**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**

**ĐỒ ÁN LẬP TRÌNH TÍNH TOÁN**

**TÊN ĐỀ TÀI :**

**Ứng dụng tìm đường đi ngắn nhất giữa các khu trong trường Đại học Bách Khoa Đà Nẵng**

Người hướng dẫn**: PHẠM MINH TUẤN**

Nhóm sinh viên thực hiện**:**

**Phan Đức Thịnh LỚP: 19TCLC\_NHAT1**

**Nguyễn Minh Quang LỚP: 19TCLC\_NHAT1**

**Phạm Minh Tiến LỚP: 19TCLC\_NHAT1**

**Hồ Ngọc Hoàng Minh LỚP: 19TCLC\_NHAT1**

**Đà Nẵng, 12/2020**

MỤC LỤC

[MỤC LỤC i](#_Toc73175368)

[DANH MỤC HÌNH VẼ iii](#_Toc73175369)

[BẢNG PHÂN CÔNG NHIỆM VỤ v](#_Toc73175370)

[LỜI MỞ ĐẦU 1](#_Toc73175371)

[1. TỔNG QUAN ĐỀ TÀI 2](#_Toc73175372)

[1.1. Mô tả bài toán 2](#_Toc73175373)

[1.2. Các phần mềm, ngôn ngữ lập trình, framwork để thực hiện đề tài 2](#_Toc73175374)

[1.3. Thu thập và xử lý dữ liệu 2](#_Toc73175375)

[1.4. Mục tiêu cần đạt được: 4](#_Toc73175376)

[2. THIẾT KẾ CƠ SỞ DỮ LIỆU 4](#_Toc73175377)

[2.1. Phân tích cơ sở dữ liệu 4](#_Toc73175378)

[2.2. Thiết kế các bảng 5](#_Toc73175379)

[2.2.1. Bảng KHU 5](#_Toc73175380)

[2.2.2. Bảng PHONGHOC 5](#_Toc73175381)

[2.2.3. Bảng CAUTHANG 6](#_Toc73175382)

[2.2.4. Bảng ADDNODE 6](#_Toc73175383)

[2.3. Xây dựng mối liên hệ giữa các bảng 7](#_Toc73175384)

[2.4. Hình ảnh chi tiết các khu 8](#_Toc73175385)

[3. THIẾT KẾ THUẬT TOÁN 9](#_Toc73175386)

[3.1. Phát biển bài toán 9](#_Toc73175387)

[3.2. Phân tích thuật toán 11](#_Toc73175388)

[3.2.1. Kết nối các Node trong đồ thị 11](#_Toc73175389)

[3.2.2. Thuật toán Dijkstra 13](#_Toc73175390)

[4. CHƯƠNG TRÌNH VÀ KẾT QUẢ 16](#_Toc73175391)

[4.1. Tổ chức chương trình 16](#_Toc73175392)

[4.1.1. Mô Hình Three-Layer 16](#_Toc73175393)

[4.1.2. Các Class trong từng phần của mô hình 3 lớp 17](#_Toc73175394)

[4.2. Kết quả 19](#_Toc73175395)

[4.2.1. Giao diện chính của chương trình 19](#_Toc73175396)

[4.2.2. Kết quả thực thi của chương trình 20](#_Toc73175397)

[4.2.3. Nhận xét 24](#_Toc73175398)

[a. Kết luận 25](#_Toc73175399)

[b. Hướng phát triển 25](#_Toc73175400)

DANH MỤC HÌNH VẼ

[*Hình 1: Hình ảnh tổng quát của trường* 3](#_Toc73175295)

[*Hình 2: Hình ảnh minh họa Khu EM* 3](#_Toc73175296)

[*Hình 3: Hình ảnh chi tiết Khu EM* 4](#_Toc73175297)

[*Hình 4: Bảng KHU* 5](#_Toc73175298)

[*Hình 5: Bảng PHONGHOC* 5](#_Toc73175299)

[*Hình 6: Bảng CAUTHANG* 6](#_Toc73175300)

[*Hình 7: Bảng ADDNODE* 6](#_Toc73175301)

[*Hình 8: Liên hệ giữa các bảng* 7](#_Toc73175302)

[*Hình 9: Hình ảnh chi tiết khu BT* 8](#_Toc73175303)

[*Hình 10: Hình ảnh chi tiết khu CC* 8](#_Toc73175304)

[*Hình 11: Hình ảnh chi tiết khu F* 8](#_Toc73175305)

[*Hình 12: Hình ảnh chi tiết khu EC* 9](#_Toc73175306)

[*Hình 13: Hình ảnh chi tiết khu EM* 9](#_Toc73175307)

[*Hình 14: Hình ảnh chi tiết khu CM* 9](#_Toc73175308)

[*Hình 15: Hình ảnh mô tả output* 10](#_Toc73175309)

[*Hình 16: Hình minh họa kết nối các phòng học, cầu thang* 12](#_Toc73175310)

[*Hình 17: Đường đi ngắn nhất từ BT102 đến BT202* 13](#_Toc73175311)

[*Hình 18:Sơ đồ thuật toán kết nối các khu* 13](#_Toc73175312)

[*Hình 19: Sơ đồ thuật toán Dijkstra* 15](#_Toc73175313)

[*Hình 20: Các class trong mô hình 3 lớp* 17](#_Toc73175314)

[*Hình 21: Giao diện chính của chương trình* 19](#_Toc73175315)

[*Hình 22: Form AddNode của chương trình* 20](#_Toc73175316)

[*Hình 23: Form Add Edge của chương trình* 21](#_Toc73175317)

[*Hình 24: Form EditNode của chương trình* 21](#_Toc73175318)

[*Hình 25: Form Delete Edge của chương trình* 22](#_Toc73175319)

[*Hình 26: Tìm đường đi ngắn nhất giữa phòng EM204 và F410* 23](#_Toc73175320)

[*Hình 27: Ảnh mô tả đường đi trong khu EM* 23](#_Toc73175321)

[*Hình 28: Ảnh mô tả đường đi trong khu F* 24](#_Toc73175322)

[*Hình 29: Ảnh mô tả đường đi qua khu EC* 24](#_Toc73175323)

# BẢNG PHÂN CÔNG NHIỆM VỤ

|  |  |
| --- | --- |
| Phạm Minh Tiến (nhóm trưởng) | * Thiết kế cơ sở dữ liệu * Thiết kế các Form cho ứng dụng * Xây dựng mô hình 3 lớp * Lấy dữ liệu từ cơ sở dữ liệu đưa vào đồ thị * Viết báo cáo |
| Nguyễn Minh Quang | * Tìm kiếm đường đi ngắn nhất   Code các chức năng:   * Add Node * Add Edge * Delete Edge * Edit Node |
| Hồ Ngọc Hoàng Minh | * Tìm kiếm đường đi ngắn nhất   Code giao diện:   * Vẽ đường đi trên giao diện * Tạo các button mô tả từng khu * Vẽ đường đi trong từng khu |
| Phan Đức Thịnh | * Hỗ trợ viết báo cáo * Thu thập dữ liệu * Tạo bảng, nhập dữ liệu vào các bảng trong cơ sở dữ liệu |

LỜI MỞ ĐẦU

Với mục đích thay thế các loại bản đồ giấy các loại ứng dụng dẫn đường trở nên phổ biến trên toàn thế giới. Thông qua ứng dụng dẫn đường mọi người có thể tìm đường đi đến nơi cần đến một cách dễ dàng thì yêu cầu quan trọng làm sao có thể có một đường đi ngắn và chính xác nhất.

Trước các yêu cầu cần thiết đó, một số giải thuật đã được xây dựng nhằm giúp xây dựng nhằm đảm bảo có một đường đi ngắn nhất như giải thuật Dijkstra. Việc tìm hiểu và xây dựng chương trình giải thuật này cũng không nằm ngoài mục đích của đồ án môn học này. Đồ án có nhiệm vụ tìm hiểu lập trình ứng dụng.

Nhằm mang lại cho mọi người những kiến thức cơ bản về thuật toán dijkstra, đồng thời đưa ra một giải pháp cụ thể để cài đặt chương trình, nhóm em đã quyết định chọn đề tài “Ứng dụng tìm đường đi ngắn nhất giữa các khu trong trường Đại học Bách Khoa Đà Nẵng”.

Nhóm em đã nghiên cứu đề tài thông qua các phương pháp:

* Phương pháp thu thập tài liệu: thu thập tài liệu từ những bài báo khoa học, các trang web tin học và một số ebook về đề tài nghiên cứu, giáo trình và các tài liệu học tập khác.
* Phương pháp phân tích và tổng hợp tài liệu: từ những tài liệu đã thu thập, tiến hành tìm hiểu, phân tích và tồng hợp nội dung liên quan đến đề tài.
* Phương pháp chuyên gia: trong quá trình nghiên cứu có sự góp ý, điều chinh từ giáo viên hướng dẫn.
* Phương pháp phân tích và tổng hợp kinh nghiệm: sau quá trình tìm hiểu và đúc kết kinh nghiệm, tiến hành tổng hợp và hoàn thiện đề tài.
* Phương pháp thực nghiệm: sau khi cài đặt xong chương trình cần xây dựng một số bộ test tiêu biểu và thực hiện kiểm tra tính đúng đắn dựa trên kết quả thu được từ chương trình.

Chúng em xin chân thành cảm ơn thầy Phạm Minh Tuấn đã hướng dẫn chúng em hoàn thành đề tài này.

# TỔNG QUAN ĐỀ TÀI

## Mô tả bài toán

Hiện trong trường Đại học Bách Khoa vẫn chưa có ứng dụng hướng dẫn đường đi cho mọi người và google map hiện vẫn chưa hỗ trợ điều đó. Ứng dụng trong đề tài này cho phép nhập tên phòng xuất phát và tên phòng cần đi đến sau đó ứng dụng sẽ vẽ ra đường đi ngắn nhất, bên cạnh đó còn có các chức năng như thêm, sửa, xóa,…

Để thực hiện được thuật toán tìm đường đi ngắn nhất giữa 2 phòng như đã nêu ở trên ta dùng thuật toán Dijkstra. Thuật toán Dijkstra được ứng dụng để tìm ra đường đi ngắn nhất như ứng dụng phổ biến “Google map”.

## Các phần mềm, ngôn ngữ lập trình, framework để thực hiện đề tài

Ngôn ngữ lập trình C# : C# (hay C sharp) là một ngôn ngữ lập trình đơn giản, được phát triển bởi đội ngũ kỹ sư của Microsoft vào năm 2000. C# là ngôn ngữ lập trình hiện đại, hướng đối tượng và được xây dựng trên nền tảng của hai ngôn ngữ mạnh nhất là C++ và Java.

.NET framework : là 1 bộ thư viện lớp, một môi trường giúp thực thi của ứng dụng. .NET framework chứa 1 bộ thư viện rất lớn và các loại công cụ khác nhau giúp người lập trình ứng dụng xử lí hầu hết các vấn đề gặp phải trong việc phát triển ứng dụng.

IDE Visual studio : là một công cụ quen thuộc đối với các lập trình viên chuyên nghiệp, đặc biệt là những người lập trình theo hướng VB+ và C#.Đây là một phần mềm hỗ trợ đắc lực hỗ trợ công việc lập trình website.

Microsoft SQL Server : là một hệ quản trị cơ sở dữ liệu quan hệ được phát triển bởi Microsoft. Là một máy chủ cơ sở dữ liệu, một sản phẩm phần mềm có chức năng chính là lưu trữ và truy xuất dữ liệu theo yêu cầu của các ứng dụng phần mềm khác.

## Thu thập và xử lý dữ liệu

Việc thu thập dữ liệu đòi hỏi ta cần phải đi đến từng phòng học, cầu thang để có thể xác định đúng dữ liệu tọa độ pixel như ảnh dưới (dùng ứng dụng Paint) và đánh dấu chúng một cách chính xác nhất.



*Hình 1: Hình ảnh tổng quát của trường*

Các chấm đỏ thể hiện các phòng học, cầu thang trên ảnh có tọa độ x, y được tính bằng đơn vị pixel



*Hình 2: Hình ảnh minh họa Khu EM*

Như trong hình minh họa ở trên, khu EM có 6 node đỏ thể hiện lần lượt cầu thang CT11(cầu thang thứ 1 của tầng 1), phòng EM101, phòng EM102, phòng EM103, phòng EM104, và cầu thang CT21 (cầu thang thứ 2 của tầng 1).



*Hình 3: Hình ảnh chi tiết Khu EM*

Để thể hiện sự giữa các tầng khác nhau ta quy ước các node cách nhau 10 đơn vị mỗi tầng trên trục tọa độ z (ví như các node ở tầng một sẽ có z = 0, tầng hai sẽ có z = 10, …).

Mỗi tầng sẽ có các node cầu thang tương ứng như một node phòng học. Các node trên sẽ có cùng một tọa độ X hoặc Y vì cùng 1 dãy thẳng hàng. Các phòng ở tầng 2 sẽ không được đánh dấu trên ảnh mà chỉ mang tọa độ tương ứng với các phòng tầng một với sự khác nhau về tọa độ Z

## Mục tiêu cần đạt được:

* Tìm được đường đi ngắn nhất giữa hai phòng học
* Cho phép tạo thêm điểm phụ điểm để di chuyển giữa hai khu
* Cho phép tạo thêm sự liên kết giữa hai cạnh (để tạo ra đường đi)
* Cập nhật lại tọa độ các điểm phụ
* Xóa sự liên kết giữa hai cạnh (để xóa đường đi)

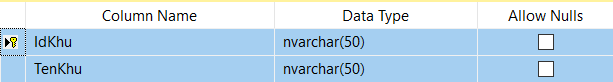
# THIẾT KẾ CƠ SỞ DỮ LIỆU

## Phân tích cơ sở dữ liệu

Trường Đại học Bách Khoa Đà Nẵng được phân chia thành nhiều khu khác nhau nên trong cơ sở dữ liệu phải có bảng thể hiện được từng khu(table KHU). Với mỗi khu ta lại có cầu thang và phòng học nên lại có thêm 2 table CAUTHANG và PHONGHOC. Trong ứng dụng còn cho phép người dùng tạo thêm các Node và các Node này sẽ được đưa về table ADDNODE

## Thiết kế các bảng

### Bảng KHU



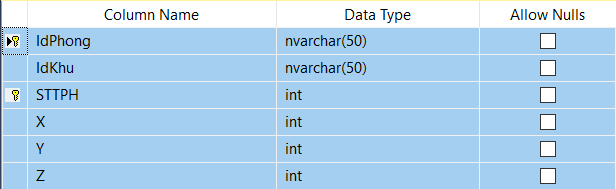
*Hình 4: Bảng KHU*

Bảng KHU được dùng để lưu trữ dữ liệu các khu trong trường

IdKhu: Mã số của từng khu, kiểu dữ liệu nvarchar(50)

TenKhu: Tên của từng khu, kiểu dữ liệu nvarchar(50)

### Bảng PHONGHOC



*Hình 5: Bảng PHONGHOC*

Bảng PHONGHOC được dùng để lưu trữ dữ liệu các phòng học

IdPhong: Tên của từng phòng trong khu, kiểu dữ liệu nvarchar(50)

IdKhu: Mã số của các khu, kiểu dữ liệu nvarchar(50)

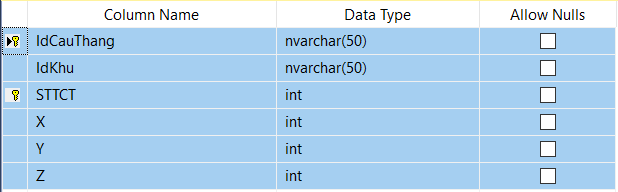
STTPH: Số thứ tự của từng phòng, kiểu dữ liệu int

X: Tọa độ của phòng học theo trục Ox, kiểu dữ liệu int

Y: Tọa độ của phòng học theo trục Oy, kiểu dữ liệu int

Z: Tọa độ của phòng học theo trục OZ, kiểu dữ liệu int

### Bảng CAUTHANG



*Hình 6: Bảng CAUTHANG*

Bảng CAUTHANG được dùng để lưu trữ dữ liệu các cầu thang

IdCauThang: Tên của từng cầu thang trong khu, kiểu dữ liệu nvarchar(50)

IDKhu: Mã số của các khu, kiểu dữ liệu nvarchar(50)

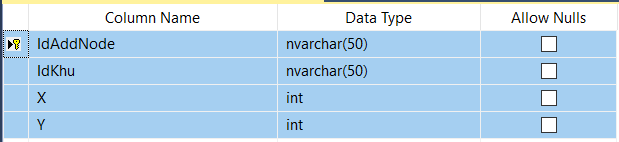
STTCT: Số thứ tự của từng cầu thang, kiểu dữ liệu int

X: Tọa độ của cầu thang theo trục Ox, kiểu dữ liệu int

Y: Tọa độ của cầu thang theo trục Oy, kiểu dữ liệu int

Z: Tọa độ của cầu thang theo trục OZ, kiểu dữ liệu int

### Bảng ADDNODE



*Hình 7: Bảng ADDNODE*

Bảng ADDNODE được dùng để lưu trữ dữ liệu các điểm phụ khi người dùng tạo thêm Node mới

IdAddNode: Tên của từng điểm phụ, kiểu dữ liệu nvarchar(50)

IDKhu: Mã số của các khu, kiểu dữ liệu nvarchar(50)

X: Tọa độ của điểm phụ theo trục Ox, kiểu dữ liệu int

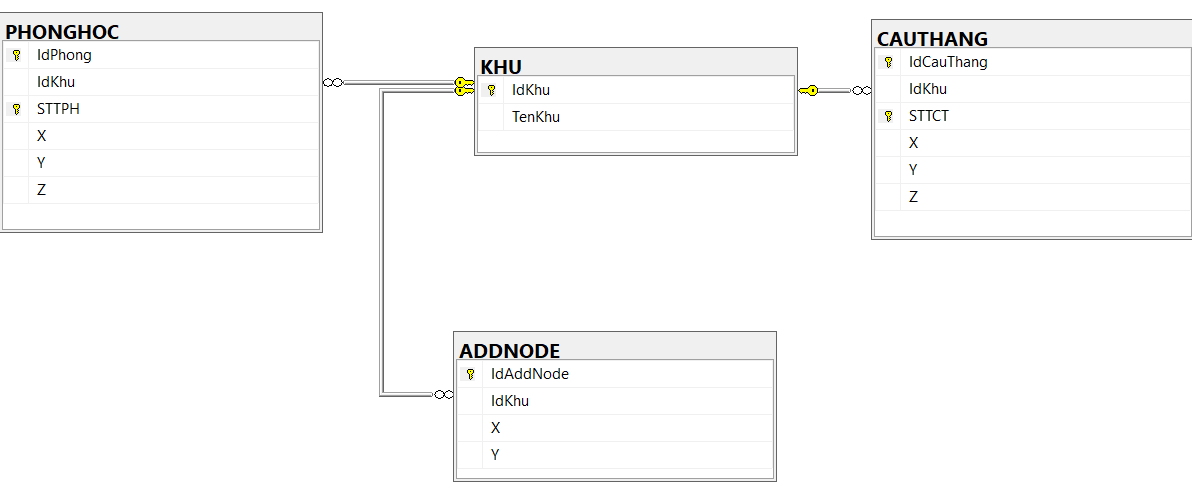
Y: Tọa độ của điểm phụ theo trục Oy, kiểu dữ liệu int

## Xây dựng mối liên hệ giữa các bảng

Một CSDL không thể thiếu đi các mối quan hệ, những thành phần chủ yếu mà chúng ta thường dùng để liên kết thông tin giữa các đối tượng có trong CSDL. Các quan hệ mà chúng ta mô tả trong CSDL có nhiệm vụ nối kết các thông tin giữa các đối tượng một các chính xác và đầy đủ mối quan hệ giữa các đối tượng.

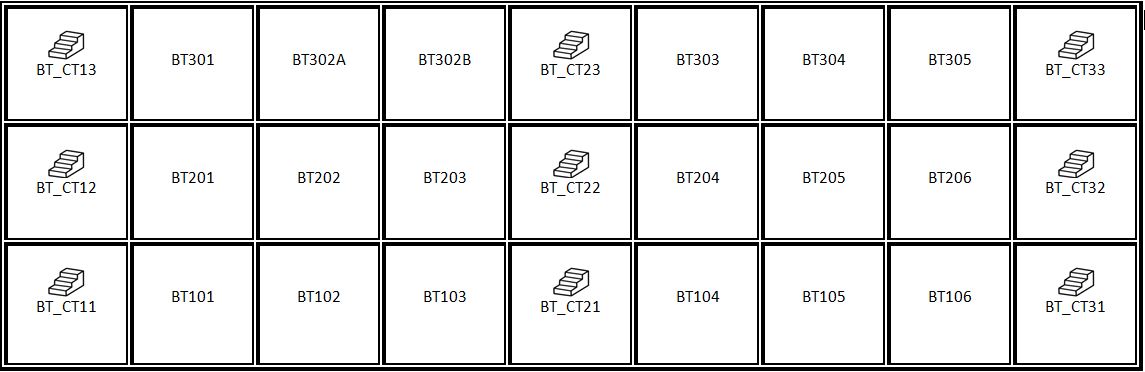
Quan hệ giữa 2 bảng :

* KHU và CAUTHANG : Là Quan hệ 1-n. Vì 1 khu thì có thể có nhiều cầu thang
* KHU và PHONGHOC : Là Quan hệ 1-n. Vì 1 khu thì có thể có nhiều phòng học
* KHU và ADDNODE : Là Quan hệ 1-n. Vì 1 khu thì có thể có nhiều điểm phụ

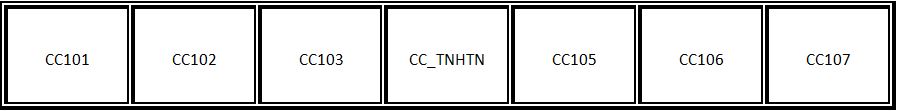


*Hình 8: Liên hệ giữa các bảng*

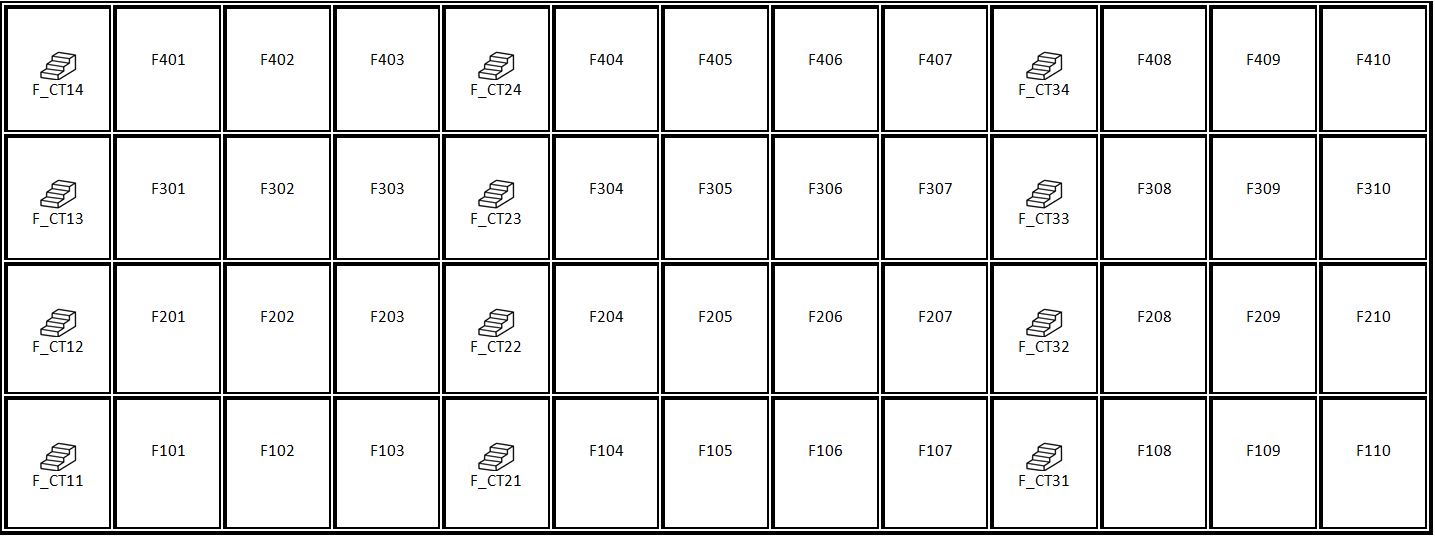
## Hình ảnh chi tiết các khu



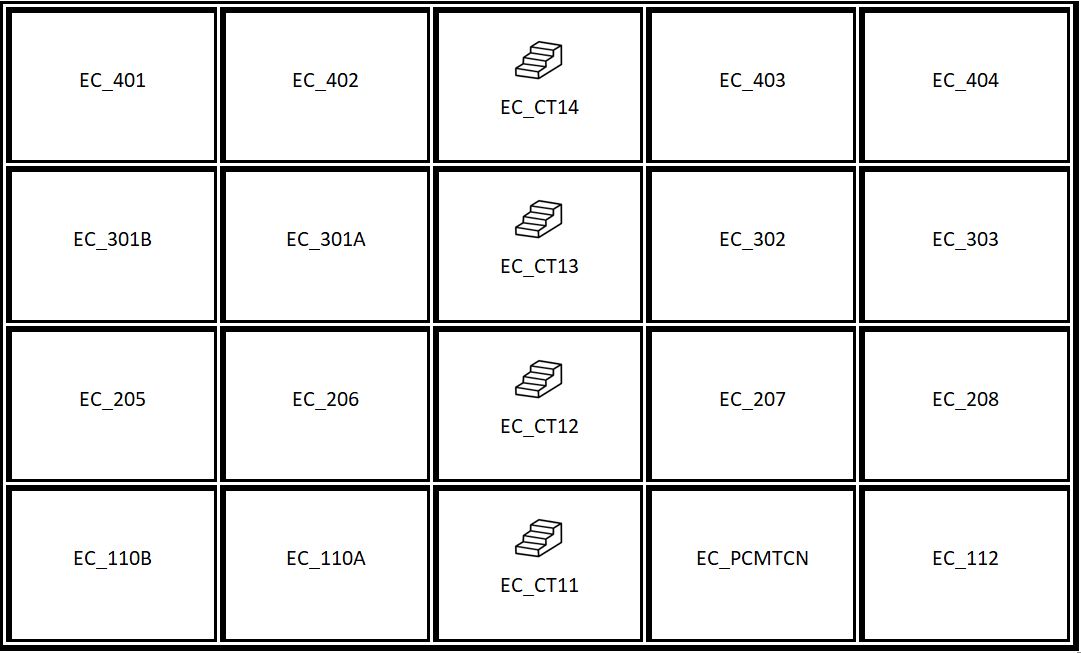
*Hình 9: Hình ảnh chi tiết khu BT*



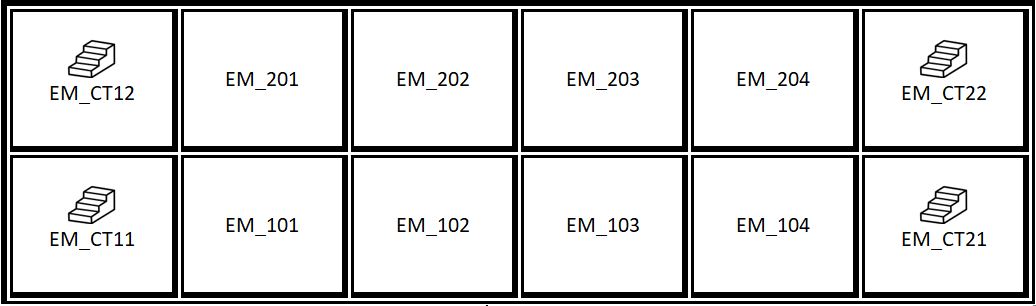
*Hình 10: Hình ảnh chi tiết khu CC*



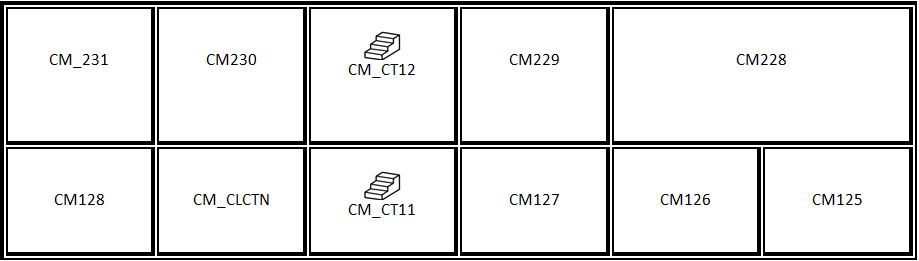
*Hình 11: Hình ảnh chi tiết khu F*



*Hình 12: Hình ảnh chi tiết khu EC*



*Hình 13: Hình ảnh chi tiết khu EM*



*Hình 14: Hình ảnh chi tiết khu CM*

# THIẾT KẾ THUẬT TOÁN

## Phát biển bài toán

Trong khuôn viên trường Đại học Bách Khoa Đà Nẵng có rất nhiều khu, mỗi khu có các phòng học khác nhau. Có rất nhiều cách để di chuyển từ phòng học này đến phòng học khác. Nhiệm vụ của chương trình là trong các đường đi có thể giữa 2 phòng, ta xác định được đường đi hợp lệ ngắn nhất.

* Mô tả Input :

Từ cơ sở dữ liệu cho ta biết được tên phòng, phòng thuộc khu nào, tọa độ (x, y, z) của các phòng học, các cầu thang.

* Mô tả Ouput:

Từ các đường đi hợp lệ tìm ra đường đi ngắn nhất. Nếu không có đường đi thì in ra “Không có đường đi giữa 2 phòng A và B”



*Hình 15: Hình ảnh mô tả output*

* Ngoài ra người dùng còn có thêm những Node phụ ở tầng 1. Những

Node này nếu có thuộc một khu X thì Node đó sẽ kết nối hết với tất cả các Node ở tầng 1 của khu X, nếu không thuộc khu nào thì Node chỉ được thêm vào đồ thị hiện tại

* Những Node được người dùng thêm vào sẽ được thêm vào table

ADDNODE và những Node trong table ADDNODE sẽ được vẽ lên ảnh với màu vàng

* Khi chỉnh sửa Node cũng chỉ có thể chỉnh sửa trên các Node trong

table ADDNODE, và ảnh sẽ được load lại tại đúng tọa độ x, y vừa được chỉnh sửa

* Việc tìm đường đi phải linh động thay đổi khi ta thực hiện các thao

tác như thêm, xóa đường đi giữa 2 Node, thêm Node,…

## Phân tích thuật toán

### Kết nối các Node trong đồ thị

Trước hết để áp dụng được thuật toán Dijkstra vào bài toán. Ta phải tạo liên kết giữa các Node (các phòng học, cầu thang) trong đồ thị.

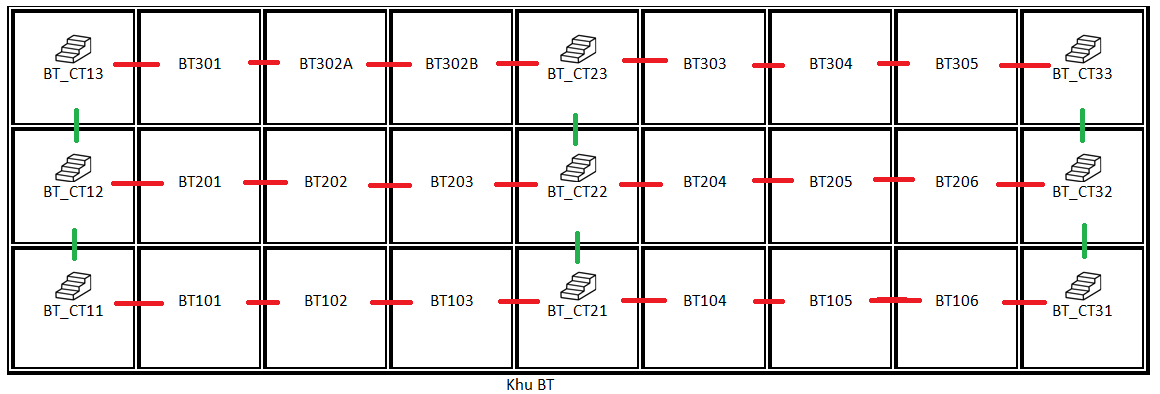
Dữ liệu đầu vào của bài toán có khoảng 150 Node nên việc kết nối chúng lại với nhau có thể tạo nên tối đa 11175 cạnh. Với số lượng lớn các cạnh như vậy ta không thể tạo các cạnh (nối 2 Node sẽ tạo nên một cạnh) một cách “thủ công”, việc này dẫn ta tới việc xây dựng một hàm mà ở đó các Node trong một khu sẽ kết nối lại được với nhau, còn việc kết nối giữ 2 khu bất kì lúc này sẽ đơn giản hơn vì số lượng các Node kết nối giữa 2 khu là không nhiều. Hàm kết nối các Node trong một khu được mô tả như ở dưới đây :

Input : Tập hợp tất cả các node ,mảng chứa tên tất cả các khu

Output : Kết nối hợp lệ tất cả các node trong cùng 1 khu với nhau .

Mô tả thuật toán :

* Bước 1 : Với mỗi tên khu ,lấy ra tất cả các node (cầu thang và phòng học) thuộc khu đó lưu vào một listNode mới
* Bước 2 : Với mỗi listNode , đếm số phòng của mỗi tầng (lưu là floorNode) , tìm ra số tầng = max(z)/10 + 1 và lưu những node cầu thang vào list ctNode
* Bước 3 : Với mỗi tầng ,ta kết nối các node ở tầng đó lần lượt theo số thứ tự liên tiếp nhau (vd 1 – 2, 2 – 3, … n-1 - n).Và kết nối các cầu thang trong ctNode lại với nhau.



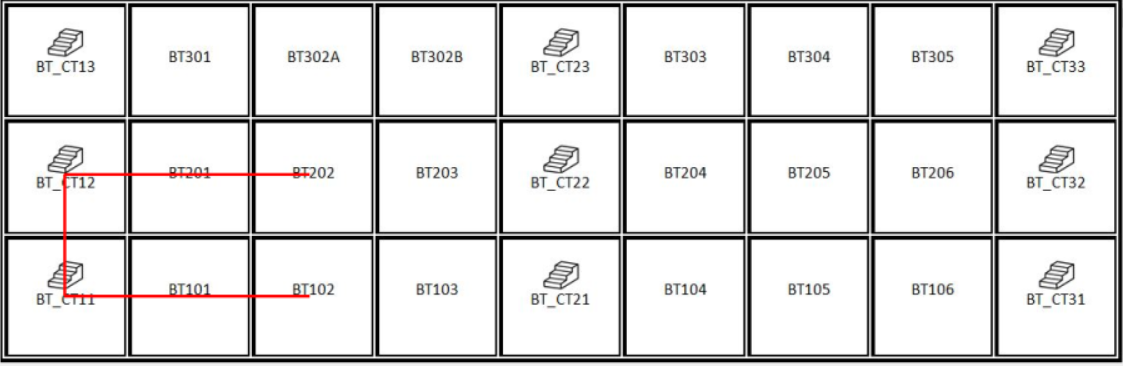
*Hình 16: Hình minh họa kết nối các phòng học, cầu thang*

Như trên là minh họa cho việc kết nối các phòng học, cầu thang của khu BT với nhau. Trong dữ liệu ta đã đánh số lần lượt từ trái qua phải (BT\_CT11 : 1, BT101 : 2, ….., BT305 : 26, BT\_CT31 : 27). Trong cùng một dãy ta sẽ kết nối các node liên tiếp nhau (vd : 1 kết nối 2, 2 kết nối 3, … cho đến hết phòng của dãy), còn các cầu thang ta kết nối theo như đường màu xanh trong hình vẽ.

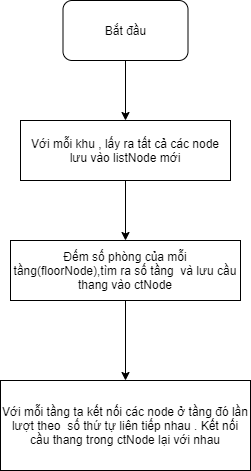
Nêu không có thuật toán trên mà thay vào đó là kết hợp tất cả các Node trong 1 khu với nhau sẽ xảy ra sai sót. Có thể kể đến một ví dụ như khi đi từ BT102 đến BT202 thì đường đi ngắn nhất giữa 2 Node sẽ là BT102 -> BT202 nhưng như vậy sẽ mâu thuẫn vì không có cách nào đi được trực tiếp từ lầu một lên lầu hai mà không qua cầu thang.

Như vậy theo thuật toán kết nối hợp lệ các điểm trong khu như trên sẽ giúp các đường đi logic hơn. Và ta sẽ thu được đường đi đúng như sau :

BT102 -> BT101 -> BT\_CT11 -> BT\_CT12 -> BT201 -> BT202



*Hình 17: Đường đi ngắn nhất từ BT102 đến BT202*



*Hình 18:Sơ đồ thuật toán kết nối các khu*

Sau khi đã kết nối được tất cả các Node trong cùng một khu và giữa các khu với nhau ta tạo được một đồ thì hoàn chỉnh. Ta sẽ tính khoảng cách giữa 2 Node theo tọa độ (x, y, z) rồi áp dụng thuật toán Dijkstra để tìm ra đường đi ngắn nhất

### Thuật toán Dijkstra

Mô tả về giải thuật Dijkstra:

* **Bước 1**: Chọn S = {} là tập các soure\_node bao gồm current\_node và passed\_node . Với current\_node là node đang được xét đến, passed\_node là các node đã được xét. current\_node đầu tiên sẽ là node đích của bài toán tìm đường đi ngắn nhất.
* **Bước 2**: Khởi tạo giải thuật với current\_node là node đích và cost(N) là giá trị của đường đi ngắn nhất từ N đến node đích.
* **Bước 3**: Xét các node kề N với current\_node . Gọi d(current\_node,N) là khoảng cách giữa node kề N và current\_node . Với p = d(current\_node,N) + cost (current\_node). Nếu p < cost(N) thì cost(N) = p . Nếu không thì cost(N) giữ nguyên giá trị .
* **Bước 4**: Sau khi xét hết các node kề N, đánh dấu current\_node thành passed\_node .
* **Bước 5**: Tìm current\_node mới với 2 điều kiện: không phải passed\_node và cost(current\_node) là nhỏ nhất
* **Bước 6**: Nếu tập S = {} chứa đủ các node của đồ thị thì dừng thuật toán. Nếu không thì quay trở lại **Bước 3**.



*Hình 19: Sơ đồ thuật toán Dijkstra*

# CHƯƠNG TRÌNH VÀ KẾT QUẢ

## Tổ chức chương trình

### Mô Hình Three-Layer

Ứng dụng được tổ chức dựa trên mô hình 3 lớp (three layer)

Mô hình three-layer gồm có 3 phần chính (GUI, BLL, DAL):

– Presentation Layer (GUI) : Lớp này có nhiệm vụ chính giao tiếp với người dùng. Nó gồm các thành phần giao diện ( win form, web form,…) và thực hiện các công việc như nhập liệu, hiển thị dữ liêu, kiểm tra tính đúng đắn dữ liệu trước khi gọi lớp BLL

– Business Logic Layer (BLL) : Layer này phân ra 2 thành nhiệm vụ :

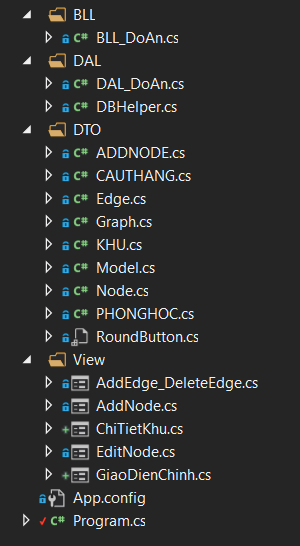
* Đây là nơi đáp ứng các yêu cầu thao tác dữ liệu của GUI layer, xử lý chính nguồn dữ liệu từ Presentation Layer trước khi truyền xuống Data Access Layer.
* Đây còn là nơi kiểm tra các ràng buộc, tính toàn vẹn và hợp lệ dữ liệu, thực hiện tính toán và xử lý các yêu cầu nghiệp vụ, trước khi trả kết quả về Presentation Layer.

– Data Access Layer (DAL) : Lớp này có chức năng giao tiếp với hệ quản trị CSDL như thực hiện các công việc liên quan đến lưu trữ và truy vấn dữ liệu ( tìm kiếm, thêm, xóa, sửa,…).

-DTO Layer (Không bắt buộc): Đây là lớp định nghĩa các table

trong database, định nghĩa cột của nó cũng như để ta gán data khi query lấy dữ liệu, giúp truyền dữ liệu giữa các tầng

### Các Class trong từng phần của mô hình 3 lớp



*Hình 20: Các class trong mô hình 3 lớp*

* View(GUI) Layer : Gồm 2 lớp
* GiaoDienChinh : Dùng để chứa giao diện chính của chương trình
* ChiTietKhu : Khi người dùng click vào button từng khu sẽ hiện ra sơ đồ khu tương ứng
* AddEdge\_DeleteEdge : Dùng để thêm đường đi giữa 2 Node hoặc xóa đường đi giữa 2 Node
* AddNode : Dùng để thêm Node phụ
* EditNode : Dùng để chỉnh sửa thông tin của Node phụ
* BLL Layer: chỉ gồm 1 class là BLL\_DoAn. Dùng để đáp ứng các yêu cầu thao tác dữ liệu của View, và kiểm tra tính toàn vẹn, hợp lệ dữ liệu, thực hiện tính toán và xử lý các yêu cầu nghiệp vụ.
* DAL Layer : Gồm 2 lớp
* DAL\_DoAn : Tương tác với BLL\_DoAn là nơi chứa các câu lệnh Query để truy vấn về lớp DBHelper.
* DBHelper : Nhận các câu lệnh Query (select, insert, delete, update…) từ DAL\_DoAn sau đó tương tác trực tiếp tới cơ sở dữ liệu
* DTO : Gồm 8 lớp, với 3 mục đích chính như sau:

+) Lấy dữ liệu từ cơ sở dữ liệu:

* KHU : Là đối tượng khu của trường Đại học Bách Khoa Đà Nẵng. được lấy dữ liệu từ table KHU trong cơ sở dữ liệu
* CAUTHANG : Là đối tượng cầu thang của các khu, được lấy từ table CAUTHANG trong cơ sở dữ liệu
* PHONGHOC : Là đối tượng phòng học của các khu, được lấy từ table PHONGHOC trong cơ sở dữ liệu
* ADDNODE : Là Node được người dùng tạo ra thêm trong quá trình sử dụng nhầm tạo ra thêm đường đi, điểm di chuyển trong đồ thị

+) Phục vụ cho thuật toán Dijkstra:

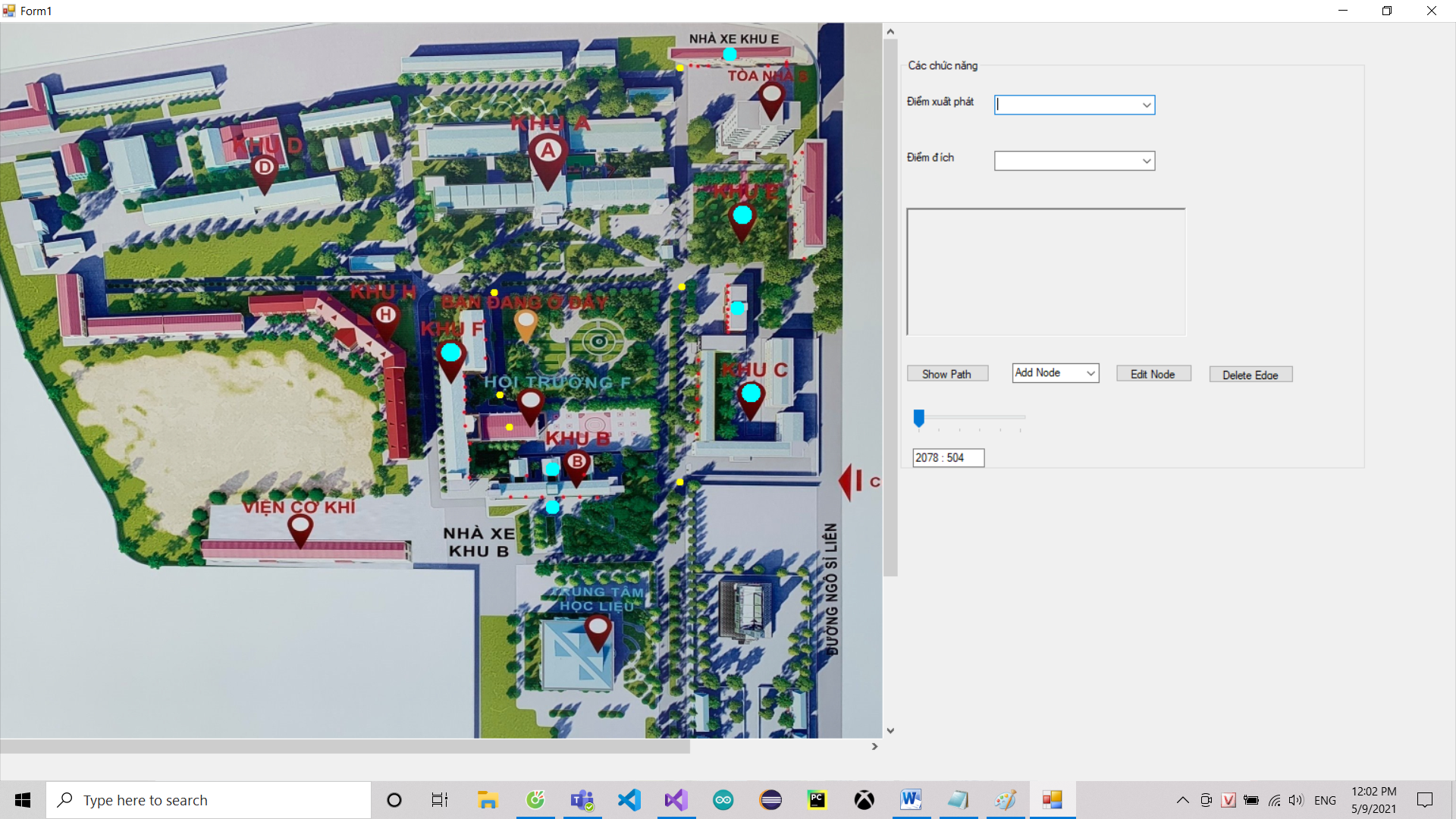
* Node : Dùng để biểu diễn cầu thang, phòng học trong đồ thị
* Edge : Dùng để biểu diễn các cạnh trong đồ thị
* Graph : Dùng để khởi tạo đồ thị, nơi chứa thuật toán Dijkstra và các hàm liên quan tới việc xử lí trên đồ thị

+) Phục vụ cho việc vẽ đường đi trên giao diện :

* Model : Giúp cho việc vẽ đường đi ngắn nhất trong một khu
* RoundButton : Tạo các Button hình tròn cho từng khu

## Kết quả

### Giao diện chính của chương trình



*Hình 21: Giao diện chính của chương trình*

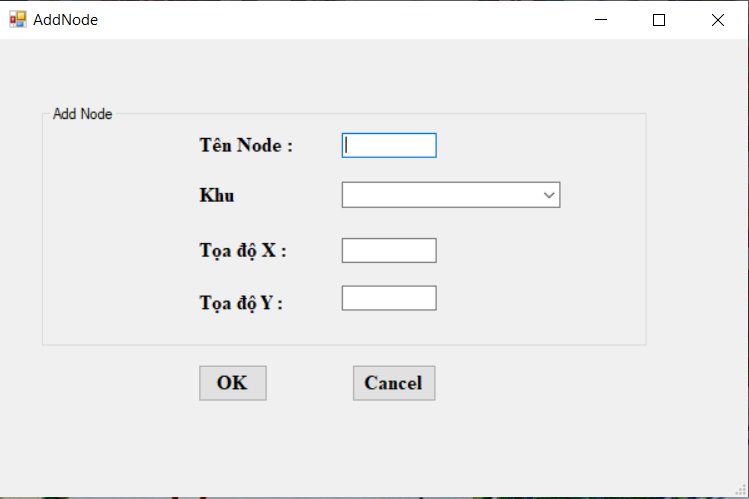
Trong giao diện ta có thể thấy :

* Picturebox : dùng để hiện đường đi trên ảnh
* Điểm xuất phát : là combobox các Node có trong đồ thị để chọn điểm xuất phát
* Điểm đích : là combobox các Node có trong đồ thị để chọn điểm đến
* Show Path : là button khi người dùng click vào thì trên picturebox và richtextbox sẽ đồng thời hiển thị ra đường đi
* Add: là một combobox trong đó có 2 chức năng là Add Node (dùng để thêm một Node) và Add Edge (dùng để thêm một cạnh)
* Edit Node : là button dùng để chỉnh sửa lại các Node phụ được người dùng thêm vào
* Delete Edge: là button dùng để xóa đường đi giữa 2 Node
* Trackbar : dùng để phóng to hoặc thu nhỏ ảnh
* Textbox ở dưới cùng dùng để hiện tọa độ của con trỏ chuột khi, con chuột di chuyển trên picturebox
* Các Button hình trong dùng để hiển thị chi tiết từng khu

### Kết quả thực thi của chương trình

1. **Các Form trong chương trình**

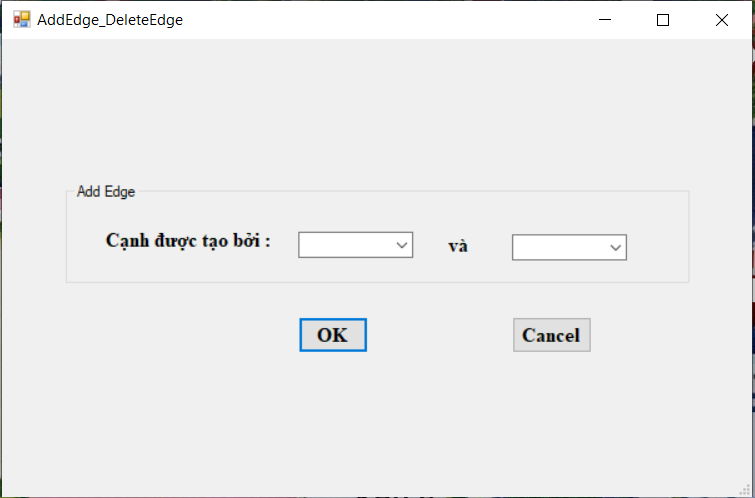
* Giao diện thêm một Node (chỉ thêm Node phụ)



*Hình 22: Form AddNode của chương trình*

Node được thêm vào sẽ được gọi là Node phụ, Node này là node ở tầng một (z = 0) và Node đó và sẽ được thêm vào table ADDNODE trong cơ sở dữ liệu

* Tên Node : Cũng là Id của Node (khóa chính)
* Khu : Khu cần thêm Node vào
* Tọa độ X, Y : Nhập tọa độ X, Y theo đơn vị pixel trong ảnh
* Giao diện thêm một cạnh

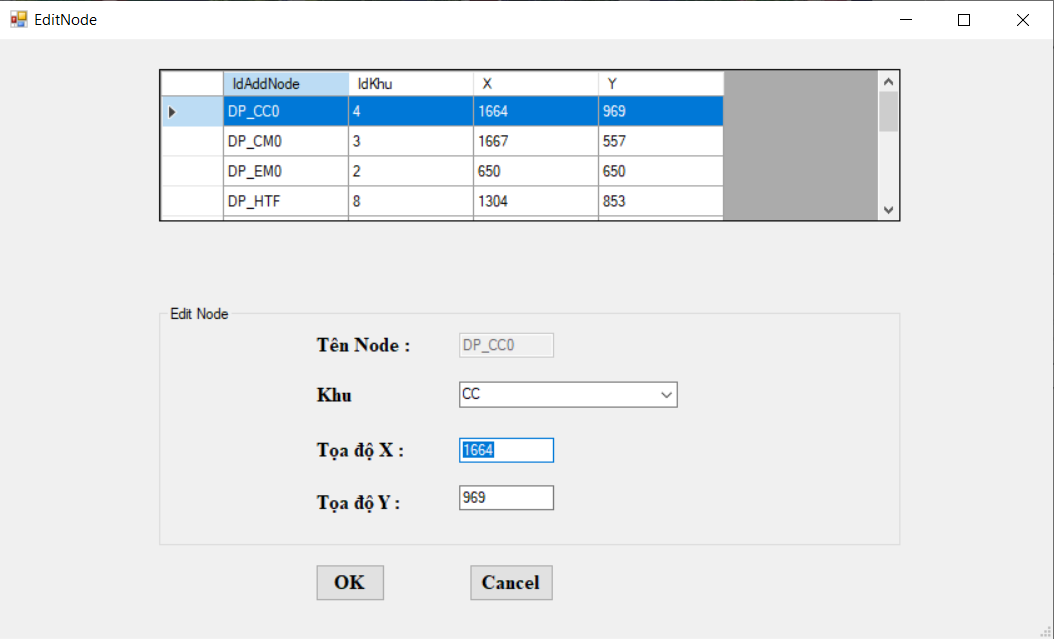


*Hình 23: Form Add Edge của chương trình*

Dùng để thêm đường đi giữa 2 Node trong đồ thị

Cả 2 combobox đều dùng để hiển thị các Node hiện có trong đồ thị

* Giao diện chỉnh sửa một Node (chỉ sửa Node phụ)



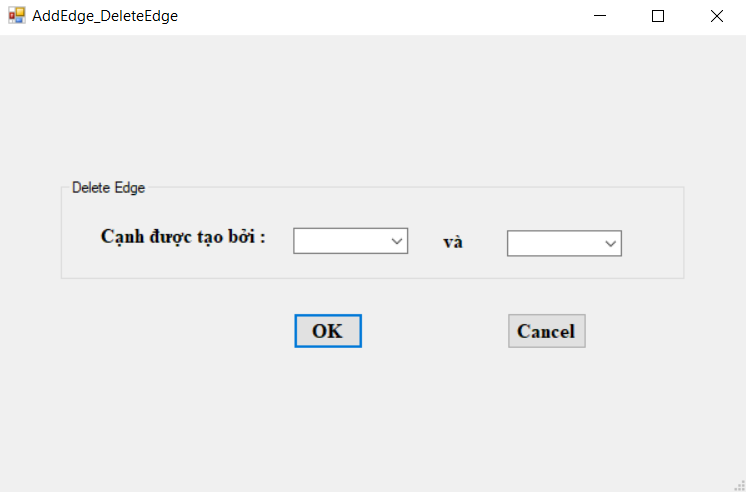
*Hình 24: Form EditNode của chương trình*

Dùng để chỉnh sửa các Node trong table ADDNODE

* Datagridview : dùng để hiển thị table ADDNODE được lấy từ cơ sở dữ liệu

Khi người dùng click vào một row trên datagridview thì dữ liệu của row đó sẽ được hiển thị xuống groupbox bên dưới, trong groupbox này gồm có :

* Tên Node : Cũng là Id của Node, textbox này sẽ không được chỉnh sửa vì đây là khóa chính
* Khu : Khu cần thêm Node vào
* Tọa độ X, Y : Nhập tọa độ X, Y theo đơn vị pixel trong ảnh
* Giao diện xóa một cạnh



*Hình 25: Form Delete Edge của chương trình*

Dùng để xóa đường đi giữa 2 Node có trong đồ thị

Cả 2 combobox đều dùng để hiển thị các Node hiện có trong đồ thị

Khi

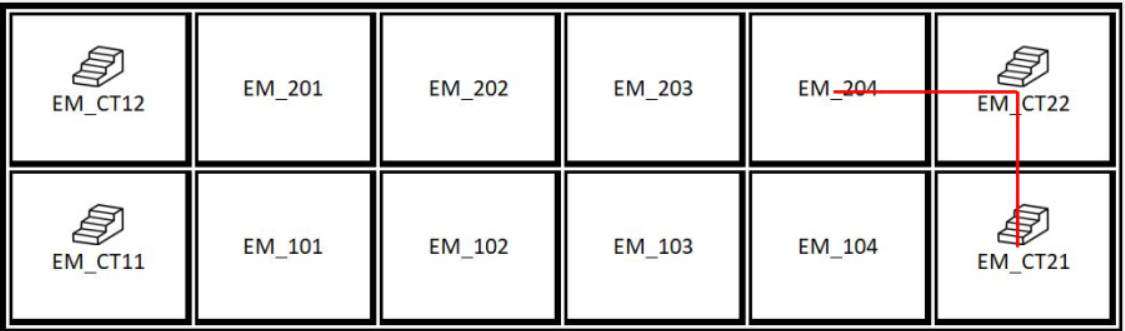
1. **Ảnh chi tiết đường đi trong một khu**

Khi người dùng chọn đường đi ở những Node trên tầng ví dụ như ta cần đi từ EM204 đến F410 thì ở picturebox chỉ hiển thị cho người dùng thấy đường đi trên mặt phẳng x, y như hình sau :

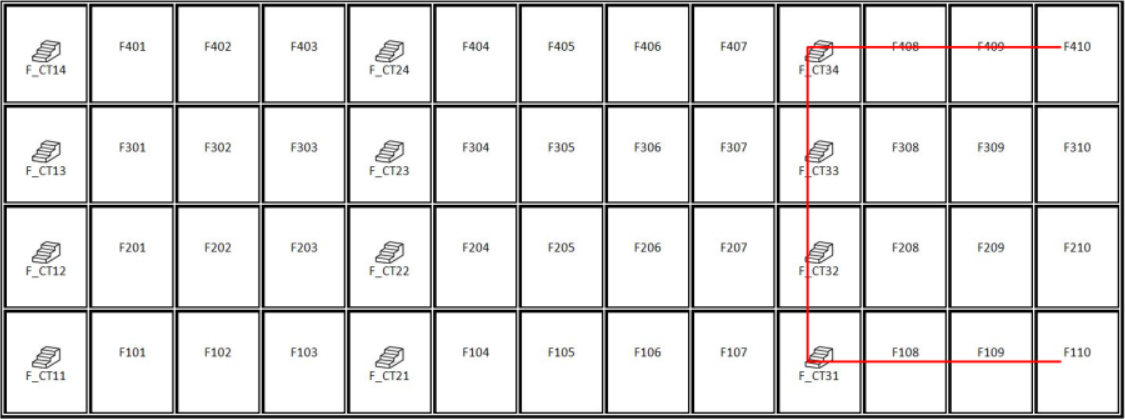


*Hình 26: Tìm đường đi ngắn nhất giữa phòng EM204 và F410*

Ở các khu sẽ có các button hình tròn có màu xanh. Khi người dùng click vào thì sẽ show ra đường đi chi tiết trong khu, như dưới đây :



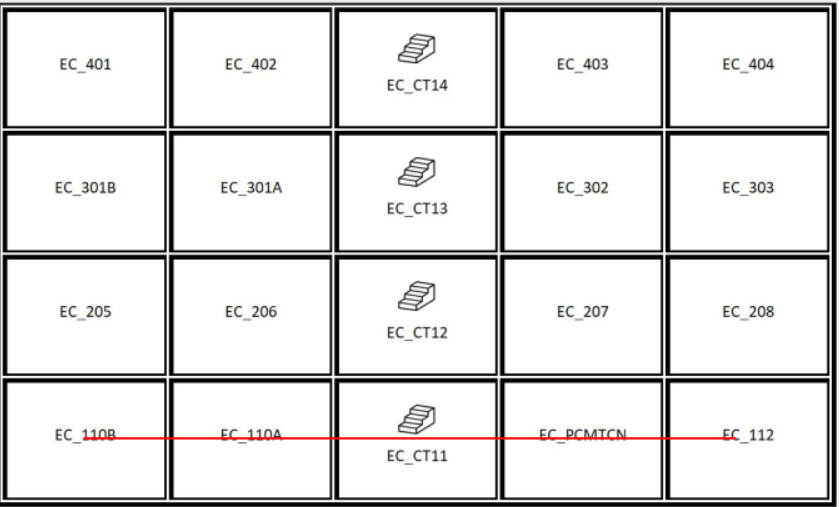
*Hình 27: Ảnh mô tả đường đi trong khu EM*



*Hình 28: Ảnh mô tả đường đi trong khu F*

Qua 2 hình trên cho ta biết cách đi trong mỗi khu như thế nào để đường đi giữa 2 phòng là ngắn nhất

Vì việc di chuyển này như trong hình 26 đi qua khu EC thì trong ảnh chi tiết của khu EC cũng sẽ thể hiện đường đi như sau :



*Hình 29: Ảnh mô tả đường đi qua khu EC*

### Nhận xét

* Thực hiện được các chức năng như : Tìm kiếm đường đi, thêm cạnh, thêm Node, sửa Node, xóa Node
* Ứng dụng tìm chính xác được đường đi ngắn nhất giữa 2 điểm có trong đồ thị
* Giao diện rõ ràng dễ dàng sử dụng
* Ứng dụng được viết dưới dạng mô hình ba lớp thể hiện được rõ ràng chức năng của từng tầng (View, BLL, DAL)
* Hiện tại trong cơ sở dữ liệu vẫn chưa có khu H, khu A, …

**KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

## Kết luận

- Ứng dụng đã đáp ứng được có chức năng cơ bản cần có của một bài toán tìm đường đi ngắn nhất

- Sinh viên biết được cách thiết kế một CSDL.

- Biết cách kết nối dữ liệu từ CSDL.

- Ứng dụng được mô hình 3 lớp

- Áp dụng được các giải thuật, cấu trúc dữ liệu và các kiến thức đã học từ môn Lập trình hướng đối tượng và Cơ sở dữ liệu vào chương trình.

## Hướng phát triển

- Thêm nhiều tính năng mới như sửa node, xóa node để phục vụ thêm cho người sử dụng

- Việc kết nối giữa các Node của 2 khu còn làm thủ công, sẽ tìm cách kết nối bằng hàm mà không cần phải tốn thời gian nhập tay

- Thêm chức sắp xếp độ dài đường đi có thể giữa 2 Node

- Tăng tốc độ mở của ứng dụng, tăng cường bảo mật của Database.

- Áp dụng thành Website cho trường.

- Sẽ đưa thêm khu H, A, toàn bộ khu C vào dữ liệu

**TÀI LIỆU THAM KHẢO**

Bài đọc về mô hình 3 lớp :

<https://topdev.vn/blog/mo-hinh-3-lop-la-gi/>

Thuật toán Dijkstra viết theo hướng đối tượng :

<https://stackabuse.com/graphs-in-java-dijkstras-algorithm/>