**TÌM ĐƯỜNG ĐI NGẮN NHẤT GIỮA 2 ĐIỂM S VÀ G TRONG KHÔNG GIAN CÓ CÁC VẬT CẢN ĐA GIÁC LỒI**

**Phạm Tấn Đạt – 18110094**

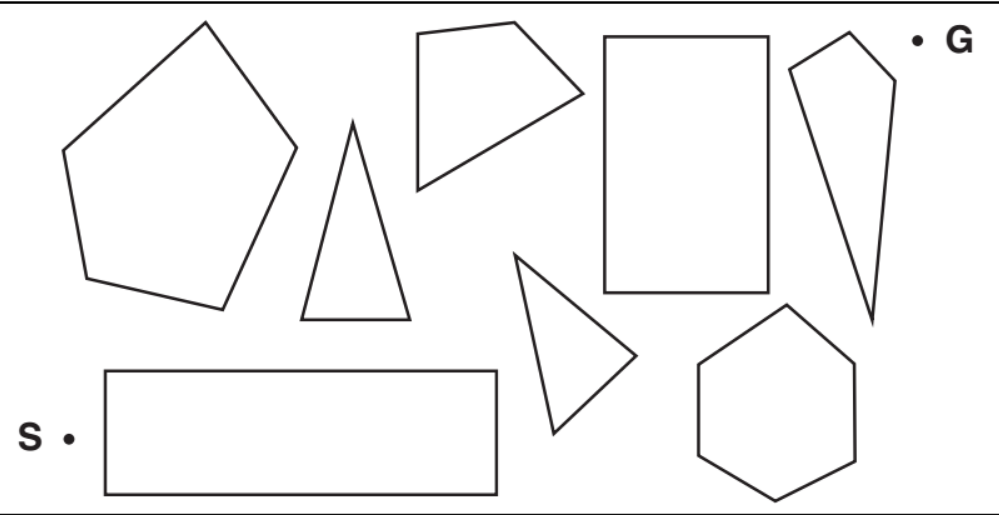
**Nguyễn Ngọc Gia Bảo – 18110079**

*Trường Đại học Sư phạm Kỹ thuật TP.HCM*

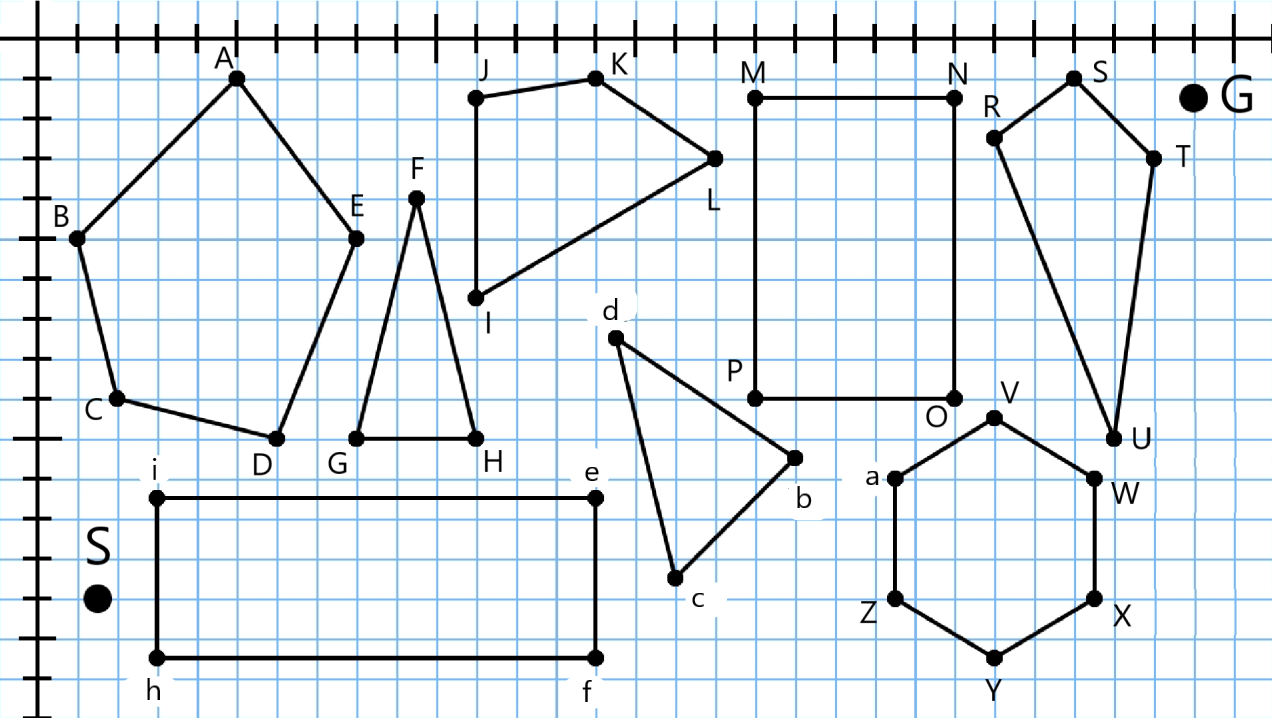
**TÓM TẮT**

Trong các ứng dụng thực tế, bài toán tìm đường đi ngắn nhất giữa hai đỉnh của một đồ thị liên thông có một ý nghĩa to lớn. Có thể dẫn về bài toán như vậy nhiều bài toán thực tế quan trọng. Ví dụ, bài toán chọn một hành trình tiết kiệm nhất (theo tiêu chuẩn hoặc khoảng cách hoặc thời gian hoặc chi phí) trên một mạng giao thông đường bộ, đường thủy hoặc đường không; bài toán chọn một phương pháp tiết kiệm nhất để đưa ra một hệ thống động lực từ trạng thái xuất phát đến trạng một trạng thái đích, bài toán lập lịch thi công các công các công đoạn trong một công trình thi công lớn, bài toán lựa chọn đường truyền tin với chi phí nhỏ nhất trong mạng thông tin, v.v… Có rất nhiều thuật toán để giải các bài toán như vậy, nhóm em sử dụng 2 thuật toán là Breadth-First Search và Depth-First Search để giải bài toán *tìm đường đi ngắn nhất giữa 2 điểm S và G trong không gian có các vật cản đa giác lồi.*

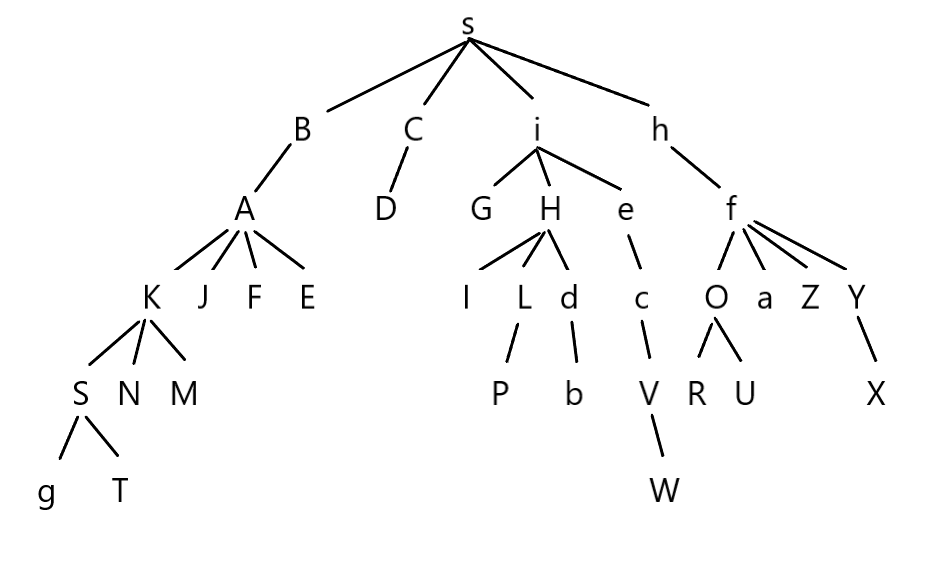
**Từ khóa:** Tìm đường đi ngắn nhất, Beadth-First Search, Depth-First Seacrh



*Hình 1. Không gian trạng thái có các chướng ngại vật theo hình đa giác lồi. S là điểm bắt đầu, G là điểm đích*

**

*Hình 2. Không gian trạng thái được mô phỏng lại*

**

*Hình 3. Cây đồ thị*

1. **ĐẶT VẤN ĐỀ:**

Với sự phát triển của AI trong xã hội con người hiện nay, nhiều lĩnh vực đã và đang sử dụng AI để thay thế con người làm việc. Vì vậy, AI có tác động rất lớn đối với xã hội và là một ngành khoa học phổ biến hiện nay.

Bài toán tìm đường đi ngắn nhất là một bài toán cổ điển trong Trí tuệ nhân tạo (AI). Vấn đề là giữa đa số đường đi giữa 2 điểm trong một đồ thị thì làm sao để có thể tìm được đường đi ngắn nhất, nhanh nhất và ít bước di chuyển nhất. Và việc tìm hiểu bài toán này là một trong những ứng dụng của AI trong việc phát triển robot giúp ích cho con người.

1. **PHƯƠNG PHÁP THỰC HIỆN:**

Có nhiều thuật toán có thể tìm đường đi ngắn nhất nhưng nhóm em sử dụng 2 thuật toán là BFS (Breadth-First Search) và DFS (Depth-First Search) để giải bài toán này.

* 1. Thuật toán BFS:

Khởi tạo ElementList gồm các thứ tự các đỉnh được duyệt theo thuật toán Breadth-First Search

Thuật toán BFS(start, target)

Bước 1(Khởi tạo):

Queue = ;

Push(Queue, start);

Bước 2 (Lặp):

While(Queue ≠ ) do

pNode = Queue.begin();

// duyệt lần lượt các đỉnh từ ElementList vào Queue

Push(Queue, ElementList.at(i));

Pop(Queue);

// duyệt đến đỉnh cần tìm thì dừng

If (pNode = target) then

<tìm đường đi từ target về start>

break;

End IF;

End While

Bước 3 (Trả lại kết quả)

Return (Đường đi từ start đến target)

End.

*Hình 4. Mã giả thuật toán BFS*

* 1. Thuật toán DFS:

Khởi tạo ElementList gồm các thứ tự các đỉnh được duyệt theo thuật toán Depth-First Search

Thuật toán DFS(start, target)

Bước 1(Khởi tạo):

Queue = ;

Push(Queue, start);

Bước 2 (Lặp):

While(Queue ≠ ) do

pNode = Queue.begin();

// duyệt lần lượt các đỉnh từ ElementList vào Queue

Push(Queue, ElementList.at(i));

Pop(Queue);

// duyệt đến đỉnh cần tìm thì dừng

If (pNode = target) then

<tìm đường đi từ target về start>

End IF;

End While

Bước 3 (Trả lại kết quả)

Return (Đường đi từ start đến target)

End.

*Hình 5, Mã giả thuật toán DFS*

Thuật toán BFS và DFS của nhóm em hoạt động như sau:

Ban đầu tạo một List gồm các đỉnh trong cây đồ thị theo thứ tự duyệt theo mỗi thuật toán. Sau đó tạo một hàng đợi Queue lưu đỉnh bắt đầu và tạo vòng lặp Queue khác rỗng. Duyệt lần lượt các đỉnh trong List bằng cách đưa lần lượt các đỉnh vào Queue và xóa phần tử đầu tiên, nếu Queue.begin() là đỉnh đích thì in ra đường đi, với thuật toán DFS là in ra đường đi và dừng.

1. **SO SÁNH 2 THUẬT TOÁN BFS VÀ DFS**

* **Depth-First Search (DFS):** duyệt theo chiều sâu

*+ Tính hiệu quả:*  
 Hiệu quả khi lời giải nằm sâu trong cây tìm kiếm và có một phương án  
chọn hướng đi chính xác.  
 Hiệu quả của chiến lược phụ thuộc vào phương án chọn hướng đi.  
 Phương án càng kém hiệu quả thì hiệu quả của chiến lược càng giảm.  
 Thuận lợi khi muốn tìm chỉ một lời giải.

+ *Lượng bộ nhớ sử dụng để lưu trữ các trạng thái:* Chỉ lưu lại các trạng thái chưa xét đến.

*+ Trường hợp xấu nhất:* Vét cạn toàn bộ

*+ Trường hợp tốt nhất*  
 Phương án chọn hướng đi tuyệt đối chính xác.  
 Lời giải được xác định một cách trực tiếp.

* **Breatdh-First Seacrh (BFS):** duyệt theo chiều rộng

*+ Tính hiệu quả:* Hiệu quả khi lời giải nằm gần gốc của cây tìm kiếm.  
 Hiệu quả của chiến lược phụ thuộc vào độ sâu của lời giải.  
 Lời giải càng xa gốc thì hiệu quả của chiến lược càng giảm.  
 Thuận lợi khi muốn tìm nhiều lời giải.

*+ Lượng bộ nhớ sử dụng để lưu trữ các trạng thái:* Phải lưu toàn bộ các trạng thái.

*+ Trường hợp xấu nhất:* Vét cạn toàn bộ

*+ Trường hợp tốt nhất*: Vét cạn toàn bộ.

* Sự khác biệt của BFS và DFS:

So sánh BFS và DFS, lợi thế lớn của DFS là nó có yêu cầu bộ nhớ thấp hơn nhiều so với BFS, bởi vì nó không cần thiết để lưu trữ tất cả các con con trỏ ở mỗi cấp. Tùy thuộc vào dữ liệu và những gì bạn đang tìm kiếm, hoặc DFS hoặc BFS có thể là thuận lợi.

1. **KẾT QUẢ**

|  |  |  |  |
| --- | --- | --- | --- |
| Thuật toán | Đường đi | Thời gian chạy lần 1 | Thời gian chạy lần 2 |
| Breadth-First Search | s B A K S g | 0,002s | 0,002s |
| Depth-First Search | s B A K S g | 0,002s | 0,002s |

1. **KẾT LUẬN**

BFS và DFS, cả hai giải thuật thuật tìm kiếm đồ thị có thời gian chạy tương tự nhưng tiêu thụ không gian khác nhau, DFS chiếm không gian tuyến tính vì chúng ta phải nhớ đường dẫn đơn với các nút chưa được khám phá, trong khi BFS giữ mọi nút trong bộ nhớ.

DFS mang lại giải pháp sâu hơn và không tối ưu, nhưng nó hoạt động tốt khi giải pháp dày đặc trong khi BFS là tối ưu để tìm kiếm mục tiêu tối ưu lúc đầu.

1. **PHÂN CÔNG**

|  |  |  |
| --- | --- | --- |
| Tên sinh viên | Mô tả công việc | Đóng góp |
| Phạm Tấn Đạt | Viết code thuật toán DFS  Kiểm tra bài báo cáo | 45% |
| Nguyễn Ngọc Gia Bảo | Vẽ đồ thị mô phỏng lại  Viết code thuật toán BFS  Viết bài cáo cáo | 55% |

1. **TÀI LIỆU THAM KHẢO**

[1]: Wikipedia, “Tìm kiếm theo chiều rộng”, 2020

<https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_theo_chi%E1%BB%81u_r%E1%BB%99ng>

[2]: Wikipedia, “Tìm kiếm theo chiều sâu”, 2020

<https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_theo_chi%E1%BB%81u_s%C3%A2u>

[3]: gadget-info, “Sự khác biệt giữa BFS và DFS”, 2019

<https://vi.gadget-info.com/difference-between-bfs>