**THỰC HÀNH BUỔI 3**

1. **BT1 - Thuật toán Moore - Dijkstra (pi và p)**

|  |
| --- |
| Q1.  #include<stdio.h>  #define MAX\_N 50  #define oo 9999  #define NO\_EDGE -1  typedef struct{  int A[MAX\_N][MAX\_N], n, m;  }Graph;  void init\_graph(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\_edge(Graph \*G, int u, int v, int w){  G->A[u][v] = w;  G->m++;  }  int mark[MAX\_N], pi[MAX\_N], p[MAX\_N];  void MooreDijkstra(Graph \*G, int s){  int u, v, it;  for(u = 1; u <= G->n; u++){  pi[u] = oo;  mark[u] = 0;  }  pi[s] = 0;  p[s] = -1;  for(it = 1; it < G->n; it++){  int j, min\_pi = oo;  for(j = 1; j <= G->n; j++)  if(mark[j] == 0 && pi[j] < min\_pi){  min\_pi = pi[j];  u = j;  }  mark[u] = 1;  for(v = 1; v <= G->n; v++)  if(G->A[u][v] != NO\_EDGE && mark[v] == 0)  if(pi[u] + G->A[u][v] < pi[v]){  pi[v] = pi[u] + G->A[u][v];  p[v] = u;  }  }  }  int main(){  Graph G;  int n, m, u, v, e, 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);  }  for(u = 1; u <= G.n; u++){  pi[u] = oo;  mark[u] = 0;  }  MooreDijkstra(&G, 1);  for(u = 1; u <= G.n; u++)  printf("pi[%d] = %d, p[%d] = %d\n", u, pi[u], u, p[u]);  } |
| Q2.  #include<stdio.h>  #define MAX\_N 50  #define oo 9999  #define NO\_EDGE -1  typedef struct{  int A[MAX\_N][MAX\_N], n, m;  }Graph;  void init\_graph(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\_edge(Graph \*G, int u, int v, int w){  G->A[u][v] = w;  G->A[v][u] = w;  G->m++;  }  int mark[MAX\_N], pi[MAX\_N], p[MAX\_N];  void MooreDijkstra(Graph \*G, int s){  int u, v, it;  for(u = 1; u <= G->n; u++){  pi[u] = oo;  mark[u] = 0;  }  pi[s] = 0;  p[s] = -1;  for(it = 1; it < G->n; it++){  int j, min\_pi = oo;  for(j = 1; j <= G->n; j++)  if(mark[j] == 0 && pi[j] < min\_pi){  min\_pi = pi[j];  u = j;  }  mark[u] = 1;  for(v = 1; v <= G->n; v++)  if(G->A[u][v] != NO\_EDGE && mark[v] == 0)  if(pi[u] + G->A[u][v] < pi[v]){  pi[v] = pi[u] + G->A[u][v];  p[v] = u;  }  }  }  int main(){  Graph G;  int n, m, u, v, e, 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);  }  for(u = 1; u <= G.n; u++){  pi[u] = oo;  mark[u] = 0;  }  MooreDijkstra(&G, 1);  for(u = 1; u <= G.n; u++)  printf("pi[%d] = %d, p[%d] = %d\n", u, pi[u], u, p[u]);  } |

1. **BT2 - Thuật toán Moore - Dijkstra (chiều dài)**

|  |
| --- |
| Q1.  #include<stdio.h>  #define MAX\_N 50  #define oo 9999  #define NO\_EDGE -1  typedef struct{  int A[MAX\_N][MAX\_N], n, m;  }Graph;  void init\_graph(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\_edge(Graph \*G, int u, int v, int w){  G->A[u][v] = w;  G->m++;  }  int mark[MAX\_N], pi[MAX\_N], p[MAX\_N];  void MooreDijkstra(Graph \*G, int s){  int u, v, it;  for(u = 1; u <= G->n; u++){  pi[u] = oo;  mark[u] = 0;  }  pi[s] = 0;  p[s] = -1;  for(it = 1; it < G->n; it++){  int j, min\_pi = oo;  for(j = 1; j <= G->n; j++)  if(mark[j] == 0 && pi[j] < min\_pi){  min\_pi = pi[j];  u = j;  }  mark[u] = 1;  for(v = 1; v <= G->n; v++)  if(G->A[u][v] != NO\_EDGE && mark[v] == 0)  if(pi[u] + G->A[u][v] < pi[v]){  pi[v] = pi[u] + G->A[u][v];  p[v] = u;  }  }  }  int main(){  Graph G;  int n, m, u, v, e, 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);  }  for(u = 1; u <= G.n; u++){  pi[u] = oo;  mark[u] = 0;  }  MooreDijkstra(&G, 1);  if(pi[n] < oo)  printf("%d",pi[n]);  else printf("-1");  } |
| Q2.  #include<stdio.h>  #define MAX\_N 50  #define oo 9999  #define NO\_EDGE -1  typedef struct{  int A[MAX\_N][MAX\_N], n, m;  }Graph;  void init\_graph(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\_edge(Graph \*G, int u, int v, int w){  G->A[u][v] = w;  G->A[v][u] = w;  G->m++;  }  int mark[MAX\_N], pi[MAX\_N], p[MAX\_N];  void MooreDijkstra(Graph \*G, int s){  int u, v, it;  for(u = 1; u <= G->n; u++){  pi[u] = oo;  mark[u] = 0;  }  pi[s] = 0;  p[s] = -1;  for(it = 1; it < G->n; it++){  int j, min\_pi = oo;  for(j = 1; j <= G->n; j++)  if(mark[j] == 0 && pi[j] < min\_pi){  min\_pi = pi[j];  u = j;  }  mark[u] = 1;  for(v = 1; v <= G->n; v++)  if(G->A[u][v] != NO\_EDGE && mark[v] == 0)  if(pi[u] + G->A[u][v] < pi[v]){  pi[v] = pi[u] + G->A[u][v];  p[v] = u;  }  }  }  int main(){  Graph G;  int n, m, u, v, e, 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);  }  for(u = 1; u <= G.n; u++){  pi[u] = oo;  mark[u] = 0;  }  MooreDijkstra(&G, 1);  if(pi[n] < oo)  printf("%d",pi[n]);  else printf("-1");  } |

1. **BT3 - Thuật toán Moore - Dijkstra (đường đi)**

|  |
| --- |
| Q1.  #include<stdio.h>  #define MAX\_N 50  #define oo 9999  #define NO\_EDGE -1  typedef struct{  int A[MAX\_N][MAX\_N], n, m;  }Graph;  void init\_graph(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\_edge(Graph \*G, int u, int v, int w){  G->A[u][v] = w;  G->m++;  }  int mark[MAX\_N], pi[MAX\_N], p[MAX\_N];  void MooreDijkstra(Graph \*G, int s){  int u, v, it;  for(u = 1; u <= G->n; u++){  pi[u] = oo;  mark[u] = 0;  }  pi[s] = 0;  p[s] = -1;  for(it = 1; it < G->n; it++){  int j, min\_pi = oo;  for(j = 1; j <= G->n; j++)  if(mark[j] == 0 && pi[j] < min\_pi){  min\_pi = pi[j];  u = j;  }  mark[u] = 1;  for(v = 1; v <= G->n; v++)  if(G->A[u][v] != NO\_EDGE && mark[v] == 0)  if(pi[u] + G->A[u][v] < pi[v]){  pi[v] = pi[u] + G->A[u][v];  p[v] = u;  }  }  }  int main(){  Graph G;  int n, m, u, v, e, 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);  }  for(u = 1; u <= G.n; u++){  pi[u] = oo;  mark[u] = 0;  }  int s, t;  scanf("%d%d",&s,&t);  MooreDijkstra(&G, s);  int path[MAX\_N], k = 0, current = t;  while(current != -1){  path[k] = current; k++;  current = p[current];  }  for(u = k - 1; u >= 0; u--){  printf("%d ",path[u]);  if(u != 0)  printf("-> ");  }  } |
| Q2.  #include<stdio.h>  #define MAX\_N 50  #define oo 9999  #define NO\_EDGE -1  typedef struct{  int A[MAX\_N][MAX\_N], n, m;  }Graph;  void init\_graph(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\_edge(Graph \*G, int u, int v, int w){  G->A[u][v] = w;  G->A[v][u] = w;  G->m++;  }  int mark[MAX\_N], pi[MAX\_N], p[MAX\_N];  void MooreDijkstra(Graph \*G, int s){  int u, v, it;  for(u = 1; u <= G->n; u++){  pi[u] = oo;  mark[u] = 0;  }  pi[s] = 0;  p[s] = -1;  for(it = 1; it < G->n; it++){  int j, min\_pi = oo;  for(j = 1; j <= G->n; j++)  if(mark[j] == 0 && pi[j] < min\_pi){  min\_pi = pi[j];  u = j;  }  mark[u] = 1;  for(v = 1; v <= G->n; v++)  if(G->A[u][v] != NO\_EDGE && mark[v] == 0)  if(pi[u] + G->A[u][v] < pi[v]){  pi[v] = pi[u] + G->A[u][v];  p[v] = u;  }  }  }  int main(){  Graph G;  int n, m, u, v, e, 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);  }  for(u = 1; u <= G.n; u++){  pi[u] = oo;  mark[u] = 0;  }  int s, t;  scanf("%d%d",&s,&t);  MooreDijkstra(&G, s);  int path[MAX\_N], k = 0, current = t;  while(current != -1){  path[k] = current; k++;  current = p[current];  }  for(u = k - 1; u >= 0; u--){  printf("%d ",path[u]);  if(u != 0)  printf("-> ");  }  } |

1. **BT4 - Mê cung số**

|  |
| --- |
| #include<stdio.h>  #define MAX\_N 50  #define oo 9999  #define NO\_EDGE -1  typedef struct{  int A[MAX\_N][MAX\_N], n, m;  }Graph;  void init\_graph(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\_edge(Graph \*G, int u, int v, int w){  G->A[u][v] = w;  G->m++;  }  int mark[MAX\_N], pi[MAX\_N], p[MAX\_N], mecung[MAX\_N];  void MooreDijkstra(Graph \*G, int s){  int u, v, it;  for(u = 1; u <= G->n; u++){  pi[u] = oo;  mark[u] = 0;  }  pi[s] = 0;  p[s] = -1;  for(it = 1; it < G->n; it++){  int j, min\_pi = oo;  for(j = 1; j <= G->n; j++)  if(mark[j] == 0 && pi[j] < min\_pi){  min\_pi = pi[j];  u = j;  }  mark[u] = 1;  for(v = 1; v <= G->n; v++)  if(G->A[u][v] != NO\_EDGE && mark[v] == 0)  if(pi[u] + G->A[u][v] < pi[v]){  pi[v] = pi[u] + G->A[u][v];  p[v] = u;  }  }  }  int main(){  Graph G;  int n, m, u, v;  scanf("%d%d",&m,&n);  init\_graph(&G, n\*m);  for(int i = 0; i < m; i++)  for(int j = 0; j < n; j++){  scanf("%d ",&u);  mecung[i\*n+j+1] = u;  }  for(int i = 0; i < m; i++)  for(int j = 0; j < n; j++){  int di[] = {-1,1,0,0};  int dj[] = {0,0,-1,1};  for(int k = 0; k < 4; k++){  int i\_ke = i + di[k];  int j\_ke = j + dj[k];  if( (i\_ke >= 0) && (i\_ke < m) && (j\_ke >= 0) && (j\_ke < n) ){  v = i\_ke \* n + j\_ke + 1;  u = i \* n + j + 1;  G.A[u][v] = mecung[v];  }  }  }  MooreDijkstra(&G, 1);  printf("%d",pi[G.n]);  } |

1. **BT5 - Ô kiều (Ngưu Lang - Chức Nữ)** (nằm ở BT bổ sung 2)
2. **BT6 - Thuật toán Bellman - Ford**

|  |
| --- |
| Q1.  #include<stdio.h>  #define MAX\_N 50  #define oo 9999  typedef struct{  int u, v, w;  }Edge;  typedef struct{  Edge edge[MAX\_N];  int n, m;  }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->edge[G->m].u = u;  G->edge[G->m].v = v;  G->edge[G->m].w = w;  G->m++;  }  int pi[MAX\_N], p[MAX\_N];  void BellmanFord(Graph \*G, int s){  int u, v, w;  for(u = 1; u <= G->n; u++)  pi[u] = oo;  pi[s] = 0;  p[s] = -1;  for(int it = 1; it < G->n; it++)  for(int k = 0; k < G->m; k++){  u = G->edge[k].u;  v = G->edge[k].v;  w = G->edge[k].w;  if(pi[u] == oo)  continue;  if(pi[u] + w < pi[v]){  pi[v] = pi[u] + w;  p[v] = u;  }  }  }  int main(){  Graph G;  int n, m, u, v, e, 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);  }  int s, t;  scanf("%d%d",&s,&t);  BellmanFord(&G, s);  if(pi[t] < oo)  printf("%d", pi[t]);  else printf("-1");  } |
| Q2.  #include<stdio.h>  #define MAX\_N 50  #define oo 9999  typedef struct{  int u, v, w;  }Edge;  typedef struct{  Edge edge[MAX\_N];  int n, m;  }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->edge[G->m].u = u;  G->edge[G->m].v = v;  G->edge[G->m].w = w;  G->m++;  }  int pi[MAX\_N], p[MAX\_N];  void BellmanFord(Graph \*G, int s){  int u, v, w;  for(u = 1; u <= G->n; u++)  pi[u] = oo;  pi[s] = 0;  p[s] = -1;  for(int it = 1; it < G->n; it++)  for(int k = 0; k < G->m; k++){  u = G->edge[k].u;  v = G->edge[k].v;  w = G->edge[k].w;  if(pi[u] == oo)  continue;  if(pi[u] + w < pi[v]){  pi[v] = pi[u] + w;  p[v] = u;  }  }  }  int main(){  Graph G;  int n, m, u, v, e, 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);  }  int s, t;  scanf("%d%d",&s,&t);  BellmanFord(&G, s);  int path[MAX\_N], k = 0, current = t;  while(current != -1){  path[k] = current; k++;  current = p[current];  }  for(u = k - 1; u >= 0; u--){  printf("%d ", path[u]);  if(u != 0)  printf("-> ");  }  } |

1. **BT7 - Thuật toán Bellman - Ford (kiểm tra chu trình âm)**

|  |
| --- |
| #include<stdio.h>  #define MAX\_N 50  #define oo 9999  typedef struct{  int u, v, w;  }Edge;  typedef struct{  Edge edge[MAX\_N];  int n, m;  }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->edge[G->m].u = u;  G->edge[G->m].v = v;  G->edge[G->m].w = w;  G->m++;  }  int pi[MAX\_N], p[MAX\_N];  int BellmanFord(Graph \*G, int s){  int u, v, w;  for(u = 1; u <= G->n; u++)  pi[u] = oo;  pi[s] = 0;  p[s] = -1;  for(int it = 1; it < G->n; it++)  for(int k = 0; k < G->m; k++){  u = G->edge[k].u;  v = G->edge[k].v;  w = G->edge[k].w;  if(pi[u] == oo)  continue;  if(pi[u] + w < pi[v]){  pi[v] = pi[u] + w;  p[v] = u;  }  }  for(int k = 0; k < G->m; k++){  u = G->edge[k].u;  v = G->edge[k].v;  w = G->edge[k].w;  if(pi[u] == oo)  continue;  if(pi[u] + w < pi[v]){  return 1;  }  }  return 0;  }  int main(){  Graph G;  int n, m, u, v, e, 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);  }  int s;  scanf("%d",&s);  if(BellmanFord(&G, s))  printf("YES");  else printf("NO");    } |

1. **BT8 - Extended traffic** (Bài này bị đóng)
2. **BT9 - Thuật toán Floyd - Warshall (đường đi ngắn nhất giữa các cặp đỉnh)**

|  |
| --- |
| Q1.  #include<stdio.h>  #define MAX\_N 40  #define oo 9999  #define NO\_EDGE 0  typedef struct{  int n, m;  int A[MAX\_N][MAX\_N];  }Graph;  void init\_graph(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\_edge(Graph \*G, int u, int v, int w){  G->A[u][v] = w;  G->m++;  }  int pi[MAX\_N][MAX\_N], next[MAX\_N][MAX\_N];  void FloydWarshall(Graph \*G){  int u, v, k;  for(u = 1; u <= G->n; u++)  for(v = 1; v <= G->n; v++){  pi[u][v] = oo;  next[u][v] = -1;  }  for(u = 1; u <= G->n; u++)  pi[u][u] = 0;  for(u = 1; u <= G->n; u++)  for(v = 1; v <= G->n; v++){  if(G->A[u][v] != NO\_EDGE){  pi[u][v] = G->A[u][v];  next[u][v] = v;  }  }  for (k = 1; k <= G->n; k++)  for (u = 1; u <= G->n; u++) {  if (pi[u][k] == oo)  continue;  for (v = 1; v <= G->n; v++){  if (pi[k][v] == oo)  continue;  if (pi[u][k] + pi[k][v] < pi[u][v]){  pi[u][v] = pi[u][k] + pi[k][v];  next[u][v] = next[u][k];  }  }  }  }  int main(){  Graph G;  int u, v, n, m, e, 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);  }  for(u = 1; u <= G.n; u++)  for(v = 1; v <= G.n; v++){  pi[u][v] = oo;  next[u][v] = -1;  }  FloydWarshall(&G);  for(u = 1; u <= G.n; u++)  for(v = 1; v <= G.n; v++){  printf("%d -> %d: ", u, v);  if(pi[u][v] == oo || pi[u][v] > 9990)  printf("NO PATH\n");  else printf("%d\n", pi[u][v]);  }  return 0;  } |
| Q2.  #include<stdio.h>  #define MAX\_N 40  #define oo 9999  #define NO\_EDGE 0  typedef struct{  int n, m;  int A[MAX\_N][MAX\_N];  }Graph;  void init\_graph(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\_edge(Graph \*G, int u, int v, int w){  G->A[u][v] = w;  G->m++;  }  int pi[MAX\_N][MAX\_N], next[MAX\_N][MAX\_N];  void FloydWarshall(Graph \*G){  int u, v, k;  for(u = 1; u <= G->n; u++)  for(v = 1; v <= G->n; v++){  pi[u][v] = oo;  next[u][v] = -1;  }  for(u = 1; u <= G->n; u++)  pi[u][u] = 0;  for(u = 1; u <= G->n; u++)  for(v = 1; v <= G->n; v++){  if(G->A[u][v] != NO\_EDGE){  pi[u][v] = G->A[u][v];  next[u][v] = v;  }  }  for (k = 1; k <= G->n; k++)  for (u = 1; u <= G->n; u++) {  if (pi[u][k] == oo)  continue;  for (v = 1; v <= G->n; v++){  if (pi[k][v] == oo)  continue;  if (pi[u][k] + pi[k][v] < pi[u][v]){  pi[u][v] = pi[u][k] + pi[k][v];  next[u][v] = next[u][k];  }  }  }  }  int main(){  Graph G;  int u, v, n, m, e, 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);  }  for(u = 1; u <= G.n; u++)  for(v = 1; v <= G.n; v++){  pi[u][v] = oo;  next[u][v] = -1;  }  FloydWarshall(&G);  for(u = 1; u <= G.n; u++){  for(v = 1; v <= G.n; v++){  if(pi[u][v] < oo){  printf("path(%d, %d): %d", u, v, u);  int current = u;  while(current != v){  current = next[current][v];  printf(" -> %d", current);  }  printf("\n");  }  else printf("path(%d, %d): NO PATH\n", u, v);  }  }  return 0;  } |

1. **BT10 - Thuật toán Floyd - Warshall (kiểm tra chu trình âm)**

|  |
| --- |
| #include<stdio.h>  #define MAX\_N 40  #define oo 9999  #define NO\_EDGE 0  typedef struct{  int n, m;  int A[MAX\_N][MAX\_N];  }Graph;  void init\_graph(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\_edge(Graph \*G, int u, int v, int w){  G->A[u][v] = w;  G->m++;  }  int pi[MAX\_N][MAX\_N], next[MAX\_N][MAX\_N];  int FloydWarshall(Graph \*G){  int u, v, k;  for(u = 1; u <= G->n; u++)  for(v = 1; v <= G->n; v++){  pi[u][v] = oo;  next[u][v] = -1;  }  for(u = 1; u <= G->n; u++)  pi[u][u] = 0;  for(u = 1; u <= G->n; u++)  for(v = 1; v <= G->n; v++){  if(G->A[u][v] != NO\_EDGE){  pi[u][v] = G->A[u][v];  next[u][v] = v;  }  }  for (k = 1; k <= G->n; k++)  for (u = 1; u <= G->n; u++) {  if (pi[u][k] == oo)  continue;  for (v = 1; v <= G->n; v++){  if (pi[k][v] == oo)  continue;  if (pi[u][k] + pi[k][v] < pi[u][v]){  pi[u][v] = pi[u][k] + pi[k][v];  next[u][v] = next[u][k];  }  }  }  for(u = 1; u <= G->n; u++)  if(pi[u][u] < 0){  return 1;  }  return 0;  }  int main(){  Graph G;  int u, v, n, m, e, 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);  }  for(u = 1; u <= G.n; u++)  for(v = 1; v <= G.n; v++){  pi[u][v] = oo;  next[u][v] = -1;  }  if(FloydWarshall(&G) == 1)  printf("YES");  else printf("NO");    return 0;  } |

\*BT bổ sung:

1. **BT 1 - 2 (Tìm đường đi ngắn nhất)**

|  |
| --- |
| Q1.  #include<stdio.h>  #define MAX\_N 50  #define oo 9999  #define NO\_EDGE -1  typedef struct{  int A[MAX\_N][MAX\_N], n, m;  }Graph;  void init\_graph(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\_edge(Graph \*G, int u, int v, int w){  G->A[u][v] = w;  G->m++;  }  int mark[MAX\_N], pi[MAX\_N], p[MAX\_N];  void MooreDijkstra(Graph \*G, int s){  int u, v, it;  for(u = 1; u <= G->n; u++){  pi[u] = oo;  mark[u] = 0;  }  pi[s] = 0;  p[s] = -1;  for(it = 1; it < G->n; it++){  int j, min\_pi = oo;  for(j = 1; j <= G->n; j++)  if(mark[j] == 0 && pi[j] < min\_pi){  min\_pi = pi[j];  u = j;  }  mark[u] = 1;  for(v = 1; v <= G->n; v++)  if(G->A[u][v] != NO\_EDGE && mark[v] == 0)  if(pi[u] + G->A[u][v] < pi[v]){  pi[v] = pi[u] + G->A[u][v];  p[v] = u;  }  }  }  int main(){  Graph G;  int n, m, u, v, e, 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);  }  for(u = 1; u <= G.n; u++){  pi[u] = oo;  mark[u] = 0;  }  MooreDijkstra(&G, 1);  if(pi[n] < oo)  printf("%d",pi[n]);  else printf("-1");  } |

|  |
| --- |
| Q2.  #include<stdio.h>  #define MAX\_N 50  #define oo 9999  #define NO\_EDGE -1  typedef struct{  int A[MAX\_N][MAX\_N], n, m;  }Graph;  void init\_graph(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\_edge(Graph \*G, int u, int v, int w){  G->A[u][v] = w;  G->m++;  }  int mark[MAX\_N], pi[MAX\_N], p[MAX\_N];  void MooreDijkstra(Graph \*G, int s){  int u, v, it;  for(u = 1; u <= G->n; u++){  pi[u] = oo;  mark[u] = 0;  }  pi[s] = 0;  p[s] = -1;  for(it = 1; it < G->n; it++){  int j, min\_pi = oo;  for(j = 1; j <= G->n; j++)  if(mark[j] == 0 && pi[j] < min\_pi){  min\_pi = pi[j];  u = j;  }  mark[u] = 1;  for(v = 1; v <= G->n; v++)  if(G->A[u][v] != NO\_EDGE && mark[v] == 0)  if(pi[u] + G->A[u][v] < pi[v]){  pi[v] = pi[u] + G->A[u][v];  p[v] = u;  }  }  }  int main(){  Graph G;  int n, m, u, v, e, 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);  }  for(u = 1; u <= G.n; u++){  pi[u] = oo;  mark[u] = 0;  }  MooreDijkstra(&G, 1);  if(pi[n] < oo)  printf("%d",pi[n]);  else printf("-1");  } |

1. **Kiểm tra chu trình âm và ứng dụng đường đi ngắn nhất**

|  |
| --- |
| Q1.  #include<stdio.h>  #define MAX\_N 1000  #define oo 9999  typedef struct{  int u, v, w;  }Edge;  typedef struct{  int n, m;  Edge edge[MAX\_N];  }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->edge[G->m].u = u;  G->edge[G->m].v = v;  G->edge[G->m].w = w;  G->m++;  }  int pi[MAX\_N], p[MAX\_N], negative\_cycle = 0;;  void BellmanFord(Graph \*G, int s){  int u, v, w, it, k;  for(u = 1; u <= G->n; u++)  pi[u] = oo;  pi[s] = 0;  p[s] = -1;  for(it = 1; it < G->n; it++){  for(k = 1; k <= G->n; k++){  u = G->edge[k].u;  v = G->edge[k].v;  w = G->edge[k].w;  if(pi[u] == oo)  continue;  if(pi[u] + w < pi[v]){  pi[v] = pi[u] + w;  p[v] = u;  }  }  }  for(k = 1; k <= G->n; k++){  u = G->edge[k].u;  v = G->edge[k].v;  w = G->edge[k].w;  if(pi[u] + w < pi[v]){  negative\_cycle = 1;  break;  }  }  }  int main(){  Graph G;  int n, m, u, v, e, 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);  }  for(u = 1; u <= G.n; u++){  pi[u] = oo;  }  negative\_cycle = 0;  BellmanFord(&G, 1);  if(negative\_cycle != 0)  printf("negative cycle");  else printf("ok");  } |
| Q2.  #include<stdio.h>  #define MAX\_N 40  #define oo 9999999  #define NO\_EDGE -1  typedef struct {  int A[MAX\_N][MAX\_N];  int m, n;  } Graph;  void init\_graph(Graph \*G, int n){  G->n = n;  G->m = 0;  int u, v;  for(u=1;u<=G->n;u++)  for(v=1;v<=G->n;v++)  G->A[u][v] = NO\_EDGE;  }  void add\_edge(Graph \*G, int u, int v, int w){  G->A[u][v] = w;  G->m++;  }  int pi[MAX\_N];  int p[MAX\_N];  int mark[MAX\_N];  void MooreDijkstra(Graph \*G, int s) {  for (int u = 1; u <= G->n; u++){  mark[u] = 0;  pi[u] = oo;  }  pi[s] = 0;  p[s] = -1;  for(int it = 1; it < G->n; it++){  int u, k, min\_pi = oo;  for(k = 1; k <= G->n; k++)  if(pi[k] < min\_pi && mark[k] == 0){  min\_pi = pi[k];  u = k;  }  mark[u] = 1;  int v;  for(v = 1; v <= G->n; v++)  if(G->A[u][v] != NO\_EDGE && mark[v] == 0)  if(pi[u] + G->A[u][v] < pi[v]){  pi[v] = pi[u] + G->A[u][v];  p[v] = u;  }  }  }  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);  }  for(u=1;u<=G.n;u++){  mark[u] = 0;  pi[u] = oo;  }  MooreDijkstra(&G, 1);  printf("%d",pi[n]);  } |

1. **Ứng dụng đường đi ngắn nhất**

|  |
| --- |
| #include<stdio.h>  #define MAX\_N 40  #define oo 9999999  #define NO\_EDGE -1  typedef struct {  int A[MAX\_N][MAX\_N];  int m, n;  } Graph;  void init\_graph(Graph \*G, int n){  G->n = n;  G->m = 0;  int u, v;  for(u=1;u<=G->n;u++)  for(v=1;v<=G->n;v++)  G->A[u][v] = NO\_EDGE;  }  void add\_edge(Graph \*G, int u, int v, int w){  G->A[u][v] = w;  G->A[v][u] = w;  G->m++;  }  int pi[MAX\_N];  int p[MAX\_N];  int mark[MAX\_N];  void MooreDijkstra(Graph \*G, int s) {  for (int u = 1; u <= G->n; u++){  mark[u] = 0;  pi[u] = oo;  }  pi[s] = 0;  p[s] = -1;  for(int it = 1; it < G->n; it++){  int u, k, min\_pi = oo;  for(k = 1; k <= G->n; k++)  if(pi[k] < min\_pi && mark[k] == 0){  min\_pi = pi[k];  u = k;  }  mark[u] = 1;  int v;  for(v = 1; v <= G->n; v++)  if(G->A[u][v] != NO\_EDGE && mark[v] == 0)  if(pi[u] + G->A[u][v] < pi[v]){  pi[v] = pi[u] + G->A[u][v];  p[v] = u;  }  }  }  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);  }  for(u=1;u<=G.n;u++){  mark[u] = 0;  pi[u] = oo;  }  int s, t;  scanf("%d%d",&s,&t);  MooreDijkstra(&G, s);  printf("%d",pi[t]);  } |

1. **Mê cung số (Number Maze)**

|  |
| --- |
| #include<stdio.h>  #define MAX\_N 50  #define oo 9999  #define NO\_EDGE -1  typedef struct{  int A[MAX\_N][MAX\_N], n, m;  }Graph;  void init\_graph(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\_edge(Graph \*G, int u, int v, int w){  G->A[u][v] = w;  G->m++;  }  int mark[MAX\_N], pi[MAX\_N], p[MAX\_N], mecung[MAX\_N];  void MooreDijkstra(Graph \*G, int s){  int u, v, it;  for(u = 1; u <= G->n; u++){  pi[u] = oo;  mark[u] = 0;  }  pi[s] = 0;  p[s] = -1;  for(it = 1; it < G->n; it++){  int j, min\_pi = oo;  for(j = 1; j <= G->n; j++)  if(mark[j] == 0 && pi[j] < min\_pi){  min\_pi = pi[j];  u = j;  }  mark[u] = 1;  for(v = 1; v <= G->n; v++)  if(G->A[u][v] != NO\_EDGE && mark[v] == 0)  if(pi[u] + G->A[u][v] < pi[v]){  pi[v] = pi[u] + G->A[u][v];  p[v] = u;  }  }  }  int main(){  Graph G;  int n, m, u, v;  scanf("%d%d",&m,&n);  init\_graph(&G, n\*m);  for(int i = 0; i < m; i++)  for(int j = 0; j < n; j++){  scanf("%d ",&u);  mecung[i\*n+j+1] = u;  }  for(int i = 0; i < m; i++)  for(int j = 0; j < n; j++){  int di[] = {-1,1,0,0};  int dj[] = {0,0,-1,1};  for(int k = 0; k < 4; k++){  int i\_ke = i + di[k];  int j\_ke = j + dj[k];  if( (i\_ke >= 0) && (i\_ke < m) && (j\_ke >= 0) && (j\_ke < n) ){  v = i\_ke \* n + j\_ke + 1;  u = i \* n + j + 1;  G.A[u][v] = mecung[v];  }  }  }  MooreDijkstra(&G, 1);  printf("%d",pi[G.n]);  } |

1. **Bellman - Ford pi và p**

|  |
| --- |
| #include<stdio.h>  #define MAX\_N 40  #define oo 9999999  typedef struct{  int u, v, w;  }Edge;  typedef struct {  Edge edge[MAX\_N];  int m, n;  }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->edge[G->m].u = u;  G->edge[G->m].v = v;  G->edge[G->m].w = w;  G->m++;  }  int pi[MAX\_N];  int p[MAX\_N];  void BellmanFord(Graph \*G, int s) {  int u, v, w, k;  for (int u = 1; u <= G->n; u++){  pi[u] = oo;  }  pi[s] = 0;  p[s] = -1;  for(int it = 1; it < G->n; it++){  for(k = 0; k < G->m; k++){  u = G->edge[k].u;  v = G->edge[k].v;  w = G->edge[k].w;  if(pi[u] == oo)  continue;  if(pi[u] + w < pi[v]){  pi[v] = pi[u] + w;  p[v] = u;  }  }  }  }  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);  }  for(u=1;u<=G.n;u++){  pi[u] = oo;  }  BellmanFord(&G, 1);  for(u=1;u<=G.n;u++)  printf("pi[%d] = %d, p[%d] = %d\n", u, pi[u], u, p[u]);  } |

1. **Floyd - Warshall**

|  |
| --- |
| #include<stdio.h>  #define MAX\_N 40  #define oo 9999  #define NO\_EDGE 0  typedef struct{  int n, m;  int A[MAX\_N][MAX\_N];  }Graph;  void init\_graph(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\_edge(Graph \*G, int u, int v, int w){  G->A[u][v] = w;  G->m++;  }  int pi[MAX\_N][MAX\_N], next[MAX\_N][MAX\_N];  void FloydWarshall(Graph \*G){  int u, v, k;  for(u = 1; u <= G->n; u++)  for(v = 1; v <= G->n; v++){  pi[u][v] = oo;  next[u][v] = -1;  }  for(u = 1; u <= G->n; u++)  pi[u][u] = 0;  for(u = 1; u <= G->n; u++)  for(v = 1; v <= G->n; v++){  if(G->A[u][v] != NO\_EDGE){  pi[u][v] = G->A[u][v];  next[u][v] = v;  }  }  for (k = 1; k <= G->n; k++)  for (u = 1; u <= G->n; u++) {  if (pi[u][k] == oo)  continue;  for (v = 1; v <= G->n; v++){  if (pi[k][v] == oo)  continue;  if (pi[u][k] + pi[k][v] < pi[u][v]){  pi[u][v] = pi[u][k] + pi[k][v];  next[u][v] = next[u][k];  }  }  }  }  int main(){  Graph G;  int u, v, n, m, e, 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);  }  for(u = 1; u <= G.n; u++)  for(v = 1; v <= G.n; v++){  pi[u][v] = oo;  next[u][v] = -1;  }  FloydWarshall(&G);  for(u = 1; u <= G.n; u++)  for(v = 1; v <= G.n; v++){  printf("%d -> %d: ", u, v);  if(pi[u][v] == oo || pi[u][v] > 9990)  printf("NO PATH\n");  else printf("%d\n", pi[u][v]);  }  return 0;  } |

1. **Bellman - Ford**

|  |
| --- |
| #include <stdio.h>  #define MAX\_N 100  #define oo 999999  #define NO\_EDGE -1  typedef struct {  int u, v;  int w;  } Edge;  typedef struct {  int n, m;  Edge edges[MAX\_N];  } 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++;  }  int pi[MAX\_N];  int p[MAX\_N];  int BellmanFord(Graph \*G, int s) {  for (int u = 1; u <= G->n; u++) {  pi[u] = oo;  }  pi[s] = 0;  p[s] = -1;  for (int it = 1; it < G->n; it++) {  for (int k = 0; k < G->m; k++) {  int u = G->edges[k].u;  int v = G->edges[k].v;  int w = G->edges[k].w;  if (pi[u] == oo)  continue;  if (pi[u] + w < pi[v]) {  pi[v] = pi[u] + w;  p[v] = u;  }  }  }  for (int k = 0; k < G->m; k++) {  int u = G->edges[k].u;  int v = G->edges[k].v;  int w = G->edges[k].w;  if (pi[u] == oo)  continue;  if (pi[u] + w < pi[v]) {  return 1;  }  }  return 0;  }  int main() {  Graph G;  int n, m, 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);  }  int s;  scanf("%d", &s);  if (BellmanFord(&G, s) == 1)  printf("YES\n");  else  printf("NO\n");  return 0;  } |

1. **Tìm số đường đi ngắn nhất (nâng cao)**

|  |
| --- |
| #include<stdio.h>  #define MAX 50  #define oo 9999  #define NO\_EDGE -1  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){  if( (u < 1 || u > G->n) || (v < 1 || v > G->n) || w < 0) return;  G->A[u][v] = w;  G->A[v][u] = w;  G->m++;  }  int pi[MAX];  int cnt[MAX];  int mark[MAX];  void dijkstra(Graph \*G, int s, int t){  for(int u = 1; u <= G->n; u++){  pi[u] = oo;  mark[u] = 0;  cnt[u] = 0;  }  pi[s] = 0;  for(int k = 1; k < G->n; k++){  int min\_pi = oo, u;  for(int v = 1; v <= G->n; v++){  if(!mark[v] && pi[v] < min\_pi){  min\_pi = pi[v];  u = v;  }  }  mark[u] = 1;  cnt[u] = 1;  for(int v = 1; v <= G->n; v++){  if(!mark[v] && G->A[u][v] != NO\_EDGE){  if(pi[u] + G->A[u][v] < pi[v]){  pi[v] = pi[u] + G->A[u][v];  cnt[v] = cnt[u];  }  else if(pi[u] + G->A[u][v] == pi[v]){  cnt[v] += cnt[u];  }  }  }  }  if(pi[t] == oo)  puts("-1 0");  else printf("%d %d\n", pi[t], cnt[t]);  }  int main(){  Graph G;  int n, m, u, v, e, w;  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);  }  dijkstra(&G, 1, n);  return 0;  } |