# 1 Dp

## 1.1 01_knapsack

```cpp
for(int i = 0; i < 100005; i++) dp[i] = dp1[i] = 205;
dp[0] = dp1[0] = 0;
for(int i = 1; i <= n; i++) {
    for(int j = 1e5+1; j >= a[i]; j--) { // 到著做回
        dp[j] = min(dp[j-a[i]] + 1, dp[j]);
    }
}
```

## 1.2 Josephus

```cpp
int josephus(int n, int k) { // n people, kth is killed
    if (n == 1) return 1;
    else return (josephus(n - 1, k) + k - 1) % n + 1;
        /* The position returned by josephus(n - 1, k)
        is adjusted because the recursive call
        josephus(n - 1, k) considers the
        original position k % n + 1 as position 1 */
}
```

## 1.3 SOS_DP

```cpp
int n = 20;
vector<int> a(1<<n);
vector<int> sos(1<<n);
//O(3^n)
for (int i = 0; i < (1<<n); ++i) {
  sos[i] = a[0];
  //iterate over all subsets of i
  for (int j = i; j > 0; j = (j - 1) & i) {
    sos[i] += a[j];
  }
}
//O(3^n)
vector<vector<int>> dp(1<<n, vector<int>(n + 1));
for (int i = 0; i < (1<<n); ++i) {
  dp[i][0] = a[i];
  for (int j = 1; j <= n; ++j) {
    dp[i][j] = dp[i][j - 1];
    if (i & (1 << (j - 1))) {
      dp[i][j] += dp[i - (1 << (j - 1))][j - 1];
    }
  }
  sos[i] = dp[i][n];
}
```

## 1.4 Bitmask

```cpp
// n個城市，m個單向邊，求從1出發走到n的所有  徑數
// 遞迴版本，存反向圖
ll alln;
ll tbl[20][1<<20]; // 建表
ll dp(int i, ll vs) {
    if(tbl[i][vs]) return tbl[i][vs];
    if(vs == alln && i == 0) return 1;
    if(vs == alln || i == 0) return 0;
    ll r = 0;
    For(j, n) {
        if(!g[i][j]) continue;
        if(vs&(1<<j)) continue;
        r += dp(j, vs|(1<<j))*g[i][j];
        r %= mod;
    }
    return tbl[i][vs] = r % mod;
}

alln = (1<<n)-1;
ans = dp(n-1, 1<<(n-1))%mod; //從最後一點遞迴回去，bitmask n
    -1位為1，其餘為0

// TLE版本，迴圈版很難壓常，存正向圖
N = (1<<n)-1; // 可表示n個bit的bitmask
dp[0][1] = 1;
for(int mask = 1; mask <= N; mask++) {
    for(int i = 0; i < n; i++) {
        if(!(1 & mask>>i)) continue;
        int mask2 = mask - (1<<i);
        for(int j = 0; j < n; j++) {
            if(!(1 & mask2>>j) || g[j][i] == 0) continue;
            dp[i][mask] += dp[j][mask2]*g[j][i]; // 非簡單
                圖，可能有重複單向邊，g[i][j]存邊數
            dp[i][mask] %= mod;
        }
    }
}
cout << dp[n-1][N] % mod << '\n';
```

## 1.5 InfinitKnapsack

```cpp
// 找零問題
int main() {    //O(n^2)
    dp[0] = 1;
    for(ll i = 1; i <= n; i++) {
        for(ll j = a[i]; j < 30001; j++) { // 順著做過去
            dp[j] += dp[j-a[i]];
            if(dp[j-a[i]]) coin[j] = i; // 此  額當前拿力哪一
                個錢幣
        }
    }
    ll ans = dp[sum]; // sum = 所求  額
    while (sum) {
        ans.push_back(coin[sum]);
        sum -= a[coin[sum]];    // 遞迴找用過哪些錢幣
    }
}
```

# 2 Data Structure

## 2.1 DSU

```cpp
class DSU{
public:
    DSU(int n ){
        this->n = n;
        reset();
    }
    int n;
    vector<int> boss;
    vector<int> rank;
    vector<int> size;
    void reset(){
        this->boss.resize(n);
        this->rank.resize(n,0);
        this->size.resize(n,0);
        for(int i =0;i<n;i++){
            boss[i] = i;
        }
    }
    int find(int x){
        if(boss[x]!= x){
            boss[x] = find(boss[x]);
        }
        return boss[x];
    }
    int get_size(int x){
        return size[find(x)];
    }
    void merge(int x, int y){
        int a = find(x);
        int b = find(y);
        if(a!=b){
            if(rank[a]<rank[b]){
                boss[a] = b;
                size[b] += size[a];
            }else if (rank[a]<rank[b]){
                boss[b] = a;
                size[a] += size[b];
            }else{
                boss[a] = b;
                size[b] += size[a];
                rank[b]++;
            }
        }
    }
    bool aresame(int a,int b){
        return find(a)==find(b);
    }
};
```

## 2.2 Monotonic Queue

```cpp
class Monotonic_queue{
private:
    deque<int> qu;
public:
    void push(int n){
        while(!qu.empty()&&qu.back()<n){
            qu.pop_back();
        }
        qu.push_back(n);
    }
    int max(){
        return qu.front();
    }
    int min(){
        return qu.back();
    }
    int  size(){
        return qu.size();
    }
    void pop(){
        qu.pop_front();
    }
};
```

## 2.3 BIT

```cpp
class BIT{
public:
    vector<int> bit;
    int N;
    BIT(int n){
        this->N = n;
        this->bit.resize(n);
    }
    void update(int x,int d){
        while(x<=N){
            bit[x]  +=d;
            x +=x&(-x);// lowest bit in x;
        }
    }
    int query(int x){
        int res = 0;
        while(x){
            res+= bit[x];
            x -= x& -x;
        }
        return res;
    }
};
```

## 2.4 Treap

```cpp
// 區間加值、反轉、rotate、刪除、插入元素、求區間
// srand(time(0))
class Treap {
   private:
    struct Node {
        int pri = rand(), size = 1;
        ll val, mn, inc = 0; bool rev = 0;
        Node *lc = 0, *rc = 0;
        Node(ll v) { val = mn = v; }
    };
    Node* root = 0;
    void rev(Node* t) {
        if (!t) return;
        swap(t->lc, t->rc), t->rev ^= 1;
    }
    void update(Node* t, ll v) {
        if (!t) return;
        t->val += v, t->inc += v, t->mn += v;
    }
    void push(Node* t) {
        if (t->rev) rev(t->lc), rev(t->rc), t->rev = 0;
        update(t->lc, t->inc), update(t->rc, t->inc);
        t->inc = 0;
    }
    void pull(Node* t) {
        t->size = 1 + size(t->lc) + size(t->rc);
        t->mn = t->val;
        if (t->lc) t->mn = min(t->mn, t->lc->mn);
        if (t->rc) t->mn = min(t->mn, t->rc->mn);
    }
    void discard(Node* t) { // 看要不要釋放記憶體
        if (!t) return;
        discard(t->lc), discard(t->rc);
        delete t;
    }
    void split(Node* t, Node*& a, Node*& b, int k) {
        if (!t) return a = b = 0, void();
        push(t);
        if (size(t->lc) < k) {
            a = t;
            split(t->rc, a->rc, b, k - size(t->lc) - 1);
            pull(a);
        } else {
            b = t;
            split(t->lc, a, b->lc, k);
            pull(b);
        }
    }
    Node* merge(Node* a, Node* b) {
        if (!a || !b) return a ? a : b;
        if (a->pri > b->pri) {
            push(a);
            a->rc = merge(a->rc, b);
            pull(a);
            return a;
        } else {
            push(b);
            b->lc = merge(a, b->lc);
            pull(b);
            return b;
        }
    }
    inline int size(Node* t) { return t ? t->size : 0; }
   public:
    int size() { return size(root); }
    void add(int l, int r, ll val) {
        Node *a, *b, *c, *d;
        split(root, a, b, r);
        split(a, c, d, l - 1);
        update(d, val);
        root = merge(merge(c, d), b);
    }
    // 反轉區間 [l, r]
    void reverse(int l, int r) {
        Node *a, *b, *c, *d;
        split(root, a, b, r);
        split(a, c, d, l - 1);
        swap(d->lc, d->rc);
        d->rev ^= 1;
        root = merge(merge(c, d), b);
    }
    // 區間 [l, r] 向右 rotate k 次， k < 0 表向左 rotate
    void rotate(int l, int r, int k) {
        int len = r - l + 1;
        Node *a, *b, *c, *d, *e, *f;
        split(root, a, b, r);
        split(a, c, d, l - 1);
        k = (k + len) % len;
        split(d, e, f, len - k);
        root = merge(merge(c, merge(f, e)), b);
    }
    // 插入一個元素 val 使其 index = i <= size
    void insert(int i, ll val) {
        if (i == size() + 1) {
            push_back(val); return;
        }
        assert(i <= size());
        Node *a, *b;
        split(root, a, b, i - 1);
        root = merge(merge(a, new Node(val)), b);
    }
    void push_back(ll val) {
        root = merge(root, new Node(val));
    }
    void remove(int l, int r) {
        int len = r - l + 1;
        Node *a, *b, *c, *d;
        split(root, a, b, l - 1);
        split(b, c, d, len);
        discard(c); // 看你要不要釋放記憶體
        root = merge(a, d);
    }
    ll minn(int l, int r) {
        Node *a, *b, *c, *d;
        split(root, a, b, r);
        split(a, c, d, l - 1);
        int ans = d->mn;
        root = merge(merge(c, d), b);
        return ans;
    }
};
```

## 2.5 Segment Tree

```cpp
class SegmentTree{
private:
    const int n;
    const vl arr;
    // vl st;
    vl summ;
    vl minn;
    vl maxx;
    vl tag;
    void pull(int l,int r,int v){
        if(r-l==1)
            return;
        // st[v]=st[2*v+1]+st[2*v+2];
        int mid=(l+r)/2;
        push(l,mid,2*v+1);
        push(mid,r,2*v+2);
        summ[v]=summ[2*v+1]+summ[2*v+2];
        // minn[v]=min(minn[2*v+1],minn[2*v+2]);
        // maxx[v]=max(maxx[2*v+1],minn[2*v+2]);
    }
    void push(int l,int r,int v){
        summ[v]+=tag[v]*(r-l);
        if(r-l==1)
            return tag[v]=0,void();
        tag[2*v+1]+=tag[v];
        tag[2*v+2]+=tag[v];
        tag[v]=0;
    }
    void build(int l,int r,int v=0){
        if(r-l==1){
            summ[v]=arr[l];
            // summ[v]=minn[v]=maxx[v]=arr[l];
            return;
        }
        int mid=(l+r)/2;
        build(l,mid,2*v+1);
        build(mid,r,2*v+2);
        pull(l,r,v);
    }

public:
    SegmentTree(vl&_arr,int _n):arr(_arr),n(_n){
        assert(arr.size()==n);
        summ.assign(4*n,0);
        // minn.assign(4*n,1e9);
        // maxx.assign(4*n,-1e9);
        tag.assign(4*n,0);
        build(0,arr.size());
    }
    void modify(int x,int val,int l,int r,int v=0){

    }
    // query sum
    loli query(int L,int R,int l,int r,int v=0){
        // dbn(L,R,l,r,v)
        push(l,r,v);
        if(l==L && R==r){
            return summ[v];
            return minn[v];
            return maxx[v];
        }
        int mid=(l+r)/2;
        if(R<=mid)
            return query(L,R,l,mid,2*v+1);
        else if(mid<=L)
            return query(L,R,mid,r,2*v+2);
        else
            return query(L,mid,l,mid,2*v+1)+query(mid,R,mid,r,2*v+2);
    }
    // plus `val` to every element in [L,R)
    void update(int L,int R,loli val,int l,int r,int v=0){
        // dbn(L,R,l,r,v)
        push(l,r,v);
        if(l==L && R==r){
            tag[v]+=val;
            push(l,r,v);
            return;
        }
        int mid=(l+r)/2;
        if(R<=mid)
            update(L,R,val,l,mid,2*v+1);
        else if(mid<=L)
            update(L,R,val,mid,r,2*v+2);
        else
            update(L,mid,val,l,mid,2*v+1),update(mid,R,val,mid,r,2*v+2);
        pull(l,r,v);
    }
};

void solve(){
    int n,q;
    cin>>n>>q;
    vl arr(n);
    for(auto&x:arr)
        cin>>x;
    SegmentTree st(arr,n);
    while(q--){
        int op=0;
        // str op;
        cin>>op;
        if(op&1){
            loli l,r,val;
            cin>>l>>r>>val;
            assert(r>=l);
            st.update(l-1,r,val,0,n);
            // loli k,u;
            // cin>>k>>u;
            // st.update(k-1,k,u-arr[k-1],0,n);
            // arr[k-1]=u;
        }else{
            int x,y;
            cin>>x>>y;
            assert(y>=x);
            cout<<st.query(x-1,y,0,n)<<endl;
        }
    }
}
```

## 2.6 Sparse Table

```cpp
int a[N], sp[__lg(N) + 1][N]{};
void init(int n) { //0-based
  for (int i = 0; i < n; ++i) {
    sp[0][i] = a[i];
  }
  for (int i = 0; i < __lg(n); ++i) {
    for (int j = 0; j+(1<<i) < n; ++j) {
      sp[i + 1][j] = max(sp[i][j], sp[i][j+(1<<i)]);
    }
  }
}
int query(int l, int r) { //[l, r]
  int p = __lg(r - l + 1);
  return max(sp[p][l], sp[p][r-(1<<p)+1]);
}
```

## 2.7 Monotonic Stack

```cpp
vector<int> monotonic_stack(vector<int> nums){
    int n = nums.size();
    vector<int> res(n);
    stack<int>  st;
    for(int i = n-1;i>=0;i--){
        while(!st.empty() && st.top()<=nums[i]){
            st.pop();
        }
        if(st.empty())res[i] = -1;
        else res[i] = st.top();
        st.push(nums[i]);
    }
    return res;
}
```

# 3 Flow

## 3.1 Maximum Simple Graph Matching

```cpp
struct GenMatch { // 1-base
  int V, pr[N];
  bool el[N][N], inq[N], inp[N], inb[N];
  int st, ed, nb, bk[N], djs[N], ans;
  void init(int _V) {
    V = _V;
    for (int i = 0; i <= V; ++i) {
      for (int j = 0; j <= V; ++j) el[i][j] = 0;
      pr[i] = bk[i] = djs[i] = 0;
      inq[i] = inp[i] = inb[i] = 0;
    }
  }
  void add_edge(int u, int v) {
    el[u][v] = el[v][u] = 1;
  }
  int lca(int u, int v) {
    fill_n(inp, V + 1, 0);
    while (1)
      if (u = djs[u], inp[u] = true, u == st) break;
      else u = bk[pr[u]];
    while (1)
      if (v = djs[v], inp[v]) return v;
      else v = bk[pr[v]];
    return v;
  }
  void upd(int u) {
    for (int v; djs[u] != nb;) {
      v = pr[u], inb[djs[u]] = inb[djs[v]] = true;
      u = bk[v];
      if (djs[u] != nb) bk[u] = v;
    }
  }
  void blo(int u, int v, queue<int> &qe) {
    nb = lca(u, v), fill_n(inb, V + 1, 0);
    upd(u), upd(v);
    if (djs[u] != nb) bk[u] = v;
    if (djs[v] != nb) bk[v] = u;
    for (int tu = 1; tu <= V; ++tu)
      if (inb[djs[tu]])
        if (djs[tu] = nb, !inq[tu])
          qe.push(tu), inq[tu] = 1;
  }
  void flow() {
    fill_n(inq + 1, V, 0), fill_n(bk + 1, V, 0);
    iota(djs + 1, djs + V + 1, 1);
    queue<int> qe;
    qe.push(st), inq[st] = 1, ed = 0;
    while (!qe.empty()) {
      int u = qe.front();
      qe.pop();
      for (int v = 1; v <= V; ++v)
        if (el[u][v] && djs[u] != djs[v] &&
          pr[u] != v) {
          if ((v == st) ||
            (pr[v] > 0 && bk[pr[v]] > 0))
            blo(u, v, qe);
          else if (!bk[v]) {
            if (bk[v] = u, pr[v] > 0) {
              if (!inq[pr[v]]) qe.push(pr[v]);
            } else
              return ed = v, void();
          }
        }
    }
  }
  void aug() {
    for (int u = ed, v, w; u > 0;)
      v = bk[u], w = pr[v], pr[v] = u, pr[u] = v,
      u = w;
  }
  int solve() {
    fill_n(pr, V + 1, 0), ans = 0;
    for (int u = 1; u <= V; ++u)
      if (!pr[u])
        if (st = u, flow(), ed > 0) aug(), ++ans;
    return ans;
  }
};
```

## 3.2 Dinic

```
#define maxn 2005
#define INF 0x3f3f3f3f
struct MaxFlow{
    struct edge{
        int  to, cap, flow,rev;
        edge( int v, int c, int f,int r) :  to(v), cap(c),
             flow(f),rev(r) {}
      };
    vector<edge> G[maxn];
    int s,t,dis[maxn],cur[maxn],vis[maxn];
    void add_edge(int from,int to,int cap){
        G[from].push_back(edge(to,cap,0,G[to].size()));
        G[to].push_back(edge(from,0,0,G[from].size()-1));
    }
    bool bfs(){
        memset(dis, -1, sizeof(dis));
        queue<int> qu;
        qu.push(s);
        dis[s] = 0;
        while (!qu.empty()) {
            int from = qu.front();
            qu.pop();
            for (auto &e: G[from]) {
                if (dis[e.to]==-1 && e.cap != e.flow) {
                    dis[e.to] = dis[from] + 1;
                    qu.push(e.to);
                }
            }
        }
        return dis[t]!=-1;
    }
    int dfs(int from,int cap){
        if(from==t ||cap==0)return cap;
        for(int &i = cur[from];i<G[from].size();i++){
            edge &e = G[from][i];
            if(dis[e.to]==dis[from]+1 && e.flow!=e.cap){
                int df = dfs(e.to,min(e.cap-e.flow,cap));
                if(df){
                    e.flow+=df;
                    G[e.to][e.rev].flow-=df;
                    return df;
                }
            }
        }
        dis[from] = -1;
        return 0;
    }
    int Maxflow(int s,int t){
        this->s = s,this->t =t;
        int flow = 0;
        int df;
        while(bfs()){
            memset(cur,0,sizeof(cur));
            while(df = dfs(s,INF)){
                flow +=df;
            }
        }
        return flow;
    }
};
int main(){
    int n = 4,m = 6;
    MaxFlow maxflow;
    for(int i =0;i<m;i++){
        int a,b,cap;
        cin >>a>>b>>cap;
        maxflow.add_edge(a,b,cap);
    }
    cout << maxflow.Maxflow(1,3)<<endl;;
}
```

# 4 Formula

## 4.1 formula

### 4.1.1 Pick 公式

給定頂點坐標均是整點的簡單多邊形，面積 = 內部格點數 + 邊上格點數/2-1

### 4.1.2 圖論

1. 對於平面圖，$F = E - V + C + 1$，C 是連通分　數
2. 對於平面圖，$E \leq 3V - 6$
3. 對於連通圖 G，最大獨立點集的大小設為 I(G)，最大匹配大小設為 M(G)，最小點覆蓋設為 Cv(G)，最小邊覆蓋設為 Ce(G)。對於任意連通圖：
   (a) $I(G) + Cv(G) = |V|$
   (b) $M(G) + Ce(G) = |V|$
4. 對於連通二分圖：
   (a) $I(G) = Cv(G)$
   (b) $M(G) = Ce(G)$
5. 最大權閉合圖：
   (a) $C(u, v) = \infty, (u, v) \in E$
   (b) $C(S, v) = W_v, W_v > 0$
   (c) $C(v, T) = -W_v, W_v < 0$
   (d) ans=$\sum_{W_v>0} W_v - flow(S, T)$
6. 最大密度子圖：
   (a) 求 $max\left(\frac{W_e+W_v}{|V'|}\right), e \in E', v \in V'$
   (b) $U = \sum_{v\in V} 2W_v + \sum_{e\in E} W_e$
   (c) $C(u, v) = W_{(u,v)}, (u, v) \in E$，雙向邊
   (d) $C(S, v) = U, v \in V$
   (e) $D_u = \sum_{(u,v)\in E} W_{(u,v)}$
   (f) $C(v, T) = U + 2g - D_v - 2W_v, v \in V$
   (g) 二分搜 $g$：
       $l = 0, r = U, eps = 1/n^2$
       if($(U \times |V| - flow(S, T))/2 > 0$) $l = mid$
       else $r = mid$
   (h) ans=$min\_cut(S, T)$
   (i) $|E| = 0$ 要特殊判斷
7. 弦圖：
   (a) 點數大於 3 的環都要有一條弦
   (b) 完美消除序　從後往前依次給每個點染色，給每個點染上可以染的最小顏色
   (c) 最大團大小 = 色數
   (d) 最大獨立集：完美消除序　從前往後能選就選
   (e) 最小團覆蓋：最大獨立集的點和他延伸的邊構成
   (f) 區間圖是弦圖
   (g) 區間圖的完美消除序　：將區間按造又端點由小到大排序
   (h) 區間圖染色：用線段樹做

### 4.1.3 dinic 特殊圖複雜度

1. 單位流：$O\left(min\left(V^{3/2}, E^{1/2}\right)E\right)$
2. 二分圖：$O\left(V^{1/2}E\right)$

### 4.1.4 0-1 分數規劃

$x_i = \{0, 1\}$，$x_i$ 可能會有其他限制，求 $max\left(\frac{\sum B_i x_i}{\sum C_i x_i}\right)$

1. $D(i, g) = B_i - g \times C_i$
2. $f(g) = \sum D(i, g)x_i$
3. $f(g) = 0$ 時 $g$ 為最佳解，$f(g) < 0$ 沒有意義
4. 因為 $f(g)$ 單調可以二分搜 $g$
5. 或用 Dinkelbach 通常比較快

```
binary_search(){
  while(r-l>eps){
    g=(l+r)/2;
    for(i:所有元素)D[i]=B[i]-g*C[i];//D(i,g)
    找出一組合法x[i]使f(g)最大;
    if(f(g)>0) l=g;
    else r=g;
  }
  Ans = r;
}
Dinkelbach(){
  g=任意  態(通常設為0);
  do{
    Ans=g;
    for(i:所有元素)D[i]=B[i]-g*C[i];//D(i,g)
    找出一組合法x[i]使f(g)最大;
    p=0,q=0;
    for(i:所有元素)
      if(x[i])p+=B[i],q+=C[i];
    g=p/q;//  新解，注意q=0的情況
  }while(abs(Ans-g)>EPS);
  return Ans;
}
```

#### 4.1.5 學長公式

1. $\sum_{d|n} \phi(n) = n$
2. $g(n) = \sum_{d|n} f(d) => f(n) = \sum_{d|n} \mu(d) \times g(n/d)$
3. Harmonic series $H_n = \ln(n)+\gamma+1/(2n)-1/(12n^2)+1/(120n^4)$
4. $\gamma = 0.57721566490153286060651209008240243104215$
5. 格雷碼 $= n \oplus (n >> 1)$
6. $SG(A+B) = SG(A) \oplus SG(B)$
7. 旋轉矩陣 $M(\theta) = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}$

#### 4.1.6 基本數論

1. $\sum_{d|n} \mu(n) = [n == 1]$
2. $g(m) = \sum_{d|m} f(d) \Leftrightarrow f(m) = \sum_{d|m} \mu(d) \times g(m/d)$
3. $\sum_{i=1}^{n} \sum_{j=1}^{m}$ 互質數 $= \sum \mu(d) \lfloor \frac{n}{d} \rfloor \lfloor \frac{m}{d} \rfloor$
4. $\sum_{i=1}^{n} \sum_{j=1}^{n} lcm(i,j) = n \sum_{d|n} d \times \phi(d)$

#### 4.1.7 排組公式

1. k 卡特 $\frac{C_n^{kn}}{n(k-1)+1}$，$C_m^n = \frac{n!}{m!(n-m)!}$
2. $H(n,m) \cong x_1 + x_2 \ldots + x_n = k, num = C_k^{n+k-1}$
3. Stirling number of $2^{nd}$,$n$ 人分 $k$ 組方法數目
   (a) $S(0,0) = S(n,n) = 1$
   (b) $S(n,0) = 0$
   (c) $S(n,k) = kS(n-1,k) + S(n-1,k-1)$
4. Bell number,$n$ 人分任意多組方法數目
   (a) $B_0 = 1$
   (b) $B_n = \sum_{i=0}^{n} S(n,i)$
   (c) $B_{n+1} = \sum_{k=0}^{n} C_k^n B_k$
   (d) $B_{p+n} \equiv B_n + B_{n+1} mod p$, p is prime
   (e) $B_{p^m+n} \equiv mB_n + B_{n+1} mod p$, p is prime
   (f) From $B_0 : 1, 1, 2, 5, 15, 52,$
   $203, 877, 4140, 21147, 115975$
5. Derangement，錯排，沒有人在自己位置上
   (a) $D_n = n!(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} \ldots + (-1)^n \frac{1}{n!})$
   (b) $D_n = (n-1)(D_{n-1} + D_{n-2}), D_0 = 1, D_1 = 0$
   (c) From $D_0 : 1, 0, 1, 2, 9, 44,$
   $265, 1854, 14833, 133496$
6. Binomial Equality
   (a) $\sum_k \binom{r}{m+k}\binom{s}{n-k} = \binom{r+s}{m+n}$
   (b) $\sum_k \binom{l}{m+k}\binom{s}{n+k} = \binom{l+s}{l-m+n}$
   (c) $\sum_k \binom{l}{m+k}\binom{s+k}{n}(-1)^k = (-1)^{l+m}\binom{s-m}{n-l}$
   (d) $\sum_{k \le l} \binom{l-k}{m}\binom{s}{k-n}(-1)^k = (-1)^{l+m}\binom{s-m-1}{l-n-m}$
   (e) $\sum_{0 \le k \le l} \binom{l-k}{m}\binom{q+k}{n} = \binom{l+q+1}{m+n+1}$
   (f) $\binom{r}{k} = (-1)^k \binom{k-r-1}{k}$
   (g) $\binom{r}{m}\binom{m}{k} = \binom{r}{k}\binom{r-k}{m-k}$
   (h) $\sum_{k \le n} \binom{r+k}{k} = \binom{r+n+1}{n}$
   (i) $\sum_{0 \le k \le n} \binom{k}{m} = \binom{n+1}{m+1}$
   (j) $\sum_{k \le m} \binom{m+r}{k} x^k y^k = \sum_{k \le m} \binom{-r}{k}(-x)^k (x+y)^{m-k}$

#### 4.1.8 冪次，冪次和

1. $a^b \% P = a^{b\%\varphi(p)+\varphi(p)}, b \ge \varphi(p)$
2. $1^3 + 2^3 + 3^3 + \ldots + n^3 = \frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4}$
3. $1^4 + 2^4 + 3^4 + \ldots + n^4 = \frac{n^5}{5} + \frac{n^4}{2} + \frac{n^3}{3} - \frac{n}{30}$
4. $1^5 + 2^5 + 3^5 + \ldots + n^5 = \frac{n^6}{6} + \frac{n^5}{2} + \frac{5n^4}{12} - \frac{n^2}{12}$
5. $0^k + 1^k + 2^k + \ldots + n^k = P(k), P(k) = \frac{(n+1)^{k+1} - \sum_{i=0}^{k-1} C_i^{k+1} P(i)}{k+1}, P(0) = n+1$
6. $\sum_{k=0}^{m-1} k^n = \frac{1}{n+1} \sum_{k=0}^{n} C_k^{n+1} B_k m^{n+1-k}$
7. $\sum_{j=0}^{m} C_j^{m+1} B_j = 0, B_0 = 1$
8. 除 $B_1 = -1/2$，剩下的奇數項都是 0
9. $B_2 = 1/6, B_4 = -1/30, B_6 = 1/42, B_8 = -1/30, B_{10} = 5/66, B_{12} = -691/2730, B_{14} = 7/6, B_{16} = -3617/510, B_{18} = 43867/798, B_{20} = -174611/330,$

#### 4.1.9 Burnside's lemma

1. $|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$
2. $X^g = t^{c(g)}$
3. $G$ 表示有幾種轉法，$X^g$ 表示在那種轉法下，有幾種是會保持對稱的，$t$ 是顏色數，$c(g)$ 是循環節不動的面數。
4. 正立方體塗三顏色，轉 0 有 $3^6$ 個元素不變，轉 90 有 6 種，每種有 $3^3$ 不變，180 有 $3 \times 3^4$，120(角) 有 $8 \times 3^2$，180(邊) 有 $6 \times 3^3$，全部 $\frac{1}{24}\left(3^6 + 6 \times 3^3 + 3 \times 3^4 + 8 \times 3^2 + 6 \times 3^3\right) = 57$

#### 4.1.10 Count on a tree

1. Rooted tree: $s_{n+1} = \frac{1}{n} \sum_{i=1}^{n} (i \times a_i \times \sum_{j=1}^{\lfloor n/i \rfloor} a_{n+1-i \times j})$
2. Unrooted tree:
   (a) Odd:$a_n - \sum_{i=1}^{n/2} a_i a_{n-i}$
   (b) Even:$Odd + \frac{1}{2} a_{n/2}(a_{n/2}+1)$
3. Spanning Tree
   (a) 完全圖 $n^{n-2}$
   (b) 一般圖 (Kirchhoff's theorem)$M[i][i] = degree(V_i)$,$M[i][j] = -1$,if have $E(i,j)$,0 if no edge. delete any one row and col in $A$, $ans = det(A)$

# 5 Geometry

## 5.1 Sort by Angle

```
bool cmp(pii a, pii b) {
#define is_neg(k) (k.y < 0 || (k.y == 0 && k.x < 0));
  int A = is_neg(a), B = is_neg(b);
  if (A != B)
    return A < B;
  if (cross(a, b) == 0)
    return (a.x*a.x + a.y*a.y) < (b.x*b.x + b.y*b.y);
  return cross(a, b) > 0;
}
```

## 5.2 Geometry

```
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
    return point(-y,x); }
  T abs2()const{//向  長度的平方
    return dot(*this); }
  T rad(const point &b)const{//  向  的弧度
return fabs(atan2(fabs(cross(b)),dot(b))); }
  T getA()const{//對x軸的弧度
```

```cpp
    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  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{//  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
```

```cpp
    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 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 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
  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 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 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);
  }
};
```

## 5.3 Convex Hull

```cpp
using pdd = pair<double, double>;
#define F first
#define S second
pdd operator-(pdd a, pdd b) {
  return {a.F - b.F, a.S - b.S};
}
double cross(pdd a, pdd b) {
  return a.F * b.S - a.S * b.F;
}
void solve() {
  int n;
  cin >> n;
  vector<pdd> pnts;
  for (int i = 0; i < n; ++i) {
    double x, y;
    cin >> x >> y;
    pnts.push_back(x, y);
  }
  sort(iter(pnts));
  vector<pdd> hull;
  for (int i = 0; i < 2; ++i) {
    int t = hull.size();
    for (pdd j: pnts) {
      while(hull.size() - t >= 2 && cross(j - hull[hull.size() - 2], hull.back() - hull[hull.size() - 2]) >= 0)
        hull.pop_back();
      hull.push_back(j);
    }
    hull.pop_back();
    reverse(iter(pnts));
  }
  double area = 0;
  for (int i=0; i < hull.size(); ++i){
    area += cross(hull[i], hull[(i + 1) % hull.size()]);
  }
  area /= 2.0;
}
```

## 5.4 Min Covering Circle

```cpp
double dis(pdd a, pdd b) {
  double dx = a.x - b.x, dy = a.y - b.y;
  return sqrt(dx*dx + dy*dy);
}
double sq(double x) {
  return x * x;
}
pdd excenter(pdd p1, pdd p2, pdd p3) {
  double a1 = p1.x - p2.x, a2 = p1.x - p3.x;
  double b1 = p1.y - p2.y, b2 = p1.y - p3.y;
  double c1 = (sq(p1.x) - sq(p2.x) + sq(p1.y) - sq(p2.y)) / 2;
  double c2 = (sq(p1.x) - sq(p3.x) + sq(p1.y) - sq(p3.y)) / 2;
  double dd = a1*b2 - a2*b1;
  return {(c1*b2 - c2*b1) / dd, (a1*c2 - a2*c1) / dd};
}
void solve(pdd a[], int n) {
  shuffle(a, a + n, rng);
  pdd center = a[0];
  double r = 0;
  for (int i = 1; i < n; ++i) {
    if (dis(center, a[i]) <= r) continue;
    center = a[i], r = 0;
    for (int j = 0; j < i; ++j) {
      if (dis(center, a[j]) <= r) continue;
      center.x = (a[i].x + a[j].x) / 2;
      center.y = (a[i].y + a[j].y) / 2;
      r = dis(center, a[i]);
      for (int k = 0; k < j; ++k) {
        if (dis(center, a[k]) <= r) continue;
        center = excenter(a[i], a[j], a[k]);
        r = dis(center, a[i]);
      }
    }
  }
  cout << fixed << setprecision(10) << r << '\n';
  cout << center.x << ' ' << center.y << '\n';
}
```

## 5.5 Point in Polygon

```cpp
const ll inf = 2000000000;
struct Point {
  ll x, y;
  Point(ll x = 0, ll y = 0):x(x), y(y){}
  Point operator+(const Point p) const {
    return Point(x + p.x, y + p.y); }
  Point operator-(const Point p) const {
    return Point(x - p.x, y - p.y); }
  ll operator*(const Point p) const { //dot
    return x * p.x + y * p.y; }
  ll operator^(const Point p) const { //cross
    return x * p.y - y * p.x; }
};
bool onseg(Point a, Point b, Point o) {
  return ((a - o) ^ (b - o)) == 0 && ((a - o) * (b - o)) <= 0;
}
int ori(Point a, Point b, Point o) {
  ll w = (a - o) ^ (b - o);
  return (w ? (w > 0 ? 1 : -1) : 0);
}
bool inters(Point a, Point b, Point c, Point d) {
  if (onseg(a, b, c) || onseg(a, b, d)) return 1;
  if (onseg(c, d, a) || onseg(c, d, b)) return 1;
  if (ori(a, b, c) * ori(a, b, d) < 0 && ori(c, d, a) * ori(c, d, b) < 0) return 1;
  return 0;
}
Point poly[maxn];
void solve(int n, Point p) {
  poly[n] = poly[0];
  int cnt = 0;
  for (int i = 0; i < n; ++i) {
    if (onseg(poly[i], poly[i + 1], p)) {
      cnt = -1;
      break;
    }
    if (inters(poly[i], poly[i + 1], p, Point(inf, p.y))) {
      ++cnt;
    }
    Point hi = (poly[i].y > poly[i + 1].y ? poly[i] : poly[i + 1]);
    if (hi.y == p.y && hi.x > p.x) {
      --cnt;
    }
  }
  if (cnt < 0)
    cout << "BOUNDARY\n";
  else if (cnt % 2)
    cout << "INSIDE\n";
  else
    cout << "OUTSIDE\n";
}
```

# 6 Graph

## 6.1 Bipartite Matching

```cpp
const int MAXN = 100;

struct Bipartite_matching{
    int mx[MAXN], my[MAXN], vy[MAXN]; //matchX, matchY, visitY
    vector<int> edge[MAXN]; //adjcent list;
    int x_cnt;
    bool dfs(int x){
        for(auto y: edge[x]){ //對 x 可以碰到的邊進  檢查
            if(vy[y] == 1) continue; //避免遞迴 error

            vy[y] = 1;
            if(my[y] == -1 || dfs(my[y])){ //分析 3
                mx[x] = y;
                my[y] = x;
                return true;
            }
        }
        return false; //分析 4
    }

    int bipartite_matching(){
        memset(mx, -1, sizeof(mx)); //分析 1,2
        memset(my, -1, sizeof(my));
        int ans = 0;
        for(int i = 0; i < x_cnt; i++){ //對每一個 x 節點進DFS(最大匹配)
            memset(vy, 0, sizeof(vy));
            if(dfs(i)) ans++;
        }
        return ans;
    }
    vector<vector<int>> get_match(){
        vector<vector<int>> res;
        for(int i =0 ;i<x_cnt;i++){
            if(mx[i]!=-1){
                res.push_back({i,mx[i]});
            }
        }
        return res;
    }
    void add_edge(int i,int j){
        edge[i].push_back(j);
    }
    void init(int x){
        x_cnt = x;
    }
};
int main(){
    int n,m;
    Bipartite_matching bm;
    for(int i = 0;i<m;i++){
        int a , b;cin >>a>>b;
        bm.add_edge(a,b);
    }
    bm.init(n);
    cout << bm.bipartite_matching()<<endl;
    auto match = bm.get_match();
    for(auto t: match){
        cout << t[0]<<" "<<t[1]<<endl;
    }

}
```

## 6.2 Tarjan SCC

```cpp
const int n = 16;
vector<vector<int>> graph;
int visit[n], low[n],t = 0;
int st[n], top =0;
bool instack[n];
int contract[n]; // 每個點收縮到的點
vector<vector<int>> block;
void dfs(int x,int parent){
    // cout <<x<<endl;
    visit[x] = low[x] = ++t;
  st[top++] = x;
  instack[x] = true;
    for(auto to: graph[x]){
        if(!visit[to])
            dfs(to,x);

        if(instack[to])
            low[x] = min(low[x],low[to]);
    }
    if(visit[x]==low[x]){ //scc 裡最早拜訪的
        int j;
        block.push_back({});
        do{
            j = st[--top];
            instack[j] = false;
            block[block.size()-1].push_back(j);
            contract[j] =x;
        }while(j!=x);
    }
}
int main(){
    for(int i =0;i<n;i++){
        if (!visit[i])
      dfs(i, i);
    }
    for(auto t: block){
        for(auto x:t){
            cout << x <<" ";
        }cout <<endl;
    }
}
```

## 6.3 Bridge

```cpp
const int n = 9;
vector<vector<int>> graph;
vector<int> visit(n, 0);
vector<int> trace(n, 0);
vector<vector<int>> bridge;
int t = 0;
void dfs(int x, int parent){
    visit[x] = ++t;
    trace[x] = x; // 最高祖先預設為自己
    for (auto to : graph[x]){
        if (visit[to]){ // back edge
            if (to != parent){
                trace[x] = to;
            }
        }
        else{ // treeedge
            dfs(to, x);
            if (visit[trace[to]] < visit[trace[x]])
                trace[x] = trace[to];

            // 子樹回不到祖先暨自身。
            if (visit[trace[to]] > visit[x])
                bridge.push_back({x, to});
        }
    }
}//call for()dfs(i,-1)
int main(){
    for(int i =0;i<9;i++){
        if(!visit[i])
            dfs(i,-1);
    }
    for(auto x: bridge){
        cout << x[0]<<" "<< x[1]<<endl;
    }
}
```

## 6.4 2 SAT

```cpp
class TwoSAT{
```

```cpp
public:
    TwoSAT(int n) : n(n), graph(2 * n), visited(2 * n, false) {}
    void addClause(int a, int b) {// 0-base;
        a *=2;
        b *=2;
        // Add implications (~a => b) and (~b => a)
        graph[a ^ 1].push_back(b);
        graph[b ^ 1].push_back(a);
    }
    bool solve() {// Find SCCs and check for contradictions
        for (int i = 0; i < 2 * n; ++i) {
            if (!visited[i]) {
                dfs1(i);
            }
        }
        reverse(processingOrder.begin(), processingOrder.end());//topological sort
        for (int i = 0; i < 2 * n; ++i) {
            visited[i] = false;
        }
        for (int node : processingOrder) {
            if (!visited[node]) {
                scc.clear();
                dfs2(node);
                if (!checkSCCConsistency()) {
                    return false;
                }
            }
        }

        return true;
    }

private:
    int n;
    vector<vector<int>> graph;
    vector<bool> visited;
    vector<int> processingOrder;
    vector<int> scc;

    void dfs1(int node) {
        visited[node] = true;
        for (int neighbor : graph[node]) {
            if (!visited[neighbor]) {
                dfs1(neighbor);
            }
        }
        processingOrder.push_back(node);
    }

    void dfs2(int node) {
        visited[node] = true;
        scc.push_back(node);
        for (int neighbor : graph[node]) {
            if (!visited[neighbor]) {
                dfs2(neighbor);
            }
        }
    }

    bool checkSCCConsistency() {
        for (int node : scc) {
            if (find(scc.begin(), scc.end(), node ^ 1) != scc.end()) {
                return false; // Contradiction found in the same SCC
            }
        }
        return true;
    }
};
int main() {
    int n, m;// Number of variables and clauses
    TwoSAT twoSat(n);
    for (int i = 0; i < m; ++i) {
        int a, b;
        twoSat.addClause(a, b);
    }
    if (twoSat.solve()) {
        cout << "Satisfiable" << endl;
    } else {
        cout << "Unsatisfiable" << endl;
    }
}
```

## 6.5 Kosaraju 2DFS

```cpp
const int n = 16;
vector<vector<int>> graph;
vector<vector<int>> reverse_graph;
int visit[n];
int contract[n]; // 每個點收縮到的點
vector<vector<int>> block;
vector<int> finish;//fake topological sort
// need graph and reverse praph
void dfs1(int x){
    visit[x] = true;
    for(auto to:graph[x]){
        if(!visit[to]){
            dfs1(to);
        }
    }
    finish.push_back(x);
}
void dfs2(int x,int c){
    contract[x] = c;
    block[c].push_back(x);
    visit[x] = true;
    for(auto to:reverse_graph[x]){
        if(!visit[to]){
            dfs2(to,c);
        }
    }
}
int main(){
    graph = {};
    reverse_graph = {};

    for(int i =0;i<n;i++){
        if (!visit[i])
      dfs1(i);
    }
    int c =0;
    memset(visit,0,sizeof(visit));
    for(int i =  n-1;i>=0;i--){
        if(!visit[finish[i]]){
            block.push_back({});
            dfs2(finish[i],c++);
        }
    }
    for(auto t: block){
        for(auto x:t){
            cout << x <<" ";
        }cout <<endl;
    }
}
```

## 6.6 Minimum Steiner Tree

```cpp
// Minimum Steiner Tree
// O(V 3^T + V^2 2^T)
struct SteinerTree { // 0-base
  static const int T = 10, N = 105, INF = 1e9;
  int n, dst[N][N], dp[1 << T][N], tdst[N];
  int vcost[N]; // the cost of vertexs
  void init(int _n) {
    n = _n;
    for (int i = 0; i < n; ++i) {
      for (int j = 0; j < n; ++j) dst[i][j] = INF;
      dst[i][i] = vcost[i] = 0;
    }
  }
  void add_edge(int ui, int vi, int wi) {
    dst[ui][vi] = min(dst[ui][vi], wi);
  }
  void shortest_path() {
    for (int k = 0; k < n; ++k)
      for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
          dst[i][j] =
            min(dst[i][j], dst[i][k] + dst[k][j]);
  }
  int solve(const vector<int> &ter) {
    shortest_path();
    int t = SZ(ter);
    for (int i = 0; i < (1 << t); ++i)
      for (int j = 0; j < n; ++j) dp[i][j] = INF;
    for (int i = 0; i < n; ++i) dp[0][i] = vcost[i];
    for (int msk = 1; msk < (1 << t); ++msk) {
      if (!(msk & (msk - 1))) {
        int who = __lg(msk);
        for (int i = 0; i < n; ++i)
          dp[msk][i] =
            vcost[ter[who]] + dst[ter[who]][i];
      }
      for (int i = 0; i < n; ++i)
        for (int submsk = (msk - 1) & msk; submsk;
             submsk = (submsk - 1) & msk)
          dp[msk][i] = min(dp[msk][i],
            dp[submsk][i] + dp[msk ^ submsk][i] -
              vcost[i]);
      for (int i = 0; i < n; ++i) {
        tdst[i] = INF;
        for (int j = 0; j < n; ++j)
          tdst[i] =
            min(tdst[i], dp[msk][j] + dst[j][i]);
      }
      for (int i = 0; i < n; ++i) dp[msk][i] = tdst[i];
    }
    int ans = INF;
    for (int i = 0; i < n; ++i)
      ans = min(ans, dp[(1 << t) - 1][i]);
    return ans;
  }
};
```

## 6.7 Dijkstra

```cpp
#define maxn 200005
vector<int> dis(maxn,-1);
vector<int> parent(maxn,-1);
vector<bool> vis(maxn,false);
vector<vector<pair<int,int>>> graph;
void dijsktra(int source){
    dis[source] =0 ;

    priority_queue<pair<int,int>,vector<pair<int,int>>,
         greater<pair<int,int>>> pq;
    pq.push({0,source});
    while(!pq.empty()){
        int from  = pq.top().second;
        pq.pop();
        // cout <<vis[from]<<endl;
        if(vis[from])continue;
        vis[from] = true;
        for(auto next : graph[from]){
            int to = next.second;
            int weight = next.first;
            // cout <<from<<' ' <<to<<' ' <<weight;
            if(dis[from]+weight< dis[to] || dis[to]==-1){
                dis[to] = dis[from]+weight;
                parent[to] = from;
                pq.push({dis[from]+weight,to});
            }
        }
    }
}
int main(){
    int startpoint;
    dijsktra(startpoint);
    //dis and parent
}
```

## 6.8 Maximum Clique Dyn

```cpp
const int N = 150;
struct MaxClique { // Maximum Clique
  bitset<N> a[N], cs[N];
  int ans, sol[N], q, cur[N], d[N], n;
  void init(int _n) {
    n = _n;
    for (int i = 0; i < n; i++) a[i].reset();
  }
  void addEdge(int u, int v) { a[u][v] = a[v][u] = 1; }
  void csort(vector<int> &r, vector<int> &c) {
    int mx = 1, km = max(ans - q + 1, 1), t = 0,
        m = r.size();
    cs[1].reset(), cs[2].reset();
    for (int i = 0; i < m; i++) {
      int p = r[i], k = 1;
      while ((cs[k] & a[p]).count()) k++;
      if (k > mx) mx++, cs[mx + 1].reset();
      cs[k][p] = 1;
      if (k < km) r[t++] = p;
    }
    c.resize(m);
    if (t) c[t - 1] = 0;
    for (int k = km; k <= mx; k++)
      for (int p = cs[k]._Find_first(); p < N;
           p = cs[k]._Find_next(p))
        r[t] = p, c[t] = k, t++;
  }
  void dfs(vector<int> &r, vector<int> &c, int l,
    bitset<N> mask) {
    while (!r.empty()) {
      int p = r.back();
      r.pop_back(), mask[p] = 0;
      if (q + c.back() <= ans) return;
      cur[q++] = p;
      vector<int> nr, nc;
      bitset<N> nmask = mask & a[p];
      for (int i : r)
        if (a[p][i]) nr.push_back(i);
      if (!nr.empty()) {
        if (l < 4) {
          for (int i : nr)
            d[i] = (a[i] & nmask).count();
          sort(nr.begin(), nr.end(),
            [&](int x, int y) { return d[x] > d[y]; });
        }
        csort(nr, nc), dfs(nr, nc, l + 1, nmask);
      } else if (q > ans) ans = q, copy_n(cur, q, sol);
      c.pop_back(), q--;
    }
  }
  int solve(bitset<N> mask = bitset<N>(
              string(N, '1'))) { // vertex mask
    vector<int> r, c;
    ans = q = 0;
    for (int i = 0; i < n; i++)
      if (mask[i]) r.push_back(i);
    for (int i = 0; i < n; i++)
      d[i] = (a[i] & mask).count();
    sort(r.begin(), r.end(),
      [&](int i, int j) { return d[i] > d[j]; });
    csort(r, c), dfs(r, c, 1, mask);
    return ans; // sol[0 ~ ans-1]
  }
} graph;
```

## 6.9 Minimum Clique Cover

```cpp
struct Clique_Cover { // 0-base, O(n2^n)
  int co[1 << N], n, E[N];
  int dp[1 << N];
  void init(int _n) {
    n = _n, fill_n(dp, 1 << n, 0);
    fill_n(E, n, 0), fill_n(co, 1 << n, 0);
  }
  void add_edge(int u, int v) {
    E[u] |= 1 << v, E[v] |= 1 << u;
  }
  int solve() {
    for (int i = 0; i < n; ++i)
      co[1 << i] = E[i] | (1 << i);
    co[0] = (1 << n) - 1;
    dp[0] = (n & 1) * 2 - 1;
    for (int i = 1; i < (1 << n); ++i) {
      int t = i & -i;
      dp[i] = -dp[i ^ t];
      co[i] = co[i ^ t] & co[t];
    }
    for (int i = 0; i < (1 << n); ++i)
      co[i] = (co[i] & i) == i;
    fwt(co, 1 << n);
    for (int ans = 1; ans < n; ++ans) {
      int sum = 0;
      for (int i = 0; i < (1 << n); ++i)
        sum += (dp[i] *= co[i]);
      if (sum) return ans;
    }
    return n;
  }
};
```

## 6.10 Floyd Warshall

```cpp
#define maxn 2005
vector<vector<int>> dis(maxn,vector<int>(maxn,9999999));
vector<vector<int>> mid(maxn,vector<int>(maxn,-1));
vector<vector<pair<int,int>>> graph;

void floyd_warshall(int n ){ // n is n nodes
  for(int i =0;i<n;i++){
        for(auto path:graph[i]){
            dis[i][path.second] = path.first;
        }
    }
  for (int i=0; i<n; i++)
    dis[i][i] = 0;
  for (int k=0; k<n; k++){
    for (int i=0; i<n; i++){
      for (int j=0; j<n; j++){
        if (dis[i][k] + dis[k][j] < dis[i][j] || dis[i][j
             ]==-1){
          dis[i][j] = dis[i][k] + dis[k][j];
          mid[i][j] = k;  // 由i點走到j點經過  k點
        }
            }
        }
    }
}
void find_path(int s, int t){   // 印出最短  徑
  if (mid[s][t] == -1) return;  // 沒有中繼點就結束
  find_path(s, mid[s][t]);    // 前半段最短  徑
  cout << mid[s][t];        // 中繼點
  find_path(mid[s][t], t);    // 後半段最短  徑
}
int main(){
    int n;
    floyd_warshall(n);
    for(int i =0;i<4;i++){
        for(int j = 0 ; j <4;j++)
            cout << dis[i][j]<<" ";
        cout << endl;
    }
    find_path(0,2);
}
```

## 6.11 Articulation Vertex

```
const int n = 9;
int t = 0;
vector<int> disc(n, -1);         // Discovery time
vector<int> low(n, -1);          // Low time
vector<int> parent_array(n, -1); // Parent in DFS tree
vector<bool> visited(n, false);
vector<bool> is_articulation(n, false);
vector<vector<int>> graph;
void dfs_articulation(int node, int parent){
    visited[node] = true;
    disc[node] = t;
    low[node] = t;
    t++;
    int children = 0;

    for (int neighbor : graph[node])
    {
        if (!visited[neighbor])
        {
            children++;
            parent_array[neighbor] = node;
            dfs_articulation(neighbor, node);
            low[node] = min(low[node], low[neighbor]);

            if (low[neighbor] >= disc[node] && parent != -1)
            {
                is_articulation[node] = true;
            }
        }
        else if (neighbor != parent)
        {
            low[node] = min(low[node], disc[neighbor]);
        }
    }

    if (parent == -1 && children > 1)
    {
        is_articulation[node] = true;
    }
}//call for() dfs(i,-1)
int main(){
    for (int i = 0; i < n; ++i) {
        if (!visited[i]) {
            dfs_articulation(i, -1);
        }
    }
    cout << "Articulation Points: ";
    for (int i = 0; i < n; ++i) {
        if (is_articulation[i]) {
            cout << i << " ";
        }
    }cout << endl;
}
```

## 6.12 Number of Maximal Clique

```
struct BronKerbosch { // 1-base
  int n, a[N], g[N][N];
  int S, all[N][N], some[N][N], none[N][N];
  void init(int _n) {
    n = _n;
    for (int i = 1; i <= n; ++i)
      for (int j = 1; j <= n; ++j) g[i][j] = 0;
  }
  void add_edge(int u, int v) {
    g[u][v] = g[v][u] = 1;
  }
  void dfs(int d, int an, int sn, int nn) {
    if (S > 1000) return; // pruning
    if (sn == 0 && nn == 0) ++S;
    int u = some[d][0];
    for (int i = 0; i < sn; ++i) {
      int v = some[d][i];
      if (g[u][v]) continue;
      int tsn = 0, tnn = 0;
      copy_n(all[d], an, all[d + 1]);
      all[d + 1][an] = v;
      for (int j = 0; j < sn; ++j)
        if (g[v][some[d][j]])
          some[d + 1][tsn++] = some[d][j];
      for (int j = 0; j < nn; ++j)
        if (g[v][none[d][j]])
          none[d + 1][tnn++] = none[d][j];
      dfs(d + 1, an + 1, tsn, tnn);
      some[d][i] = 0, none[d][nn++] = v;
    }
  }
  int solve() {
    iota(some[0], some[0] + n, 1);
    S = 0, dfs(0, 0, n, 0);
    return S;
  }
};
```

## 6.13 DominatorTree

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

## 6.14 Topological Sort

```
vector<vector<int>> graph;
vector<int> visit(10,0);
vector<int> order;
int n;
bool cycle;   // 記錄DFS的過程中是否偵測到環
void DFS(int i){ //reverse(order) is topo
  if (visit[i] == 1) {cycle = true; return;}
  if (visit[i] == 2) return;
  visit[i] = 1;
  for(auto to :graph[i])
        DFS(to);
  visit[i] = 2;
    order.push_back(i);
}//for() if(!vis[i])DFS(i)
int main() {
  for (int i=0; i<n; ++i){
    if (!visit[i])
      DFS(i);
    }
  if (cycle)
    cout << "圖上有環";
  else
    for (int i=n-1; i>=0; --i)
      cout << order[i];
}
```

## 6.15 Closest Pair

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

## 6.16 Minimum Mean Cycle

```
ll road[N][N]; // input here
struct MinimumMeanCycle {
  ll dp[N + 5][N], n;
  pll solve() {
    ll a = -1, b = -1, L = n + 1;
    for (int i = 2; i <= L; ++i)
      for (int k = 0; k < n; ++k)
        for (int j = 0; j < n; ++j)
          dp[i][j] =
            min(dp[i - 1][k] + road[k][j], dp[i][j]);
    for (int i = 0; i < n; ++i) {
      if (dp[L][i] >= INF) continue;
      ll ta = 0, tb = 1;
      for (int j = 1; j < n; ++j)
        if (dp[j][i] < INF &&
          ta * (L - j) < (dp[L][i] - dp[j][i]) * tb)
          ta = dp[L][i] - dp[j][i], tb = L - j;
      if (ta == 0) continue;
      if (a == -1 || a * tb > ta * b) a = ta, b = tb;
    }
    if (a != -1) {
      ll g = __gcd(a, b);
      return pll(a / g, b / g);
    }
    return pll(-1LL, -1LL);
  }
  void init(int _n) {
    n = _n;
    for (int i = 0; i < n; ++i)
      for (int j = 0; j < n; ++j) dp[i + 2][j] = INF;
  }
};
```

## 6.17 Planar

```
class Graph {
public:
    int V;
    vector<vector<int>> adj;
    Graph(int vertices) : V(vertices), adj(vertices) {}
    void addEdge(int u, int v) {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
};

bool containsSubgraph(const Graph& graph, const vector<int>&
     subgraph) {
    unordered_set<int> subgraphVertices(subgraph.begin(),
         subgraph.end());
    for (int vertex : subgraphVertices) {
        for (int neighbor : graph.adj[vertex]) {
            if (subgraphVertices.count(neighbor) == 0) {
                bool found = true;
                for (int v : subgraph) {
                    if (v != vertex && v != neighbor) {
                        if (graph.adj[v].size() < 3) {
                            found = false;
                            break;
                        }
                    }
                }
                if (found)
                    return true;
            }
        }
    }
    return false;
}

bool isPlanar(const Graph& graph) {
    // Subgraphs isomorphic to K  and K ,
    vector<int> k5 = {0, 1, 2, 3, 4};       // Vertices of K
    vector<int> k33a = {0, 1, 2};           // Vertices of K
         ,  (part A)
    vector<int> k33b = {3, 4, 5};           // Vertices of K
         ,  (part B)

    if (containsSubgraph(graph, k5) || containsSubgraph(graph
        , k33a) || containsSubgraph(graph, k33b)) {
        return false; // The graph is non-planar
    }
    return true; // The graph is planar
}
int main() {
    int vertices, edges;
    Graph graph(vertices);
    for (int i = 0; i < edges; ++i) {
        int u, v;cin >> u >> v;
        graph.addEdge(u, v);
    }
    if (isPlanar(graph)) {
        cout << "The graph is planar." << endl;
    } else {
        cout << "The graph is non-planar." << endl;
    }
}
```

## 6.18 Heavy Light Decomposition

```
int dep[N], pa[N], sz[N], nxt[N];
int id[N], rt[N];
int dfs(int u, int lst, int d = 0) {
  dep[u] = d;
  pa[u] = lst;
  sz[u] = 1;
  nxt[u] = -1;
  for (int v: g[u]) {
    if (v == lst) continue;
    sz[u] += dfs(v, u, d + 1);
    if (nxt[u] == -1 || sz[v] > sz[nxt[u]]) {
      nxt[u] = v;
    }
  }
  return sz[u];
}
int tn = 0;
void mapId(int u, int lst, int root) {
  id[u] = ++tn;
  rt[u] = root;
  if (~nxt[u]) mapId(nxt[u], u, root);
  for (int v: g[u]) {
    if (v == lst || v == nxt[u]) continue;
    mapId(v, u, v);
  }
}
void solve() {
  while (rt[a] != rt[b]) {
    if (dep[rt[a]] > dep[rt[b]]) swap(a, b);
    //...
    b = pa[rt[b]];
  }
  if (a != b) {
    if (id[a] > id[b]) swap(a, b);
    //...
  } else {
    //...
  }
}
```

## 6.19 Centroid Decomposition

```
int sz[maxn]{};
bool ok[maxn]{};
int get_subtree_size(int u, int lst) {
  sz[u] = 1;
  for (int v: g[u]) {
    if (v == lst || ok[v]) continue;
    sz[u] += get_subtree_size(v, u);
  }
  return sz[u];
}
int get_centroid(int u, int lst, int tree_size) {
  for (int v: g[u]) {
    if (v == lst || ok[v]) continue;
    if (2 * sz[v] >= tree_size) {
      return get_centroid(v, u, tree_size);
    }
  }
```

```cpp
  return u;
}
void centroid_decomp(int u = 1) { //1-based
  int centroid = get_centroid(u, u, get_subtree_size(u, u));
  //...
  ok[centroid] = 1;
  for (int v: g[centroid]) if (!ok[v]) {
    centroid_decomp(v);
  }
}
```

## 6.20 KM_O

```cpp
  // 二分圖最大權完美匹配
#define MAXN 100
#define INF INT_MAX
int g[MAXN][MAXN],lx[MAXN],ly[MAXN],slack_y[MAXN];
int px[MAXN],py[MAXN],match_y[MAXN],par[MAXN];
int n;
void adjust(int y){//把增廣 上所有邊反轉
  match_y[y]=py[y];
  if(px[match_y[y]]!=-2)
    adjust(px[match_y[y]]);
}
bool dfs(int x){//DFS找增廣
  for(int y=0;y<n;++y){
    if(py[y]!=-1)continue;
    int t=lx[x]+ly[y]-g[x][y];
    if(t==0){
      py[y]=x;
      if(match_y[y]==-1){
        adjust(y);
        return 1;
      }
      if(px[match_y[y]]!=-1)continue;
      px[match_y[y]]=y;
      if(dfs(match_y[y]))return 1;
    }else if(slack_y[y]>t){
      slack_y[y]=t;
      par[y]=x;
    }
  }
  return 0;
}
inline int km(){
  memset(ly,0,sizeof(int)*n);
  memset(match_y,-1,sizeof(int)*n);
  for(int x=0;x<n;++x){
    lx[x]=-INF;
    for(int y=0;y<n;++y){
      lx[x]=max(lx[x],g[x][y]);
    }
  }
  for(int x=0;x<n;++x){
    for(int y=0;y<n;++y)slack_y[y]=INF;
    memset(px,-1,sizeof(int)*n);
    memset(py,-1,sizeof(int)*n);
    px[x]=-2;
    if(dfs(x))continue;
    bool flag=1;
    while(flag){
      int cut=INF;
      for(int y=0;y<n;++y)
        if(py[y]==-1&&cut>slack_y[y])cut=slack_y[y];
      for(int j=0;j<n;++j){
        if(px[j]!=-1)lx[j]-=cut;
        if(py[j]!=-1)ly[j]+=cut;
        else slack_y[j]-=cut;
      }
      for(int y=0;y<n;++y){
        if(py[y]==-1&&slack_y[y]==0){
          py[y]=par[y];
          if(match_y[y]==-1){
            adjust(y);
            flag=0;
            break;
          }
          px[match_y[y]]=y;
          if(dfs(match_y[y])){
            flag=0;
            break;
          }
        }
      }
    }
  }
  int ans=0;
  for(int y=0;y<n;++y) if(g[match_y[y]][y]!=-INF) ans += g[match_y[y]][y];
  return ans;
}
```

## 6.21 Minimum Arborescence

```cpp
struct zhu_liu { // O(VE)
  struct edge {
    int u, v;
    ll w;
  };
  vector<edge> E; // 0-base
  int pe[N], id[N], vis[N];
  ll in[N];
  void init() { E.clear(); }
  void add_edge(int u, int v, ll w) {
    if (u != v) E.pb(edge{u, v, w});
  }
  ll build(int root, int n) {
    ll ans = 0;
    for (;;) {
      fill_n(in, n, INF);
      for (int i = 0; i < SZ(E); ++i)
        if (E[i].u != E[i].v && E[i].w < in[E[i].v])
          pe[E[i].v] = i, in[E[i].v] = E[i].w;
      for (int u = 0; u < n; ++u) // no solution
        if (u != root && in[u] == INF) return -INF;
      int cntnode = 0;
      fill_n(id, n, -1), fill_n(vis, n, -1);
      for (int u = 0; u < n; ++u) {
        if (u != root) ans += in[u];
        int v = u;
        while (vis[v] != u && !~id[v] && v != root)
          vis[v] = u, v = E[pe[v]].u;
        if (v != root && !~id[v]) {
          for (int x = E[pe[v]].u; x != v;
               x = E[pe[x]].u)
            id[x] = cntnode;
          id[v] = cntnode++;
        }
      }
      if (!cntnode) break; // no cycle
      for (int u = 0; u < n; ++u)
        if (!~id[u]) id[u] = cntnode++;
      for (int i = 0; i < SZ(E); ++i) {
        int v = E[i].v;
        E[i].u = id[E[i].u], E[i].v = id[E[i].v];
        if (E[i].u != E[i].v) E[i].w -= in[v];
      }
      n = cntnode, root = id[root];
    }
    return ans;
  }
};
```

## 6.22 Maximum Clique

```cpp
struct Maximum_Clique {
  typedef bitset<MAXN> bst;
  bst N[MAXN], empty;
  int p[MAXN], n, ans;
  void BronKerbosch2(bst R, bst P, bst X) {
    if (P == empty && X == empty)
      return ans = max(ans, (int)R.count()), void();
    bst tmp = P | X;
    int u;
    if ((R | P | X).count() <= ans) return;
    for (int uu = 0; uu < n; ++uu) {
      u = p[uu];
      if (tmp[u] == 1) break;
    }
    // if (double(clock())/CLOCKS_PER_SEC > .999)
    // return;
    bst now2 = P & ~N[u];
    for (int vv = 0; vv < n; ++vv) {
      int v = p[vv];
      if (now2[v] == 1) {
        R[v] = 1;
        BronKerbosch2(R, P & N[v], X & N[v]);
        R[v] = 0, P[v] = 0, X[v] = 1;
      }
    }
  }
  void init(int _n) {
    n = _n;
    for (int i = 0; i < n; ++i) N[i].reset();
  }
  void add_edge(int u, int v) {
    N[u][v] = N[v][u] = 1;
  }
  int solve() { // remember srand
    bst R, P, X;
    ans = 0, P.flip();
    for (int i = 0; i < n; ++i) p[i] = i;
    random_shuffle(p, p + n), BronKerbosch2(R, P, X);
    return ans;
  }
};
```

# 7 Math

## 7.1 Pollard Rho

```cpp
// does not work when n is prime
ll f(ll x,ll mod){ return add(mul(x,x,mod),1,mod); }
ll pollard_rho(ll n){
  if(!(n&1)) return 2;
  while(1){
    ll y=2,x=rand()%(n-1)+1,res=1;
    for(int sz=2;res==1;y=x,sz*=2)
      for(int i=0;i<sz&&res<=1;++i)
        x=f(x,n),res=__gcd(abs(x-y),n);
    if(res!=0&&res!=n) return res;
  }
}
```

## 7.2 Expression

```cpp
 * 支援處  四則運算的工具。給四則運算的字  ，檢查格式並計算其值。如果
 * 格式不合法，會丟出錯誤。複雜度 O(字  長度) 。支援的符號有四則運算
 * 和求餘數，先乘除後加減。可以使用括號、或前置正負號。數字開頭可以為
 * 零或禁止為零。可以兼容或禁止多重前置號 (  如 --1 視為 1 、+-+-1
 * 視為 -1) 。空字  視為不合法。運算範圍限於 long long 。如果試圖除
 * 以零或對零求餘也會丟出錯誤。
 */
void req(bool b) { if (!b) throw ""; }
const int B = 2; // 可以調整成 B 進位
class Expr {
   private:
    deque<char> src;
    Expr(const string& s) : src(s.begin(), s.end()) {}
    inline char top() {
        return src.empty() ? '\0' : src.front();
    }
    inline char pop() {
        char c = src.front(); src.pop_front(); return c;
    }
    ll n() {
        ll ret = pop() - '0';
        //  要禁止數字以 0 開頭，加上這
        // req(ret || !isdigit(top()));
        while (isdigit(top())) ret = B * ret + pop() - '0';
        return ret;
    }
    ll fac() {
        if (isdigit(top())) return n();
        if (top() == '-') { pop(); return -fac(); }
        if (top() == '(') {
            pop();
            ll ret = expr(1);
            req(pop() == ')');
            return ret;
        }
        //  要允許前置正號，加上這
        // if(top() == '+') { pop(); return fac(); }
        throw "";
    }
    ll term() {
        ll ret = fac(); char c = top();
        while (c == '*' || c == '/' || c == '%') {
            pop();
            if (c == '*') ret *= fac();
            else {
                ll t = fac(); req(t);
                if (c == '/') ret /= t; else ret %= t;
            }
            c = top();
        } return ret;
    }
    ll expr(bool k) {
        ll ret = term();
        while (top() == '+' || top() == '-')
            if (pop() == '+') ret += term();
            else ret -= term();
        req(top() == (k ? ')' : '\0'));
        return ret;
    }
   public:
    // 給定數學運算的字  ，求其值。  格式不合法，丟出錯誤。
    static ll eval(const string& s) {
        //  要禁止多重前置號，加上這四
        // req(s.find("--") == -1); // 禁止多重負號
        // req(s.find("-+") == -1);
        // req(s.find("+-") == -1);
        // req(s.find("++") == -1);
        return Expr(s).expr(0);
    }
};
```

## 7.3 Miller Robin

```cpp
// n < 4,759,123,141       3 : 2, 7, 61
// n < 1,122,004,669,633   4 : 2, 13, 23, 1662803
// n < 3,474,749,660,383   6 : pirmes <= 13
// n < 2^64                7 : 2, 325, 9375, 28178, 450775, 9780504, 1795265022

//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;
}
```

## 7.4 整數分塊

```cpp
for(int l=1,r;l<=n;l=r+1){
    r=n/(n/l);
    ans+=(r-l+1)*(n/l);
}

// sum is the prefix of mobius function
// 求 1<=x<=n, 1<=y<=m 且 gcd(x,y)==1 的二元组数  。
for(int l=1,r;l<=min(n,m);l=r+1){
    r=min(n/(n/l),m/(m/l));
    ans+=(sum[r]-sum[l-1])*(n/l)*(m/l);
}
```

## 7.5 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  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]
}
```

## 7.6 Karatsuba

```
// N is power of 2
template<typename Iter>
void DC(int N, Iter tmp, Iter A, Iter B, Iter res){
    fill(res,res+2*N,0);
    if (N<=32){
        for (int i=0; i<N; i++)
            for (int j=0; j<N; j++)
                res[i+j] += A[i]*B[j];
        return;
    }
    int n = N/2;
    auto a = A+n, b = A;
    auto c = B+n, d = B;
    DC(n,tmp+N,a,c,res+2*N);
    for (int i=0; i<N; i++){
        res[i+N] += res[2*N+i];
        res[i+n] -= res[2*N+i];
    }
    DC(n,tmp+N,b,d,res+2*N);
    for (int i=0; i<N; i++){
        res[i] += res[2*N+i];
        res[i+n] -= res[2*N+i];
    }
    auto x = tmp;
    auto y = tmp+n;
    for (int i=0; i<n; i++) x[i] = a[i]+b[i];
    for (int i=0; i<n; i++) y[i] = c[i]+d[i];
    DC(n,tmp+N,x,y,res+2*N);
    for (int i=0; i<N; i++)
        res[i+n] += res[2*N+i];
}
// DC(1<<16,tmp.begin(),A.begin(),B.begin(),res.begin());
```

## 7.7 fpow

```
ll fpow(ll b, ll p, ll mod) {
  ll res = 1;
  while (p) {
    if (p & 1) res = res * b % mod;
    b = b * b % mod, p >>= 1;
  }
  return res;
}
```

## 7.8 Big Number

```
template<typename T>
inline string to_string(const T& x){
  stringstream ss;
  return ss<<x,ss.str();
}
struct bigN:vector<ll>{
  const static int base=1000000000,width=log10(base);
  bool negative;
  bigN(const_iterator a,const_iterator b):vector<ll>(a,b){}
  bigN(string s){
    if(s.empty())return;
    if(s[0]=='-')negative=1,s=s.substr(1);
    else negative=0;
    for(int i=int(s.size())-1;i>=0;i-=width){
      ll t=0;
      for(int j=max(0,i-width+1);j<=i;++j)
        t=t*10+s[j]-'0';
      push_back(t);
    }
    trim();
  }
  template<typename T>
    bigN(const T &x):bigN(to_string(x)){}
  bigN():negative(0){}
  void trim(){
    while(size()&&!back())pop_back();
    if(empty())negative=0;
  }
  void carry(int _base=base){
    for(size_t i=0;i<size();++i){
      if(at(i)>=0&&at(i)<_base)continue;
      if(i+1u==size())push_back(0);
      int r=at(i)%_base;
      if(r<0)r+=_base;
      at(i+1)+=(at(i)-r)/_base,at(i)=r;
    }
  }
  int abscmp(const bigN &b)const{
    if(size()>b.size())return 1;
    if(size()<b.size())return -1;
    for(int i=int(size())-1;i>=0;--i){
      if(at(i)>b[i])return 1;
      if(at(i)<b[i])return -1;
    }
    return 0;
  }
  int cmp(const bigN &b)const{
    if(negative!=b.negative)return negative?-1:1;
    return negative?-abscmp(b):abscmp(b);
  }
  bool operator<(const bigN&b)const{return cmp(b)<0;}
  bool operator>(const bigN&b)const{return cmp(b)>0;}
  bool operator<=(const bigN&b)const{return cmp(b)<=0;}
  bool operator>=(const bigN&b)const{return cmp(b)>=0;}
  bool operator==(const bigN&b)const{return !cmp(b);}
  bool operator!=(const bigN&b)const{return cmp(b)!=0;}
  bigN abs()const{
    bigN res=*this;
    return res.negative=0, res;
  }
  bigN operator-()const{
    bigN res=*this;
    return res.negative=!negative,res.trim(),res;
  }
  bigN operator+(const bigN &b)const{
    if(negative)return -(-(*this)+(-b));
    if(b.negative)return *this-(-b);
    bigN res=*this;
    if(b.size()>size())res.resize(b.size());
    for(size_t i=0;i<b.size();++i)res[i]+=b[i];
    return res.carry(),res.trim(),res;
  }
  bigN operator-(const bigN &b)const{
    if(negative)return -(-(*this)-(-b));
    if(b.negative)return *this+(-b);
    if(abscmp(b)<0)return -(b-(*this));
    bigN res=*this;
    if(b.size()>size())res.resize(b.size());
    for(size_t i=0;i<b.size();++i)res[i]-=b[i];
    return res.carry(),res.trim(),res;
  }
  bigN operator*(const bigN &b)const{
    bigN res;
    res.negative=negative!=b.negative;
    res.resize(size()+b.size());
    for(size_t i=0;i<size();++i)
      for(size_t j=0;j<b.size();++j)
        if((res[i+j]+=at(i)*b[j])>=base){
          res[i+j+1]+=res[i+j]/base;
          res[i+j]%=base;
        }
    return res.trim(),res;
  }
  bigN operator/(const bigN &b)const{
    int norm=base/(b.back()+1);
    bigN x=abs()*norm;
    bigN y=b.abs()*norm;
    bigN q,r;
    q.resize(x.size());
    for(int i=int(x.size())-1;i>=0;--i){
      r=r*base+x[i];
      int s1=r.size()<=y.size()?0:r[y.size()];
      int s2=r.size()<y.size()?0:r[y.size()-1];
      int d=(ll(base)*s1+s2)/y.back();
      r=r-y*d;
      while(r.negative)r=r+y,--d;
      q[i]=d;
    }
    q.negative=negative!=b.negative;
    return q.trim(),q;
  }
  bigN operator%(const bigN &b)const{
    return *this-(*this/b)*b;
  }
  friend istream& operator>>(istream &ss,bigN &b){
    string s;
    return ss>>s, b=s, ss;
  }
  friend ostream& operator<<(ostream &ss,const bigN &b){
    if(b.negative)ss<<'-';
    ss<<(b.empty()?0:b.back());
    for(int i=int(b.size())-2;i>=0;--i)
      ss<<setw(width)<<setfill('0')<<b[i];
    return ss;
  }
  template<typename T>
    operator T(){
      stringstream ss;
      ss<<*this;
      T res;
      return ss>>res,res;
    }
};
```

## 7.9 modinv

```
// 解 (ax == 1) mod p 。p 必須是質數，a 是正整數。
ll modinv(ll a, ll p) {
    if (p == 1) return 0;
    ll pp = p, y = 0, x = 1;
    while (a > 1) {
        ll q = a / p, t = p;
        p = a % p, a = t, t = y, y = x - q * y, x = t;
    }
    if (x < 0) x += pp;
    return x;
}
// 解 (ax == b) mod p 。p 必須是質數，a 和 b 是正整數。
ll modinv(ll a, ll b, ll p) {
    ll ret = modinv(a, p);
    return ret * b % p;
}
```

## 7.10 Matrix

- 旋轉矩陣

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{1}$$

- 縮放矩陣

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{2}$$

- 反射矩陣

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 2u_x^2 - 1 & 2u_x u_y \\ 2u_x u_y & 2u_y^2 - 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{3}$$

- 正投影

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} u_x^2 & u_x u_y \\ u_x u_y & u_y^2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{4}$$

## 7.11 Discrete Sqrt

```
int order(ll b, ll p) {
    if (__gcd(b, p) != 1) return -1;
    int ret = 2;
    while (++ret)
        if (fastpow(b, ret, p) == 1) break;
    return ret;
}
// 把 fastpow 也抄過 ，會用到。
// 問 (x^2 = y) mod p 的解。回傳 -1 表示 x 無解。
ll dsqrt(ll y, ll p) {
    if (__gcd(y, p) != 1) return -1;
    if (fastpow(y, (p - 1 / 2), p) == p - 1) return -1;
    int e = 0;
    ll s = p - 1;
    while (!(s & 1)) s >>= 1, e++;
    int q = 2;
    while (1)
        if (fastpow(q, (p - 1) / 2, p) == p - 1)
            break;
        else q++;
    ll x = fastpow(y, (s + 1) / 2, p);
    ll b = fastpow(y, s, p);
    ll g = fastpow(q, s, p);
    while (1) {
        int m;
        for (m = 0; m < e; m++) {
            int o = order(p, b);
            if (o == -1) return -1;
            if (o == fastpow(2, m, p)) break;
        }
        if (m == 0) return x;
        x = x * fastpow(g, fastpow(2, e - m - 1), p) % p;
        g = fastpow(g, fastpow(2, e - m), p), p);
        b = b * g % p;
        if (b == 1) return x;
        e = m;
    }
}
```

## 7.12 Euler Totient Function

```
ll phi[maxn];
for (int i = 0; i < maxn; ++i) {
  phi[i] = i;
}
for (int i = 2; i < maxn; ++i) if (phi[i] == i) {
  phi[i] = i - 1; //prime
  for (int j = 2; i*j < maxn; ++j) { //overflow
    phi[i*j] = (phi[i*j] / i) * (i - 1);
  }
}
```

## 7.13 Discretize

```
void __solve(vector<int> u){
    sort(ALL(u));
    u.resize(unique(ALL(u))-u.begin());
    vi nums(u.size()+50,0);
    vi discrete(u.size()+50);

    for(int i =0;i<u.size();i++){
        discrete[i] = upper_bound(ALL(u),u[i])-u.begin();
        cout << u[i]<<" "<<discrete[i]<<endl;
    }
}
signed main(){
    vector<int> u = {4,13,14,152312,12314,31234,123,3};
    __solve(u);
}
```

## 7.14 Fraction

```
#define cfl(str) (const frac& f) const { return str; }
#define cll(str) (ll l) const { return str; }
#define lfl(str) (ll l, const frac& f) { return str; }
#define ff inline frac operator
#define bb inline bool operator
#define fff inline friend frac operator
#define fbb inline friend bool operator

class frac {
   private: ll x, y;
   public:
    frac() : x(0), y(1) {}
    frac(ll v) : x(v), y(1) {}
    frac(ll xx, ll yy, bool f = 0) : x(xx), y(yy) {
        assert(y != 0);
        if (!f) {
            ll g = __gcd(x, y);
            x /= g, y /= g;
            if (y < 0) x *= -1, y *= -1;
        }
    }
    // 以下斟酌使用，不必全抄
    ff = (ll l) { return frac(l); }
    ff - () const { return frac(-x, y, 1); }
    ff ! () const {  // 倒數
        return x > 0 ? frac(y, x, 1) : frac(-y, -x, 1);
    }

    bb >  cfl(x * f.y > y * f.x)
    bb <  cfl(x * f.y < y * f.x)
    bb <= cfl(x * f.y <= y * f.x)
    bb >= cfl(x * f.y >= y * f.x)
    bb == cfl(x == f.x && y == f.y)
    bb != cfl(x != f.x || y != f.y)
    ff +  cfl(frac(x * f.y + y * f.x, y * f.y))
    ff -  cfl(frac(x * f.y - y * f.x, y * f.y))
    ff *  cfl(frac(x * f.x, y * f.y))
    ff /  cfl(frac(x * f.y, y * f.x))

    bb >  cll(x > l * y)
    bb <  cll(x < l * y)
    bb >= cll(x >= l * y)
    bb <= cll(x <= l * y)
    bb == cll(x == l * y)
    bb != cll(x != l * y)
    ff +  cll(frac(x + l * y, y))
    ff -  cll(frac(x - l * y, y))
    ff *  cll(frac(l * x, y))
    ff /  cll(frac(x, l * y))

    fbb <  lfl(f > l)
    fbb >  lfl(f < l)
    fbb <= lfl(f >= l)
    fbb >= lfl(f <= l)
    fbb == lfl(f == l)
    fbb != lfl(f != l)
    fff +  lfl(f + l)
    fff -  lfl(-f + l)
    fff *  lfl(f * l)
    fff /  lfl(!f * l)

    inline operator double() { return (double)x / y; }
    inline friend frac abs(const frac& f) {
        return frac(abs(f.x), f.y, 1);
    }
    inline friend ostream& operator <<
         (ostream & out, const frac& f) {
        out << f.x;
        if (f.y != 1) out << '/' << f.y;
        return out;
    }
};
```

## 7.15 Floor Ceil

```
int floor(int a,int b){
  return a/b-(a%b&&a<0^b<0);
}
int ceil(int a,int b){
  return a/b+(a%b&&a<0^b>0);
}
```

## 7.16 extGCD

```
int extgcd(int a,int b,int &x,int &y){//a*x +b*y = 1
    if(b==0){
        x = 1;
        y = 0;
        return a;  //到達遞歸邊界開始向上一層返回
    }
    int r = extgcd(b,a%b,x,y);
    int temp=y;    //把x y變成上一層的
    y = x - (a / b) * y;
    x = temp;
    return r;     //得到a b的最大公因數
}
int main(){
    int a = 55,b = 80;
    int x,y;//a*x+b*y = 1;
    int GCD = extgcd(a,b,x,y);
}
```

## 7.17 FFT

```
//OI Wiki
#include <complex>
using cd = complex<double>;
const double PI = acos(-1);
void change(vector<cd> &y) {
  vector<int> rev(y.size());
  for (int i = 0; i < y.size(); ++i) {
    rev[i] = rev[i >> 1] >> 1;
    if (i & 1) {
      rev[i] |= y.size() >> 1;
    }
  }
  for (int i = 0; i < y.size(); ++i) {
    if (i < rev[i]) {
      swap(y[i], y[rev[i]]);
    }
  }
}
void fft(vector<cd> &y, bool inv) {
  change(y);
  for (int h = 2; h <= y.size(); h <<= 1) {
    cd wn(cos(2 * PI / h), sin(2 * PI / h));
    for (int j = 0; j < y.size(); j += h) {
      cd w(1, 0);
      for (int k = j; k < j + h / 2; ++k) {
        cd u = y[k];
        cd t = w * y[k + h / 2];
        y[k] = u + t;
        y[k + h / 2] = u - t;
        w = w * wn;
      }
    }
  }
  if (inv) {
    reverse(begin(y) + 1, end(y));
    for (int i = 0; i < y.size(); ++i) {
      y[i] /= y.size();
    }
  }
}
void solve() {
  int n;
  int m = 1 << (__lg(n) + 1); //power of 2
  vector<cd> a(m), b(m);
  //...
  fft(a, 0);
  fft(b, 0);
  vector<cd> c(m);
  for (int i = 0; i < m; ++i) {
    c[i] = a[i] * b[i];
  }
  fft(c, 1);
  for (auto p: c) {
    int ans = int(p.real() + 0.25);
  }
}
```

## 7.18 Simplex Algorithm

```
const int MAXN = 111;
const int MAXM = 111;
const double eps = 1E 10;
double a[MAXN][MAXM], b[MAXN], c[MAXM], d[MAXN][MAXM];
double x[MAXM];
int ix[MAXN + MAXM]; // !!! array all indexed from 0
// max{cx} subject to {Ax<=b,x>=0}
// n: constraints, m: vars !!!
// x[] is the optimal solution vector
// usage :
// value = simplex(a, b, c, N, M);
double simplex(double a[MAXN][MAXM], double b[MAXN],
               double c[MAXM], int n, int m){
    ++m;
    int r = n, s = m   1;
    memset(d, 0, sizeof(d));
    for (int i = 0; i < n + m; ++i)
        ix[i] = i;
    for (int i = 0; i < n; ++i){
        for (int j = 0; j < m   1; ++j)
            d[i][j] =  a[i][j];
        d[i][m   1] = 1;
        d[i][m] = b[i];
        if (d[r][m] > d[i][m])
            r = i;
    }
    for (int j = 0; j < m   1; ++j)
        d[n][j] = c[j];
    d[n + 1][m   1] =  1;
    for (double dd;;){
        if (r < n){
            int t = ix[s];
            ix[s] = ix[r + m];
            ix[r + m] = t;
            d[r][s] = 1.0 / d[r][s];
            for (int j = 0; j <= m; ++j)
                if (j != s)
                    d[r][j] *=  d[r][s];
            for (int i = 0; i <= n + 1; ++i)
                if (i != r)
                {
                    for (int j = 0; j <= m; ++j)
                        if (j != s)
                            d[i][j] += d[r][j] * d[i][s];
                    d[i][s] *= d[r][s];
                }
        }
        r =  1;
        s =  1;
        for (int j = 0; j < m; ++j)
            if (s < 0 || ix[s] > ix[j])
            {
                if (d[n + 1][j] > eps ||
                    (d[n + 1][j] >  eps && d[n][j] > eps))
                    s = j;
            }
        if (s < 0)
            break;
        for (int i = 0; i < n; ++i)
            if (d[i][s] <  eps)
            {
                if (r < 0 ||
                    (dd = d[r][m] / d[r][s]   d[i][m] / d[i][
                        s]) <  eps ||
                    (dd < eps && ix[r + m] > ix[i + m]))
                    r = i;
            }
        if (r < 0)
            return  1; // not bounded
    }
    if (d[n + 1][m] <  eps)
        return  1; // not executable
    double ans = 0;
    for (int i = 0; i < m; i++)
        x[i] = 0;
    for (int i = m; i < n + m; ++i){ // the missing
        enumerated x[i] = 0 if (ix[i] < m   1){
            ans += d[i   m][m] * c[ix[i]];
            x[ix[i]] = d[i m][m];
        }
    }
    return ans;
}
```

## 7.19 mu

```
int mu[MAXN];
bool isnp[MAXN];
vector<int> primes;
void init(int n){
    mu[1] = 1;
    for (int i = 2; i <= n; i++){
        if (!isnp[i])
            primes.push_back(i), mu[i] = -1; // 质数为-1
        for (int p : primes){
            if (p * i > n)
                break;
            isnp[p * i] = 1;
            if (i % p == 0){
                mu[p * i] = 0; // 有平方因数为0
                break;
            }
            else
                mu[p * i] = mu[p] * mu[i]; // 互质，  用积性
                    函数性质
        }
    }
}
```

## 7.20 Chinese Remainder

```
// Chinese remainder theorem (special case): find z such that
// z % x = a, z % y = b.  Here, z is unique modulo M = lcm(x,
    y).
// Return (z,M).  On failure, M = -1.
PII chinese_remainder_theorem(int x, int a, int y, int b) {
  int s, t;
  int d = extended_euclid(x, y, s, t);
  if (a%d != b%d) return make_pair(0, -1);
  return make_pair(mod(s*b*x+t*a*y,x*y)/d, x*y/d);
}

// Chinese remainder theorem: find z such that
// z % x[i] = a[i] for all i.  Note that the solution is
// unique modulo M = lcm_i (x[i]).  Return (z,M).  On
// failure, M = -1.  Note that we do not require the a[i]'s
// to be relatively prime.
PII chinese_remainder_theorem(const VI &x, const VI &a) {
  PII ret = make_pair(a[0], x[0]);
  for (int i = 1; i < x.size(); i++) {
    ret = chinese_remainder_theorem(ret.second, ret.first, x[
        i], a[i]);
    if (ret.second == -1) break;
  }
  return ret;
}

// computes x and y such that ax + by = c; on failure, x = y
    =-1
void linear_diophantine(int a, int b, int c, int &x, int &y)
    {
  int d = gcd(a,b);
  if (c%d) {
    x = y = -1;
  } else {
    x = c/d * mod_inverse(a/d, b/d);
    y = (c-a*x)/b;
  }
}
```

## 7.21 Numbers

### 7.21.1 Bernoulli numbers

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

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

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

### 7.21.2 Stirling numbers of the second kind

Partitions of $n$ distinct elements into exactly $k$ groups.

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

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

## 7.22 Prime Count

```
int64_t PrimeCount(int64_t n) {
  if (n <= 1) return 0;
  const int v = sqrt(n);
  vector<int> smalls(v + 1);
  for (int i = 2; i <= v; ++i) smalls[i] = (i + 1) / 2;
  int s = (v + 1) / 2;
  vector<int> roughs(s);
  for (int i = 0; i < s; ++i) roughs[i] = 2 * i + 1;
  vector<int64_t> larges(s);
  for (int i = 0; i < s; ++i) larges[i] = (n / (2 * i + 1) +
      1) / 2;
  vector<bool> skip(v + 1);
  int pc = 0;
  for (int p = 3; p <= v; ++p) {
    if (smalls[p] > smalls[p - 1]) {
      int q = p * p;
      pc++;
      if (1LL * q * q > n) break;
      skip[p] = true;
      for (int i = q; i <= v; i += 2 * p) skip[i] = true;
      int ns = 0;
      for (int k = 0; k < s; ++k) {
        int i = roughs[k];
        if (skip[i]) continue;
        int64_t d = 1LL * i * p;
        larges[ns] = larges[k] - (d <= v ? larges[smalls[d] -
             pc] : smalls[n / d]) + pc;
        roughs[ns++] = i;
      }
      s = ns;
      for (int j = v / p; j >= p; --j) {
        int c = smalls[j] - pc;
        for (int i = j * p, e = min(i + p, v + 1); i < e; ++i
            ) smalls[i] -= c;
      }
    }
  }
  for (int k = 1; k < s; ++k) {
    const int64_t m = n / roughs[k];
    int64_t s = larges[k] - (pc + k - 1);
    for (int l = 1; l < k; ++l) {
      int p = roughs[l];
      if (1LL * p * p > m) break;
      s -= smalls[m / p] - (pc + l - 1);
    }
    larges[0] -= s;
  }
  return larges[0];
}
```

## 7.23 Multiple Power

```
//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;
```

```cpp
  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;
}
```

## 7.24 Determinant

```cpp
double determinant(vector<vector<double>>& matrix) {
    int n = matrix.size();
    if (n == 1) {
        return matrix[0][0];
    }
    double det = 0;
    for (int i = 0; i < n; i++) {
        vector<vector<double>> submatrix(n - 1, vector<double>(n - 1));
        for (int j = 1; j < n; j++) {
            for (int k = 0; k < n; k++) {
                if (k < i) {
                    submatrix[j - 1][k] = matrix[j][k];
                } else if (k > i) {
                    submatrix[j - 1][k - 1] = matrix[j][k];
                }
            }
        }
        double submatrix_det = determinant(submatrix);// 遞迴計算  式
        det += (i % 2 == 0 ? 1 : -1) * matrix[0][i] * submatrix_det;// 累積  式值
    }
    return det;
}
```

# 8 Misc

## 8.1 Mo's Algorithm

```cpp
struct Query {
  int L, R;
  //...
};
vector<Query> query;
void solve() { //K = n / sqrt(q)
  sort(iter(query), [&](Query &a, Query &b) {
    if (a.L / K != b.L / K) return a.L < b.L;
    return a.L / K % 2 ? a.R < b.R : a.R > b.R;
  });
  int L = 0, R = 0;
  for (auto x: query) {
    while (R < x.R) add(arr[++R]);
    while (L > x.L) add(arr[--L]);
    while (R > x.R) sub(arr[R--]);
    while (L < x.L) sub(arr[L++]);
    //...
  }
}
```

## 8.2 pbds

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

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

int32_t main() {
  ordered_set<int64_t> rbt;
  // .insert(x); .erase(x)
  // .lower_bound(x); .upper_bound(x): iter
  // .find_by_order(k): find k-th small value(iter)
  // .order_of_key(x): return x is k-th big
  // .join(rbt2): merge with no mutiple same element
  // .split(key, rbt2): rbt keeps value <= key, others to rbt2
}
```

## 8.3 Misc

```cpp
mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
int randint(int lb, int ub){
  return uniform_int_distribution<int>(lb, ub)(rng);
} //static unsigned x = 19; ++(x *= 0xdefaced);

#define SECs ((double)clock() / CLOCKS_PER_SEC)

struct KeyHasher{
  size_t operator()(const Key& k) const {
    return k.first + k.second * 100000;
  }
};
typedef unordered_map<Key,int,KeyHasher> map_t;

__lg
__gcd

int __builtin_ffs(unsigned int x)
int __builtin_ffsl(unsigned long)
int __builtin_ffsll(unsigned long long)
返回右起第一個1的位置
Returns one plus the index of the least significant 1-bit of x, or if x is zero, returns zero.

int __builtin_clz(unsigned int x)
int __builtin_clzl(unsigned long)
int __builtin_clzll(unsigned long long)
返回左起第一個1之前0的個數
Returns the number of leading 0-bits in x, starting at the most significant bit position. If x is 0, the result is undefined.

int __builtin_ctz(unsigned int x)
int __builtin_ctzl(unsigned long)
int __builtin_ctzll(unsigned long long)
返回右起第一個1之後的0的個數
Returns the number of trailing 0-bits in x, starting at the least significant bit position. If x is 0, the result is undefined.

int __builtin_popcount(unsigned int x)
int __builtin_popcountl(unsigned long)
int __builtin_popcountll(unsigned long long)
返回1的個數
Returns the number of 1-bits in x.

int __builtin_parity(unsigned int x)
int __builtin_parityl(unsigned long)
int __builtin_parityll(unsigned long long)
返回1的個數的奇偶性(1的個數 mod 2的值)
Returns the parity of x, i.e. the number of 1-bits in x modulo 2.
```

# 9 String

## 9.1 Hashing

```cpp
const ll P = 401, M = 998244353;

ll hashes[10005], modp[10005];
ll hashp(string s, bool saveval) {
  ll val = 0;
  int index = 0;
  for (char c: s) {
    val = ((val * P) % M + c) % M;
    if (saveval) hashes[index++] = val;
  }
  return val;
}
```

```cpp
void init(int base, int exp) {
  ll b = 1;
  modp[0] = 1;
  for (int i = 0; i < exp; i++) {
    b = (b * base) % M;
    modp[i + 1] = b;
  }
}
ll subseq(int l, int r) { //[l, r]
  if (l == 0) return hashes[r];
  return ((hashes[r] - hashes[l-1] * modp[r-l+1]) % M + M) % M;
}
```

## 9.2 Trie

```cpp
struct node {
  int ch[26]{};
  int cnt = 0;
};
struct Trie {
  vector<node> t;
  void init() {
    t.clear();
    t.emplace_back(node());
  }
  void insert(string s) {
    int ptr = 0;
    for (char i: s) {
      if (!t[ptr].ch[i - 'a']) {
        t[ptr].ch[i - 'a'] = (int)t.size();
        t.emplace_back(node());
      }
      ptr = t[ptr].ch[i - 'a'];
    }
    t[ptr].cnt++;
  }
} trie;
```

## 9.3 Zvalue

```cpp
vector<int> Zvalue(string &s) { //t + # + s
  vector<int> Z(s.size());
  int x = 0, y = 0;
  for (int i=0; i<s.size(); ++i) {
    Z[i] = max(0, min(y - i + 1, Z[i - x]));
    while (i + Z[i] < s.size() && s[Z[i]] == s[i + Z[i]])
      x = i, y = i + Z[i], ++Z[i];
  }
  return Z;
}
```

## 9.4 KMP

```cpp
int F[maxn]{};
vector<int> match(string& s, string& t) {
  int p = F[0] = -1;
  for (int i = 1; i < t.size(); ++i) {
    while (p != -1 && t[p + 1] != t[i]) p = F[p];
    if (t[p + 1] == t[i]) ++p;
    F[i] = p;
  }
  p = -1;
  vector<int> v;
  for (int i = 0; i < s.size(); ++i) {
    while (p != -1 && t[p + 1] != s[i]) p = F[p];
    if (t[p + 1] == s[i]) ++p;
    if (p == t.size() - 1) v.push_back(i - p), p = F[p];
  }
  return v; //0-based
}
```

## 9.5 Manacher

```cpp
int z[maxn * 2]{};
int manacher(string& s) {
  string t = "#";
  for (char c: s) t += c, t += '#';
  int l = 0, r = 0, ans = 0; //l: mid, r: right
  for (int i = 1; i < t.size(); ++i) {
    z[i] = (r > i ? min(z[2 * l - i], r - i) : 1);
    while (i - z[i] >= 0 && i + z[i] < t.size()) {
      if (t[i - z[i]] == t[i + z[i]])
        ++z[i];
      else
        break;
    }
    if (i + z[i] > r) r = i + z[i], l = i;
  }
  for (int i = 1; i < t.size(); ++i) ans = max(ans, z[i] - 1);
  string res;
  for (int i = 1; i < t.size(); ++i) if (ans == z[i] - 1) {
    for (int j = i - ans + 1; j < i + ans; ++j) if (t[j] != '#') {
      res += t[j];
    }
    break;
  }
  return ans;
}
```

# 10 Tree

## 10.1 LCA

```cpp
int n, logn,t=0;
vector<vector<int>> graph;
vector<vector<int>> ancestor;
vector<int> tin, tout;
void dfs(int x){
    tin[x] = t++;
  for(auto y:graph[x]){
        if(y!= ancestor[x][0]){
            ancestor[y][0] = x;
            dfs(y);
        }
    }
    tout[x] = t++;
}
bool is_ancestor(int x, int y){
  return tin[x] <= tin[y] && tout[x] >= tout[y];
}
void table(){
  for (int i=1; i<logn; i++)// 上  輩祖先、上四輩祖先、上八輩祖先、……
    for (int x=0; x<n; ++x)
      ancestor[x][i] = ancestor[ancestor[x][i-1]][i-1];
}

int kth_ancestor(int x, int k){
  for (int i=0; i<logn; i++)// k拆解成二進位位數，找到第k祖先。不斷上升逼近之。
    if (k & (1<<i))
      x = ancestor[x][i];
  return x;
}

void rooted_tree(int root){// build the tree with root at "root"
  ancestor[root][0] = root;
  dfs(root);
  table();
}
int LCA(int x,int y){
    if (is_ancestor(x, y)) return x;
  if (is_ancestor(y, x)) return y;
    for (int i=logn-1; i>=0; i--)
    if (!is_ancestor(ancestor[x][i], y))
      x = ancestor[x][i];
  return ancestor[x][0];
}
int main(){
    graph = {
        {1,2},
        {3},
        {5,6},
        {7},
        {},
        {},
        {},
        {8},
        {4},
    };
    n = 9;
    logn = ceil(log2(n));
    ancestor.resize(n,vector<int>(logn));
    tin.resize(n);
    tout.resize(n);

    rooted_tree(0);
    while(true){
        int a,b;
        cin >>a>>b;
        cout <<LCA(a,b)<<endl;;
    }
}
int main(){
    n = 9;
    logn = ceil(log2(n));
    ancestor.resize(n,vector<int>(logn));
    tin.resize(n);
    tout.resize(n);
    rooted_tree(0);
    while(true){
        int a,b;
        cin >>a>>b;
        cout <<LCA(a,b)<<endl;;
    }
}
```

## 10.2 Diameter

```cpp
vector<vector<int>> graph;
int diameter = 0;
int dfs(int start, int parent){
    int h1 = 0, h2 = 0;
    for (auto child : graph[start]){
        if (child != parent){
            int h = dfs(child, start) + 1;
            if (h > h1){
                h2 = h1;
                h1 = h;
            }
            else if (h > h2){
                h2 = h;
            }
        }
    }
    diameter = max(diameter, h1 + h2);
    return h1;
}
// call diameter
int main(){
    dfs(0,-1);
    cout << diameter<<endl;
}
```

## 10.3 Radius

```cpp
// Perform DFS to find the farthest node and its distance from the given node
pair<int, int> dfs(int node, int distance, vector<bool> & visited, const vector<vector<int>> &adj_list){
    visited[node] = true;
    int max_distance = distance;
    int farthest_node = node;

    for (int neighbor : adj_list[node]){
        if (!visited[neighbor]){
            auto result = dfs(neighbor, distance + 1, visited, adj_list);
            if (result.first > max_distance){
                max_distance = result.first;
                farthest_node = result.second;
            }
        }
    }

    return make_pair(max_distance, farthest_node);
}

// Calculate the radius of the tree using DFS
int tree_radius(const vector<vector<int>> &adj_list){
    int num_nodes = adj_list.size();
    vector<bool> visited(num_nodes, false);

    // Find the farthest node from the root (node 0)
    auto farthest_result = dfs(0, 0, visited, adj_list);

    // Reset visited array
    fill(visited.begin(), visited.end(), false);

    // Calculate the distance from the farthest node
    int radius = dfs(farthest_result.second, 0, visited, adj_list).first;

    return radius;
}
int main() {
    vector<vector<int>> adj_list;
    int radius = tree_radius(adj_list);
    cout << "Tree radius: " << radius << endl;
    return 0;
}
```

## 10.4 Spanning Tree

```cpp
const int V = 100, E = 1000;
struct Edge {int a, b, c;} e[E];  // edge list
bool operator<(Edge e1, Edge e2) {return e1.c < e2.c;}

int p[V];
void init() {for (int i=0; i<V; ++i) p[i] = i;}
int find(int x) {return x == p[x] ? x : (p[x] = find(p[x]));}
void merge(int x, int y) {p[find(x)] = find(y);}

void Kruskal(){
  init();
  sort(e, e+E);
  int i, j;
  for (i = 0, j = 0; i < V-1 && j < E; ++i){
    while (find(e[j].a) == find(e[j].b)) j++;
    merge(e[j].a, e[j].b);
    cout << "起點：" << e[j].a<< "終點：" << e[j].b<< "權重：" << e[j].c;
    j++;
  }
  if (i == V-1) cout << "得到最小生成樹";
  else          cout << "得到最小生成森   ";
}
```

# NYCU_Segmentree Codebook

## Contents