# 数学

## 素数筛法

//ural 1086 cryptography

#include <cstdio>

#include <cstring>

#include <iostream>

using namespace std;

const int MAXN = 2000001;

int prime[MAXN], phi[MAXN];

bool isprime[MAXN];

//求出1..n的素数以及欧拉函数

int seive(int n)

{

int p = 0;

memset(isprime, true, sizeof(isprime));

isprime[0] = isprime[1] = false;

for (int i = 2; i <= n; ++ i)

{

if (isprime[i])

{

phi[i] = i - 1;

prime[p ++] = i;

}

for (int j = 0; j < p && i \* prime[j] <= n; ++ j)

{

isprime[i \* prime[j]] = false;

if (i % prime[j] == 0)

{

phi[i \* prime[j]] = phi[i] \* prime[j];

break;

}

else

phi[i \* prime[j]] = phi[i] \* (prime[j] - 1);

}

}

return p;

}

int main()

{

seive(MAXN);

int n;

scanf("%d", &n);

for (int i = 0; i < n; ++ i)

{

int x;

scanf("%d", &x);

printf("%d\n", prime[x - 1]);

}

return 0;

}

## 扩展欧几里得

/\*

如果gcd(a,b)=d,则存在x,y使ax + by = d

\*/

long long extend\_gcd(long long a, long long b)

{

if (b == 0)

{

x = 1; y = 0;

return a;

}

long long gcd = extend\_gcd(b, a % b);

long long t = x;

x = y; y = t - a / b \* y;

return gcd;

}

## 其他数论知识

### 欧拉函数

int phi(int n)

{

int ret = n;

for (int i = 2; (long long)i \* i <= n; ++ i)

if (n % i == 0)

{

while (n % i == 0)

n /= i;

ret -= ret / i;

}

if (n != 1)

ret -= ret / n;

return ret;

}

### SigmaGCD

//枚举gcd

long long SigmaGcd(int n)

{

long long sum = 0;

for (int i = 1; (long long)i \* i <= n; ++ i)

if (n % i == 0)

{

sum += i \* phi(n / i);

if (i \* i != n)

sum += n / i \* phi(i);

}

return sum;

}

# 图论

## 最短路SPFA（加叛负环）

bool spfa()

{

for (int i = 1; i <= n; i ++)

{

dis[i] = INF;

cnt[i] = 0;

inq[i] = false;

}

queue<int> q;

q.push(1);

inq[1] = false;

dis[1] = 0;

cnt[1] ++;

while (!q.empty())

{

int u = q.front();

for (int p = now[u]; p; p = pre[p])

{

int v = son[p];

if (dis[v] > dis[u] + val[p])

{

dis[v] = dis[u] + val[p];

if (!inq[v])

{

cnt[v] ++;

inq[v] = true;

if (cnt[v] > n) return false;

q.push(v);

}

}

}

inq[u] = false;

q.pop();

}

return true;

}

## K短路

//poj 2449

#include <iostream>

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <queue>

#include <vector>

#include <algorithm>

using namespace std;

const int MAXN = 1010;

const int MAXM = 100010;

const int INF = 1684300900;

struct Edge;

struct Node;

bool inq[MAXN];

int first[MAXN], d[MAXN], cnt[MAXN];

int x[MAXM], y[MAXM], dis[MAXM];

vector<Edge> eds;

struct Edge

{

int y, dis, next;

};

struct Node

{

int x, dis;

bool operator < (const Node& rhf) const

{

return dis + d[x] > rhf.dis + d[rhf.x];

}

};

void add\_edge(int x, int y, int d)

{

eds.push\_back(Edge { y, d, first[x]});

first[x] = eds.size() - 1;

}

void spfa(int S)

{

memset(d, 100, sizeof(d));

memset(inq, 0, sizeof(inq));

queue<int> q;

q.push(S); d[S] = 0; inq[S] = true;

while (!q.empty())

{

int x = q.front(); q.pop(); inq[x] = false;

for (int p = first[x]; p != -1; p = eds[p].next)

{

int y = eds[p].y;

if (d[y] > d[x] + eds[p].dis)

{

d[y] = d[x] + eds[p].dis;

if (!inq[y])

{

inq[y] = true;

q.push(y);

}

}

}

}

}

int a\_star(int S, int T, int K)

{

if (d[S] == INF) return -1;

memset(cnt, 0, sizeof(cnt));

priority\_queue<Node> heap;

heap.push(Node { S, 0});

while (!heap.empty())

{

Node t = heap.top(); heap.pop();

cnt[t.x] ++;

if (cnt[t.x] > K) continue;

if (t.x == T && cnt[t.x] == K)

return t.dis;

for (int p = first[t.x]; p != -1; p = eds[p].next)

{

int y = eds[p].y;

heap.push(Node {y, t.dis + eds[p].dis});

}

}

return -1;

}

int main()

{

int n, m, S, T, K;

scanf("%d%d", &n, &m);

eds.clear();

memset(first, 255, sizeof(first));

for (int i = 0; i < m; ++ i)

{

scanf("%d%d%d", &x[i], &y[i], &dis[i]);

add\_edge(y[i], x[i], dis[i]);

}

scanf("%d%d%d", &S, &T, &K);

spfa(T);

eds.clear();

memset(first, 255, sizeof(first));

for (int i = 0; i < m; ++ i)

add\_edge(x[i], y[i], dis[i]);

if (S == T) K ++;

printf("%d\n", a\_star(S, T, K));

return 0;

}

# 数据结构

# 字符串

# 计算几何

# 动态规划和贪心

# 其他

## 高精度