```c
// Tim luong cuc dai tren mang (FordFullKerson)

#include <stdio.h>

#define MAXN 500
#define NO_EDGE 0
#define INFINITY 999

// min function
int min(int a, int b) {
  return (a < b) ? a : b;
}

// Queue
typedef struct {
  int data[100];
  int front, rear;
} Queue;

void make_null_queue(Queue* Q) {
  Q->front = -1;
  Q->rear = -1;
}

void push(Queue* Q, int x) {
  ++Q->rear;
  if (Q->front == -1 ) {
     ++Q->front;
  }
  Q->data[Q->rear] = x;
}

int top(Queue* Q) {
  return Q->data[Q->front];
}

void pop(Queue* Q) {
  ++Q->front;
}

int empty(Queue* Q) {
  return Q->front > Q->rear;
}
```

```
// Graph
typedef struct {
  int C[MAXN][MAXN]; // Kha nang thong qua cua cung
  int F[MAXN][MAXN]; // Luong tren cung
  int n;
} 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->C[i][j] = NO_EDGE;
    }
  }
}

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

// Label
typedef struct {
  int dir; // >0: +, <0: -, 0: chua co nhan
  int pre; // dinh truoc
  int sigma; // Luong tang Luon
} Label;
Label labels[MAXN];

void init_flow(Graph* G) {
  int u, v;

  for (u = 1; u <= G->n; ++u) {
    for (v = 1; v <= G->n; ++v) {
        G->F[u][v] = 0;
    }
  }
}

int FordFullkerson(Graph* G, int s, int t) {
    //I. Khoi tao luong = 0, gan F[u][v] = 0 voi moi u, voi
    init_flow(G);
```

```
int sum_flow = 0;
Queue Q;
do {
    // Buoc 1 - xoa nhan cac dinh va gan nhan cho s
    // 1.1 Xoa tat ca cac nhan
    int u, v;
    for (u = 1; u <= G->n; ++u) {
        labels[u].dir = 0;
    }

    // 1.2 Gan nhan s: (+, s, oo)
    labels[s].dir = 1;
    labels[s].pre = s;
    labels[s].sigma = INFINITY;

    // 1.3 Khoi tao Q rong, dua s vao Queue
    make_null_queue(&Q);
    push(&Q, s);

    // Buoc 2, 3 - Lap gan nhan cho cac dinh
    int found = 0;
    while (!empty(&Q)) {
        // Lay 1 dinh trong Q ra => u
        int u = top(&Q);
        pop(&Q);
        for (v = 1; v <= G->n; ++v) {
            // Xet gan nhan cho cac dinh ke voi x, cung thuan
            if (labels[v].dir == 0 && G->C[u][v] != NO_EDGE &&
            G->F[u][v] < G->C[u][v]) {
                labels[v].dir = +1; // Cung thuan
                labels[v].pre = u;
                labels[v].sigma = min(labels[u].sigma,
                G->C[u][v] - G->F[u][v]);
                push(&Q, v); //printf("v = %d", v);
            }

            // Xet gan nhan cho cac dinh ke voi x, cung nghich
            if (labels[v].dir == 0 && G->C[v][u] != NO_EDGE &&
            G->F[v][u] > 0) {
                labels[v].dir = -1; // Cung nghich
                labels[v].pre = u;
                labels[v].sigma =
                min(labels[u].sigma, G->F[v][u]);
```

```
                    push(&Q, v);
                }
            }

            // Neu t duoc ga nhan => tim duoc duong tang luong, thoat
vong Lap
            if (labels[t].dir != 0) {
                found = 1;
                break;
            }
        }

        if (found == 1) {
            // Buoc 4, 5, 6 - tang luong
            int x = t;
            int sigma = labels[t].sigma;
            sum_flow += sigma; // luong tang them
            while (x != s) {
                int u = labels[x].pre;
                if (labels[x].dir > 0) { // tang luong
                    G->F[u][x] += sigma;
                } else { // giam luong
                    G->F[x][u] -= sigma;
                }
                x = u;
            }
        } else {
            break;
        }
    } while(1);

    return sum_flow;
}

int main() {
  Graph G;
  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 max_flow = FordFullkerson(&G, 1, n);

    printf("Max flow: %d\n", max_flow);

    printf("X0:");
    for (u = 1; u <= n; ++u) {
        if (labels[u].dir != 0) {
            printf(" %d", u);
        }
    }

    printf("\nY0:");
    for (u = 1; u <= n; ++u) {
        if (labels[u].dir == 0) {
            printf(" %d", u);
        }
    }

    return 0;
}
```