```c
// Prim tim cay khung nho nhat (Phien ban 2)
#include <stdio.h>
#define MAXN 1000
#define NO_EDGE 0
#define INFINITY 9999999

// Graph
typedef struct {
  int n;
  int L[MAXN][MAXN];
} Graph;

void init_graph(Graph* G, int n) {
  G->n = n;
  int i, j;
  for (i = 1; i <= n; ++i) {
    for (j = 1; j <= n; ++j) {
      G->L[i][j] =NO_EDGE;
    }
  }
}

void add_edge(Graph* G, int x, int y, int w) {
    G->L[x][y] = w;
    G->L[y][x] = w;
}

int mark[MAXN];
int pi[MAXN];
int p[MAXN];

int Prime(Graph* G, Graph* T) {
    init_graph(T, G->n); // Khoi tao cay T rong

  int i, u, v, it;
  for (i = 1; i <= G->n; ++i) {
    pi[i] = INFINITY;
    mark[i] = 0;
  }

  mark[1] = 1;
  pi[1] = 0;
```

```
  for (v = 1; v <= G->n; ++v) {
      if (G->L[1][v] != NO_EDGE) {
          pi[v] = G->L[1][v]; // Gan pi[v] = trong so cung (1, v)
          p[v] = 1; // Dinh s gan voi v la dinh 1
      }
  }

  int sum_w = 0; // Tong trong so cua cay T
  for (it = 1; it < G->n; ++it) {
      //1. Tim u gan voi S nhat (tim u co pi[u] nho nhat)
    int min_dist = INFINITY, min_u;
    for (u = 1; u <= G->n; ++u) {
      if (mark[u] == 0 && pi[u] < min_dist) {
        min_dist = pi[u];
        min_u = u;
      }
    }

    u = min_u; // Dinh u co pi[u] nho nhat
    //2. Danh dau min_u
    mark[min_u] = 1;

    //3. Dua cung (min_u, p[min_u], min_dist) vao T
    add_edge(T, min_u, p[min_u], min_dist);
    sum_w += min_dist;

    //4. Cap nhat lai pi va p cua cac dinh ke voi u
    for (v = 1; v <= G->n; ++v) {
      if (G->L[u][v] != NO_EDGE && mark[v] == 0) {
        if (G->L[u][v] < pi[v]) {
          pi[v] = G->L[u][v];
          p[v] = u;
        }
      }
    }
  }
  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);
    add_edge(&G, u, v, w);
  }

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

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

  for (u = 1; u <= T.n; ++u) {
      for (v = u + 1; v <= T.n; ++v) {
          if (T.L[u][v]) {
              printf("%d %d %d\n", u, v, T.L[u][v]);
          }
      }
  }
  return 0;
}
```