# Code Template for ACM-ICPC

zzuzxy

September 26, 2018

# Contents

# 1 1 数据结构

## 1.1 二叉树

```
// 通过中序遍历和后序遍历建立二叉树
//https://vjudge.net/problem/UVA-548


#include<bits/stdc++.h>

using namespace std;
const int maxn = 1e5+10;
const int INF = 1e8;
int in_order[maxn],post_order[maxn],l[maxn],r[maxn];
int n;
int read_order(int *a)
{
    string s;
    if(!getline(cin,s)) return false;
    stringstream ss(s);
    n = 0;
    int v;
    while(ss >> v)
        a[n++] = v;
    return n > 0;
}
int build_tree(int L1,int R1,int L2,int R2)
{
    if(L1 > R1)
        return 0;
    int root = post_order[R2];
    int p = L1;
    while(in_order[p] != root)
        p++;
    int cnt = p-L1;
    l[root] = build_tree(L1,p-1,L2,L2+cnt-1);
    r[root] = build_tree(p+1,R1,L2+cnt,R2-1);
    return root;
}
int best,bestsum;
void dfs(int a,int b)
{
    if(!l[a] && !r[a])
    {
        b += a;
        if(bestsum > b||(bestsum == b&&best > a))
        {
            best = a;
            bestsum = b;
        }
    }
    if(l[a]) dfs(l[a],b+a);
    if(r[a]) dfs(r[a],b+a);
}
```

```
int main(void)
{
    while(read_order(in_order))
    {
        read_order(post_order);
        build_tree(0,n-1,0,n-1);
//        cout<<0<<endl;
        bestsum = INF;
        dfs(post_order[n-1],0);
        cout<<best<<endl;
    }

    return 0;
}
```

## 1.2 堆

```
// 堆的插入和删除操作

void Insert(int vv)
{
    int t = sz++;
    h[t] = vv;
    while(t > 1)
    {
        if(h[t] < h[t/2])
        {
            swap(h[t],h[t/2]);
            t /= 2;
        }
        else break;
    }
}
int Down(int i)
{
    int t;
    while(i * 2 <= n)
    {
        if(h[i] > h[2*i])
            t = 2*i;
        else
            t = i;
        if(i*2+1 <= n&&h[i*2+1] < h[t])
          t = i*2+1;
        if(i == t)
            break;
        swap(h[t],h[i]);
        i = t;
    }
}
```

## 1.3 字符串

### 1.3.1 1 Trie(前缀树)

```
const int maxnode = 4e5+100;
const int sigma_size = 26;
struct Trie
{
    int ch[maxnode][sigma_size];
    int val[maxnode];
    int sz;
    Trie()
    {
        sz = 1;
        memset(ch[0],0,sizeof(ch[0]));
    }
    int idx(char c)
    {
        return c-'a';
    }
    void init(void)
    {
        memset(ch,0,sizeof(ch));
        memset(val,0,sizeof(val));
    }
    void insert(char *s,int v)
    {
        int u = 0, n = strlen(s);
        for(int i = 0; i < n; ++i)
        {
            int c = idx(s[i]);
            if(!ch[u][c])
            {
                memset(ch[sz],0,sizeof(ch[sz]));
                val[sz] = 0;
                ch[u][c] = sz++;
            }
            u = ch[u][c];
        }
        val[u] = v;
    }
    int query(char *s,int  t)
    {
        int sum = 0;
        int u = 0,n = strlen(s);
        for(int i =  0; i < n; ++i)
        {
            int c = idx(s[i]);
            if(ch[u][c])
            {
                if(val[ch[u][c]])
                    sum = (sum+ans[i+t+1]) % mod;
            }
            else
                return sum;
```

```
            u = ch[u][c];
        }
        return sum;
    }

};
```

### 1.3.2 2 KMP

```
#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int    prime = 999983;
const int    INF = 0x7FFFFFFF;
const LL     INFF =0x7FFFFFFFFFFFFFFF;
const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-6;
const LL mod = 20071027 ;
int f[1100];
char ch[100];
void getFail(char *P,int *f)
{
    int m = strlen(P);
    f[0] = 0,f[1] = 0;
    for(int i = 1;i < m; ++i)
    {
        int j = f[i];
        while(j && P[i] != P[j]) j = f[j];
        f[i+1] = P[i] == P[j] ? j + 1: 0;

    }

}
void find(char * T,char * P,int* f)
{
    int n = strlen(T),m = strlen(P);
    getFail(P,f);
    int j = 0;
    for(int i =  0;i < n; ++i)
    {
        while(j&&P[j] != T[i]) j = f[j];
        if(P[j] == T[i]) j++;
        if(j == m) printf("%d\n",i-m+1);
    }
}

int main(void)
{
    cin>>ch;
```

```
    getFail(ch,f);
    printf("%d",f[strlen(ch)-1]);

    return 0;
}
```

### 1.3.3 3 AC 自动机

```
const int SIGMA_SIZE = 26;
const int MAXNODE = 11000;
const int MAXS = 150 + 10;


struct AhoCorasickAutomata {
  int ch[MAXNODE][SIGMA_SIZE];
  int f[MAXNODE];     // fail 函数
  int val[MAXNODE];  // 每个字符串的结尾结点都有一个非 0 的 val
  int last[MAXNODE]; // 输出链表的下一个结点
  int sz;

  void init() {
    sz = 1;
    memset(ch[0], 0, sizeof(ch[0]));
  }

  // 字符 c 的编号
  int idx(char c) {
    return c-'a';
  }

  // 插入字符串。v 必须非 0
  void insert(char *s, int v) {
    int u = 0, n = strlen(s);
    for(int i = 0; i < n; i++) {
      int c = idx(s[i]);
      if(!ch[u][c]) {
        memset(ch[sz], 0, sizeof(ch[sz]));
        val[sz] = 0;
        ch[u][c] = sz++;
      }
      u = ch[u][c];
    }
    val[u] = v;
  }

  // 递归打印以结点 j 结尾的所有字符串
  void print(int j) {
    if(j) {
      print(last[j]);
    }
  }

  // 在 T 中找模板
  int find(char* T) {
```

```
    int n = strlen(T);
    int j = 0; // 当前结点编号，初始为根结点
    for(int i = 0; i < n; i++) { // 文本串当前指针
      int c = idx(T[i]);
      while(j && !ch[j][c]) j = f[j]; // 顺着细边走，直到可以匹配
      j = ch[j][c];
      if(val[j]) print(j);
      else if(last[j]) print(last[j]); // 找到了！
    }
  }

  // 计算 fail 函数
  void getFail() {
    queue<int> q;
    f[0] = 0;
    // 初始化队列
    for(int c = 0; c < SIGMA_SIZE; c++) {
      int u = ch[0][c];
      if(u) { f[u] = 0; q.push(u); last[u] = 0; }
    }
    // 按 BFS 顺序计算 fail
    while(!q.empty()) {
      int r = q.front(); q.pop();
      for(int c = 0; c < SIGMA_SIZE; c++) {
        int u = ch[r][c];
        if(!u) continue;
        q.push(u);
        int v = f[r];
        while(v && !ch[v][c]) v = f[v];
        f[u] = ch[v][c];
        last[u] = val[f[u]] ? f[u] : last[f[u]];
      }
    }
  }

};
```

#### 1.3.4 4 KMP-KMP 变形

```
//https://www.nowcoder.com/acm/contest/119/E

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

const int N=200010;
int a[N],b[N];
int x[N],y[N],nxt[N];

void kmp_pre(int x[],int m,int nxt[])
{
    int i,j;
    j=nxt[0]=-1;
    i=0;
    while(i<m) {
```

```
        while(-1!=j && (x[i]!=x[j]&&x[j]!=-1))j=nxt[j];
        nxt[++i]=++j;
    }
}

int KMP_Count(int x[],int m,int y[],int n)
{
//    for (int i=0;i<n;i++) {
//        printf("%d ",y[i]);
//    }
//    puts("");
//    for (int i=0;i<m;i++) {
//        printf("%d ",x[i]);
//    }
//    puts("");
    int i,j;
    int ans=0;
    kmp_pre(x,m,nxt);
    i=j=0;
    while(i<n) {
        while(-1!=j && !(y[i]==x[j]||(x[j]==-1&&(y[i]==-1||j-y[i]<0)))) j=nxt[j];
        i++;
        j++;
        if(j>=m) {
            ans++;
            j=nxt[j];
        }
    }
    return ans;
}

int main()
{
    int n,m,k;
    scanf("%d%d",&n,&k);
    memset(x,-1,sizeof(x));
    memset(y,-1,sizeof(y));
    map<int,int> pre;
    for (int i=0;i<n;i++) {
        scanf("%d",&a[i]);
        auto pos=pre.find(a[i]);
        if (pos!=pre.end()) {
            y[i]=i-pos->second;
        }
        pre[a[i]]=i;
    }
    scanf("%d",&m);
    pre.clear();
    for (int i=0;i<m;i++) {
        scanf("%d",&b[i]);
        auto pos=pre.find(b[i]);
        if (pos!=pre.end()) {
            x[i]=i-pos->second;
        }
```

```
        pre[b[i]]=i;
    }
    printf("%d\n",KMP_Count(x,m,y,n));
    return 0;
}
```

### 1.3.5 5 字符串 hash

```
// 字符串 hash, 查找在字符串中至少出现 k 次的最长字符串
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;

const int maxn  = 40000+10;
const int x = 123;
int n,m,pos;

unsigned long long H[maxn],xp[maxn];

unsigned long long Hash[maxn];
int Rank[maxn];

int cmp(const int &a,const int &b){
        return Hash[a] < Hash[b] ||(Hash[a] == Hash[b] &&a <b );
}

int possible(int L){
        int c = 0;
        pos = -1;
        for(int i = 0;i < n-L+1; ++i){
                Rank[i] = i;
                Hash[i] = H[i]-H[i+L]*xp[L];

        }
        sort(Rank,Rank+n-L+1,cmp);
        for(int i = 0;i < n-L+1; ++i){
                if(i == 0||Hash[Rank[i]] != Hash[Rank[i-1]]) c = 0;
                if(++c >= m) pos = max(pos,Rank[i]);
        }
        return pos >= 0;
}

char s[maxn];
int main(void)
{
    while((scanf("%d",&m)) == 1&&m){
        scanf("%s",s);
        n = strlen(s);
        H[n] = 0;
        for(int i = n-1;i >= 0; i--) H[i] = H[i+1]*x+(s[i]-'a');
        xp[0] = 1;
        for(int i = 1;i <= n; ++i) xp[i] = xp[i-1]*x;
        if(!possible(1)) printf("none\n");
```

```
    else{
            int L = 1,R = n;
        while(R >= L){
                int M = (R+L)/2;
                if(possible(M)) L = M+1;
                else R = M-1;
        }
        possible(R);
        printf("%d %d\n",R,pos);
       }
    }

   return 0;
}
```

### 1.3.6 6 后缀数组

```
const int maxn = 1e6 + 10;

struct SuffixArray {
  int s[maxn];      // 原始字符数组（最后一个字符应必须是 0，而前面的字符必须非 0）
  int sa[maxn];     // 后缀数组
  int rank[maxn];   // 名次数组. rank[0] 一定是 n-1，即最后一个字符
  int height[maxn]; // height 数组
  int t[maxn], t2[maxn], c[maxn]; // 辅助数组
  int n; // 字符个数

  void clear() { n = 0; memset(sa, 0, sizeof(sa)); }

  // m 为最大字符值加 1。调用之前需设置好 s 和 n
  void build_sa(int m) {
    int i, *x = t, *y = t2;
    for(i = 0; i < m; i++) c[i] = 0;
    for(i = 0; i < n; i++) c[x[i] = s[i]]++;
    for(i = 1; i < m; i++) c[i] += c[i-1];
    for(i = n-1; i >= 0; i--) sa[--c[x[i]]] = i;
    for(int k = 1; k <= n; k <<= 1) {
      int p = 0;
      for(i = n-k; i < n; i++) y[p++] = i;
      for(i = 0; i < n; i++) if(sa[i] >= k) y[p++] = sa[i]-k;
      for(i = 0; i < m; i++) c[i] = 0;
      for(i = 0; i < n; i++) c[x[y[i]]]++;
      for(i = 0; i < m; i++) c[i] += c[i-1];
      for(i = n-1; i >= 0; i--) sa[--c[x[y[i]]]] = y[i];
      swap(x, y);
      p = 1; x[sa[0]] = 0;
      for(i = 1; i < n; i++)
        x[sa[i]] = y[sa[i-1]]==y[sa[i]] && y[sa[i-1]+k]==y[sa[i]+k] ? p-1 : p++;
      if(p >= n) break;
      m = p;
    }
  }

  void build_height() {
```

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

# 2 2 动态规划

## 2.1 1 最长上升子序列.cpp

```
//最长上升子序列 The longest increasing sequence

template  <class It>
int n_lisLength(It begin,It end)
{
    typedef typename iterator_traits<It>::value_type T;
    T inf = 1<<30;
    vector<T> best(end-begin,inf);
    for(It i = begin; i != end; ++i)
        *lower_bound(best.begin(),best.end(),*i) = *i;
    return lower_bound(best.begin(),best.end(),inf) - best.begin();

}
```

# 3 3 图论

## 3.1 DFS

### 3.1.1 1. 无向图的割点和桥

```
SPF POJ - 1523
// 如果有割点，那么割点与子节点边就是割边
int dfs(int u,int fa){
    int lowu = pre[u] = ++dfs_clock;
    int child = 0;
    for(int i = 0;i < G[u].size(); ++i){
        int v = G[u][i];
        if(!pre[v]){
            child++;
            int lowv = dfs(v,u);
            lowu = min(lowu,lowv);
            if(lowv >= pre[u]){
                iscut[u]++;
            }
        }
        else if(pre[v] < pre[u] && v != fa){
            lowu = min(lowu,pre[v]);
        }
    }
```

```
    if(fa < 0&&child == 1) iscut[u] = 0;
    else if(fa < 0&&child >= 2) iscut[u] = child-1;
    return low[u] = lowu;
}
```

如果要输出去掉割点之后的联通分量的个数，需要谈判根的情况

```
#include<iostream>
#include<cstdio>
#include<cctype>
#include<cstring>
#include<algorithm>
#include<vector>
#include<stack>
#include<map>
#include<queue>
#include<cmath>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
#define Pb push_back
#define  FI first
#define  SE second
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define IOS ios::sync_with_stdio(false)
#define DEBUG cout<<endl<<"DEBUG"<<endl;
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int    prime = 999983;
const int    INF = 0x7FFFFFFF;
const LL     INFF =0x7FFFFFFFFFFFFFFF;
const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-6;
const LL     mod = 1e9 + 7;
LL qpow(LL a,LL b){LL s=1;while(b>0){if(b&1)s=s*a%mod;a=a*a%mod;b>>=1;}return s;}
LL gcd(LL a,LL b) {return b?gcd(b,a%b):a;}
int dr[2][4] = {1,-1,0,0,0,0,-1,1};
typedef pair<int,int> P;
const int maxn = 1000+100;
// cosnt int maxm = 1e6+100
int pre[maxn];
int dfs_clock = 0;
vector<int> G[maxn];
int iscut[maxn];
int low[maxn];

void init(){
    dfs_clock = 1;
    rep(i,1,maxn) G[i].clear();
    me(iscut);
    me(low);
    me(pre);
}
```

```
int dfs(int u,int fa){
    int lowu = pre[u] = ++dfs_clock;
    int child = 0;
    for(int i = 0;i < G[u].size(); ++i){
        int v = G[u][i];
        if(!pre[v]){
            child++;
            int lowv = dfs(v,u);
            lowu = min(lowu,lowv);
            if(lowv >= pre[u]){
                iscut[u]++;
            }
        }
        else if(pre[v] < pre[u] && v != fa){
            lowu = min(lowu,pre[v]);
        }
    }
    if(fa < 0&&child == 1) iscut[u] = 0;
    else if(fa < 0&&child >= 2) iscut[u] = child-1;
    return low[u] = lowu;
}
// #define Debug
int main(void)
{
    #ifdef Debug
    freopen("input.txt","r",stdin);
    freopen("output.txt","w+",stdout);
    #endif
    int kase = 0;
    while(1){
        init();
        int u,v;
        int t = 0;
        while(scanf("%d",&u)==1&&u != 0){
          t++;
          scanf("%d",&v);
          G[u].Pb(v);
          G[v].Pb(u);
        }
        if(t==0)break;
        // rep(i,1,maxn) if(!G[i].empty()){

        //  dfs(i,-1);
        //  break;
        // }
        dfs(1,-1);
        int num = 0;
        rep(i,1,1001) if(iscut[i]) num++;

        printf("Network #%d\n",++kase);
        if(num > 0)
        {
            rep(i,1,1001) if(iscut[i]){
            printf("  SPF node %d leaves %d subnets\n",i,iscut[i]+1);
```

```
        }
        }
        else
            printf("  No SPF nodes\n");
        if(kase) puts("");
    }

   return 0;
}
```

### 3.1.2 2. 无向图的双连通分量

```
// 无向图的点联通分量

const int maxn= 1000+10;
int pre[maxn],iscut[maxn],bccno[maxn],dfs_clock,bcc_cnt;
vector<int> G[maxn],bcc[maxn];

stack<Edge> S;
int dfs(int u,int fa){
  int lowu = pre[u] = ++dfs_clock;
  int child = 0;
  for(int i = 0;i < G[u].size(); ++i){
    int v = G[u][i];
    Edge e = (Edge) {u,v};
    if(!pre[v]){
      S.push(e);
      child++;
      int lowv = dfs(v,u);
      lowu = min(lowu,lowv);
      if(lowv >= pre[u]){
        iscut[u] = true;
        bcc_cnt++;
        bcc[bcc_cnt].clear();
        for(;;){
          Edge x = S.top(); S.pop();
          if(bccno[x.u] != bcc_cnt) {bcc[bcc_cnt].push_back(x.u); bccno[x.u] = bcc_cnt;}
          if(bccno[x.v] != bcc_cnt) {bcc[bcc_cnt].push_back(x.v); bccno[x.v] = bcc_cnt;}
          if(x.u == u&&x.v == v) break;

        }
      }
    }
    else if(pre[v] < pre[u]&&v != fa){
      S.push(e);lowu = min(pre[v],lowu);
    }
  }
  if(fa < 0&& child == 1) iscut[u] = 0;
  return lowu;
}
void find_bcc(int n){
  memset(pre,0,sizeof(pre));
  memset(iscut,0,sizeof(iscut));
  memset(bccno,0,sizeof(bccno));
```

```
  dfs_clock = bcc_cnt = 0;
  for(int i = 0;i < n; ++i) if(!pre[i]) dfs(i,-1);

}

//无向图的边-双联通分量
// 第一边 dfs 求出所有的割边，然后第二边 dfs 求出所有边—双连通分量（不经过割边）
```

### 3.1.3 3 有向图的强联通分量

```
// tarjan 算法
const int maxn = 2e4+100;

vector<int> G[maxn];
int pre[maxn],lowlink[maxn],sccno[maxn],dfs_clock,scc_cnt;
stack<int> S;
void dfs(int u){
    pre[u] = lowlink[u] = ++dfs_clock;
    S.push(u);
    for(int i = 0;i < G[u].size(); ++i){
        int v = G[u][i];
        if(!pre[v]){
            dfs(v);
            lowlink[u] = min(lowlink[u],lowlink[v]);

        }
    else if(!sccno[v]){
        lowlink[u] = min(lowlink[u],pre[v]);
        }
    }
    if(lowlink[u] == pre[u]){
        scc_cnt++;
        for(;;){
            int x = S.top(); S.pop();
            sccno[x] = scc_cnt;
            if(x == u) break;
        }
    }

}
void find_scc(int n){
    dfs_clock= scc_cnt = 0;
    me(sccno),me(pre);
    rep(i,0,n) if(!pre[i]) dfs(i);
}
// kosaraju



const int maxn = 2e4+100;
vector<int> G[maxn],G2[maxn];
vector<int> S;
int vis[maxn],sccno[maxn],scc_cnt;
void dfs1(int u){
```

```
        if(vis[u]) return ;
        vis[u] = 1;
        for(int i = 0;i < G[u].size(); ++i) dfs1(G[u][i]);
        S.push_back(u);
}
void dfs2(int u){
        if(sccno[u]) return  ;
        sccno[u] = scc_cnt;
          for(int i = 0;i < G2[u].size(); ++i) dfs2(G2[u][i]);
}
void find_scc(int n){
        scc_cnt = 0;
        S.clear();
        memset(sccno,0,sizeof(sccno));
        memset(vis,0,sizeof(vis));
        for(int i = 0;i < n; ++i) dfs1(i);
    for(int i = n-1;i >= 0;--i){
            if(!sccno[S[i]]) {
                    scc_cnt++;
                    dfs2(S[i]);
            }
    }
}
```

#### 3.1.4 4 2-sat 问题

```
// O(n*m) 复杂度不确定

const int maxn = 2000 + 10;

struct TwoSAT {
  int n;
  vector<int> G[maxn*2];
  bool mark[maxn*2];
  int S[maxn*2], c;

  bool dfs(int x) {
    if (mark[x^1]) return false;
    if (mark[x]) return true;
    mark[x] = true;
    S[c++] = x;
    for (int i = 0; i < G[x].size(); i++)
      if (!dfs(G[x][i])) return false;
    return true;
  }

  void init(int n) {
    this->n = n;
    for (int i = 0; i < n*2; i++) G[i].clear();
    memset(mark, 0, sizeof(mark));
  }

  // x = xval or y = yval
  void add_clause(int x, int xval, int y, int yval) {
```

```
    x = x * 2 + xval;
    y = y * 2 + yval;
    G[x].push_back(y^1);// G[0].Pb(1)
    G[y].push_back(x^1);// G[1].Pb(0);
  }

  bool solve() {
    for(int i = 0; i < n*2; i += 2)
      if(!mark[i] && !mark[i+1]) {
        c = 0;
        if(!dfs(i)) {
          while(c > 0) mark[S[--c]] = false;
          if(!dfs(i+1)) return false;
        }
      }
    return true;
  }
};
```

## 3.2 LCA

### 3.2.1 1 DFS+RMQ

```
#include<cstdio>
#include<cstring>
#include<vector>
#include<cmath>
#include<iostream>
using namespace std;

const int maxn = 40000+100;
const int maxlogv = 17;
struct Edge{
        int to,weight;
        Edge(int t,int w):to(t),weight(w){};
};
vector<Edge> G[maxn];

int id[maxn],dis[maxn];
int vs[maxn*2],depth[maxn*2];
int dp[maxn*2][maxlogv];
void dfs(int node,int fa,int d,int &k){
           id[node] = k;
           vs[k] = node;
         depth[k++] = d;
         // dis[node] = distance;
         for(int i = 0;i < G[node].size(); ++i){
                 Edge &t = G[node][i];
                 if(t.to == fa) continue;
                 dis[t.to] = dis[node]+t.weight;
                 dfs(t.to,node,d+1,k);
        vs[k] = node;
        depth[k++] = d;
         }
}
```

```
void init_rmq(int n){

        for(int i = 0;i < n ; ++i) dp[i][0] = i;
    for(int j = 1;(1<<j) <= n; ++j){
            for(int i = 0;i + (1<<j)-1 < n; ++i){
                    if(depth[dp[i][j-1]]< depth[dp[i+(1<<(j-1))][j-1]])
                            dp[i][j] = dp[i][j-1];
                    else
                            dp[i][j] = dp[i+(1<<(j-1))][j-1];

            }
    }
}
int query(int l,int r){
        int k = 0;
        while((1<<(k+1)) <= r-l+1) k++;
         if(depth[dp[l][k]] < depth[dp[r-(1<<k)+1][k]])
                 return dp[l][k];
         else
                 return dp[r-(1<<k)+1][k];
}
int lca(int u,int v){
        return vs[query(min(id[u],id[v]),max(id[u],id[v]))];
}
void init(int n){
        int k = 0;
        dfs(0,-1,0,k);
        init_rmq(2*n-1);
}
int main(void){
    int n,m,q;
    while(~scanf("%d%d",&n,&m)){
            for(int i = 0;i < n; ++i) G[i].clear();
            int u,v,w;
            for(int i = 0;i < m; ++i){
                    scanf("%d%d%d",&u,&v,&w);
                    u--,v--;
                    G[u].push_back(Edge(v,w));
                    G[v].push_back(Edge(u,w));
            }
            init(n);
            scanf("%d",&q);
            while(q--){
                    int u,v;
                    scanf("%d %d",&u,&v);
                    u--,v--;
                    int f = lca(u,v);
                    printf("%d\n",dis[u]+dis[v]-2*dis[f]);
            }
    }
        return 0;
}
```

### 3.2.2 2 倍增算法

```
// POJ1330
// LCA 的倍增算法

#include<vector>
#include<cstdio>
#include<cstring>
using namespace std;

const int maxn = 1e4+100;
const int maxlogv = 14;
vector<int> G[maxn];
int root;

int parent[maxlogv][maxn];
int depth[maxn];

void dfs(int v,int p,int d){
        parent[0][v] = p;
        depth[v] = d;
        for(int i = 0;i < G[v].size(); ++i){
                if(G[v][i] != p){
                        dfs(G[v][i],v,d+1);
                }
        }
}
void init(int V){
        dfs(root,-1,0);
        for(int k = 0;k+1 < maxlogv; ++k){
                for(int v = 0; v < V; ++v){
                        if(parent[k][v] < 0) parent[k+1][v] = -1;
                        else parent[k+1][v] =  parent[k][parent[k][v]];

                }
        }
}

int lca(int u,int v){
        if(depth[u] > depth[v]) swap(u,v);
        for(int k = 0;k < maxlogv; ++k){
                if(((depth[v] - depth[u]) >> k)& 1){
                        v = parent[k][v];
                }

        }
        if(u == v) return u;
        for(int k = maxlogv-1; k >= 0; --k){
                if(parent[k][u] != parent[k][v]){
                        u = parent[k][u];
                        v = parent[k][v];
                }
        }
        return parent[0][u];
```

```
}
bool OUT[maxn];
int main(void)
{

        int T;
    scanf("%d",&T);
    while(T--){
            int n;
            for(int i = 0;i < n; ++i) G[i].clear();
            memset(OUT,0,sizeof(OUT));
            scanf("%d",&n);
            for(int i = 1;i  < n; ++i) {
                    int u,v;
                    scanf("%d %d",&u,&v);
                    u--,v--;
                    G[u].push_back(v);
            OUT[v] = 1;
            }
            for(int i = 0;i < n; ++i) if(!OUT[i]){
            root = i;
            break;
            }
            init(n);
            int u,v;
            scanf("%d %d",&u,&v);
            u--,v--;
        printf("%d\n",lca(u,v)+1);
    }

   return 0;
}
```

## 3.3 Maxflow

### 3.3.1 1 Dinic

```
// dinic
#include <cstdio>//C 语言 io
#include <cstring>//以下是 c 语言常用头文件
#include <cmath>
#include <cstdlib>
#include <ctime>
#include <cctype>
#include <cstring>
#include <cmath>
#include <iostream>//c++IO
#include <sstream>
#include <string>
#include <list>//c++ 常用容器
#include <vector>
#include <set>
#include <map>
#include <queue>
#include <stack>
```

```cpp
#include <algorithm>//c++ 泛型的一些函数
#include <functional>//用来提供一些模版
#define fo0(i,n) for(int i = 0;i < n; ++i)
#define fo1(i,n) for(int i = 1;i <= n; ++i)
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int    prime = 999983;
const int    INF = 0x7FFFFFFF;
const LL     INFF =0x7FFFFFFFFFFFFFFF;
const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-6;
const LL     mod = 1e9 + 7;
const int LEN = 20000+1000;
const int maxn = 1e8;
struct Edge{
  int from,to,cap,flow;
  Edge(int u,int v,int w,int f): from(u),to(v),cap(w),flow(f){}
};
struct Dinic{
   int n,m,s,t;
   vector<Edge> edges;
   vector<int> G[LEN];
   int a[LEN];
   int vis[LEN];
   int d[LEN];
   int cur[LEN];//好吧就是点，代表该点在一次求增广的过程中搜索到了那条边，意思就是从这条边往下肯定搜索不到结
   void init(int n)
   {
       this->n  = n;
       for(int i = 0;i < n; ++i)
        G[i].clear();
       edges.clear();
   }
   void Add(int u,int v,int w)
   {
       edges.push_back(Edge(u,v,w,0));
       edges.push_back(Edge(v,u,0,0));
       m = edges.size();
       G[u].push_back(m-2);
       G[v].push_back(m-1);
   }
   bool Bfs(void)//分层
   {
      me(d);
      me(vis);
      d[s] = 0;
      vis[s] = 1;

      queue<int> Q;
```

```
    Q.push(s);
    while(!Q.empty())
    {
        int q = Q.front();Q.pop();

        for(size_t i = 0;i < G[q].size();++i)
        {
            Edge &tmp = edges[G[q][i]];
            if(!vis[tmp.to]&&tmp.cap>tmp.flow)
            {
                vis[tmp.to] = 1;
                d[tmp.to] = d[q] + 1;
                Q.push(tmp.to);
            }
        }
    }
    return vis[t];
}
int Dfs(int node,int a)
{

    if(node == t||a == 0)
     return a;
    int flow =  0,f;
    for(int &i = cur[node];i < G[node].size();++i)
    {
       Edge &tmp = edges[G[node][i]];
       if(d[tmp.to]==d[node]+1&&(f=Dfs(tmp.to,min(a,tmp.cap-tmp.flow)))>0)
       {
           flow += f;
           tmp.flow += f;
           edges[G[node][i]^1].flow -= f;
           a -= f;
           if(a==0)
             break;
       }
    }
    return flow;
}
int MaxFlow(int s,int t)
{
    this->s = s;
    this->t = t;
    int flow = 0;
    while(Bfs())
    {
        me(cur);
        flow += Dfs(s,maxn);
    }
    return flow;

}
```

```
};
Dinic dinic;
int main()
{
    int N,M,S,T;
    while(cin>>N>>M)
    {
        S =1, T = N;
        dinic.init(N);
        int u,v,w;
        for(int i = 0;i < M;++i)
        {
            scanf("%d %d %d",&u,&v,&w);
            dinic.Add(u,v,w);
        }
        int ans = 0;
        ans = dinic.MaxFlow(S,T);
        printf("%d\n",ans);

    }



    return 0;
}
```

### 3.3.2 2 ISAP

```
// 点的下标从零开始，注意初始化
#include<cstdio>
#include<cstring>
#include<queue>
#include<vector>
#include<algorithm>
using namespace std;

const int maxn = 10000 + 10;
const int INF = 1000000000;

struct Edge {
  int from, to, cap, flow;
};

bool operator < (const Edge& a, const Edge& b) {
  return a.from < b.from || (a.from == b.from && a.to < b.to);
}

struct ISAP {
  int n, m, s, t;
  vector<Edge> edges;
  vector<int> G[maxn];   // 邻接表，G[i][j] 表示结点 i 的第 j 条边在 e 数组中的序号
  bool vis[maxn];        // BFS 使用
  int d[maxn];           // 从起点到 i 的距离
  int cur[maxn];        // 当前弧指针
```

```
int p[maxn];           // 可增广路上的上一条弧
int num[maxn];         // 距离标号计数

void AddEdge(int from, int to, int cap) {
  edges.push_back((Edge){from, to, cap, 0});
  edges.push_back((Edge){to, from, 0, 0});
  m = edges.size();
  G[from].push_back(m-2);
  G[to].push_back(m-1);
}

bool BFS() {
  memset(vis, 0, sizeof(vis));
  queue<int> Q;
  Q.push(t);
  vis[t] = 1;
  d[t] = 0;
  while(!Q.empty()) {
    int x = Q.front(); Q.pop();
    for(int i = 0; i < G[x].size(); i++) {
      Edge& e = edges[G[x][i]^1];
      if(!vis[e.from] && e.cap > e.flow) {
        vis[e.from] = 1;
        d[e.from] = d[x] + 1;
        Q.push(e.from);
      }
    }
  }
  return vis[s];
}

void init(int n) {
  this->n = n;
  for(int i = 0; i < n; i++) G[i].clear();
  edges.clear();
}


int Augment() {
  int x = t, a = INF;
  while(x != s) {
    Edge& e = edges[p[x]];
    a = min(a, e.cap-e.flow);
    x = edges[p[x]].from;
  }
  x = t;
  while(x != s) {
    edges[p[x]].flow += a;
    edges[p[x]^1].flow -= a;
    x = edges[p[x]].from;
  }
  return a;
}
```

```
  int Maxflow(int s, int t) {
    this->s = s; this->t = t;
    int flow = 0;
    BFS();
    memset(num, 0, sizeof(num));
    for(int i = 0; i < n; i++) num[d[i]]++;
    int x = s;
    memset(cur, 0, sizeof(cur));
    while(d[s] < n) {
      if(x == t) {
        flow += Augment();

        x = s;
      }
      int ok = 0;
      for(int i = cur[x]; i < G[x].size(); i++) {
        Edge& e = edges[G[x][i]];
        if(e.cap > e.flow && d[x] == d[e.to] + 1) { // Advance
          ok = 1;
          p[e.to] = G[x][i];
          cur[x] = i; // 注意
          x = e.to;
          break;
        }
      }
      if(!ok) { // Retreat
        int m = n-1; // 初值注意
        for(int i = 0; i < G[x].size(); i++) {
          Edge& e = edges[G[x][i]];
          if(e.cap > e.flow) m = min(m, d[e.to]);
        }
        if(--num[d[x]] == 0) break;
        num[d[x] = m+1]++;
        cur[x] = 0; // 注意
        if(x != s) x = edges[p[x]].from;
      }
    }
    return flow;
  }
};


ISAP g;

int main() {

 int N,M;
 int S,T;
 scanf("%d %d",&N,&M);
 scanf("%d %d",&S,&T);
 int u,v,w;
 g.init(N);
 while(M--){
         scanf("%d %d %d",&u,&v,&w);
```

```
        u--,v--;
    g.AddEdge(u,v,w);
 }
 printf("%d",g.Maxflow(S-1,T-1));


  return 0;
}
```

### 3.3.3 3 MCMF

```
// 最小费用最大流, 下标从 1 开始

#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
#define Pb push_back
#define  FI first
#define  SE second
#define For(i,a,b) for(int i = a; i < b; ++i)
#define IOS ios::sync_with_stdio(false)
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int    prime = 999983;
const int    INF = 1e8;
const LL     INFF =0x7FFFFFFFFFFFFFFF;
const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-6;
const LL     mod = 1e9 + 7;
LL qpow(LL a,LL b){LL s=1;while(b>0){if(b&1)s=s*a%mod;a=a*a%mod;b>>=1;}return s;}
LL gcd(LL a,LL b) {return b?gcd(b,a%b):a;}
int dr[2][4] = {1,-1,0,0,0,0,-1,1};
typedef pair<int,int> P;
struct Edge{
   int from,to,cap,flow,cost;
};
const int maxn = 5000+100;
struct MCMF{
        int n,m,s,t;
        vector<Edge> edges;
        vector<int> G[maxn];
        int inq[maxn];
        int d[maxn];
        int p[maxn];
        int a[maxn];
        void init(int n){
                this->n = n;
                for(int i = 0;i < n; ++i) G[i].clear();
                edges.clear();
        }
        void AddEdge(int from,int to,int cap,int cost){
```

```
                edges.push_back((Edge){from,to,cap,0,cost});
                edges.push_back((Edge){to,from,0,0,-cost});
                int m = edges.size();
                G[from].push_back(m-2);
                G[to].push_back(m-1);

        }
        bool BellmanFord(int s,int t,int &flow,int &cost){
                for(int i = 0;i < n; ++i) d[i] = INF;
                memset(inq,0,sizeof(inq));
                d[s] = 0,inq[s] = 1;p[s] = 0,a[s] = INF;

                queue<int> Q;
                Q.push(s);
                while(!Q.empty()){

                        int u = Q.front(); Q.pop();
                        inq[u] = 0;
                        for(int i = 0;i < G[u].size(); ++i){
                                Edge& e = edges[G[u][i]];
                                if(e.cap > e.flow &&d[e.to] > d[u]+e.cost){
                                        d[e.to] = d[u]+e.cost;
                                        p[e.to] = G[u][i];
                                        a[e.to] = min(a[u],e.cap-e.flow);
                                        if(!inq[e.to]) {
                                                Q.push(e.to); inq[e.to] = 1;
                                        }
                                }
                        }
                }

                if(d[t] == INF) return false;

                flow += a[t];
                cost += d[t]*a[t];
                int u = t;
                while(u != s){
                        edges[p[u]].flow += a[t];
                        edges[p[u]^1].flow -= a[t];
                        u = edges[p[u]].from;
                }
                return true;
        }
        int Mincost(int s,int t,int &flow,int &cost){
                 flow = 0,cost = 0;

                while(BellmanFord(s,t,flow,cost));
                return cost;
        }

};
MCMF mcmf;
int main(void)
{
```

```
        int n,m,s,t;
        scanf("%d %d %d %d",&n,&m,&s,&t);
        int u,v,w,c;
        mcmf.init(n+1);
        while(m--){
                scanf("%d %d %d %d",&u,&v,&w,&c);
                mcmf.AddEdge(u,v,w,c);
        }
    int flow,cost;
    flow = 0,cost = 0;
    mcmf.Mincost(s,t,flow,cost);
        printf("%d %d\n",flow,cost);


   return 0;
}
```

## 3.4 二分图

### 3.4.1 KM

```
const int maxn = 500+5;
struct KM{
        int n;
        vector<int> G[maxn];
        int W[maxn][maxn];
        int Lx[maxn];
        int Ly[maxn];
        int Left[maxn];
        bool S[maxn],T[maxn];
        void init(int n){
                this->n = n;
                for(int i = 1;i <= n; ++i) G[i].clear();
                memset(W,0,sizeof(W));
        }
        void AddEdge(int u,int v,int w){
                G[u].push_back(v);
                W[u][v] = w;
        }
        bool match(int u){
                S[u] = true;
                for(int i =0;i < G[u].size(); ++i){
                        int v = G[u][i];
                        if(Lx[u]+Ly[v] == W[u][v]&&!T[v]){
                                T[v] = true;
                                if(Left[v] == -1||match(Left[v])){
                                   Left[v] = u;
                                   return true;
                                }
                        }
                }
                return false;
        }
        void update(){
                int a = INF;
```

```
                for(int u = 0;u < n; ++u)
                 if(S[u])
                   for(int i = 0;i < G[u].size(); ++i){
                           int v = G[u][i];
                           if(!T[v])
                             a = min(a,Lx[u]+Ly[v]-W[u][v]);
                   }
                for(int i = 0;i < n; ++i){
                        if(S[i]) Lx[i] -= a;
                        if(T[i]) Ly[i] += a;
                }
        }
        void solve(){
                for(int i = 0;i < n; ++i){
                        Lx[i] = *max_element(W[i],W[i]+n);
                        Left[i] = -1;
                        Ly[i] = 0;
                }
                for(int u = 0;u < n; ++u){
                        for(;;){
                                for(int i = 0;i < n; ++i) S[i] = T[i]  = 0;
                                if(match(u)) break;
                                else update();
                        }
                }
        }
};
```

### 3.4.2 匈牙利算法

```
#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
#define Pb push_back
#define  FI first
#define  SE second
#define For(i,a,b) for(int i = a; i < b; ++i)
#define IOS ios::sync_with_stdio(false)
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int    prime = 999983;
const int    INF = 0x7FFFFFFF;
const LL     INFF =0x7FFFFFFFFFFFFFFF;
const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-6;
const LL     mod = 1e9 + 7;
LL qpow(LL a,LL b){LL s=1;while(b>0){if(b&1)s=s*a%mod;a=a*a%mod;b>>=1;}return s;}
LL gcd(LL a,LL b) {return b?gcd(b,a%b):a;}
int dr[2][4] = {1,-1,0,0,0,0,-1,1};
typedef pair<int,int> P;
const int maxn = 1000+10;
```

```
vector<int> G[maxn];
int match[maxn];
bool used[maxn];
int N,M;
bool dfs(int v){
        used[v] = true;
        for(int i = 0;i < G[v].size(); ++i){
                int u = G[v][i],w = match[u];
                if(w < 0||!used[w]&&dfs(w)){
                        match[v] = u;
                        match[u] = v;
                        return true;
                }
        }
        return false;
}
int main(void)
{
    scanf("%d %d",&N,&M);

    while(M--){
            int u,v;
            scanf("%d %d",&u,&v);
            G[u].Pb(v);
            G[v].Pb(u);
        }
        int ans = 0;
        memset(match,-1,sizeof(match));
        for(int i = 1;i <= N; ++i){
                if(match[i] < 0){
                        memset(used,0,sizeof(used));
                        if(dfs(i)){
                                ans++;
                        }
                }
        }
   cout<<ans<<endl;
   return 0;
}
```

## 3.5 最小生成树

### 3.5.1 1 Krustral 卡鲁斯卡尔算法

```
/*
复杂度 E*log(E), 适用于稀疏图
https://vjudge.net/problem/HDU-1863
*/


#include<bits/stdc++.h>

using namespace std;

const int maxn = 100+100;
```

```
struct Edge//边
{
    int from,to,cost;
    bool operator< ( const Edge & a)
    {
        return cost < a.cost;
    }
};
Edge edge[maxn];
int F[maxn];
int Find(int x)//并查集算法
{
    return x == F[x] ? x:F[x] = Find(F[x]);
}
int main(void)
{
    int N,M;
    while(cin>>N>>M&&N)// N 代表的是道路数量，M 代表村庄的数量
    {
        for(int i = 0; i <= M; ++i)
            F[i] = i;
        for(int i = 0; i <  N; ++i)
        {
             Edge &t = edge[i];
            scanf("%d %d %d",&t.from,&t.to,&t.cost);
        }
        sort(edge,edge+N);// 对边进行排序
        int sum = 0;
        int num = M;
        for(int i = 0;i < N ; ++i)// 一个个将边加进去
        {
            Edge t = edge[i];
            if(Find(t.from) == Find(t.to))
                continue;
            F[Find(t.from)] = F[Find(t.to)];
            sum += t.cost;
            num--;
        }
        if(num == 1)
            cout<<sum<<endl;
        else
            cout<<"?"<<endl;
    }


    return 0;
}
```

### 3.5.2 2 prim 算法

```
/*
prim 算法是进行加点，使用于稠密图，可以选择用堆或者不用
不用堆 O（V*V);
用堆   O(E * log(V));
```

```
https://vjudge.net/problem/HDU-1863
*/


typedef pair<int,int> P;
const int LEN = 2e6+100;
int Away[LEN];//记录从当前已选结点到 j 节点的路径的最小值
bool  vis[LEN];
int N,M;//N 道路数目, M 村庄个数
vector<vector<P> > vec(LEN);
int main()
{
        cin>>M>>N;

        int from,to,weight;
        while(N--)
        {
            scanf("%d %d %d",&from,&to,&weight);
            vec[from].push_back(P(weight,to));
            vec[to].push_back(P(weight,from));
        }// 添加边


        for(int i = 2; i <= M; ++i)
            Away[i] = INF;//初始化 Away 数组
        Away[1] = 0;
        int Left = M;
        int All_cost = 0;
        priority_queue<P,vector<P>,greater<P> > q;// 小顶堆
        q.push(P(0,1));
        while(!q.empty()&&Left>0)
        {
            P tmp = q.top();q.pop();
            int To = tmp.second;
            if(vis[To])
                continue;
            vis[To] = 1;
            Left--;
            All_cost += tmp.first;
            for(int  i = 0; i < vec[To].size(); ++i)// 更新 Away 数组
            {
                P &t = vec[To][i];
                if(!vis[t.second] && Away[t.second] > t.first)
                {
                    Away[t.second] = t.first;
                    q.push(t);
                }
            }
        }

            cout<<All_cost<<endl;
```

```
    return 0;
}
```

### 3.5.3 3 最小限制生成树

```
// 限制某一点的度数不能超过 K
#include<cstring>
#include<map>
#include<cstdio>
#include<iostream>
#include<algorithm>
#include<set>
using namespace std;
#define me(ar) memset(ar,0,sizeof(ar))
const  int    INF = 1e8;
//.....................................................
const  int LEN = 30;
int K;
int n,m;
struct Edge
{
    int x,y;
    int weight;
    bool operator <(const Edge &a) const
    {
        return weight < a.weight;
    }
} edge[LEN*LEN+10];//邻接表存边,Kruskal 算法要用
int dis[LEN][LEN];//邻接矩阵
int sign[LEN][LEN];//记录那些边已经在生成树里面了
int vis[LEN];//记录是否相连
int F[LEN];//并查集所用
int Father[LEN];//由 i 到 i+1 度限制生成树需要用动态规划求解, 用来状态转移
int Best[LEN];//Best[i] 指的是由当前节点到 park 这些边中最长边是多少
int Find(int x)//并查集所用 Find 函数
{
    return x == F[x]?x:F[x] = Find(F[x]);
}
void  Dfs(int x)//Dfs 动态规划记忆化搜索
{
//    vis[x] = 1;
    for(int i = 1;i <= n; ++i )
    {
        if(sign[i][x]&!vis[i])//如果有边相连并且下一个节点没有被访问
        {
            if(x==0)
                    Best[i] = -INF;//与 park 直接相连的边不能删除

            else
                      Best[i] = max(Best[x],dis[x][i]);//状态转移方程
            Father[i] = x;
            vis[i] = 1;
            Dfs(i);
        }
```

```
    }
}
void init(){
          for(int i = 0;i < LEN; ++i)
            F[i] = i;
        me(sign);//初始化标记数组
        me(vis);
        //初始化邻接矩阵
        for(int i = 0;i < LEN; ++i)
            for(int j = 0;j < LEN; ++j)
             dis[i][j] = INF;
}
int main(void)
{
    while(cin>>m)
    {
        //初始化并查集数组
        init();
        n = 0;//用来记录共有多少个节点
        // set<string> se;
        map<string,int> ma;//将地点编号
        ma["Park"] = 0;//将 park 加入节点
        string s1,s2;
        int a,b;
        int weight = 0;
        for(int i = 0; i < m; ++i)
        {
            cin>>s1>>s2>>weight;
            if(s1 == "Park"||ma[s1] != 0)
                a = ma[s1];//如果节点已编号，则直接使用
            else
                a = ma[s1] = ++n;//如果没有编号，编号
            if(s2 =="Park"||ma[s2]!=0)
                b = ma[s2];
            else
                b = ma[s2] = ++n;
            dis[a][b] = dis[b][a] = weight;
            edge[i].x = a;
            edge[i].y = b;
            edge[i].weight = weight;
        }
        //求最小生成树
        int ans = 0;//kruskal 算法求最小生成树
        sort(edge,edge+m);
        for(int i = 0;i < m; ++i)
        {
            int x = edge[i].x;
            int y = edge[i].y;
             weight = edge[i].weight;
            if(x==0||y==0)//去除掉 park 这个点
                continue;
            int xx = Find(x);
            int yy = Find(y);
            if(xx!=yy)
```

```
            {
                F[xx] = F[yy];
                ans += weight;
                sign[x][y] = sign[y][x] = 1;
            }
        }


        cin>>K;//最小 k 度生成树
        int Min[LEN];//用来记录每一个最小生成树到 park 点的最小路径
        for(int i = 0;i < LEN; ++i)
            Min[i] = INF;//初始化
        int index[LEN];//用来记录最小路径的点
        for(int i = 1;i <= n; ++i)
        {
            if(dis[i][0]<Min[Find(i)])
            {
                Min[Find(i)] = dis[i][0];
                index[Find(i)] = i;
            }
        }
////          cout<<se.size()<<endl;
        int m = 0;//用来记录除去 park 点即 0 点之后共有多少个连通分量
        for(int i = 1;i <= n; ++i)
        {
            if(Min[i] != INF)
            {
                ans += Min[i];
                sign[index[i]][0] = sign[0][index[i]] = 1;//将这个最小路径的点与 park 相连
                m++;
            }
        }
        int MMin = ans;
        for(int i = m + 1; i <= K; ++i)//从 m+1 到 K 求最小 i 度生成树
        {
            me(vis);
            vis[0] = 1;
            Dfs(0);
            int select = -1;//select 用来记录选择哪个与 park 点相连是最小的
            int sum = INF;
            for(int i = 1;i <= n; ++i)
            {
                if(!sign[0][i] && dis[0][i] != INF)
                {
                    if(dis[i][0]-Best[i]<sum)
                    {
                        select  = i;
                        sum = dis[i][0]-Best[i];
                    }
                }
            }
            if(select == -1)//如果找不到, 就跳出循环
                break;
            ans += sum;
```

```
            sign[select][0] = sign[0][select] = 1;
            MMin = min(MMin,ans);
            for(int i = select; i != 0; i = Father[i])
            {
                if(dis[Father[i]][i]==Best[select])
                {
                    sign[i][Father[i]] = sign[Father[i]][i] = 0;
                    break;
                }
            }
            cout<<ans<<endl;

        }
        printf("Total miles driven: %d\n",MMin);
       // cout<<MMin<<endl;
    }
    return 0;
}
```

### 3.5.4 4 次小生成树

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<string>
#include<algorithm>
#include<cmath>
#include<vector>
#include<queue>
#define ll long long
using namespace std;

int getint()
{
    int i=0,f=1;char c;
    for(c=getchar();(c<'0'||c>'9')&&c!='-';c=getchar());
    if(c=='-')f=-1,c=getchar();
    for(;c>='0'&&c<='9';c=getchar())i=(i<<3)+(i<<1)+c-'0';
    return i*f;
}

const int N=100005,M=300005;
struct node
{
    int x,y,w;
    inline friend bool operator < (const node &a,const node &b)
    {
        return a.w<b.w;
    }
}bian[M];
int n,m;
int id[N],fa[N][20],mx1[N][20],mx2[N][20],dep[N];
int tot,first[N],nxt[N<<1],to[N<<1],w[N<<1];
ll totlen,ans;
```

```
bool chs[M];

void add(int x,int y,int z)
{
    nxt[++tot]=first[x],first[x]=tot,to[tot]=y,w[tot]=z;
}

int find(int x)
{
    return id[x]==x?x:id[x]=find(id[x]);
}

void kruskal()
{
    for(int i=1;i<=n;i++)id[i]=i;
    sort(bian+1,bian+m+1);
    int cnt=0;
    for(int i=1;i<=m;i++)
    {
        int x=find(bian[i].x),y=find(bian[i].y);
        if(x!=y)
        {
            cnt++;
            totlen+=bian[i].w;
            chs[i]=true;
            add(bian[i].x,bian[i].y,bian[i].w);
            add(bian[i].y,bian[i].x,bian[i].w);
            id[y]=x;
            if(cnt==n-1)break;
        }
    }
}

void dfs(int u)
{
    for(int i=1;i<20;i++)fa[u][i]=fa[fa[u][i-1]][i-1];
    for(int i=1;i<20;i++)mx1[u][i]=max(mx1[u][i-1],mx1[fa[u][i-1]][i-1]);
    for(int i=1;i<20;i++)
    {
        mx2[u][i]=max(mx2[u][i-1],mx2[fa[u][i-1]][i-1]);
        if(mx1[u][i-1]<mx1[fa[u][i-1]][i-1]&&mx2[u][i]<mx1[u][i-1])
            mx2[u][i]=mx1[u][i-1];
        if(mx1[u][i-1]>mx1[fa[u][i-1]][i-1]&&mx1[fa[u][i-1]][i-1]>mx2[u][i])
            mx2[u][i]=mx1[fa[u][i-1]][i-1];
    }
    for(int e=first[u];e;e=nxt[e])
    {
        int v=to[e];
        if(v==fa[u][0])continue;
        fa[v][0]=u;mx1[v][0]=w[e];
        dep[v]=dep[u]+1;
        dfs(v);
    }
}
```

```
int Find(int x,int y,int len)
{
    int Mx1=0,Mx2=0;
    if(dep[x]<dep[y])swap(x,y);
    int delta=dep[x]-dep[y];
    for(int i=19;i>=0;i--)
        if(delta&(1<<i))
        {
            if(Mx1>mx1[x][i]&&mx1[x][i]>Mx2)Mx2=mx1[x][i];
            if(Mx1<mx1[x][i])Mx2=max(Mx1,mx2[x][i]),Mx1=mx1[x][i];
            x=fa[x][i];
        }
    if(x==y)return Mx1==len?Mx2:Mx1;
    for(int i=19;i>=0;i--)
        if(fa[x][i]!=fa[y][i])
        {
            if(Mx1>mx1[x][i]&&mx1[x][i]>Mx2)Mx2=mx1[x][i];
            if(Mx1<mx1[x][i])Mx2=max(Mx1,mx2[x][i]),Mx1=mx1[x][i];
            x=fa[x][i];
            if(Mx1>mx1[y][i]&&mx1[y][i]>Mx2)Mx2=mx1[y][i];
            if(Mx1<mx1[y][i])Mx2=max(Mx1,mx2[y][i]),Mx1=mx1[y][i];
            y=fa[y][i];
        }
    if(Mx1>mx1[x][0]&&mx1[x][0]>Mx2)Mx2=mx1[x][0];
    if(Mx1<mx1[x][0])Mx2=max(Mx1,mx2[x][0]),Mx1=mx1[x][0];
    x=fa[x][0];
    if(Mx1>mx1[y][0]&&mx1[y][0]>Mx2)Mx2=mx1[y][0];
    if(Mx1<mx1[y][0])Mx2=max(Mx1,mx2[y][0]),Mx1=mx1[y][0];
    y=fa[y][0];
    return Mx1==len?Mx2:Mx1;
}

void solve(int e)
{
    int x=bian[e].x,y=bian[e].y,len=bian[e].w;
    int tmp=Find(x,y,len);
    ans=min(ans,totlen-tmp+len);
}

int main()
{
    //freopen("lx.in","r",stdin);
    n=getint(),m=getint();
    for(int i=1;i<=m;i++)
    {
        bian[i].x=getint();
        bian[i].y=getint();
        bian[i].w=getint();
    }
    kruskal();
    dfs(1);
    ans=1e18;
    for(int i=1;i<=m;i++)
```

```
        if(!chs[i])solve(i);
    printf("%lld",ans);
}
```

## 3.6 最短路

### 3.6.1 1 Dijkstra

```
#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
#define Pb push_back
#define  FI first
#define  SE second
#define For(i,a,b) for(int i = a; i < b; ++i)
#define IOS ios::sync_with_stdio(false)
using namespace std;
typedef long long LL;
//typedef unsigned long long ULL;
//const int    prime = 999983;
//const int    INF = 0x7FFFFFFF;
//const LL     INFF =0x7FFFFFFFFFFFFFFF;
//const double pi = acos(-1.0);
//const double inf = 1e18;
//const double eps = 1e-6;
//const LL     mod = 1e9 + 7;
//LL qpow(LL a,LL b){LL s=1;while(b>0){if(b&1)s=s*a%mod;a=a*a%mod;b>>=1;}return s;}
//LL gcd(LL a,LL b) {return b?gcd(b,a%b):a;}
//int dr[2][4] = {1,-1,0,0,0,0,-1,1};
//typedef pair<int,int> P;
struct Dijkstra{
     #define maxn 1234
     #define INF   123456789
     int n,m;
     int s,t;

     int dis[maxn],M[maxn][maxn];
     bool vis[maxn];
     void init(){
             scanf("%d %d %d %d",&n,&m,&s,&t);
             int u,v,c;
        for(int i = 1;i <= n; ++i)
          for(int j = 1;j <= n; ++j)
            if(i != j)
               M[i][j] = INF;
             for(int i = 0;i < m; ++i){
                     scanf("%d %d %d",&u,&v,&c);
                     M[u][v] = M[v][u] = min(M[u][v],c);
                 }
         }
        void solve(){
            memset(vis,0,sizeof(vis));
                fill(dis+1,dis+n+1,INF);
                dis[s] = 0;
```

```
                for(int i = 1;i <= n; ++i){
                        int x,Min = INF;
                        for(int j = 1;j <= n; ++j){
                                if(!vis[j]&&dis[j] <= Min)
                                  Min = dis[x=j];
                        }
                        vis[x] = 1;

                    for(int j = 1;j <= n; ++j){
                         if(!vis[j]&&dis[j] > dis[x]+M[x][j])
                           dis[j] = dis[x]+M[x][j];
                        }

                }
                        printf("%d\n",dis[t]);
        }
};
Dijkstra Dij;
int main(void)
{
  Dij.init();
  Dij.solve();

   return 0;
}
// 加了堆优化的 dij

#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
#define Pb push_back
#define  FI first
#define  SE second
#define For(i,a,b) for(int i = a; i < b; ++i)
#define IOS ios::sync_with_stdio(false)
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;


int dr[2][4] = {1,-1,0,0,0,0,-1,1};
typedef pair<int,int> P;
struct Edge{
        int u,v,d;
        Edge(int uu,int vv,int dd):u(uu),v(vv),d(dd){
        }
};
struct Dijstra{
        #define maxn 123456
        #define INF  123456789
        int N,M,S,T;

        typedef pair<int,int> P;
```

```
        vector<Edge> edges;
        vector<int> G[maxn];
        bool done[maxn];
        int d[maxn];
        int p[maxn];
        void init(){
                for(int i = 1;i <= N; ++i) G[i].clear();
                edges.clear();
                scanf("%d %d %d %d",&N,&M,&S,&T);
        //        cout<<N<<M<<S<<T<<endl;
                int u,v,w;
                for(int i = 1;i <= M; ++i){
                        scanf("%d %d %d",&u,&v,&w);
                        AddEdge(u,v,w);
                        AddEdge(v,u,w);
                }

        }
        void AddEdge(int u,int v,int d){
                edges.push_back(Edge(u,v,d));
                int m = edges.size();
                G[u].push_back(m-1);
        }
        void solve(){
                priority_queue<P,vector<P>,greater<P>> Q;
                for(int i = 1;i <= N; ++i) d[i] = INF;
                d[S] = 0;
                memset(done,0,sizeof(done));
                Q.push(P(0,S));
                while(!Q.empty()){
                        P x = Q.top(); Q.pop();
                        int u = x.second;
                        if(done[u]) continue;
                        done[u] = true;
                        for(int i = 0;i <G[u].size(); ++i){
                                Edge &e = edges[G[u][i]];
                                if(!done[e.v]&&d[e.v] > d[u]+e.d){
                                        d[e.v] = d[u]+e.d;
                                        p[e.v] = G[u][i];
                                        Q.push(P(d[e.v],e.v));
                                }
                        }
                }

                printf("%d\n",d[T]);
        }
};
Dijstra Dij;
int main(void)
{
  Dij.init();
  Dij.solve();

   return 0;
```

```
}
```

### 3.6.2 2 Bellman-ford

```
#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
#define Pb push_back
#define  FI first
#define  SE second
#define For(i,a,b) for(int i = a; i < b; ++i)
#define IOS ios::sync_with_stdio(false)
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int    prime = 999983;
const int    INF = 0x7FFFFFFF;
const LL     INFF =0x7FFFFFFFFFFFFFFF;
const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-6;
const LL     mod = 1e9 + 7;
LL qpow(LL a,LL b) {
    LL s=1;
    while(b>0) {
        if(b&1)
            s=s*a%mod;
        a=a*a%mod;
        b>>=1;
    }
    return s;
}
LL gcd(LL a,LL b) {
    return b?gcd(b,a%b):a;
}
int dr[2][4] = {1,-1,0,0,0,0,-1,1};
typedef pair<int,int> P;
struct Edge{
        int from,to,dist;
        Edge(int u,int v,int d):from(u),to(v),dist(d){
        }
};
struct Bellman_ford {
    #define maxn  1234567
    bool inq[maxn];// 用来记录入队次数
    int  cnt[maxn], d[maxn], p[maxn];
    // cnt 来记录入队次数，大于 n 就退出，d 用来记录最短距离，p 用来记录路径
        int n,m;
        int s,t;
        vector<Edge> edges;
        vector<int> G[maxn];
        void AddEdge(int from,int to,int dist){
                edges.push_back(Edge(from,to,dist));
```

```
            edges.push_back(Edge(to,from,dist));
        int         m = edges.size();
            G[from].push_back(m-2);
            G[to].push_back(m-1);
        }
    void init(){

        scanf("%d %d %d %d",&n,&m,&s,&t);
        int u,v,c;
        for(int i = 0;i < m; ++i){
          scanf("%d %d %d",&u,&v,&c);
            AddEdge(u,v,c);
          }
    ///         cout<<"test"<<endl;
        }
    bool bellman_ford() {
        queue<int> Q;
        memset(inq,0,sizeof(inq));
        memset(cnt,0,sizeof(cnt));
        for(int i = 1; i <= n; ++i)
            d[i] = INF;
        d[s] = 0;
        inq[s] = true;
        Q.push(s);

        while(!Q.empty()) {
            int u = Q.front();
            Q.pop();
            inq[u] = false;
            for(int i = 0; i < G[u].size(); ++i) {
                Edge &e = edges[G[u][i]];
                if(d[u] < INF&& d[e.to]  > d[u]+e.dist) {
                    d[e.to] = d[u]+e.dist;
                    p[e.to] = G[u][i];
                    if(!inq[e.to]) {
                        Q.push(e.to);
                        inq[e.to] = true;
                        if(++cnt[e.to] > n)
                            return false;
                    }
                }
            }
        }
        printf("%d\n",d[t]);

       }
};
Bellman_ford bell;
int main(void) {
   bell.init();
   bell.bellman_ford();

    return 0;
}
```

### 3.6.3 3 floyed

```
// https://hihocoder.com/problemset/problem/1089?sid=1348128
#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
#define Pb push_back
#define  FI first
#define  SE second
#define For(i,a,b) for(int i = a; i < b; ++i)
#define IOS ios::sync_with_stdio(false)
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int    prime = 999983;
const int    INF = 0x7FFFFFFF;
const LL     INFF =0x7FFFFFFFFFFFFFFF;
const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-6;
const LL     mod = 1e9 + 7;
LL qpow(LL a,LL b){LL s=1;while(b>0){if(b&1)s=s*a%mod;a=a*a%mod;b>>=1;}return s;}
LL gcd(LL a,LL b) {return b?gcd(b,a%b):a;}
int dr[2][4] = {1,-1,0,0,0,0,-1,1};
typedef pair<int,int> P;
struct Floyd{
        // 复杂度 O(n^3)
        #define maxn 300
        int d[maxn][maxn];
        int n,m;
        void init(void){
                scanf("%d %d",&n,&m);
                for(int i = 1;i <= n ;++i)
                 for(int j = 1;j <= n; ++j)
                    if(i != j)
                       d[i][j] = INF;
                int u,v,c;
                for(int i = 0;i < m; ++i){
                        scanf("%d %d %d",&u,&v,&c);
                        d[u][v] = d[v][u] = min(d[v][u],c);
                }
        }
        void floyd(void){
                for(int k = 1; k <= n; ++k)
                 for(int i = 1;i <= n ;++i)
                   for(int j = 1;j <= n; ++j)
                     if(d[i][k] < INF&&d[j][k] < INF)
                        d[i][j] = min(d[i][j],d[i][k]+d[j][k]);
        }
        void print(void){
                for(int i = 1;i <= n; ++i){
                        for(int j = 1;j <= n; ++j)
                          printf("%d%c",d[i][j]," \n"[j==n]);
```

```
                }
        }

};
Floyd floyd;
int main(void)
{
         floyd.init();
         floyd.floyd();
         floyd.print();


   return 0;
}
```

# 4 4 数学

## 4.1 3 FWT 模板.cpp

```
// 异或
void FWT(int *a,int N,int opt){
        const int inv2 = qpow(2,mod-2);
        // j 是区间开始点, i 是区间距离, k 是具体位置, j+k,i+j+k 就是在 a 数组中的坐标
        for(int i = 1;i < N; i <<= 1){
                for(int p = i<<1,j = 0;j < N; j += p){
                        for(int k = 0;k < i; ++k){
                int X = a[j+k],Y = a[i+j+k];
                a[j+k] = (X+Y)%mod;
                a[i+j+k] = (X+mod-Y)%mod;
                if(opt == -1) a[j+k] = 1ll*a[j+k]*inv2%mod,a[i+j+k] = 1ll*a[i+j+k]*inv2%mod;


                        }
                }
        }
}

 或
if(opt == 1) F[i+j+k] = (F[i+j+k]+F[j+k]) %mod;
else         F[i+j+k] = (F[i+j+k+mod-F[j+k]) %mod;
和
if(opt == 1) F[j+k] = (F[j+k]+F[i+j+k]) %mod;
else         F[j+k] = (F[j+k] +mod-F[i+j+k])%mod;
```

## 4.2 FFT

### 4.2.1 FFT

```
const double PI = acos(-1.0);
struct Complex
{
    double r,i;
    Complex(double _r = 0,double _i = 0){
        r = _r; i = _i;
```

```
    }
    Complex operator +(const Complex &b) {
        return Complex(r+b.r,i+b.i);
    }
    Complex operator -(const Complex &b) {
        return Complex(r-b.r,i-b.i);
    }
    Complex operator *(const Complex &b){
        return Complex(r*b.r-i*b.i,r*b.i+i*b.r);
    }
};

void FFT(Complex y[],int n ,int on)
{
    for(int i = 0, j = 0; i < n; i++) {
        if(j > i) swap(y[i], y[j]);
        int k = n;
        while(j & (k >>= 1)) j &= ~k;
            j |= k;
     }
    for(int h = 2;h <= n;h <<= 1){
        Complex wn(cos(-on*2*PI/h),sin(-on*2*PI/h));
        for(int j = 0;j < n;j += h){
            Complex w(1,0);
            for(int k = j;k < j+h/2;k++){
                Complex u = y[k];
                Complex t = w*y[k+h/2];
                y[k] = u+t;
                y[k+h/2] = u-t;
                w = w*wn;
            }
        }
    }
    if(on == -1)
        for(int i = 0;i < n;i++)
            y[i].r /= n;
}
```

### 4.2.2 kuangbin

```
#include <stdio.h>
#include <iostream>
#include <string.h>
#include <algorithm>
#include <math.h>
using namespace std;

const double PI = acos(-1.0);
struct complex
{
    double r,i;
    complex(double _r = 0,double _i = 0)
    {
        r = _r; i = _i;
```

```
    }
    complex operator +(const complex &b)
    {
        return complex(r+b.r,i+b.i);
    }
    complex operator -(const complex &b)
    {
        return complex(r-b.r,i-b.i);
    }
    complex operator *(const complex &b)
    {
        return complex(r*b.r-i*b.i,r*b.i+i*b.r);
    }
};
void change(complex y[],int len)
{
    int i,j,k;
    for(i = 1, j = len/2;i < len-1;i++)
    {
        if(i < j)swap(y[i],y[j]);
        k = len/2;
        while( j >= k)
        {
            j -= k;
            k /= 2;
        }
        if(j < k)j += k;
    }
}
void fft(complex y[],int len,int on)
{
    change(y,len);
    for(int h = 2;h <= len;h <<= 1)
    {
        complex wn(cos(-on*2*PI/h),sin(-on*2*PI/h));
        for(int j = 0;j < len;j += h)
        {
            complex w(1,0);
            for(int k = j;k < j+h/2;k++)
            {
                complex u = y[k];
                complex t = w*y[k+h/2];
                y[k] = u+t;
                y[k+h/2] = u-t;
                w = w*wn;
            }
        }
    }
    if(on == -1)
        for(int i = 0;i < len;i++)
            y[i].r /= len;
}

const int MAXN = 400040;
```

```
complex x1[MAXN];
int a[MAXN/4];
long long num[MAXN];//100000*100000 会超 int
long long sum[MAXN];

int main()
{
    int T;
    int n;
    scanf("%d",&T);
    while(T--)
    {
        scanf("%d",&n);
        memset(num,0,sizeof(num));
        for(int i = 0;i < n;i++)
        {
            scanf("%d",&a[i]);
            num[a[i]]++;
        }
        sort(a,a+n);
        int len1 = a[n-1]+1;
        int len = 1;
        while( len < 2*len1 )len <<= 1;
        for(int i = 0;i < len1;i++)
            x1[i] = complex(num[i],0);
        for(int i = len1;i < len;i++)
            x1[i] = complex(0,0);
        fft(x1,len,1);
        for(int i = 0;i < len;i++)
            x1[i] = x1[i]*x1[i];
        fft(x1,len,-1);
        for(int i = 0;i < len;i++)
            num[i] = (long long)(x1[i].r+0.5);
        len = 2*a[n-1];
        //减掉取两个相同的组合
        for(int i = 0;i < n;i++)
            num[a[i]+a[i]]--;
        //选择的无序，除以 2
        for(int i = 1;i <= len;i++)
        {
            num[i]/=2;
        }
        sum[0] = 0;
        for(int i = 1;i <= len;i++)
            sum[i] = sum[i-1]+num[i];
        long long cnt = 0;
        for(int i = 0;i < n;i++)
        {
            cnt += sum[len]-sum[a[i]];
            //减掉一个取大，一个取小的
            cnt -= (long long)(n-1-i)*i;
            //减掉一个取本身，另外一个取其它
            cnt -= (n-1);
            //减掉大于它的取两个的组合
```

```
            cnt -= (long long)(n-1-i)*(n-i-2)/2;
        }
        //总数
        long long tot = (long long)n*(n-1)*(n-2)/6;
        printf("%.7lf\n",(double)cnt/tot);
    }
    return 0;
}
```

### 4.2.3 lrj

```
#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int    prime = 999983;
const int    INF = 0x7FFFFFFF;
const LL     INFF =0x7FFFFFFFFFFFFFFF;
//const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-6;
const LL     mod = 1e9 + 7;
int dr[2][4] = {1,-1,0,0,0,0,-1,1};
// UVa12298 Super Poker II
// Rujia Liu

const long double PI = acos(0.0) * 2.0;

typedef complex<double> CD;

// Cooley-Tukey 的 FFT 算法，迭代实现。inverse = false 时计算逆 FFT
inline void FFT(vector<CD> &a, bool inverse) {
  int n = a.size();
  // 原地快速 bit reversal
  for(int i = 0, j = 0; i < n; i++) {
    if(j > i) swap(a[i], a[j]);
    int k = n;
    while(j & (k >>= 1)) j &= ~k;
    j |= k;
  }

  double pi = inverse ? -PI : PI;
  for(int step = 1; step < n; step <<= 1) {
    // 把每相邻两个“step 点 DFT”通过一系列蝴蝶操作合并为一个“2*step 点 DFT”
    double alpha = pi / step;
    // 为求高效，我们并不是依次执行各个完整的 DFT 合并，而是枚举下标 k
    // 对于一个下标 k，执行所有 DFT 合并中该下标对应的蝴蝶操作，即通过 E[k] 和 O[k] 计算 X[k]
    // 蝴蝶操作参考：http://en.wikipedia.org/wiki/Butterfly_diagram
    for(int k = 0; k < step; k++) {
      // 计算 omega^k. 这个方法效率低，但如果用每次乘 omega 的方法递推会有精度问题。
      // 有更快更精确的递推方法，为了清晰起见这里略去
```

```
      CD omegak = exp(CD(0, alpha*k));
      for(int Ek = k; Ek < n; Ek += step << 1) { // Ek 是某次 DFT 合并中 E[k] 在原始序列中的下标
        int Ok = Ek + step; // Ok 是该 DFT 合并中 O[k] 在原始序列中的下标
        CD t = omegak * a[Ok]; // 蝴蝶操作：x1 * omega^k
        a[Ok] = a[Ek] - t;  // 蝴蝶操作：y1 = x0 - t
        a[Ek] += t;         // 蝴蝶操作：y0 = x0 + t
      }
    }
  }

  if(inverse)
    for(int i = 0; i < n; i++) a[i] /= n;
}

// 用 FFT 实现的快速多项式乘法
inline vector<double> operator * (const vector<double>& v1, const vector<double>& v2) {
  int s1 = v1.size(), s2 = v2.size(), S = 2;
  while(S < s1 + s2) S <<= 1;
  vector<CD> a(S,0), b(S,0); // 把 FFT 的输入长度补成 2 的幂，不小于 v1 和 v2 的长度之和
  for(int i = 0; i < s1; i++) a[i] = v1[i];
  FFT(a, false);
  for(int i = 0; i < s2; i++) b[i] = v2[i];
  FFT(b, false);
  for(int i = 0; i < S; i++) a[i] *= b[i];
  FFT(a, true);
  vector<double> res(s1 + s2 - 1);
  for(int i = 0; i < s1 + s2 - 1; i++) res[i] = a[i].real(); // 虚部均为 0
  return res;
}
```

### 4.3 template

```
// 适用范围，求 n 次多项式第 x 项的值


namespace polysum {
    #define rep(i,a,n) for (int i=a;i<n;i++)
    #define per(i,a,n) for (int i=n-1;i>=a;i--)
    const int D=1e6+10;
    ll a[D],f[D],g[D],p[D],p1[D],p2[D],b[D],h[D][2],C[D];
    ll powmod(ll a,ll b){ll res=1;a%=mod;assert(b>=0);for(;b;b>>=1){if(b&1)res=res*a%mod;a=a*a%mod;}retu
    //.........................
   // 已知 a_i 的 d 次多项式，求第 n 项
    ll calcn(int d,ll *a,ll n) { // a[0].. a[d]  a[n]
        if (n<=d) return a[n];
        p1[0]=p2[0]=1;
        rep(i,0,d+1) {
            ll t=(n-i+mod)%mod;
            p1[i+1]=p1[i]*t%mod;
        }
        rep(i,0,d+1) {
            ll t=(n-d+i+mod)%mod;
            p2[i+1]=p2[i]*t%mod;
        }
```

```
        ll ans=0;
        rep(i,0,d+1) {
            ll t=g[i]*g[d-i]%mod*p1[i]%mod*p2[d-i]%mod*a[i]%mod;
            if ((d-i)&1) ans=(ans-t+mod)%mod;
            else ans=(ans+t)%mod;
        }
        return ans;
    }
    // 初始化, 初始化的时候记得将 D 的值
    void init(int M) {
        f[0]=f[1]=g[0]=g[1]=1;
        rep(i,2,M+5) f[i]=f[i-1]*i%mod;
        g[M+4]=powmod(f[M+4],mod-2);
        per(i,1,M+4) g[i]=g[i+1]*(i+1)%mod;
    }
// 已知 a_i, 并且知道 a_i 是 m 次多项式
   ll polysum(ll m,ll *a,ll n) { // a[0].. a[m] \sum_{i=0}^{n} a[i]
        ll b[D];
        ll b[D];
        for(int i=0;i<=m;i++) b[i]=a[i];
        b[m+1]=calcn(m,b,m+1);
        rep(i,1,m+2) b[i]=(b[i-1]+b[i])%mod;
        return calcn(m+1,b,n);// m 次多项式的和是 m+1 次多项式
    }

    ll qpolysum(ll R,ll n,ll *a,ll m) {
     // a[0].. a[m] \sum_{i=0}^{n-1} a[i]*R^i
        if (R==1) return polysum(n,a,m);
        a[m+1]=calcn(m,a,m+1);
        ll r=powmod(R,mod-2),p3=0,p4=0,c,ans;
        h[0][0]=0;h[0][1]=1;
        rep(i,1,m+2) {
            h[i][0]=(h[i-1][0]+a[i-1])*r%mod;
            h[i][1]=h[i-1][1]*r%mod;
        }
        rep(i,0,m+2) {
            ll t=g[i]*g[m+1-i]%mod;
            if (i&1) p3=((p3-h[i][0]*t)%mod+mod)%mod,p4=((p4-h[i][1]*t)%mod+mod)%mod;
            else p3=(p3+h[i][0]*t)%mod,p4=(p4+h[i][1]*t)%mod;
        }
        c=powmod(p4,mod-2)*(mod-p3)%mod;
        rep(i,0,m+2) h[i][0]=(h[i][0]+h[i][1]*c)%mod;
        rep(i,0,m+2) C[i]=h[i][0];
        ans=(calcn(m,C,n)*powmod(R,n)-c)%mod;
        if (ans<0) ans+=mod;
        return ans;
    }
} // polysum::init();
```

## 4.4 数论

### 4.4.1 1 加法

```
string add(string a,string b)
{
```

```
    string c;
    int len1=a.length();
    int len2=b.length();
    int len=max(len1,len2);
    for(int i=len1;i<len;i++)
        a="0"+a;
    for(int i=len2;i<len;i++)
        b="0"+b;
    int ok=0;
    for(int i=len-1;i>=0;i--)
    {
        char temp=a[i]+b[i]-'0'+ok;
        if(temp>'9')
        {
            ok=1;
            temp-=10;
        }
        else ok=0;
        c=temp+c;
    }
    if(ok) c="1"+c;
    return c;
}
```

### 4.4.2 1 逆元

```
// 欧几里得扩展
long long ex_gcd(long long a,long long b,long long &x,long long &y)
{
    if(b == 0)
    {
        x = 1;
        y = 0;
        return a;
    }
    long long m = ex_gcd(b,a%b,y,x);
     y  -= a/b * x;
    return m;
}
int main()
{
    long long a,b,x,y;
    cin>>a>>b; //求 a 关于 b 的逆元
    if(ex_gcd(a,b,x,y)==1)
        cout<<(x%b+b)%b<<endl;
    else
        cout<<"None"<<endl;
    return 0;
}
// 费马小定理求逆元
qpow(a,p-2,p);
// 逆元打表

   int inv[10000];
```

```
    int p;
    cin>>p;
    inv[1] = 1;
    for(int i = 2;i < p; ++i)
    {
        inv[i] = (p - p/i*inv[p%i]%p)%p;
    }
    for(int i = 1;i < p; ++i)
        cout<<inv[i]<<" ";
    cout<<endl;
    for(int i = 1;i < p; ++i)
        cout<<i * inv[i] % p<<" ";

 // 快速阶乘逆元

 const int maxn = 1e5+10;
long long fac[maxn],invfac[maxn];
void init(int n){
    fac[0] = 1;
    for(int i = 1;i <= n; ++i) fac[i] = fac[i-1]*i%mod;
    invfac[n] = qpow(fac[n],mod-2);
    for(int i = n-1;i >= 0; --i) invfac[i] = invfac[i+1]*(i+1)%mod;
}
```

#### 4.4.3 2 减法

```
string sub(string a,string b)
{
    string c;
    bool ok=0;
    int len1=a.length();
    int len2=b.length();
    int len=max(len1,len2);
    for(int i=len1;i<len;i++)
        a="0"+a;
    for(int i=len2;i<len;i++)
        b="0"+b;
    if(a<b)
    {
        string temp=a;
        a=b;
        b=temp;
        ok=1;
    }
    for(int i=len-1;i>=0;i--)
    {
        if(a[i]<b[i])
        {
            a[i-1]-=1;
            a[i]+=10;
        }
        char temp=a[i]-b[i]+'0';
        c=temp+c;
    }
```

```
    int pos=0;
    while(c[pos]=='0' && pos<len) pos++;
    if(pos==len) return "0";
    if(ok) return "-"+c.substr(pos);
    return c.substr(pos);
}
```

#### 4.4.4 3 乘法

```
string mul(string a,int b)
{
    string c;
    char s;
    int len=a.length();
    int ok=0;
    for(int i=len-1;i>=0;i--)
    {
        int temp=(a[i]-'0')*b+ok;
        ok=temp/10;
        s=temp%10+'0';
        c=s+c;
    }
    while(ok)
    {
        s=ok%10+'0';
        c=s+c;
        ok/=10;
    }
    return c;
}
```

#### 4.4.5 4 除法

```
string div(string a,int b)
{
    string c;
    int len=a.length();
    int ans=0;
    char s;
    for(int i=0;i<len;i++)
    {
        ans=ans*10+a[i]-'0';
        s=ans/b+'0';
        ans%=b;
        c+=s;
    }
    int pos=0;
    while(pos<len && c[pos]=='0') pos++;
    if(pos==len) return "0";
    return c.substr(pos);
}
```

### 4.4.6 Euler

#### 欧拉函数打表
O(nlog(n))
```

```
const int maxn = 1e6+100;
int phi[maxn],Prime[maxn];

void init2(int n){
        for(int i = 1;i <= n; ++i) phi[i] = i;
    for(int i = 2;i <= n; ++i){
            if(i == phi[i]){
               for(int j = i; j <= n; j += i) phi[j] = phi[j]/i*(i-1);
            }
    }
}
```
线性筛 O(n)
```
const int maxn = 1e6+100;
bool check[maxn];
int phi[maxn],Prime[maxn];
void init(int MAXN){
        int N = maxn-1;
    memset(check,false,sizeof(check));
    phi[1] = 1;
    int tot = 0;
    for(int i = 2;i <= N; ++i){
            if(!check[i]){
                    Prime[tot++] = i;
                    phi[i] = i-1;
            }
            for(int j = 0;j < tot; ++j){
                    if(i*Prime[j] > N) break;
                    check[i*Prime[j]] = true;
                    if(i%Prime[j] == 0){
                            phi[i*Prime[j]] = phi[i]*Prime[j];
                            break;
                    }
                    else{
                            phi[i*Prime[j]] = phi[i]*(Prime[j]-1);
                    }
            }
    }

}
```

### 4.4.7 lucas , 组合数

```
LL qpow(LL a,LL b,LL m){
        LL ans = 1;
        a %= m;
```
```

```
        while(b > 0){
                if(b&1)
                   ans = ans*a%m;
                   a = a*a%m;
                   b >>= 1;
        }
        return ans;
}
LL C(LL n,LL m,LL p){
        if(m > n) return 0;
        LL tmp1 = 1,tmp2 = 1;
        m = min(n-m,m);
        for(LL i = 1;i <= m; ++i){
                tmp1 = tmp1*(n-m+i)%p;
                tmp2 = tmp2*i%p;
        }
        return tmp1*qpow(tmp2,p-2,p)%p;
}
LL lucas(LL n, LL m, LL p){
        if(m == 0)
          return 1;
        return lucas(n/p,m/p,p)*C(n%p,m%p,p)%p;
}
```

### 4.4.8 miller-rabin-Pollard-rho

```
// 可以对一个 2^63 的素数进行判断。
```

可以分解比较大的数的因子。

```
#include<stdio.h>
#include<string.h>
#include<iostream>
#include<math.h>
#include<stdlib.h>
#include<time.h>
using namespace std;


typedef long long LL;
#define maxn 10000

LL factor[maxn];
int tot;
const int S=20;
LL muti_mod(LL a,LL b,LL c){    //返回 (a*b) mod c,a,b,c<2^63
    a%=c;
    b%=c;
    LL ret=0;
    while (b){
        if (b&1){
            ret+=a;
            if (ret>=c) ret-=c;
        }
```

```
        a<<=1;
        if (a>=c) a-=c;
        b>>=1;
    }
    return ret;
}

LL pow_mod(LL x,LL n,LL mod){  //返回 x^n mod c , 非递归版
    if (n==1) return x%mod;
    int bit[90],k=0;
    while (n){
        bit[k++]=n&1;
        n>>=1;
    }
    LL ret=1;
    for (k=k-1;k>=0;k--){
        ret=muti_mod(ret,ret,mod);
        if (bit[k]==1) ret=muti_mod(ret,x,mod);
    }
    return ret;
}

bool check(LL a,LL n,LL x,LL t){   //以 a 为基, n-1=x*2^t, 检验 n 是不是合数
    LL ret=pow_mod(a,x,n),last=ret;
    for (int i=1;i<=t;i++){
        ret=muti_mod(ret,ret,n);
        if (ret==1 && last!=1 && last!=n-1) return 1;
        last=ret;
    }
    if (ret!=1) return 1;
    return 0;
}

bool Miller_Rabin(LL n){
    LL x=n-1,t=0;
    while ((x&1)==0) x>>=1,t++;
    bool flag=1;
    if (t>=1 && (x&1)==1){
        for (int k=0;k<S;k++){
            LL a=rand()%(n-1)+1;
            if (check(a,n,x,t)) {flag=1;break;}
            flag=0;
        }
    }
    if (!flag || n==2) return 0;
    return 1;
}

LL gcd(LL a,LL b){
    if (a==0) return 1;
    if (a<0) return gcd(-a,b);
    while (b){
        LL t=a%b; a=b; b=t;
    }
```

```
    return a;
}

LL Pollard_rho(LL x,LL c){
    LL i=1,x0=rand()%x,y=x0,k=2;
    while (1){
        i++;
        x0=(muti_mod(x0,x0,x)+c)%x;
        LL d=gcd(y-x0,x);
        if (d!=1 && d!=x){
            return d;
        }
        if (y==x0) return x;
        if (i==k){
            y=x0;
            k+=k;
        }
    }
}

void findfac(LL n){            //递归进行质因数分解 N
    if (!Miller_Rabin(n)){
        factor[tot++] = n;
        return;
    }
    LL p=n;
    while (p>=n) p=Pollard_rho(p,rand() % (n-1) +1);
    findfac(p);
    findfac(n/p);
}

int main()
{
   // srand(time(NULL));//POJ 上 G++ 要去掉这句话
    int T;
    scanf("%d",&T);
    long long n;
    while(T--)
    {
        scanf("%I64d",&n);
        if (!Miller_Rabin(n)) {printf("Prime\n"); continue; }
        tot = 0;
        findfac(n);
        long long ans=factor[0];
        for(int i=1;i<tot;i++)
          if(factor[i]<ans)ans=factor[i];
        printf("%I64d\n",ans);
    }
    return 0;
}
```

### 4.4.9 快速数论变换

```
const int mod = 998244353;
LL qpow(LL a,LL b){LL s=1;while(b>0){if(b&1)s=s*a%mod;a=a*a%mod;b>>=1;}return s;}
const int g = 3;  //原根
LL quick_mod(LL a,LL b)
{
    LL ans=1;
    for(;b;b/=2)
    {
        if(b&1)
            ans=ans*a%mod;
        a=a*a%mod;
    }
    return ans;
}
int rev(int x,int r)  //蝴蝶操作
{
    int ans=0;
    for(int i=0; i<r; i++)
    {
        if(x&(1<<i))
        {
            ans+=1<<(r-i-1);
        }
    }
    return ans;
}
void NTT(int n, LL  A[],int on) // 长度为 N (2 的次数)
{
    int r=0;
    for(;; r++)
    {
        if((1<<r)==n)
            break;
    }
    for(int i=0; i<n; i++)
    {
        int tmp=rev(i,r);
        if(i<tmp)
            swap(A[i],A[tmp]);
    }
    for(int s=1; s<=r; s++)
    {
        int m=1<<s;
        LL wn=quick_mod(g,(mod-1)/m);
        for(int k=0; k<n; k+=m)
        {
            LL  w=1;
            for(int j=0; j<m/2; j++)
            {
                LL t,u;
                t=w*(A[k+j+m/2]%mod)%mod;
                u=A[k+j]%mod;
```

```
                A[k+j]=(u+t)%mod;
                A[k+j+m/2]=((u-t)%mod+mod)%mod;
                w=w*wn%mod;
            }
        }
    }
    if(on==-1)
    {
        for(int i=1;i<n/2;i++)
            swap(A[i],A[n-i]);
        LL inv=quick_mod(n,mod-2);
        for(int i=0;i<n;i++)
            A[i]=A[i]%mod*inv%mod;
    }

}
```

#### 4.4.10 欧拉筛和埃氏筛

```
void Era_s(void){
    check[1] = 1;
    tot = 1;
    for(int i = 2;i < maxn; ++i){
        if(!check[i]){
        Prime[tot++] = i;
        for(int j = i+i;j < maxn; ++j)  check[j] = 1;
        }
    }
}
void Euler_s(void){
    check[1] = 1;
    tot = 1;
    int n = 1e6;
    for(int i = 2;i <= n; ++i){
         if(!check[i]) Prime[tot++] = i;
         for(int j = 1;j < tot; ++j){
            if(i*Prime[j] > n) break;
            check[i*Prime[j]] = 1;
            if(i % Prime[j] == 0) break;
         }
    }
}
```

#### 4.4.11 素性检测

```
#include<bits/stdc++.h>

using namespace std;
//typedef long long LL;
const int LEN  = 1e6+1;
bool vis[LEN];
//int prime[LEN];
int Prime[LEN];
int cnt = 1;
typedef unsigned long long LL;
```

```
LL modular_multi(LL x,LL y,LL mo) {
        LL t;
        x%=mo;
        for(t=0;y;x=(x<<1)%mo,y>>=1)
                if (y&1)
                        t=(t+x)%mo;
        return t;
}

LL modular_exp(LL num,LL t,LL mo) {
        LL ret=1,temp=num%mo;
        for(;t;t>>=1,temp=modular_multi(temp,temp,mo))
                if (t&1)
                        ret=modular_multi(ret,temp,mo);
        return ret;
}

bool miller_rabin(LL n) {
        if (n==2||n==7||n==61)
        return true;
        if (n==1||(n&1)==0)
        return false;
        int t=0,num[3]={2,7,61};//2,7,61 对 unsigned int 内的所有数够用了，最小不能判断的数为 4 759 123 1
        LL a,x,y,u=n-1;
        while((u&1)==0)
        t++,u>>=1;
        for(int i=0;i<3;i++) {
                a=num[i];
                x=modular_exp(a,u,n);
                for(int j=0;j<t;j++) {
                        y=modular_multi(x,x,n);
                        if (y==1&&x!=1&&x!=n-1)
                                return false;
            //其中用到定理，如果对模 n 存在 1 的非平凡平方根，则 n 是合数。
            //如果一个数 x 满足方程 x^2 1 (mod n)，但 x 不等于对模 n 来说 1 的两个‘平凡’平方根：1 或-1，
                        x=y;
                }
                if (x!=1)//根据费马小定理，若 n 是素数，有 a^(n-1) 1(mod n). 因此 n 不可能是素数
                        return false;
        }
        return true;
}
void init(void)
{
    int n = LEN -1;
    for(int i = 2; i <= n; ++i)
    {
        if(!vis[i])
        {
            Prime[cnt++] = i;
            for(LL j = (LL)i *  i; j <= n; j += i)
                vis[j] = 1;
        }
```

```
    }
}
bool isPrime(LL n)
{
        if(n < 1e6)
        {
                for(LL i = 1;i < cnt&&Prime[i] < n; ++i)
        {
                if(n % Prime[i] == 0)
                 return false;
        }
        return true;
        }
        else
         return miller_rabin(n);
 }

int main(void)
{
        init();

        int T;
        cin>>T;
        while(T--)
        {
           LL n;
           cin>>n;
           if(isPrime(n))
              cout<<"Yes"<<endl;
           else
              cout<<"No"<<endl;
        }

        return 0;
}
```

#### 4.4.12 素数筛

```
Eratosthenes筛法（埃拉托斯特尼筛法）
onst int maxn = 1e6+10;
bool check[maxn];
int Prime[maxn];
int tot = 1;
void Eratosthenes(void){
        const int n = maxn -1;
        memset(check,0,sizeof(check));
        for(int i = 2;i < n; ++i){
                if(!check[i]){
                        Prime[tot++] = i;
                        for(int j = i+i;j < n;j += i) check[j] = 1;
                }
        }
}
```

```
欧拉筛
```
```
const int maxn = 1e6+10;
bool check[maxn];
int Prime[maxn];
int tot = 1;
void Euler_shai(void){
        int n = maxn-1;
        memset(check,0,sizeof(check));
        for(int i = 2;i <= n; ++i){
                if(!check[i]){
                        Prime[tot++] = i;
                }
                        for(int j = 1;j < tot; ++j){
                                if(i*Prime[j] > n) break;
                                    check[i*Prime[j]]  =1 ;
                            if(i % Prime[j]==0) break;
                        }
        }
}

```

#### 4.4.13 逆元打表

```
 int inv[10000];
    int p;
    cin>>p;
    inv[1] = 1;
    for(int i = 2;i < p; ++i)
    {
        inv[i] = (p - p/i*inv[p%i]%p)%p;
    }
    for(int i = 1;i < p; ++i)
        cout<<inv[i]<<" ";
    cout<<endl;
    for(int i = 1;i < p; ++i)
        cout<<i * inv[i] % p<<" ";
```

### 4.5 矩阵快速幂.cpp

```
// 注意修改 maxn 的值，要不然容易 T

const int maxn = 100;
int n;
struct Matrix{
        int n,m;
        Matrix(int nn = 1,int mm = 1):n(nn),m(mm){ memset(a,0,sizeof(a));};
        long long a[maxn][maxn];
};
// void print(const Matrix &a)
// {
//         for(int i = 1;i <= a.n; ++i,cout<<endl)
```

```
//          for(int j= 1;j <= a.m; ++j)
//             cout<<a.a[i][j]<<" ";
// }
Matrix operator*(Matrix a,Matrix b)
{
        Matrix c(a.n,b.m);
        for(int i = 1;i <= a.n; ++i)
        {
                for(int j = 1;j <= b.m; ++j)
                {
                        for(int k = 1;k <= a.m; ++k)
                        {
                                c.a[i][j] += a.a[i][k] * b.a[k][j];
                                c.a[i][j] %= mod;
                        }
                }
        }
//        print(c);
        return c;
}
```

### 4.6 自适应辛普森积分.cpp

```
double F(double x)
{
        //Simpson 公式用到的函数
}
double simpson(double a, double b)//三点 Simpson 法, 这里要求 F 是一个全局函数
{
        double c = a + (b - a) / 2;
        return (F(a) + 4 * F(c) + F(b))*(b - a) / 6;
}
double asr(double a, double b, double eps, double A)//自适应 Simpson 公式（递归过程）。已知整个区间 [a,b]
{
        double c = a + (b - a) / 2;
        double L = simpson(a, c), R = simpson(c, b);
        if (fabs(L + R - A) <= 15 * eps)return L + R + (L + R - A) / 15.0;
        return asr(a, c, eps / 2, L) + asr(c, b, eps / 2, R);
}
double asr(double a, double b, double eps)//自适应 Simpson 公式（主过程）
{
        return asr(a, b, eps, simpson(a, b));
}
```

# 5 5 几何

## 5.1 2D

### 5.1.1 PSLG

```
typedef vector<Point> Polygon;
double PolygonArea(Polygon poly)
{
    double area = 0;
```

```
    int n = poly.size();
    for(int i = 1; i < n-1; i++)
        area += Cross(poly[i]-poly[0], poly[(i+1)%n]-poly[0]);
    return area/2;
}

struct Edge
{
    int from, to; // 起点，终点，左边的面编号
    double ang;
    Edge(int f,int t,double a):from(f),to(t),ang(a) {}
};

const int maxn = 10000 + 10; // 最大边数

// 平面直线图（PSGL）实现
struct PSLG
{
    int n, m, face_cnt;//face_cnt 面数
    double x[maxn], y[maxn];
    vector<Edge> edges;//储存边
    vector<int> G[maxn];//指向边
    int vis[maxn*2];  // 每条边是否已经访问过
    int left[maxn*2]; // 左面的编号
    int prev[maxn*2]; // 相同起点的上一条边（即顺时针旋转碰到的下一条边）的编号

    vector<Polygon> faces;//faces 储存面
    double area[maxn]; // 每个 polygon 的面积

    void init(int n)
    {
        this->n = n;
        for(int i = 0; i < n; i++)
            G[i].clear();
        edges.clear();
        faces.clear();
    }

    // 有向线段 from->to 的极角
    double getAngle(int from, int to)
    {
        return atan2(y[to]-y[from], x[to]-x[from]);
    }

    void AddEdge(int from, int to)
    {
        edges.push_back((Edge){ from, to, getAngle(from, to)});
        edges.push_back((Edge){ to, from, getAngle(to, from)});
        m = edges.size();
        G[from].push_back(m-2);
        G[to].push_back(m-1);
    }

    // 找出 faces 并计算面积
```

```
    void Build()
    {
        for(int u = 0; u < n; u++)
        {
            // 给从 u 出发的各条边按极角排序
            int d = G[u].size();
            for(int i = 0; i < d; i++)
                for(int j = i+1; j < d; j++) // 这里偷个懒，假设从每个点出发的线段不会太多
                    if(edges[G[u][i]].ang > edges[G[u][j]].ang)
                        swap(G[u][i], G[u][j]);
            for(int i = 0; i < d; i++)
                prev[G[u][(i+1)%d]] = G[u][i];
        }

        memset(vis, 0, sizeof(vis));
        face_cnt = 0;
        for(int u = 0; u < n; u++)
            for(int i = 0; i < G[u].size(); i++)
            {
                int e = G[u][i];
                if(!vis[e])   // 逆时针找圈
                {
                    face_cnt++;
                    Polygon poly;
                    for(;;)
                    {
                        vis[e] = 1;
                        left[e] = face_cnt;
                        int from = edges[e].from;
                        poly.push_back(Point(x[from], y[from]));
                        e = prev[e^1];
                        if(e == G[u][i])
                            break;
                        assert(vis[e] == 0);
                    }
                    faces.push_back(poly);
                }
            }

        for(int i = 0; i < faces.size(); i++)
        {
            area[i] = PolygonArea(faces[i]);
        }
    }
};
```

### 5.1.2 二维几何模板

```
#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
#define forn(i,n) for(int i = 0;i < n; ++i)
using namespace std;
```

```
typedef long long LL;
typedef unsigned long long ULL;
const int    prime = 999983;
const int    INF = 0x7FFFFFFF;
const LL     INFF =0x7FFFFFFFFFFFFFFF;
const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-10;
const LL     mod = 1e9 + 7;
struct Point
{
    double x,y;

    Point(double x = 0,double y = 0):x(x),y(y) {}

};
typedef Point Vector;
Vector operator + (Vector A,Vector B)
{
    return Vector(A.x + B.x,A.y + B.y);
}
Vector operator - (Vector A,Vector B)
{
    return Vector(A.x-B.x,A.y-B.y);
}
Vector operator / (Vector A,double p)
{
    return Vector(A.x/p,A.y/p);
}
Vector operator * (Vector A,double p)
{
    return Vector(A.x*p,A.y*p);
}
double angle(Vector v)//求向量的角度从 0 到 2*pi
{
    return atan2(v.y,v.x);
}
int dcmp(double x)
{
    if(fabs(x)<eps)
        return 0;
    else
        return x < 0?-1:1;
}
bool operator < (const Point &a,const Point &b)
{
    if(dcmp(a.x-b.x)==0)
        return a.y<b.y;
    else
        return a.x<b.x;
}


bool operator == (const Point &a,const Point &b)
```

```
{
    return !dcmp(a.x-b.x)&&!dcmp(a.y-b.y);
}
double Dot(Vector A,Vector B)
{
    return A.x*B.x+A.y*B.y;
}
double Length(Vector A)
{
    return sqrt(A.x*A.x+A.y*A.y);
}
double Angle(Vector A,Vector B)
{
    return acos(Dot(A,B)/Length(A)/Length(B));
}
double Cross(Vector A,Vector B)
{
    return A.x*B.y - A.y*B.x;
}
double Area2(Point A,Point B,Point C)
{
    return Cross(B-A,C-A);
}
Vector Rotate(Vector A,double rad)
{
    return Vector (A.x*cos(rad)-A.y*sin(rad),A.x*sin(rad)+A.y*cos(rad));
}
Vector Normal(Vector A)//单位法线
{
    double L = Length(A);
    return Vector(-A.y/L,A.x/L);
}
//调用前确保直线有唯一交点，当且仅当 Cross(v,w) 非 0
Point Get_Line_Intersection(Point P,Vector v,Point Q,Vector w)
{
    Vector u = P - Q;
    double t = Cross(w,u)/Cross(v,w);
    return P+v*t;
}
double Distance_To_Line(Point P,Point A,Point B)//点到直线的距离
{
    Vector v1 = B-A,v2 = P-A;
    return fabs(Cross(v1,v2)/Length(v1));
}
double Distance_To_Segment(Point P,Point A,Point B)
{
    if(A==B)
        return Length(P-A);
    Vector v1 = B-A,v2 = P-A,v3 = P-B;
    if(dcmp(Dot(v1,v2))<0)
        return Length(v1);
    else if(dcmp(Dot(v1,v3))>0)
        return Length(v3);
    else
```

```
        return fabs(Cross(v1,v2))/Length(v1);
}
Point Get_Line_Projection(Point P,Point A,Point B)//求投影点
{
    Vector v = B- A;
    return A + v*(Dot(v,P-A)/Dot(v,v));
}
//线段相交判定 相交不在线段的端点
bool Segment_Proper_Intersection(Point a1,Point a2,Point b1,Point b2)
{
    double c1 =  Cross(a2-a1,b1-a1),c2 = Cross(a2-a1,b2-a1),
           c3 =  Cross(b2-b1,a2-b1),c4 = Cross(b2-b1,a1-b1);
    return dcmp(c1)*dcmp(c2)<0&&dcmp(c3)*dcmp(c4)<0;
}
//判断点是否在线段上 (不包括端点)
bool Onsegment(Point p,Point a1,Point a2)
{
    return dcmp(Cross(a1-p,a2-p))==0&&dcmp(Dot(a1-p,a2-p))<0;
}
```

### 5.1.3 二维凸包

```
//计算凸包, 输入点数组 p, 个数为 p, 输出点数组为 ch。函数返回凸包顶点数
//输入不能有重复节点
//如果精度要求搞需要用 dcmp 判断
//如果不希望在边上右点, 需要将 <= 改为 <
int ConvexHull(Point *p,int n ,Point *ch)
{
    sort(p,p+n);
    int m = 0;
    for(int i = 0;i < n; ++i)
    {
        while(m>1&& Cross(ch[m-1]-ch[m-2],p[i]-ch[m-2])<=0) m--;
        ch[m++] = p[i];

    }
    int k = m;
    for(int i = n-2; i >= 0; --i)
    {
        while(m > k&& Cross(ch[m-1]-ch[m-2],p[i]-ch[m-2]) <= 0) m--;
        ch[m++] = p[i];
    }
    if(n > 1) m--;
    return m;
}
```

### 5.1.4 判断点是否在多边形内

```
typedef vector<Point> Polygon;
int isPointInPolygon(Point p,Polygon poly)
{
    int n  = poly.size();
    int wn = 0;
    for(int i = 0;i < n; ++i)
    {
```

```
        if(Onsegment(p,poly[i],poly[(i+1)%n])) return -1;
        int k = dcmp(Cross(poly[(i+1)%n]-poly[i],p-poly[i]));
        int d1 = dcmp(poly[i].y-p.y);
        int d2 = dcmp(poly[(i+1)%n].y-p.y);
        if(k>0&&d1 <= 0&&d2 > 0) wn ++;
        if(k<0&&d2 <= 0&&d1 > 0) wn --;
    }
    if(wn != 0)  return 1;
    return 0;
}
```

### 5.1.5 圆与多边形相交的面积

```
#include <iostream>
#include <cstdio>
#include <string>
#include <cmath>
#include <iomanip>
#include <ctime>
#include <climits>
#include <cstdlib>
#include <cstring>
#include <algorithm>
#include <queue>
#include <vector>
#include <set>
#include <map>
using namespace std;
typedef unsigned int UI;
typedef long long LL;
typedef unsigned long long ULL;
typedef long double LD;
const double pi = acos(-1.0);
const double e = exp(1.0);
const double eps = 1e-8;
const int maxn = 400;
double x, y, h;
double vx, vy;
double R;
int n;
struct point
{
    double x, y;
    point(double _x=0.0, double _y=0.0)
        : x(_x), y(_y) {}
    point operator - (const point & p)
    {
        return point(x-p.x, y-p.y);
    }
    double sqrx()
    {
        return sqrt(x*x+y*y);
    }
} p[maxn];
```

```
double xmult(point & p1, point & p2, point & p0);
double distancex(point & p1, point & p2);
point intersection(point u1, point u2, point v1, point v2);
void intersection_line_circle(point c, double r, point l1, point l2, point & p1, point & p2);
point ptoseg(point p, point l1, point l2);
double distp(point & a, point & b);
double Direct_Triangle_Circle_Area(point a, point b, point o, double r);


double xmult(point & p1, point & p2, point & p0)
{
    return (p1.x-p0.x)*(p2.y-p0.y)-(p1.y-p0.y)*(p2.x-p0.x);
}

double distancex(point & p1, point & p2)
{
    return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
}

point intersection(point u1, point u2, point v1, point v2)
{
    point ret = u1;
    double t = ((u1.x-v1.x)*(v1.y-v2.y)-(u1.y-v1.y)*(v1.x-v2.x))
             / ((u1.x-u2.x)*(v1.y-v2.y)-(u1.y-u2.y)*(v1.x-v2.x));
    ret.x += (u2.x-u1.x)*t;
    ret.y += (u2.y-u1.y)*t;
    return ret;
}

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

point ptoseg(point p, point l1, point l2)
{
    point t = p;
    t.x += l1.y-l2.y;
    t.y += l2.x-l1.x;
    if (xmult(l1, t, p)*xmult(l2, t, p)>eps)
        return distancex(p, l1)<distancex(p, l2) ? l1 : l2;
    return intersection(p, t, l1, l2);
}
```

```
double distp(point & a, point & b)
{
    return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
}

double Direct_Triangle_Circle_Area(point a, point b, point o, double r)
{
    double sign = 1.0;
    a = a-o;
    b = b-o;
    o = point(0.0, 0.0);
    if (fabs(xmult(a, b, o)) < eps)
        return 0.0;
    if (distp(a, o) > distp(b, o))
    {
        swap(a, b);
        sign = -1.0;
    }
    if (distp(a, o) < r*r+eps)
    {
        if (distp(b, o) < r*r+eps)
            return xmult(a, b, o)/2.0*sign;
        point p1, p2;
        intersection_line_circle(o, r, a, b, p1, p2);
        if (distancex(p1, b) > distancex(p2, b))
            swap(p1, p2);
        double ret1 = fabs(xmult(a, p1, o));
        double ret2 = acos((p1.x*b.x+p1.y*b.y)/p1.sqrx()/b.sqrx())*r*r;
        double ret = (ret1+ret2)/2.0;
        if (xmult(a, b, o)<eps && sign>0.0 || xmult(a, b, o)>eps && sign<0.0)
            ret = -ret;
        return ret;
    }
    point ins = ptoseg(o, a, b);
    if (distp(o, ins)>r*r-eps)
    {
        double ret = acos((a.x*b.x+a.y*b.y)/a.sqrx()/b.sqrx())*r*r/2.0;
        if (xmult(a, b, o)<eps && sign>0.0 || xmult(a, b, o)>eps && sign<0.0)
            ret = -ret;
        return ret;
    }
    point p1, p2;
    intersection_line_circle(o, r, a, b, p1, p2);
    double cm = r/(distancex(o, a)-r);
    point m = point((o.x+cm*a.x)/(1+cm), (o.y+cm*a.y)/(1+cm));
    double cn = r/(distancex(o, b)-r);
    point n = point((o.x+cn*b.x)/(1+cn), (o.y+cn*b.y)/(1+cn));
    double ret1 = acos((m.x*n.x+m.y*n.y)/m.sqrx()/n.sqrx())*r*r;
    double ret2 = acos((p1.x*p2.x+p1.y*p2.y)/p1.sqrx()/p2.sqrx())*r*r-fabs(xmult(p1, p2, o));
    double ret = (ret1-ret2)/2.0;
    if (xmult(a, b, o)<eps && sign>0.0 || xmult(a, b, o)>eps && sign<0.0)
        ret = -ret;
    return ret;
}
```

```
double Inter(double x,double y,double R,int n,point *area){
        area[n] = area[0];
        point temp = point(x, y);
        double sum = 0;
        for (int i=0; i<n-1; i++)
            sum += Direct_Triangle_Circle_Area(area[i], area[i+1], temp, R);

        sum += Direct_Triangle_Circle_Area(area[n-1], area[0], temp, R);
        return fabs(sum);
}
double Cross(point A,point B)
{
    return A.x*B.y - A.y*B.x;
}
int N,M;
double PolygonArea (point * p,int n)
{
    double area = 0;
    for(int i = 1; i < n - 1; ++i)
    {
        area += Cross(p[i]-p[0],p[i+1]-p[0]);
    }
    return fabs(area/2);
}

int dcmp(double x)
{
    if(fabs(x)<eps)
        return 0;
    else
        return x < 0?-1:1;
}
double S ;
double xi,yi,P,Q;
bool check(double R){
        //        cout<<xi<<" "<<yi<<" "<<P<<" "<<Q<<endl;
//        printf("r = %lf Intersect = %lf\n",R,Inter(xi,yi,R,N,p) );
//        printf("%lf\n",(1-P/Q)*S);
        return  dcmp(Inter(xi,yi,R,N,p) - (1-P/Q)*S) > 0;
}
int main()
{

        cin>>N;
        for(int i=0;i< N;i++)
        {
            scanf("%lf%lf",&p[i].x,&p[i].y);
        }

    S= PolygonArea(p,N);
    //cout<<S<<endl;
    cin>>M;
        for(int i = 0;i < M; ++i){
```

```
                scanf("%lf %lf %lf %lf",&xi,&yi,&P,&Q);

                double l = 0,r = 1e6;
                for(int j = 0;j < 100; ++j){
                        double mid = l+(r-l)/2;
                        if(check(mid))
                             r = mid;
                        else
                             l = mid;
                 // printf("%lf %lf\n",l,r);
                }
                printf("%.8lf\n",r);
        }

    return 0;
}
```

#### 5.1.6 求圆与直线的交点

```
int getLineCircleIntersection(Point A, Point B, Point C, double r, double& t1, double& t2,vector<Point>
  // 初始方程：(A.x + t(B.x - A.x) - C.x)^2 + (A.y + t(B.y - A.y) - C.y)^2 = r^2
  // 整理得：(at + b)^2 + (ct + d)^2 = r^2
  double a = B.x - A.x;
  double b = A.x - C.x;
  double c = B.y - A.y;
  double d = A.y - C.y;
  // 展开得：(a^2 + c^2)t^2 + 2(ab + cd)t + b^2 + d^2 - r^2 = 0, 即 et^2 + ft + g = 0
  double e = a * a + c * c;
  double f = 2 * (a * b + c * d);
  double g = b * b + d * d - r * r;
  double delta = f * f - 4 * e * g; // 判别式
  if(dcmp(delta) < 0) return 0; // 相离
  if(dcmp(delta) == 0){ // 相切
    t1 = t2 = -f / (2 * e);
    sol.push_back(A+(B-A)*t1);
    return 1;
  }
  t1 = (-f - sqrt(delta)) / (2 * e);
  t2 = (-f + sqrt(delta)) / (2 * e);
   sol.push_back(A+(B-A)*t1);
   sol.push_back(A+(B-A)*t2);
  return 2;
}
```

### 5.2 3D

#### 5.2.1 三维几何的基本操作

```
#include <bits/stdc++.h>

using namespace std;
struct Point3
{
    double x,y,z;
    Point3(double x = 0,double y = 0,double z = 0):x(x),y(y),z(z) {}
```

```
};
typedef Point3 Vector3;

Vector3 operator +(Vector3 v1,Vector3 v2)
{
    return Vector3(v1.x+v2.x,v1.y+v2.y,v1.z+v2.z);
}
Vector3 operator -(Vector3 v1,Vector3 v2)
{
    return Vector3(v1.x-v2.x,v1.y-v2.y,v1.z-v2.z);
}
Vector3 operator *(Vector3 v,double c)
{
    return Vector3(v.x*c,v.y*c,v.z*c);
}
Vector3 operator /(Vector3 v,double c)
{
    return Vector3(v.x/c,v.y/c,v.z/c);
}
double  Dot(Vector3 A,Vector3 B)
{
    return  A.x*B.x+A.y*B.y+A.z*B.z;
}
double Length(Vector3 A)
{
    return  sqrt(Dot(A,A));
}
double Angle(Vector3 A,Vector3 B)
{
   return acos(Dot(A,B)/(2*Length(A)*Length(B)));
}
double DistanceToplane(const Point3 &p,const Point3 &p0,const Vector3& n)
{
    return fabs(Dot(p-p0,n))/Length(n);
}
Point3  GetPlaneProjection(const Point3&p,const Point3&p0,const Vector3&n)
{
    return p-n*Dot(p-p0,n);
}
//直线 p1-p2 到平面 p0-n 的交点。 假定交点唯一存在
Point3 LinePlaneIntetsection(Point3 p1,Point3 p2,Point3 p0,Vector3 n)
{
    Vector3 v= p2 - p1;
//    /*if(dcmp(Dot(v,n))==0)
//    {
//        if(dcmp(Dot(p1-p0,n))==0)
//            直线在平面上
//        else
//            直线与平面平行
//    }
//    */
    double t  = Dot(n,p0-p1)/Dot(n,p2-p1);
    return p1 + v*t;
}
```

### 5.2.2 三维几何的模版

```
#include <bits/stdc++.h>
const double eps = 1e-6;
using namespace std;

struct Point3
{
    double x,y,z;
    Point3(double x = 0,double y = 0,double z = 0):x(x),y(y),z(z) {}
};
typedef Point3 Vector3;
int dcmp(double d)
{
    if(fabs(d)< eps)
        return 0;
    else
        return d < 0?-1:1;
}
Vector3 operator +(Vector3 v1,Vector3 v2)
{
    return Vector3(v1.x+v2.x,v1.y+v2.y,v1.z+v2.z);
}
Vector3 operator -(Vector3 v1,Vector3 v2)
{
    return Vector3(v1.x-v2.x,v1.y-v2.y,v1.z-v2.z);
}
Vector3 operator *(Vector3 v,double c)
{
    return Vector3(v.x*c,v.y*c,v.z*c);
}
Vector3 operator /(Vector3 v,double c)
{
    return Vector3(v.x/c,v.y/c,v.z/c);
}
bool operator ==(Point3 A,Point3 B)
{
   return !dcmp(A.x-B.x)&&!dcmp(A.y-B.y)&&!dcmp(A.z-B.z);
}
double  Dot(Vector3 A,Vector3 B)
{
    return  A.x*B.x+A.y*B.y+A.z*B.z;
}
double Length(Vector3 A)
{
    return  sqrt(Dot(A,A));
}
double Angle(Vector3 A,Vector3 B)//求两向量的夹角
{
    return acos(Dot(A,B)/(2*Length(A)*Length(B)));
}
double DistanceToplane(const Point3 &p,const Point3 &p0,const Vector3& n)//
{
    return fabs(Dot(p-p0,n))/Length(n);
```

```
}
Point3  GetPlaneProjection(const Point3&p,const Point3&p0,const Vector3&n)
{
    return p-n*Dot(p-p0,n);
}
//直线 p1-p2 到平面 p0-n 的交点。 假定交点唯一存在
Point3 LinePlaneIntetsection(Point3 p1,Point3 p2,Point3 p0,Vector3 n)
{
    Vector3 v= p2 - p1;
//    /*if(dcmp(Dot(v,n))==0)
//    {
//        if(dcmp(Dot(p1-p0,n))==0)
//            直线在平面上
//        else
//            直线与平面平行
//    }
//    */
    double t  = Dot(n,p0-p1)/Dot(n,p2-p1);
    return p1 + v*t;
}
Point3 LinePlaneIntetsection(Point3 p1,Point3 p2,double A,double B,double C,double D)
{
    Vector3 v = p2-p1;
    double t = (A*p1.x+B*p1.y+C*p1.z+D)/(A*(p1.x-p2.x)+B*(p1.y-p2.y)+C*(p1.z-p2.z));
    return p1 + v*t;
}
Vector3 Cross(Vector3 A,Vector3 B)
{
    return Vector3(A.y*B.z-A.z*B.y,A.z*B.x-A.x*B.z,A.x*B.y-A.y*B.x);
}
double Area2(Point3 A,Point3 B,Point3 C)
{
    return Length(Cross(B-A,C-A));
}
////已知平面的三点，求出点法式
//Vector3 Solven(Point3 A,Point3 B,Point3 C)
//{
//    return Cross(B-A,C-A);
//}
//判断一个点是否在三角形内，可以用面积法
bool PointInTri(Point3 P,Point3 A,Point3 B,Point3 C)
{
    double area1 = Area2(P,A,B);
    double area2 = Area2(P,A,C);
    double area3 = Area2(P,B,C);
    double area4 = Area2(A,B,C);
    return dcmp(area1+area2+area3-area4)==0;
}
//判断线段是否与三角形相交
bool TriSegIntersection(Point3 P0,Point3 P1,Point3 P2,Point3 A,Point3 B,Point3 &P)
{
    Vector3 n = Cross(P1-P0,P2-P0);

    if(dcmp(Dot(n,B-A))==0)
```

```
        return false;

    double t = Dot(n,P0-A)/Dot(n,B-A);
    if(dcmp(t) < 0 || dcmp(t-1) > 0)
        return false;
    P = A + (B-A) * t;
    return PointInTri(P,P0,P1,P2);
}
double DitantceToLine(Point3 P,Point3 A,Point3 B)
{
    return Length(Cross(A-P,B-P))/Length(A-B);
}
double DistanceToSegment(Point3 P,Point3 A,Point3 B)
{
   if(A==B) return Length(P-A);
   Vector3 v1 = B - A, v2 = P - A,v3 = P-B;
   if(dcmp(Dot(v1,v2)) == 0) return Length(v2);
   if(dcmp(Dot(v1,v3)) >  0) return Length(v3);
   return Length(Cross(v1,v2))/Length(v1);
}
double Volume6(Point3 A,Point3 B,Point3 C,Point3 D)
{
    return Dot(D-A,Cross(B-A,C-A));
}
//
int  main(void)
{

    Point3 A(0,0,0),B(0,100,0),C(100,0,0),D(25,25,0);
    cout<<PointInTri(D,A,B,C)<<endl;
    return 0;
}
```

### 5.2.3 三维凸包

```
struct Face{
    int v[3];
    Vector3 normal(Vector *P)
    {
        return Cross(P[v[1]]-P[v[0]],P[v[2]]-P[v[0]]);
    }
    int cansee(Point *P,int i)const
    {
        return Dot(P[i]-P[v[0]],normal(P)) > 0?1 : 0;
    }
};
vector <Face> CH3D(Point3* P,int n)
{
    vector <Face> cur;
    cur.push_back((Face){{0,1,2}});
    cur.push_back((Face){{2,1,0}});
    for(int i = 3;i < n; ++i)
    {
        vector<Face> next;
```

```
        //计算每条边“左面”的可见性
        for(int j= 0;j < cur.size(); ++j)
        {
            Face &f = cur[j];
            int res = f.cansee(P,i);
            if(!res) next.push_back(f);
            for(int k = 0;k < 3; ++k)
                vis[f.v[k]][f.v[(k+1)%3]] = res;
        }
        for(int j = 0;j < cur.size(); ++j)
        {
            for(int k = 0;k < 3; ++k)
            {
                int a  = cur[j].v[k],b = cur[j].v[(k+1)%3];
                if(vis[a][b] != vis[b][a]&&vis[a][b])//(a,b) 是分界线，左边对 P[i] 可见
                 next.push_back((Face){{a,b,i}});
            }
        }
        cnr = next;
    }
    return cur;
}
double rand01() {return rand() / (double) RAND_MAX;}//0-1 的随机数
double randeps() {return (rand01()-0.5) * eps;}
Point3 add_noise(Point3 p)
{
    return Point3(p.x + randeps(),p.y+randeps(),p.z+randeps());
}

//.....................................................................
struct Face{
    int v[3];
    Vector3 normal(Vector *P)
    {
        return Cross(P[v[1]]-P[v[0]],P[v[2]]-P[v[0]]);
    }
    int cansee(Point *P,int i)const
    {
        return Dot(P[i]-P[v[0]],normal(P)) > 0?1 : 0;
    }
};
vector <Face> CH3D(Point3* P,int n)
{
    vector <Face> cur;
    cur.push_back((Face){{0,1,2}});
    cur.push_back((Face){{2,1,0}});
    for(int i = 3;i < n; ++i)
    {
        vector<Face> next;
        //计算每条边“左面”的可见性
        for(int j= 0;j < cur.size(); ++j)
        {
            Face &f = cur[j];
            int res = f.cansee(P,i);
```

```
            if(!res) next.push_back(f);
            for(int k = 0;k < 3; ++k)
                vis[f.v[k]][f.v[(k+1)%3]] = res;
        }
        for(int j = 0;j < cur.size(); ++j)
        {
            for(int k = 0;k < 3; ++k)
            {
                int a  = cur[j].v[k],b = cur[j].v[(k+1)%3];
                if(vis[a][b] != vis[b][a]&&vis[a][b])//(a,b) 是分界线，左边对 P[i] 可见
                 next.push_back((Face){{a,b,i}});
            }
        }
        cnr = next;
    }
    return cur;
}
double rand01() {return rand() / (double) RAND_MAX;}//0-1 的随机数
double randeps() {return (rand01()-0.5) * eps;}
Point3 add_noise(Point3 p)
{
    return Point3(p.x + randeps(),p.y+randeps(),p.z+randeps());
}
```

#### 5.2.4 维度转换为三维坐标

```
// 经纬度转换为球坐标
double torad(double deg)
{
    return deg/180*acos(-1);
}
void get_coordinate(double R,double lat,double lng,double &x,double &y,double &z)
{
    lat = torad(lat);
    lng = torad(lng);
    x = R*cos(lat)*cos(lng);
    y = R*cos(lat)*sin(lng);
    z = R*sin(lat);
}
```

# 6 6 其它

## 6.1 IO

### 6.1.1 fread

```
namespace io {
    const int L = 1 << 20 | 1;
    char ibuf[L], *iS, *iT, c, obuf[L], *oS = obuf, *oT = obuf + L - 1, qu[55]; int f, qr;
#ifdef whzzt
    #define gc() getchar()
#else
    #define gc() (iS == iT ? (iT = (iS = ibuf) + fread (ibuf, 1, L, stdin), iS == iT ? EOF : *iS ++) :
#endif
```

```
    template <class I>
    inline void gi (I &x) {
        for (f = 1, c = gc(); c < '0' || c > '9'; c = gc()) if (c == '-') f = -1;
        for (x = 0; c <= '9' && c >= '0'; c = gc()) x = x * 10 + (c & 15); x *= f;
    }
    inline void flush () {
        fwrite (obuf, 1, oS - obuf, stdout);
    }
    inline void putc (char x) {
        *oS ++ = x;
        if (oS == oT) flush (), oS = obuf;
    }
    template <class I>
    void print (I x) {
        if (!x) putc ('0'); if (x < 0) putc ('-'), x = -x;
        while (x) qu[++ qr] = x % 10 + '0', x /= 10;
        while (qr) putc (qu[qr --]);
    }
    struct io_ff { ~io_ff() { flush(); } } _io_ff_;
}
using io :: gi;
using io :: putc;
using io :: print;
```

### 6.1.2 fread2

```
namespace IO{
    #define BUF_SIZE 100000
    #define OUT_SIZE 100000
    #define ll long long
    //fread->read

    bool IOerror=0;
    inline char nc(){
        static char buf[BUF_SIZE],*p1=buf+BUF_SIZE,*pend=buf+BUF_SIZE;
        if (p1==pend){
            p1=buf; pend=buf+fread(buf,1,BUF_SIZE,stdin);
            if (pend==p1){IOerror=1;return -1;}
            //{printf("IO error!\n");system("pause");for (;;);exit(0);}
        }
        return *p1++;
    }
    inline bool blank(char ch){return ch==' '||ch=='\n'||ch=='\r'||ch=='\t';}
    inline void read(int &x){
        bool sign=0; char ch=nc(); x=0;
        for (;blank(ch);ch=nc());
        if (IOerror)return;
        if (ch=='-')sign=1,ch=nc();
        for (;ch>='0'&&ch<='9';ch=nc())x=x*10+ch-'0';
        if (sign)x=-x;
    }
    inline void read(ll &x){
        bool sign=0; char ch=nc(); x=0;
        for (;blank(ch);ch=nc());
```

```
    if (IOerror)return;
    if (ch=='-')sign=1,ch=nc();
    for (;ch>='0'&&ch<='9';ch=nc())x=x*10+ch-'0';
    if (sign)x=-x;
}
inline void read(double &x){
    bool sign=0; char ch=nc(); x=0;
    for (;blank(ch);ch=nc());
    if (IOerror)return;
    if (ch=='-')sign=1,ch=nc();
    for (;ch>='0'&&ch<='9';ch=nc())x=x*10+ch-'0';
    if (ch=='.'){
        double tmp=1; ch=nc();
        for (;ch>='0'&&ch<='9';ch=nc())tmp/=10.0,x+=tmp*(ch-'0');
    }
    if (sign)x=-x;
}
inline void read(char *s){
    char ch=nc();
    for (;blank(ch);ch=nc());
    if (IOerror)return;
    for (;!blank(ch)&&!IOerror;ch=nc())*s++=ch;
    *s=0;
}
inline void read(char &c){
    for (c=nc();blank(c);c=nc());
    if (IOerror){c=-1;return;}
}
//fwrite->write
struct Ostream_fwrite{
    char *buf,*p1,*pend;
    Ostream_fwrite(){buf=new char[BUF_SIZE];p1=buf;pend=buf+BUF_SIZE;}
    void out(char ch){
        if (p1==pend){
            fwrite(buf,1,BUF_SIZE,stdout);p1=buf;
        }
        *p1++=ch;
    }
    void print(int x){
        static char s[15],*s1;s1=s;
        if (!x)*s1++='0';if (x<0)out('-'),x=-x;
        while(x)*s1++=x%10+'0',x/=10;
        while(s1--!=s)out(*s1);
    }
    void println(int x){
        static char s[15],*s1;s1=s;
        if (!x)*s1++='0';if (x<0)out('-'),x=-x;
        while(x)*s1++=x%10+'0',x/=10;
        while(s1--!=s)out(*s1); out('\n');
    }
    void print(ll x){
        static char s[25],*s1;s1=s;
        if (!x)*s1++='0';if (x<0)out('-'),x=-x;
        while(x)*s1++=x%10+'0',x/=10;
```

```
            while(s1--!=s)out(*s1);
        }
        void println(ll x){
            static char s[25],*s1;s1=s;
            if (!x)*s1++='0';if (x<0)out('-'),x=-x;
            while(x)*s1++=x%10+'0',x/=10;
            while(s1--!=s)out(*s1); out('\n');
        }
        void print(double x,int y){
            static ll mul[]={1,10,100,1000,10000,100000,1000000,10000000,100000000,
                1000000000,10000000000LL,100000000000LL,1000000000000LL,10000000000000LL,
                100000000000000LL,1000000000000000LL,10000000000000000LL,100000000000000000LL};
            if (x<-1e-12)out('-'),x=-x;x*=mul[y];
            ll x1=(ll)floor(x); if (x-floor(x)>=0.5)++x1;
            ll x2=x1/mul[y],x3=x1-x2*mul[y]; print(x2);
            if (y>0){out('.'); for (size_t i=1;i<y&&x3*mul[i]<mul[y];out('0'),++i); print(x3);}
        }
        void println(double x,int y){print(x,y);out('\n');}
        void print(char *s){while (*s)out(*s++);}
        void println(char *s){while (*s)out(*s++);out('\n');}
        void flush(){if (p1!=buf){fwrite(buf,1,p1-buf,stdout);p1=buf;}}
        ~Ostream_fwrite(){flush();}
    }Ostream;
    inline void print(int x){Ostream.print(x);}
    inline void println(int x){Ostream.println(x);}
    inline void print(char x){Ostream.out(x);}
    inline void println(char x){Ostream.out(x);Ostream.out('\n');}
    inline void print(ll x){Ostream.print(x);}
    inline void println(ll x){Ostream.println(x);}
    inline void print(double x,int y){Ostream.print(x,y);}
    inline void println(double x,int y){Ostream.println(x,y);}
    inline void print(char *s){Ostream.print(s);}
    inline void println(char *s){Ostream.println(s);}
    inline void println(){Ostream.out('\n');}
    inline void flush(){Ostream.flush();}
    #undef ll
    #undef OUT_SIZE
    #undef BUF_SIZE
};
```

### 6.1.3 保留小数

```
#include <bits/stdc++.h>
using namespace std;
const double pi = acos(-1.0);
int main(void)
{
    for(int i = 0;i < 5; ++i)
    printf("%.*f\n",i,pi);
    for(int i = 0;i < 5; ++i)
        cout<<setiosflags(ios::fixed)<<setprecision(i)<<pi<<endl;
    return 0;
}
```

### 6.1.4 读取整数

```
//读取正负整数
inline int input(void)
{
    int num = 0;
    char c;
    int flag = 0;
    while((c = getchar()) < '0' || c > '9') flag = c=='-' ? 1:flag;
    while(c  >= '0' && c <= '9')
        num = num * 10 + c - '0',c = getchar();
    if(flag) num = -num;
    return num;
}
```

## 6.2 c++ 中处理 2 进制的一些函数.cpp

Built-in Function: int __builtin_ffs (unsigned int x)

Returns one plus the index of the least significant 1-bit of x, or if x is zero, returns zero.
返回右起第一个‘1’的位置。

Built-in Function: int __builtin_clz (unsigned int x)

Returns the number of leading 0-bits in x, starting at the most significant bit position. If x is 0, th
返回左起第一个‘1’之前0的个数。

Built-in Function: int __builtin_ctz (unsigned int x)

Returns the number of trailing 0-bits in x, starting at the least significant bit position. If x is 0,
返回右起第一个‘1’之后的0的个数。

Built-in Function: int __builtin_popcount (unsigned int x)

Returns the number of 1-bits in x.
返回‘1’的个数。

Built-in Function: int __builtin_parity (unsigned int x)

Returns the parity of x, i.e. the number of 1-bits in x modulo 2.
返回‘1’的个数的奇偶性。

Built-in Function: int __builtin_ffsl (unsigned long)

Similar to __builtin_ffs, except the argument type is unsigned long.

Built-in Function: int __builtin_clzl (unsigned long)

Similar to __builtin_clz, except the argument type is unsigned long.

Built-in Function: int __builtin_ctzl (unsigned long)

Similar to __builtin_ctz, except the argument type is unsigned long.

Built-in Function: int __builtin_popcountl (unsigned long)

Similar to __builtin_popcount, except the argument type is unsigned long.

Built-in Function: int __builtin_parityl (unsigned long)

Similar to __builtin_parity, except the argument type is unsigned long.

Built-in Function: int __builtin_ffsll (unsigned long long)

Similar to __builtin_ffs, except the argument type is unsigned long long.

Built-in Function: int __builtin_clzll (unsigned long long)

Similar to __builtin_clz, except the argument type is unsigned long long.

Built-in Function: int __builtin_ctzll (unsigned long long)

Similar to __builtin_ctz, except the argument type is unsigned long long.

Built-in Function: int __builtin_popcountll (unsigned long long)

Similar to __builtin_popcount, except the argument type is unsigned long long.

Built-in Function: int __builtin_parityll (unsigned long long)

Similar to __builtin_parity, except the argument type is unsigned long long.

## 6.3 测量程序的运行时间.cpp