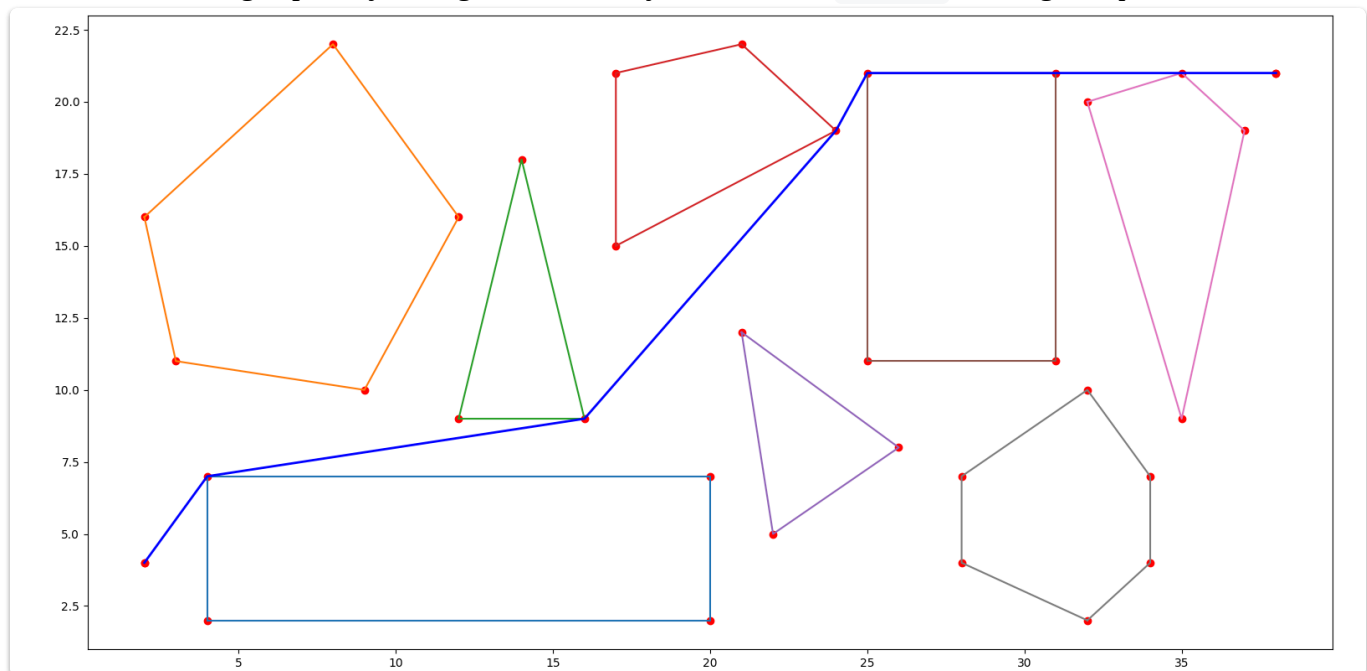


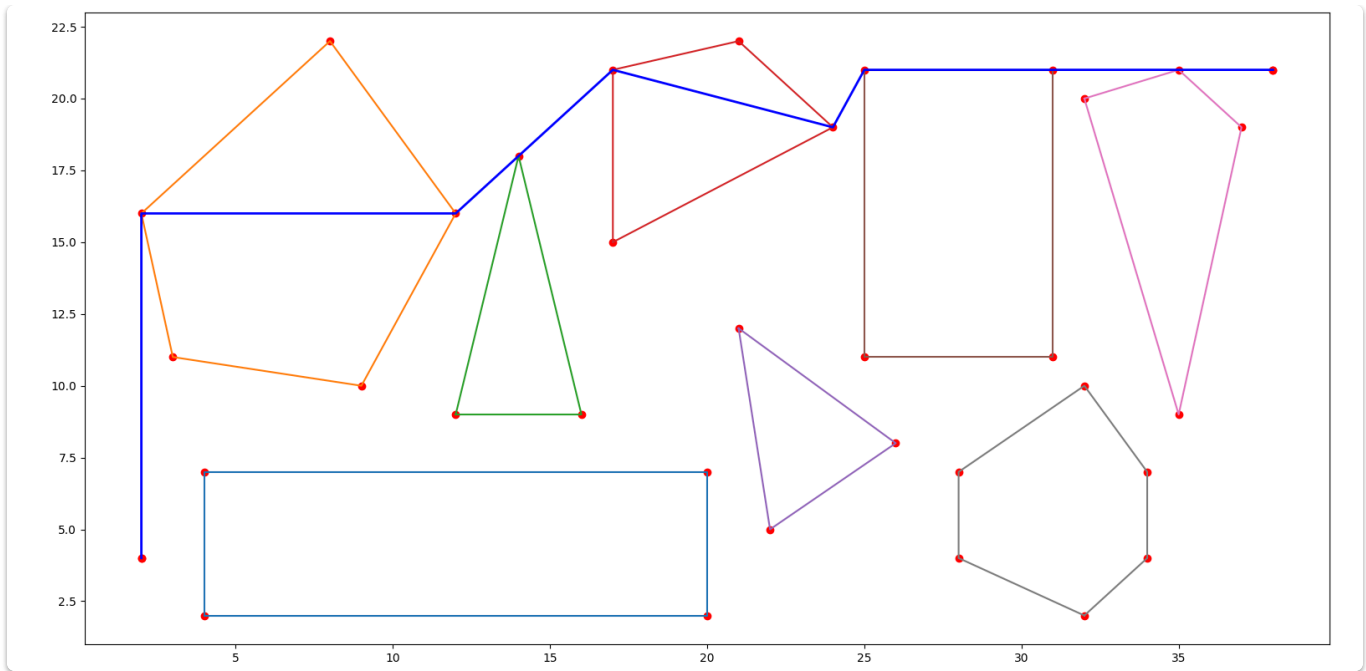
- [1. Cài đặt và thực thi thuật toán](#)
- [2. Áp dụng với BFS, DFS và UCS, cài đặt trên máy tính](#)
  - [Cài đặt thuật toán BFS và thực thi thuật toán BFS](#)
  - [Cài đặt và thực thi thuật toán DFS](#)
  - [Cài đặt và thực thi thuật toán UCS](#)
  - [Đánh giá chung](#)
- [3. Sửa lại code](#)
  - [A-star](#)
  - [GBFS](#)
  - [BFS](#)
  - [DFS](#)
  - [UCS](#)

## 1. Cài đặt và thực thi thuật toán

Thuật toán cô cung cấp, chạy không bị lỗi, và chạy với function `a-star` ra đúng kết quả như sau:



Tuy nhiên, khi chạy với function là `greedy` thì có một vấn đề xảy ra, thuật toán vẫn đi được đến đích nhưng nó lại đi xuyên qua một số polygons, kết quả như sau:



## 2. Áp dụng với BFS, DFS và UCS, cài đặt trên máy tính

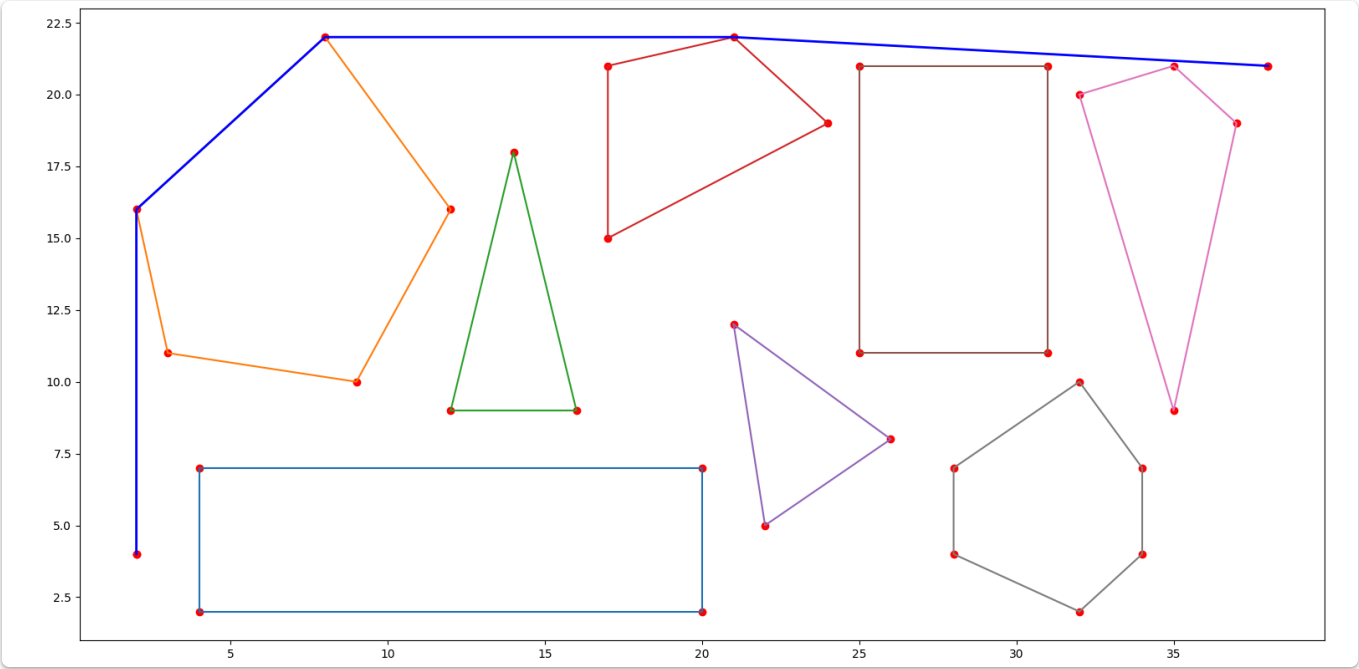
*Phần chạy thuật toán là đang chạy với code ban đầu, chưa fix phần có thể đi xuyên qua polygon.*

### Cài đặt thuật toán BFS và thực thi thuật toán BFS



```
1 def bfs(graph, start, goal):
2     visited = set()
3     queue = Queue()
4
5     queue.put(start)
6     start.pre = None
7
8     while True:
9         if queue.empty():
10             raise Exception('No way Exception')
11         current_node = queue.get()
12
13         if current_node in visited:
14             continue
15
16         visited.add(current_node)
17
18         if current_node == goal:
19             return current_node
20
21         for node in graph.can_see(current_node):
22             if node not in visited and node not in queue.queue:
23                 queue.put(node)
24                 node.pre = current_node
25
26     return None # không tìm thấy đường đi
```

Kết quả chạy thuật toán BFS với code ban đầu:

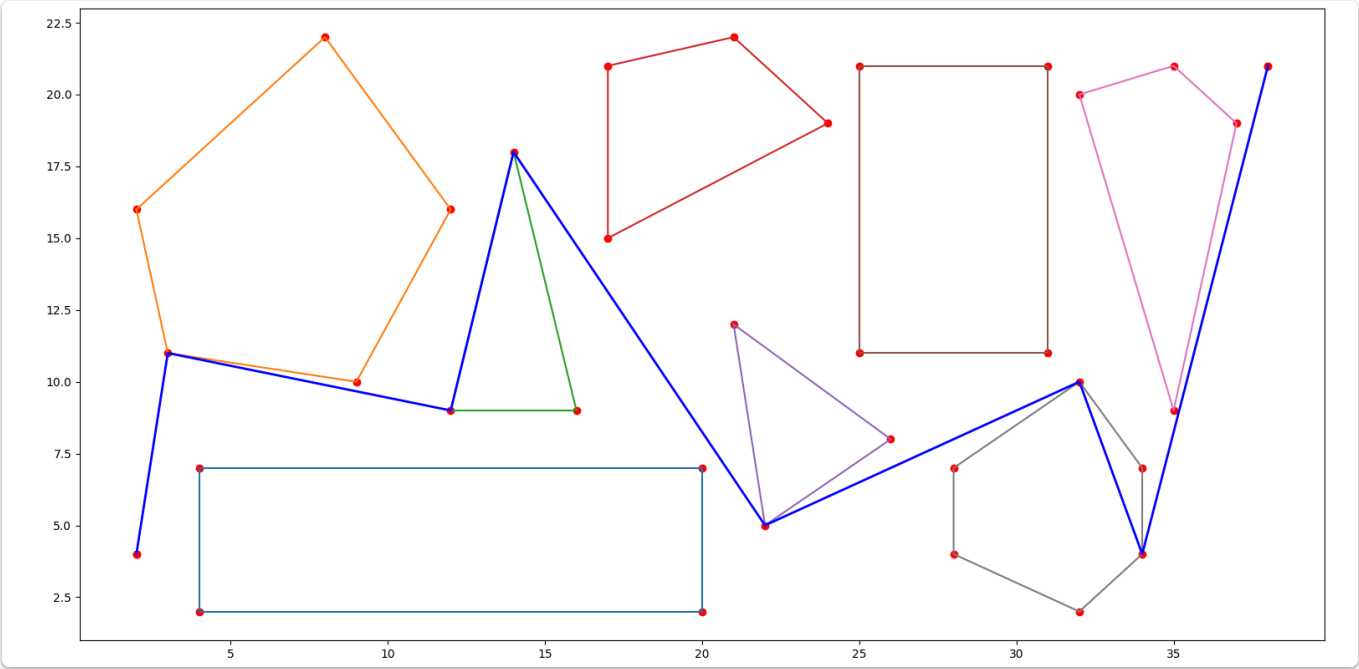


## Cài đặt và thực thi thuật toán DFS



```
1 def dfs(graph, start, goal):
2     visited = set()
3     stack = []
4
5     stack.append(start)
6     start.pre = None
7
8     while True:
9         if stack == []:
10             raise Exception('No way Exception')
11         current_node = stack.pop()
12
13         if current_node in visited:
14             continue
15
16         visited.add(current_node)
17
18         if current_node == goal:
19             return current_node
20
21         for node in graph.can_see(current_node):
22             if node not in visited and node not in stack:
23                 stack.append(node)
24                 node.pre = current_node
25
26     return None # không tìm thấy đường đi
```

Kết quả chạy thuật toán DFS với code ban đầu:

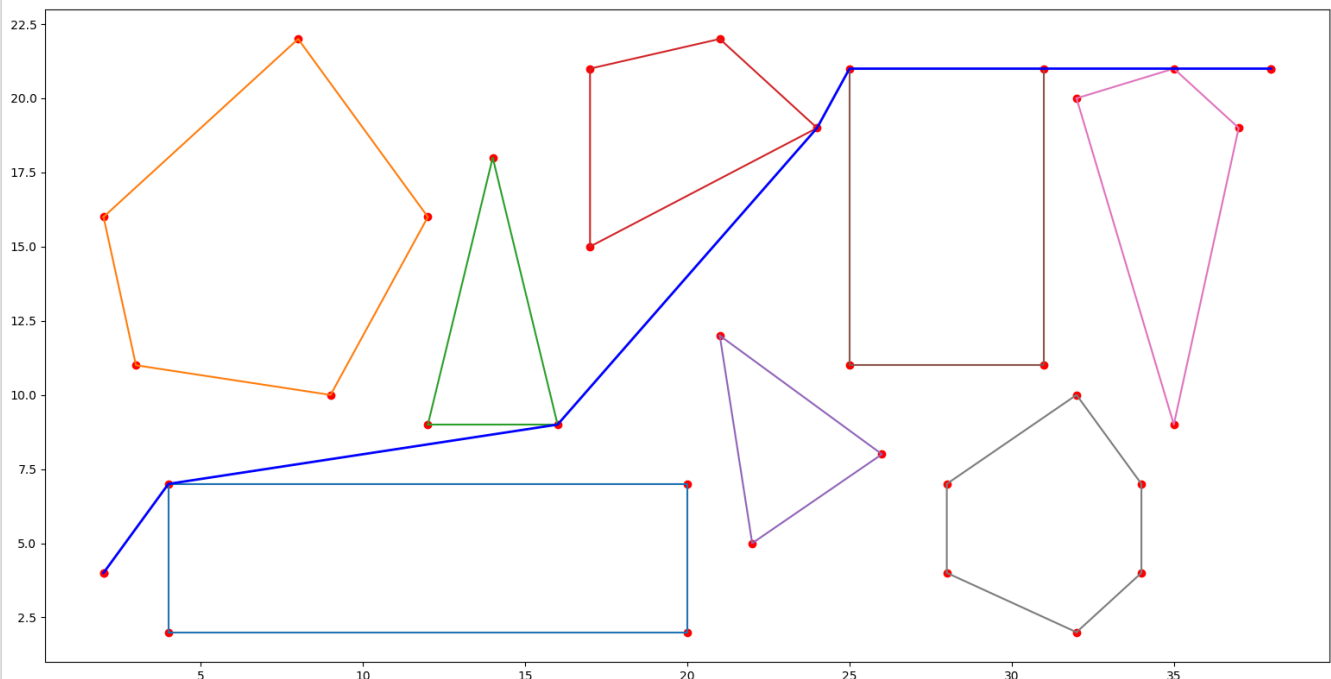


## Cài đặt và thực thi thuật toán UCS



```
1 def ucs(graph, start, goal):
2     visited = set()
3     queue = PriorityQueue()
4
5     queue.put((0, start))
6
7     cost = {start: 0}
8
9     while not queue.empty():
10         current_cost, current_node = queue.get()
11
12         if current_node in visited:
13             continue
14
15         visited.add(current_node)
16
17         if current_node == goal:
18             return current_node
19
20         for node in graph.can_see(current_node):
21             new_cost = cost[current_node] + euclid_distance(current_node, node)
22             if node not in cost or new_cost < cost[node]:
23                 cost[node] = new_cost
24                 queue.put((new_cost, node))
25                 node.pre = current_node
26
27     return None # không tìm thấy đường đi
```

Kết quả chạy thuật toán UCS với code ban đầu:



## Đánh giá chung

Mọi thuật toán đều có thể tìm được đích đến, tuy nhiên trong thuật toán *DFS* và *GBFS* (như chạy ở phần 1) lại có hiện tượng đường đi đi xuyên qua polygon. Như vậy chắc hàm `can_see()` trong class `Graph` có vấn đề gì đó mà đã trả về trong `see_list` lại có các điểm vốn không thể nhìn thấy.

## 3. Sửa lại code

Ta kiểm tra lại code ban đầu của hàm `can_see()` trong class `Graph`

```
1 def can_see(self, start):
2     see_list = list()
3     cant_see_list = list()
4
5     for polygon in self.polygons:
6         for edge in self.polygons[polygon]:
7             for point in self.get_points():
8                 if start == point:
9                     cant_see_list.append(point)
10                if start in self.get_polygon_points(polygon):
11                    for poly_point in self.get_polygon_points(polygon):
12                        if poly_point not in self.get_adjacent_points(start):
13                            cant_see_list.append(poly_point)
14                if point not in cant_see_list:
15                    if start.can_see(point, edge):
16                        if point not in see_list:
17                            see_list.append(point)
18                    elif point in see_list:
19                        see_list.remove(point)
20                        cant_see_list.append(point)
21                else:
22                    cant_see_list.append(point)
23     return see_list
```


Ở phần code này:

```
1 if start in self.get_polygon_points(polygon):
2     for poly_point in self.get_polygon_points(polygon):
3         if poly_point not in self.get_adjacent_points(start):
4             cant_see_list.append(poly_point)
```



Ta thấy rằng có một vấn đề ở đây, việc thêm `poly_point` vào `cant_see_list` đang bị thiếu điều kiện, không thể cho thấy rằng mối liên hệ giao nhau = rỗng của `see_list` và `cant_see_list` nên dẫn đến trường hợp dù điểm không nhìn thấy được vẫn có thể lọt vào `see_list` gây ra hiện tượng đi xuyên polygon.

Cách khắc phục: Thêm phần kiểm tra điều kiện như sau: *Nếu không nằm trong `see_list` thì ta thêm vào `cant_see_list`, ngược lại loại bỏ nó khỏi `see_list`*



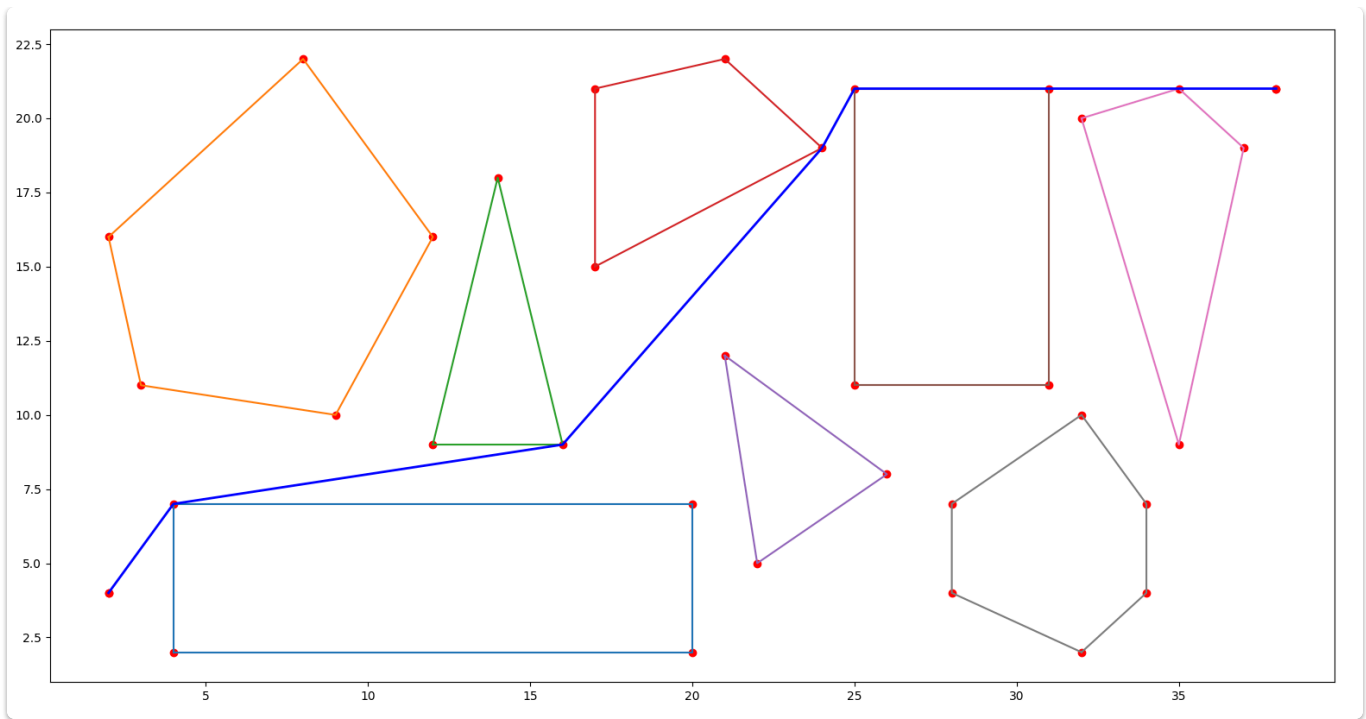
```
1  if poly_point not in see_list:
2      cant_see_list.append(poly_point)
3  else:
4      see_list.remove(poly_point)
```

Vậy hàm `can_see` hoàn chỉnh mới của `Graph` sẽ như sau:

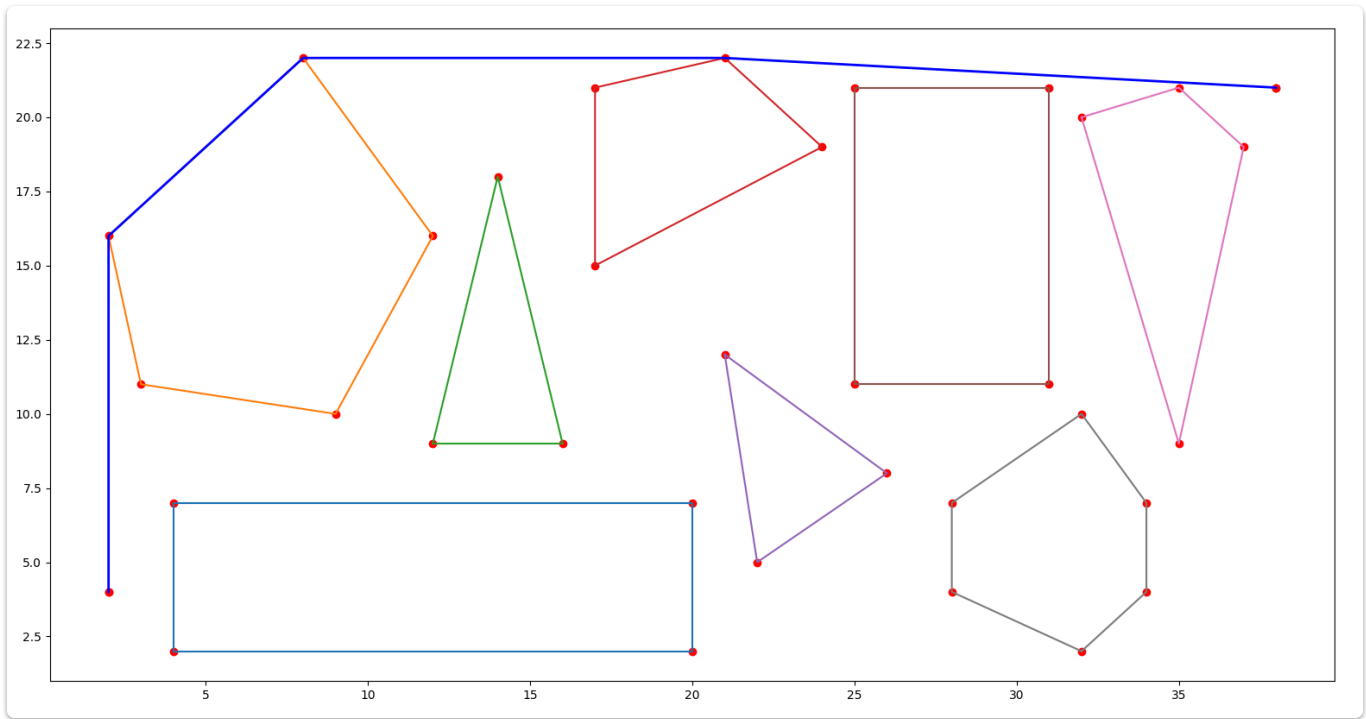
```
1 def can_see(self, start):
2     see_list = list()
3     cant_see_list = list()
4
5     for polygon in self.polygons:
6         for edge in self.polygons[polygon]:
7             for point in self.get_points():
8                 if start == point:
9                     cant_see_list.append(point)
10                if start in self.get_polygon_points(polygon):
11                    for poly_point in self.get_polygon_points(polygon):
12                        if poly_point not in self.get_adjacent_points(start):
13                            if poly_point not in see_list:
14                                cant_see_list.append(poly_point)
15                            else:
16                                see_list.remove(poly_point)
17                if point not in cant_see_list:
18                    if start.can_see(point, edge):
19                        if point not in see_list:
20                            see_list.append(point)
21                    elif point in see_list:
22                        see_list.remove(point)
23                        cant_see_list.append(point)
24                else:
25                    cant_see_list.append(point)
26     return see_list
```

Ta thử chạy lại với tất cả thuật toán

## A-star

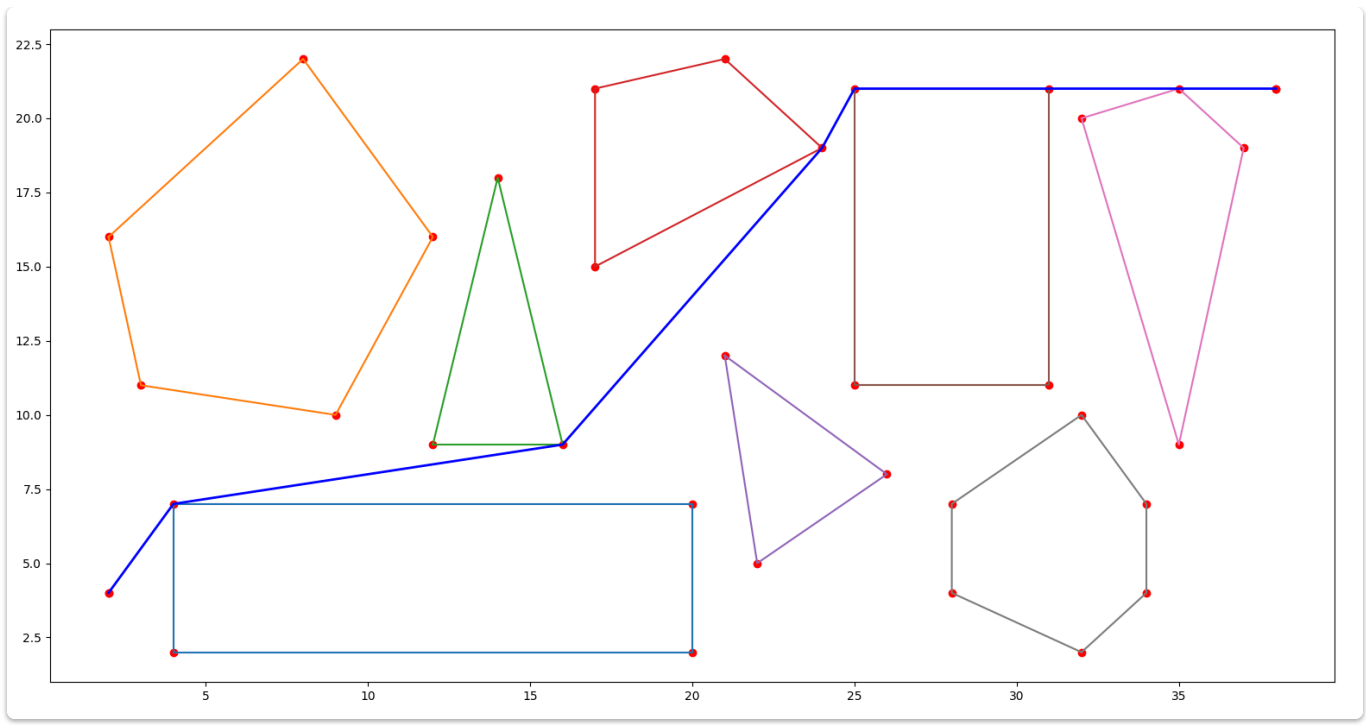


**GBFS**



**BFS**





Như vậy hiện tượng đi xuyên polygon đã không còn xảy ra nữa, các thuật toán đã chạy đúng.