**Câu 1: BFS- Duyệt theo chiều sâu  
Trình bày:**

Dfs(u){

Thăm(u);

vs[u] = 1;

for v ∈ ke(u) do

if (vs[v] = 0) {

e[v]= u; Dfs(v);

**Code C++:**int a[100][100], n, u, vs[100], e[100];

void DfsDequy(int u) { int v;

cout << u << ” ”;

vs[u]= 1;

for (v= 1; v<=n; v++)

if (vs[v]==0 && a[u][v]==1){ e[v]= u;

DfsDequy(v);

**Câu 2: BFS- Duyệt theo chiều rộng  
Trình bày:**

: Bfs(u){ q = ∅;

<Đưa u vào q>;

vs[u] = 1;

while q ≠ ∅ { <Lấy v ra khỏi q>; Thăm(v);

for i ∈ ke(v) do

if (vs[i] = 0) { <Đưa i vào q>;

vs[i] = 1; e[i] = v;

**Code C++:**

int a[100][100], n, u, vs[100], e[100], q[100];

void Bfs(int u){ int v, dq = 1, cq = 0;

cq++; q[cq] = u; vs[u] = 1;

while (dq <= cq){ v = q[dq]; dq++;

cout << v << ” ”;

for (int i= 1; i<=n; i++)

if (vs[i]==0 && a[v][i]==1) {

cq++; q[cq] = i; vs[i] = 1; e[i]= v; }

**Câu 3: Tìm số TPLT bằng DFS, BFS  
DFS:**int a[100][100], vs[100], n;

int ltDfs() { int v;

for (v= 1; v<= n; v++) vs[v]= 0;

DfsDequy(1);

for (v= 1; v<= n; v++) if (vs[v]== 0) return(0);

return(1);  
**BFS:**

int a[100][100], vs[100], n;

int ltBfs() {int v;

for (v= 1; v<= n; v++) vs[v]= 0;

Bfs(1);

for (v= 1; v<= n; v++) if (vs[v]== 0) return(0);

return(1);

**Câu 4: Euler**

**Trình bày:**

**Bước 1**(Kiểm tra điều kiện): Nếu G không thỏa mãn điều kiện thì kt= 0, nếu có chu trình Euler thì kt= 1; nếu có đường đi Euler thì kt= 2;

**Bước 2**:

(2.1) Nếu kt= 0 ⇒ thông báo đồ thị không có chu trình/đường đi Euler và dừng;

(2.2) Nếu kt= 1 ⇒ chọn u là đỉnh cho trước và chuyển sang bước 3;

(2.3) Nếu kt= 2 ⇒ chọn u là đỉnh bậc lẻ có số hiệu nhỏ nhất và chuyển sang bước 3;

**Bước 3** (Xây dựng chu trình/đường đi Euler bắt đầu từ đỉnh u):

(3.1) Tạo mảng CE để ghi chu trình/ đường đi Euler và Stack để xếp các đỉnh sẽ xét. Xếp đỉnh u vào Stack;

(3.2) Xét đỉnh v nằm trên cùng của Stack và thực hiện:

- Nếu v là đỉnh cô lập thì lấy v ra khỏi Stack và đưa vào CE.

- Nếu v có đỉnh kề là x thì đưa x vào Stack sau đó xóa cạnh nối v với x;

(3.3) Quay lại (3.2) cho tới khi stack rỗng;

**Bước 4**: Xuất chu trình/đường đi Euler chứa trong CE theo thứ tự ngược lại.

**Code C++:**

void ceu(int a[][], int n, int u) {int top=0, v;

top++; s[top]= u; k= 0;

while (top > 0) {int v= s[top];

int ok= 1;

for (int x= 1; x<= n; x++)

if (a[v][x] ==1){top++; s[top]= x; ok= 0;

a[v][x]= 0; a[x][v]= 0; break;

}

if (ok== 1) {k++, ce[k]= v; top--;

}

}

for (v= k; v> 0; v--) cout << ce[v] << “ “;

}

**Câu 5: DijStra**

**Trình bày:  
Khởi tạo**: ∀v∈G: d[v]= a[s][v]; e[v]= s; vs[v]= 0;

(1) Bắt đầu tìm kiếm từ s: d[s]= 0; e[s]= 0; vs[s]= 1;

(2) Tìm đỉnh u sao cho d[u]= min{d[v] | vs[v] = 0};

Nếu không tìm được thì chuyển sang (5). Nếu tìm được thì sang (3).

(3) Đặt vs[u]= 1;

(4) Đối với tất cả v ∈ G thỏa mãn (vs[v]= 0) & (d[v]> d[u] + a[u][v]) thì thay thế: e[v]= u; d[v]= d[u] + a[u][v]; và quay lại (2).

(5) Xuất d[v] và đường đi từ s đến v.

**Code C++:**  
int n, s, a[100][100], d[100], e[100], vs[100];

void Dijkstra(int s){int u, v;

for (v=1; v<= n; v++){d[v]= a[s][v]; e[v]=s; vs[v]= 0}

d[s]= 0; e[s] = 0; vs[s]= 1;

while (1){int u= 0, mind= 32767;

for (v=1; v<= n; v++) if (vs[v]==0 && d[v]< mind){

u= v; mind= d[v];}

if (u== 0) return; vs[u]= 1;

for (v=1; v<=n; v++)

if (vs[v]== 0 && d[v]> d[u]+a[u][v]) {d[v]= d[u] + a[u][v]; e[v] = u; }

**Câu 6: BellMan Ford**

**Trình bày**

Khởi tạo: d[v]= a[s][v]; e[v]= s;

(1) Bắt đầu tìm kiếm từ s: d[s]= 0; e[s]= 0; ok= 0;

(2) Thực hiện n-1 lần lặp:

(2.1) ok= 1;

(2.2) Với mọi đỉnh v ∈ V\{s} thực hiện:

Với mọi đỉnh u ∈ V thực hiện

Nếu (d[v]> d[u] + a[u][v]) thì thay thế:

{ e[v]= u; d[v]= d[u] + a[u][v]; ok= 0; }

(2.3) Nếu (ok= 1) chuyển (3);

(3) Nếu (ok= 1) Xuất d[v] và e[v];

Nếu (ok= 0) Xuất thông báo G chứa chu trình âm;

**Code C++:**

int n, s, a[100][100], d[100], e[100];

int BellmanFord(int s){int dem, u, v;

for (v=1; v<= n; v++){d[v]= a[s][v]; e[v]=s;}

d[s]= 0; e[s] = 0; int ok= 0;

for (dem=1; dem<= n-1; dem++){ int ok= 1;

for (v=1; v<=n; v++)

if (v!=s) for (u=1; u<=n; u++)

if (d[v] > d[u] + a[u][v]) {

d[v] = d[u] + a[u][v]; e[v] = u; ok= 0;}

if (ok== 1) return(1);}

return(0); }

**Câu 7: Floy  
Trình bày:** Khởi tạo: d[i][j]= a[i][j]; e[i][j]= i;

- Với mọi k ∈ G, i ∈ G, j ∈ G sao cho

(d[i][j]> d[i][k] + d[k][j]) thì thay thế:

e[i][j]= k; d[i][j]= d[i][k] + d[k][j];

- Xuất d[i][j] và e[i][j].

**Code C++:** int n, a[100][100], d[100] ][100], e[100] ][100];

void Floyd(){int i, j, k;

for (i=1;i<= n; i++)

for (j=1; j<= n; j++{d[i][j]= a[i][j]; e[i][j]= i;}

for (k=1; k<= n; k++)

for (i=1; i<=n; i++)

for (j=1; j<=n; j++)

if (d[i][j] > d[i][k] + d[k][j]) {

d[i][j] > d[i][k] + d[k][j]; e[i][j] = k; }

**Câu 8: Prim**

**Trình bày** Khởi tạo: T= ∅; VT = {s}; WT= 0;

while (V\VT ≠ ∅) {

<Tìm e = (u, v) có trọng số nhỏ nhất, u ∈ VT, v ∈ V\VT>;

if (Tìm được e) {T = T ∪ {e};

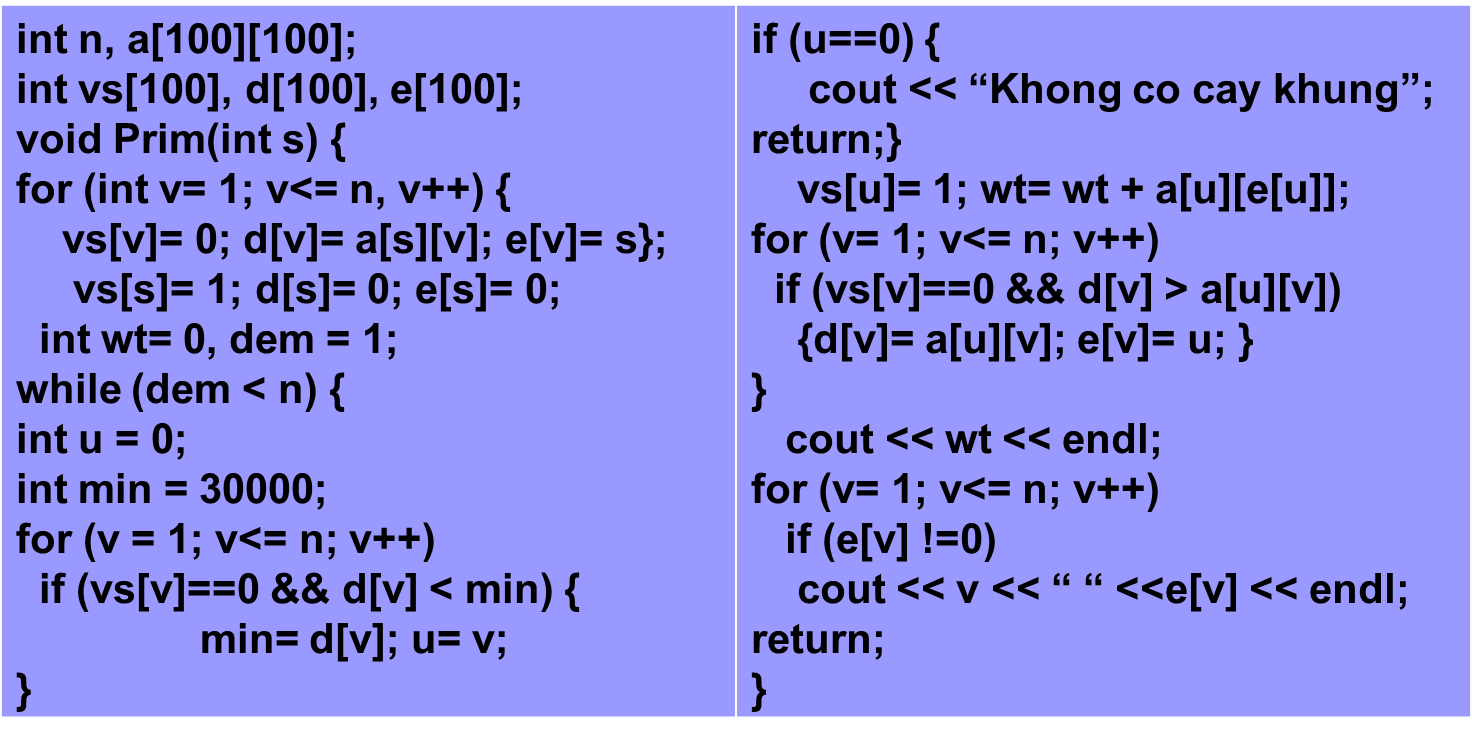
WT= WT + trọng số của e;

VT = VT ∪ {v};

}

else return (G không có cây khung); }

return (T và WT);

Code C++: 

**Câu 9: Kruskal**

**Trình bày Khởi tạo:**

Sắp xếp các cạnh theo thứ tự tăng của trọng số e1, …, em;

T= ∅; WT= 0; k = 0;

for (i=1; i<= m; i++){

if ( T ∪ {ei} không chứa chu trình) {

T = T ∪ {ei}; WT= WT + trọng số của ei;

k++;

if (k = n-1) return (T và WT); }

}

Return (G không có cây khung);   
Code C++:

