Contents

1 Math

1.1 FindPrime

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

//查找[0,2^15]中的所有質數 共有3515

const int MAXN = 32768;  //2^15=32768
bool primes[MAXN];
vector<int> p;  //3515

//質數篩法Sieve of Eratosthenes
inline void findPrimes() {
    for (int i = 0; i < MAXN; i++) {
        primes[i] = true;
    }
    primes[0] = false;
    primes[1] = false;
    for (int i = 4; i < MAXN; i += 2) {
        //將2的倍數全部刪掉(偶數不會是質數)
        primes[i] = false;
    }
    //開始逐個檢查--->小心i*i會有overflow問題--->使用long long
    for (long long i = 3; i < MAXN; i += 2) {
        if (primes[i]) {
            //如果之前還未被刪掉 才做篩法
            for (long long j = i * i; j < MAXN; j += i) {
                //從i*i開始(因為i*2,i*3...都被前面處理完了
                primes[j] = false;
            }
        }
    }
    //蒐集所有質數
    for (int i = 0; i < MAXN; i++) {
        if (primes[i]) {
            p.emplace_back(i);
        }
    }
}
```

2 Graph

2.1 Kruskal

```cpp
#include <bits/stdc++.h>
using namespace std;
// Kruskal (MST) 節點從0號開始
struct Edge {
    int v, w, wt;
    Edge(int a, int b, int c) {
        v = a;
        w = b;
        wt = c;
    }
    bool operator<(const Edge &e) const {
        return wt < e.wt;
    }
};

const int maxN = 100000 + 5;  // maxN個節點
int parent[maxN];
vector<Edge> edges;

int do_find(int p) {
    while (parent[p] >= 0) {
        p = parent[p];
    }
    return p;
}

void do_union(int p, int q) {
    if (parent[p] > parent[q]) {
        parent[q] += parent[p];
        parent[p] = q;
    } else {
        parent[p] += parent[q];
        parent[q] = p;
    }
}

int m, n, ta, tb, tc, weight;

int main() {
    while (~scanf("%d %d", &m, &n)) {
        for (int i = 0; i < n; i++) {
            scanf("%d %d %d", &ta, &tb, &tc);
            edges.push_back({ta, tb, tc});
        }
        sort(edges.begin(), edges.end());
        for (int i = 0; i <= m; i++) {
            parent[i] = -1;
        }
        weight = 0;
        for (auto e : edges) {
            ta = do_find(e.v);
            tb = do_find(e.w);
            if (ta != tb) {
                weight += e.wt;
                do_union(ta, tb);
            }
        }
        printf("%d\n", weight);
    }
    return 0;
}
```