Tên: Phan Hồng Trâm

MSSV: 21110414

Báo cáo

Thực hành Nhập môn Trí tuệ nhân tạo tuần 4

Viết báo cáo trình bày:

- **Nếu chương trình bị báo lỗi thì lỗi ở dòng nào và sửa lại như thế nào? (nếu có)**

- Chương trình bị báo lỗi ở hàm **`main()`**, cụ thể ở dòng 220 vì lỗi ký tự khi đọc file `'Input.txt'`.

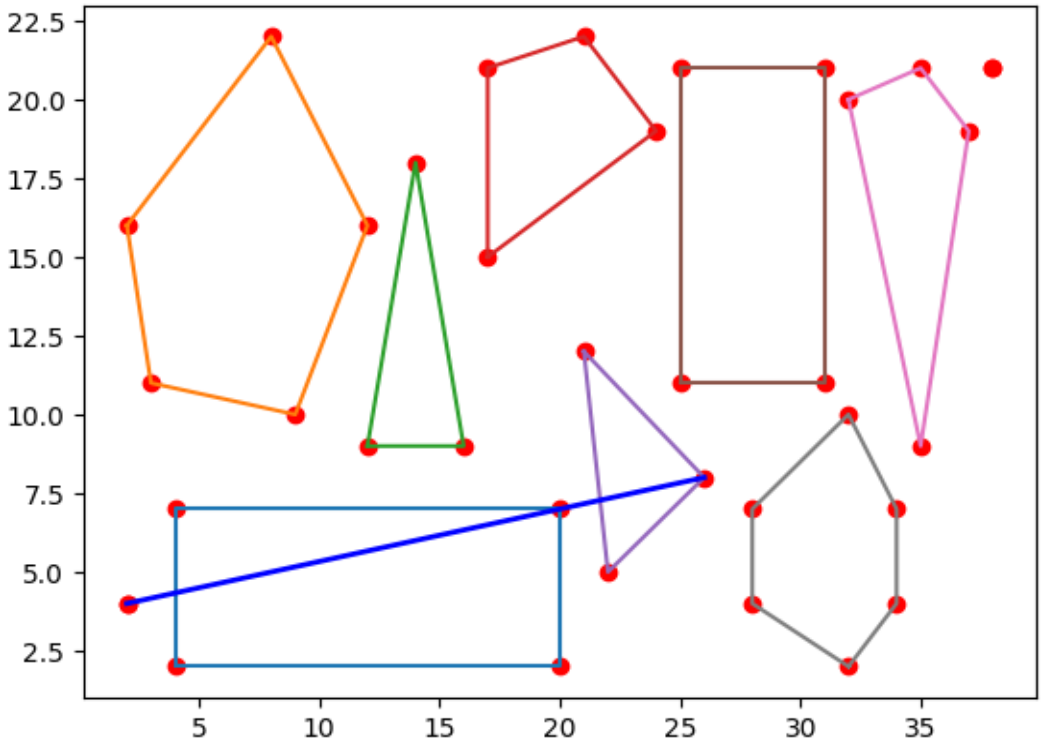
```
215 def main():
216     poly_list = list(list())
217     x = list()
218     y = list()
219     n_poygon = 0
220     with open('Input.txt', 'r') as f:
221         line = f.readline()
222         line = line.strip()
223         line = line.split()
224         line = list(map(int, line))
225         n_polygon = line[0]
```

➔ Sửa: Thêm **`encoding = 'utf-8-sig'`**

```
285     with open('Input.txt', 'r', encoding = 'utf-8-sig') as f:
286         line = f.readline()
287         line = line.strip()
288         line = line.split()
289         line = list(map(int, line))
```

❖ **Các thuật toán đã cho sẵn code như trên chạy ra kết quả đúng không?Nếu chưa đúng thì em sửa lại như thế nào cho phù hợp?**

- Sau khi sửa lỗi trên, chương trình chạy được nhưng ra kết quả sai:



- Lí do: Lỗi vị trí căn lề của các dòng **`if-else`** trong **`def can_see`** của **`class Graph`**, điều đó dẫn đến sai về lỗi logic. Chương trình chạy chỉ xét đến nút **`start`** và điều kiện nếu **`point not in see list`** thì thêm **`point`** vào **`see_list`**. Mà **không xét** đến cái **`point`** khác làm cho chương trình chỉ chạy đến điểm thứ hai là ngừng.

```
def can_see(self, start):
    see_list = list()
    cant_see_list = list()

    for polygon in self.polygons:
        for edge in self.polygons[polygon]:
            for point in self.get_points():
                if start == point:
                    cant_see_list.append(point)
                if start in self.get_polygon_points(polygon):
                    for poly_point in self.get_polygon_points(polygon):
                        if poly_point not in self.get_adjacent_points(start):
                            cant_see_list.append(poly_point)
                if point not in cant_see_list:
                    if start.can_see(point, edge):
                            if point not in see_list:
                                see_list.append(point)
                        elif point in see_list:
                            see_list.remove(point)
                            cant_see_list.append(point)
                        else:
                            cant_see_list.append(point)
        return see_list
```

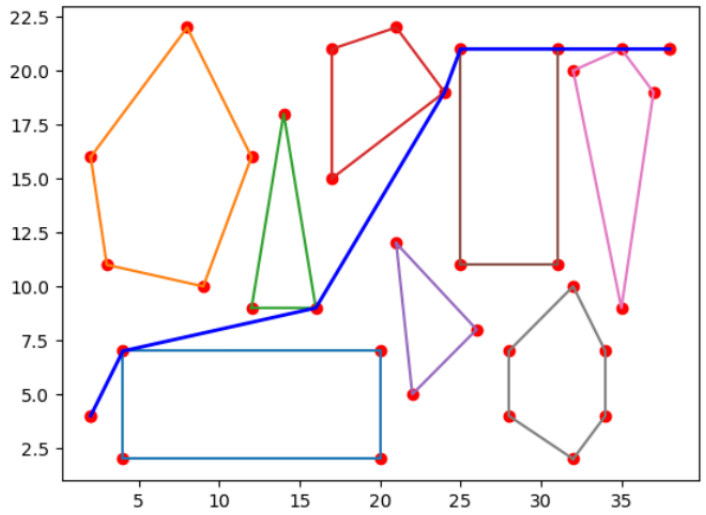
➔ Sửa lỗi: Chúng ta căn lề lại các điều kiện **`if-else`** cho hàm **`can_see`**

```
103     def can_see(self, start):
104         see_list = list()
105         cant_see_list = list()
106 
107         for polygon in self.polygons:
108             for edge in self.polygons[polygon]:
109                 for point in self.get_points():
110                     if start == point:
111                         cant_see_list.append(point)
112                     if start in self.get_polygon_points(polygon):
113                         for poly_point in self.get_polygon_points(polygon):
114                             if poly_point not in self.get_adjacent_points(start):
115                                 if poly_point not in see_list:
116                                     cant_see_list.append(poly_point)
117                                 else:
118                                     see_list.remove(poly_point)
119                     if point not in cant_see_list:
120                         if start.can_see(point, edge):
121                             if point not in see_list:
122                                 see_list.append(point)
123                         elif point in see_list:
124                             see_list.remove(point)
125                             cant_see_list.append(point)
126                         else:
127                             cant_see_list.append(point)
128         return see_list
```

➲ Khi đó, chương trình hết báo lỗi và chạy ra kết quả đúng:

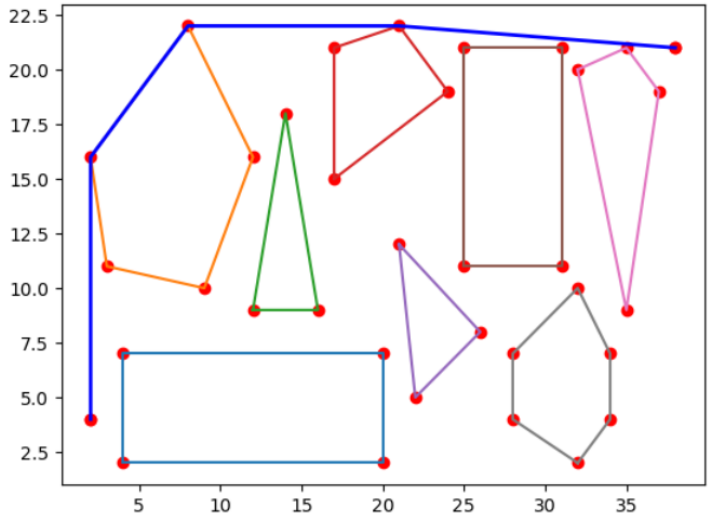
➢ A star:

```
[(2, 4), -1] -> [(4, 7), 0] -> [(16, 9), 2] -> [(24, 19), 3] -> [(25, 21), 5] -> [(31, 21), 5] -> [(35, 21), 6] -> [(38, 21), -
1]
```



➢ Greedy:

```
[(2, 4), -1] -> [(2, 16), 1] -> [(8, 22), 1] -> [(21, 22), 3] -> [(38, 21), -1]
```



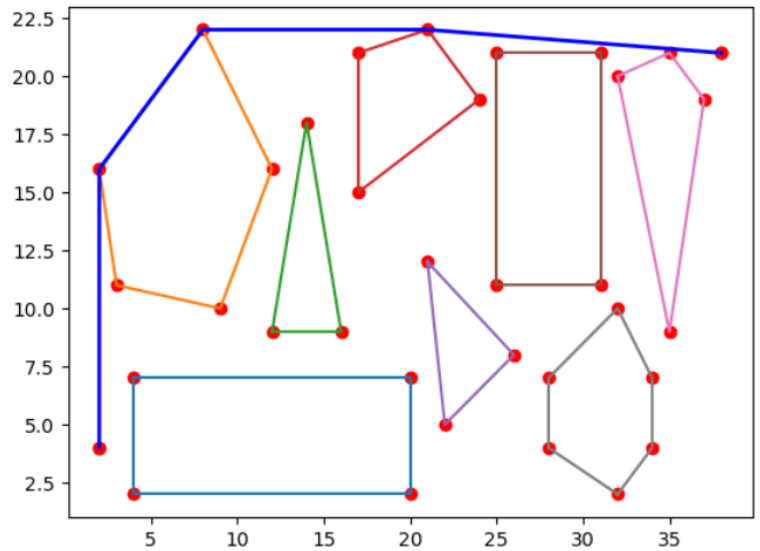
❖ **Áp dụng bài toán với các thuật toán BFS, DFS và UCS và cài đặt chúng trên máy tính.**

➢ BFS:

```
204  def bfs(graph, start, goal):
205      visited = defaultdict(bool)
206      frontier = Queue()
207
208      frontier.put(start)
209      visited[start] =  True
210
211      parent = dict()
212      parent[start] = None
213
214      while True:
215          if frontier.empty():
216              raise Exception('No way Exception')
217          current_node = frontier.get()
218          visited[current_node] = True
219
220          # Kiểm tra current_node có là end hay không
221          if current_node == goal:
222              return current_node
223
224          for node in graph.can_see(current_node):
225              if not visited[node]:
226                  frontier.put(node)
227                  node.pre = current_node
228                  visited[node] = True
```

➲ Kết quả thuật toán BFS:

```
[(2, 4), -1] -> [(2, 16), 1] -> [(8, 22), 1] -> [(21, 22), 3] -> [(38, 21), -1]
```

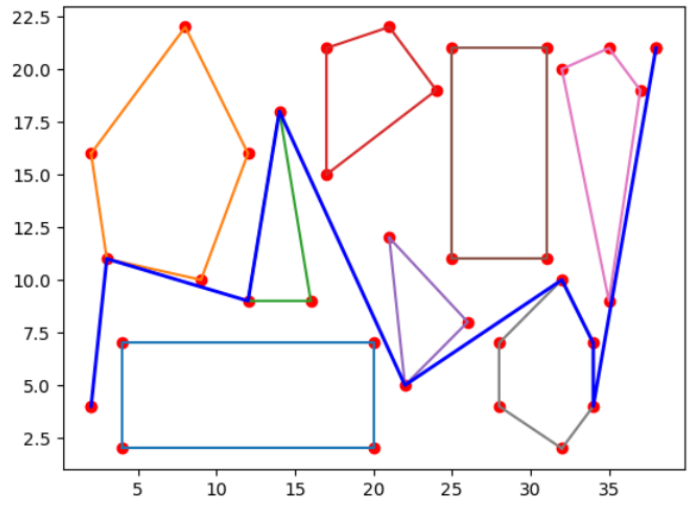


➢ DFS:

```
230  def dfs(graph, start, goal):
231      visited = defaultdict(bool)
232      frontier = []
233
234      # them node start vào frontier và visited
235      frontier.append(start)
236      visited[start] =  True
237
238      # start không có trong node cha
239      parent = dict()
240      parent[start] = None
241      while True:
242          if not frontier:
243              raise Exception('No way Exception')
244          current_node = frontier.pop()
245          visited[current_node] = True
246          # Kiểm tra current node có là end hay không
247          if current_node == goal:
248              return current_node
249
250          for node in graph.can_see(current_node):
251              if not visited[node]:
252                  frontier.append(node)
253                  node.pre = current_node
254                  visited[node] = True
```

➲ Kết quả thuật toán DFS:

```
[(2, 4), -1] -> [(3, 11), 1] -> [(12, 9), 2] -> [(14, 18), 2] -> [(22, 5), 4] -> [(32, 10), 7] -> [(34, 7), 7] -> [(34, 4), 7]
-> [(38, 21), -1]
```



➢ UCS:

```
256  def ucs(graph, start, goal):
257      visited = defaultdict(bool)
258      frontier = PriorityQueue()
259      # thêm node start vào frontier và visited
260      frontier.put((0, start))
261      visited[start] = True
262      # start không có node cha
263      parent = dict()
264      parent[start] = None
265
266      while True:
267          if frontier.empty():
268              raise Exception('No way Exception')
269          current_cost, current_node = frontier.get()
270          visited[current_node] = True
271
272          # kiểm tra current node có phải là end hay không
273          if current_node == goal:
274              return current_node
275          for nodei in graph.can_see(current_node):
276              new_cost = current_cost + euclid_distance(current_node, nodei)
277              if not visited[nodei] or new_cost < nodei.g:
278                  frontier.put((new_cost, nodei))
279                  nodei.g = new_cost
280                  nodei.pre = current_node
281                  visited[nodei] = True
```

➲ Kết quả thuật toán UCS:

```
[(2, 4), -1] -> [(4, 7), 0] -> [(16, 9), 2] -> [(24, 19), 3] -> [(25, 21), 5] -> [(31, 21), 5] -> [(35, 21), 6] -> [(38, 21), -1]
```