**THỰC HÀNH BUỔI 5**

1. **BT5.1. Tìm cây khung bằng giải thuật Kruskal**

|  |
| --- |
| #include<stdio.h>  #define MAX 9999  typedef struct{  int u, v, w;  }Edge;  typedef struct{  Edge edges[MAX];  int n, m;  }Graph;  void init(Graph \*G, int n){  G->n = n;  G->m = 0;  }  void add(Graph \*G, int u, int v, int w){  G->edges[G->m].u = (u < v ? u : v);  G->edges[G->m].v = (u > v ? u : v);  G->edges[G->m].w = w;  G->m++;  }  int parent[MAX];  int findroot(int u){  while(parent[u] != u)  u = parent[u];  return u;  }  int Kruskal(Graph \*G, Graph \*T){  int u, v, w;  for(int i = 0; i < G->m; i++)  for(int j = i + 1; j < G->m; j++)  if(G->edges[i].w > G->edges[j].w){  Edge temp = G->edges[i];  G->edges[i] = G->edges[j];  G->edges[j] = temp;  }  init(T, G->n);  for(u = 1; u <= G->n; u++)  parent[u] = u;  int sum\_w = 0;  for(int e = 0; e < G->m; e++){  u = G->edges[e].u;  v = G->edges[e].v;  w = G->edges[e].w;  int root\_u = findroot(u);  int root\_v = findroot(v);  if(root\_u != root\_v){  add(T, u, v, w);  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(&G, n);  for(e = 0; e < m; e++){  scanf("%d%d%d",&u,&v,&w);  add(&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);  }  } |

1. **BT5.2. Tìm cây khung có trọng lượng nhỏ nhất bằng giải thuật Prim**

|  |
| --- |
| #include<stdio.h>  #define MAX 50  #define oo 9999  #define NO\_EDGE 0  typedef struct{  int A[MAX][MAX], n, m;  }Graph;  void init(Graph \*G, int n){  G->n = n;  G->m = 0;  for(int u = 1; u <= G->n; u++){  for(int v = 1; v <= G->n; v++){  G->A[u][v] = NO\_EDGE;  }  }  }  void add(Graph \*G, int u, int v, int w){  G->A[u][v] += w;  G->A[v][u] += w;  G->m++;  }  int pi[MAX];  int p[MAX];  int mark[MAX];  int Prim(Graph \*G, Graph \*T, int s){  int u, v, i, x;  for(u = 1; u <= G->n; u++){  pi[u] = oo;  p[u] = -1;  mark[u] = 0;  }  pi[s] = 0;  for(i = 1; i < G->n; i++){  int min\_dist = oo;  for(x = 1; x <= G->n; x++){  if(!mark[x] && pi[x] < min\_dist){  min\_dist = pi[x];  u = x;  }  }  mark[u] = 1;  for(v = 1; v <= G->n; v++){  if(!mark[v] && G->A[u][v] != NO\_EDGE && pi[v] > G->A[u][v]){  pi[v] = G->A[u][v];  p[v] = u;  }  }  }  init(T, G->n);  int sum\_w = 0;  for(u = 1; u <= G->n; u++){  if(p[u] != -1){  int w = G->A[p[u]][u];  add(T, p[u], u, w);  sum\_w += w;  }  }  return sum\_w;  }  int main(){  Graph G, T;  int n, m, u, v, w, e;  scanf("%d%d",&n, &m);  init(&G, n);  for(e = 0; e < m; e++){  scanf("%d%d%d", &u, &v, &w);  add(&G, u, v, w);  }  init(&T, n);  int sum\_w = 0;  sum\_w = Prim(&G, &T, 1);  printf("%d\n", sum\_w);  for(u = 1; u <= T.n; u++){  for(v = 1; v <= T.n; v++){  if(T.A[u][v] != NO\_EDGE && u < v)  printf("%d %d %d\n", u, v, T.A[u][v]);  }  }  } |

1. **BT5.3. Ứng dụng cây khung**

|  |
| --- |
| Q1.  #include<stdio.h>  #define MAX 9999  typedef struct{  int u, v, w;  }Edge;  typedef struct{  Edge edge[MAX];  int n, m;  }Graph;  void init(Graph \*G, int n){  G->n = n;  G->m = 0;  }  void add(Graph \*G, int u, int v, int w){  G->edge[G->m].u = u;  G->edge[G->m].v = v;  G->edge[G->m].w = w;  G->m++;  }  int parent[MAX];  int findroot(int u){  while(parent[u] != u)  u = parent[u];  return u;  }  int Kruskal(Graph \*G, Graph \*T){  int u, v, w;  for(int i = 0; i < G->m; i++)  for(int j = i + 1; j < G->m; j++)  if(G->edge[i].w > G->edge[j].w){  Edge temp = G->edge[i];  G->edge[i] = G->edge[j];  G->edge[j] = temp;  }  init(T, G->n);  for(u = 1; u <= G->n; u++)  parent[u] = u;  int sum\_w = 0;  for(int e = 0; e < G->m; e++){  u = G->edge[e].u;  v = G->edge[e].v;  w = G->edge[e].w;  int root\_u = findroot(u);  int root\_v = findroot(v);  if(root\_u != root\_v){  add(T, u, v, w);  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(&G, n);  for(e = 0; e < m; e++){  scanf("%d%d%d",&u,&v,&w);  add(&G, u, v, w);  }  int sum\_w\_chuaxoa = 0;  for(e = 0; e < m; e++){  sum\_w\_chuaxoa += G.edge[e].w;  }  int sum\_w = Kruskal(&G, &T);  printf("%d\n", sum\_w\_chuaxoa - sum\_w);    } |
| Q2.  #include<stdio.h>  #define MAX 50  #define oo 9999  #define NO\_EDGE 0  typedef struct{  int A[MAX][MAX], n, m;  }Graph;  void init(Graph \*G, int n){  G->n = n;  G->m = 0;  for(int u = 1; u <= G->n; u++){  for(int v = 1; v <= G->n; v++){  G->A[u][v] = NO\_EDGE;  }  }  }  void add(Graph \*G, int u, int v, int w){  G->A[u][v] += w;  G->A[v][u] += w;  G->m++;  }  int pi[MAX];  int p[MAX];  int mark[MAX];  void Prim(Graph \*G, Graph \*T, int s){  int u, v, i, x;  for(u = 1; u <= G->n; u++){  pi[u] = oo;  p[u] = -1;  mark[u] = 0;  }  pi[s] = 0;  for(i = 1; i <= G->n; i++){  int min\_dist = oo;  for(x = 1; x <= G->n; x++){  if(!mark[x] && pi[x] < min\_dist){  min\_dist = pi[x];  u = x;  }  }  mark[u] = 1;  printf("%d\n", u);  for(v = 1; v <= G->n; v++){  if(!mark[v] && G->A[u][v] != NO\_EDGE && pi[v] > G->A[u][v]){  pi[v] = G->A[u][v];  p[v] = u;  }  }  }  init(T, G->n);  int sum\_w = 0;  for(u = 1; u <= G->n; u++){  if(p[u] != -1){  int w = G->A[p[u]][u];  add(T, p[u], u, w);  sum\_w += w;  }  }  }  int main(){  Graph G, T;  int n, m, u, v, w, e;  scanf("%d%d",&n, &m);  init(&G, n);  for(e = 0; e < m; e++){  scanf("%d%d%d", &u, &v, &w);  add(&G, u, v, w);  }  init(&T, n);  Prim(&G, &T, 1);  } |
| Q3.  #include<stdio.h>  #define MAX 50  #define oo 9999  #define NO\_EDGE 0  typedef struct{  int A[MAX][MAX], n, m;  }Graph;  void init(Graph \*G, int n){  G->n = n;  G->m = 0;  for(int u = 1; u <= G->n; u++){  for(int v = 1; v <= G->n; v++){  G->A[u][v] = NO\_EDGE;  }  }  }  void add(Graph \*G, int u, int v, int w){  G->A[u][v] += w;  G->A[v][u] += w;  G->m++;  }  int pi[MAX];  int p[MAX];  int mark[MAX];  int Prim(Graph \*G, Graph \*T, int s){  int u, v, i, x;  for(u = 1; u <= G->n; u++){  pi[u] = oo;  p[u] = -1;  mark[u] = 0;  }  pi[s] = 0;  for(i = 1; i < G->n; i++){  int min\_dist = oo;  for(x = 1; x <= G->n; x++){  if(!mark[x] && pi[x] < min\_dist){  min\_dist = pi[x];  u = x;  }  }  mark[u] = 1;  for(v = 1; v <= G->n; v++){  if(!mark[v] && G->A[u][v] != NO\_EDGE && pi[v] > G->A[u][v]){  pi[v] = G->A[u][v];  p[v] = u;  }  }  }  init(T, G->n);  int sum\_w = 0;  for(u = 1; u <= G->n; u++){  if(p[u] != -1){  int w = G->A[p[u]][u];  add(T, p[u], u, w);  sum\_w += w;  }  }  return sum\_w;  }  int main(){  Graph G, T;  int n, m, u, v, w, e, k;  scanf("%d%d%d",&n, &m, &k);  init(&G, n);  for(e = 0; e < m; e++){  scanf("%d%d%d", &u, &v, &w);  add(&G, u, v, w);  }  init(&T, n);  int sum\_w = 0;  sum\_w = Prim(&G, &T, 1);  if(sum\_w <= k)  printf("OK");  else printf("%d",sum\_w - k);  } |
| Q4.  #include<stdio.h>  #define MAX 50  #define oo 9999  #define NO\_EDGE 0  typedef struct{  int A[MAX][MAX], n, m;  }Graph;  void init(Graph \*G, int n){  G->n = n;  G->m = 0;  for(int u = 1; u <= G->n; u++){  for(int v = 1; v <= G->n; v++){  G->A[u][v] = NO\_EDGE;  }  }  }  void add(Graph \*G, int u, int v, int w){  G->A[u][v] += w;  G->A[v][u] += w;  G->m++;  }  int pi[MAX];  int p[MAX];  int mark[MAX];  int Prim(Graph \*G, Graph \*T, int s){  int u, v, i, x;  for(u = 1; u <= G->n; u++){  pi[u] = oo;  p[u] = -1;  mark[u] = 0;  }  pi[s] = 0;  for(i = 1; i < G->n; i++){  int min\_dist = oo;  for(x = 1; x <= G->n; x++){  if(!mark[x] && pi[x] < min\_dist){  min\_dist = pi[x];  u = x;  }  }  mark[u] = 1;  for(v = 1; v <= G->n; v++){  if(!mark[v] && G->A[u][v] != NO\_EDGE && pi[v] > G->A[u][v]){  pi[v] = G->A[u][v];  p[v] = u;  }  }  }  init(T, G->n);  int sum\_w = 0;  for(u = 1; u <= G->n; u++){  if(p[u] != -1){  int w = G->A[p[u]][u];  add(T, p[u], u, w);  sum\_w += w;  }  }  return sum\_w;  }  int main(){  Graph G, T;  int n, m, u, v, w, e;  scanf("%d%d",&n, &m);  init(&G, n);  for(e = 0; e < m; e++){  scanf("%d%d%d", &u, &v, &w);  add(&G, u, v, w);  }  init(&T, n);  int sum\_w = 0;  sum\_w = Prim(&G, &T, 1);  printf("%d\n", sum\_w);  } |
| Q5.  #include<stdio.h>  #define MAX 50  #define oo 9999  typedef struct{  int u, v, w, d;  }Edge;  typedef struct{  Edge edge[MAX];  int n, m;  }Graph;  void init(Graph \*G, int n){  G->n = n;  G->m = 0;  }  void add(Graph \*G, int u, int v, int w, int d){  G->edge[G->m].u = u;  G->edge[G->m].v = v;  G->edge[G->m].w = w;  G->edge[G->m].d = d;  G->m++;  }  int parent[MAX];  int findroot(int u){  while(parent[u] != u)  u = parent[u];  return u;  }  void Kruskal(Graph \*G, Graph \*T){  int u, v, w, d;  for(int i = 0; i < G->m; i++)  for(int j = i + 1; j < G->m; j++)  if(G->edge[i].w \* G->edge[i].d > G->edge[j].w \* G->edge[j].d){  Edge temp = G->edge[i];  G->edge[i] = G->edge[j];  G->edge[j] = temp;  }  init(T, G->n);  for(u = 1; u <= G->n; u++)  parent[u] = u;  int sum\_w = 0;  for(int e = 0; e < G->m; e++){  u = G->edge[e].u;  v = G->edge[e].v;  w = G->edge[e].w;  d = G->edge[e].d;  int root\_u = findroot(u);  int root\_v = findroot(v);  if(root\_u != root\_v){  add(T, u, v, w, d);  parent[root\_v] = root\_u;  sum\_w += w;  }  }  }  int main(){  Graph G, T;  int n, m, u, v, w, d, e;  scanf("%d%d",&n,&m);  init(&G, n);  for(e = 0; e < m; e++){  scanf("%d%d%d%d",&u,&v,&w,&d);  add(&G, u, v, w, d);  }  Kruskal(&G, &T);  int cp = 0;  for(int i = 0; i < T.m; i++){  cp += T.edge[i].w \* T.edge[i].d;  }  printf("%d", cp);  } |
| Q6.  #include<stdio.h>  #define MAX 50  #define oo 9999  #define NO\_EDGE 0  typedef struct{  int A[MAX][MAX], n, m;  }Graph;  void init(Graph \*G, int n){  G->n = n;  G->m = 0;  for(int u = 1; u <= G->n; u++){  for(int v = 1; v <= G->n; v++){  G->A[u][v] = NO\_EDGE;  }  }  }  void add(Graph \*G, int u, int v, int w){  G->A[u][v] += w;  G->A[v][u] += w;  G->m++;  }  int pi[MAX];  int p[MAX];  int mark[MAX];  int Prim(Graph \*G, Graph \*T, int s){  int u, v, i, x;  for(u = 1; u <= G->n; u++){  pi[u] = oo;  p[u] = -1;  mark[u] = 0;  }  pi[s] = 0;  for(i = 1; i < G->n; i++){  int min\_dist = oo;  for(x = 1; x <= G->n; x++){  if(!mark[x] && pi[x] < min\_dist){  min\_dist = pi[x];  u = x;  }  }  mark[u] = 1;  for(v = 1; v <= G->n; v++){  if(!mark[v] && G->A[u][v] != NO\_EDGE && pi[v] > G->A[u][v]){  pi[v] = G->A[u][v];  p[v] = u;  }  }  }  init(T, G->n);  int sum\_w = 0;  for(u = 1; u <= G->n; u++){  if(p[u] != -1){  int w = G->A[p[u]][u];  add(T, p[u], u, w);  sum\_w += w;  }  }  return sum\_w;  }  int main(){  Graph G, T;  int n, m, u, v, w, e;  scanf("%d%d",&n, &m);  init(&G, n);  for(e = 0; e < m; e++){  scanf("%d%d%d", &u, &v, &w);  add(&G, u, v, w);  }  init(&T, n);  int sum\_w = 0;  sum\_w = Prim(&G, &T, 1);  printf("%d\n", sum\_w);  } |

1. **Thuật toán tìm cây khung từ đồ thị có hướng Chuliu - Edmonds (k có trong giáo trình)**

=> Lấy test case trong bt lý thuyết r chạy code so đáp án.

|  |
| --- |
| #include<stdio.h>  #define MAXN 100  #define MAXM 500  #define MAXIT 10  #define oo 99999  typedef struct{  int u, v, w, link;  }Edge;  typedef struct{  int n, m;  Edge edge[MAXM];  }Graph;  typedef struct{  int n;  int parent[MAXN];  int weight[MAXN];  int link[MAXN];  }Tree;  void init\_graph(Graph \*G, int n){  G->n = n;  G->m = 0;  }  void init\_tree(Tree \*T, int n){  T->n = n;  int i;  for(i = 1; i <= n; i++){  T->parent[i] = -1;  T->weight[i] = oo;  T->link[i] = -1;  }  }  void add\_edge(Graph \*G, int u, int v, int w, int link){  G->edge[G->m].u = u;  G->edge[G->m].v = v;  G->edge[G->m].w = w;  G->edge[G->m].link = link;  G->m++;  }  void buildH(Graph \*G, int root, Tree \*H){  init\_tree(H, G->n);  int e;  for(e = 0; e < G->m; e++){  int u = G->edge[e].u;  int v = G->edge[e].v;  int w = G->edge[e].w;  int link = G->edge[e].link;  if(w < H->weight[v]){  H->parent[v] = u;  H->weight[v] = w;  H->link[v] = link;  }  }  H->parent[root] = -1;  H->weight[root] = 0;  }  int id[MAXN];  int color[MAXN];  int find\_cycles(Tree \*H, int root){  int i, u, no = 0;  for(i = 1; i <= H->n; i++){  id[i] = -1;  color[i] = -1;  }  for(i = 1; i <= H->n; i++){  int u = i;  while(u != root && id[u] == -1 && color[u] != i){  color[u] = i;  u = H->parent[u];  }  if(color[u] == i){  no++;  int v = H->parent[u];  while(v != u){  id[v] = no;  v = H->parent[v];  }  id[u] = no;  }  }  return no;  }  void contract(Graph \*G, Tree \*H, int no, Graph \*G1){  init\_graph(G1, no);  int e;  for(e = 0; e < G->m; e++){  int u = G->edge[e].u;  int v = G->edge[e].v;  int w = G->edge[e].w;  if(id[u] != id[v])  add\_edge(G1, id[u], id[v], w - H->weight[v], e);  }  }  void expand(Tree \*H, Graph \*G1, Tree \*H1){  int i;  for(i = 1; i <= H->n; i++)  if(H->parent[i] != -1){  Edge pe = G1->edge[H->link[i]];  H1->parent[pe.v] = pe.u;  H1->weight[pe.v] += H->weight[i];  H1->link[pe.v] = pe.link;  }  }  void ChuLiu(Graph \*G0, int s, Tree \*T){  Graph G[MAXIT];  Tree H[MAXIT];  int i, e;  int t = 0;  int root = s;  G[0] = \*G0;  while(1){  buildH(&G[t], root, &H[t]);  int no = find\_cycles(&H[t], root);  if(no == 0) break;  for(i = 1; i <= H[t].n; i++){  if(id[i] == -1)  id[i] = ++no;  }  contract(&G[t], &H[t], no, &G[t+1]);  root = id[root];  t++;  }  int k;  for(k = t; k > 0; k--)  expand(&H[k], &G[k-1], &H[k-1]);  \*T = H[0];  }  int main(){  Graph G;  int n, m, i, e, u, v, w;  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, -1);  }  printf("\n");  Tree T;  ChuLiu(&G, 1, &T);  for(i = 1; i <= T.n; i++)  if(T.parent[i] != -1)  printf("%d -> %d: %d\n", T.parent[i], i, T.weight[i]);  return 0; //kkk  }  Link drive: https://drive.google.com/file/d/1HRXyA3R4thDvqOYgqu3qq9sTX5dzoPTt/view |

1. **BT5.4. Tìm luồng cực đại trong mạng**

|  |
| --- |
| #include<stdio.h>  #define MAX 50  #define NO\_EDGE 0  #define oo 9999  typedef struct{  int dir, p, sigma;  }Label;  Label labels[MAX];  typedef struct{  int C[MAX][MAX], F[MAX][MAX], n, m;  }Graph;  void init\_graph(Graph \*G, int n){  G->n = n;  G->m = 0;  }  void init\_flow(Graph \*G){  for(int u = 1; u <= G->n; u++)  for(int v = 1; v <= G->n; v++)  G->F[u][v] = NO\_EDGE;  }  typedef struct{  int hangdoi[MAX], front, rear;  }Queue;  void make\_null\_Queue(Queue \*Q){  Q->front = 0;  Q->rear = -1;  }  void enQueue(Queue \*Q, int u){  Q->hangdoi[++Q->rear] = u;  }  int front(Queue \*Q){  return Q->hangdoi[Q->front];  }  void deQueue(Queue \*Q){  Q->front++;  }  int emptyQueue(Queue \*Q){  return Q->front > Q->rear;  }  int min(int a, int b){  return a < b ? a : b;  }  int FordFulkerson(Graph \*G, int s, int t){  init\_flow(G);  int u, v;  int max\_flow = 0;  Queue Q;  do{  for(u = 1; u <= G->n; u++)  labels[u].dir = 0;  labels[s].dir = 0;  labels[s].p = s;  labels[s].sigma = oo;  make\_null\_Queue(&Q);  enQueue(&Q, s);  int found = 0;  while(!emptyQueue(&Q)){  int u = front(&Q); deQueue(&Q);  for(v = 1; v <= G->n; v++){  if(G->C[u][v] != NO\_EDGE && labels[v].dir == 0 && G->F[u][v] < G->C[u][v]){  labels[v].dir = +1;  labels[v].p = u;  labels[v].sigma = min(labels[u].sigma, G->C[u][v] - G->F[u][v]);  enQueue(&Q, v);  }  }  for(int x = 1; x <= G->n; x++){  if(G->C[x][u] != NO\_EDGE && labels[x].dir == 0 && G->F[x][u] > 0){  labels[x].dir = -1;  labels[x].p = u;  labels[x].sigma = min(labels[u].sigma, G->F[x][u]);  enQueue(&Q, x);  }  }  if(labels[t].dir != 0){  found = 1;  break;  }  }  if(found == 1){  int sigma = labels[t].sigma;  u = t;  while(u != s){  int p = labels[u].p;  if(labels[u].dir > 0)  G->F[p][u] += sigma;  else G->F[u][p] -= sigma;  u = p;  }  max\_flow += sigma;  }  else break;  }while(1);  return max\_flow;  }  int main(){  Graph G;  int n, m, u, v, e, c;  scanf("%d%d",&n,&m);  init\_graph(&G, n);  for(e = 0; e < m; e++){  scanf("%d%d%d",&u,&v,&c);  G.C[u][v] = c;  }  int max\_flow = FordFulkerson(&G, 1, n);  printf("Max flow: %d\n", max\_flow);  printf("S: ");  for(u = 1; u <= n; u++)  if(labels[u].dir != 0)  printf("%d ", u);  printf("\n");  printf("T: ");  for(u = 1; u <= n; u++)  if(labels[u].dir == 0)  printf("%d ", u);  return 0;  } |

1. **BT5.5. Kiểm tra luồng hợp lệ:**

|  |
| --- |
| #include<stdio.h>  #define MAX 50  #define NO\_EDGE 0  #define oo 9999  typedef struct{  int dir, p, sigma;  }Label;  Label labels[MAX];  typedef struct{  int C[MAX][MAX], F[MAX][MAX], n, m;  }Graph;  void init\_graph(Graph \*G, int n){  G->n = n;  G->m = 0;  }  int kiemtraluong(Graph \*G, int n, int s, int t){  for (int u = 1; u <= G->n; u++) {  for (int v = 1; v <= G->n; v++) {  if (G->F[u][v] > G->C[u][v]) {  return 0;  }  }  }  int out\_s = 0, in\_t = 0;  for (int u = 1; u <= G->n; u++) {  out\_s += G->F[s][u];  in\_t += G->F[u][t];  }  if (out\_s != in\_t) {  return 0;  }  for (int u = 2; u <= n-1; u++) {  int in\_u = 0, out\_u = 0;  for (int v = 1; v <= n; v++) {  in\_u += G->F[v][u];  out\_u += G->F[u][v];  }  if (in\_u != out\_u) {  return 0;  }  }  return 1;  }  int main(){  Graph G;  int n, m, u, v, e, c, f;  scanf("%d%d",&n,&m);  init\_graph(&G, n);  for(e = 0; e < m; e++){  scanf("%d%d%d%d",&u,&v,&c,&f);  G.C[u][v] = c;  G.F[u][v] = f;  }  if(kiemtraluong(&G, n, 1, G.n))  printf("YES");  else printf("NO");  return 0;  } |