Họ tên: Đinh Xuân Khang

MSSV: 22280042

BÁO CÁO THỰC HÀNH TUẦN 4

Chương trình:

Chương trình đã cho bị báo lỗi cảnh báo:

DeprecationWarning: NotImplemented should not be used in a boolean context return list(filter(None.__ne__, [edge.get_adjacent(point) for edge in self.edges]))

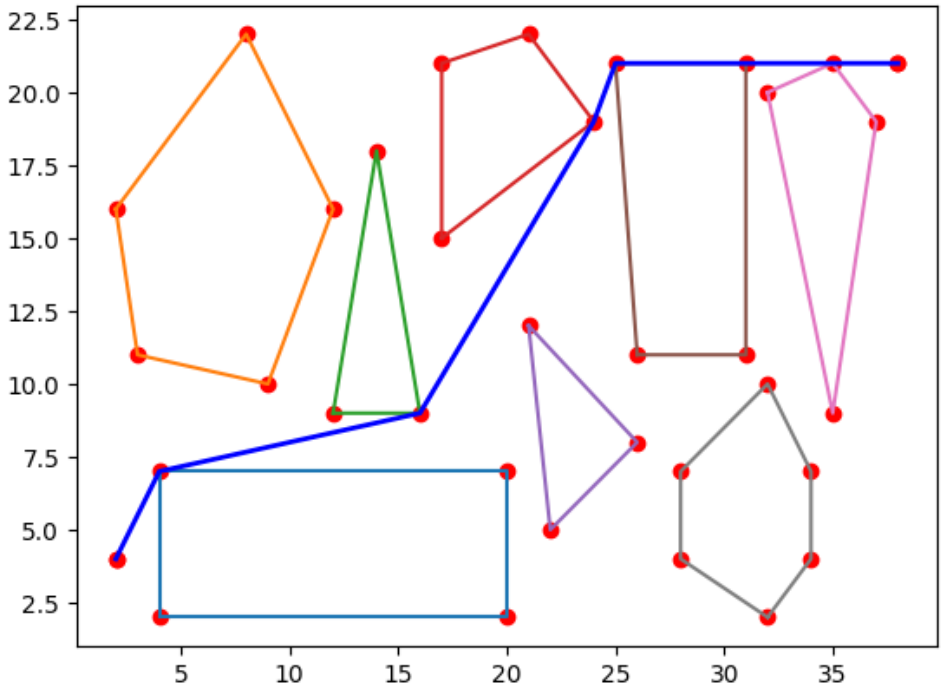
Lỗi ở hàm `def get_adjacent_points(self, point)` của `class Graph`. Ta thay thế thành:

```
def get_adjacent_points(self, point):
  return list(filter(lambda x: x is not None, [edge.get_adjacent(point) for edge in self.edges]))
```

Cài đặt thuật toán đã cho sẵn (A* Search, Greedy Best First Search):

Thuật toán A* Search:

Kết quả:

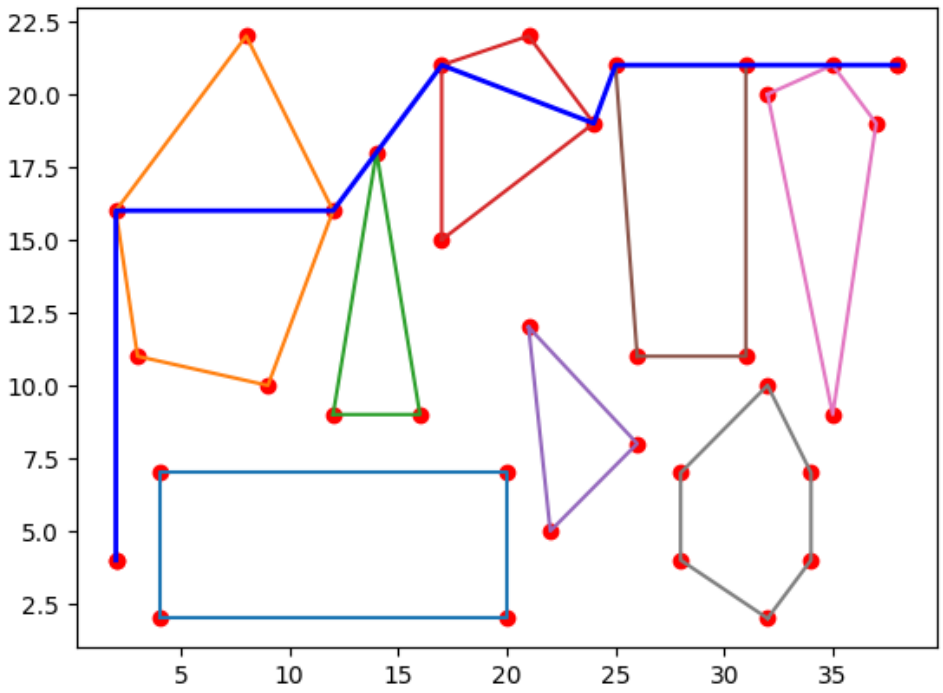


Đường đi của thuật toán A Search.*

Nhận xét: Thuật toán đưa ra đường đi đúng và hợp lệ.

Thuật toán Greedy Best First Search:

Kết quả:



Đường đi của thuật toán GBFS.

Nhận xét: Thuật toán đưa ra đường đi không hợp lệ khi mà đường đi đi xuyên qua nhiều đa giác.

Vấn đề trong source code:

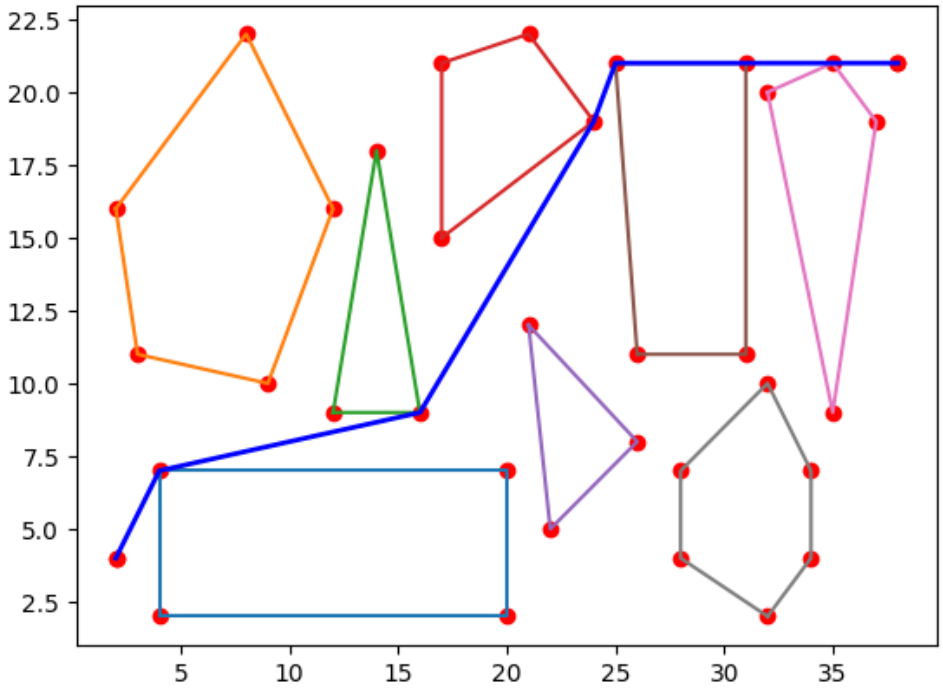
Qua đó, source code có vấn đề cần sửa lại để cho phù hợp là tránh lỗi đường đi đi xuyên qua các đa giác. Để sửa lại vấn đề này ta cần chỉnh sửa lại hàm `def can_see` kiểm tra đường đi có xuyên qua đa giác hay không.

```
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:
                            # kiểm tra xem điểm có nằm trong polygon không
                            if point in see_list:
                                see_list.remove(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 in see_list:
                                        see_list.remove(poly_point)
                                    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
```

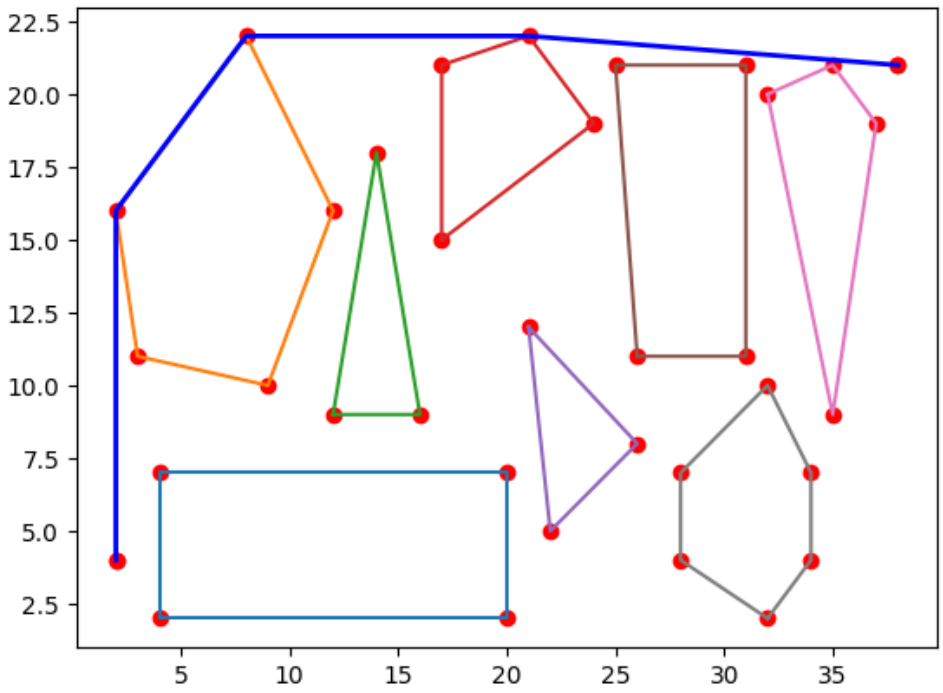
Chạy lại thuật toán A* Search:



Đường đi của thuật toán A Search sau khi sửa source code.*

Nhận xét: Đường đi của thuật toán vẫn giống như ban đầu.

Chạy lại thuật toán Greedy Best First Search:



Đường đi của thuật toán GBFS sau khi sửa source code.

Nhận xét: Đường đi của thuật toán đã trở nên đúng và hợp lệ so với đường đi trước đó.

Cài đặt các thuật toán BFS, DFS, UCS:

Thuật toán BFS:

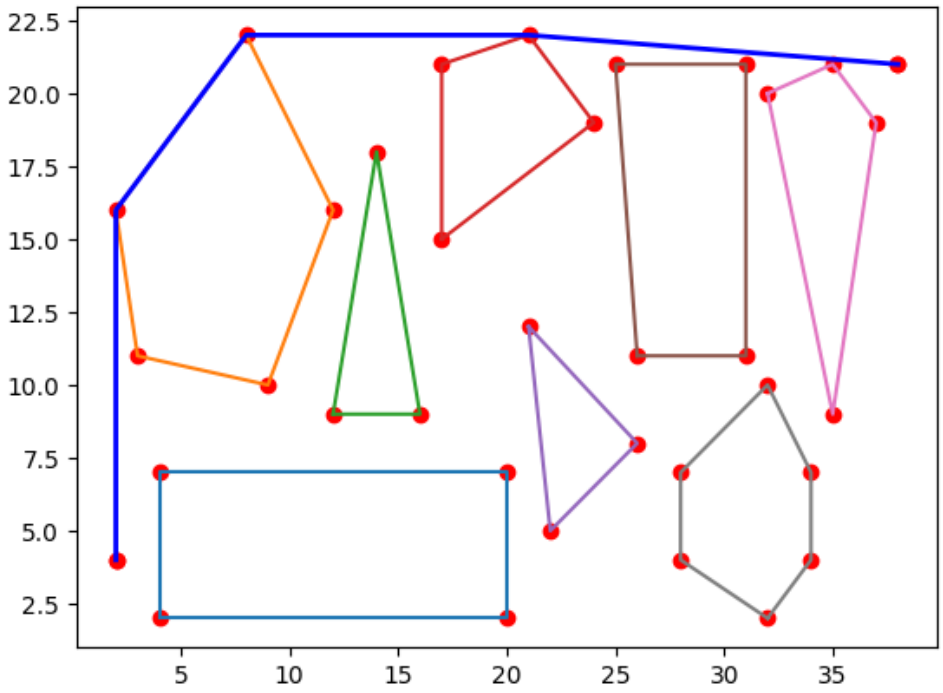
Source code:

```
def BFS(graph, start, goal):
    closed = set()
    queue = Queue()
    queue.put(start)

    if start not in closed:
        closed.add(start)
```

```
    while not queue.empty():
        node = queue.get()
        if node == goal:
            return node
        for i in graph.can_see(node):
            if i not in closed:
                closed.add(i)
                i.pre = node
                queue.put(i)
    return node
```

Kết quả:



Đường đi của thuật toán BFS.

Nhận xét: Thuật toán đưa ra đường đi đúng và hợp lệ và giống với thuật toán GBFS.

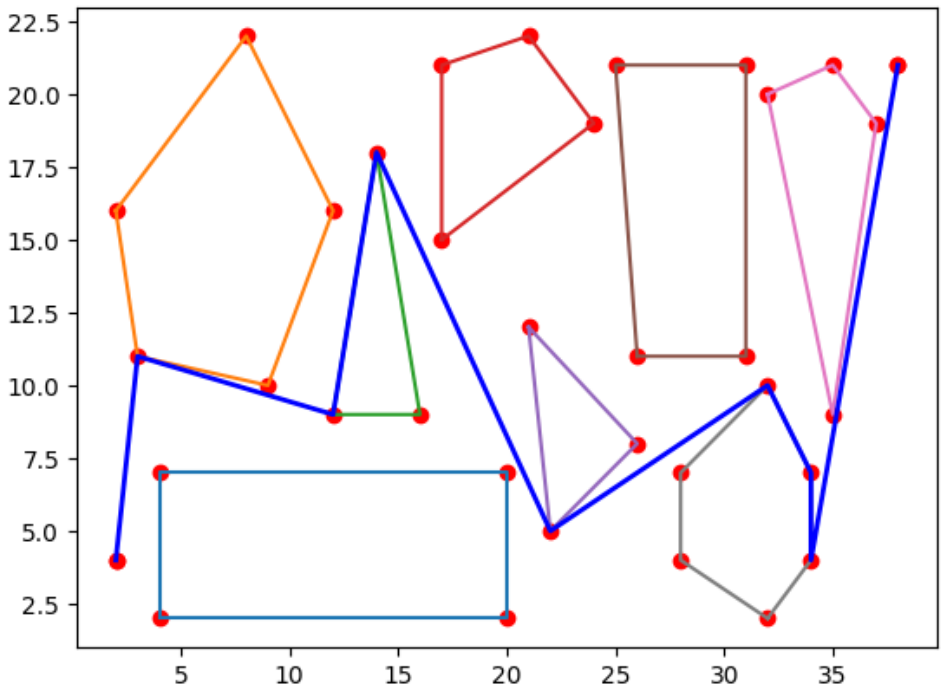
Thuật toán DFS:

Source code:

```
def DFS(graph, start, goal):
    closed = set()
    stack = []
    stack.append(start)

    if start not in closed:
        closed.add(start)
    while not stack == []:
        node = stack.pop()
        if node == goal:
            return node
        for i in graph.can_see(node):
            if i not in closed:
                closed.add(i)
                i.pre = node
                stack.append(i)
    return node
```

Kết quả:



Đường đi của thuật toán DFS.

Nhận xét: Thuật toán đưa ra đường đi đúng và hợp lệ.

Thuật toán UCS:

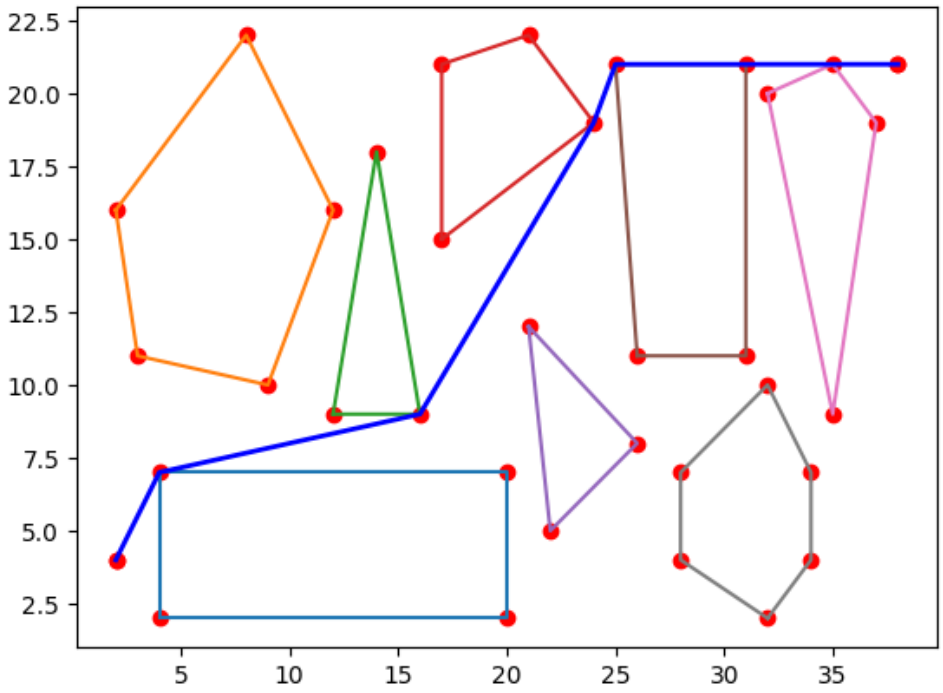
Source code:

```
def UCS(graph, start, goal):
    closed = set()
    queue = PriorityQueue()
    queue.put((0, start))

    if start not in closed:
        closed.add(start)
    while not queue.empty():
        w, node = queue.get()
        if node == goal:
```

```
                return node
            for i in graph.can_see(node):
                new_cost = node.g + euclid_distance(node, i)
                if i not in closed or new_cost < i.g:
                    closed.add(i)
                    i.g = new_cost
                    i.pre = node
                    new_cost = i.g
                    queue.put((new_cost, i))
    return node
```

Kết quả:



Đường đi của thuật toán UCS.

Nhận xét: Thuật toán đưa ra đường đi đúng và hợp lệ và giống với thuật toán A* Search.