

# Bài tập Thực hành Nhập môn Trí tuệ Nhân tạo tuần 4

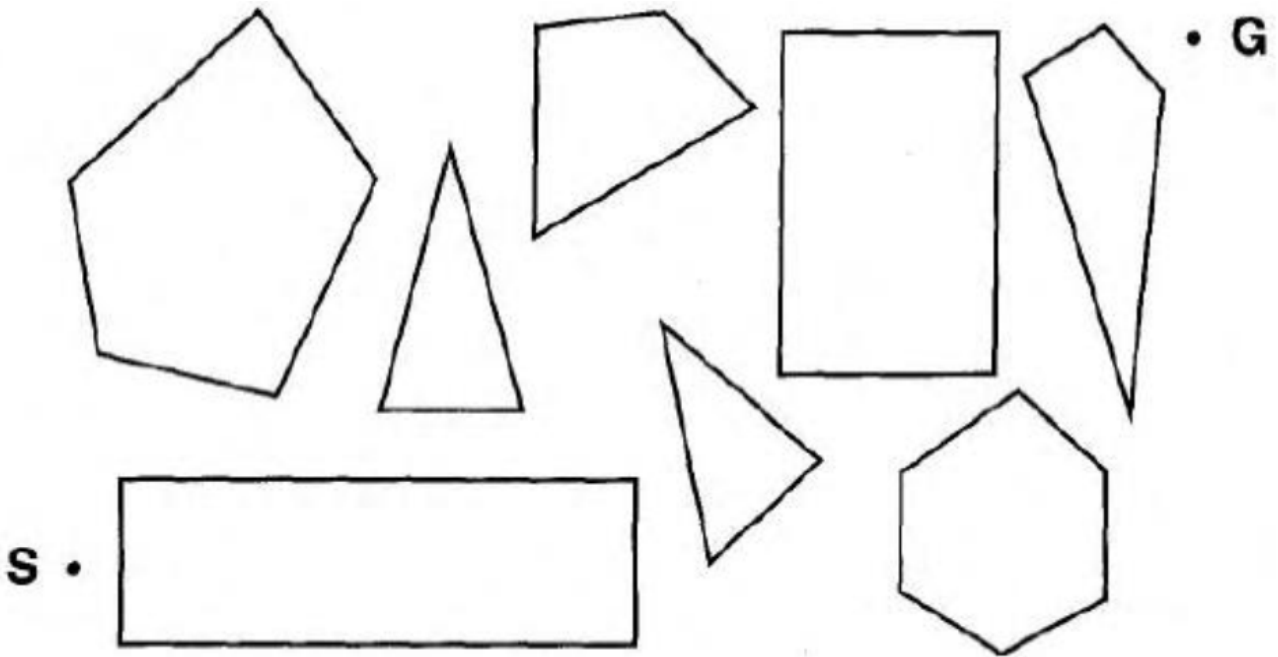
Nguyễn Lê Ngọc Duy - 20280023 - 20KDL1

## Mục lục

<b>1</b>	<b>Giới thiệu bài toán.</b>	<b>2</b>
<b>2</b>	<b>Các giải thuật tìm kiếm.</b>	<b>2</b>
<b>3</b>	<b>Giải bài toán.</b>	<b>6</b>
3.1	Trả lời câu hỏi 1a. . . . .	6
3.2	Trả lời câu hỏi 1b. . . . .	6
3.3	Trả lời câu hỏi 1c. . . . .	7
3.4	Hiệu chỉnh thuật toán. . . . .	8
<b>4</b>	<b>Cài đặt chương trình.</b>	<b>10</b>
4.1	Ghi chú. . . . .	10
4.2	Định dạng file đầu vào. . . . .	10
<b>5</b>	<b>Nhận xét.</b>	<b>11</b>

## 1 Giới thiệu bài toán.

Trong tuần này chúng ta sẽ vận dụng tất cả các thuật toán tìm kiếm đã được tìm hiểu ở các tuần trước để giải quyết bài toán được đặt ra trong tuần này như sau: Tìm đường đi ngắn nhất từ điểm xuất phát là điểm  $S$  tới điểm kết thúc là điểm  $G$  trong mặt phẳng hai chiều với các chướng ngại vật là các đa giác lồi như hình 1.



Hình 1: Hình vẽ mô tả bài toán

Trước khi tiến hành giải bài toán, ta cần nhắc lại các thuật toán tìm kiếm đã được chúng ta tìm hiểu ở các tuần trước đó.

## 2 Các giải thuật tìm kiếm.

Ở phần này, chúng ta sẽ chỉ nhắc lại mã giả của các giải thuật tìm kiếm đã được tìm hiểu ở các tuần trước đó (xem các thuật toán từ 1 đến 5).

---

**Thuật toán 1** Thuật toán Depth First Search - DFS

---

**Đầu vào:** đồ thị  $G$ , node xuất phát  $start$ , node kết thúc  $end$ .

**Đầu ra:** đường đi từ  $start$  đến  $end$

**begin**

```
1: Khởi tạo danh sách  $L$  chứa trạng thái ban đầu.  
2: while true do  
3:   if  $L$  rỗng then  
4:     Thông báo không có đường đi.  
5:     break  
6:   end if  
7: end while  
8: Loại trạng thái  $u$  ở đầu danh sách  $L$ .  
9: if  $u$  là trạng thái kết thúc then  
10:  return đường đi từ  $start$  đến  $u$ .  
11: end if  
12: Lấy các trạng thái  $v$  kề với  $u$  và thêm vào đầu danh sách  $L$ .  
13: for mỗi trạng thái  $v$  kề với  $u$  do  
14:    $father(v) = u$   
15: end for  
end
```

---

---

**Thuật toán 2** Thuật toán Breath First Search - BFS

---

**Đầu vào:** đồ thị  $G$ , node xuất phát  $start$ , node kết thúc  $end$ .

**Đầu ra:** đường đi từ  $start$  đến  $end$

**begin**

```
1: Khởi tạo danh sách  $L$  chứa trạng thái ban đầu.  
2: while true do  
3:   if  $L$  rỗng then  
4:     Thông báo không có đường đi.  
5:     break  
6:   end if  
7: end while  
8: Loại trạng thái  $u$  ở đầu danh sách  $L$ .  
9: if  $u$  là trạng thái kết thúc then  
10:  return đường đi từ  $start$  đến  $u$ .  
11: end if  
12: Lấy các trạng thái  $v$  kề với  $u$  và thêm vào cuối danh sách  $L$ .  
13: for mỗi trạng thái  $v$  kề với  $u$  do  
14:    $father(v) = u$ ;  
15: end for  
end
```

---

---

**Thuật toán 3** Thuật toán Uniform Cost Search - UCS

---

**Đầu vào:** đồ thị  $G$ , node xuất phát  $start$ , node kết thúc  $end$ .

**Đầu ra:** đường đi từ  $start$  đến  $end$  và chi phí của đường đi

**begin**

```
1: Tạo hàng đợi ưu tiên rỗng  $NganChua$ .
2: Thêm nút  $start$  vào  $NganChua$ .
3: loop
4:   if  $NganChua$  rỗng then
5:     return không tồn tại đường đi.
6:   else
7:      $Nut = \text{TimChiPhiNhoNhat}(NganChua)$ 
8:     if  $\text{KiemTraCauHoiDich}(BaiToan)$  trên  $\text{TrangThai}(Nut)$  đúng then
9:       return  $\text{LoiGiai}(Nut)$ 
10:    end if
11:  end if
12: end loop
13:  $\text{LoiGiai} = \text{Mo}(Nut, BaiToan)$ 
14:  $NganChua = \text{ThemTatCa}(\text{LoiGiai}, NganChua)$ 
```

**end**

---

---

**Thuật toán 4** Thuật toán Greedy Best First Search (GBFS)

---

**Đầu vào:** Bài toán.

**Đầu ra:** Lời giải hoặc thông báo: Không tồn tại lời giải.

```
1:  $\text{insert}(\text{state} = \text{initial\_state}, \text{priority} = 0)$  into  $\text{search.queue}$ ;
2: while  $\text{search.queue}$  not empty do
3:    $\text{current\_queue.entry} = \text{pop item from front of search.queue}$ 
4:    $\text{current\_state} = \text{current\_queue.entry.state}$ ;
5:    $\text{current\_heuristic} = \text{current\_queue.entry.heuristic}$ ;
6:    $\text{starting\_counter} = \text{counter from current\_queue.entry}$ ;
7:    $\text{applicable\_actions} = \text{array of actions applicable in current\_state}$ ;
8:   for all  $\text{index } ?i$  in  $\text{applicable\_actions}$   $\geq \text{starting\_counter}$  do
9:      $\text{current\_action} = \text{applicable\_actions}[?i]$ ;
10:     $\text{successor\_state} = \text{current\_state.apply}(\text{current\_action})$ ;
11:    if  $\text{successor\_state}$  is goal state then
12:      return  $\text{solution\_path}$ ;
13:    end if
14:     $\text{successor\_heuristic} = \text{heuristic value of successor\_state}$ ;
15:    if  $\text{successor\_heuristic} < \text{current\_heuristic}$  then
16:       $\text{insert}(\text{current\_state}, \text{current\_heuristic}, ?i + 1)$  to  $\text{search.queue}$ ;
17:       $\text{insert}(\text{successor\_state}, \text{successor\_heuristic}, 0)$  to  $\text{search.queue}$ ;
18:      break for;
19:    else
20:       $\text{insert}(\text{successor\_state}, \text{successor\_heuristic}, 0)$  to  $\text{search.queue}$ ;
21:    end if
22:  end for
23: end while
```

---

**Thuật toán 5** Thuật toán A\***Đầu vào:** Bài toán.**Đầu ra:** Lời giải hoặc thông báo: Không tồn tại lời giải.

```

1: OPEN =  $T_0$ 
2:  $g(T_0) = 0, h(T_0) = 0, f(T_0) = 0$ 
3: CLOSE =  $\emptyset$ 
4: loop
5:   if OPEN rỗng then
6:     Bài toán vô nghiệm.
7:     exit
8:   else
9:     Lấy  $T_{\max}$  ra khỏi OPEN
10:    Đưa  $T_{\max}$  vào CLOSE
11:    if  $T_{\max}$  là trạng thái đích then
12:      Lời giải là  $T_{\max}$ 
13:      exit
14:    else
15:      Tạo danh sách tất cả các trạng thái kế tiếp  $T_K$  của  $T_{\max}$ 
16:      for each  $T_K$  do
17:         $g(T_K) = g(T_{\max}) + \text{cost}(T_{\max}, T_K)$ 
18:        if tồn tại  $T_{K'}$  trong OPEN trùng với  $T_K$  then
19:          if  $g(T_K) < g(T_{K'})$  then
20:             $g(T_{K'}) = g(T_K)$ 
21:            Tính lại  $f(T_{K'})$ 
22:            FATHER( $T_{K'}$ ) =  $T_{\max}$ 
23:          end if
24:        end if
25:        if tồn tại  $T_{K'}$  trong CLOSE trùng với  $T_K$  then
26:          if  $g(T_K) < g(T_{K'})$  then
27:             $g(T_{K'}) = g(T_K)$ 
28:            Tính lại  $f(T_{K'})$ 
29:            FATHER( $T_{K'}$ ) =  $T_{\max}$ 
30:          end if
31:        end if
32:        Lan truyền sự thay đổi của  $f$  và  $g$  cho tất cả các trạng thái kế tiếp của  $T_i$  (ở tất cả các cấp) đã được lưu trữ trong CLOSE và OPEN.
33:        if  $T_K$  chưa xuất hiện trong cả OPEN và CLOSE then
34:          Thêm  $T_K$  vào OPEN
35:           $f(T_K) = g(T_K) + h(T_K)$ 
36:        end if
37:      end for
38:    end if
39:  end if
40: end loop

```

### 3 Giải bài toán.

#### 3.1 Trả lời câu hỏi 1a.

Ta có thể định nghĩa lại bài toán như sau: Cho một không gian  $W = \mathbb{R}^2$  và một tập hợp hữu hạn các đa giác lồi (cũng chính là các chương ngại vật  $O$ ), mục tiêu của chúng ta là di chuyển tác nhân (được biểu diễn bởi một điểm) từ vị trí ban đầu  $(x_s, y_s)$  tới vị trí kết thúc  $(x_g, y_g)$  sao cho đường đi được tạo thành là ngắn nhất có thể và đường đi đó phải đi qua những khoảng trống, định nghĩa bởi không gian  $F = W - O$ , hay nói cách khác, tác nhân không thể di chuyển qua các đa giác lồi  $O$ .

Khi đó tập không gian trạng thái của chúng ta chính là tất cả các vị trí  $(x, y) \in \mathbb{R}^2$  nằm trong không gian  $F$ . Do số lượng các chương ngại vật là hữu hạn, cho nên ta chỉ ra được sự tồn tại của không gian  $F$  này. Trong không gian  $\mathbb{R}^2$ , số lượng các điểm  $(x, y)$  nằm trong này là vô hạn, do đó số lượng các điểm trong không gian  $F$  cũng là vô hạn (tuy nhiên số lượng các điểm này không nhiều hơn số lượng các điểm trong không gian  $\mathbb{R}^2$ ).

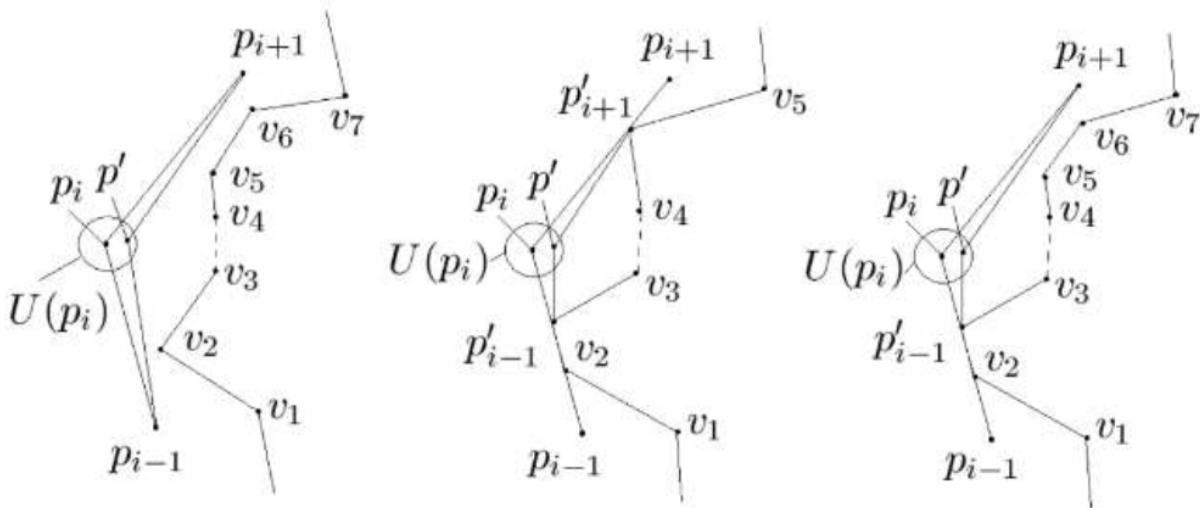
Có rất nhiều đường đi từ vị trí ban đầu  $(x_s, y_s)$  tới vị trí kết thúc  $(x_g, y_g)$ , nhưng chỉ có một đường đi tối ưu được hình thành bởi tập hợp các đỉnh thuộc các đa giác chương ngại vật và 2 cạnh nối 2 đầu của đường gấp khúc này đến lần lượt đỉnh đầu và đỉnh cuối của đường đi tối ưu.

#### 3.2 Trả lời câu hỏi 1b.

Trước khi trả lời câu hỏi này, ta cần định nghĩa đường đi ngắn nhất Euclidean: Đường đi ngắn nhất từ một đỉnh  $p$  nằm trên đa giác đến đỉnh  $q$  bất kì nào khác được biểu diễn bởi  $\rho = \langle p, p_1, p_2, \dots, p_k, q \rangle$ , trong đó mỗi đỉnh  $p_1, p_2, \dots, p_k$  là một đỉnh của một đa giác. (\*)

Ta cần chứng minh rằng nhận định trên (cũng chính là ý của câu hỏi 1b) là đúng. Để chứng minh nhận định này, ta sẽ dùng phương pháp phản chứng. Đầu tiên, ta giả sử rằng có ít nhất một đỉnh  $p_i$  nào đó trong đường đi ngắn nhất *không* là đỉnh của đa giác. Ta xét các đỉnh từng đôi một không thẳng hàng, nghĩa là 3 đỉnh bất kì luôn tạo thành một tam giác.

Ta chia thành 3 trường hợp như hình 2:



Hình 2: Minh họa ba trường hợp cần chứng minh

**Trường hợp 1:** Cả hai cạnh  $p_{i-1}p_i$  và  $p_i p_{i+1}$  đều không là tiếp tuyến của đa giác hay không tiếp xúc với bất cứ đỉnh nào của đa giác

Dựa vào hình vẽ, tồn tại một vùng nhỏ  $U$  quanh  $p_i$  sao cho tồn tại một điểm  $p'_i \in U$  thoả mãn

$$d(p_{i-1}, p'_i) + d(p'_i, p_{i+1}) < d(p_{i-1}, p_i) + d(p_i, p_{i+1})$$

Suy ra nếu ta thay  $p_i$  bởi  $p'_i$ , ta có thể thu được một đường đi ngắn hơn, mâu thuẫn với giả sử  $\rho$  là đường đi ngắn nhất.

**Trường hợp 2: Cả hai cạnh  $p_{i-1}p_i$  và  $p_i p_{i+1}$  đều nằm trên cạnh của đa giác hay tiếp xúc với bất cứ đỉnh nào của đa giác**

Xét 2 đỉnh  $p'_{i-1}$  và  $p'_{i+1}$  gần nhất sao cho  $p'_{i-1}p_i$  và  $p_i p'_{i+1}$  là tiếp tuyến của đa giác. Tương tự **Trường hợp 1**, ta có thể tìm được một điểm  $p'_i$  sao cho

$$d(p_{i-1}, p'_i) + d(p'_i, p_{i+1}) < d(p_{i-1}, p_i) + d(p_i, p_{i+1})$$

Điều này mâu thuẫn với giả sử  $\rho$  là đường đi ngắn nhất.

**Trường hợp 3: Trong hai cạnh  $p_{i-1}p_i$  và  $p_i p_{i+1}$  chỉ có một cạnh là tiếp tuyến của đa giác**

Trường hợp này là sự kết hợp của hai trường hợp trên, do đó hoàn toàn tương tự, ta cũng thu được điều mâu thuẫn với giả sử.

Như vậy, ta đã chứng minh được rằng giả sử  $\rho$  là đường đi ngắn nhất là sai. Do đó, ta đã chứng minh được (\*) đúng.

Từ nhận xét đó, ta nhận thấy rằng để có thể đi đến điểm cuối (bằng cách đi qua các đỉnh của các đa giác) bằng đường đi ngắn nhất, ta chỉ cần xem xét các đỉnh của các đa giác, từ đó có thể yên tâm bỏ qua các điểm  $(x, y)$  khác trong không gian  $F$ . Nói cách khác, ta có thể định nghĩa lại không gian trạng thái như sau:  $S = \{(x, y) \in \mathbb{R}^2 | (x, y) \text{ là một đỉnh của } O\}$ . Độ lớn của không gian này chính là số lượng các đỉnh của tập đa giác chướng ngại  $O$ , và không gian này hữu hạn cũng như là nhỏ hơn rất nhiều so với không gian  $F$ .

### 3.3 Trả lời câu hỏi 1c.

Để có thể giải quyết bài toán, trước hết ta cần xác định xem các đỉnh nào có thể nhìn thấy và đi đến được từ một điểm bất kỳ nào đó trên đồ thị. Để làm được điều đó, trước tiên ta cần xác định những điểm có thể *nhìn thấy* được.

Trước tiên, ta nhắc lại định nghĩa về phương trình đường thẳng. Cho 2 điểm  $A(x_1, y_1)$  và  $B(x_2, y_2)$ .

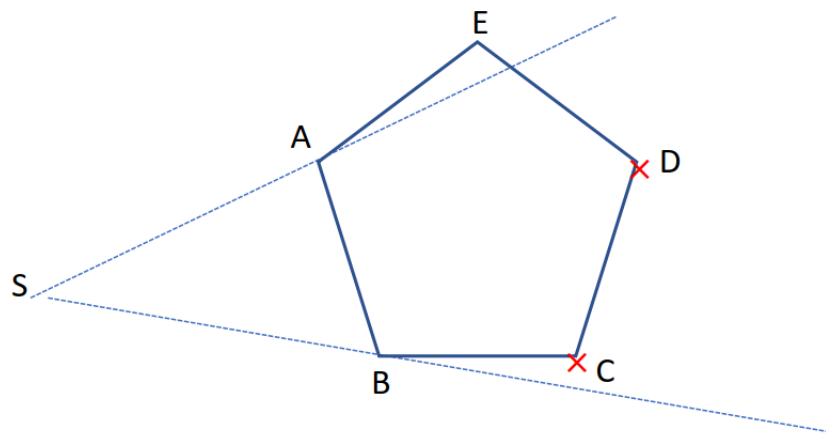
Khi đó phương trình đường thẳng tạo bởi đoạn thẳng  $AB$  có dạng:  $(d) : \frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}$ .

Xét hai điểm  $C(x_3, y_3)$  và  $D(x_4, y_4)$ . Khi đó, ta có thể xác định được vị trí tương đối của hai điểm này đối với đường thẳng  $(d)$  như sau:

- $d(C)d(D) \geq 0 \Rightarrow C$  và  $D$  nằm cùng phía đối với đường thẳng  $(d)$ .
- $d(C)d(D) \leq 0 \Rightarrow C$  và  $D$  nằm khác phía đối với đường thẳng  $(d)$ .

Bây giờ ta xét bài toán tìm những điểm thấy được từ một điểm cho trước. Ta sẽ dùng ý tưởng tìm các điểm *nhìn thấy* được từ một đỉnh đến một đa giác như sau: Từ đỉnh ta đang xét (giả sử là đỉnh  $S$ ) và các cạnh của đa giác (giả sử các cạnh đó lần lượt là  $AB, BC, CD, DE, EA$ ), ta xét lần lượt các cạnh, giả sử là cạnh  $AB$ . Những điểm nào nằm trong cung  $ASB$  sẽ là những điểm không nhìn thấy được, những điểm đó được đánh dấu x đỏ như hình 3, các điểm còn lại là các điểm thấy được.

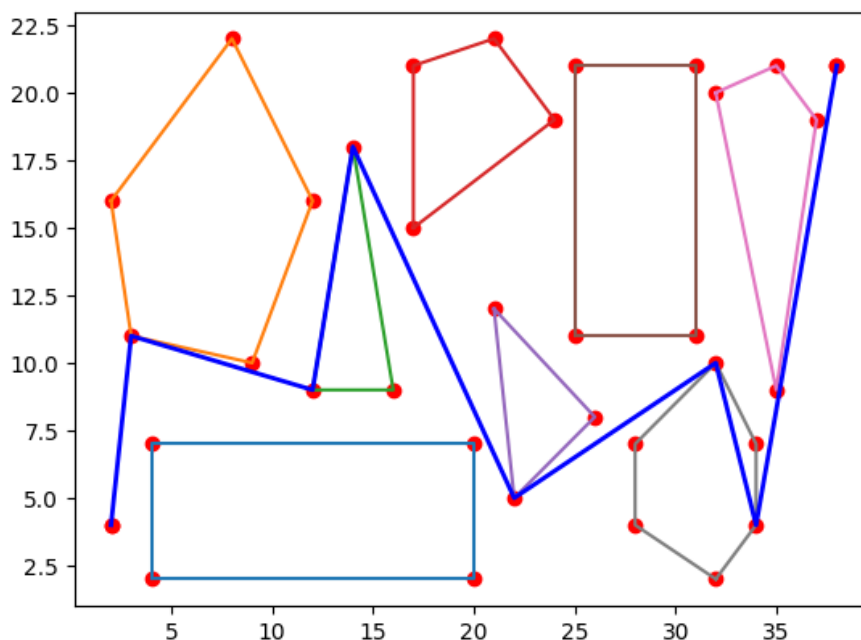
Như vậy, từ ý tưởng trên, ta có thể tìm được tất cả các điểm *nhìn thấy* được từ một đỉnh cho trước đến các đa giác lân cận như thuật toán 6 ở bên dưới.



Hình 3: Minh họa ý tưởng tìm các điểm *nhìn thấy* được từ một đỉnh đến một đa giác

### 3.4 Hiệu chỉnh thuật toán.

Một vấn đề của thuật toán trên chính là thuật toán trên chỉ giúp ta tìm được những điểm *nhìn thấy* hay có thể đi qua được. Trên thực tế, không phải điểm nào *nhìn thấy* được là có thể *đi qua* được. Một trường hợp như thế được minh họa như hình 4. Ta có thể thấy được đỉnh có tọa độ  $(31, 11)$  *nhìn thấy* được đỉnh có tọa độ  $(34, 3)$ , tuy nhiên ta không thể đi từ đỉnh có tọa độ  $(31, 11)$  đến đỉnh có tọa độ  $(34, 3)$  vì hai đỉnh này là đường chéo của lục giác.



Hình 4: Một đường đi không hợp lệ

Để có thể giải quyết vấn đề trên, sau khi ta xác định được các điểm *nhìn thấy* được, ta sẽ loại bỏ đi những đỉnh tạo với đỉnh đang xét một đường chéo của đa giác lân cận, từ đó có thuật toán 7. Kết hợp



---

**Thuật toán 6** Thuật toán tìm tất cả các điểm *nhìn thấy* được

---

**Đầu vào:** Đỉnh  $S$  và các đa giác lân cận  $O_1, O_2, \dots, O_n$ .

**Đầu ra:** Tập hợp  $V_{see}$  các đỉnh nhìn thấy được.

```

1:  $V =$  tập hợp tất cả các đỉnh của các đa giác  $O_1, O_2, \dots, O_n$ .
2:  $E =$  tập hợp tất cả các cạnh của các đa giác  $O_1, O_2, \dots, O_n$ .
3:  $V_{see} = \emptyset$ 
4: for each cạnh  $AB$  trong  $E$  do
5:    $d_1 =$  đường thẳng đi qua  $S$  và  $A$ .
6:    $d_2 =$  đường thẳng đi qua  $S$  và  $B$ .
7:    $d_3 =$  đường thẳng đi qua  $A$  và  $B$ .
8:   for each đỉnh  $Q \in V \setminus \{S, A, B\}$  do
9:     if ( $Q$  và  $B$  nằm cùng phía với  $d_1$ )  $\wedge$  ( $Q$  và  $A$  nằm cùng phía với  $d_2$ )  $\wedge$  ( $Q$  và  $S$  nằm khác
        phía với  $d_3$ ) then
10:        $Q$  là đỉnh không nhìn thấy được từ  $P$ .
11:     else
12:        $Q$  là đỉnh nhìn thấy được từ  $P$ .
13:        $V_{see} = V_{see} \cup \{Q\}$ 
14:     end if
15:   end for
16: end for
```

---

với các thuật toán tìm kiếm ở trên, ta có thể tiến hành giải quyết bài toán đã cho. Để minh họa, ta sẽ giải quyết bài toán trên với đỉnh đầu tiên là đỉnh có tọa độ  $(2, 4)$  và đỉnh kết thúc có tọa độ  $(38, 21)$ .

---

**Thuật toán 7** Thuật toán tìm tất cả các điểm *đi qua* được

---

**Đầu vào:** Đỉnh  $S$  và các đa giác lân cận  $O_1, O_2, \dots, O_n$ .

**Đầu ra:** Tập hợp  $V_{pass}$  các đỉnh có thể đi qua được.

```

1:  $V_{see} =$  tập hợp các đỉnh nhìn thấy được từ  $S$ .
2:  $V_{pass} = \emptyset$ 
3: for each đỉnh  $Q \in V_{see}$  do
4:   if  $Q$  không nằm trên đường chéo của đa giác lân cận then
5:      $Q$  là đỉnh có thể đi qua được.
6:      $V_{pass} = V_{pass} \cup \{Q\}$ 
7:   end if
8: end for
```

---

## 4 Cài đặt chương trình.

### 4.1 Ghi chú.

Chương trình sử dụng ngôn ngữ Python phiên bản 3.9.13, với file đầu vào là `input.txt` và mã nguồn được lưu trong file `script.py`.

### 4.2 Định dạng file đầu vào.

Dữ liệu của file đầu vào gồm (xem hình 5):

- Dòng thứ nhất gồm 5 số nguyên  $N, S_x, S_y, G_x, G_y$  cách nhau bởi khoảng trắng, trong đó:
  - $N$  là số đa giác nằm trong mặt phẳng ( $0 \leq N \leq 100$ ).
  - $(S_x, S_y)$  là toạ độ đỉnh xuất phát.
  - $(G_x, G_y)$  là toạ độ đỉnh kết thúc.
- $N$  dòng tiếp theo gồm các số nguyên  $M, X_1, Y_1, \dots, X_M, Y_M$  cách nhau bởi khoảng trắng, trong đó:
  - $M$  là số đỉnh của đa giác.
  - $(X_i, Y_i)$  là toạ độ thực của đỉnh thứ  $i$  trong đa giác.

```

1  8 2 4 38 21
2  4 4 2 4 7 20 7 20 2
3  5 2 16 3 11 9 10 12 16 8 22
4  3 12 9 16 9 14 18
5  4 17 15 17 21 21 22 24 19
6  3 21 12 22 5 26 8
7  4 25 11 31 11 31 21 25 21
8  4 32 20 35 21 37 19 35 9
9  6 32 2 28 4 28 7 32 10 34 7 34 4

```

Hình 5: Định dạng file đầu vào `input.txt`.

## 5 Nhận xét.