1 Data_Structure

1.1 Sparse Table

```
struct SparseTable {
    const int N;
    vector<vector<ll>> bitarray;
    // Constructs a sparse table with size same as vec.
    SparseTable(const vector<ll>& vec) : N(vec.size()) {
        bitarray.assign(__lg(N) + 1, vector<ll>(N));
        bitarray[0] = vec;
        for (int i = 1; (1 << i) <= N; i++) {
            int k = N - (1 << i);
            for (int j = 0; j < k; j++) {
                // Comparator may be needed.
                bitarray[i][j] = max(bitarray[i - 1][j],
                    bitarray[i - 1][j + (1 << (i - 1))]);
            }
        }
    }
    // Queries the maximum element in [left_inc, right_inc]
        where left_inc and right_inc in [0, n).
    ll query(int left_inc, int right_inc) {
        int k = __lg(right_inc - left_inc + 1);
        return max(bitarray[k][left_inc], bitarray[k][
            right_inc - (1 << k) + 1]);
    }
};
```

1.2 Binary Indexed Tree

```
struct BIT {
    int N;
    vector<ll> bitarray;
    // Constructs a Binary Indexed Tree where n is size of
        array.
    BIT(int n) : N(n) { bitarray.assign(N + 1, 0); }
    ll query(int index) {
        index++;
        ll ret = 0;
        while (index > 0) ret += bitarray[index], index -=
            index & -index;
        return ret;
    }
    // Increases element at index by val, where index in [1,
        n].
    void add(int index, ll val) {
        index++;
        while (index <= N) bitarray[index] += val, index +=
            index & -index;
    }
    // Queries sum in [left_inc, right_inc], where left_inc
        and right_inc in [1, n].
    ll query(int left_inc, int right_inc) { return query(
        right_inc) - query(left_inc - 1); }
};
```

1.3 Fenwick Tree

```
struct FenwickTree {
    const int N;
    vector<ll> d, dd;
    // Constructs a Fenwick Tree where n is size of array.
    FenwickTree(int n) : N(n) {
        d.assign(N + 2, 0);
        dd.assign(N + 2, 0);
    }
    ll query(int index) {
        index++;
        ll s = 0, ss = 0;
        ll c = index + 1;
        while (index > 0) {
            s += d[index], ss += dd[index];
            index -= index & -index;
        }
        return c * s - ss;
    }
    void add(int index, ll val) {
        index++;
        int c = index;
        while (index <= N) {
            d[index] += val, dd[index] += c * val;
            index += index & -index;
        }
    }
    // Queries sum in [left_inc, right_inc], where left_inc
        and right_inc in [1, n].
    ll query(int left_inc, int right_inc) { return query(
        right_inc) - query(left_inc - 1); }
    // Increases all elements in [left_inc, right_inc] by val
        , where left_inc and right_inc in [1, n].
    void add(int left_inc, int right_inc, ll val) { add(
        left_inc, val), add(right_inc + 1, -val); }
};
```

1.4 線段樹

```
#define INF 2147483647
struct Node {
    int add_tag, chg_tag, Min, Max, sum;
    Node():add_tag(0), chg_tag(0) {}
};
struct ST {
    int sz;
    Node nd[maxn*4];
    ST(){}
    ST(int a[], int n):sz(n) {
        build(a,0,n-1,1);
    }
    void build(int a[],int l,int r,int i) {
        if(l==r) {
            nd[i].Min = nd[i].Max = nd[i].sum = a[l];
            return;
        }
        int mid = (l+r)>>1;
        int lid = i<<1, rid = (i<<1)|1;
        build(a,l,mid,lid), build(a,mid+1,r,rid);
        nd[i].Min = min(nd[lid].Min, nd[rid].Min);
        nd[i].Max = max(nd[lid].Max, nd[rid].Max);
        nd[i].sum = nd[lid].sum + nd[rid].sum;
    }
    void push(int l,int r,int i,int lid,int rid) {
        if(nd[i].chg_tag) {
            nd[i].sum = nd[i].chg_tag * (r-l+1);
            nd[i].Max = nd[i].Min = nd[i].chg_tag;
            if(lid) nd[lid].chg_tag = nd[i].chg_tag, nd[lid].
                add_tag = 0;
            if(rid) nd[rid].chg_tag = nd[i].chg_tag, nd[rid].
                add_tag = 0;
            nd[i].chg_tag = 0;
        }
        if(nd[i].add_tag) {
            nd[i].sum += nd[i].add_tag * (r-l+1);
            nd[i].Max += nd[i].add_tag, nd[i].Min += nd[i].
                add_tag;
            if(lid) nd[lid].add_tag += nd[i].add_tag;
            if(rid) nd[rid].add_tag += nd[i].add_tag;
            nd[i].add_tag = 0;
        }
    }
    void add(int ql,int qr,int l,int r,int i,int val) {
        if(!val) return;
        int lid = i<<1, rid = (i<<1)|1;
        if(l==r) lid = rid = 0;
        push(l,r,i,lid,rid);
        if(r<ql || qr<l) return;
        if(ql<=l && r<=qr) {
            nd[i].add_tag += val;
        }
        else {
            int mid = (l+r)>>1;
            add(ql,qr,l,mid,lid,val), add(ql,qr,mid+1,r,rid,
                val);
            nd[i].Max = max(nd[lid].Max + nd[lid].add_tag, nd
                [rid].Max + nd[rid].add_tag);
            nd[i].Min = min(nd[lid].Min + nd[lid].add_tag, nd
                [rid].Min + nd[rid].add_tag);
            nd[i].sum = nd[lid].sum + nd[lid].add_tag*(mid-l
                +1) + nd[rid].sum + nd[rid].add_tag*(r-mid);
        }
    }
    void chg(int ql,int qr,int l,int r,int i,int val) {
        int lid = i<<1, rid = (i<<1)|1;
        if(l==r) lid = rid = 0;
        push(l,r,i,lid,rid);
        if(r<ql || qr<l) return;
        if(ql<=l && r<=qr) {
            nd[i].chg_tag = val;
        }
        else {
            int mid = (l+r)>>1;
            chg(ql,qr,l,mid,lid,val), chg(ql,qr,mid+1,r,rid,
                val);
            nd[i].Max = max(nd[lid].chg_tag?nd[lid].chg_tag:
                nd[lid].Max, nd[rid].chg_tag?nd[rid].chg_tag
                :nd[rid].Max);
            nd[i].Min = min(nd[lid].chg_tag?nd[lid].chg_tag:
                nd[lid].Min, nd[rid].chg_tag?nd[rid].chg_tag
                :nd[rid].Min);
            nd[i].sum = (nd[lid].chg_tag?nd[lid].chg_tag*(mid
                -l+1):nd[lid].sum) + (nd[rid].chg_tag?nd[rid
                ].chg_tag*(r-mid):nd[rid].sum);
        }
    }
    int query_min(int ql,int qr,int l,int r,int i) {
```

```
        int lid = i<<1, rid = (i<<1)|1;
        if(l==r) lid = rid = 0;
        push(l,r,i,lid,rid);
        if(r<ql || qr<l) return INF;
        if(ql<=l && r<=qr) return nd[i].Min;
        int mid = (l+r)>>1;
        return min(query_min(ql,qr,l,mid,lid), query_min(ql,
            qr,mid+1,r,rid));
    }
    int query_max(int ql,int qr,int l,int r,int i) {
        int lid = i<<1, rid = (i<<1)|1;
        if(l==r) lid = rid = 0;
        push(l,r,i,lid,rid);
        if(r<ql || qr<l) return -INF;
        if(ql<=l && r<=qr) return nd[i].Max;
        int mid = (l+r)>>1;
        return max(query_max(ql,qr,l,mid,lid), query_max(ql,
            qr,mid+1,r,rid));
    }
    int query_sum(int ql,int qr,int l,int r,int i) {
        int lid = i<<1, rid = (i<<1)|1;
        if(l==r) lid = rid = 0;
        push(l,r,i,lid,rid);
        if(r<ql || qr<l) return 0;
        if(ql<=l && r<=qr) return nd[i].sum;
        int mid = (l+r)>>1;
        return query_sum(ql,qr,l,mid,lid) + query_sum(ql,qr,
            mid+1,r,rid);
    }
};
```

1.5 持久化線段樹

```
//POJ 2104 //k-th number
#include<bits/stdc++.h>
#define maxn 100005
using namespace std;
int a[maxn],b[maxn],root[maxn];
int cnt;
struct node {
    int sum, L_son, R_son;
} tree[maxn<<5];
int create(int _sum, int _L_son, int _R_son) {
    int idx = ++cnt;
    tree[idx].sum = _sum, tree[idx].L_son = _L_son, tree[idx
        ].R_son = _R_son;
    return idx;
}
void Insert(int &root,int pre_rt,int pos,int L,int R) {
    root = create(tree[pre_rt].sum+1,tree[pre_rt].L_son,tree[
        pre_rt].R_son);
    if(L==R) return;
    int M = (L+R)>>1;
    if(pos<=M) Insert(tree[root].L_son,tree[pre_rt].L_son,pos
        ,L,M);
    else Insert(tree[root].R_son,tree[pre_rt].R_son,pos,M+1,R
        );
}
int query(int L_id,int R_id,int L,int R,int K) {
    if(L==R) return L;
    int M = (L+R)>>1;
    int s = tree[tree[R_id].L_son].sum - tree[tree[L_id].
        L_son].sum;
    if(K<=s) return query(tree[L_id].L_son,tree[R_id].L_son,L
        ,M,K);
    return query(tree[L_id].R_son,tree[R_id].R_son,M+1,R,K-s)
        ;
}
int main() {
    int n,m; scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++) {
        scanf("%d",&a[i]);
        b[i] = a[i];
    }
    sort(b+1,b+1+n); //離散化
    int b_sz = unique(b+1,b+1+n)-(b+1);
    cnt = root[0] = 0;
    for(int i=1;i<=n;i++) {
        int pos = lower_bound(b+1,b+1+b_sz,a[i])-b;
        Insert(root[i],root[i-1],pos,1,b_sz);
    }
    while(m--) {
        int l,r,k; scanf("%d%d%d",&l,&r,&k);
        int pos = query(root[l-1],root[r],1,b_sz,k);
        printf("%d\n",b[pos]);
    } return 0;
}
```

1.6 Treap

```
//POJ 3580 區間反轉+加值 詢問min
#include<bits/stdc++.h>
using namespace std;
#define maxn 100005
#define INF 2147483647
struct Treap
{
    Treap *L, *R;
    int min_val, size, val;
    bool rev_tag;
    int add_tag;
    Treap(int _val):L(NULL),R(NULL),min_val(_val),size(1),
        rev_tag(false),add_tag(0),val(_val){}
    void pull()
    {
        if(L) L->push();
        if(R) R->push();
        size =  (L ? L->size : 0) + (R ? R->size : 0) + 1;
        min_val = min( min( val, L ? L->min_val : INF), R ? R
            ->min_val : INF);
    }
    void push()
    {
        val += add_tag;
        min_val += add_tag;
        if(L) L->add_tag += add_tag;
        if(R) R->add_tag += add_tag;
        add_tag = 0;
        if(rev_tag)
        {
            swap(L,R);
            if(L) L->rev_tag ^= 1;
            if(R) R->rev_tag ^= 1;
            rev_tag = false;
        }
    }
};
inline int Random()
{
    static int x = 7122;
    return (x = x * 0xdefaced + 1) & INF;
}
void split(Treap *p, Treap *&a, Treap *&b, int k)
{
    int sz = p ? p->size : 0;
    if(!p) a = b = NULL;
    else if(k<=0) a = NULL, b = p;
    else
    {
        p->push();
        if(p->L && p->L->size >=k)
        {
            b = p;
            split(p->L, a, b->L, k);
        }
        else
        {
            a = p;
            split(p->R, a->R, b, k - (p->L ? p->L->size : 0)
                -1 );
        }
        p->pull();
    }
}
Treap* merge(Treap *a, Treap *b)
{
    if(!a) return b;
    if(!b) return a;
    if(Random()%(a->size+b->size) < a->size)
    {
        a->push();
        a->R = merge(a->R,b);
        a->pull();
        return a;
    }
    b->push();
    b->L = merge(a,b->L);
    b->pull();
    return b;
}
void insert(Treap *&p, int x, int pos)
{
    Treap *a, *b;
    Treap *t = new Treap(x);
    split(p,a,b,pos);
    p = merge(merge(a,t),b);
}
void del(Treap *&p, int x)
{
    Treap *a, *b, *c, *d;
    split(p,a,d,x-1);
    split(d,b,c,1);
    p = merge(a,c);
}
void reverse(Treap *&p, int x,int y)
{
    Treap *a, *b ,*c;
    split(p,a,c,y);
    split(a,a,b,x-1);
```

```
    b->rev_tag ^= 1;
    p = merge(merge(a,b),c);
}
void add(Treap *&p, int x,int y,int v)
{
    Treap *a, *b ,*c;
    split(p,a,c,y);
    split(a,a,b,x-1);
    b->add_tag += v;
    p = merge(merge(a,b),c);
}
int Min(Treap *&p, int x, int y)
{
    Treap *a, *b, *c;
    split(p,a,c,y);
    split(a,a,b,x-1);
    int ans = b->min_val + b->add_tag;
    p = merge(merge(a,b),c);
    return ans;
}
void revolve(Treap *&p, int x, int y, int t)
{
    Treap *a, *b ,*c, *d ,*e;
    split(p,a,c,y);
    split(a,a,b,x-1);
    split(b,d,e,(y-x+1)-t%(y-x+1));
    p = merge(merge(a,merge(e,d)),c);
}

int main()
{
    int n,m;
    scanf("%d",&n);
    int a[maxn];
    for(int i=1; i<=n; i++)
        scanf("%d",&a[i]);
    Treap* root = new Treap(a[1]);
    for(int i=2; i<=n; i++)
        insert(root,a[i],i);
    scanf("%d",&m);
    while(m--)
    {
        char s[10];
        scanf(" %s",s);
        if(strcmp(s,"ADD")==0)
        {
            int x, y, d;
            scanf("%d%d%d",&x,&y,&d);
            add(root,x,y,d);
        }
        else if(strcmp(s,"REVERSE")==0)
        {
            int x, y;
            scanf("%d%d",&x,&y);
            reverse(root,x,y);
        }
        else if(strcmp(s,"REVOLVE")==0)
        {
            int x, y, t;
            scanf("%d%d%d",&x,&y,&t);
            revolve(root,x,y,t);
        }
        else if(strcmp(s,"INSERT")==0)
        {
            int x, p;
            scanf("%d%d",&x,&p);
            insert(root,p,x);
        }
        else if(strcmp(s,"DELETE")==0)
        {
            int x;
            scanf("%d",&x);
            del(root,x);
        }
        else if(strcmp(s,"MIN")==0)
        {
            int x, y;
            scanf("%d%d",&x,&y);
            printf("%d\n",Min(root,x,y));
        }
    }
    return 0;
}
```

1.7 Dynamic_KD_tree

```
template<typename T,size_t kd>//有kd個維度
struct kd_tree{
  struct point{
    T d[kd];
    T dist(const point &x)const{
      T ret=0;
      for(size_t i=0;i<kd;++i)ret+=abs(d[i]-x.d[i]);
      return ret;
    }
    bool operator==(const point &p){
      for(size_t i=0;i<kd;++i)
        if(d[i]!=p.d[i])return 0;
      return 1;
    }
    bool operator<(const point &b)const{
      return d[0]<b.d[0];
    }
  };
private:
  struct node{
    node *l,*r;
    point pid;
    int s;
    node(const point &p):l(0),r(0),pid(p),s(1){}
    ~node(){delete l,delete r;}
    void up(){s=(l?l->s:0)+1+(r?r->s:0);}
  }*root;
  const double alpha,loga;
  const T INF;//記得要給INF，表示極大值
  int maxn;
  struct __cmp{
    int sort_id;
    bool operator()(const node*x,const node*y)const{
      return operator()(x->pid,y->pid);
    }
    bool operator()(const point &x,const point &y)const{
      if(x.d[sort_id]!=y.d[sort_id])
        return x.d[sort_id]<y.d[sort_id];
      for(size_t i=0;i<kd;++i)
        if(x.d[i]!=y.d[i])return x.d[i]<y.d[i];
      return 0;
    }
  }cmp;
  int size(node *o){return o?o->s:0;}
  vector<node*> A;
  node* build(int k,int l,int r){
    if(l>r) return 0;
    if(k==kd) k=0;
    int mid=(l+r)/2;
    cmp.sort_id = k;
    nth_element(A.begin()+l,A.begin()+mid,A.begin()+r+1,cmp);
    node *ret=A[mid];
    ret->l = build(k+1,l,mid-1);
    ret->r = build(k+1,mid+1,r);
    ret->up();
    return ret;
  }
  bool isbad(node*o){
    return size(o->l)>alpha*o->s||size(o->r)>alpha*o->s;
  }
  void flatten(node *u,typename vector<node*>::iterator &it){
    if(!u)return;
    flatten(u->l,it);
    *it=u;
    flatten(u->r,++it);
  }
  void rebuild(node*&u,int k){
    if((int)A.size()<u->s)A.resize(u->s);
    auto it=A.begin();
    flatten(u,it);
    u=build(k,0,u->s-1);
  }
  bool insert(node*&u,int k,const point &x,int dep){
    if(!u) return u=new node(x), dep<=0;
    ++u->s;
    cmp.sort_id=k;
    if(insert(cmp(x,u->pid)?u->l:u->r,(k+1)%kd,x,dep-1)){
      if(!isbad(u))return 1;
      rebuild(u,k);
    }
    return 0;
  }
  node *findmin(node*o,int k){
    if(!o)return 0;
    if(cmp.sort_id==k)return o->l?findmin(o->l,(k+1)%kd):o;
    node *l=findmin(o->l,(k+1)%kd);
    node *r=findmin(o->r,(k+1)%kd);
    if(l&&!r)return cmp(l,o)?l:o;
    if(!l&&r)return cmp(r,o)?r:o;
    if(!l&&!r)return o;
    if(cmp(l,r))return cmp(l,o)?l:o;
    return cmp(r,o)?r:o;
  }
  bool erase(node *&u,int k,const point &x){
    if(!u)return 0;
    if(u->pid==x){
      if(u->r);
      else if(u->l) u->r=u->l, u->l=0;
      else return delete(u),u=0, 1;
      --u->s;
      cmp.sort_id=k;
      u->pid=findmin(u->r,(k+1)%kd)->pid;
      return erase(u->r,(k+1)%kd,u->pid);
    }
    cmp.sort_id=k;
    if(erase(cmp(x,u->pid)?u->l:u->r,(k+1)%kd,x))
      return --u->s, 1;
    return 0;
```

```
  }
  T heuristic(const T h[])const{
    T ret=0;
    for(size_t i=0;i<kd;++i)ret+=h[i];
    return ret;
  }
  int qM;
  priority_queue<pair<T,point>> pQ;
  void nearest(node *u,int k,const point &x,T *h,T &mndist){
    if(u==0||heuristic(h)>=mndist)return;
    T dist=u->pid.dist(x),old=h[k];
    /*mndist=std::min(mndist,dist);*/
    if(dist<mndist){
      pQ.push(std::make_pair(dist,u->pid));
      if((int)pQ.size()==qM+1)
        mndist=pQ.top().first,pQ.pop();
    }
    if(x.d[k]<u->pid.d[k]){
      nearest(u->l,(k+1)%kd,x,h,mndist);
      h[k] = abs(x.d[k]-u->pid.d[k]);
      nearest(u->r,(k+1)%kd,x,h,mndist);
    }else{
      nearest(u->r,(k+1)%kd,x,h,mndist);
      h[k] = abs(x.d[k]-u->pid.d[k]);
      nearest(u->l,(k+1)%kd,x,h,mndist);
    }
    h[k]=old;
  }
  vector<point>in_range;
  void range(node *u,int k,const point&mi,const point&ma){
    if(!u)return;
    bool is=1;
    for(int i=0;i<kd;++i)
      if(u->pid.d[i]<mi.d[i]||ma.d[i]<u->pid.d[i])
        { is=0;break; }
    if(is) in_range.push_back(u->pid);
    if(mi.d[k]<=u->pid.d[k])range(u->l,(k+1)%kd,mi,ma);
    if(ma.d[k]>=u->pid.d[k])range(u->r,(k+1)%kd,mi,ma);
  }
public:
  kd_tree(const T &INF,double a=0.75):
  root(0),alpha(a),loga(log2(1.0/a)),INF(INF),maxn(1){}
  ~kd_tree(){delete root;}
  void clear(){delete root,root=0,maxn=1;}
  void build(int n,const point *p){
    delete root,A.resize(maxn=n);
    for(int i=0;i<n;++i)A[i]=new node(p[i]);
    root=build(0,0,n-1);
  }
  void insert(const point &x){
    insert(root,0,x,__lg(size(root))/loga);
    if(root->s>maxn)maxn=root->s;
  }
  bool erase(const point &p){
    bool d=erase(root,0,p);
    if(root&&root->s<alpha*maxn)rebuild();
    return d;
  }
  void rebuild(){
    if(root)rebuild(root,0);
    maxn=root->s;
  }
  T nearest(const point &x,int k){
    qM=k;
    T mndist=INF,h[kd]={};
    nearest(root,0,x,h,mndist);
    mndist=pQ.top().first;
    pQ = priority_queue<pair<T,point>>();
    return mndist;//回傳離x第k近的點的距離
  }
  const vector<point> &range(const point&mi,const point&ma){
    in_range.clear();
    range(root,0,mi,ma);
    return in_range;//回傳介於mi到ma之間的點vector
  }
  int size(){return root?root->s:0;}
};
```

1.8 Heavy Light

```
#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;
}
```

1.9 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);
}
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);
}
```

2 DP

2.1 Edit Distance

```
int edit_distance(string s, string t) {
    int n = s.size(), m = t.size();
    memset(dp, 0, sizeof(dp));
    for (int i = 1; i <= n; i++) dp[i][0] = i;
    for (int i = 1; i <= m; i++) dp[0][i] = i;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            dp[i][j] = min(dp[i -1][j - 1] + !(s[i - 1] == t[
                j - 1]), min(dp[i][j - 1], dp[i - 1][j]) +
                1);
    return dp[n][m];
}
```

2.2 LCIS

```
// Longest Common Increasing Subsequense
#include<bits/stdc++.h>
using namespace std;
#define LEN 505
int main()
{
    int n,m;
    scanf("%d%d",&n,&m);
    int a[LEN],b[LEN];
    for(int i=1; i<=n; i++)
        scanf("%d", &a[i]);
    for(int i=1; i<=m; i++)
        scanf("%d", &b[i]);
    int dp[LEN][LEN] = {};  // dp[i][j]:以b[j]結尾的LCIS長度
    int pre[LEN][LEN] = {};  // 用來回溯
    for(int i=1; i<=n; i++)
    {
        int p = 0;
        for(int j=1; j<=m; j++)
        {
            if(a[i]!=b[j])
            {
                dp[i][j] = dp[i-1][j];
                pre[i][j] = j;
                if( a[i]>b[j] && dp[i-1][j]>dp[i-1][p] )
                    p = j;
            }
            else
            {
                dp[i][j] = dp[i-1][p]+1;
                pre[i][j] = p;
            }
        }
    }
    int len = 0, p = 0;
    for(int j=1; j<=m; j++)
    {
        if(dp[n][j]>len)
        {
            len = dp[n][j];
            p = j;
        }
    }
    printf("LEN = %d\n", len);
    vector<int> ans;
    for(int i=n; i>=1; i--)
    {
        if(a[i]==b[p])
            ans.push_back(b[p]);
        p = pre[i][p];
    }
    while(ans.size())
    {
        printf("%d ",ans.back());
        ans.pop_back();
    }
    return 0;
}
```

2.3 Bounded_Knapsack

```
namespace {
    static const int MAXW = 1000005;
    static const int MAXN = 1005;
    struct BB {
        int w, v, c;
        BB(int w = 0, int v = 0, int c = 0): w(w), v(v), c(c)
            {}
        bool operator<(const BB &x) const {
            return w * c  < x.w * x.c;
        }
```

```
    };
    static int run(BB A[], int dp[], int W, int N) {
        static int MQ[MAXW][2];
        for (int i = 0, sum = 0; i < N; i++) {
            int w = A[i].w, v = A[i].v, c = A[i].c;
            sum = min(sum + w*c, W);
            for (int j = 0; j < w; j++) {
                int l = 0, r = 0;
                MQ[l][0] = 0, MQ[l][1] = dp[j];
                for (int k = 1, tw = w+j, tv = v; tw <= sum
                     && k <= c; k++, tw += w, tv += v) {
                    int dpv = dp[tw] - tv;
                    while (l <= r && MQ[r][1] <= dpv) r--;
                    r++;
                    MQ[r][0] = k, MQ[r][1] = dpv;
                    dp[tw] = max(dp[tw], MQ[l][1] + tv);
                }
                for (int k = c+1, tw = (c+1)*w+j, tv = (c+1)*
                     v; tw <= sum; k++, tw += w, tv += v) {
                    if (k - MQ[l][0] > c) l++;
                    int dpv = dp[tw] - tv;
                    while (l <= r && MQ[r][1] <= dpv) r--;
                    r++;
                    MQ[r][0] = k, MQ[r][1] = dpv;
                    dp[tw] = max(dp[tw], MQ[l][1] + tv);
                }
            }
        }
    }
    static int knapsack(int C[][3], int N, int W) { // O(WN)
        vector<BB> A;
        for (int i = 0; i < N; i++) {
            int w = C[i][0], v = C[i][1], c = C[i][2];
            A.push_back(BB(w, v, c));
        }
        assert(N < MAXN);
        static int dp1[MAXW+1], dp2[MAXW+1];
        BB Ar[2][MAXN];
        int ArN[2] = {};
        memset(dp1, 0, sizeof(dp1[0])*(W+1));
        memset(dp2, 0, sizeof(dp2[0])*(W+1));
        sort(A.begin(), A.end());
        int sum[2] = {};
        for (int i = 0; i < N; i++) {
            int ch = sum[1] < sum[0];
            Ar[ch][ArN[ch]] = A[i];
            ArN[ch]++;
            sum[ch] = min(sum[ch] + A[i].w*A[i].c, W);
        }
        run(Ar[0], dp1, W, ArN[0]);
        run(Ar[1], dp2, W, ArN[1]);
        int ret = 0;
        for (int i = 0, j = W, mx = 0; i <= W; i++, j--) {
            mx = max(mx, dp2[i]);
            ret = max(ret, dp1[j] + mx);
        }
        return ret;
    }
}
int main() {
    int W, N;
    assert(scanf("%d %d", &W, &N) == 2);
    int C[MAXN][3];
    for (int i = 0; i < N; i++)
        assert(scanf("%d %d %d", &C[i][1], &C[i][0], &C[i
             ][2]) == 3);
    printf("%d\n", knapsack(C, N, W));
    return 0;
}
```

2.4 1D1D

```
int t, n, L, p;
char s[MAXN][35];
ll sum[MAXN] = {0};
long double dp[MAXN] = {0};
int prevd[MAXN] = {0};
long double pw(long double a, int n) {
    if ( n == 1 ) return a;
    long double b = pw(a, n/2);
    if ( n & 1 ) return b*b*a;
    else return b*b;
}
long double f(int i, int j) {
    // cout << (sum[i] - sum[j]+i-j-1-L) << endl;
    return pw(abs(sum[i] - sum[j]+i-j-1-L), p) + dp[j];
}
struct INV {
    int L, R, pos;
};
INV stk[MAXN*10];
int top = 1, bot = 1;
void update(int i) {
    while ( top > bot && i < stk[top].L && f(stk[top].L, i) <
         f(stk[top].L, stk[top].pos) ) {
        stk[top - 1].R = stk[top].R;
        top--;
    }
    int lo = stk[top].L, hi = stk[top].R, mid, pos = stk[top
        ].pos;
    // if ( i >= lo ) lo = i + 1;
    while ( lo != hi ) {
        mid = lo + (hi - lo) / 2;
        if ( f(mid, i) < f(mid, pos) ) hi = mid;
        else lo = mid + 1;
    }
    if ( hi < stk[top].R ) {
        stk[top + 1] = (INV) { hi, stk[top].R, i };
        stk[top++].R = hi;
    }
}
int main() {
    cin >> t;
    while ( t-- ) {
        cin >> n >> L >> p;
        dp[0] = sum[0] = 0;
        for ( int i = 1 ; i <= n ; i++ ) {
            cin >> s[i];
            sum[i] = sum[i-1] + strlen(s[i]);
            dp[i] = numeric_limits<long double>::max();
        }
        stk[top] = (INV) {1, n + 1, 0};
        for ( int i = 1 ; i <= n ; i++ ) {
            if ( i >= stk[bot].R ) bot++;
            dp[i] = f(i, stk[bot].pos);
            update(i);
            // cout << (ll) f(i, stk[bot].pos) << endl;
        }
        if ( dp[n] > 1e18 ) {
            cout << "Too hard to arrange" << endl;
        } else {
            vector<PI> as;
            cout << (ll)dp[n] << endl;
        }
    } return 0;
}
```

3 Graph

3.1 Dijkstra

```
// Queries mininum path from src to dest in a graph where src
     and dest are connected.
// The n vertices are supposed to be marked [0, n); edge
     should have size n.
int dijkstra(int src, int dest, const vector<vector<pii>>&
     edge) {
    const int N = edge.size();
    bool nvis[N] = {0};
    // A comparator may be required.
    priority_queue<pii, vector<pii>, greater<pii>> q;
    q.emplace(0, src);
    while (!q.empty()) {
        int v = q.top().second;
        int d = q.top().first;
        q.pop();
        if (v == dest) return d;
        if (nvis[v]) continue;
        nvis[v] = true;
        for (auto& e : edge[v]) {
            if (!nvis[e.second]) {
                // Fit the comparator
                q.emplace(d + e.first, e.second);
            }
        }
    }
    throw "src and dest are not connected.";
}

// Queries minuimum path from src to all the other vertices
     in a graph where all vertices are connected.
// The n vertices are supposed to be marked [0, n); edge
     should have size n.
vector<int> dijkstra(int src, const vector<vector<pii>>& edge
     ) {
    const int N = edge.size();
    vector<int> mindist(N, -1);
    int nvis = 0;
    // A comparator may be required.
    priority_queue<pii, vector<pii>, greater<pii>> q;
    q.emplace(0, src);
    while (nvis < N) {
        if (q.empty()) throw "Not all vertices connected.";
        int v = q.top().second;
        int d = q.top().first;
        q.pop();
        if (mindist[v] != -1) continue;
        mindist[v] = d;
        nvis++;
        for (auto& e : edge[v]) {
```

```
            if (mindist[e.second] == -1) {
                // Fit the comparator
                q.emplace(d + e.first, e.second);
            }
        }
    }
    return mindist;
}
```

3.2 Bellman Ford

```
#include<vector>
#include<iostream>
using namespace std;
#define maxn 2000
#define INF 100000000
typedef pair<int,int> pii;
vector<pii> G[maxn];
int dis[maxn];
bool BellmanFord(int n,int s)
{
    for(int i=1; i<=n; i++)
        dis[i] = INF;
    dis[s] = 0;
    bool relax;
    for(int r=1; r<=n; r++) //O(VE)
    {
        relax = false;
        for(int i=1; i<=n; i++)
            for(pii e:G[i])
                if( dis[i] + e.second < dis[e.first] )
                    dis[e.first] = dis[i] + e.second, relax =
                        true;
    }
    return relax; //有負環
}
int main()
{
    int n,m; cin >> n >> m;
    for(int i=0; i<m; i++)
    {
        int a,b,w;
        cin >> a >> b >> w;
        G[a].push_back(pii(b,w));
    }
    int s,t;
    cin >> s >> t;
    if(BellmanFord(n,s)) cout << "There is a minus-cycle.\n";
    else cout << "dis = " << dis[t] << endl;
    return 0;
}
```

3.3 Floyd Warshall

```
#include<iostream>
#include<vector>
using namespace std;
#define maxn 505
#define INF 10000000
typedef pair<int,int> pii;
vector<pii> G[maxn];
int dist[maxn][maxn];
void FloydWarshalll(int n)
{
    for(int i=1; i<=n; i++)
        for(int j=1; j<=n; j++)
            dist[i][j] = i==j?0:INF;
    for(int i=1; i<=n; i++)
        for(pii j:G[i])
            dist[i][j.first] = j.second;
    for(int k=1; k<=n; k++) //DP //O(n^3)
        for(int i=1; i<=n; i++)
            for(int j=1; j<=n; j++)
                dist[i][j] = min(dist[i][j],dist[i][k]+dist[k
                    ][j]);
}
int main()
{
    int n,m,q;
    cin >> n >> m >> q;
    for(int i=0; i<m; i++)
    {
        int u,v,d;
        cin >> u >> v >> d;
        G[u].push_back(pii(v,d));
    }
    FloydWarshalll(n);
    while(q--)
    {
        int u,v;
        cin >> u >> v;
        if(dist[u][v]==INF)printf("(%d,%d) not connected\n",u
            ,v);
        printf("Dist(%d,%d)=%d\n",u,v,dist[u][v]);
    }
    return 0;
}
```

3.4 SPFA

```
#include<vector>
#include<queue>
#include<iostream>
using namespace std;
#define maxn 2000
#define INF 100000000
typedef pair<int,int> pii;
vector<pii> G[maxn];
int dis[maxn];
void SPFA(int n,int s) //O(kE) k~2.
{
    for(int i=1; i<=n; i++)
        dis[i] = INF;
    dis[s] = 0;
    queue<int> q;
    q.push(s);
    bool inque[maxn] = {};
    while(!q.empty())
    {
        int u = q.front(); q.pop();
        inque[u] = false;
        for(pii e:G[u])
        {
            int v = e.first , w = e.second;
            if( dis[u] + w < dis[v])
            {
                if(!inque[v]) q.push(v), inque[v] = true;
                dis[v] = dis[u] + w;
            }
        }
    }
}
int main()
{
    int n,m; cin >> n >> m;
    for(int i=0; i<m; i++)
    {
        int a,b,w;
        cin >> a >> b >> w;
        G[a].push_back(pii(b,w));
    }
    int s,t;
    cin >> s >> t;
    SPFA(n,s);
    if(dis[t]==INF) cout << "dis = INF\n";
    else cout << "dis = " << dis[t] << endl;
    return 0;
}
```

3.5 Kruskal

```
#include<iostream>
#include<algorithm>
using namespace std;
typedef pair<int,int> pii;
typedef pair<int,pii> piii;
#define w first
#define x second.first
#define y second.second
#define maxm 100000
#define maxn 10000
struct UFT //Union-Find Tree
{
    int sz;
    int p[maxn];
    UFT(int _sz) {
        sz = _sz;
        for(int i=1; i<=sz; i++)
            p[i] = i;
    }
    inline int par(int a) { //parent
        return p[a] = ( a==p[a] ? a : par(p[a]));
    }
    inline bool same(int a, int b) { //check if a and b are
         in the same set
        return par(a) == par(b);
    }
    inline void uni(int a, int b) { //union the sets of a and
         b
        p[p[a]] = p[b];
    }
};
int main()
{
    piii e[maxm];
```

```
    int n,m; cin >> n >> m;
    for(int i=0; i<m; i++)
        cin >> e[i].x >> e[i].y >> e[i].w;
    UFT uft(n); sort(e,e+m); //sort the edges
    int cnt = 0, cost = 0;
    for(int i=0; i<m && cnt<n-1; i++)
    {
        if( uft.same(e[i].x,e[i].y) ) continue;
        uft.uni(e[i].x,e[i].y);
        cnt++; cost += e[i].w;
    }
    if(cnt<n-1) cout << "Not connected!\n";
    else cout << "Cost = " << cost << endl;
    return 0;
}
```

3.6 Prim

```
// Queries minimum path of spanning tree of a graph, where all vertices are connected, using Prim's algorithm.
// The n vertices are supposed to be marked [0, n); edge should have size n.
int minpath(const vector<vector<pii>>& edge) {
    const int N = edge.size();
    bool vis[N] = {0};
    priority_queue<pii, vector<pii>, greater<pii>> q;

    vis[0] = true;
    int nvis = 1;
    for (auto& e : edge[0]) q.push(e);

    while (nvis < N) {
        int d = q.top().first;
        int v = q.top().second;
        q.pop();
        if (vis[v]) continue;
        cout << d << endl;
        vis[v] = true;
        if (++nvis == N) return d;
        for (auto& e : edge[v]) {
            if (!vis[e.second]) q.emplace(d + e.first, e.second);
        }
    }
    throw "Never reaches here.";
}
```

3.7 LCA

```
//LCA //Tarjan's algorithm
#define maxv 100
int LCA[maxv][maxv]; //lowest common ancestor
vector<int> v[maxv]; //adjacency lists
int p[maxv]; //parent
bool visit[maxv]; //false
int n; //the number of vertex
void init() { for(int i=0; i<n; i++) p[i]=i; }
int parent(int x) { return (p[x]==x) ? (p[x]=parent(x)) : x; }
int dfs(int u) {
    visit[u] = true;
    for(int i=0; i<n; i++)
        if(visit[i]) LCA[i][u] = LCA[u][i] = parent(i);
    for(int i=v[u].size()-1; i>=0; i--) {
        int uu = v[u][i];
        if(!visit[uu]) { dfs(uu); p[uu] = u; }
    }
}
void Tarjan() { init(); dfs(0); }
```

3.8 Tarjan

```
割點
點 u 為割點 if and only if 滿足 1. or 2.
1. u 爲樹根，且 u 有多於一個子樹。
2. u 不爲樹根，且滿足存在 (u,v) 爲樹枝邊 (或稱父子邊，即 u 爲 v 在搜索樹中的父親)，使得 DFN(u) <= Low(v)。
------------------------------------------------------
橋
一條無向邊 (u,v) 是橋 if and only if (u,v) 爲樹枝邊，且滿足 DFN(u) < Low(v)。

// 0 base
struct TarjanSCC{
    static const int MAXN = 1000006;
    int n, dfn[MAXN], low[MAXN], scc[MAXN], scn, count;
    vector<int> G[MAXN];
    stack<int> stk;
    bool ins[MAXN];
    void tarjan(int u) {
        dfn[u] = low[u] = ++count;
        stk.push(u);
        ins[u] = true;
        for(auto v:G[u]) {
            if(!dfn[v]) {
                tarjan(v);
                low[u] = min(low[u], low[v]);
            } else if(ins[v]) {
                low[u] = min(low[u], dfn[v]);
            }
        }
        if(dfn[u] == low[u]) {
            int v;
            do {
            v = stk.top(); stk.pop();
            scc[v] = scn;
            ins[v] = false;
            } while(v != u);
            scn++;
        }
    }
    void getSCC(){
        memset(dfn,0,sizeof(dfn));
        memset(low,0,sizeof(low));
        memset(ins,0,sizeof(ins));
        memset(scc,0,sizeof(scc));
        count = scn = 0;
        for(int i = 0 ; i < n ; i++ )
            if(!dfn[i]) tarjan(i);
    }
} SCC;
```

3.9 Min Mean Cycle

```
#include<cfloat> //for DBL_MAX
int dp[MAXN][MAXN]; // 1-base,O(NM)
vector<tuple<int,int,int>> edge;
double mmc(int n){ //allow negative weight
    const int INF = 0x3f3f3f3f;
    for(int t=0; t<n; ++t){
        memset(dp[t+1],0x3f,sizeof(dp[t+1]));
        for(const auto &e:edge) {
            int u,v,w;
            tie(u,v,w) = e;
            dp[t+1][v] = min(dp[t+1][v],dp[t][u]+w);
        }
    }
    double res = DBL_MAX;
    for(int u=1; u<=n; ++u) {
        if(dp[n][u]==INF) continue;
        double val = -DBL_MAX;
        for(int t=0;t<n;++t)
            val = max(val,(dp[n][u]-dp[t][u])*1.0/(n-t));
        res = min(res,val);
    }
    return res;
}
```

3.10 2-SAT

```
const int MAXN = 2020;
struct TwoSAT{
    static const int MAXv = 2*MAXN;
    vector<int> GO[MAXv],BK[MAXv],stk;
    bool vis[MAXv];
    int SC[MAXv];
    void imply(int u,int v){ // u imply v
        GO[u].push_back(v);
        BK[v].push_back(u);
    }
    int dfs(int u,vector<int>*G,int sc){
        vis[u]=1, SC[u]=sc;
        for (int v:G[u])if (!vis[v])
            dfs(v,G,sc);
        if  (G==GO)  stk.push_back(u);
    }
    int scc(int n=MAXv){
        memset(vis,0,sizeof(vis));
        for (int i=0; i<n; i++)
            if (!vis[i])  dfs(i,GO,-1);
        memset(vis,0,sizeof(vis));
        int sc=0;
        while (!stk.empty()){
            if (!vis[stk.back()])
                dfs(stk.back(),BK,sc++);
            stk.pop_back();
        }
    }
}  SAT;
int main(){
    SAT.scc(2*n);
    bool ok  =  1;
    for (int i=0; i<n; i++){
        if (SAT.SC[2*i]==SAT.SC[2*i+1])  ok  =  0;
```

```
    }
    if (ok) {
        for (int i=0; i<n; i++)
            if (SAT.SC[2*i]>SAT.SC[2*i+1])
                cout << i << endl;
    }
    else puts("NO");
}
void warshall(){
    bitset<2003> d[2003];
    for (int k=0; k<n; k++)
        for (int i=0; i<n; i++)
            if (d[i][k]) d[i] |= d[k];
}
```

3.11 生成樹數量

```
// D : degree-matrix
// A : adjacent-matrix
// 無向圖
    // (u,v)
    // A[u][v]++, A[v][u]++
    // D[u][u]++, D[v][v]++
    // G = D-A
    // abs(det(G去掉i-col和i-row))
    // 生成樹的數量
// 有向圖
    // A[u][v]++
    // D[v][v]++ (in-deg)
    // 以i為root的樹形圖數量
    // 所有節點都能到達root
```

3.12 BCC_edge

```
邊雙連通
任意兩點間至少有兩條不重疊的路徑連接，找法：
1. 標記出所有的橋
2. 對全圖進行 DFS，不走橋，每一次 DFS 就是一個新的邊雙連通
// from BCW
struct BccEdge {
  static const int MXN = 100005;
  struct Edge { int v,eid; };
  int n,m,step,par[MXN],dfn[MXN],low[MXN];
  vector<Edge> E[MXN];
  DisjointSet djs;
  void init(int _n) {
    n = _n; m = 0;
    for (int i=0; i<n; i++) E[i].clear();
    djs.init(n);
  }
  void add_edge(int u, int v) {
    E[u].PB({v, m});
    E[v].PB({u, m});
    m++;
  }
  void DFS(int u, int f, int f_eid) {
    par[u] = f;
    dfn[u] = low[u] = step++;
    for (auto it:E[u]) {
      if (it.eid == f_eid) continue;
      int v = it.v;
      if (dfn[v] == -1) {
        DFS(v, u, it.eid);
        low[u] = min(low[u], low[v]);
      } else {
        low[u] = min(low[u], dfn[v]);
      }
    }
  }
  void solve() {
    step = 0;
    memset(dfn, -1, sizeof(int)*n);
    for (int i=0; i<n; i++) {
      if (dfn[i] == -1) DFS(i, i, -1);
    }
    djs.init(n);
    for (int i=0; i<n; i++) {
      if (low[i] < dfn[i]) djs.uni(i, par[i]);
    }
  }
} graph;
```

4 Flow_Matching

4.1 Dinic

```
const int maxn = 1e5 + 10;
const int INF = 1e9;
const long long INF64 = 1e18;
struct edge{
    int to, cap, rev;
};
vector<edge> G[maxn];
int n, m, s, t, a, b, c, iter[maxn], level[maxn];
void bfs(int s) {
    memset(level, -1, sizeof(level));
    queue<int> q;
    level[s] = 0;
    q.push(s);
    while (q.size()) {
        int u = q.front(); q.pop();
        for (edge e: G[u]) {
            if (e.cap > 0 && level[e.to] < 0) {
                level[e.to] = level[u] + 1;
                q.push(e.to);
            }
        }
    }
}
int dfs(int v, int t, int f) {
    if (v == t) return f;
    for (int &i = iter[v]; i < G[v].size(); i++) {
        edge &e = G[v][i];
        if (e.cap > 0 && level[v] < level[e.to]) {
            int d = dfs(e.to, t, min(f, e.cap));
            if (d > 0) {
                e.cap -= d;
                G[e.to][e.rev].cap += d;
                return d;
            }
        }
    }
    return 0;
}
int dinic(int s, int t) {
    int flow = 0;
    while (true) {
        bfs(s);
        if (level[t] < 0) return flow;
        memset(iter, 0, sizeof(iter));
        int f;
        while ((f = dfs(s, t, INF)) > 0)
            flow += f;
    }
}
void init(int n) {
    for (int i = 0; i < n; i++) G[i].clear();
}
int main() {
    cin >> n >> m >> s >> t;
    init(n);
    while (m--) {
        cin >> a >> b >> c;
        G[a].push_back((edge){b, c, (int)G[b].size()});
        G[b].push_back((edge){a, 0, (int)G[a].size() - 1});
    }
    cout << dinic(s, t) << '\n';
    return 0;
}
```

4.2 KM

```
#define MAXN 405
#define INF 0x3f3f3f3f3f3f3f3f
int n;// 1-base，0表示沒有匹配
LL g[MAXN][MAXN]; //input graph
int My[MAXN],Mx[MAXN]; //output match
LL lx[MAXN],ly[MAXN],pa[MAXN],Sy[MAXN];
bool vx[MAXN],vy[MAXN];
void augment(int y){
    for(int x, z; y; y = z) {
        x = pa[y], z = Mx[x];
        My[y] = x, Mx[x] = y;
    }
}
void bfs(int st){
    for(int i=1; i<=n; ++i)
        Sy[i] = INF, vx[i]=vy[i]=0;
    queue<int> q; q.push(st);
    while(1) {
        while(q.size()) {
            int x=q.front(); q.pop();
            vx[x]=1;
            for(int y=1; y<=n; ++y) if(!vy[y]){
                LL t = lx[x]+ly[y]-g[x][y];
                if(t==0) {
                    pa[y]=x;
                    if(!My[y]) {augment(y);return;}
                    vy[y]=1,q.push(My[y]);
                }
                else if(Sy[y]>t) pa[y]=x,Sy[y]=t;
            }
        }
        LL cut = INF;
        for(int y=1; y<=n; ++y) if(!vy[y])
            cut = min(cut, Sy[y]);
        for(int j=1; j<=n; ++j) {
            if(vx[j]) lx[j] = cut;
            if(vy[j]) ly[j] += cut;
            else Sy[j] -= cut;
        }
        for(int y=1; y<=n; ++y) if(!vy[y]){
            if(Sy[y]==0) {
                if(!My[y]) {augment(y);return;}
                vy[y]=1,q.push(My[y]);
            }
        }
    }
}
int main() {
    cin >> n;
    for(int x=1; x<=n; ++x) for(int y=1; y<=n; ++y)
        cin >> g[x][y];
    for(int x=1; x<=n; ++x) {
        lx[x] = -INF;
        for(int y=1; y<=n; ++y)
            lx[x] = max(lx[x], g[x][y]);
    }
    for(int x=1; x<=n; ++x) bfs(x);
    LL ans = 0;
    for(int y=1; y<=n; ++y) ans += g[My[y]][y];
    cout << ans << endl;
    for(int x=1; x<=n; ++x) cout << (x==1?"":" ") << My[x];
    cout << endl;
    return 0;
}
```

```
        }
        LL cut = INF;
        for(int y=1; y<=n; ++y)
            if(!vy[y]&&cut>Sy[y]) cut=Sy[y];
        for(int j=1; j<=n; ++j) {
            if(vx[j]) lx[j] -= cut;
            if(vy[j]) ly[j] += cut;
            else Sy[j] -= cut;
        }
        for(int y=1; y<=n; ++y) {
            if(!vy[y]&&Sy[y]==0) {
                if(!My[y]){augment(y);return;}
                vy[y]=1, q.push(My[y]);
            }
        }
    }
}
LL KM() {
    memset(My,0,sizeof(int)*(n+1));
    memset(Mx,0,sizeof(int)*(n+1));
    memset(ly,0,sizeof(LL)*(n+1));
    for(int x=1; x<=n; ++x){
        lx[x] = -INF;
        for(int y=1; y<=n; ++y)
            lx[x] = max(lx[x],g[x][y]);
    }
    for(int x=1; x<=n; ++x) bfs(x);
    LL ans = 0;
    for(int y=1; y<=n; ++y) ans += g[My[y]][y];
    return ans;
}
```

4.3 Ford Fulkerson

```
const int maxn = 1e5 + 10, INF = 1e9;
const long long INF64 = 1e18;
struct edge{
    int to, cap, rev;
};
vector<edge> G[maxn];
int n, m, s, t, a, b, c;
bool vis[maxn];
int dfs(int v, int t, int f) {
    cout << v << ' ' << t << ' ' << f << '\n';
    if (v == t) return f;
    vis[v] = true;
    for (edge &e: G[v]) {
        if (!vis[e.to] && e.cap > 0) {
            int d = dfs(e.to, t, min(f, e.cap));
            if (d > 0) {
                e.cap -= d, G[e.to][e.rev].cap += d;
                return d;
            }
        }
    }
    return 0;
}
int ford_fulkerson(int s, int t) {
    int flow = 0, f;
    for (int i = 0; i < n; i++) {
        cout << i << " : ";
        for (edge e: G[i])
            cout << '(' << e.to << ',' << e.cap << ')' << ' '
                 ;
        cout << '\n';
    }
    do {
        memset(vis, false, sizeof(vis));
        f = dfs(s, t, INF);
        for (int i = 0; i < n; i++) {
            cout << i << " : ";
            for (edge e: G[i])
                cout << '(' << e.to << ',' << e.cap << ')' <<
                     ' ';
            cout << '\n';
        }
        cout << f << '\n';
        flow += f;
    } while (f > 0);
    return flow;
}
void init(int n) {
    for (int i = 0; i < n; i++) G[i].clear();
}
int main() {
    cin >> n >> m >> s >> t;
    init(n);
    while (m--) {
        cin >> a >> b >> c;
        G[a].push_back((edge){b, c, (int)G[b].size()});
        G[b].push_back((edge){a, 0, (int)G[a].size() - 1});
    }
    cout << ford_fulkerson(s, t) << '\n';
    return 0;
}
```

4.4 Min Cost Max Flow

```
template<typename TP>
struct MCMF{
    static const int MAXN=440;
    static const TP INF=999999999;
    struct edge{
        int v,pre;
        TP r,cost;
        edge(int v,int pre,TP r,TP cost):v(v),pre(pre),r(r),
             cost(cost){}
    };
    int n,S,T;
    TP dis[MAXN],PIS,ans;
    bool vis[MAXN];
    vector<edge> e;
    int g[MAXN];
    void init(int _n){
        memset(g,-1,sizeof(int)*((n=_n)+1));
        e.clear();
    }
    void add_edge(int u,int v,TP r,TP cost,bool directed=
         false){
        e.push_back(edge(v,g[u],r,cost));
        g[u]=e.size()-1;
        e.push_back(
        edge(u,g[v],directed?0:r,-cost));
        g[v]=e.size()-1;
    }
    TP augment(int u,TP CF){
        if(u==T||!CF)return ans+=PIS*CF,CF;
        vis[u]=1;
        TP r=CF,d;
        for(int i=g[u];~i;i=e[i].pre){
            if(e[i].r&&!e[i].cost&&!vis[e[i].v]){
                d=augment(e[i].v,min(r,e[i].r));
                e[i].r-=d;
                e[i^1].r+=d;
                if(!(r-=d))break;
            }
        }
        return CF-r;
    }
    bool modlabel(){
        for(int u=0;u<=n;++u)dis[u]=INF;
        static deque<int>q;
        dis[T]=0,q.push_back(T);
        while(q.size()){
            int u=q.front();q.pop_front();
            TP dt;
            for(int i=g[u];~i;i=e[i].pre){
                if(e[i^1].r&&(dt=dis[u]-e[i].cost)<dis[e[i].v
                     ]){
                    if((dis[e[i].v]=dt)<=dis[q.size()?q.front
                         ():S]){
                        q.push_front(e[i].v);
                    }else q.push_back(e[i].v);
                }
            }
        }
        for(int u=0;u<=n;++u)
            for(int i=g[u];~i;i=e[i].pre)
                e[i].cost+=dis[e[i].v]-dis[u];
        return PIS+=dis[S], dis[S]<INF;
    }
    TP mincost(int s,int t){
        S=s,T=t;
        PIS=ans=0;
        while(modlabel()){
            do memset(vis,0,sizeof(bool)*(n+1));
            while(augment(S,INF));
        }return ans;
    }
};
```

4.5 Hopcroft Karp

```
// https://github.com/voidrank/acm-icpc-library/blob/master/
    code/hopcroft-karp.cpp
int n, m, vis[maxn], level[maxn], pr[maxn], pr2[maxn];
vector<int> edge[maxn]; // for Left
bool dfs(int u) {
    vis[u] = true;
    for (vector<int>::iterator it = edge[u].begin(); it !=
         edge[u].end(); ++it) {
        int v = pr2[*it];
        if (v == -1 || (!vis[v] && level[u] < level[v] && dfs
             (v))) {
            pr[u] = *it, pr2[*it] = u;
            return true;
        }
    } return false;
```

```
}
int hopcroftKarp() {
    memset(pr, -1, sizeof(pr)); memset(pr2, -1, sizeof(pr2));
    for (int match = 0; ;) {
        queue<int> Q;
        for (int i = 1; i <= n; ++i) {
            if (pr[i] == -1) {
                level[i] = 0;
                Q.push(i);
            } else level[i] = -1;
        }
        while (!Q.empty()) {
            int u = Q.front(); Q.pop();
            for (vector<int>::iterator it = edge[u].begin();
                 it != edge[u].end(); ++it) {
                int v = pr2[*it];
                if (v != -1 && level[v] < 0) {
                    level[v] = level[u] + 1;
                    Q.push(v);
                }
            }
        }
        for (int i = 1; i <= n; ++i) vis[i] = false;
        int d = 0;
        for (int i = 1; i <= n; ++i) if (pr[i] == -1 && dfs(i
             )) ++d;
        if (d == 0) return match;
        match += d;
    }
}
```

4.6 SW-MinCut

```
// all pair min cut
// global min cut
struct SW { // O(V^3)
    static const int MXN = 514;
    int n, vst[MXN], del[MXN];
    int edge[MXN][MXN], wei[MXN];
    void init(int _n){
        n = _n; FZ(edge); FZ(del);
    }
    void addEdge(int u, int v, int w) {
        edge[u][v] += w; edge[v][u] += w;
    }
    void search(int &s, int &t) {
        FZ(vst); FZ(wei);
        s = t = -1;
        while (true){
            int mx=-1, cur=0;
            for (int i=0; i<n; i++)
                if (!del[i] && !vst[i] && mx<wei[i])
                    cur = i, mx = wei[i];
            if (mx == -1) break;
            vst[cur] = 1;
            s = t; t = cur;
            for (int i=0; i<n; i++)
                if (!vst[i] && !del[i]) wei[i] += edge[cur][i
                    ];
        }
    }
    int solve() {
        int res = 2147483647;
        for (int i=0,x,y; i<n-1; i++) {
            search(x,y);
            res = min(res,wei[y]);
            del[y] = 1;
            for (int j=0; j<n; j++)
                edge[x][j] = (edge[j][x] += edge[y][j]);
        }
        return res;
    }
} graph;
```

4.7 Stable Marriage

```
// 演算法筆記
1. N位男士各自向自己最喜愛的女士求婚。
2. N位女士各自從自己的求婚者中，挑最喜愛的那位男士訂婚，但是
    往後可背約。
   沒有求婚者的女士，就只好等等。
3. 失敗的男士們，只好各自向自己次喜愛的女士求婚。
4. N位女士各自從自己的求婚者中，挑最喜歡的那位男士訂婚，但是
    往後可背約。
   已訂婚卻有更喜愛的男士求婚的女士，就毀約，改為與此男士訂
    婚。
   沒有求婚者的女士，就只好再等等。
5. 重複3. 4.直到形成N對伴侶為止。
// Jinkela
queue<int> Q;
for ( i : 所有考生 ) {
    設定在第0志願;
    Q.push(考生i);
}
while(Q.size()){
    當前考生=Q.front();Q.pop();
    while ( 此考生未分發 ) {
        指標移到下一志願;
        if ( 已經沒有志願 or 超出志願總數 ) break;
        計算該考生在該科系加權後的總分;
        if ( 不符合科系需求 ) continue;
        if ( 目前科系有餘額 ) {
            依加權後分數高低順序將考生id加入科系錄取名單中;
            break;
        }
        if ( 目前科系已額滿 ) {
            if ( 此考生成績比最低分數還高 ) {
                依加權後分數高低順序將考生id加入科系錄取名單;
                Q.push(被踢出的考生);
            }
        }
    }
}
```

5 Math

5.1 Matrix

```
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;
  }
};
```

5.2 模逆元

```
// Queries value x such that (ax == 1) mod m.
ll modinv(ll a, ll m) {
    if (m == 1) return 0;
    ll m0 = m, y = 0, x = 1;
    while (a > 1) {
        ll q = a / m;
        ll t = m;
        m = a % m, a = t;
        t = y;
        y = x - q * y;
        x = t;
    }
    if (x < 0) x += m0;
    return x;
}
```

5.3 Euler Function

```
// Queries phi(x) value.
int phi(int x) {
    int ret = x;
    for (int p = 2; p * p <= x; p++) {
        if (x % p == 0) {
            while (x % p == 0) x /= p;
            ret -= ret / p;
        }
    }
    if (x > 1) ret -= ret / x;
    return ret;
}

// Queries all phi(x) values where x in [0, n).
vector<int> phi_in(int n) {
    vector<bool> prime(n, 1);
    vector<int> ret(n);
    prime[0] = prime[1] = false;
    for (int i = 0; i < n; i++) ret[i] = i;
    for (int i = 2; i < n; i++) {
        if (!prime[i]) continue;
        ret[i]--;
        for (int j = i * 2; j < n; j += i) {
            prime[j] = false;
            ret[j] = ret[j] / i * (i - 1);
        }
    }
    ret[1] = 0;
    return ret;
}
```

5.4 Miller Rabin

```
//From jacky860226
typedef long long LL;
inline LL mul(LL a,LL b,LL m){//a*b%m
    return (a%m)*(b%m)%m;
}
/*LL mul(LL a,LL b,LL m){//a*b%m
    a %= m, b %= m;
    LL y = (LL)((double)a*b/m+0.5); //fast for m < 2^58
    LL r = (a*b-y*m)%m;
    return r<0 ? r+m : r;
}*/
template<typename T> T pow(T a,T b,T mod) //a^b%mod
{
    T ans = 1;
    while(b)
    {
        if(b&1) ans = mul(ans,a,mod);
        a = mul(a,a,mod);
        b >>= 1;
    }
    return ans;
}
template<typename T> bool isprime(T n, int num) //num = 3,7
{
    int sprp[3] = {2,7,61}; //int範圍可解
    //int llsprp[7] =
        {2,325,9375,28178,450775,9780504,1795265022}; //至少
        unsigned long long範圍
    if(n==2) return true;
    if(n<2 || n%2==0) return false;
    //n-1 = u * 2^t
    int t = 0;
    T u = n-1;
    while(u%2==0) u >>= 1, t++;
    for(int i=0; i<num; i++)
    {
        T a = sprp[i]%n;
        if(a==0 || a==1 || a==n-1) continue;
        T x = pow(a,u,n);
        if(x==1 || x==n-1) continue;
        for(int j=1; j<t; j++)
        {
            x = mul(x,x,n);
            if(x==1) return false;
            if(x==n-1) break;
        }
        if(x!=n-1) return false;
    }
    return true;
}
```

5.5 質因數分解

```
LL func(const LL n,const LL mod,const int c) {
  return (LLmul(n,n,mod)+c+mod)%mod;
}

LL pollorrho(const LL n, const int c) {//循環節長度
  LL a=1, b=1;
  a=func(a,n,c)%n;
  b=func(b,n,c)%n; b=func(b,n,c)%n;
  while(gcd(abs(a-b),n)==1) {
    a=func(a,n,c)%n;
    b=func(b,n,c)%n; b=func(b,n,c)%n;
  }
  return gcd(abs(a-b),n);
}

void prefactor(LL &n, vector<LL> &v) {
  for(int i=0;i<12;++i) {
    while(n%prime[i]==0) {
      v.push_back(prime[i]);
      n/=prime[i];
    }
  }
}

void smallfactor(LL n, vector<LL> &v) {
  if(n<MAXPRIME) {
    while(isp[(int)n]) {
      v.push_back(isp[(int)n]);
      n/=isp[(int)n];
    }
    v.push_back(n);
  } else {
    for(int i=0;i<primecnt&&prime[i]*prime[i]<=n;++i) {
      while(n%prime[i]==0) {
        v.push_back(prime[i]);
        n/=prime[i];
      }
    }
    if(n!=1) v.push_back(n);
  }
}

void comfactor(const LL &n, vector<LL> &v) {
  if(n<1e9) {
    smallfactor(n,v);
    return;
  }
  if(Isprime(n)) {
    v.push_back(n);
    return;
  }
  LL d;
  for(int c=3;;++c) {
    d = pollorrho(n,c);
    if(d!=n) break;
  }
  comfactor(d,v);
  comfactor(n/d,v);
}

void Factor(const LL &x, vector<LL> &v) {
  LL n = x;
  if(n==1) { puts("Factor 1"); return; }
  prefactor(n,v);
  if(n==1) return;
```

```
  comfactor(n,v);
  sort(v.begin(),v.end());
}

void AllFactor(const LL &n,vector<LL> &v) {
  vector<LL> tmp;
  Factor(n,tmp);
  v.clear();
  v.push_back(1);
  int len;
  LL now=1;
  for(int i=0;i<tmp.size();++i) {
    if(i==0 || tmp[i]!=tmp[i-1]) {
      len = v.size();
      now = 1;
    }
    now*=tmp[i];
    for(int j=0;j<len;++j)
      v.push_back(v[j]*now);
  }
}
```

5.6 快速冪

```
// Queries a ^ p.
ll fastpow(ll a, int p) {
    ll ret = 1;
    while (p) {
        if (p & 1) ret *= a;
        a *= a, p >>= 1;
    }
    return ret;
}

// Queries (a ^ p) mod m.
ll fastpow(ll a, ll p, ll m) {
    ll ret = 1;
    while (p) {
        if (p & 1) ret = ret * a % m;
        a = a * a % m, p >>= 1;
    }
    return ret;
}
```

5.7 實根

```
// 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
}
```

5.8 SG

```
Anti Nim (取走最後一個石子者敗) :
先手必勝 if and only if
1. 「所有」堆的石子數都為 1 且遊戲的 SG 值為 0。
2. 「有些」堆的石子數大於 1 且遊戲的 SG 值不為 0。
-----------------------------------------------------
Anti-SG (決策集合為空的遊戲者贏) :
定義 SG 值為 0 時，遊戲結束。
則先手必勝 if and only if
1. 遊戲中沒有單一遊戲的 SG 函數大於 1 且遊戲的 SG 函數為 0。
2. 遊戲中某個單一遊戲的 SG 函數大於 1 且遊戲的 SG 函數不為 0
    。
-----------------------------------------------------
Sprague-Grundy :
1. 雙人、回合制
2. 資訊完全公開
3. 無隨機因素
4. 可在有限步內結束
5. 沒有和局
6. 雙方可採取的行動相同

SG(S) 的值為 0：後手(P)必勝
不為 0：先手(N)必勝
int mex(set S) {
  // find the min number >= 0 that not in the S
  // e.g. S = {0, 1, 3, 4} mex(S) = 2
}
state = []
int SG(A) {
  if (A not in state) {
    S = sub_states(A)
    if( len(S) > 1 ) state[A] = reduce(operator.xor, [SG(B)
        for B in S])
    else state[A] = mex(set(SG(B) for B in next_states(A)))
  } return state[A]
}
```

5.9 外星模運算

```
//a[0]^(a[1]^a[2]^...)
#define maxn 1000000
int euler[maxn+5];
bool is_prime[maxn+5];
void init_euler(){
  is_prime[1]=1;//一不是質數
  for(int i=1;i<=maxn;i++)euler[i]=i;
  for(int i=2;i<=maxn;i++){
    if(!is_prime[i]){//是質數
      euler[i]--;
      for(int j=i<<1;j<=maxn;j+=i){
        is_prime[j]=1;
        euler[j]=euler[j]/i*(i-1);
      }
    }
  }
}
LL pow(LL a,LL b,LL mod){//a^b%mod
  LL ans=1;
  for(;b;a=a*a%mod,b>>=1)
    if(b&1)ans=ans*a%mod;
  return ans;
}
bool isless(LL *a,int n,int k){
  if(*a==1)return k>1;
  if(--n==0)return *a<k;
  int next=0;
  for(LL b=1;b<k;++next)
    b*=*a;
  return isless(a+1,n,next);
}
LL high_pow(LL *a,int n,LL mod){
  if(*a==1||--n==0)return *a%mod;
  int k=0,r=euler[mod];
  for(LL tma=1;tma!=pow(*a,k+r,mod);++k)
    tma=tma*(*a)%mod;
  if(isless(a+1,n,k))return pow(*a,high_pow(a+1,n,k),mod);
  int tmd=high_pow(a+1,n,r), t=(tmd-k+r)%r;
  return pow(*a,k+t,mod);
}
LL a[1000005];
int t,mod;
int main(){
  init_euler();
  scanf("%d",&t);
```

```
  #define n 4
  while(t--){
    for(int i=0;i<n;++i)scanf("%lld",&a[i]);
    scanf("%d",&mod);
    printf("%lld\n",high_pow(a,n,mod));
  }
  return 0;
}
```

5.10 FFT

```
template<typename T,typename VT=vector<complex<T> > >
struct FFT{
    const T pi;
    FFT(const T pi=acos((T)-1)):pi(pi){}
    unsigned bit_reverse(unsigned a,int len){
        a=((a&0x55555555U)<<1)|((a&0xAAAAAAAAU)>>1);
        a=((a&0x33333333U)<<2)|((a&0xCCCCCCCCU)>>2);
        a=((a&0x0F0F0F0FU)<<4)|((a&0xF0F0F0F0U)>>4);
        a=((a&0x00FF00FFU)<<8)|((a&0xFF00FF00U)>>8);
        a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)>>16);
        return a>>(32-len);
    }
    void fft(bool is_inv,VT &in,VT &out,int N){
        int bitlen=__lg(N),num=is_inv?-1:1;
        for(int i=0;i<N;++i) out[bit_reverse(i,bitlen)]=in[i
            ];
        for(int step=2;step<=N;step<<=1){
            const int mh=step>>1;
            for(int i=0;i<mh;++i){
                complex<T> wi=exp(complex<T>(0,i*num*pi/mh));
                for(int j=i;j<N;j+=step){
                    int k=j+mh;
                    complex<T> u=out[j], t=wi*out[k];
                    out[j]=u+t;
                    out[k]=u-t;
                }
            }
        }
        if(is_inv) for(int i=0;i<N;++i) out[i]/=N;
    }
};
```

6 String

6.1 Rolling Hash

```
// Queires the first index where sub appears in str; -1
    indicates no matching.
int rollhash(string& str, string& sub) {
    const int X = 1e6 + 99, MOD = 1e9 + 9;
    ll xx = 1;
    for (int i = 0; i < sub.size(); i++) xx = xx * X % MOD;
    ll subhash = 0;
    for (char c : sub) subhash = (subhash * X + c) % MOD;

    vector<ll> hash = {0};
    for (int i = 0; i < str.size(); i++) {
        hash.push_back((hash.back() * X + str[i]) % MOD);
        if (i >= sub.size()) {
            ll h = (hash.back() - hash[i - sub.size() + 1] *
                xx % MOD + MOD) % MOD;
            if (h == subhash) return i - sub.size() + 1;
        }
    }
    return -1;
}
```

6.2 Trie

```
#include<bits/stdc++.h>
using namespace std;
class Trie
{
private:
    struct Node
    {
        int cnt = 0;
        int sum = 0;
        Node *tr[128] = {};
        ~Node()
        {
            for(int i=0;i<128;i++)
                if(tr[i])delete tr[i];
        }
    };
    Node* root;
public:
    void insert(char *s)
    {
        Node* ptr = root;
        for(;*s;s++)
        {
            if(!ptr->tr[*s])
                ptr->tr[*s] = new Node();
            ptr = ptr->tr[*s];
            ptr->sum++;
        }
        ptr->cnt++;
    }
    inline int count(char *s)
    {
        Node *ptr = find(s);
        return ptr ? ptr->cnt : 0;
    }
    Node* find(char *s)
    {
        Node* ptr = root;
        for(;*s;s++)
        {
            if(!ptr->tr[*s])return 0;
            ptr = ptr->tr[*s];
        }
        return ptr;
    }
    bool erase(char *s)
    {
        Node *ptr = find(s);
        if(!ptr)return false;
        int num = ptr->cnt;
        if(!num)return false;
        ptr = root;
        for(;*s;s++)
        {
            Node *tmp = ptr;
            ptr = ptr->tr[*s];
            ptr->sum -= num;
            if(!ptr->sum)
            {
                delete ptr;
                tmp->tr[*s] = 0;
                return true;
            }
        }
    }
    Trie(){root = new Node();}
    ~Trie(){delete root;}
};

int main()
{
    Trie *trie = new Trie();
    string op;
    char s[20];
    while(cin>>op)
    {
        if(op=="END")
        {
            delete trie;
            break;
        }
        else if(op=="insert")
        {
            cin >> s;
            trie->insert(s);
        }
        else if(op=="erase")
        {
            cin >> s;
            if(trie->erase(s))printf("Success erase");
            else printf("Fail erase");
        }
        else if(op=="count")
        {
            cin >> s;
            printf("%d\n",trie->count(s));
        }
    }
    return 0;
}
```

6.3 AC 自動機

```
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nA)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];
    }
  }
};
```

6.4 KMP

```
// KMP fail function.
int* kmp_fail(string& s) {
    int* f = new int[s.size()];
    int p = f[0] = -1;
    for (int i = 1; s[i]; i++) {
        while (p != -1 && s[p + 1] != s[i]) p = f[p];
        if (s[p + 1] == s[i]) p++;
        f[i] = p;
    }
    return f;
}

// Queries the counts sub appears in str.
int kmp_count(string& str, string& sub) {
    int* fail = kmp_fail(sub);
    int p = -1, ret = 0;
    for (int i = 0; i < str.size(); i++) {
        while (p != -1 && sub[p + 1] != str[i]) p = fail[p];
        if (sub[p + 1] == str[i]) p++;
        if (p == sub.size() - 1) p = fail[p], ret++;
    }
    delete[] fail;
    return ret;
}

// Queries the first index where sub appears in str; -1
    indicates no matching.
int kmp(string& str, string& sub) {
    int* fail = kmp_fail(sub);
    int i, j = 0;
    while (i < str.size() && j < sub.size()) {
        if (sub[j] == str[i])
            i++, j++;
        else if (j == 0)
            i++;
        else
            j = fail[j - 1] + 1;
    }
    delete[] fail;
    return j == sub.size() ? (i - j) : -1;
}
```

6.5 Z

```
//資芽2018
#include<iostream>
#include<string>
using namespace std;
void z_build(string &s, int *z)
{
    int bst = z[0] = 0;
    for(int i=1; s[i]; i++)
    {
        if( z[bst]+bst<i ) z[i] = 0;
        else z[i] = min( z[bst]+bst-i , z[i-bst] );
        while( s[z[i]] == s[i+z[i]] ) z[i]++;
        if( z[i]+i > z[bst]+bst ) bst = i;
    }
}
//s在t中出現幾次
int z_match(string &s, string &t)
{
    int ans = 0;
    int lens = s.length(), lent = t.length();
    int* z = new int[lens+lent+5];
    string st = s+"$"+t;
    z_build(st,z);
    for(int i=lens+1; i<=lens+lent; i++)
        if(z[i]==lens) ans++;
    return ans;
}
int main()
{
    string s,t;
    cin >> s >> t;
    cout << z_match(s,t) << endl;
    return 0;
}
```

6.6 BWT

```cpp
const int N = 8;              // 字串長度
int s[N+N+1] = "suffixes";    // 字串，後面預留一倍空間。
int sa[N];                    // 後綴陣列
int pivot;
int cmp(const void* i, const void* j) {
    return strncmp(s+*(int*)i, s+*(int*)j, N);
}
// 此處便宜行事，採用 O(N²logN) 的後綴陣列演算法。
void BWT() {
    strncpy(s + N, s, N);
    for (int i=0; i<N; ++i) sa[i] = i;
    qsort(sa, N, sizeof(int), cmp);
    // 當輸入字串的所有字元都相同，必須當作特例處理。
    // 或者改用stable sort。
    for (int i=0; i<N; ++i)
        cout << s[(sa[i] + N-1) % N];
    for (int i=0; i<N; ++i)
        if (sa[i] == 0) {
            pivot = i;
            break;
        }
}
// Inverse BWT
const int N = 8;              // 字串長度
char t[N+1] = "xuffessi";     // 字串
int pivot;
int next[N];
void IBWT() {
    vector<int> index[256];
    for (int i=0; i<N; ++i)
        index[t[i]].push_back(i);
    for (int i=0, n=0; i<256; ++i)
        for (int j=0; j<index[i].size(); ++j)
            next[n++] = index[i][j];
    int p = pivot;
    for (int i=0; i<N; ++i)
        cout << t[p = next[p]];
}
```

6.7 Suffix_Array_LCP

```cpp
#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]);
}
```

6.8 LPS

```cpp
char t[1001];             // 原字串
char s[1001 * 2];         // 穿插特殊字元之後的t
int z[1001 * 2], L, R;    // 源自Gusfield's Algorithm
// 由a往左、由b往右，對稱地作字元比對。
int extend(int a, int b)
{
    int i = 0;
    while (a-i>=0 && b+i<N && s[a-i] == s[b+i]) i++;
    return i;
}
void longest_palindromic_substring()
{
    int N = strlen(t);
    // t穿插特殊字元，存放到s。
    // （實際上不會這麼做，都是組算索引值。）
    memset(s, '.', N*2+1);
    for (int i=0; i<N; ++i) s[i*2+1] = t[i];
    N = N*2+1;
    // s[N] = '\0';    // 可做可不做
    // Manacher's Algorithm
    z[0] = 1;
    L = R = 0;
    for (int i=1; i<N; ++i)  {
        int ii = L - (i - L);   // i的映射位置
        int n = R + 1 - i;
        if (i > R)  {
            z[i] = extend(i, i);
            L = i;
            R = i + z[i] - 1;
        }
        else if (z[ii] == n)  {
            z[i] = n + extend(i-n, i+n);
            L = i;
            R = i + z[i] - 1;
        }
        else z[i] = min(z[ii], n);
    }
    // 尋找最長迴文子字串的長度。
    int n = 0, p = 0;
    for (int i=0; i<N; ++i)
        if (z[i] > n)  n = z[p = i];
    // 記得去掉特殊字元。
    cout << "最長迴文子字串的長度是" << (n-1) / 2;
    // 印出最長迴文子字串，記得別印特殊字元。
    for (int i=p-z[p]+1; i<=p+z[p]-1; ++i)
        if (i & 1)  cout << s[i];
}
```

7 Geometry

7.1 Geometry

```cpp
//Copy from Jinkela
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 graham_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 graham(vector<point<T> > &s){//凸包
  sort(s.begin(),s.end(),graham_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
  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);
  }
};
template<typename T>
struct triangle3D{
  point3D<T> a,b,c;
  triangle3D(){}
  triangle3D(const point3D<T> &a,const point3D<T> &b,const point3D<T> &c):a(a),b(b),c(c){}
  bool point_in(const point3D<T> &p)const{//點在該平面上的投影在三角形中
    return line3D<T>(b,c).same_side(p,a)&&line3D<T>(a,c).same_side(p,b)&&line3D<T>(a,b).same_side(p,c);
  }
};
template<typename T>
struct tetrahedron{//四面體
  point3D<T> a,b,c,d;
  tetrahedron(){}
  tetrahedron(const point3D<T> &a,const point3D<T> &b,const point3D<T> &c,const point3D<T> &d):a(a),b(b),c(c),d(d){}
  T volume6()const{//體積的六倍
    return (d-a).dot((b-a).cross(c-a));
  }
  point3D<T> centroid()const{
```

```
    return (a+b+c+d)/4;
  }
  bool point_in(const point3D<T> &p)const{
    return triangle3D<T>(a,b,c).point_in(p)&&triangle3D<T>(c,
        d,a).point_in(p);
  }
};
template<typename T>
struct convexhull3D{
  static const int MAXN=1005;
  struct face{
    int a,b,c;
    face(int a,int b,int c):a(a),b(b),c(c){}
  };
  vector<point3D<T>> pt;
  vector<face> ans;
  int fid[MAXN][MAXN];
  void build(){
    int n=pt.size();
    ans.clear();
    memset(fid,0,sizeof(fid));
    ans.emplace_back(0,1,2);//注意不能共線
    ans.emplace_back(2,1,0);
    int ftop = 0;
    for(int i=3, ftop=1; i<n; ++i,++ftop){
      vector<face> next;
      for(auto &f:ans){
        T d=(pt[i]-pt[f.a]).dot((pt[f.b]-pt[f.a]).cross(pt[f.
            c]-pt[f.a]));
        if(d<=0) next.push_back(f);
        int ff=0;
        if(d>0) ff=ftop;
        else if(d<0) ff=-ftop;
        fid[f.a][f.b]=fid[f.b][f.c]=fid[f.c][f.a]=ff;
      }
      for(auto &f:ans){
        if(fid[f.a][f.b]>0 && fid[f.a][f.b]!=fid[f.b][f.a])
          next.emplace_back(f.a,f.b,i);
        if(fid[f.b][f.c]>0 && fid[f.b][f.c]!=fid[f.c][f.b])
          next.emplace_back(f.b,f.c,i);
        if(fid[f.c][f.a]>0 && fid[f.c][f.a]!=fid[f.a][f.c])
          next.emplace_back(f.c,f.a,i);
      }
      ans=next;
    }
  }
  point3D<T> centroid()const{
    point3D<T> res(0,0,0);
    T vol=0;
    for(auto &f:ans){
      T tmp=pt[f.a].dot(pt[f.b].cross(pt[f.c]));
      res=res+(pt[f.a]+pt[f.b]+pt[f.c])*tmp;
      vol+=tmp;
    }
    return res/(vol*4);
  }
};
```

7.2 旋轉卡尺

```
typedef pair<long long, long long> pt;
const int maxn = 1e6 + 10;
pt operator-(const pt& p1 , const pt& p2) {
    return pt(p1.x - p2.x, p1.y - p2.y);
}
long long cross(const pt& p1, const pt& p2) {
    return p1.x * p2.y - p1.y * p2.x;
}
long long dis(pt a, pt b) {
    return (a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b
        .y);
}
vector<pt> ch;
pt p[maxn];
double shoelace_formula(vector<pt> &v) {
    int n = v.size();
    double ans = 0;
    for (int i = 0; i < n; i++)
        ans += (v[i].x * v[(i + 1) % n].y);
    for (int i = 0; i < n; i++)
        ans -= (v[i].y * v[(i + 1) % n].x);
    return abs(ans / 2);
}
double farthest_dis(vector<pt> &v) {
    int k = 1, n = v.size();
    long long ans = 0;
    if (n == 2) return dis(v[0], v[1]);
    for (int i = 0; i < n; i++) {
        while (llabs(cross(v[i] - v[(k + 1) % n], v[(i + 1) %
             n] - v[(k + 1) % n])) >= llabs(cross(v[i] - v[k
            ], v[(i + 1) % n] - v[k])))
            k = (k + 1) % n;
        ans = max(ans, max(dis(v[i], v[k]), dis(v[(i + 1) % n
            ], v[k])));
    } return sqrt(ans);
}
int n;
int main() {
    cin >> n;
    for (int i = 0; i < n; i++) cin >> p[i].x >> p[i].y;
    sort(p, p + n);
    for (int i = 0; i < n; i++) {
        while (ch.size() >= 2 && cross(ch[ch.size() - 1] - ch
            [ch.size() - 2], p[i] - ch[ch.size() - 2]) <= 0)
            ch.pop_back();
        ch.push_back(p[i]);
    }
    for (int i = n - 2, t = ch.size() + 1; i >= 0; i--) {
        while (ch.size() >= t && cross(ch[ch.size() - 1] - ch
            [ch.size() - 2], p[i] - ch[ch.size() - 2]) <= 0)
            ch.pop_back();
        ch.push_back(p[i]);
    }
    if (n >= 2) ch.pop_back();
    cout << setprecision(6) << fixed << shoelace_formula(ch)
         << '\n';
    cout << setprecision(6) << fixed << farthest_dis(ch) << '
        \n';
    return 0;
}
```

7.3 最近點對

```
template<typename T> struct Point
{
    T x,y;
    Point(){}
    Point(const T &x,const T &y):x(x),y(y){}
    inline T dist(Point b) {
        return sqrt((x-b.x)*(x-b.x)+(y-b.y)*(y-b.y));
    }
    static bool cmpx(const Point &a,const Point &b) {
        if(a.x==b.x)return a.y<b.y;
        return a.x<b.x;
    }
    static bool cmpy(const Point &a,const Point &b) {
        if(a.y==b.y)return a.x<b.x;
        return a.y<b.y;
    }
};
template<typename T,typename _IT = Point<T>* >
void DC(T &d, _IT p, _IT t, int L, int R) //Divide and
     Conquer //NlgN
{
    if(L>=R) return;
    int mid = (L+R)>>1;
    DC(d,p,t,L,mid);
    DC(d,p,t,mid+1,R);
    int N = 0;
    for(int i=mid; i>=L && p[mid].x-p[i].x<d; i--) t[N++] = p
        [i];
    for(int i=mid+1; i<=R && p[i].x-p[mid].x<d; i++) t[N++] =
         p[i];
    sort(t,t+N,t->cmpy);
    for(int i=0; i<N-1; i++)
        for(int j=1; j<=3 && i+j<N; j++)
            d = min(d,t[i].dist(t[i+j]));
}
template<typename T,typename _IT = Point<T>* >
void closest_pair(T &d,_IT p, _IT t, int n) {
    sort(p,p+n,p->cmpx); DC(d,p,t,0,n-1);
}
int main() {
    Point<double> p[maxn],t[maxn];
    int n; scanf("%d",&n);
    for(int i=0; i<n; i++) scanf("%lf%lf",&p[i].x,&p[i].y);
    double d = INF; closest_pair(d,p,t,n);
    printf("distance = %lf\n",d);
    return 0;
}
```

7.4 最小覆蓋圓

```
using PT = point<T>;
using CPT = const PT;
PT circumcenter(CPT &a,CPT &b,CPT &c){
  PT u = b-a, v =c-a;
  T c1 = u.abs2()/2, c2 = v.abs2()/2;
  T d = u.cross(v);
  return PT(a.x+(v.y*c1-u.y*c2)/d, a.y+(u.x*c2-v.x*c1)/d);
}
void solve(PT p[], int n, PT &c, T &r2){
  random_shuffle(p,p+n);
  c = p[0]; r2 = 0; // c,r2 = 圓心,半徑平方
  for(int i=1; i<n; i++)
    if( (p[i]-c).abs2() > r2)
    {
      c=p[i]; r2=0;
      for(int j=0; j<i; j++)
```

```
        if( (p[j]-c).abs2() > r2)
        {
          c.x = (p[i].x+p[j].x)/2;
          c.y = (p[i].y+p[j].y)/2;
          r2 = (p[j]-c).abs2();
          for(int k=0; k<j; k++)
            if( (p[k]-c).abs2() > r2)
            {
              c = circumcenter(p[i],p[j],p[k]);
              r2 = (p[i]-c).abs2();
            }
        }
    }
}
```

7.5 Rectangle Union Area

```
const int maxn = 1e5 + 10;
struct rec{
    int t, b, l, r;
} r[maxn];
int n, cnt[maxn << 2];
long long st[maxn << 2], ans = 0;
vector<int> x, y;
vector<pair<pair<int, int>, pair<int, int>>> v;
void modify(int t, int l, int r, int ql, int qr, int v) {
    if (ql <= l && r <= qr) cnt[t] += v;
    else {
        int m = (l + r) >> 1;
        if (qr <= m) modify(t << 1, l, m, ql, qr, v);
        else if (ql >= m) modify(t << 1 | 1, m, r, ql, qr, v)
            ;
        else modify(t << 1, l, m, ql, m, v), modify(t << 1 |
            1, m, r, m, qr, v);
    }
    if (cnt[t]) st[t] = y[r] - y[l];
    else if (r - l == 1) st[t] = 0;
    else st[t] = st[t << 1] + st[t << 1 | 1];
}
int main() {
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> r[i].l >> r[i].r >> r[i].b >> r[i].t;
        if (r[i].l > r[i].r) swap(r[i].l, r[i].r);
        if (r[i].b > r[i].t) swap(r[i].b, r[i].t);
        x.push_back(r[i].l);
        x.push_back(r[i].r);
        y.push_back(r[i].b);
        y.push_back(r[i].t);
    }
    sort(x.begin(), x.end());
    sort(y.begin(), y.end());
    x.erase(unique(x.begin(), x.end()), x.end());
    y.erase(unique(y.begin(), y.end()), y.end());
    for (int i = 0; i < n; i++) {
        r[i].l = lower_bound(x.begin(), x.end(), r[i].l) - x.
            begin();
        r[i].r = lower_bound(x.begin(), x.end(), r[i].r) - x.
            begin();
        r[i].b = lower_bound(y.begin(), y.end(), r[i].b) - y.
            begin();
        r[i].t = lower_bound(y.begin(), y.end(), r[i].t) - y.
            begin();
        v.emplace_back(make_pair(r[i].l, 1), make_pair(r[i].b
            , r[i].t));
        v.emplace_back(make_pair(r[i].r, -1), make_pair(r[i].
            b, r[i].t));
    }
    sort(v.begin(), v.end(), [](pair<pair<int, int>, pair<int
        , int>> a, pair<pair<int, int>, pair<int, int>> b){
        if (a.first.first != b.first.first) return a.first.
            first < b.first.first;
        return a.first.second > b.first.second;
    });
    for (int i = 0; i < v.size(); i++) {
        if (i) ans += (x[v[i].first.first] - x[v[i - 1].first
            .first]) * st[1];
        modify(1, 0, y.size(), v[i].second.first, v[i].second
            .second, v[i].first.second);
    }
    cout << ans << '\n';
    return 0;
}
```

8 Other

8.1 BuiltIn

```
//gcc專用
//unsigned int ffs
//unsigned long ffsl
//unsigned long long ffsll
#include<stdio.h>
int main()
{
    unsigned int x;
    while(scanf("%u",&x)==1)
    {
        printf("右起第一個1的位置");
        printf("%d\n",__builtin_ffs(x));
        printf("左起第一個1之前0的個數:");
        printf("%d\n",__builtin_clz(x));
        printf("右起第一個1之後0的個數:");
        printf("%d\n",__builtin_ctz(x));
        printf("1的個數:");
        printf("%d\n",__builtin_popcount(x));
        printf("1的個數的奇偶性:");
        printf("%d\n",__builtin_parity(x));
    }
    return 0;
}
```

8.2 莫隊算法-區間眾數

```
using namespace std;
const int maxn = 1e6 + 10;
struct query {
    int id, bk, l, r;
};
int arr[maxn], cnt[maxn], d[maxn], n, m, bk, mx;
pair<int,int> ans[maxn];
vector<query> q;
bool cmp(query x,query y) {
    return (x.bk < y.bk || (x.bk == y.bk) && x.r < y.r);
}
void add(int pos) {
    d[cnt[arr[pos]]]--;
    cnt[arr[pos]]++;
    d[cnt[arr[pos]]]++;
    if(d[mx + 1] > 0) mx++;
}
void del(int pos) {
    d[cnt[arr[pos]]]--;
    cnt[arr[pos]]--;
    d[cnt[arr[pos]]]++;
    if(d[mx] == 0) mx--;
}
void mo(int n, int m) {
    sort(q.begin(), q.end(), cmp);
    for(int i = 0, cl = 1, cr = 0; i < m; i++) {
        while(cr < q[i].r) add(++cr);
        while(cl > q[i].l) add(--cl);
        while(cr > q[i].r) del(cr--);
        while(cl < q[i].l) del(cl++);
        ans[q[i].id] = make_pair(mx, d[mx]);
    }
}
int main(){
    cin >> n >> m;
    bk = (int)sqrt(n + 0.5);
    for(int i = 1; i <= n; i++)
        cin >> arr[i];
    q.resize(m);
    for(int i = 0; i < m; i++) {
        cin >> q[i].l >> q[i].r;
        q[i].id = i,q[i].bk = (q[i].l - 1) / bk;
    }
    mo(n, m);
    for(int i = 0; i < m; i++)
        cout << ans[i].first << ' ' << ans[i].second << '\n';
    return 0;
}
```

8.3 CNF

```
#define MAXN 55
struct CNF{
  int s,x,y;//s->xy | s->x, if y==-1
  int cost;
  CNF(){}
  CNF(int s,int x,int y,int c):s(s),x(x),y(y),cost(c){}
};
int state;//規則數量
map<char,int> rule;//每個字元對應到的規則，小寫字母為終端字符
vector<CNF> cnf;
void init(){
  state=0;
  rule.clear();
  cnf.clear();
}
void add_to_cnf(char s,const string &p,int cost){
  //加入一個s -> <p>的文法，代價為cost
```

```
18    if(rule.find(s)==rule.end())rule[s]=state++;
19    for(auto c:p)if(rule.find(c)==rule.end())rule[c]=state++;
20    if(p.size()==1){
21      cnf.push_back(CNF(rule[s],rule[p[0]],-1,cost));
22    }else{
23      int left=rule[s];
24      int sz=p.size();
25      for(int i=0;i<sz-2;++i){
26        cnf.push_back(CNF(left,rule[p[i]],state,0));
27        left=state++;
28      }
29      cnf.push_back(CNF(left,rule[p[sz-2]],rule[p[sz-1]],cost))
            ;
30    }
31  }
32  vector<long long> dp[MAXN][MAXN];
33  vector<bool> neg_INF[MAXN][MAXN];//如果花費是負的可能會有無限
        小的情形
34  void relax(int l,int r,const CNF &c,long long cost,bool neg_c
        =0){
35    if(!neg_INF[l][r][c.s]&&(neg_INF[l][r][c.x]||cost<dp[l][r][
          c.s])){
36      if(neg_c||neg_INF[l][r][c.x]){
37        dp[l][r][c.s]=0;
38        neg_INF[l][r][c.s]=true;
39      }else dp[l][r][c.s]=cost;
40    }
41  }
42  void bellman(int l,int r,int n){
43    for(int k=1;k<=state;++k)
44      for(auto c:cnf)
45        if(c.y==-1)relax(l,r,c,dp[l][r][c.x]+c.cost,k==n);
46  }
47  void cyk(const vector<int> &tok){
48    for(int i=0;i<(int)tok.size();++i){
49      for(int j=0;j<(int)tok.size();++j){
50        dp[i][j]=vector<long long>(state+1,INT_MAX);
51        neg_INF[i][j]=vector<bool>(state+1,false);
52      }
53      dp[i][i][tok[i]]=0;
54      bellman(i,i,tok.size());
55    }
56    for(int r=1;r<(int)tok.size();++r){
57      for(int l=r-1;l>=0;--l){
58        for(int k=l;k<r;++k)
59          for(auto c:cnf)
60            if(~c.y)relax(l,r,c,dp[l][k][c.x]+dp[k+1][r][c.y]+c
                .cost);
61        bellman(l,r,tok.size());
62      }
63    }
64  }
```

NCTU-Pusheen Codebook

Contents