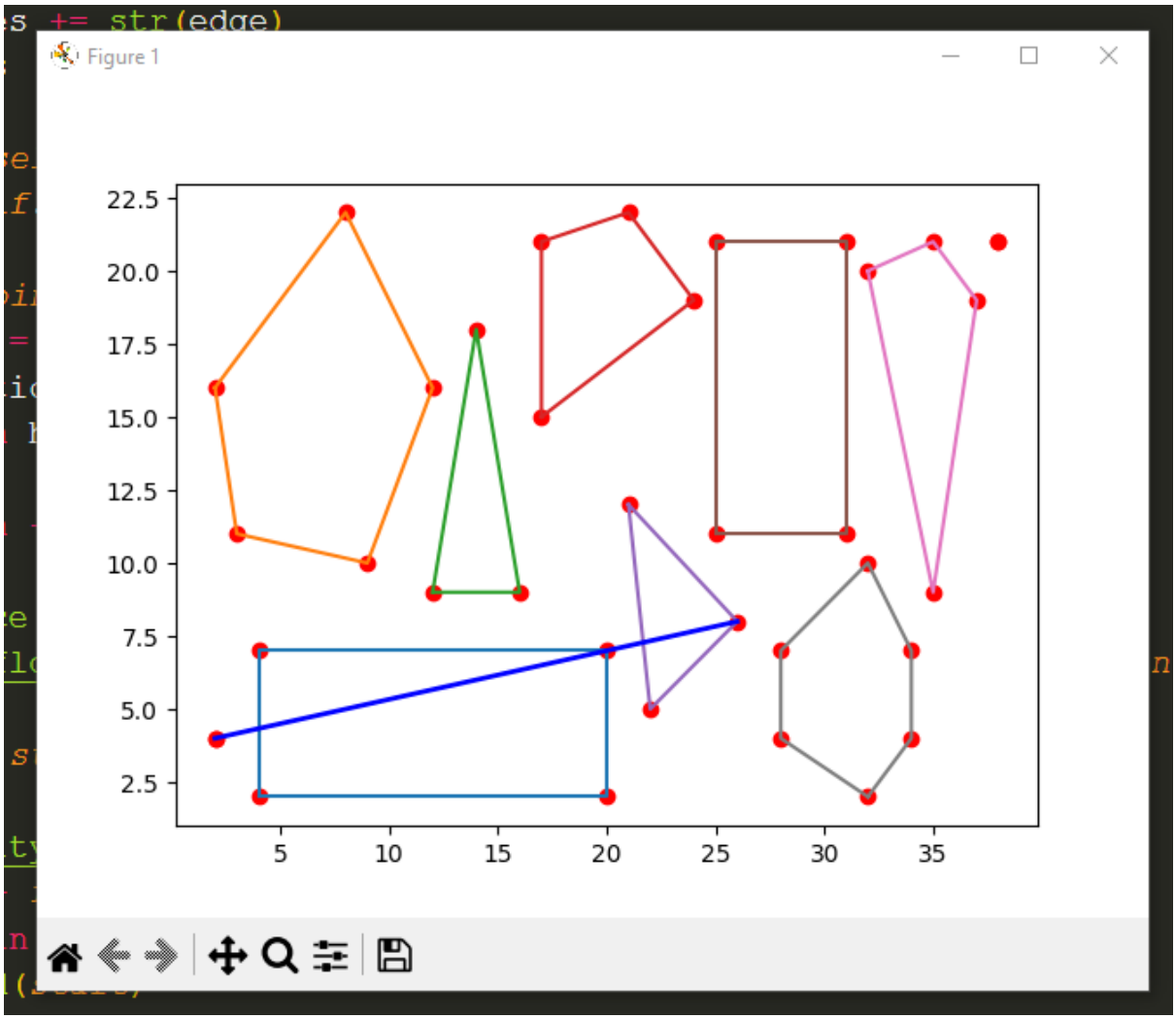
21110324 – Lương Đăng Khôi

- Sau khi có được file code và file Input.txt ta tiến hành chạy thử và được kết quả như sau:



Ta thấy có 2 vấn đề:
+ 1 là lộ trình kết thúc trước khi đến được điểm goal.
+ 2 là lộ trình bất thường khi đi xuyên qua các cạnh.
=> Điều này cho thấy hàm xét các điểm mù hoặc hàm xét các điểm nhìn thấy bị lỗi và cũng như thuật toán tìm lộ trình bị sai dẫn đến goal bị sai.

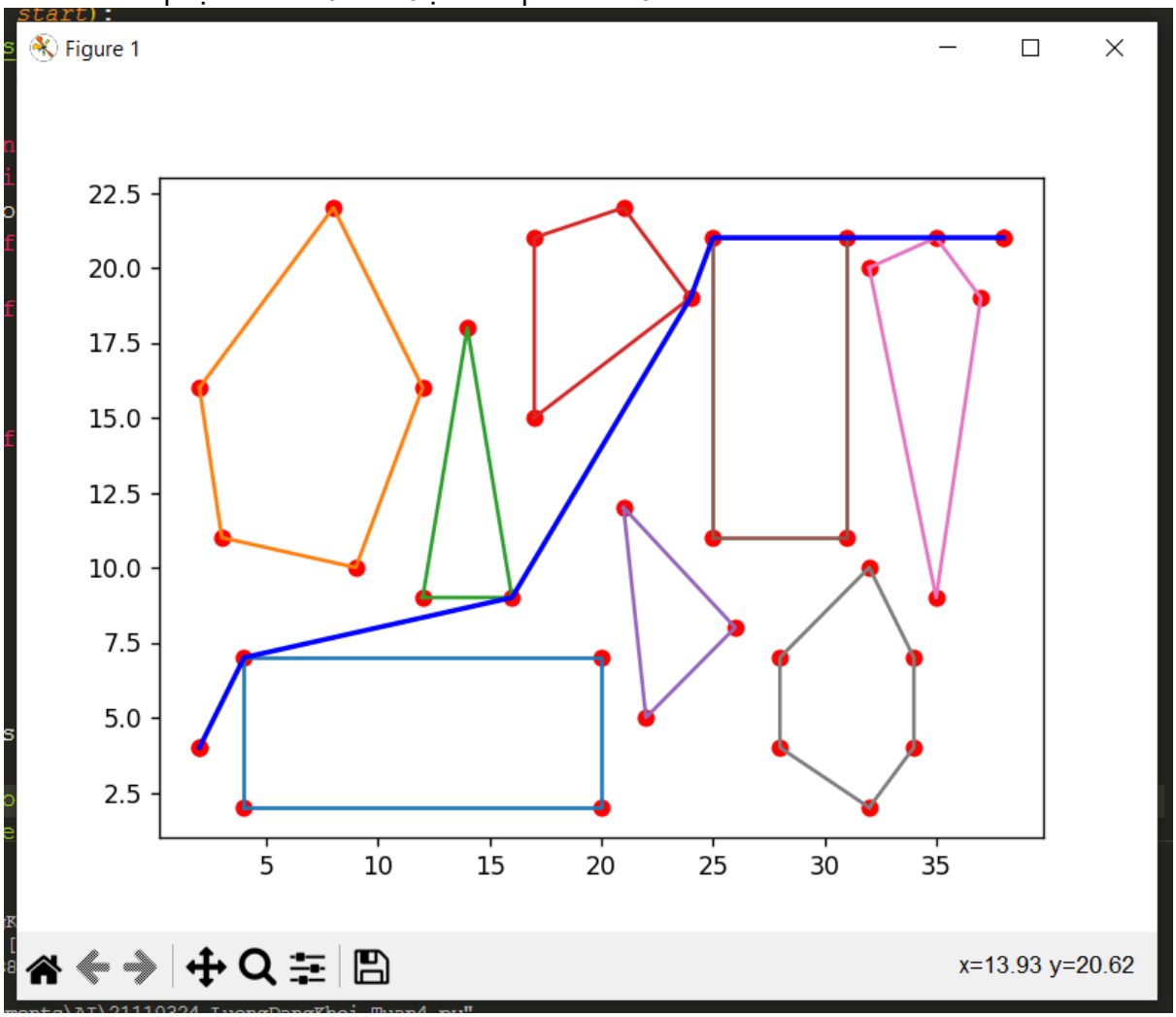
Tiến hành kiểm tra phương thức ‘can_see’ trong class Graph như sau:

```
102
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                                 cant_see_list.append(poly_point)
116                     if point not in cant_see_list:
117                         if start.can_see(point, edge):
118                             if point not in see_list:
119                                 see_list.append(point)
120                             elif point in see_list:
121                                 see_list.remove(point)
122                                 cant_see_list.append(point)
123                             else:
124                                 cant_see_list.append(point)
125         return see_list
126
```

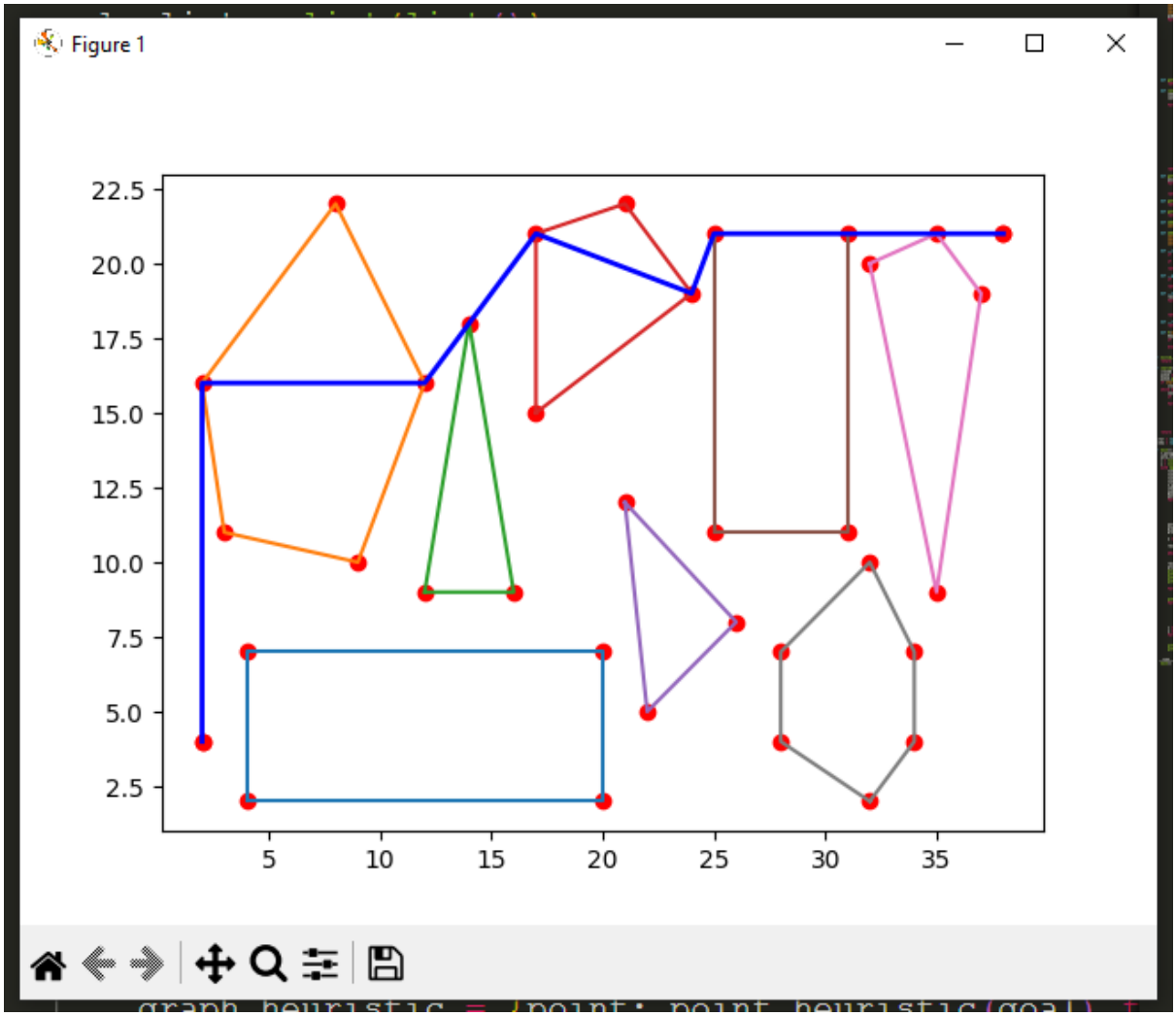
- Đầu tiên dễ thấy nhất là trong phương thức này return see_list không đúng chỗ vì nếu return ở vị trí đó thì vòng lặp chỉ chạy đúng 1 lần (vòng lặp vô nghĩa nếu luôn chỉ chạy 1 lần). Thế nên ta sửa lại return cùng cấp với vòng lặp for đầu tiên.
- Tiếp theo ta thấy vấn đề ở câu lệnh điều kiện ‘if point not in can’t_see_list:’ ta thấy dòng elif chứa điều kiện ‘point in see_list’ phải nhỏ hơn câu điều kiện trên chỉ 1 cấp nên ta sửa lại như hình bên dưới.

```
102
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                                 cant_see_list.append(poly_point)
116                     if point not in cant_see_list:
117                         if start.can_see(point, edge):
118                             if point not in see_list:
119                                 see_list.append(point)
120                         elif point in see_list:
121                             see_list.remove(point)
122                             cant_see_list.append(point)
123                         else:
124                             cant_see_list.append(point)
125         return see_list
126
```

Sau đó ta tiếp tục kiểm thử và được kết quả bên dưới:



- Kết quả trên là lộ trình tìm được bởi thuật toán A_star. Tiếp tục kiểm tra khi dùng greedy và kết quả như sau:



- Ta thấy điểm đầu đã đến được điểm cuối, vấn đề là lộ trình đã đi xuyên các vật cản, như vậy ta thấy một lần nữa phương thức 'can_see' có sai sót.

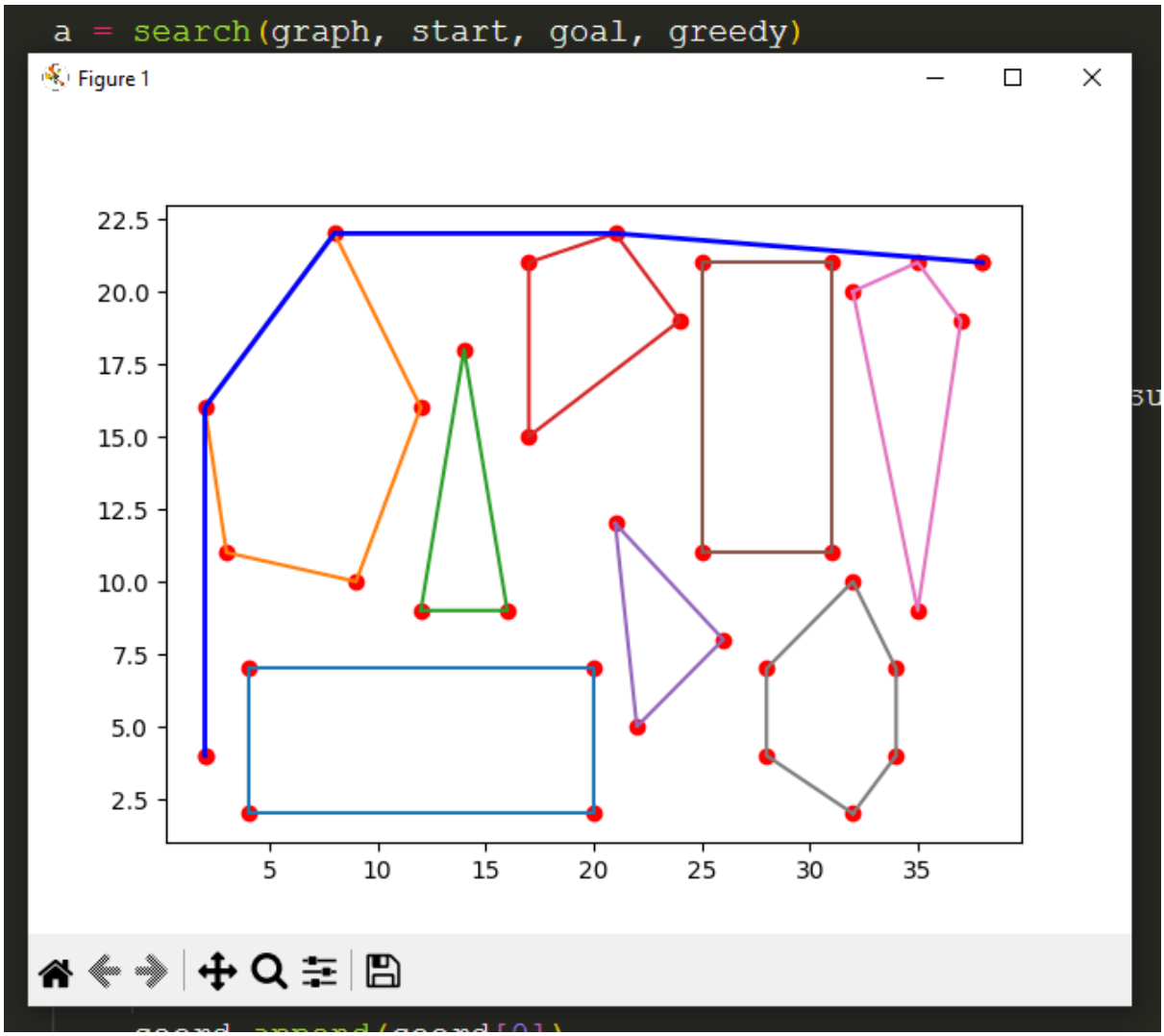
- Sau khi xem xét lại phương thức trên ta thấy vấn đề là việc thêm 'poly_point' vào 'cant_see_list' bị thiếu điều kiện, bởi vì sẽ có trường hợp một điểm đã có trong see_list nhưng lại bị thêm vào cant_see_list dẫn đến dữ liệu về các điểm mù và các điểm nhìn thấy bị sai do đó có hiện tượng đi xuyên vật cản như hình trên. Cách khắc phục là ta sẽ thêm câu điều kiện để kiểm tra xem điểm 'poly_point' nếu nằm trong 'cant_see_list' thì không nằm trong 'see_list' được và ngược lại.

```
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)
```

- Từ code cũ ở trên ta thêm điều kiện như đã nói thành code mới như ảnh bên dưới.

```
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):
                            if poly_point not in see_list:
                                cant_see_list.append(poly_point)
                            else:
                                see_list.remove(poly_point)
```

- Sau khi sửa lại phương thức trên ta kiểm tra lại kết quả:



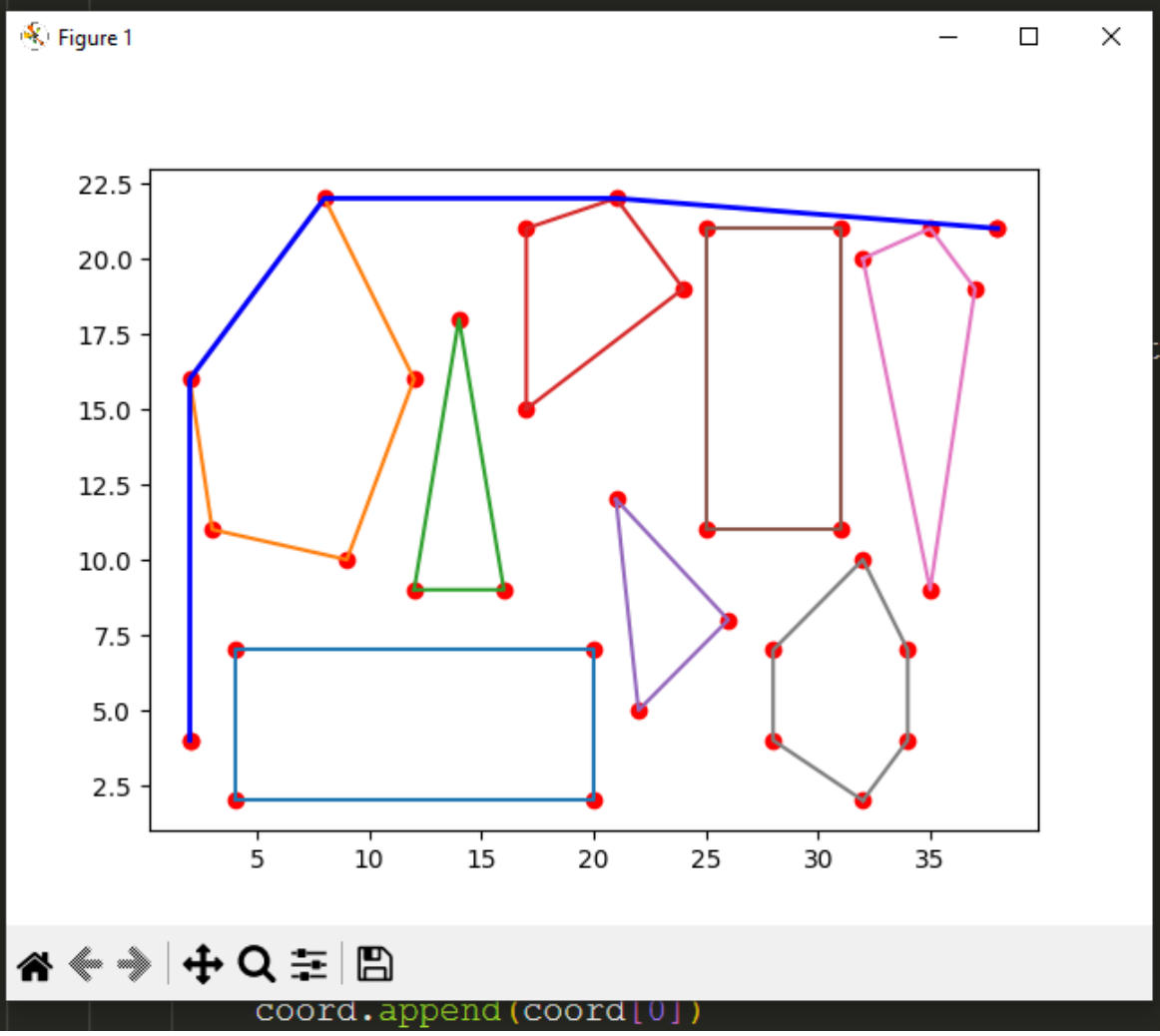
- Như vậy A_star và Greedy đã hoạt động bình thường và cho ra kết quả đúng với các giao ước ban đầu. Tiếp tục dùng các kiến thức đã học, thêm thuật toán BFS, DFS và UCS

*BFS:

```
def bfs(graph, start, goal):
    visited = []
    frontier = Queue()

    frontier.put(start)
    visited.append(start)

    parent = dict()
    parent[start] = None

    while True:
        if frontier.empty():
            raise Exception('No way Exception')
        current_node = frontier.get()
        visited.append(current_node)

        if current_node == goal:
            return current_node

        for node in graph.can_see(current_node):
            if node not in visited:
                frontier.put(node)
                node.pre = current_node
                visited.append(node)
```

- Kết quả chạy:

```
a = bfs(graph, start, goal)
```



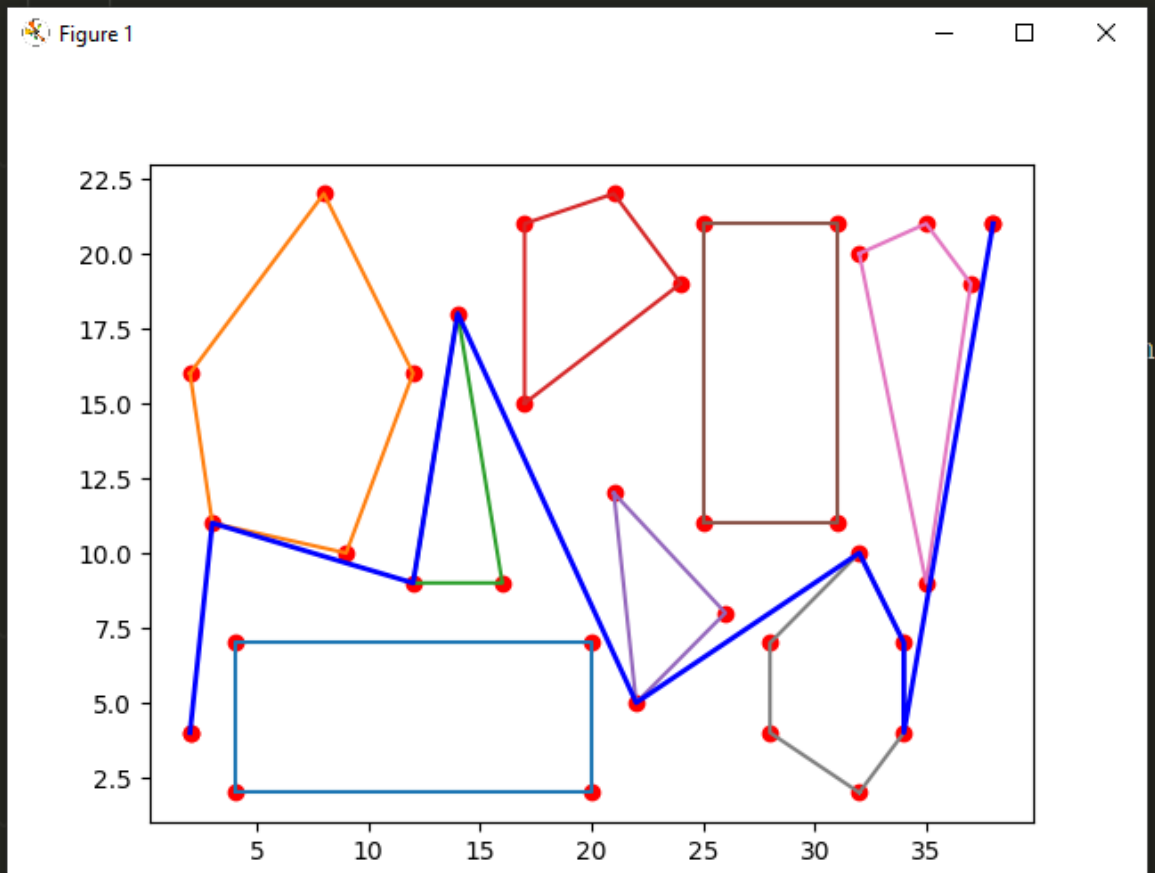

```
coord.append(coord[0])
```

* DFS:

```
def dfs(graph, start, goal):
    visited = []
    frontier = []


    frontier.append(start)
    visited.append(start)


    parent = dict()
    parent[start] = None
    while True:
        if frontier == []:
            raise Exception('No way Exception')
        current_node = frontier.pop()
        visited.append(current_node)

        if current_node == goal:
            return current_node

        for node in graph.can_see(current_node):
            if node not in visited:
                frontier.append(node)
                node.pre = current_node
                visited.append(node)
```

- Kết quả chạy:

```
a = dfs(graph, start, goal)
```



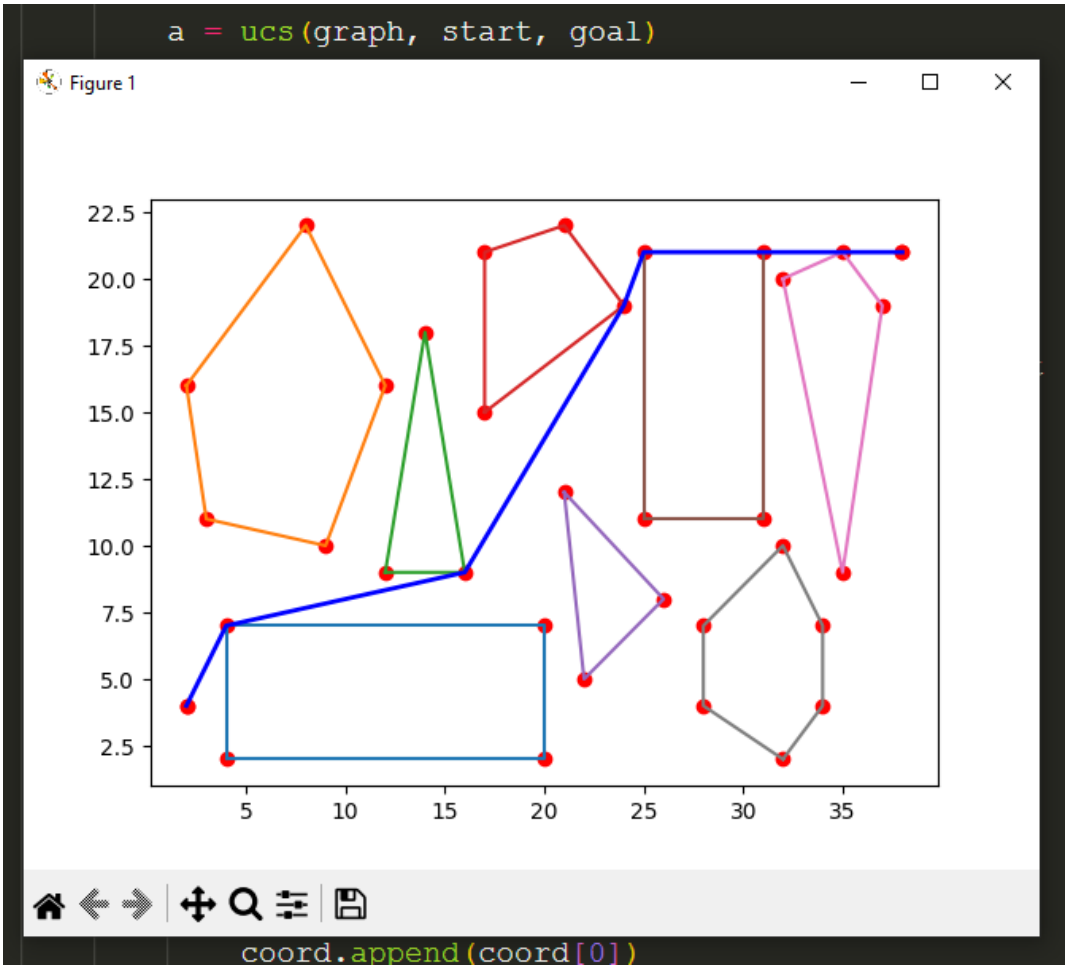

```
coord.append(coord[0])
```

*UCS:

```
def ucs(graph, start, goal):
    visited = []
    frontier = PriorityQueue()

    frontier.put((0, start))
    visited.append(start)

    parent = dict()
    parent[start] = None
    path_found = False
    while True:
        cost, current_node = frontier.get()
        visited.append(current_node)

        if current_node == goal:
            return current_node
        for node in graph.can_see(current_node):
            new_cost = current_node.g + euclid_distance(current_node, node)
            if node not in visited or new_cost < node.g:
                frontier.put((new_cost, node))
                node.g = new_cost
                node.pre = current_node
                visited.append(node)
```

- Kết quả chạy:

```
a = ucs(graph, start, goal)
```




```
coord.append(coord[0])
```

- Các thuật tuán BFS, DFS, UCS đã được cài đặt và hoạt động với các kết quả như trên đã trình bày.

- Các thuật toán đã cho sẵn code chạy ra kết quả đúng, lộ trình tuân theo các giao ước như yêu cầu.