }
    }
    int mh=n;
    for(int i=cur[u]=g[u];~i;i=e[i].pre){
      if(e[i].r&&d[e[i].v]<mh)mh=d[e[i].v];
    }
    if(!--gap[d[u]])d[s]=n;
    else ++gap[d[u]=++mh];
    return CF-tf;
  }
  T isap(int s,int t,bool clean=true){
    memset(d,0,sizeof(int)*(n+1));
    memset(gap,0,sizeof(int)*(n+1));
    memcpy(cur,g,sizeof(int)*(n+1));
    if(clean) for(size_t i=0;i<e.size();++i)
      e[i].r=e[i].cap;
    T MF=0;
    for(gap[0]=n;d[s]<n;)MF+=dfs(s,s,t);
    return MF;
  }
  vector<int> cut_e;//最小割邊集
  bool vis[MAXN];
  void dfs_cut(int u){
    vis[u]=1;//表示u屬於source的最小割集
    for(int i=g[u];~i;i=e[i].pre)
      if(e[i].r>0&&!vis[e[i].v])dfs_cut(e[i].v);
  }
  T min_cut(int s,int t){
    T ans=isap(s,t);
    memset(vis,0,sizeof(bool)*(n+1));
    dfs_cut(s), cut_e.clear();
    for(int u=0;u<=n;++u)if(vis[u])
      for(int i=g[u];~i;i=e[i].pre)
        if(!vis[e[i].v])cut_e.push_back(i);
    return ans;
  }
};
```

4.4 MinCostMaxFlow

```cpp
template<class Cap_t, class Cost_t>
class MCMF {
public:
  struct Edge {
    int from;
    int to;
    Cap_t cap;
    Cost_t cost;
    Edge(int u, int v, Cap_t _cap, Cost_t _cost) : from(u), to(v), cap(_cap), cost(_cost) {}
  };

  static constexpr Cap_t EPS = static_cast<Cap_t>(1e-9);

  int n;
  vector<Edge> edges;
  vector<vector<int>> g;
  vector<Cost_t> d;
  vector<bool> in_queue;
  vector<int> previous_edge;

  MCMF() {}
  MCMF(int _n) : n(_n+1), g(_n+1), d(_n+1), in_queue(_n+1), previous_edge(_n+1) {}

  void add_edge(int u, int v, Cap_t cap, Cost_t cost) {
    assert(0 <= u && u < n);
    assert(0 <= v && v < n);
    g[u].push_back(edges.size());
    edges.emplace_back(u, v, cap, cost);
    g[v].push_back(edges.size());
    edges.emplace_back(v, u, 0, -cost);
  }

  bool spfa(int s, int t) {
    bool found = false;
    fill(d.begin(), d.end(), numeric_limits<Cost_t>::max());
    d[s] = 0;
    in_queue[s] = true;
    queue<int> que;
    que.push(s);
    while(!que.empty()) {
      int u = que.front();
      que.pop();
      if(u == t) {
        found = true;
      }
      in_queue[u] = false;
      for(auto& id : g[u]) {
        const Edge& e = edges[id];
        if(e.cap > EPS && d[u] + e.cost < d[e.to]) {
          d[e.to] = d[u] + e.cost;
          previous_edge[e.to] = id;
          if(!in_queue[e.to]) {
            que.push(e.to);
            in_queue[e.to] = true;
          }
        }
      }
    }
    return found;
  }

  pair<Cap_t, Cost_t> flow(int s, int t, Cap_t f = numeric_limits<Cap_t>::max()) {
    assert(0 <= s && s < n);
    assert(0 <= t && t < n);
    Cap_t cap = 0;
    Cost_t cost = 0;
    while(f > 0 && spfa(s, t)) {
      Cap_t send = f;
      int u = t;
      while(u != s) {
        const Edge& e = edges[previous_edge[u]];
        send = min(send, e.cap);
        u = e.from;
      }
      u = t;
      while(u != s) {
        Edge& e = edges[previous_edge[u]];
        e.cap -= send;
        Edge& b = edges[previous_edge[u] ^ 1];
        b.cap += send;
        u = e.from;
      }
      cap += send;
      f -= send;
      cost += send * d[t];
    }
    return make_pair(cap, cost);
  }
};
```

4.5 Property

```
最大流 = 最小割
最大獨立集 = 補圖最大團 = V - 最小頂點覆蓋
二分圖最大匹配 = 二分圖最小頂點覆蓋
二分圖最大匹配加s,t點 = 最大流
```

5 Graph

5.1 2-SAT

```cpp
struct two_sat{
  SCC s;
  vector<bool>ans;
  int have_ans = 0;
  int n;
  two_sat(int _n) : n(_n) {
    ans.resize(n+1);
    s = SCC(2*n);
  }
  int inv(int x){
    if(x>n)return x-n;
    return x+n;
  }
  void add_or_clause(int u, bool x, int v, bool y){
    if(!x)u = inv(u);
    if(!y)v = inv(v);
    s.add_edge(inv(u), v);
    s.add_edge(inv(v), u);
  }
  void check(){
    if(have_ans!=0)return;
    s.build();
    for(int i = 0;i<=n;++i){
      if(s.scc[i]==s.scc[inv(i)]){
        have_ans = -1;
        return;
      }
      ans[i] = (s.scc[i]<s.scc[inv(i)]);
    }
    have_ans = 1;
  }
};
```

5.2 Bellman Ford

```cpp
vector<tuple<int,int,int>> Edges;
int BellmanFord(int s, int e, int N) {
    const int INF = INT_MAX / 2;
    vector<int> dist(N, INF);

    dist[s] = 0;
    bool update;
    for(int i=1;i<=N;++i) {
        update = false;
        for(auto [v, u, w] : Edges)
        {
            if (dist[u] > dist[v] + w)
            {
                dist[u] = dist[v] + w;
                update = true;
            }
        }

        if (!update)
            break;
        if (i == N) // && update
            return -1; // gg !
    }
    return dist[e];
}
```

5.3 Dijkstra

```cpp
int Dijkstra(int s, int e, int N) {
    const int INF = INT_MAX / 2;
    vector<int> dist(N, INF);
    vector<bool> used(N, false);

    using T = tuple<int,int>;
    priority_queue<T, vector<T>, greater<T>> pq;

    dist[s] = 0;
    pq.emplace(0, s); // (w, e) 讓 pq 優先用 w 來比較

    while (!pq.empty()) {
        tie(std::ignore, s) = pq.top();
        pq.pop();

        if ( used[s] ) continue;
        used[s] = true; // 每一個點都只看一次

        for (auto [e, w] : V[s]) {
            if (dist[e] > dist[s] + w) {
                dist[e] = dist[s] + w;
                pq.emplace(dist[e], e);
            }
        }
    }

    return dist[e];
}
```

5.4 Dominator tree

```cpp
struct dominator_tree{
  static const int MAXN=5005;
  int n;// 1-base
  vector<int> G[MAXN], rG[MAXN];
  int pa[MAXN], dfn[MAXN], id[MAXN], dfnCnt;
  int semi[MAXN], idom[MAXN], best[MAXN];
  vector<int> tree[MAXN]; // tree here
  void init(int _n){
    n = _n;
    for(int i=1; i<=n; ++i)
      G[i].clear(), rG[i].clear();
  }
  void add_edge(int u, int v){
    G[u].push_back(v);
    rG[v].push_back(u);
  }
  void dfs(int u){
    id[dfn[u]=++dfnCnt]=u;
    for(auto v:G[u]) if(!dfn[v])
      dfs(v),pa[dfn[v]]=dfn[u];
  }
  int find(int y,int x){
    if(y <= x) return y;
    int tmp = find(pa[y],x);
    if(semi[best[y]] > semi[best[pa[y]]])
      best[y] = best[pa[y]];
    return pa[y] = tmp;
  }
  void tarjan(int root){
    dfnCnt = 0;
    for(int i=1; i<=n; ++i){
      dfn[i] = idom[i] = 0;
      tree[i].clear();
      best[i] = semi[i] = i;
    }
    dfs(root);
    for(int i=dfnCnt; i>1; --i){
      int u = id[i];
      for(auto v:rG[u]) if(v=dfn[v]){
        find(v,i);
        semi[i]=min(semi[i],semi[best[v]]);
      }
      tree[semi[i]].push_back(i);
      for(auto v:tree[pa[i]]){
        find(v, pa[i]);
        idom[v] = semi[best[v]]==pa[i] ? pa[i] : best[v];
      }
      tree[pa[i]].clear();
    }
    for(int i=2; i<=dfnCnt; ++i){
      if(idom[i] != semi[i])
        idom[i] = idom[idom[i]];
      tree[id[idom[i]]].push_back(id[i]);
    }
  }
}dom;
```

5.5 Floyd Warshall

```cpp
int d[100][100];
void FloydWarshall(int N){
    for(int k=0;k<N;++k)
        for(int i=0;i<N;++i)
            for(int j=0;j<N;++j)
                if(d[i][j] > d[i][k] + d[k][j])
                    d[i][j] = d[i][k] + d[k][j];
}
```

5.6 SCC

```cpp
struct SCC{
  int n,cnt = 0,dfn_cnt = 0;
  vector<vector<int>>g;
  vector<int>sz,scc,low,dfn;
  stack<int>st;
  vector<bool>vis;
  SCC(int _n = 0) : n(_n){
    sz.resize(n+5),scc.resize(n+5),low.resize(n+5),dfn.resize(n+5),vis.resize(n+5);
    g.resize(n+5);
  }
  inline void add_edge(int u, int v){
    g[u].push_back(v);
  }
  inline void build(){
    function<void(int, int)>dfs = [&](int u, int dis){
      low[u] = dfn[u] = ++dfn_cnt,vis[u] = 1;
      st.push(u);
      for(auto v:g[u]){
        if(!dfn[v]){
          dfs(v, dis+1);
          low[u] = min(low[u],low[v]);
        }
        else if(vis[v]){
          low[u] = min(low[u],dfn[v]);
        }
      }
      if(low[u]==dfn[u]){
        ++cnt;
        while(vis[u]){
          auto v = st.top();
          st.pop();
          vis[v] = 0;
          scc[v] = cnt;
          sz[cnt]++;
        }
      }
    };
    for(int i = 0;i<=n;++i){
      if(!scc[i]){
        dfs(i, 1);
      }
    }
  }
  vector<vector<int>> compress(){
    vector<vector<int>>ans(cnt+1);
    for(int u = 0;u<=n;++u){
      for(auto v:g[u]){
        if(scc[u] == scc[v]){
          continue;
        }
        ans[scc[u]].push_back(scc[v]);
      }
    }
    for(int i = 0;i<=cnt;++i){
      sort(ans[i].begin(), ans[i].end());
      ans[i].erase(unique(ans[i].begin(), ans[i].end()), ans[i].end());
    }
    return ans;
  }
};
```

5.7 SPFA

```
vector<long long> spfa(vector<vector<pair<int, int>>> G, int S) {
  int n = G.size(); // 假設點的編號為 0 ~ n-1
  vector<long long> d(n, INF);
  vector<bool> in_queue(n, false);
  vector<int> cnt(n, 0);
  queue<int> Q;
  d[S] = 0;
  auto enqueue = [&](int u) {
    in_queue[u] = true; Q.emplace(u);
  };
  enqueue(S);
  while (Q.size()) {
    int u = Q.front();
    Q.pop();
    in_queue[u] = false;
    for (auto [v, cost] : G[u])
      if (d[v] > d[u] + cost) {
        if (++cnt[u] >= n) return {}; // 存在負環
        d[v] = d[u] + cost;
        if (!in_queue[v]) enqueue(v);
      }
  }
  return d;
}
```

5.8 判斷二分圖

```

vector<int> G[MAXN];
int color[MAXN]; // -1: not colored, 0: black, 1: white
/* color the connected component where u is */
/* parameter col: the color u should be colored */
bool coloring(int u, int col) {
    if(color[u] != -1) {
        if(color[u] != col) return false;
        return true;
    }
    color[u] = col;
    for(int v : G[u])
        if(!coloring(v, col ^ 1))
            return false;
    return true;
}

//check if a graph is a bipartite graph

bool checkBipartiteG(int n) {
    for(int i = 1; i <= n; i++)
          color[i] = -1;
    for(int i = 1; i <= n; i++)
        if(color[i] == -1 &&
            !coloring(i, 0))
            return false;
    return true;
}
```

5.9 判斷平面圖

```
//做smoothing,把degree <= 2的點移除
//O(n^3)
using AdjacencyMatrixTy = vector<vector<bool>>;
AdjacencyMatrixTy smoothing(AdjacencyMatrixTy &G) {
  size_t N = G.size(), Change = 0;
  do {
    Change = 0;
    for(size_t u = 0; u < N; ++u) {
      vector<size_t> E;
      for(size_t v = 0; v < N && E.size() < 3; ++v)
        if(G[u][v] && u != v) E.emplace_back(v);
      if(E.size() == 1 || E.size() == 2) {
        ++Change;
        for(auto v : E) G[u][v] = G[v][u] = false;
      }
      if(E.size() == 2) {
        auto [a,b] = make_pair(E[0], E[1]);
        G[a][b] = G[b][a] = true;
      }
    }
  }
  while(Change);
  return G;
}

//計算Degree
//O(n^2)
vector<size_t> getDegree(const AdjacencyMatrixTy &G) {
  size_t N = G.size();
  vector<size_t> Degree(N);
  for(size_t u = 0; u < N; ++u)
    for(size_t v = u + 1; v < N; ++v) {
      if(!G[u][v]) continue;
      ++Degree[u], ++Degree[v];
    }
  return Degree;
}

//判斷是否為K5 or K33
//O(n)
bool is_K5_or_K33(const vector<size_t> &Degree) {
  unordered_map<size_t, size_t> Num;
  for(auto Val : Degree) ++Num[Val];
  size_t N = Degree.size();
  bool isK5 = Num[4] == 5 && Num[4] + Num[0] == N;
  bool isK33 = Num[3] == 6 && Num[3] + Num[0] == N;
  return isK5 || isK33;
}
```

5.10 判斷環

```

vector<int> G[MAXN];
bool visit[MAXN];
/* return if the connected component where u is contains a cycle*/
bool dfs(int u, int pre) {
    if(visit[u])    return true;
    visit[u] = true;
    for(int v : G[u])
        if(v != pre && dfs(v, u))
            return true;
    return false;
}

//check if a graph contains a cycle

bool checkCycle(int n) {
    for(int i = 1; i <= n; i++)
        if(!visit[i] && dfs(i, -1))
            return true;
    return false;
}
```

5.11 最大團

```
struct MaxClique{
  static const int MAXN=105;
  int N,ans;
  int g[MAXN][MAXN],dp[MAXN],stk[MAXN][MAXN];
  int sol[MAXN],tmp[MAXN];//sol[0~ans-1]為答案
  void init(int n){
    N=n;//0-base
    memset(g,0,sizeof(g));
  }
  void add_edge(int u,int v){
    g[u][v]=g[v][u]=1;
  }
  int dfs(int ns,int dep){
    if(!ns){
      if(dep>ans){
        ans=dep;
        memcpy(sol,tmp,sizeof tmp);
        return 1;
      }else return 0;
    }
    for(int i=0;i<ns;++i){
      if(dep+ns-i<=ans)return 0;
      int u=stk[dep][i],cnt=0;
      if(dep+dp[u]<=ans)return 0;
      for(int j=i+1;j<ns;++j){
        int v=stk[dep][j];
        if(g[u][v])stk[dep+1][cnt++]=v;
      }
      tmp[dep]=u;
      if(dfs(cnt,dep+1))return 1;
    }
    return 0;
  }
  int clique(){
    int u,v,ns;
    for(ans=0,u=N-1;u>=0;--u){
      for(ns=0,tmp[0]=u,v=u+1;v<N;++v)
        if(g[u][v])stk[1][ns++]=v;
      dfs(ns,1),dp[u]=ans;
    }
    return ans;
  }
};
```

5.12 枚舉極大團 Bron-Kerbosch

```cpp
//O(3^n / 3)
struct maximalCliques{
  using Set = vector<int>;
  size_t n; //1-base
  vector<Set> G;
  static Set setUnion(const Set &A, const Set &B){
    Set C(A.size() + B.size());
    auto it = set_union(A.begin(),A.end(),B.begin(),B.end(),C.begin());
    C.erase(it, C.end());
    return C;
  }
  static Set setIntersection(const Set &A, const Set &B){
    Set C(min(A.size(), B.size()));
    auto it = set_intersection(A.begin(),A.end(),B.begin(),B.end(),C.begin());
    C.erase(it, C.end());
    return C;
  }
  static Set setDifference(const Set &A, const Set &B){
    Set C(min(A.size(), B.size()));
    auto it = set_difference(A.begin(),A.end(),B.begin(),B.end(),C.begin());
    C.erase(it, C.end());
    return C;
  }
  void BronKerbosch1(Set R, Set P, Set X){
    if(P.empty()&&X.empty()){
      // R form an maximal clique
      return;
    }
    for(auto v: P){
      BronKerbosch1(setUnion(R,{v}), setIntersection(P,G[v]), setIntersection(X,G[v]));
      P = setDifference(P,{v});
      X = setUnion(X,{v});
    }
  }
  void init(int _n){
    G.clear();
    G.resize((n = _n) + 1);
  }
  void addEdge(int u, int v){
    G[u].emplace_back(v);
    G[v].emplace_back(u);
  }
  void solve(int n){
    Set P;
    for(int i=1; i<=n; ++i){
      sort(G[i].begin(), G[i].end());
G[i].erase(unique(G[i].begin(), G[i].end()), G[i].end());
      P.emplace_back(i);
    }
    BronKerbosch1({}, P, {});
  }
};

//判斷圖G是否能3塗色：
//枚舉圖G的極大獨立集I (極大獨立集 = 補圖極大團)
//若存在I使得G-I形成二分圖，則G可以三塗色
//反之則不能3塗色
```

5.13 橋連通分量

```cpp
vector<pii> findBridges(const vector<vector<int>>& g) {
  int n = (int) g.size();
  vector<int> id(n, -1), low(n);
  vector<pii> bridges;
  function<void(int, int)> dfs = [&](int u, int p) {
    static int cnt = 0;
    id[u] = low[u] = cnt++;
    for(auto v : g[u]) {
      if(v == p) continue;
      if(id[v] != -1) low[u] = min(low[u], id[v]);
      else {
        dfs(v, u);
        low[u] = min(low[u], low[v]);
        if(low[v] > id[u]) bridges.EB(u, v);
      }
    }
  };
  for(int i = 0; i < n; ++i) {
    if(id[i] == -1) dfs(i, -1);
  }
  return bridges;
}
```

5.14 雙連通分量＆割點

```cpp
struct BCC_AP{
  int dfn_cnt = 0,bcc_cnt = 0,n;
  vector<int>dfn,low,ap,bcc_id;
  stack<int>st;
  vector<bool>vis,is_ap;
  vector<vector<int>>bcc;
  BCC_AP(int _n):n(_n){
    dfn.resize(n+5),low.resize(n+5),bcc.resize(n+5),vis.resize(n+5),is_ap.resize(n+5),bcc_id.resize(n+5);
  }
  inline void build(const vector<vector<int>>&g,int u,int p = -1){
    int child = 0;
    dfn[u] = low[u] = ++dfn_cnt;
    st.push(u);
    vis[u] = 1;
    if(g[u].empty() and p==-1){
      bcc_id[u] = ++bcc_cnt;
      bcc[bcc_cnt].push_back(u);
      return;
    }
    for(auto v:g[u]){
      if(v==p)continue;
      if(!dfn[v]){
        build(g,v,u);
        child++;
        if(dfn[u]<=low[v]){
          is_ap[u] = 1;
          bcc_id[u] = ++bcc_cnt;
          bcc[bcc_cnt].push_back(u);
          while(vis[v]){
            bcc_id[st.top()] = bcc_cnt;
            bcc[bcc_cnt].push_back(st.top());
            vis[st.top()] = 0;
            st.pop();
          }
        }
        low[u] = min(low[u],low[v]);
      }
      low[u] = min(low[u],dfn[v]);
    }
    if(p==-1 and child<2)is_ap[u] = 0;
    if(is_ap[u])ap.push_back(u);
  }
};
```

6 Math

6.1 Basic

```cpp
template<typename T>
void gcd(const T &a,const T &b,T &d,T &x,T &y){
  if(!b) d=a,x=1,y=0;
  else gcd(b,a%b,d,y,x), y-=x*(a/b);
}
long long int phi[N+1];
void phiTable(){
  for(int i=1;i<=N;i++)phi[i]=i;
  for(int i=1;i<=N;i++)for(x=i*2;x<=N;x+=i)phi[x]-=phi[i];
}
void all_divdown(const LL &n) {// all n/x
  for(LL a=1;a<=n;a=n/(n/(a+1))){
    // dosomething;
  }
}
const int MAXPRIME = 1000000;
int iscom[MAXPRIME], prime[MAXPRIME], primecnt;
int phi[MAXPRIME], mu[MAXPRIME];
void sieve(void){
  memset(iscom,0,sizeof(iscom));
  primecnt = 0;
  phi[1] = mu[1] = 1;
  for(int i=2;i<MAXPRIME;++i) {
    if(!iscom[i]) {
      prime[primecnt++] = i;
      mu[i] = -1;
      phi[i] = i-1;
    }
    for(int j=0;j<primecnt;++j) {
      int k = i * prime[j];
      if(k>=MAXPRIME) break;
      iscom[k] = prime[j];
      if(i%prime[j]==0) {
        mu[k] = 0;
        phi[k] = phi[i] * prime[j];
        break;
      } else {
        mu[k] = -mu[i];
        phi[k] = phi[i] * (prime[j]-1);
      }
    }
  }
}

bool g_test(const LL &g, const LL &p, const vector<LL> &v) {
  for(int i=0;i<v.size();++i)
    if(modexp(g,(p-1)/v[i],p)==1)
      return false;
  return true;
}
LL primitive_root(const LL &p) {
  if(p==2) return 1;
  vector<LL> v;
  Factor(p-1,v);
  v.erase(unique(v.begin(), v.end()), v.end());
  for(LL g=2;g<p;++g)
    if(g_test(g,p,v))
      return g;
  puts("primitive_root NOT FOUND");
  return -1;
}
int Legendre(const LL &a, const LL &p) { return modexp(a%p,(p-1)/2,p); }

LL inv(const LL &a, const LL &n) {
  LL d,x,y;
  gcd(a,n,d,x,y);
  return d==1 ? (x+n)%n : -1;
}

int inv[maxN];
LL invtable(int n,LL P){
  inv[1]=1;
  for(int i=2;i<n;++i)
    inv[i]=(P-(P/i))*inv[P%i]%P;
}

LL log_mod(const LL &a, const LL &b, const LL &p) {
  // a ^ x = b ( mod p )
  int m=sqrt(p+.5), e=1;
  LL v=inv(modexp(a,m,p), p);
  map<LL,int> x;
  x[1]=0;
  for(int i=1;i<m;++i) {
    e = LLmul(e,a,p);
    if(!x.count(e)) x[e] = i;
  }
  for(int i=0;i<m;++i) {
    if(x.count(b)) return i*m + x[b];
    b = LLmul(b,v,p);
  }
  return -1;
}
```

```cpp

LL Tonelli_Shanks(const LL &n, const LL &p) {
  // x^2 = n ( mod p )
  if(n==0) return 0;
  if(Legendre(n,p)!=1) while(1) { puts("SQRT ROOT does not exist"); }
  int S = 0;
  LL Q = p-1;
  while( !(Q&1) ) { Q>>=1; ++S; }
  if(S==1) return modexp(n%p,(p+1)/4,p);
  LL z = 2;
  for(;Legendre(z,p)!=-1;++z)
  LL c = modexp(z,Q,p);
  LL R = modexp(n%p,(Q+1)/2,p), t = modexp(n%p,Q,p);
  int M = S;
  while(1) {
    if(t==1) return R;
    LL b = modexp(c,1L<<(M-i-1),p);
    R = LLmul(R,b,p);
    t = LLmul( LLmul(b,b,p), t, p);
    c = LLmul(b,b,p);
    M = i;
  }
  return -1;
}

template<typename T>
T Euler(T n){
  T ans=n;
  for(T i=2;i*i<=n;++i){
    if(n%i==0){
      ans=ans/i*(i-1);
      while(n%i==0)n/=i;
    }
  }
  if(n>1)ans=ans/n*(n-1);
  return ans;
}

//Chinese_remainder_theorem
template<typename T>
T pow_mod(T n,T k,T m){
  T ans=1;
  for(n=(n>=m?n%m:n);k;k>>=1){
    if(k&1)ans=ans*n%m;
    n=n*n%m;
  }
  return ans;
}
template<typename T>
T crt(vector<T> &m,vector<T> &a){
  T M=1,tM,ans=0;
  for(int i=0;i<(int)m.size();++i)M*=m[i];
  for(int i=0;i<(int)a.size();++i){
    tM=M/m[i];
    ans=(ans+(a[i]*tM%M)*pow_mod(tM,Euler(m[i])-1,m[i])%M)%M;
    /*如果m[i]是質數，Euler(m[i])-1=m[i]-2，就不用算Euler了*/
  }
  return ans;
}
```

6.2 Bit Set

```cpp
void sub_set(int S){
  int sub=S;
  do{
    //對某集合的子集合的處理
    sub=(sub-1)&S;
  }while(sub!=S);
}
void k_sub_set(int k,int n){
  int comb=(1<<k)-1,S=1<<n;
  while(comb<S){
    //對大小為k的子集合的處理
    int x=comb&-comb,y=comb+x;
    comb=((comb&~y)/x>>1)|y;
  }
}
```

6.3 ExtendGCD

```cpp
// ax + by = gcd(a, b)
ll ext_gcd(ll a, ll b, ll& x, ll& y) {
  if(b == 0) {
    x = 1, y = 0;
    return a;
  }
  ll x1, y1;
  ll g = ext_gcd(b, a % b, x1, y1);
  x = y1, y = x1 - (a / b) * y1;
  return g;
}
```

6.4 FastPow

```cpp
ll modexp(ll x, ll k, ll p) {
    ll ans = 1;
    for(int i = 1; i <= k; i <<= 1) {
        if(i & k) ans *= x, ans %= p;
        x *= x, x %= p;
    }
    return ans;
}
```

6.5 FFT

```cpp
// Fast-Fourier-Transform
using cd = complex<double>;
const double PI = acos(-1);

void FFT(vector<cd>& a, bool inv) {
  int n = (int) a.size();
  for(int i = 1, j = 0; i < n; ++i) {
    int bit = n >> 1;
    for(; j & bit; bit >>= 1) {
      j ^= bit;
    }
    j ^= bit;
    if(i < j) {
      swap(a[i], a[j]);
    }
  }
  for(int len = 2; len <= n; len <<= 1) {
    const double ang = 2 * PI / len * (inv ? -1 : +1);
    cd rot(cos(ang), sin(ang));
    for(int i = 0; i < n; i += len) {
      cd w(1);
      for(int j = 0; j < len / 2; ++j) {
        cd u = a[i + j], v = a[i + j + len / 2] * w;
        a[i + j] = u + v;
        a[i + j + len / 2] = u - v;
        w *= rot;
      }
    }
  }
  if(inv) {
    for(auto& x : a) {
      x /= n;
    }
  }
}

vector<int> multiply(const vector<int>& a, const vector<int>& b) {
  vector<cd> fa(a.begin(), a.end());
  vector<cd> fb(b.begin(), b.end());
  int n = 1;
  while(n < (int) a.size() + (int) b.size() - 1) {
    n <<= 1;
  }
  fa.resize(n);
  fb.resize(n);
  FFT(fa, false);
  FFT(fb, false);
  for(int i = 0; i < n; ++i) {
    fa[i] *= fb[i];
  }
  FFT(fa, true);
  vector<int> c(a.size() + b.size() - 1);
  for(int i = 0; i < (int) c.size(); ++i) {
    c[i] = round(fa[i].real());
  }
  return c;
}
```

6.6 FWT

```cpp
vector<int> F_OR_T(vector<int> f, bool inverse){
  for(int i=0; (2<<i)<=f.size(); ++i)
    for(int j=0; j<f.size(); j+=2<<i)
      for(int k=0; k<(1<<i); ++k)
        f[j+k+(1<<i)] += f[j+k]*(inverse?-1:1);
  return f;
}
vector<int> rev(vector<int> A) {
  for(int i=0; i<A.size(); i+=2)
    swap(A[i],A[i^(A.size()-1)]);
  return A;
}
vector<int> F_AND_T(vector<int> f, bool inverse){
  return rev(F_OR_T(rev(f), inverse));
}
vector<int> F_XOR_T(vector<int> f, bool inverse){
  for(int i=0; (2<<i)<=f.size(); ++i)
    for(int j=0; j<f.size(); j+=2<<i)
      for(int k=0; k<(1<<i); ++k){
        int u=f[j+k], v=f[j+k+(1<<i)];
        f[j+k+(1<<i)] = u-v, f[j+k] = u+v;
      }
  if(inverse) for(auto &a:f) a/=f.size();
  return f;
}
```

6.7 Gauss-Jordan

```cpp
int GaussJordan(vector<vector<ld>>& a) {
  // -1 no sol, 0 inf sol
  int n = SZ(a);
  REP(i, n) assert(SZ(a[i]) == n + 1);
  REP(i, n) {
    int p = i;
    REP(j, n) {
      if(j < i && abs(a[j][j]) > EPS) continue;
      if(abs(a[j][i]) > abs(a[p][i])) p = j;
    }
    REP(j, n + 1) swap(a[i][j], a[p][j]);
    if(abs(a[i][i]) <= EPS) continue;
    REP(j, n) {
      if(i == j) continue;
      ld delta = a[j][i] / a[i][i];
      FOR(k, i, n + 1) a[j][k] -= delta * a[i][k];
    }
  }
  bool ok = true;
  REP(i, n) {
    if(abs(a[i][i]) <= EPS) {
      if(abs(a[i][n]) > EPS) return -1;
      ok = false;
    }
  }
  return ok;
}
```

6.8 InvGCD

```cpp
pair<long long, long long> inv_gcd(long long a, long long b) {
  a %= b;
  if(a < 0) a += b;
  if(a == 0) return {b, 0};
  long long s = b, t = a;
  long long m0 = 0, m1 = 1;
  while(t) {
    long long u = s / t;
    s -= t * u;
    m0 -= m1 * u;
    swap(s, t);
    swap(m0, m1);
  }
  if(m0 < 0) m0 += b / s;
  return {s, m0};
}
```

6.9 LinearCongruence

```cpp
pair<LL,LL> LinearCongruence(LL a[],LL b[],LL m[],int n) {
  // a[i]*x = b[i] ( mod m[i] )
  for(int i=0;i<n;++i) {
    LL x, y, d = extgcd(a[i],m[i],x,y);
    if(b[i]%d!=0) return make_pair(-1LL,0LL);
    m[i] /= d;
    b[i] = LLmul(b[i]/d,x,m[i]);
  }
  LL lastb = b[0], lastm = m[0];
  for(int i=1;i<n;++i) {
    LL x, y, d = extgcd(m[i],lastm,x,y);
    if((lastb-b[i])%d!=0) return make_pair(-1LL,0LL);
    lastb = LLmul((lastb-b[i])/d,x,(lastm/d))*m[i];
    lastm = (lastm/d)*m[i];
    lastb = (lastb+b[i])%lastm;
  }
  return make_pair(lastb<0?lastb+lastm:lastb,lastm);
}
```

6.10 LinearSieve

```cpp
vector<bool> is_prime;
vector<int> primes, phi, mobius, least;
void linear_sieve(int n) {
  n += 1;
  is_prime.resize(n);
  least.resize(n);
  fill(2 + begin(is_prime),end(is_prime),true);
  phi.resize(n); mobius.resize(n);
  phi[1] = mobius[1] = 1;
  least[0] = 0,least[1] = 1;
  for(int i = 2; i < n; ++i) {
    if(is_prime[i]) {
      primes.push_back(i);
      phi[i] = i - 1;
      mobius[i] = -1;
      least[i] = i;
    }
    for(auto j : primes) {
      if(i * j >= n) break;
      is_prime[i * j] = false;
      least[i * j] = j;
      if(i % j == 0) {
        mobius[i * j] = 0;
        phi[i * j] = phi[i] * j;
        break;
      } else {
        mobius[i * j] = mobius[i] * mobius[j];
        phi[i * j] = phi[i] * phi[j];
      }
    }
  }
}
```

6.11 Lucas

```cpp
ll C(ll n, ll m, ll p){// n!/m!/(n-m)!
  if(n<m) return 0;
  return f[n]*inv(f[m],p)%p*inv(f[n-m],p)%p;
}
ll L(ll n, ll m, ll p){
  if(!m) return 1;
  return C(n%p,m%p,p)*L(n/p,m/p,p)%p;
}
ll Wilson(ll n, ll p){ // n!%p
  if(!n)return 1;
  ll res=Wilson(n/p, p);
  if((n/p)%2) return res*(p-f[n%p])%p;
  return  res*f[n%p]%p; //(p-1)!%p=-1
}
```

6.12 Matrix

```cpp
template<typename T>
struct Matrix{
  using rt = std::vector<T>;
  using mt = std::vector<rt>;
  using matrix = Matrix<T>;
  int r,c;
  mt m;
  Matrix(int r,int c):r(r),c(c),m(r,rt(c)){}
  rt& operator[](int i){return m[i];}
  matrix operator+(const matrix &a){
    matrix rev(r,c);
    for(int i=0;i<r;++i)
      for(int j=0;j<c;++j)
        rev[i][j]=m[i][j]+a.m[i][j];
    return rev;
  }
  matrix operator-(const matrix &a){
    matrix rev(r,c);
    for(int i=0;i<r;++i)
      for(int j=0;j<c;++j)
        rev[i][j]=m[i][j]-a.m[i][j];
    return rev;
  }
  matrix operator*(const matrix &a){
    matrix rev(r,a.c);
    matrix tmp(a.c,a.r);
    for(int i=0;i<a.r;++i)
      for(int j=0;j<a.c;++j)
        tmp[j][i]=a.m[i][j];
    for(int i=0;i<r;++i)
      for(int j=0;j<a.c;++j)
        for(int k=0;k<c;++k)
          rev.m[i][j]+=m[i][k]*tmp[j][k];
    return rev;
  }
  bool inverse(){
    Matrix t(r,r+c);
    for(int y=0;y<r;y++){
      t.m[y][c+y] = 1;
      for(int x=0;x<c;++x)
        t.m[y][x]=m[y][x];
    }
    if( !t.gas() )
      return false;
    for(int y=0;y<r;y++)
      for(int x=0;x<c;++x)
        m[y][x]=t.m[y][c+x]/t.m[y][y];
    return true;
  }
  T gas(){
    vector<T> lazy(r,1);
    bool sign=false;
    for(int i=0;i<r;++i){
      if( m[i][i]==0 ){
        int j=i+1;
        while(j<r&&!m[j][i])j++;
        if(j==r)continue;
        m[i].swap(m[j]);
        sign=!sign;
      }
      for(int j=0;j<r;++j){
        if(i==j)continue;
        lazy[j]=lazy[j]*m[i][i];
        T mx=m[j][i];
        for(int k=0;k<c;++k)
          m[j][k]=m[j][k]*m[i][i]-m[i][k]*mx;
      }
    }
    T det=sign?-1:1;
    for(int i=0;i<r;++i){
      det = det*m[i][i];
      det = det/lazy[i];
      for(auto &j:m[i])j/=lazy[i];
    }
    return det;
  }
};
```

6.13 Miller-Rabin

```cpp
bool is_prime(ll n, vector<ll> x) {
  ll d = n - 1;
  d >>= __builtin_ctzll(d);
  for(auto a : x) {
    if(n <= a) break;
    ll t = d, y = 1, b = t;
    while(b) {
      if(b & 1) y = i128(y) * a % n;
      a = i128(a) * a % n;
      b >>= 1;
    }
    while(t != n - 1 && y != 1 && y != n - 1) {
      y = i128(y) * y % n;
      t <<= 1;
    }
    if(y != n - 1 && t % 2 == 0) return 0;
  }
  return 1;
}
bool is_prime(ll n) {
  if(n <= 1) return 0;
  if(n % 2 == 0) return n == 2;
  if(n < (1LL << 30)) return is_prime(n, {2, 7, 61});
  return is_prime(n, {2, 325, 9375, 28178, 450775, 9780504, 1795265022});
}
```

6.14 Numbers

- Bernoulli numbers

  $B_0 - 1, B_1^{\pm} = \pm\frac{1}{2}, B_2 = \frac{1}{6}, B_3 = 0$

  $\sum_{j=0}^{m} \binom{m+1}{j} B_j = 0$, EGF is $B(x) = \frac{x}{e^x - 1} = \sum_{n=0}^{\infty} B_n \frac{x^n}{n!}$.

  $S_m(n) = \sum_{k=1}^{n} k^m = \frac{1}{m+1} \sum_{k=0}^{m} \binom{m+1}{k} B_k^{+} n^{m+1-k}$

- Stirling numbers of the second kind Partitions of n distinct elements into exactly k groups.

  $S(n,k) = S(n-1,k-1) + kS(n-1,k), S(n,1) = S(n,n) = 1$

  $S(n,k) = \frac{1}{k!} \sum_{i=0}^{k} (-1)^{k-i} \binom{k}{i} i^n$

  $x^n = \sum_{i=0}^{n} S(n,i)(x)_i$

- Pentagonal number theorem

  $\prod_{n=1}^{\infty} (1 - x^n) = 1 + \sum_{k=1}^{\infty} (-1)^k \left(x^{k(3k+1)/2} + x^{k(3k-1)/2}\right)$

- Catalan numbers

  $C_n^{(k)} = \frac{1}{(k-1)n+1} \binom{kn}{n}$

  $C^{(k)}(x) = 1 + x[C^{(k)}(x)]^k$

- Eulerian numbers

  Number of permutations $\pi \in S_n$ in which exactly k elements are greater than the previous element. k j:s s.t. $\pi(j) > \pi(j+1)$, $k+1$ j:s s.t. $\pi(j) \geq j$, k j:s s.t. $\pi(j) > j$.

  $E(n,k) = (n-k)E(n-1,k-1) + (k+1)E(n-1,k)$

  $E(n,0) = E(n,n-1) = 1$

  $E(n,k) = \sum_{j=0}^{k}(-1)^j\binom{n+1}{j}(k+1-j)^n$

6.15 Pollard-Rho

```cpp
void PollardRho(map<ll, int>& mp, ll n) {
  if(n == 1) return;
  if(is_prime(n)) return mp[n]++, void();
  if(n % 2 == 0) {
    mp[2] += 1;
    PollardRho(mp, n / 2);
    return;
  }
  ll x = 2, y = 2, d = 1, p = 1;
  #define f(x, n, p) ((i128(x) * x % n + p) % n)
  while(1) {
    if(d != 1 && d != n) {
      PollardRho(mp, d);
      PollardRho(mp, n / d);
      return;
    }
    p += (d == n);
    x = f(x, n, p), y = f(f(y, n, p), n, p);
    d = __gcd(abs(x - y), n);
  }
  #undef f
}

vector<ll> get_divisors(ll n) {
  if(n == 0) return {};
  map<ll, int> mp;
  PollardRho(mp, n);
  vector<pair<ll, int>> v(ALL(mp));
  vector<ll> res;
  auto f = [&](auto f, int i, ll x) -> void {
    if(i == SZ(v)) {
      res.pb(x);
      return;
    }
    for(int j = v[i].second; ; j--) {
      f(f, i + 1, x);
      if(j == 0) break;
      x *= v[i].first;
    }
  };
  f(f, 0, 1);
  sort(ALL(res));
  return res;
}
```

6.16 Theorem

- Modular Arithmetic

$$(a+b) \bmod m = (a \bmod m + b \bmod m) \bmod m$$

$$(a-b) \bmod m = (a \bmod m - b \bmod m) \bmod m$$

$$(a \cdot b) \pmod m = ((a \bmod m) \cdot (b \bmod m)) \bmod m$$

$$a^b \bmod m = (a \bmod m)^{b \bmod m-1} \bmod m$$

- Cramer's rule

$$\begin{matrix} ax+by=e \\ cx+dy=f \end{matrix} \Rightarrow \begin{matrix} x = \dfrac{ed-bf}{ad-bc} \\ y = \dfrac{af-ec}{ad-bc} \end{matrix}$$

- Kirchhoff's Theorem

  Denote L be a $n \times n$ matrix as the Laplacian matrix of graph G, where $L_{ii} = d(i)$, $L_{ij} = -c$ where c is the number of edge (i,j) in G.

  - The number of undirected spanning in G is $|\det(\tilde{L}_{11})|$.
  - The number of directed spanning tree rooted at r in G is $|\det(\tilde{L}_{rr})|$.

- Tutte's Matrix

  Let D be a $n \times n$ matrix, where $d_{ij} = x_{ij}$ (x_{ij} is chosen uniformly at random) if $i < j$ and $(i,j) \in E$, otherwise $d_{ij} = -d_{ji}$. $\frac{rank(D)}{2}$ is the maximum matching on G.

- Cayley's Formula

  - Given a degree sequence $d_1, d_2, \ldots, d_n$ for each labeled vertices, there are $\frac{(n-2)!}{(d_1-1)!(d_2-1)!\cdots(d_n-1)!}$ spanning trees.
  - Let $T_{n,k}$ be the number of labeled forests on n vertices with k components, such that vertex $1, 2, \ldots, k$ belong to different components. Then $T_{n,k} = kn^{n-k-1}$.

- Erd□s–Gallai theorem

  A sequence of nonnegative integers $d_1 \geq \cdots \geq d_n$ can be represented as the degree sequence of a finite simple graph on n vertices if and only if $d_1 + \cdots + d_n$ is even and $\displaystyle\sum_{i-1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min(d_i, k)$ holds for every $1 \leq k \leq n$.

- Gale–Ryser theorem

  A pair of sequences of nonnegative integers $a_1 \geq \cdots \geq a_n$ and $b_1, \ldots, b_n$ is bigraphic if and only if $\displaystyle\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} b_i$ and $\displaystyle\sum_{i=1}^{k} a_i \leq \sum_{i=1}^{n} \min(b_i, k)$ holds for every $1 \leq k \leq n$.

- Fulkerson–Chen–Anstee theorem

  A sequence $(a_1, b_1), \ldots, (a_n, b_n)$ of nonnegative integer pairs with $a_1 \geq \cdots \geq a_n$ is digraphic if and only if $\displaystyle\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} b_i$ and $\displaystyle\sum_{i=1}^{k} a_i \leq \sum_{i=1}^{k} \min(b_i, k-1) + \sum_{i=k+1}^{n} \min(b_i, k)$ holds for every $1 \leq k \leq n$.

- M□bius inversion formula

  - $f(n) = \sum_{d|n} g(d) \Leftrightarrow g(n) = \sum_{d|n} \mu(d) f(\frac{n}{d})$
  - $f(n) = \sum_{n|d} g(d) \Leftrightarrow g(n) = \sum_{n|d} \mu(\frac{d}{n}) f(d)$

- Spherical cap

  - A portion of a sphere cut off by a plane.
  - r: sphere radius, a: radius of the base of the cap, h: height of the cap, θ: $\arcsin(a/r)$.
  - Volume $= \pi h^2(3r-h)/3 = \pi h(3a^2+h^2)/6 = \pi r^3(2+\cos\theta)(1-\cos\theta)^2/3$.
  - Area $= 2\pi rh = \pi(a^2+h^2) = 2\pi r^2(1-\cos\theta)$.

6.17 找實根

```cpp
// an*x^n + ... + a1x + a0 = 0;
int sign(double x){
  return x < -eps ? -1 : x > eps;
}

double get(const vector<double>&coef, double x){
  double e = 1, s = 0;
  for(auto i : coef) s += i*e, e *= x;
  return s;
}

double find(const vector<double>&coef, int n, double lo, double hi){
  double sign_lo, sign_hi;
  if( !(sign_lo = sign(get(coef,lo))) ) return lo;
  if( !(sign_hi = sign(get(coef,hi))) ) return hi;
  if(sign_lo * sign_hi > 0) return INF;
  for(int stp = 0; stp < 100 && hi - lo > eps; ++stp){
    double m = (lo+hi)/2.0;
    int sign_mid = sign(get(coef,m));
    if(!sign_mid) return m;
    if(sign_lo*sign_mid < 0) hi = m;
    else lo = m;
  }
  return (lo+hi)/2.0;
}

vector<double> cal(vector<double>coef, int n){
  vector<double>res;
  if(n == 1){
    if(sign(coef[1])) res.pb(-coef[0]/coef[1]);
    return res;
  }
  vector<double>dcoef(n);
  for(int i = 0; i < n; ++i) dcoef[i] = coef[i+1]*(i+1);
  vector<double>droot = cal(dcoef, n-1);
  droot.insert(droot.begin(), -INF);
  droot.pb(INF);
  for(int i = 0; i+1 < droot.size(); ++i){
    double tmp = find(coef, n, droot[i], droot[i+1]);
    if(tmp < INF) res.pb(tmp);
  }
  return res;
}

int main () {
  vector<double>ve;
  vector<double>ans = cal(ve, n);
  // 視情況把答案 +eps，避免 -0
}
```

6.18 質因數分解

```cpp
//CSES Counting Divisors
#include<bits/stdc++.h>
using namespace std;

int n;

vector<int> primes;
vector<int> LPs;

void sieve(int n) {
    LPs.assign(n+1,1);
    for(int i=2;i<n;i++) {
        if(LPs[i]==1) {
            primes.emplace_back(i);
            LPs[i] = i;
        }
        for(auto p:primes) {
            if(1LL*i*p > n) break;
            LPs[i*p] = p;
            if(i%p==0) break;
        }
    }
}

signed main() {
    cin>>n;
    sieve((int)1e6);
    map<int,int> divisor;
    while(n--) {
        divisor.clear();
        int x;cin>>x;
        while(x>1) {
          divisor[LPs[x]]++;
            x/=LPs[x];
        }
        int ans = 1;
      for(auto &[x,y] : divisor) ans *= (y+1);
        cout<<ans;
        cout<<'\n';
    }
}
```

7 Square root decomposition

7.1 MoAlgo

```
struct qry{
  int ql,qr,id;
};
template<class T>struct Mo{
  int n,m;
  vector<pii>ans;
  Mo(int _n,int _m): n(_n),m(_m){
    ans.resize(m);
  }
  void solve(vector<T>&v,vector<qry>&q){
    int l = 0,r = -1;
    vector<int>cnt,cntcnt;
    cnt.resize(n+5);
    cntcnt.resize(n+5);
    int mx = 0;
    function<void(int)>add = [&](int pos){
      cntcnt[cnt[v[pos]]]--;
      cnt[v[pos]]++;
      cntcnt[cnt[v[pos]]]++;
      mx = max(mx,cnt[v[pos]]);
    };
    function<void(int)>sub = [&](int pos){
      if(!--cntcnt[cnt[v[pos]]] and cnt[v[pos]]==mx)mx--;
      cnt[v[pos]]--;
      cntcnt[cnt[v[pos]]]++;
      mx = max(mx,cnt[v[pos]]);
    };
    sort(all(q),[&](qry a,qry b){
      static int B = max((int)1,n/max((int)sqrt(m),(int)1));
      if(a.ql/B!=b.ql/B)return a.ql<b.ql;
      if((a.ql/B)&1)return a.qr>b.qr;
      return a.qr<b.qr;
    });
    for(auto [ql,qr,id]:q){
      while(l>ql)add(--l);
      while(r<qr)add(++r);
      while(l<ql)sub(l++);
      while(r>qr)sub(r--);
      ans[id] = {mx,cntcnt[mx]};
    }
  }
};
```

7.2 分塊 cf455D

```
const ll block_siz = 320;
const ll maxn = 100005;
ll a[maxn];
ll cnt[block_siz+1][maxn]; // i-th block, k's cou
deque<ll> q[block_siz+1];

void print_all(ll n)
{
    for(int i=0;i<n;i++)
    {
        cout << q[i/block_siz][i-i/block_siz*block_siz] << ' ';
    }
    cout << endl << endl;
}

int main()
{   Crbubble
    ll n,m,i,k,t;
    ll l,r,ord,pre,id,id2, ans = 0;
    cin >> n;
    for(i=0;i<n;i++)
    {
        cin >> a[i];
        id = i/block_siz;
        q[id].push_back(a[i]);
        cnt[id][a[i]]++;
    }
    cin >> t;
    while(t--)
    {
        cin >> ord >> l >> r;
        l = (l+ans-1)%n+1 -1;
        r = (r+ans-1)%n+1 -1;
        if(l > r) swap(l,r);
        id = l/block_siz; l %= block_siz;
        id2 = r/block_siz; r %= block_siz;
        if(ord == 1)
        {
            if(id == id2)
            {
                pre = q[id][r];
                for(i=r;i>l;i--)
                {
                    q[id][i] = q[id][i-1];
                }
                q[id][l] = pre;
            }
            else
            {
                pre = q[id].back();
                cnt[id][pre]--;
                q[id].pop_back();

                for(i=id+1;i<id2;i++)
                {
                    q[i].push_front(pre);
                    cnt[i][pre]++;
                    pre = q[i].back();
                    cnt[i][pre]--;
                    q[i].pop_back();
                }
                q[id2].push_front(pre);
                cnt[id2][pre]++;
                pre = q[id2][r+1];
                cnt[id2][pre]--;
                q[id2].erase(q[id2].begin()+r+1);

                q[id].insert(q[id].begin()+l, pre);
                cnt[id][pre]++;
            }
            //print_all(n);
        }
        else
        {   // query m cnt
            cin >> m;
            m = (m+ans-1)%n+1;
            ans = 0;
            if(id == id2)
            {
                for(i=l;i<=r;i++) ans += (q[id][i] == m);
            }
            else
            {
                for(i=l;i<block_siz;i++) ans += (q[id][i] == m);
                for(i=0;i<=r;i++) ans += (q[id2][i] == m);
                for(i=id+1;i<id2;i++) ans += cnt[i][m];
            }
            cout << ans << endl;
        }
    }
    return 0;
}
```

7.3 莫隊

```
void remove(idx);  // TODO: remove value at idx from data structure
void add(idx);     // TODO: add value at idx from data structure
int get_answer();  // TODO: extract the current answer of the data structure

int block_size;

struct Query {
    int l, r, idx;
    bool operator<(Query other) const
    {
        return make_pair(l / block_size, r) <
               make_pair(other.l / block_size, other.r);
    }
};

vector<int> mo_s_algorithm(vector<Query> queries) {
    vector<int> answers(queries.size());
    sort(queries.begin(), queries.end());

    // TODO: initialize data structure

    int cur_l = 0;
    int cur_r = -1;
    // invariant: data structure will always reflect the range [cur_l, cur_r]
    for (Query q : queries) {
        while (cur_l > q.l) {
            cur_l--;
            add(cur_l);
        }
        while (cur_r < q.r) {
            cur_r++;
            add(cur_r);
        }
        while (cur_l < q.l) {
            remove(cur_l);
            cur_l++;
        }
        while (cur_r > q.r) {
            remove(cur_r);
            cur_r--;
        }
        answers[q.idx] = get_answer();
    }
    return answers;
}
```

8 Tree

8.1 centroidDecomposition

```
vector<vector<int>>g;
vector<int>sz,tmp;
vector<bool>vis;//visit_centroid
int tree_centroid(int u,int n){
  function<void(int,int)>dfs1 = [&](int u,int p){
    sz[u] = 1;
    for(auto v:g[u]){
      if(v==p)continue;
      if(vis[v])continue;
      dfs1(v,u);
      sz[u]+=sz[v];
    }
  };
  function<int(int,int)>dfs2 = [&](int u,int p){
    for(auto v:g[u]){
      if(v==p)continue;
      if(vis[v])continue;
      if(sz[v]*2<n)continue;
      return dfs2(v,u);
    }
    return u;
  };
  dfs1(u,-1);
  return dfs2(u,-1);
}
int cal(int u,int p = -1,int deep = 1){
  int ans = 0;
  tmp.pb(deep);
  sz[u] = 1;
  for(auto v:g[u]){
    if(v==p)continue;
    if(vis[v])continue;
    ans+=cal(v,u,deep+1);
    sz[u]+=sz[v];
  }
  //calcuate the answer
  return ans;
}
int centroid_decomposition(int u,int tree_size){
  int center = tree_centroid(u,tree_size);
  vis[center] = 1;
  int ans = 0;
  for(auto v:g[center]){
    if(vis[v])continue;
    ans+=cal(v);
    for(int i = sz(tmp)-sz[v];i<sz(tmp);++i){
      //update
    }
  }
  while(!tmp.empty()){
    //roll_back(tmp.back())
    tmp.pop_back();
  }
  for(auto v:g[center]){
    if(vis[v])continue;
    ans+=centroid_decomposition(v,sz[v]);
  }
  return ans;
}
```

8.2 HeavyLight

```
#include<vector>
#define MAXN 100005
int siz[MAXN],max_son[MAXN],pa[MAXN],dep[MAXN];
int link_top[MAXN],link[MAXN],cnt;
vector<int> G[MAXN];
void find_max_son(int u){
  siz[u]=1;
  max_son[u]=-1;
  for(auto v:G[u]){
    if(v==pa[u])continue;
    pa[v]=u;
    dep[v]=dep[u]+1;
    find_max_son(v);
    if(max_son[u]==-1||siz[v]>siz[max_son[u]])max_son[u]=v;
    siz[u]+=siz[v];
  }
}
void build_link(int u,int top){
  link[u]=++cnt;
  link_top[u]=top;
  if(max_son[u]==-1)return;
  build_link(max_son[u],top);
  for(auto v:G[u]){
    if(v==max_son[u]||v==pa[u])continue;
    build_link(v,v);
  }
}
int find_lca(int a,int b){
  //求LCA，可以在過程中對區間進行處理
  int ta=link_top[a],tb=link_top[b];
  while(ta!=tb){
    if(dep[ta]<dep[tb]){
      swap(ta,tb);
      swap(a,b);
    }
    //這裡可以對a所在的鏈做區間處理
    //區間為(link[ta],link[a])
    ta=link_top[a=pa[ta]];
  }
  //最後a,b會在同一條鏈，若a!=b還要在進行一次區間處理
  return dep[a]<dep[b]?a:b;
}
```

8.3 HLD

```
struct heavy_light_decomposition{
  int n;
  vector<int>dep,father,sz,mxson,topf,id;
  vector<vector<int>>g;
  heavy_light_decomposition(int _n = 0) : n(_n) {
    g.resize(n+5);
    dep.resize(n+5);
    father.resize(n+5);
    sz.resize(n+5);
    mxson.resize(n+5);
    topf.resize(n+5);
    id.resize(n+5);
  }
  void add_edge(int u, int v){
    g[u].push_back(v);
    g[v].push_back(u);
  }
  void dfs(int u,int p){
    dep[u] = dep[p]+1;
    father[u] = p;
    sz[u] = 1;
    mxson[u] = 0;
    for(auto v:g[u]){
      if(v==p)continue;
      dfs(v,u);
      sz[u]+=sz[v];
      if(sz[v]>sz[mxson[u]])mxson[u] = v;
    }
  }
  void dfs2(int u,int top){
    static int idn = 0;
    topf[u] = top;
    id[u] = ++idn;
    if(mxson[u])dfs2(mxson[u],top);
    for(auto v:g[u]){
      if(v!=father[u] and v!=mxson[u]){
        dfs2(v,v);
      }
    }
  }
  void build(int root){
    dfs(root,0);
    dfs2(root,root);
  }
  vector<pair<int, int>> path(int u,int v){
    vector<pair<int, int>>ans;
    while(topf[u]!=topf[v]){
      if(dep[topf[u]]<dep[topf[v]])swap(u,v);
      ans.push_back({id[topf[u]], id[u]});
      u = father[topf[u]];
    }
    if(id[u]>id[v])swap(u,v);
    ans.push_back({id[u], id[v]});
    return ans;
  }
};
```

8.4 LCA

```
const int MAXN=200000; // 1-base
const int MLG=__lg(MAXN) + 1; //log2(MAXN)+1;
int pa[MLG+2][MAXN+5];
int dep[MAXN+5];
vector<int> G[MAXN+5];
void dfs(int x,int p=0){//dfs(root);
  pa[0][x]=p;
  for(int i=0;i<=MLG;++i)
    pa[i+1][x]=pa[i][pa[i][x]];
  for(auto &i:G[x]){
    if(i==p)continue;
    dep[i]=dep[x]+1;
    dfs(i,x);
  }
}
inline int jump(int x,int d){
  for(int i=0;i<=MLG;++i)
    if((d>>i)&1) x=pa[i][x];
  return x;
}
inline int find_lca(int a,int b){
  if(dep[a]>dep[b])swap(a,b);
  b=jump(b,dep[b]-dep[a]);
  if(a==b)return a;
  for(int i=MLG;i>=0;--i){
    if(pa[i][a]!=pa[i][b]){
      a=pa[i][a];
      b=pa[i][b];
    }
  }
  return pa[0][a];
}

//用樹壓平做
#define MAXN 100000
typedef vector<int >::iterator VIT;
int dep[MAXN+5],in[MAXN+5];
int vs[2*MAXN+5];
int cnt;/*時間戳*/
vector<int >G[MAXN+5];
void dfs(int x,int pa){
  in[x]=++cnt;
  vs[cnt]=x;
  for(VIT i=G[x].begin();i!=G[x].end();++i){
    if(*i==pa)continue;
    dep[*i]=dep[x]+1;
    dfs(*i,x);
    vs[++cnt]=x;
  }
}

inline int find_lca(int a,int b){
  if(in[a]>in[b])swap(a,b);
  return RMQ(in[a],in[b]);
}
```

8.5 link cut tree

```
struct splay_tree{
  int ch[2],pa;//子節點跟父母
  bool rev;//反轉的懶惰標記
  splay_tree():pa(0),rev(0){ch[0]=ch[1]=0;}
};
vector<splay_tree> nd;
//有的時候用vector會TLE，要注意
//這邊以node[0]作為null節點
bool isroot(int x){//判斷是否為這棵splay tree的根
  return nd[nd[x].pa].ch[0]!=x&&nd[nd[x].pa].ch[1]!=x;
}
void down(int x){//懶惰標記下推
  if(nd[x].rev){
    if(nd[x].ch[0])nd[nd[x].ch[0]].rev^=1;
    if(nd[x].ch[1])nd[nd[x].ch[1]].rev^=1;
    swap(nd[x].ch[0],nd[x].ch[1]);
    nd[x].rev=0;
  }
}
void push_down(int x){//所有祖先懶惰標記下推
  if(!isroot(x))push_down(nd[x].pa);
  down(x);
}
void up(int x){}//將子節點的資訊向上更新
void rotate(int x){//旋轉，會自行判斷轉的方向
  int y=nd[x].pa,z=nd[y].pa,d=(nd[y].ch[1]==x);
  nd[x].pa=z;
  if(!isroot(y))nd[z].ch[nd[z].ch[1]==y]=x;
  nd[y].ch[d]=nd[x].ch[d^1];
  nd[nd[y].ch[d]].pa=y;
  nd[y].pa=x,nd[x].ch[d^1]=y;
  up(y),up(x);
```

```cpp
}
void splay(int x){//將x伸展到splay tree的根
  push_down(x);
  while(!isroot(x)){
    int y=nd[x].pa;
    if(!isroot(y)){
      int z=nd[y].pa;
      if((nd[z].ch[0]==y)^(nd[y].ch[0]==x)) rotate(y);
      else rotate(x);
    }
    rotate(x);
  }
}
int access(int x){
  int last=0;
  while(x){
    splay(x);
    nd[x].ch[1]=last;
    up(x);
    last=x;
    x=nd[x].pa;
  }
  return last;//access後splay tree的根
}
void access(int x,bool is=0){//is=0就是一般的access
  int last=0;
  while(x){
    splay(x);
    if(is&&!nd[x].pa){
      //printf("%d\n",max(nd[last].ma,nd[nd[x].ch[1]].ma));
    }
    nd[x].ch[1]=last;
    up(x);
    last=x;
    x=nd[x].pa;
  }
}
void query_edge(int u,int v){
  access(u);
  access(v,1);
}
void make_root(int x){
  access(x),splay(x);
  nd[x].rev^=1;
}
void make_root(int x){
  nd[access(x)].rev^=1;
  splay(x);
}
void cut(int x,int y){
  make_root(x);
  access(y);
  splay(y);
  nd[y].ch[0]=0;
  nd[x].pa=0;
}
void cut_parents(int x){
  access(x);
  splay(x);
  nd[nd[x].ch[0]].pa=0;
  nd[x].ch[0]=0;
}
void link(int x,int y){
  make_root(x);
  nd[x].pa=y;
}
int find_root(int x){
  x=access(x);
  while(nd[x].ch[0])x=nd[x].ch[0];
  splay(x);
  return x;
}
int query(int u,int v){
//傳回uv路徑splay tree的根結點
//這種寫法無法求LCA
  make_root(u);
  return access(v);
}
int query_lca(int u,int v){
//假設求鏈上點權的總和，sum是子樹的權重和，data是節點的權重
  access(u);
  int lca=access(v);
  splay(u);
  if(u==lca){
    //return nd[lca].data+nd[nd[lca].ch[1]].sum
  }else{
    //return nd[lca].data+nd[nd[lca].ch[1]].sum+nd[u].sum
  }
}
struct EDGE{
  int a,b,w;
}e[10005];
int n;
vector<pair<int,int>> G[10005];
//first表示子節點，second表示邊的編號
int pa[10005],edge_node[10005];
//pa是父母節點，暫存用的，edge_node是每個編被存在哪個點裡面的陣列
void bfs(int root){
//在建構的時候把每個點都設成一個splay tree
  queue<int > q;
  for(int i=1;i<=n;++i)pa[i]=0;
  q.push(root);
  while(q.size()){
    int u=q.front();
    q.pop();
    for(auto P:G[u]){
      int v=P.first;
      if(v!=pa[u]){
        pa[v]=u;
        nd[v].pa=u;
        nd[v].data=e[P.second].w;
        edge_node[P.second]=v;
        up(v);
        q.push(v);
      }
    }
  }
}
void change(int x,int b){
  splay(x);
  //nd[x].data=b;
  up(x);
}
```

8.6 Tree centroid

```cpp
//找出其中一個樹重心
vector<int> size;

int ans = -1;
void dfs(int u, int parent = -1) {
  size[u] = 1;
  int max_son_size = 0;
  for (auto v : Tree[u]) {
    if (v == parent) continue;
    dfs(v, u);
    size[u] += size[v];
    max_son_size = max(max_son_size, size[v]);
  }
  max_son_size = max(max_son_size, n - size[u]);
  if (max_son_size <= n / 2) ans = u;
}
```

8.7 Tree diameter

```cpp
//dfs兩次
vector<int> level;

void dfs(int u, int parent = -1) {
  if(parent == -1) level[u] = 0;
  else level[u] = level[parent] + 1;
  for (int v : Tree[u]) {
    if (v == parent) continue;
    dfs(v, u);
  }
}

dfs(1); // 隨便選一個點
int a = max_element(level.begin(), level.end()) - level.begin();
dfs(a); // a 必然是直徑的其中一個端點
int b = max_element(level.begin(), level.end()) - level.begin();
cout << level[b] << endl;

//紀錄每個點的最長距離跟次長距離
vector<int> D1, D2; // 最遠、次遠距離
int ans = 0; // 直徑長度

void dfs(int u, int parent = -1) {
  D1[u] = D2[u] = 0;
  for (int v : Tree[u]) {
    if (v == parent) continue;
    dfs(v, u);
    int dis = D1[v] + 1;
    if (dis > D1[u]) {
      D2[u] = D1[u];
      D1[u] = dis;
    } else
      D2[u] = max(D2[u], dis);
  }
  ans = max(ans, D1[u] + D2[u]);
}
```

8.8 樹壓平

```cpp
//紀錄in & out
vector<int> Arr;
vector<int> In, Out;
void dfs(int u) {
  Arr.push_back(u);
  In[u] = Arr.size() - 1;
  for (auto v : Tree[u]) {
    if (v == parent[u])
      continue;
    parent[v] = u;
    dfs(v);
  }
  Out[u] = Arr.size() - 1;
}

//進去出來都紀錄
vector<int> Arr;
void dfs(int u) {
  Arr.push_back(u);
  for (auto v : Tree[u]) {
    if (v == parent[u])
      continue;
    parent[v] = u;
    dfs(v);
  }
  Arr.push_back(u);
}

//用Treap紀錄
Treap *root = nullptr;
vector<Treap *> In, Out;
void dfs(int u) {
  In[u] = new Treap(cost[u]);
  root = merge(root, In[u]);
  for (auto v : Tree[u]) {
    if (v == parent[u])
      continue;
    parent[v] = u;
    dfs(v);
  }
  Out[u] = new Treap(0);
  root = merge(root, Out[u]);
}
//Treap紀錄Parent
struct Treap {
  Treap *lc = nullptr, *rc = nullptr;
  Treap *pa = nullptr;
  unsigned pri, size;
  long long Val, Sum;
  Treap(int Val):
    pri(rand()), size(1),
    Val(Val), Sum(Val) {}
  void pull();
};

```

```cpp
void Treap::pull() {
  size = 1;
  Sum = Val;
  pa = nullptr;
  if (lc) {
    size += lc->size;
    Sum += lc->Sum;
    lc->pa = this;
  }
  if (rc) {
    size += rc->size;
    Sum += rc->Sum;
    rc->pa = this;
  }
}
//找出節點在中序的編號
size_t getIdx(Treap *x) {
  assert(x);
  size_t Idx = 0;
  for (Treap *child = x->rc; x;) {
    if (child == x->rc)
      Idx += 1 + size(x->lc);
    child = x;
    x = x->pa;
  }
  return Idx;
}
//切出想要的東西
void move(Treap *&root, int a, int b) {
  size_t a_in = getIdx(In[a]), a_out = getIdx(Out[a]);
  auto [L, tmp] = splitK(root, a_in - 1);
  auto [tree_a, R] = splitK(tmp, a_out - a_in + 1);
  root = merge(L, R);
  tie(L, R) = splitK(root, getIdx(In[b]));
  root = merge(L, merge(tree_a, R));
}
```

9 string

9.1 AC 自動機

```cpp
template<char L='a',char R='z'>
class ac_automaton{
  struct joe{
    int next[R-L+1],fail,efl,ed,cnt_dp,vis;
    joe():ed(0),cnt_dp(0),vis(0){
      for(int i=0;i<=R-L;++i)next[i]=0;
    }
  };
public:
  std::vector<joe> S;
  std::vector<int> q;
  int qs,qe,vt;
  ac_automaton():S(1),qs(0),qe(0),vt(0){}
  void clear(){
    q.clear();
    S.resize(1);
    for(int i=0;i<=R-L;++i)S[0].next[i]=0;
    S[0].cnt_dp=S[0].vis=qs=qe=vt=0;
  }
  void insert(const char *s){
    int o=0;
    for(int i=0,id;s[i];++i){
      id=s[i]-L;
      if(!S[o].next[id]){
        S.push_back(joe());
        S[o].next[id]=S.size()-1;
      }
      o=S[o].next[id];
    }
    ++S[o].ed;
  }
  void build_fail(){
    S[0].fail=S[0].efl=-1;
    q.clear();
    q.push_back(0);
    ++qe;
    while(qs!=qe){
      int pa=q[qs++],id,t;
      for(int i=0;i<=R-L;++i){
        t=S[pa].next[i];
        if(!t)continue;
        id=S[pa].fail;
        while(~id&&!S[id].next[i])id=S[id].fail;
        S[t].fail=~id?S[id].next[i]:0;
        S[t].efl=S[S[t].fail].ed?S[t].fail:S[S[t].fail].efl;
        q.push_back(t);
        ++qe;
      }
    }
  }
  /*DP出每個前綴在字串s出現的次數並傳回所有字串被s匹配成功的次數O(N+M)*/
  int match_0(const char *s){
    int ans=0,id,p=0,i;
    for(i=0;s[i];++i){
      id=s[i]-L;
      while(!S[p].next[id]&&p)p=S[p].fail;
      if(!S[p].next[id])continue;
      p=S[p].next[id];
      ++S[p].cnt_dp;/*匹配成功則它所有後綴都可以被匹配(DP計算)*/
    }
    for(i=qe-1;i>=0;--i){
      ans+=S[q[i]].cnt_dp*S[q[i]].ed;
      if(~S[q[i]].fail)S[S[q[i]].fail].cnt_dp+=S[q[i]].cnt_dp;
    }
    return ans;
  }
  /*多串匹配走efl邊並傳回所有字串被s匹配成功的次數O(N*M^1.5)*/
  int match_1(const char *s)const{
    int ans=0,id,p=0,t;
    for(int i=0;s[i];++i){
      id=s[i]-L;
      while(!S[p].next[id]&&p)p=S[p].fail;
      if(!S[p].next[id])continue;
      p=S[p].next[id];
      if(S[p].ed)ans+=S[p].ed;
      for(t=S[p].efl;~t;t=S[t].efl){
        ans+=S[t].ed;/*因為都走efl邊所以保證匹配成功*/
      }
    }
    return ans;
  }
  /*枚舉(s的子字串∩A)的所有相異字串各恰一次並傳回次數O(N*M^(1/3))*/
  int match_2(const char *s){
    int ans=0,id,p=0,t;
    ++vt;
    /*把戳記vt+=1，只要vt沒溢位，所有S[p].vis==vt就會變成false
    這種利用vt的方法可以O(1)歸零vis陣列*/
    for(int i=0;s[i];++i){
      id=s[i]-L;
      while(!S[p].next[id]&&p)p=S[p].fail;
      if(!S[p].next[id])continue;
      p=S[p].next[id];
      if(S[p].ed&&S[p].vis!=vt){
        S[p].vis=vt;
        ans+=S[p].ed;
      }
      for(t=S[p].efl;~t&&S[t].vis!=vt;t=S[t].efl){
        S[t].vis=vt;
        ans+=S[t].ed;/*因為都走efl邊所以保證匹配成功*/
      }
    }
    return ans;
  }
  /*把AC自動機變成真的自動機*/
  void evolution(){
    for(qs=1;qs!=qe;){
      int p=q[qs++];
      for(int i=0;i<=R-L;++i)
        if(S[p].next[i]==0)S[p].next[i]=S[S[p].fail].next[i];
    }
  }
};
```

9.2 KMP

```cpp
const int N = 1e6+5;
/*產生fail function*/
void kmp_fail(char *s,int len,int *fail){
  int id=-1;
  fail[0]=-1;
  for(int i=1;i<len;++i){
    while(~id&&s[id+1]!=s[i])id=fail[id];
    if(s[id+1]==s[i])++id;
    fail[i]=id;
  }
}
vector<int> match_index;
/*以字串B匹配字串A，傳回匹配成功的數量(用B的fail)*/
int kmp_match(char *A,int lenA,char *B,int lenB,int *fail){
  int id=-1,ans=0;
  for(int i=0;i<lenA;++i){
    while(~id&&B[id+1]!=A[i])id=fail[id];
    if(B[id+1]==A[i])++id;
    if(id==lenB-1){/*匹配成功*/
      ++ans, id=fail[id];
      match_index.emplace_back(i + 1 -lenB);
    }
  }
  return ans;
}
```

9.3 manacher

```cpp
//找最長迴文子字串
//原字串: asdsasdsa
//先把字串變成這樣: @#a#s#d#s#a#s#d#s#a#
void manacher(char *s,int len,int *z){
  int l=0,r=0;
  for(int i=1;i<len;++i){
    z[i]=r>i?min(z[2*l-i],r-i):1;
    while(s[i+z[i]]==s[i-z[i]])++z[i];
    if(z[i]+i>r)r=z[i]+i,l=i;
  }//ans = max(z)-1
}
```

9.4 minimal string rotation

```cpp
//找最小循環表示法起始位置
int min_string_rotation(const string &s){
  int n=s.size(),i=0,j=1,k=0;
  while(i<n&&j<n&&k<n){
    int t=s[(i+k)%n]-s[(j+k)%n];
    ++k;
    if(t){
      if(t>0)i+=k;
      else j+=k;
      if(i==j)++j;
      k=0;
    }
  }
  return min(i,j);//最小循環表示法起始位置
}
```

9.5 reverseBWT

```cpp
const int MAXN = 305, MAXC = 'Z';
int ranks[MAXN], tots[MAXC], first[MAXC];
void rankBWT(const string &bw){
  memset(ranks,0,sizeof(int)*bw.size());
  memset(tots,0,sizeof(tots));
  for(size_t i=0;i<bw.size();++i)
    ranks[i] = tots[int(bw[i])]++;
}
void firstCol(){
  memset(first,0,sizeof(first));
  int totc = 0;
  for(int c='A';c<='Z';++c){
    if(!tots[c]) continue;
    first[c] = totc;
    totc += tots[c];
  }
}
string reverseBwt(string bw,int begin){
  rankBWT(bw), firstCol();
  int i = begin; //原字串最後一個元素的位置
  string res;
  do{
    char c = bw[i];
    res = c + res;
    i = first[int(c)] + ranks[i];
  }while( i != begin );
  return res;
}
```

9.6 Rolling Hash

```
//Rolling Hash(10 Hash) CF 1800 D. Remove
     Two Letters

#include <bits/stdc++.h>
using namespace std;

constexpr long long power(long long x, long
     long n, int m) {
  if(m == 1) return 0;
  unsigned int _m = (unsigned int)(m);
  unsigned long long r = 1;
  x %= m;
  if(x < 0) {
    x += m;
  }
  unsigned long long y = x;
  while(n) {
    if(n & 1) r = (r * y) % _m;
    y = (y * y) % _m;
    n >>= 1;
  }
  return r;
}

template<int HASH_COUNT, bool
     PRECOMPUTE_POWERS = false>
class Hash {
public:
  static constexpr int MAX_HASH_PAIRS = 10;

  // {mul, mod}
  static constexpr const pair<int, int>
       HASH_PAIRS[] = {{827167801,
       999999937},
                                  {998244353,
                                       999999929},
                                  {146672737,
                                       922722049},
                                  {204924373,
                                       952311013},
                                  {585761567,
                                       955873937},
                                  {484547929,
                                       901981687},
                                  {856009481,
                                       987877511},
                                  {852853249,
                                       996724213},
                                  {937381759,
                                       994523539},
                                  {116508269,
                                       993179543}};

  Hash() : Hash("") {}

  Hash(const string& s) : n(s.size()) {
    static_assert(HASH_COUNT > 0 &&
         HASH_COUNT <= MAX_HASH_PAIRS);
    for(int i = 0; i < HASH_COUNT; ++i) {
      const auto& p = HASH_PAIRS[i];
      pref[i].resize(n);
      pref[i][0] = s[0];
      for(int j = 1; j < n; ++j) {
        pref[i][j] = (1LL * pref[i][j - 1] *
             p.first + s[j]) % p.second;
      }
    }
    if(PRECOMPUTE_POWERS) {
      build_powers(n);
    }
  }

  void add_char(char c) {
    for(int i = 0; i < HASH_COUNT; ++i) {
      const auto& p = HASH_PAIRS[i];
      pref[i].push_back((1LL * (n == 0 ? 0 :
           pref[i].back()) * p.first + c) %
           p.second);
    }
    n += 1;
    if(PRECOMPUTE_POWERS) {
      build_powers(n);
    }
  }

  // Return hash values for [l, r)
  array<int, HASH_COUNT> substr(int l, int r
       ) {
    array<int, HASH_COUNT> res{};
    for(int i = 0; i < HASH_COUNT; ++i) {
      res[i] = substr(i, l, r);
    }
    return res;
  }

  array<int, HASH_COUNT> merge(const vector<
       pair<int, int>>& seg) {
    array<int, HASH_COUNT> res{};
    for(int i = 0; i < HASH_COUNT; ++i) {
      const auto& p = HASH_PAIRS[i];
      for(auto [l, r] : seg) {
        res[i] = (1LL * res[i] * get_power(i
             , r - l) + substr(i, l, r)) % p.
             second;
      }
    }
    return res;
  }

  // build powers up to x^k
  void build_powers(int k) {
    for(int i = 0; i < HASH_COUNT; ++i) {
      const auto& p = HASH_PAIRS[i];
      int sz = (int) POW[i].size();
      if(sz > k) {
        continue;
      }
      if(sz == 0) {
        POW[i].push_back(1);
        sz = 1;
      }
      while(sz <= k) {
        POW[i].push_back(1LL * POW[i].back()
             * p.first % p.second);
        sz += 1;
      }
    }
  }

  inline int size() const {
    return n;
  }

private:
  int n;
  static vector<int> POW[MAX_HASH_PAIRS];
  array<vector<int>, HASH_COUNT> pref;

  int substr(int k, int l, int r) {
    assert(0 <= k && k < HASH_COUNT);
    assert(0 <= l && l <= r && r <= n);
    const auto& p = HASH_PAIRS[k];
    if(l == r) {
      return 0;
    }
    int res = pref[k][r - 1];
    if(l > 0) {
      res -= 1LL * pref[k][l - 1] *
           get_power(k, r - l) % p.second;
    }
    if(res < 0) {
      res += p.second;
    }
    return res;
  }

  int get_power(int a, int b) {
    if(PRECOMPUTE_POWERS) {
      build_powers(b);
      return POW[a][b];
    }
    const auto& p = HASH_PAIRS[a];
    return power(p.first, b, p.second);
  }
};
template<int A, bool B> vector<int> Hash<A,
     B>::POW[Hash::MAX_HASH_PAIRS];
void solve() {
  int n;
  string s;
  cin >> n >> s;
  Hash<10, true> h(s);
  set<array<int, 10>> used;
  for(int i = 0; i + 1 < n; ++i) {
    used.insert(h.merge({{0, i}, {i + 2, n
         }}));
  }
  cout << used.size() << "\n";
}

int main() {
  ios::sync_with_stdio(false);
  cin.tie(0);
  int tt;
  cin >> tt;
  while(tt--) {
    solve();
  }
  return 0;
}
```

9.7 suffix array lcp

```
#define radix_sort(x,y){\
  for(i=0;i<A;++i)c[i]=0;\
  for(i=0;i<n;++i)c[x[y[i]]]++;\
  for(i=1;i<A;++i)c[i]+=c[i-1];\
  for(i=n-1;~i;--i)sa[--c[x[y[i]]]]=y[i];\
}
#define AC(r,a,b)\
  r[a]!=r[b]||a+k>=n||r[a+k]!=r[b+k]
void suffix_array(const char *s,int n,int *
     sa,int *rank,int *tmp,int *c){
  int A='z'+1,i,k,id=0;
  for(i=0;i<n;++i)rank[tmp[i]=i]=s[i];
  radix_sort(rank,tmp);
  for(k=1;id<n-1;k<<=1){
    for(id=0,i=n-k;i<n;++i)tmp[id++]=i;
    for(i=0;i<n;++i)
      if(sa[i]>=k)tmp[id++]=sa[i]-k;
    radix_sort(rank,tmp);
    swap(rank,tmp);
    for(rank[sa[0]]=id=0,i=1;i<n;++i)
      rank[sa[i]]=id+=AC(tmp,sa[i-1],sa[i]);
    A=id+1;
  }
}
//h:高度數組 sa:後綴數組 rank:排名
void suffix_array_lcp(const char *s,int len,
     int *h,int *sa,int *rank){
  for(int i=0;i<len;++i)rank[sa[i]]=i;
  for(int i=0,k=0;i<len;++i){
    if(rank[i]==0)continue;
    if(k)--k;
    while(s[i+k]==s[sa[rank[i]-1]+k])++k;
    h[rank[i]]=k;
  }
  h[0]=0;// h[k]=lcp(sa[k],sa[k-1]);
}
```

9.8 Trie

```
template<int ALPHABET = 26, char MIN_CHAR = 'a'>
class trie {
public:
  struct Node {
    int go[ALPHABET];
    Node() {
      memset(go, -1, sizeof(go));
    }
  };

  trie() {
    newNode();
  }

  inline int next(int p, int v) {
    return nodes[p].go[v] != -1 ? nodes[p].go[v] : nodes[p].go[v] = newNode();
  }

  inline void insert(const vector<int>& a, int p = 0) {
    for(int v : a) {
      p = next(p, v);
    }
  }

  inline void clear() {
    nodes.clear();
    newNode();
  }

  inline int longest_common_prefix(const vector<int>& a, int p = 0) const {
    int ans = 0;
    for(int v : a) {
      if(nodes[p].go[v] != -1) {
        ans += 1;
        p = nodes[p].go[v];
      } else {
        break;
      }
    }
    return ans;
  }

private:
  vector<Node> nodes;

  inline int newNode() {
    nodes.emplace_back();
    return (int) nodes.size() - 1;
  }
};
```

9.9 Z

```
void z_alg(char *s,int len,int *z){
  int l=0,r=0;
  z[0]=len;
  for(int i=1;i<len;++i){
    z[i]=i>r?0:(i-l+z[i-l]<z[l]?z[i-l]:r-i+1);
    while(i+z[i]<len&&s[i+z[i]]==s[z[i]])++z[i];
    if(i+z[i]-1>r)r=i+z[i]-1,l=i;
  }
}
```

10 tools

10.1 bitset

```
bitset<size> b(a):長度為size，初始化為a
b[i]:第i位元的值(0 or 1)
b.size():有幾個位元
b.count():有幾個1
b.set():所有位元設為1
b.reset():所有位元設為0
b.flip():所有位元反轉
```

10.2 Bsearch

```
//Lower bound
int lower_bound(int arr[], int n, int val) {
    int l = 0, r = n-1, mid, ret = -1;//沒搜到return -1
    while (l <= r) {
        mid = (l+r)/2;
        if (arr[mid] >= val) ret = mid, r = mid-1;
        else l = mid+1;
    }
    return ret;
}
```

10.3 Counting Sort

```
vector<unsigned> counting_sort(const vector<unsigned> &Arr, unsigned K) {
  vector<unsigned> Bucket(k, 0);
  for(auto x: Arr)
    ++Bucket[x];
  partial_sum(Bucket.begin(), Bucket.end(), Bucket.begin());
  vector<unsigned> Ans(Arr.size());
  for(auto Iter = Arr.rbegin(); Iter != Arr.rend(); ++Iter) Ans[--Bucket[*Iter]] = *Iter;
  return Ans;
}
```

10.4 DuiPai

```
#include<bits/stdc++.h>
using namespace std;
int main(){
  string sol,bf,make;
  cout<<"Your solution file name :";
  cin>>sol;
  cout<<"Brute force file name :";
  cin>>bf;
  cout<<"Make data file name :";
  cin>>make;
  system(("g++ "+sol+" -o sol").c_str());
  system(("g++ "+bf+" -o bf").c_str());
  system(("g++ "+make+" -o make").c_str());
  for(int t = 0;t<10000;++t){
    system("./make > ./1.in");
    double st = clock();
        system("./sol < ./1.in > ./1.ans");
        double et = clock();
        system("./bf < ./1.in > ./1.out");
        if(system("diff ./1.out ./1.ans")) {
      printf("\033[0;31mWrong Answer\033[0m on test #%d",t);
            return 0;
        }
    else if(et-st>=2000){
      printf("\033[0;32mTime limit exceeded\033[0m on test #%d, Time %.0lfms\n",t,et-st);
      return 0;
    }
    else {
            printf("\033[0;32mAccepted\033[0m on test #%d, Time %.0lfms\n", t, et - st);
        }
  }
}
```

10.5 HashMap

```
struct splitmix64_hash {
  static ull splitmix64(ull x) {
    x += 0x9e3779b97f4a7c15;
    x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
    x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
    return x ^ (x >> 31);
  }

  ull operator()(ull x) const {
    static const ull FIXED_RANDOM = RAND;
    return splitmix64(x + FIXED_RANDOM);
  }
};

template<class T, class U, class H = splitmix64_hash> using hash_map = gp_hash_table<T, U, H>;
template<class T, class H = splitmix64_hash> using hash_set = hash_map<T, null_type, H>;
```

10.6 pragma

```
#pragma GCC optimize("Ofast,unroll-loops")
#pragma GCC target("sse,sse2,ssse3,sse4,popcnt,abm,mmx,avx,tune=native")
#pragma GCC optimize("inline")
#pragma GCC optimize("-fgcse")
#pragma GCC optimize("-fgcse-lm")
#pragma GCC optimize("-fipa-sra")
#pragma GCC optimize("-ftree-pre")
#pragma GCC optimize("-ftree-vrp")
#pragma GCC optimize("-fpeephole2")
#pragma GCC optimize("-ffast-math")
#pragma GCC optimize("-fsched-spec")
#pragma GCC optimize("-falign-jumps")
#pragma GCC optimize("-falign-loops")
#pragma GCC optimize("-falign-labels")
#pragma GCC optimize("-fdevirtualize")
#pragma GCC optimize("-fcaller-saves")
#pragma GCC optimize("-fcrossjumping")
#pragma GCC optimize("-fthread-jumps")
#pragma GCC optimize("-funroll-loops")
#pragma GCC optimize("-fwhole-program")
#pragma GCC optimize("-freorder-blocks")
#pragma GCC optimize("-fschedule-insns")
#pragma GCC optimize("inline-functions")
#pragma GCC optimize("-ftree-tail-merge")
#pragma GCC optimize("-fschedule-insns2")
#pragma GCC optimize("-fstrict-aliasing")
#pragma GCC optimize("-fstrict-overflow")
#pragma GCC optimize("-falign-functions")
#pragma GCC optimize("-fcse-skip-blocks")
#pragma GCC optimize("-fcse-follow-jumps")
#pragma GCC optimize("-fsched-interblock")
#pragma GCC optimize("-fpartial-inlining")
#pragma GCC optimize("no-stack-protector")
#pragma GCC optimize("-freorder-functions")
#pragma GCC optimize("-findirect-inlining")
#pragma GCC optimize("-fhoist-adjacent-loads")
#pragma GCC optimize("-frerun-cse-after-loop")
#pragma GCC optimize("inline-small-functions")
#pragma GCC optimize("-finline-small-functions")
#pragma GCC optimize("-ftree-switch-conversion")
#pragma GCC optimize("-foptimize-sibling-calls")
#pragma GCC optimize("-fexpensive-optimizations")
```

```
#pragma GCC optimize("-funsafe-loop-optimizations")
#pragma GCC optimize("inline-functions-called-once")
#pragma GCC optimize("-fdelete-null-pointer-checks")
```

10.7 relabel

```
template<class T>
vector<int> Discrete(const vector<T>&v){
  vector<int>ans;
  vector<T>tmp(v);
  sort(begin(tmp),end(tmp));
  tmp.erase(unique(begin(tmp),end(tmp)),end(tmp));
  for(auto i:v)ans.push_back(lower_bound(begin(tmp),end(tmp),i)-tmp.begin()+1);
  return ans;
}
```

10.8 Template

```
#include <bits/extc++.h>
#include <bits/stdc++.h>
#pragma GCC optimize("O3,unroll-loops")
#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")
#define IOS ios::sync_with_stdio(0),cin.tie(0),cout.tie(0)
#define int long long
#define double long double
#define pb push_back
#define sz(x) (int)(x).size()
#define all(v) begin(v),end(v)
#define debug(x) cerr<<#x<<" = "<<x<<'\n'
#define LINE cout<<"\n------------------\n"
#define endl '\n'
#define VI vector<int>
#define F first
#define S second
#define MP(a,b) make_pair(a,b)
#define rep(i,m,n) for(int i = m;i<=n;++i)
#define res(i,m,n) for(int i = m;i>=n;--i)
#define gcd(a,b) __gcd(a,b)
#define lcm(a,b) a*b/gcd(a,b)
#define Case() int _;cin>>_;for(int Case = 1;Case<=_;++Case)
#define pii pair<int,int>
using namespace __gnu_cxx;
using namespace __gnu_pbds;
using namespace std;
template <typename K, typename cmp = less<K>, typename T = thin_heap_tag> using _heap = __gnu_pbds::priority_queue<K, cmp, T>;
template <typename K, typename M = null_type> using _hash = gp_hash_table<K, M>;
const int N = 1e6+5,L = 20,mod = 1e9+7;
const long long inf = 2e18+5;
const double eps = 1e-7,pi = acos(-1);
void solve(){
}
signed main(){
  IOS;
  solve();
}

//使用內建紅黑樹
template<class T, typename cmp=less<>>struct _tree{//#include<bits/extc++.h>
  tree<pair<T,int>,null_type,cmp,rb_tree_tag,tree_order_statistics_node_update>st;
  int id = 0;
  void insert(T x){st.insert({x,id++});}
  void erase(T x){st.erase(st.lower_bound({x,0}));}
  int order_of_key(T x){return st.order_of_key(*st.lower_bound({x,0}));}
  T find_by_order(int x){return st.find_by_order(x)->first;}
  T lower_bound(T x){return st.lower_bound({x,0})->first;}
  T upper_bound(T x){return st.upper_bound({x,(int)1e9+7})->first;}
  T smaller_bound(T x){return (--st.lower_bound({x,0}))->first;}
};
```

10.9 template bubble

```
#include<bits/stdc++.h>
#define lim 1000000007
#define ll long long
#define endl "\n"
#define Crbubble cin.tie(0); ios_base::sync_with_stdio(false);
#define aqua clock_t qua = clock();
#define aquaa cout << "Aqua says: " << (double)(clock()-qua)/CLOCKS_PER_SEC << " sec!\n";
#define random_set(m,n) random_device rd; \
                        mt19937 gen=mt19937(rd()); \
                        uniform_ll_distribution<ll> dis(m,n); \
                        auto rnd=bind(dis,gen);
```

10.10 TenarySearch

```
// return the maximum of $f(x)$ in $[l, r]$
double ternary_search(double l, double r) {
  while(r - l > EPS) {
    double m1 = l + (r - l) / 3;
    double m2 = r - (r - l) / 3;
    double f1 = f(m1), f2 = f(m2);
    if(f1 < f2) l = m1;
    else r = m2;
  }
  return f(l);
}

// return the maximum of $f(x)$ in $(l, r]$
int ternary_search(int l, int r) {
  while(r - l > 1) {
    int mid = (l + r) / 2;
    if(f(m) > f(m + 1)) r = m;
    else l = m;
  }
  return r;
}
```

ACM ICPC Team Reference - Angry Crow Takes Flight!

Contents

ACM ICPC Judge Test - Angry Crow Takes Flight!

C++ Resource Test

```cpp
#include <bits/stdc++.h>
using namespace std;

namespace system_test {

const size_t KB = 1024;
const size_t MB = KB * 1024;
const size_t GB = MB * 1024;

size_t block_size, bound;
void stack_size_dfs(size_t depth = 1) {
  if (depth >= bound)
    return;
  int8_t ptr[block_size]; // 若無法編譯將 block_size 改成常數
  memset(ptr, 'a', block_size);
  cout << depth << endl;
  stack_size_dfs(depth + 1);
}

void stack_size_and_runtime_error(size_t block_size, size_t bound = 1024) {
  system_test::block_size = block_size;
  system_test::bound = bound;
  stack_size_dfs();
}

double speed(int iter_num) {
  const int block_size = 1024;
  volatile int A[block_size];
  auto begin = chrono::high_resolution_clock::now();
  while (iter_num--)
    for (int j = 0; j < block_size; ++j)
      A[j] += j;
  auto end = chrono::high_resolution_clock::now();
  chrono::duration<double> diff = end - begin;
  return diff.count();
}

void runtime_error_1() {
  // Segmentation fault
  int *ptr = nullptr;
  *(ptr + 7122) = 7122;
}

void runtime_error_2() {
  // Segmentation fault
  int *ptr = (int *)memset;
  *ptr = 7122;
}

void runtime_error_3() {
  // munmap_chunk(): invalid pointer
  int *ptr = (int *)memset;
  delete ptr;
}

void runtime_error_4() {
  // free(): invalid pointer
  int *ptr = new int[7122];
  ptr += 1;
  delete[] ptr;
}

void runtime_error_5() {
  // maybe illegal instruction
  int a = 7122, b = 0;
  cout << (a / b) << endl;
}

void runtime_error_6() {
  // floating point exception
  volatile int a = 7122, b = 0;
  cout << (a / b) << endl;
}

void runtime_error_7() {
  // call to abort.
  assert(false);
}

} // namespace system_test

#include <sys/resource.h>
void print_stack_limit() { // only work in Linux
  struct rlimit l;
  getrlimit(RLIMIT_STACK, &l);
  cout << "stack_size = " << l.rlim_cur << " byte" << endl;
}
```