```c
// Kruskal Tim cay khung nho nhat
#include <stdio.h>
#define MAXN 1000

// Graph
typedef struct {
    int u, v;
    int w;
} Edge;
typedef struct {
  int n, m;
  Edge edges[MAXN];
} Graph;

void init_graph(Graph* G, int n) {
  G->n = n;
  G->m = 0;
}

void add_edge(Graph* G, int u, int v, int w) {
    G->edges[G->m].u = u;
    G->edges[G->m].v = v;
    G->edges[G->m].w = w;
    ++G->m;
}

//Kruskal

int parent[MAXN];

int findRoot(int u) {
  if (parent[u] == u) {
    return u;
  }
  return findRoot(parent[u]);
}

int Kruskal (Graph* G, Graph* T) {
  //Sap xep cac cung cua G theo thu tu trong so tang dan
  int i, j;
  for (i = 0; i < G->m; ++i) {
    for(j = i + 1; j < G->m; ++j) {
      if (G->edges[i].w > G->edges[j].w) {
```

```
        Edge tmp = G->edges[i];
        G->edges[i] = G->edges[j];
        G->edges[j] = tmp;
      }
    }
  }

  // Khoi tao T rong
  init_graph(T, G->n);
  int u;
  for (u = 1; u <= G->n; ++u) {
    parent[u] = u; // Moi dinh u la mot bo phan lien thong
  }

  int sum_w = 0;

  // Duye qua cac cung cua G (da sap xep)
  int e;
  for (e = 0; e < G->m; ++e) {
    int u = G->edges[e].u;
    int v = G->edges[e].v;
    int w = G->edges[e].w;
    int root_u = findRoot(u);
    int root_v = findRoot(v);

    if (root_u != root_v) {
      add_edge(T, u, v, w);
      // Gop 2 BPLT root_u va root v lai
      parent[root_v] = root_u;
      sum_w += w;
    }
  }

  return sum_w;

}

int main() {
  Graph G, T;
  int n, m, u, v, w, e;

  scanf("%d%d", &n, &m);
  init_graph(&G, n);
```

```
  for (e = 0; e < m; ++e) {
    scanf("%d%d%d", &u, &v, &w);

    if (u > v) {
       int tmp = u;
       u = v;
       v = tmp;
    }
    add_edge(&G, u, v, w);
  }

  int sum_w = Kruskal(&G, &T);

  printf("%d\n", sum_w);

  for (e = 0; e < T.m; ++e) {
      printf("%d %d %d \n", T.edges[e].u, T.edges[e].v, T.edges[e].w);
  }
  return 0;
}
```