CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT 2

LAB 5 – ĐỒ THỊ VÀ BIỂU DIỄN ĐỒ THỊ BẰNG DANH SÁCH KỀ (4 TIẾT)

I. Mục tiêu

Sau khi thực hành, sinh viên cần:

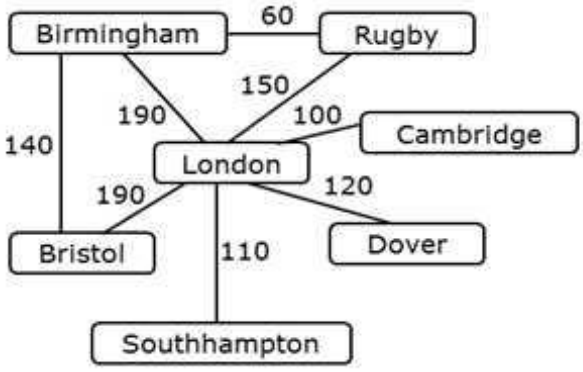
- Nắm vững định nghĩa đồ thị, các khái niệm liên quan đến đồ thị.
- Cài đặt được kiểu dữ liệu đồ thị và các thao tác, phép toán trên đồ thị được biểu diễn bởi danh sách kề.
- Vận dụng kiến thức đã học để giải một số bài toán thực tế.

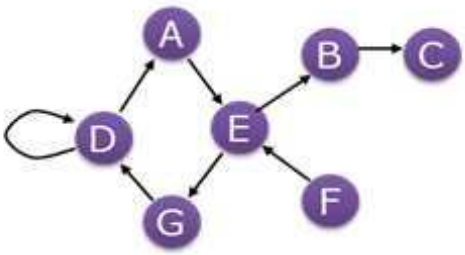
II. Yêu cầu

- Sinh viên phải hoàn thành ***4 bài*** thuộc mục V. Mỗi bài tập tạo một project, xóa các thư mục debug của project này. Sau đó chép các project vào thư mục: Lab5_CTK40_HoTen_MSSV_Nhom#. Nén thư mục , đặt tên tập tin nén theo dạng sau: Lab5_CTK40_HoTen_MSSV_Nhom#.rar.
  Ví dụ: Lab5_CTK40_NguyenVanA_161111_Nhom4.rar.
- Sinh viên sẽ nộp bài Lab qua mạng tại phòng lab theo hướng dẫn của giáo viên.

III. Ôn tập lý thuyết

1. *Đồ thị và các định nghĩa*





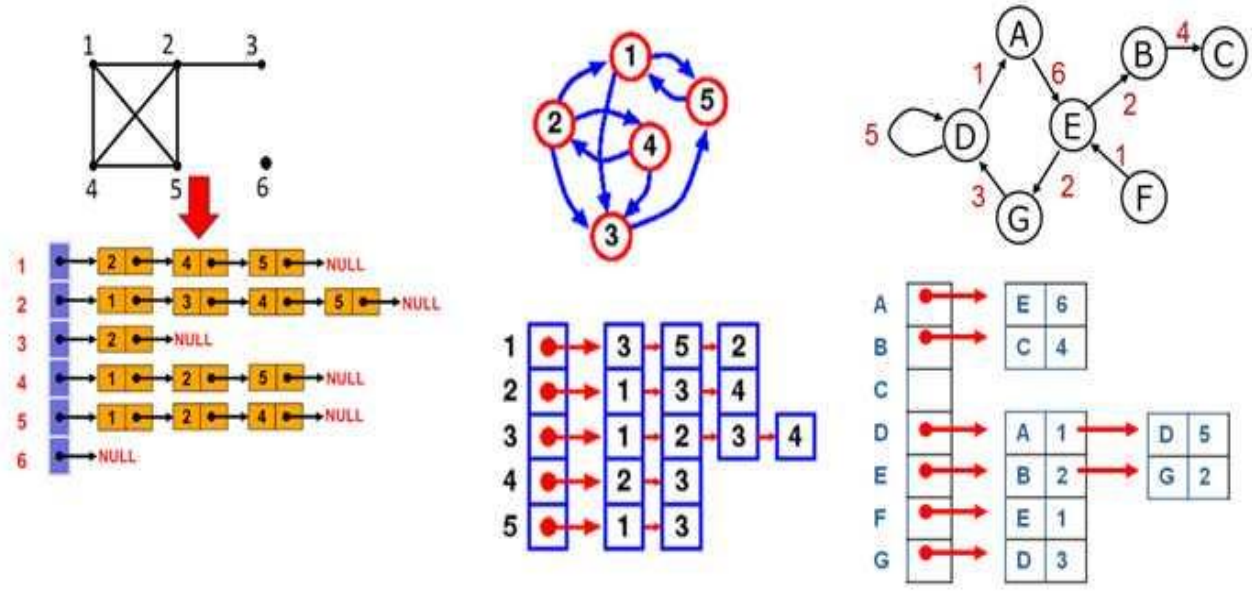
Một đồ thị G được định nghĩa là G = (V, E). Trong đó:

- V(G) là tập hữu hạn các đỉnh và khác rỗng. E(G) là tập các cung (cạnh) = Tập các cặp đỉnh (u, v) mà u, v ϵ V.
- Các đỉnh còn gọi là các nút (node) hay điểm (point)
- Mỗi cung nối giữa hai đỉnh v, w có thể ký hiệu là cặp (v,w). Hai đỉnh có thể trùng nhau. Nếu cặp (v,w) có thứ tự thì gọi là cung có thứ tự hay cạnh có hướng. Ngược lại ta nói là cung không có thứ tự hay cạnh vô hướng.
- Đỉnh kề: Hai đỉnh được gọi là kề nhau nếu chúng được nối với nhau bởi một cạnh. Trong trường hợp này, hai nút đỉnh được gọi là hàng xóm (neighbors) của nhau.
- Đường đi là một dãy tuần tự các đỉnh v_1, v_2, …, v_n sao cho (v_i, v_{i+1}) là một cung trên đồ thị. Đỉnh v_1 được gọi là đỉnh đầu, đỉnh v_n được gọi là đỉnh cuối. Độ dài đường đi là số cạnh trên đường đi.
- Đỉnh X gọi là có thể đi đến được từ đỉnh Y nếu tồn tại một đường đi từ đỉnh Y đến đỉnh X.
- Đường đi đơn là đường đi mà mọi đỉnh trên đó đều khác nhau, ngoại trừ đỉnh đầu và cuối có thể trùng nhau.

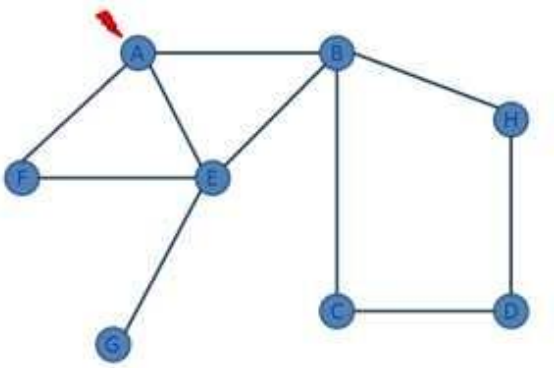
- Chu trình là đường đi có đỉnh đầu trùng với đỉnh cuối. Chu trình đơn là một đường đi đơn có đỉnh đầu và đỉnh cuối trùng nhau và có độ dài ít nhất là 1.
- Cấp của đồ thị là số đỉnh của đồ thị. Kích thước của đồ thị là số cạnh của đồ thị. Đồ thị rỗng là đồ thị có kích thước bằng 0
- Bậc của một đỉnh là số cạnh nối với đỉnh đó
- Trong nhiều ứng dụng, ta thường kết hợp các giá trị (value) hoặc nhãn (label) cho các cạnh hoặc đỉnh. Khi đó, ta gọi là đồ thị có trọng số.
- Nhãn có thể có kiểu tùy ý và dùng để biểu diễn tên, chi phí, khoảng cách, …
- Đồ thị có hướng là đồ thị mà các cạnh của nó là có hướng. Nghĩa là các cặp đỉnh (v,w) có phân biệt thứ tự.
- Đồ thị vô hướng là đồ thị mà các cặp đỉnh tương ứng với các cạnh của nó không phân biệt thứ tự.

2. *Biểu diễn đồ thị*

Biểu diễn bằng danh sách kề



3. *Các phương pháp duyệt đồ thị*



Kết quả: A B C D H E F G

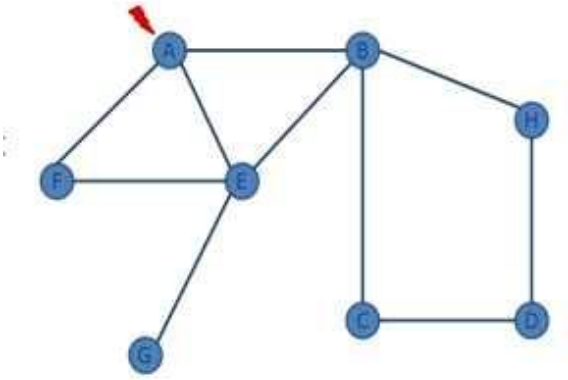
Duyệt đồ thị là một thủ tục có hệ thống để khám phá đồ thị bằng cách kiểm tra tất cả các đỉnh và các cạnh của nó.

Có hai cách duyệt đồ thị phổ biến

- Duyệt đồ thị theo chiều sâu (Depth First Search - DFS)
- Duyệt đồ thị theo chiều rộng (Breadth First Search -BFS)

Cách 1: Duyệt đồ thị theo chiều sâu

- B1. Khởi gán tất cả các đỉnh đều chưa được duyệt
- B2. Xuất phát từ một đỉnh **v** bất kỳ của đồ thị, đánh dấu đỉnh v đã được duyệt
- B3. Xử lý đỉnh v
- B4. Với mỗi đỉnh w chưa được duyệt và kề với v, ta thực hiện đệ quy quá trình trên cho w.



Kết quả: A B E F C H G D

Cách 2: Duyệt đồ thị theo chiều rộng

- B1. Khởi gán tất cả các đỉnh đều chưa được duyệt
- B2. Xuất phát từ một đỉnh **v** bất kỳ của đồ thị, đánh dấu đỉnh v đã được duyệt. Đưa v vào hàng đợi.
- B3. Lấy x ra khỏi hàng đợi. Xử lý đỉnh x.
- B4. Với mỗi đỉnh w chưa được duyệt và kề với x, ta đánh dấu w đã được duyệt. Đưa w vào hàng đợi.
- B5. Quay lại B3.

IV. Hướng dẫn thực hành

1. *Tạo dự án*

Sinh viên có thể chọn cài đặt kiểu dữ liệu đồ thị theo ngôn ngữ C++.

- Tạo dự án mới, đặt tên là Lab5_GraphAsList
- Trong thư mục Header Files, tạo ra các files sau:
  - menu.h: Định nghĩa các thao tác thực thi của chương trình
  - common.h: Định nghĩa các hằng số và kiểu dữ liệu đồ thị
  - graph.h: Định nghĩa các thao tác trên đồ thị
- Trong thư mục Source Files, tạo tập tin: program.cpp

2. *Định nghĩa kiểu dữ liệu đồ thị (Biểu diễn bằng danh sách kề)*

- Nhấp đôi chuột vào file common.h, định nghĩa các hằng số và kiểu dữ liệu như sau:

```
#define TAB         '\t'   // Khoảng TAB
#define EOL         '\n'   // Xuống dòng
#define UPPER       100    // Số ptu tối đa
#define ZERO        0      // Giá trị 0
#define MAX         30     // Số đỉnh tối đa
#define INF         1000   // Vô cùng
```

```
#define YES         1      // Đã xét
#define NO          0      // Chưa xét
#define NULLDATA    -1     // Giả giá trị rỗng

// Định nghĩa các kiểu dữ liệu
typedef char LabelType;
typedef int CostType;

// Định nghĩa cấu trúc một cạnh
struct Edge
{
	int Marked; // Trạng thái
	char Target; // Đỉnh cuối
	CostType Weight; // Trọng số
	Edge* Next; // Cạnh tiếp
};

// Định nghĩa cấu trúc của một đỉnh
struct Vertex
{
	LabelType Label; // Nhãn của đỉnh
	int Visited; // Trạng thái
	Edge* EdgeList; // DS cạnh kề
};

struct Path // Một đoạn đường đi
{
	CostType Length; // Độ dài đđi
	int Parent; // Đỉnh trước
};
typedef Path* PathPtr;

typedef Edge* EdgePtr;
typedef Vertex* VertexPtr;

// Định nghĩa kiểu dữ liệu đồ thị
struct GraphADT
{
	bool Directed; // ĐT có hướng?
	int NumVertices; // Số đỉnh
	int NumEdges; // Số cạnh
	VertexPtr Vertices[MAX]; // DS cac đỉnh kề
};
```

- Trong tập tin program.cpp, nhập đoạn mã sau:

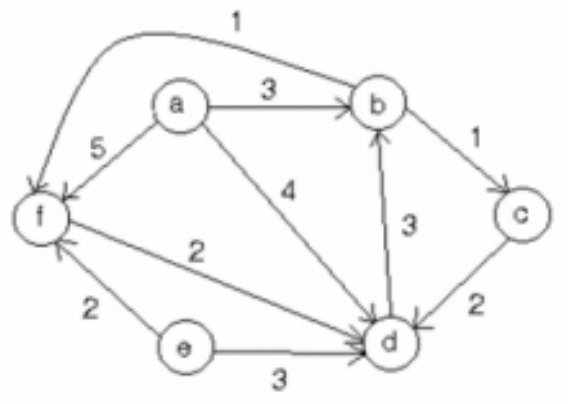
```
#include <iostream>
#include <conio.h>
#include <fstream>
#include <stack> // Dùng cho Stack trong thư viện có sẵn
#include <queue> // Dùng cho Queue trong thư viện có sẵn

using namespace std;

#include "common.h"
#include "graph.h"

void main()
{
	_getch();
}
```

Vào menu Build > chọn Build Solution để biên dịch chương trình, kiểm tra lỗi. Nếu có lỗi, kiểm tra lại mã nguồn bạn đã viết. Nếu chương trình không có lỗi, ta sẽ cài đặt các thao tác, phép toán trên đồ thị.

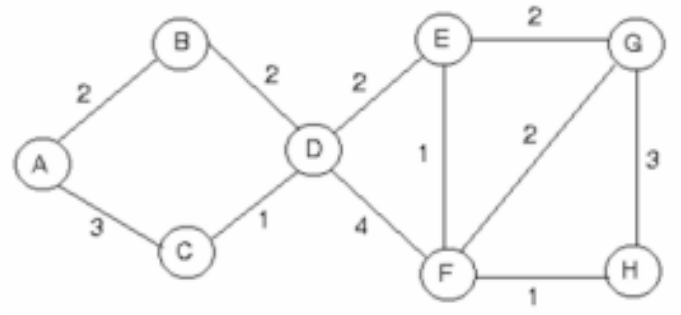
V. Bài tập thực hành (có HD)

Bài 1. Với mỗi đồ thị sau, hãy tạo các tập tin (sử dụng chương trình notepad, wordpad, …) lưu trữ chúng theo định dạng:

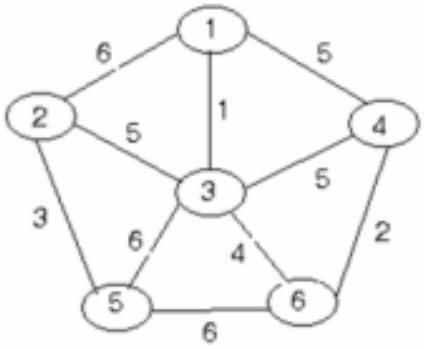
- Dòng đầu: Ghi số N (số đỉnh của đồ thị)
- Dòng thứ 2: Ghi số M (số cạnh của đồ thị)
- Dòng thứ 3: Ghi số 0 (nếu đồ thị vô hướng) hoặc 1 (nếu đồ thị có hướng)
- N dòng kế tiếp, mỗi dòng ghi nhãn của 1 đỉnh
- N dòng tiếp theo, mỗi dòng ghi N giá trị biểu diễn cho ma trận kề hay ma trận trọng số (gồm N dòng, N cột). Quy ước, nếu giữa hai đỉnh v và w có cạnh nối thì ghi giá trị trọng số của cạnh đó. Nếu không có cạnh nối thì ghi 0 (nếu v = w) hoặc 1000 (=INF nếu v khác w).



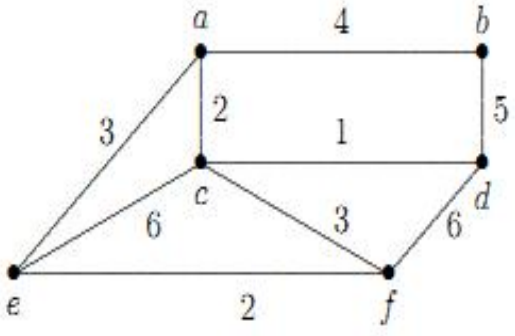
a)



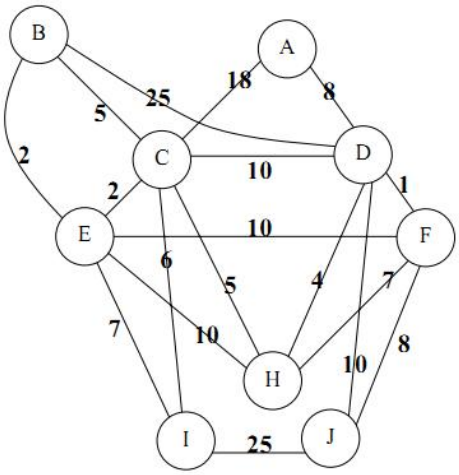
b)



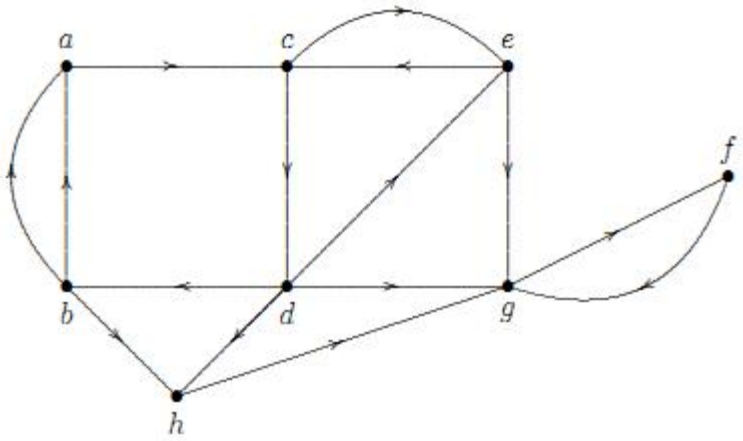
c)



d)



e)



f)

Bài 2. Cài đặt các thao tác cơ bản trên kiểu dữ liệu đồ thị – Viết hàm trong tập tin graph.h (C++)

a. Tạo một đỉnh có nhãn lab

```
// Tạo một đỉnh có nhãn lab
VertexPtr CreateVertex(LabelType lab)
{
	Khai báo và khởi tạo một đỉnh v có kiểu VertexPtr
	Gán nhãn lab cho đỉnh v
	Gán đỉnh v chưa xét
	Gán danh sách đỉnh liên kết từ v = NULL
	Trả về đỉnh v
}


// Tạo một cạnh chỉ đến đỉnh có nhãn label, trọng số w
EdgePtr CreateEdge(char lab, CostType w)
{
	Khai báo và khởi tạo biến động e có kiểu EdgePtr
	Lưu nhãn đỉnh đích lab vào e
	Lưu trọng số w vào e
	Đánh dấu e chưa xét
	Gán liên kết kế tiếp từ e = NULL
	Trả về cạnh e
}
```

b. Hiển thị thông tin của một đỉnh

```
//Tìm vị trí đỉnh v trong đồ thị g
int FindIndexVertex(Graph g, char v)
{
	for (int i = 0; i < Số đỉnh của đồ thị g; i++)
	{
		if (Nếu đỉnh thứ i có nhãn = v)
			Trả về i;
	}
	Trả về -1;
}
// Xuất danh sách các cạnh của 1 đỉnh được trỏ bởi ds
void XuatDSCanh(EdgePtr ds)
{
	while (ds khác rỗng)
	{
		Xuất ra màn hình Target và Weight;
```

```
                Đến liên kết kế tiếp;
        }
}

// Hiển thị thông tin đỉnh v trong đồ thị g
void Xuat1Dinh(char v, Graph g)
{
        Khai báo biến s và dùng hàm FindIndexVertex để đến đỉnh v
        if (không tìm thấy v)
                cout << “Khong co dinh “ << v << ”trong do thi”;
        else
        {
                Xuất nhãn đỉnh của s;
                Xuất danh sách các cạnh của s
        }
}
```

c. *Hàm khởi tạo một đồ thị*

```
// Khởi tạo một đồ thị
// directed = true: Đồ thị có hướng
Graph InitGraph(bool directed)
{
        Khai báo và khởi tạo đồ thị g
        Khởi tạo số cạnh của g = 0
        Khởi tạo số đỉnh của g = 0
        Gán đồ thị có hướng hay không có hướng cho đồ thị g
        // Khởi tạo danh sách đỉnh
        for (int i = 0; i < Số đỉnh của đồ thị; i++)
        {
                g->Vertices[i] = NULL;
        }
        return g;
}
```

d. *Hàm thiết lập lại trạng thái của các đỉnh trong đồ thị*

```
// Thiết lập lại trạng thái của các đỉnh
void ResetFlags(Graph &g)
{
        for (int i = 0; i < Số đỉnh của đồ thị; i++)
                g->Vertices[i]->Visited = NO; // Thiết lập trường Visited của đỉnh i = NO
}
```

e. *Hàm kiểm tra 2 đỉnh có kề nhau hay không (2 đỉnh được nối với nhau bởi ít nhất một cạnh)*

```
//Tìm vị trí đỉnh
int FindIndexVertex(Graph g, char vertex)
{
        for (int i = 0; i < Số đỉnh của đồ thị; i++)
        {
                if (nhãn của đỉnh thứ I = vertex)
                        return i;
        }
        return -1;
}

// Tìm cạnh nối 2 đỉnh start và end
EdgePtr FindEdge(Graph g, char start, char end)
{
        // Bắt đầu tìm từ cạnh kề đầu tiên với start
        Tìm đỉnh đầu start bằng hàm FindIndexVertex
        Khai báo và gán danh sách cạnh
        // Tìm cạnh với đỉnh cuối là end
        while (danh sách cạnh khác NULL) // Duyệt qua các cạnh kề
```

```
	{
		if (nhãn của danh sách cạnh = end)
			break; // Nếu tìm thấy thì dừng.
		Đi đến cạnh kế tiếp; // Nếu ko thì sang cạnh tiếp
	}
	return e;//trả về null
}

// Kiểm tra hai đỉnh u, v có kề nhau (có cạnh nối giữa chúng) hay không?
int IsConnected(Graph g, int start, int end)
{
	Khai báo biến e có kiểu EdgePtr;
	Gán biến e= FindEdge(g, start, end);
	return (e != NULL);//true false
}
```

f. Hàm thêm một đỉnh có nhãn lab vào đồ thị

```
//Thêm đỉnh v vào đồ thị g
void AddVertex(Graph &g, VertexPtr v)
{
	g->Vertices[g->NumVertices] = v;//thêm vào cuối
	g->NumVertices++;//tăng số đỉnh
}
```

g. Hàm thêm một cạnh nối giữa hai đỉnh của đồ thị

```
// Thêm một cạnh e vào cạnh v (nối vào danh sách v)
void AddEdge(EdgePtr &v, EdgePtr e)
{

	if (v == NULL)
		v = e;
	else
	{
		AddEdge(v->Next, e);

	}
}
// Thêm một cạnh chỉ đến đỉnh có nhãn lab với trọng số w vào danh sách của đỉnh v

void AddEdge(Graph &g, char v, char lab, CostType w)
{

	if (v == NULL)
		v = lab;
	else
	{

		if (g->Directed == false)//Nếu không có hướng
		{
			EdgePtr h, k;
			h = CreateEdge(lab, w);
			k = CreateEdge(v, w);
			AddEdge(g->Vertices[FindIndexVertex(g, v)]->EdgeList, h);
			AddEdge(g->Vertices[FindIndexVertex(g, lab)]->EdgeList, k);
			g->NumEdges = g->NumEdges + 2; //Do thêm 2 cạnh
		}
		else//neu co huong
		{
			EdgePtr a;
			a = CreateEdge(lab, w);
			AddEdge(g->Vertices[FindIndexVertex(g, v)]->EdgeList, a);
			g->NumEdges = g->NumEdges + 1;
		}
```

```
	}
}
```

h. Hàm lưu đồ thị xuống file

```
// Lưu đồ thị xuống file
void WriteGraph(Graph g, char* filename)
{
	ofstream out(filename); // Khai báo biến và mở tập tin để ghi
	out << g->NumVertices << '\n'; // Lưu số đỉnh
	out << g->NumEdges << '\n'; // Lưu số cạnh
	out << g->Directed << '\n'; // Lưu loại đồ thị
	// Lưu tên các đỉnh
	for (int i = 0; i < g->NumVertices; i++)
		out << g->Vertices[i]->Label << '\n';
	// Lưu ma trận kề
	for (int i = 0; i < g->NumVertices; i++)
	{
		for (int j = 0; j < g->NumVertices; j++)
		{
			if (i == j)
				out << "0" << "\t";
			else
				if (IsConnected_2(g, g->Vertices[i]->Label, g->Vertices[j]-
>Label))
				{
					EdgePtr temp = FindEdge(g, g->Vertices[i]->Label, g-
>Vertices[j]->Label);
					out << g->Vertices[j]->EdgeList->Weight << TAB;
				}
				else
				{
					out << "1000" << TAB;
				}
		}
		if (i != g->NumVertices - 1)
			out << "\n";
	}
	out.close();// Đóng tập tin
}
```

i. Hàm tạo đồ thị từ dữ liệu được lưu trong file

```
// Đọc dữ liệu từ file để tạo đồ thị
int OpenGraph(Graph &g, char* filename)
{
	ifstream in(filename);
	if (in.is_open())
	{
		int n = 0, m = 0, w = 0;
		bool d = false;
		char lab;
		in >> n;	//doc so dinh
		in >> m;	//doc so canh
		in >> d;	//doc loai do thi
		g = InitGraph(d);
		for (int i = 0; i < n; i++)
		{
			in >> lab;
			VertexPtr v = CreateVertex(lab);
			AddVertex(g, v);
		}
		lab = ' ';
		int indexVertex;
```

```
		for (int i = 0; i < m; i++)
		{
			char dinh, dich;
			int trongSo;
			in >> dinh;
			in >> dich;
			in >> trongSo;
			if (lab != dinh)
			{
				lab = dinh;
				indexVertex = FindIndexVertex(g, dinh);
			}

			if (trongSo == NULL)
			{
				trongSo = 1;
			}
			EdgePtr e = CreateEdge(dich, trongSo);
			AddEdge(g->Vertices[indexVertex]->EdgeList, e);
			g->NumEdges++;
		}
		in.close();
		return 1;
	}
	else
	{
		cout << "\nLoi doc file nhap lai ten file\n";
		system("pause");
		return 0;
	}
}
```

j. Hàm tìm đỉnh đầu tiên chưa được xét và kề với đỉnh hiện tại (curr)

```
// Tìm đỉnh đầu tiên kề với cur mà chưa xét
char FindFirstAdjacentVertex(Graph g, char cur)
{
	// Duyệt qua mọi đỉnh
	for (int i = 0; i < g->NumVertices; i++)
	{
		if (g->Vertices[i]->Label == cur)
		{
			EdgePtr p = g->Vertices[i]->EdgeList;
			while (p != NULL)
			{
				for (int j = 0; j < g->NumVertices; j++)
					if (g->Vertices[j]->Label == p->Target&&g->Vertices[j]-
>Visited == NO)
						return p->Target; // Thỏa đk -> tìm thấy
				p = p->Next;
			}
		}
	}
	return NULLDATA; // Không tìm thấy

}
```

Bài 3. Các phương pháp duyệt đồ thị.

a. Hàm duyệt đồ thị theo chiều sâu (dạng lặp)

```
// Duyệt đồ thị theo chiều sâu (Depth First Search)
// Dạng lặp, sử dụng Stack
void DFS(Graph g, char start)
{
	int index = FindIndexVertex(g, start);
```

```
        g->Vertices[index]->Visited = YES;
        cout << start << TAB;
        stack<char> st;
        st.push(start);
        char cur, adj;
        while (!st.empty())
        {
                cur = st.top();
                adj = TimDinhDauTien(g, cur);
                if (adj == NULLDATA)
                        st.pop();
                else
                {
                        int index = FindIndexVertex(g, adj);
                        g->Vertices[index]->Visited = YES;
                        cout << adj << TAB;
                        st.push(adj);
                }
        }
}
```

b. Hàm duyệt đồ thị theo chiều rộng

```
// Duyệt đồ thị theo chiều rộng
// Dạng lặp, sử dụng Queue

void BFS(Graph g, char start)
{
        int index = FindIndexVertex(g, start);
        g->Vertices[index]->Visited = YES;
        queue<char> que;
        que.push(start);
        char cur, adj;
        cur = que.front();
        cout << cur << TAB;
        while (!que.empty())
        {
                adj = TimDinhDauTien(g, cur);
                if (adj == NULLDATA)
                {
                        que.pop();
                        if (que.size() != 0)
                                cur = que.front();
                        else
                                break;
                }
                else
                {
                        index = FindIndexVertex(g, adj);
                        g->Vertices[index]->Visited = YES;
                        cout << adj << TAB;
                        que.push(adj);
                }
        }
        ResetFlags(g);
}
```

Bài 4. Kiểm tra chương trình và xem kết quả: Hoàn thiện hàm main và viết hàm trong tập tin menu.h (C++)

VI. Bài tập

Bài 1. Đồ thị biểu diễn bằng danh sách cạnh

- Tìm hiểu cách biểu diễn đồ thị bằng danh sách cạnh (Soạn slide thuyết trình)
- Cài đặt chương trình theo yêu cầu tương tự như V ở trên.

Bài 2. Đồ thị biểu diễn bằng ma trận liên thuộc

- Tìm hiểu cách biểu diễn đồ thị bằng ma trận liên thuộc (Soạn slide thuyết trình)
- Cài đặt chương trình theo yêu cầu tương tự như V ở trên.

=== HẾT ===