**TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**-----***-----**

# BÁO CÁO BÀI TẬP LỚN

# MÔN HỌC "CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT"

***Đề tài:***

***Ứng dụng đồ thị dưới dạng ma trận kề, để tìm đường đi ngắn nhất giữa hai địa điểm bằng giải thuật BFS (Breadth-first Search).***

***GV:*** ***TS. Lê Xuân Trường***

***Sinh viên thực hiện:*** ***Lê Minh Tính - 2154050301***

## 1. Mục đích

Xây dựng chương trình nhằm thu thập dữ liệu các địa điểm trong một khu vực nhất định, sau đó giải quyết vấn đề tìm một đường đi ngắn nhất giữa hai địa điểm trên bản đồ đã thu thập.

## 2. Công việc cần thực hiện

- Xây dựng hệ thống ngôn ngữ: Tiếng Anh, Tiếng Việt và menu trực quan, thuật giải tối ưu trong chương trình.
- Xây dựng hàm đọc ngôn ngữ từ tập tin.
- Xây dựng hàm đọc đồ thị hoặc nhập đồ thị để thu thập thông tin đồ thị theo yêu cầu.
- Xây dựng hàng đợi Queue thông qua danh sách liên kết đơn .
- Xây dựng giải thuật duyệt theo chiều rộng BFS (Breadth-first search) thông qua hàng đợi.
- Xây dựng giải thuật tìm đường đi ngắn nhất dựa trên giải thuật tìm kiếm theo chiều rộng BFS.
- Làm báo cáo bài tập lớn.
- Bảo vệ bài tập lớn.

## 3. Yêu cầu

- Kết quả làm bài tập lớn: Báo cáo bài tập lớn.
- Báo cáo bài tập lớn phải được trình bày theo mẫu quy định, báo cáo được kết xuất thành tệp định dạng PDF và nộp qua email.
- Hạn nộp báo cáo bài tập lớn: Trước thi giữa kì.

**MỤC LỤC**

# DANH MỤC CÁC TỪ VIẾT TẮT

| Từ | Ý nghĩa |
|---|---|
| BFS | Breadth-first search |

# MỞ ĐẦU

***Bài toán:***

Quản lý các địa điểm của một thành phố bằng giải thuật BFS với đồ thị được nhập, giải thuật tìm kiếm đường đi ngắn nhất tương ứng.

***Yêu cầu:***

- Xây dựng đồ thị có các đỉnh:
  - Hàm khởi tạo.
  - Hàm nhập (đọc).
  - Hàm xuất .
- Xây dựng giải thuật BFS - Tìm kiếm đường đi gần nhất:
  - Hàm khởi tạo .
  - Hàm duyệt BFS.
  - Hàm tìm đường đi ngắn nhất.
  - Hàm xuất.
  - Các phương thức bổ sung: Chọn kiểu ngôn ngữ; Kiểu nhập đồ thị từ tập tin hoặc từ bàn phím; In thông tin địa điểm, đồ thị.

# 1. Ý TƯỞNG

- Đầu vào là 1 đồ thị dưới dạng ma trận vuông (NxN) gồm các đỉnh tương ứng với các địa điểm trên bản đồ.
- Tiến hành lưu các địa điểm này theo thứ tự các đỉnh được sắp xếp trong bản đồ.
- Tiếp theo dùng phép duyệt theo chiều rộng để duyệt tất cả các địa điểm với địa điểm bắt đầu (địa điểm hiện tại của người dùng) và duyệt đến địa điểm cuối cùng được xếp ở bản đồ.
- Tuy nhiên sau khi duyệt ta vẫn chưa linh động được địa điểm kết thúc mà người dùng muốn. Ta tiến hành lưu lại các địa điểm sau mỗi lần duyệt và với địa điểm bắt đầu và địa điểm kết thúc thay đổi được.

- Cuối cùng ta tìm được đường đi ngắn nhất từ địa điểm đầu đến địa điểm kết thúc trên bản đó, thoả mãn yêu cầu của bài toán.

# 2. CÀI ĐẶT ĐỒ THỊ

## 2.1. Ma trận kề

- Cho đồ thị G = (V, E) vô hướng không có trọng số, ta đánh các số các đỉnh của đồ thị bằng một số tự nhiên: 1, 2, …., n.
- Xây dựng ma trận vuông biểu diễn đồ thị: Ma trận vuông nxn được gọi là ma trận kề của G sao cho A[i][j] bằng 1 nếu i kề j và bằng 0 nếu i không kề j.

## 2.2. Khởi tạo đồ thị

### 2.2.1. Mục đích

- Mục đích của hàm khởi tạo dùng để khởi tạo dữ liệu cho đồ thị.

### 2.2.2. Hàm xử lý

```
void InitG() {
   n = 0;
}
```

## 2.3. Nhập đồ thị thủ công

### 2.3.1. Mục đích

- Nhập đồ thị theo yêu cầu của người dùng, thay thế nhiều kiểu đồ thị khác nhau thông qua ma trận dưới dạng nhị phân được biểu diễn trên máy tính.

### 2.3.2. Hàm xử lý

```
void inputGraph()
```

```
{
    cout << verte;
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        cout << nameverte;
        cin.ignore();
        getline(cin, vertex[i]);
        for (int j = 0; j < n; j++){
            cout << "A["<<i<<"]["<<j<<"] = ";
            cin >> A[i][j];
            do{
                if(A[i][j] < 0 || A[i][j] > 1){
                    cout << againIn << endl;
                    cout << "A["<<i<<"]["<<j<<"] = ";
                    cin >> A[i][j];
                }
            }while(A[i][j] < 0 || A[i][j] > 1);
        }
    }
}
```

## 2.4. Đọc đồ thị từ tập tin

### 2.4.1. Mục đích

- Trường hợp chuẩn bị 1 đồ thị có nhiều đỉnh mà người dùng khó nhập được từ bàn phím, chúng ta sẽ đọc vào một đồ thị từ người dùng dưới dạng ma trận với nhiều đỉnh đáp ứng nhu cầu.
- Thay vì viết trực tiếp vào chương trình, chúng ta áp dụng lại kiến thức đọc tập tin để những dòng lệnh thêm trực quan, gọn gàng hơn.

### 2.4.2. Hàm xử lý

```
void inputGraphFile()
{
    ifstream f;
    f.open("Graph-MapTPHCM.txt");
    if(f.fail()){
        return;
    }
    else
    {
        f >> n;
        f.ignore();

        for (int i = 0; i < n; i++)
        {
            if(i==n-1) getline(f, vertex[i]);
            else getline(f, vertex[i], ',');

        }

        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
                f >> A[i][j];
        }
    }
    f.close();
}
```

# 3. GIẢI THUẬT BFS

## 3.1. Xây dựng hàng đợi (Queue)

- Hàng đợi (Queue) là danh sách chứa các phần tử được quản lý theo thứ tự sau: Phần tử được thêm vào trước, sẽ được lấy ra trước.

**Các thao tác:**

- init: khởi tạo Queue rỗng.
- Push: cho phần tử vào Queue.
- Pop: lấy phần tử từ trong Queue.
- isEmpty: kiểm tra Queue có rỗng không.

## 3.2. Duyệt đồ thị theo chiều rộng (BFS)

- Duyệt trên tất cả các đỉnh của một mức trong không gian bài toán trước khi chuyển sang các đỉnh ở mức tiếp theo.

### 3.2.1. Giải thuật

- Bước 1: Giả sử v là đỉnh bắt đầu, cho v vào trong Q (Queue).
- Bước 2: Nếu Q khác rỗng, gọi u là phần tử được lấy từ Q, và xuất u ra màn hình.
- Bước 3: Tìm các đỉnh v kề với u, cho v vào trong Q quay lại bước 1.

### 3.2.2. Hàm xử lý

```
int C[100], bfs[100];
int nbfs = 0;
void Init()
{
   for (int i = 0; i < n; i++)
      C[i] = 1;
}
```

```
void BFS(int v)
{
   int p, w;
   PushQ(v);
   C[v] = 0;
   while(!isEmptyQ())
   {
      PopQ(p);
      bfs[nbfs] = p;
      nbfs++;
      for ( w = 0; w < n; w++)
         if (C[w] && A[p][w] == 1)
         {
            PushQ(w);
            C[w] = 0;
         }
   }
}
```

## 4. ĐƯỜNG ĐI NGẮN NHẤT (BFS)

### 4.1. Giới thiệu

- Tìm đường đi ngắn nhất của các đỉnh trên đồ thị cũng là mấu chốt của bài toán tìm đường đi ngắn nhất giữa 2 địa điểm trên bản đồ của 1 thành phố.
- Bài toán này được ứng dụng dựa vào phần đồ thị, ở đây chúng ta sử dụng giải thuật BFS để giải quyết bài toán.

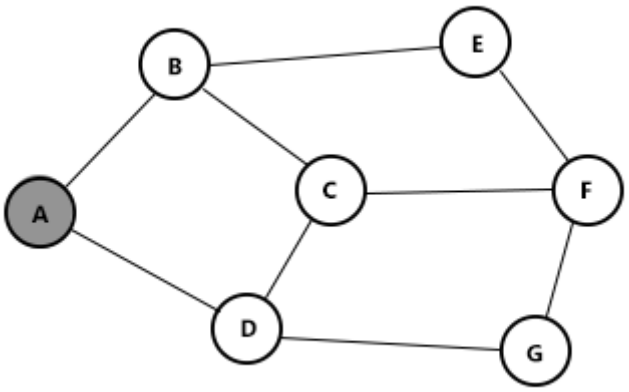
## 4.2. Ý tưởng

- Tìm các đỉnh bằng cách duyệt theo chiều rộng BFS.
- Từ đỉnh xuất phát đi tới đỉnh kề của đỉnh xuất phát này, Ta tiến hành Lưu lại đỉnh cha của đỉnh kề đó, và tiếp tục cho đến khi gặp đỉnh kết thúc (không còn đỉnh nào).
- Sau đó đi ngược lại từ đỉnh kết thúc đến đỉnh cha của đỉnh kết thúc, và tiếp tục đến khi gặp đỉnh xuất phát. Ta tìm được đường đi ngắn nhất
- Nhờ cơ chế bôi đen và lưu đỉnh cha ta tìm được đường đi ngắn nhất. Khi bôi đen khiến 1 đỉnh chỉ xét được 1 lần và có thể xem được đường đi từ đỉnh Kết thúc đến đỉnh xuất phát dựa vào việc lưu đỉnh cha.

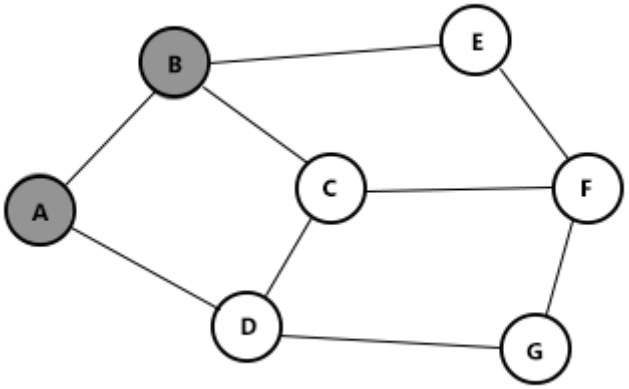
## 4.2. Mô tả ý tưởng

### 4.2.1. Hình ảnh minh hoạ

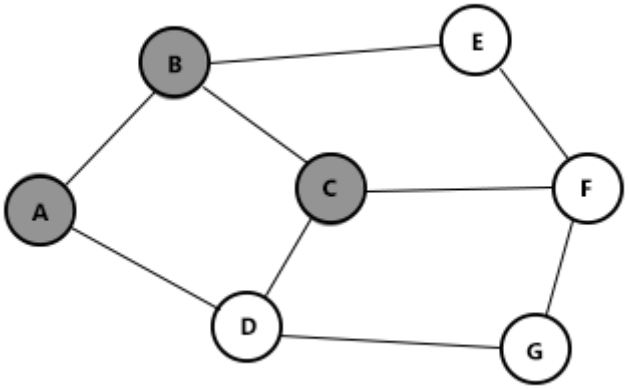
- Hình 1: Xuất phát từ đỉnh A



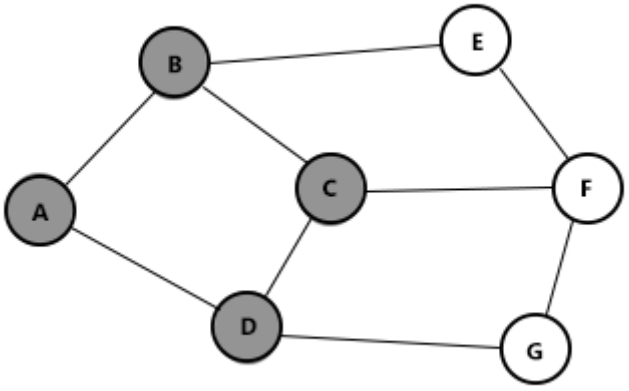

- Hình 2: Chọn đỉnh B, nút cha của đỉnh B là đỉnh A



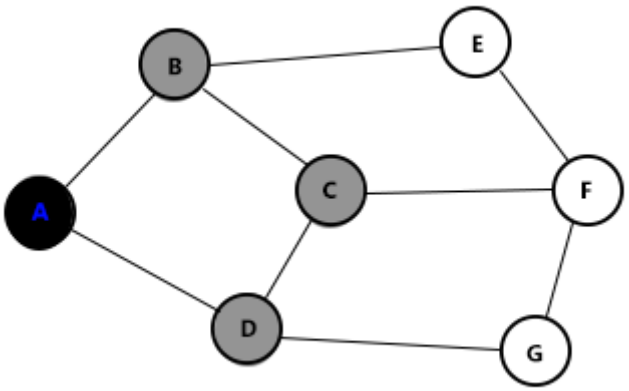

- Hình 3: Chọn đỉnh C, nút cha của đỉnh C là đỉnh A



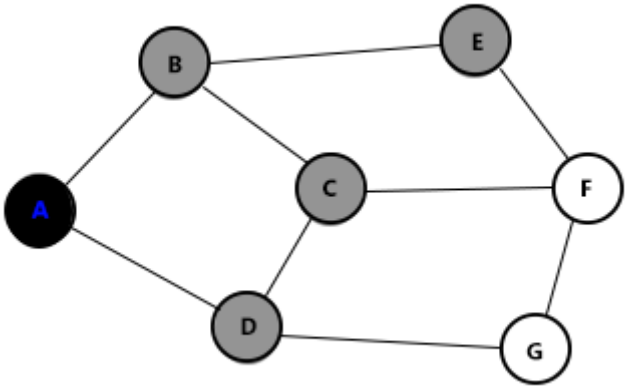

- Hình 4: Chọn đỉnh D, nút cha của đỉnh D là đỉnh A



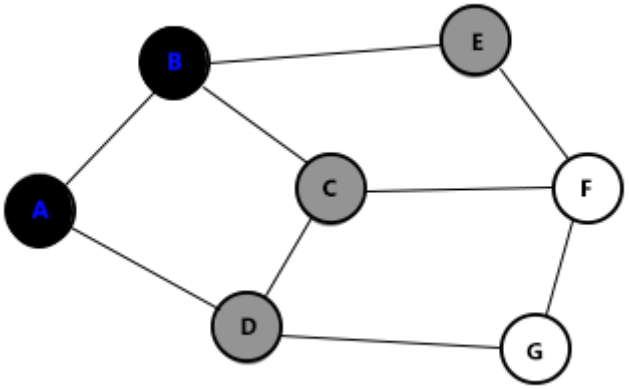
- Hình 5: Đã đi hết tất cả các đỉnh kề của đỉnh A, tiến hành bôi đen đỉnh A



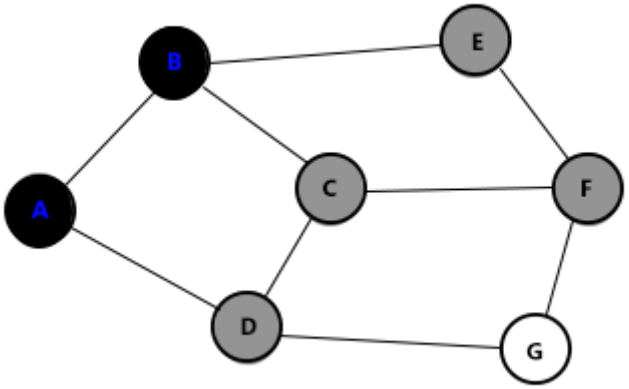

- Hình 6: Xuất phát từ đỉnh B, chọn đỉnh E, nút cha của đỉnh E là B



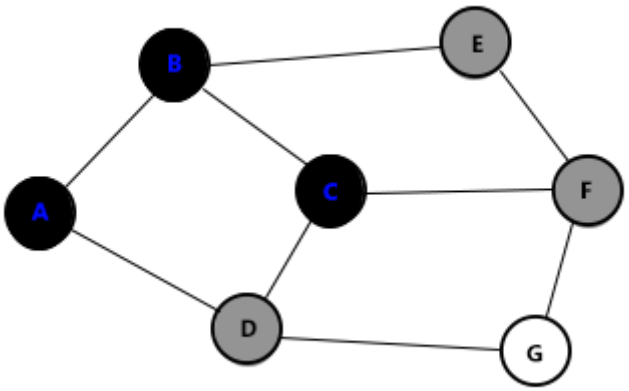

- Hình 7: Đã đi hết tất cả các đỉnh kề của đỉnh B, tiến hành bôi đen đỉnh B



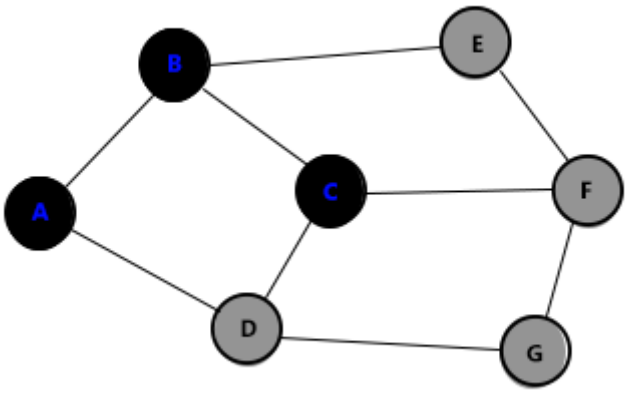

- Hình 8: Xuất phát từ đỉnh C, chọn đỉnh F, nút cha của đỉnh F là C



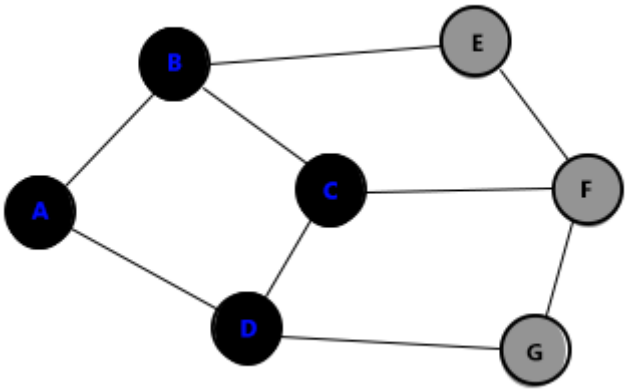

- Hình 9: Đã đi hết tất cả các đỉnh kề của đỉnh C, tiến hành bôi đen đỉnh C



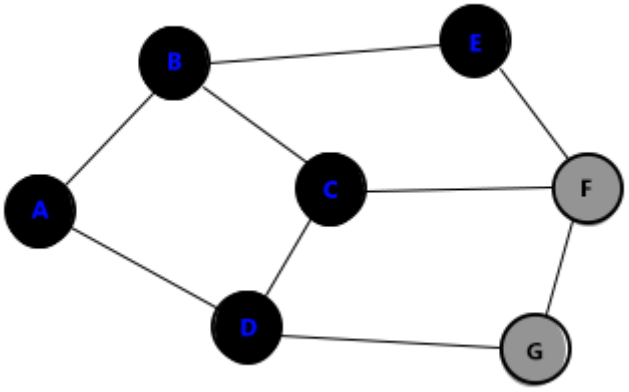
- Hình 10: Xuất phát từ đỉnh D, chọn đỉnh G, nút cha của đỉnh G là D



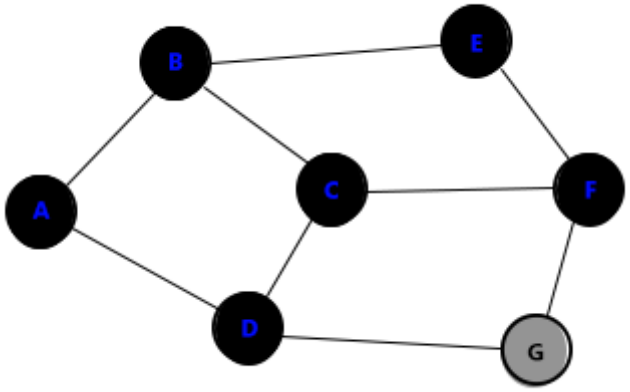

- Hình 11: Đã đi hết tất cả các đỉnh kề của đỉnh D, tiến hành bôi đen đỉnh D




- Hình 12: Đã đi hết tất cả các đỉnh kề của đỉnh E, tiến hành bôi đen đỉnh E

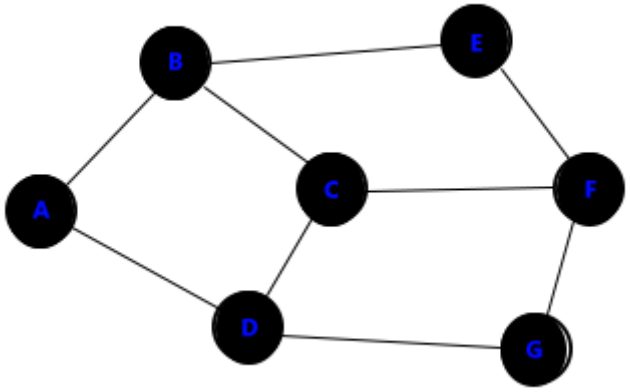



- Hình 13: Đã đi hết tất cả các đỉnh kề của đỉnh F, tiến hành bôi đen đỉnh F




N

- Hình 14: Đã đi hết tất cả các đỉnh kề của đỉnh G, tiến hành bôi đen đỉnh G




### 4.2.2. Kết luận mô tả

- Chúng ta vừa đi qua hết tất cả các đỉnh trong đồ thị, và mỗi lần đi đến đỉnh mới, ta đều lưu lại đỉnh cha của đỉnh mới.
- Dựa vào những đỉnh cha này, ta tìm được đường đi ngắn nhất bằng cách đi ngược từ đỉnh Kết thúc, đến đỉnh cha của đỉnh Kết thúc, rồi đến đỉnh cha của đỉnh tiếp theo, tiếp tục như thế đến đỉnh Bắt đầu.

## 4.3. Giải thuật

- Bước 1: Giả sử v là đỉnh bắt đầu, tiến hành khởi tạo giá trị cho các phần tử bằng 1 vòng lặp để bắt đầu mảng lưu các đỉnh cha.
- Bước 2: Thêm v vào hàng đợi Q (Queue).
- Bước 3: Nếu Q không rỗng, lấy p trong hàng đợi Q ra.
- Bước 4: Tiến hành tìm đỉnh kề chưa xét.
    - Gọi w là các đỉnh kề của p chưa có trong tập bfs[].
    - Cho p vào trong tập bfs[w] để lưu đỉnh cha của w.
    - Cho w vào hàng đợi Q.
- Bước 5: Cuối cùng gọi end là đỉnh kết thúc, tiến hành đệ quy v và end thực thi bằng mảng bfs đã lưu các đỉnh cha để tìm đường đi ngắn nhất.

## 4.4. Cài đặt giải thuật

- Bước 1: Khởi tạo mảng chứa đỉnh chưa xét và mảng lưu các đỉnh cha.

```
void Init2()
{
   for (int i = 0; i < n; i++)
   {
      C[i] = 1;
      bfs[i] = -1; //mảng lưu đỉnh cha
   }
}
```

- Bước 2: Viết hàm xử lý đường đi ngắn nhất.

```
void FindNearRoad_BFS(int v)
{
   int p;
   PushQ(v);
   C[v] = 0;
   while(!isEmptyQ())
   {
      PopQ(p);
      //tìm đỉnh kề chưa xét
      for(int w=0; w<n; w++)
         if(C[w]== 1 && A[p][w] == 1)
         {
            PushQ(w);
            C[p] = 0;
            bfs[w] = p; //lưu đỉnh cha của w
         }
      C[p] = 2; //duyệt hết đỉnh kề của p
   }
```

```
}
```

- Bước 3: Tiến hành đệ quy để in các đỉnh đã lưu, tìm đường đi ngắn nhất.

```
void outputRoads(int v, int end)
{
    if(v == end)
        cout << end;
    else
        if(bfs[end] == -1)
            cout << nopath << endl;
        else
        {
            outputRoads(v, bfs[end]);
            cout << " -> " << end;
        }
}
```

## 5. HÌNH ẢNH CHƯƠNG TRÌNH SAU KHI HOÀN THÀNH

### 5.1. Các tập tin đi kèm

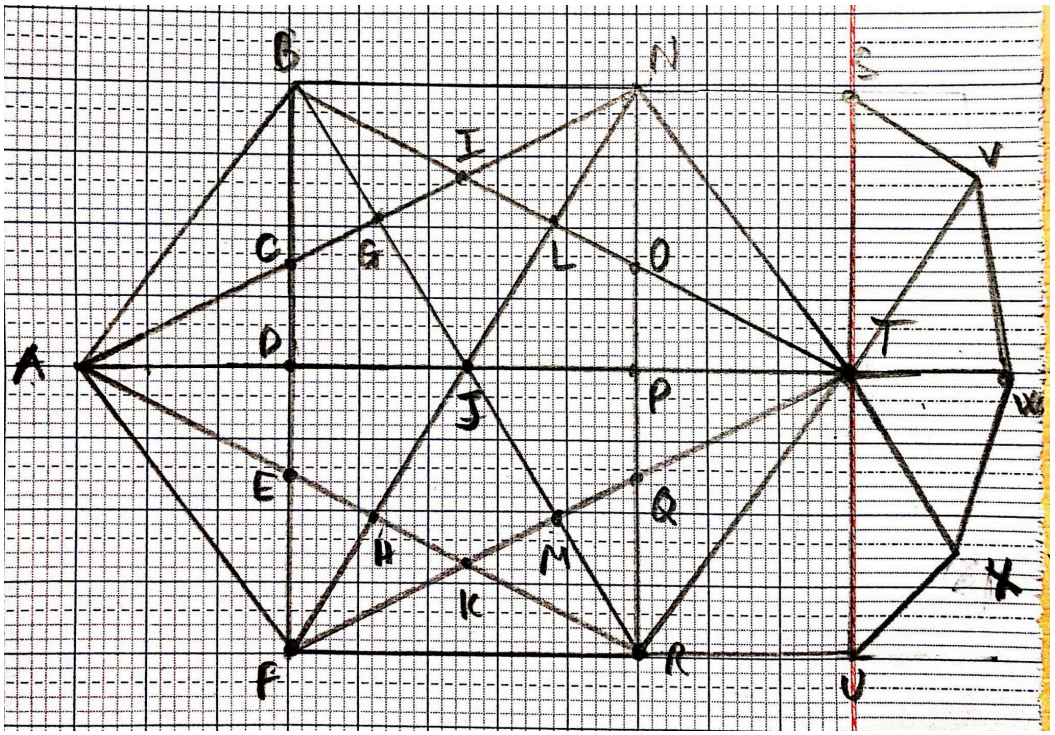
- Ngôn ngữ Tiếng Việt

```
Quay lai Menu hoac Thoat (1 | 0):
Tam biet ban nhe!
Khong co chuc nang!
Dang tai
Dang xoa
-> Thuc hien thanh cong <-
Nhap vi tri hien tai cua ban:
Nhap vi tri ban can den:
Duong dan giup ban di den
Nhap so dinh cua Do thi:
Nhap ten dinh:
Vui long nhap gia tri cua Do thi (0 hoac 1)!
Ban Do dang Rong! Vui long cap nhat lai Ban Do.
DANH SACH CAC DIA DIEM
Ghi chu: [0 - 23]: Vi tri cua cac Dia Diem
Khong co duong di!
```

- Ngôn ngữ Tiếng Anh

```
Back to Menu or Exit (1 | 0):
Goodbye to you!
No function!
Loading
Do Cleaning
-> Successful implementation <-
Enter your current location:
Enter the location you need to go to:
The path to get you to
Enter the number of vertices of the graph:
Enter vertex name:
Please enter the value in the graph (0 or 1)!
Map is empty! Please update the Map again.
LIST OF PLACES
Note: [0 - 23]: Location of Locations
No path!
```

- Đồ Thị và Biểu diễn đồ thị
  - *Ảnh đồ thị:*




*(Với 24 đỉnh A,B,C,D,.. tương ứng với 24 địa điểm trong bản đồ được bắt đầu từ đỉnh A ở vị trí 0)*

- *Ảnh biểu diễn đồ thị: Với hàng thứ nhất là tổng số địa điểm, hàng thứ 2 là các quận (huyện) ở TP.HCM, và phần còn là đồ thị.*

```
24
Thu Duc District,1 District,2 District,3 District,4 D
0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0
1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0
0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0
0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0
0 1 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0
0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 1 0 0 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1
```

## 5.2. Tìm đường đi ngắn nhất

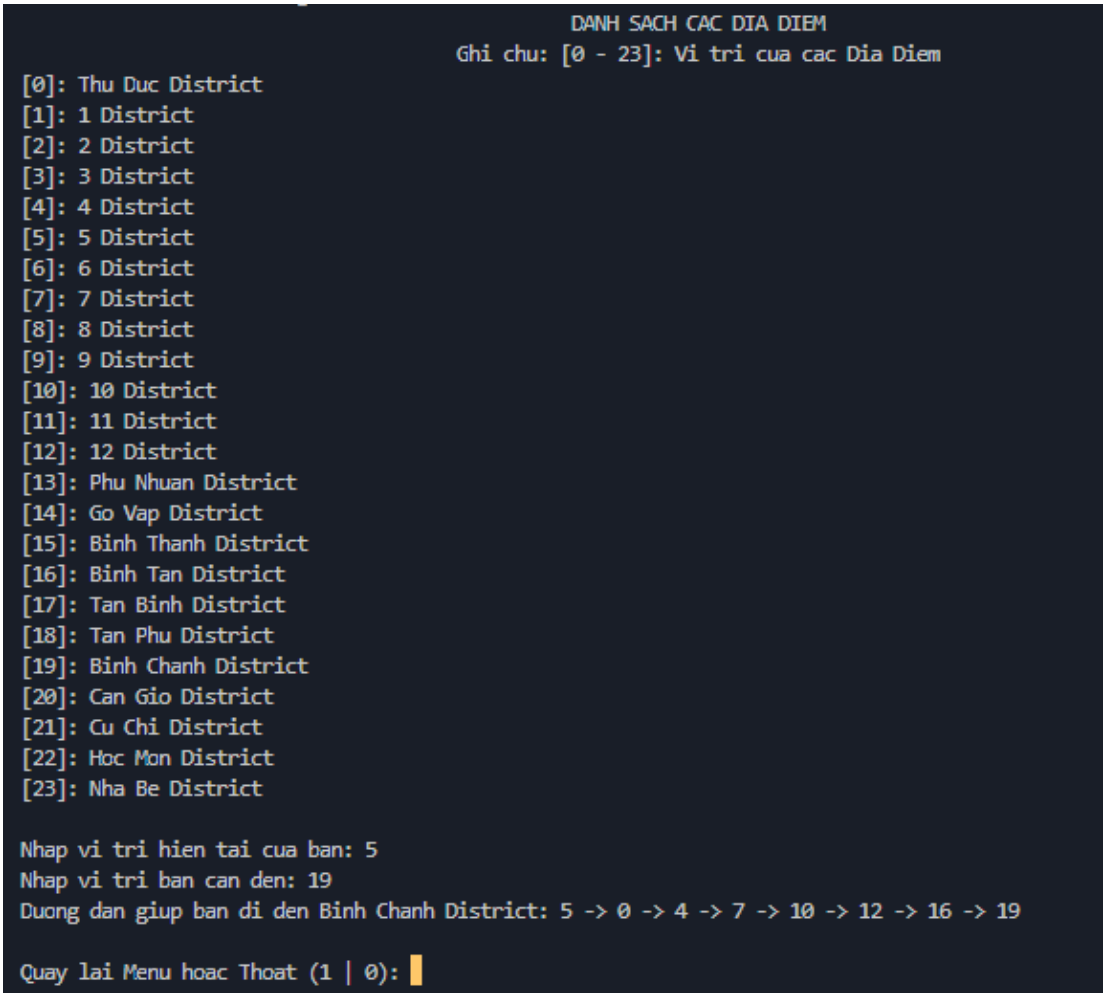
- Menu:

```
---------------------------------
|       NGON NGU (LANGUAGE)     |
---------------------------------
| E. Tieng Anh (English)        |
| V. Tieng Viet (Vietnamese)    |
---------------------------------
       Chon (Choose) E/V:
```

```
-----------------------------------------------------
|              TiM KIEM TRONG THANH PHO             |
|----------------- CAP NHAT | XOA -------------------|
| [1] | Cap nhat Ban Do tu dong (Tap Tin)           |
| [2] | Cap nhat Ban Do thu cong (Bieu dien Do thi) |
| [3] | Xoa tat ca vi tri tren Ban Do               |
|-------------------- DUYET | IN ---------------------|
| [a] | In cac Dia Diem duoc cap nhat tren Ban Do   |
| [b] | In cac Dia Diem Duyet theo chieu rong (BFS) |
| [c] | In Do thi cac dia diem tren Ban Do          |
|----------------- TIM KIEM | KHAC ------------------|
| [s] | Tim duong di ngan nhat giua 2 dia diem      |
| [x] | Doi ngon ngu                                |
| [0] | Thoat                                       |
-----------------------------------------------------
       -> Chon chuc nang:
```

```
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
| Ghi chu: - Cap nhat Ban Do truoc khi su dung nhe! |
|          - Su dung ban do mac dinh: [1]           |
|          - Thay doi Ban Do khac: [3] -> [2]       |
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

- Thực hiện tìm đường đi ngắn nhất:




## KẾT LUẬN

- Em đã giải quyết được những yêu cầu của bài tập lớn.

- Những điểm nào em chưa làm được hoặc làm chưa tối ưu theo yêu cầu của bài toán, mong thầy phản hồi để em khắc phục tốt hơn.