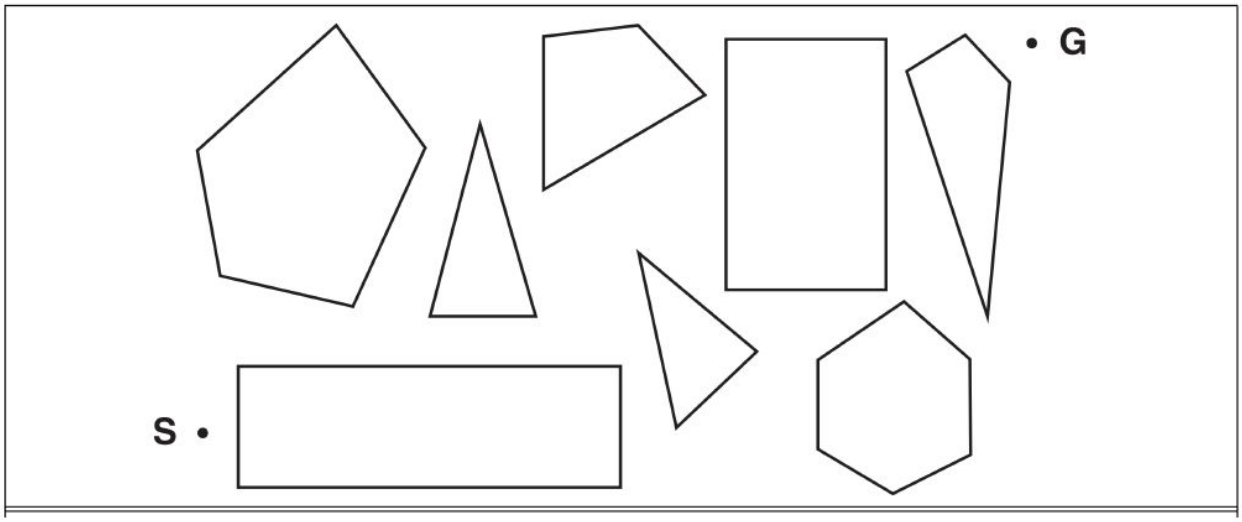
SSN College of Engineering
Department of Computer Science and Engineering
CS1504—Artificial Intelligence
Assignment 3: Path Finding

Name: Aviansh Gupta
Reg No: 185001028
Class & Section: CSE - A

Problem Statement:

Consider an autonomous mobile robot in a crowded environment that needs to find an efficient path from its current location S to a desired location G. As an idealization of the situation, assume that the obstacles (whatever they may be) are abstracted by polygons. The problem now reduces to finding the shortest path between two points in a plane that has convex polygonal obstacles.



State Space Formulation

Since, the state space is continuous so there are infinitely many states possible and as a result infinitely many paths to the goal state.
For this problem, we can consider the start and goal point to be vertices.
We know that,
The shortest distance between two points is a straight line, and if it is not possible to travel in a straight line because some obstacle is in the way, then the next shortest distance is a sequence of line segments, end-to-end, that deviate from the straight line as little as possible. So the first segment of this sequence must go from the start point to a tangent point on an obstacle - any path that gives the polygon a wider girth would be longer. Because the obstacles are polygons, the tangent point must be vertices of the obstacles and hence the entire path must go from vertex to vertex. So now the state space is the set of vertices, of which there are 35 in the above figure.

1. **Initial State:** Start vertex (S)
2. **Actions:** It will return all the possible successors of the current state. Successors of the current state will be all the vertices that can be connected to the current state with a straight line without any intersection with any of the edges of the polygons.
3. **Goal Test:** If the value of heuristic of that state is zero then it's the goal state.
4. **Final State:** Goal vertex (G)
5. **Path Cost:** Sum of euclidean distance between all the points in the path from start state to the goal.
6. **Heuristic Function:** Here straight line distance between the start state and the goal state is used as heuristic.

File Description

1. **pathfinding.py:** It contains the implementation of the Problem abstract class and the Analysis class for the empirical analysis of the search strategies used.

2. **helper.py:** It contains the following:
    - Problem abstract class
    - Node class for defining states.
    - Graph class for creating a graph.
    - InstrumentedProblem class for analysis.
    - Breadth First Search Implementation
    - Depth First Search Implementation
    - Greedy Best First Search
    - A* Search

3. **generator.py:** It contains the following:
    - Point class
    - **Search Space** class with the following functions:
        - generate_state_space: used for generating state space with polygon obstacles.
        - convex_hull: used to generate a polygon from a given set of points.
        - remove_middle: a utility function for convex_hull.

      Polygons are created by generating a random MxN grid and then dividing it into some small grids and then generating some random points in that respective grid. After that just generate the convex hull for that set of points which will give a polygon.
    - **visibility_graph** class with the following functions:
        - poly_edges: returns a list of edges of all the polygons.
        - Get_poly_id: returns id of polygon in which v is a vertex and its index.
        - create_visibility_graph: returns a graph of form {v1:{n1:c1, n2:c2}} where v1 is a vertex n1, n2 are its neighbours and c1, c2 are the cost corresponding to the edge v1n1, v1,n2 resp.
        - dointersect: returns true if two line segments are intersecting.

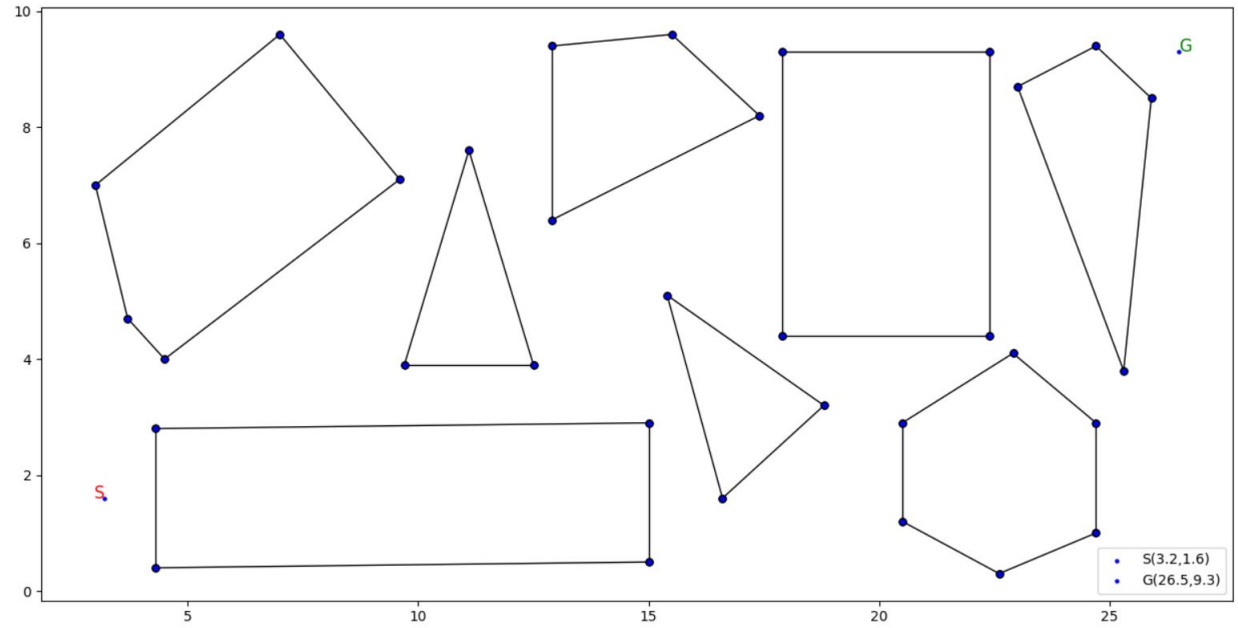
Algorithms Used

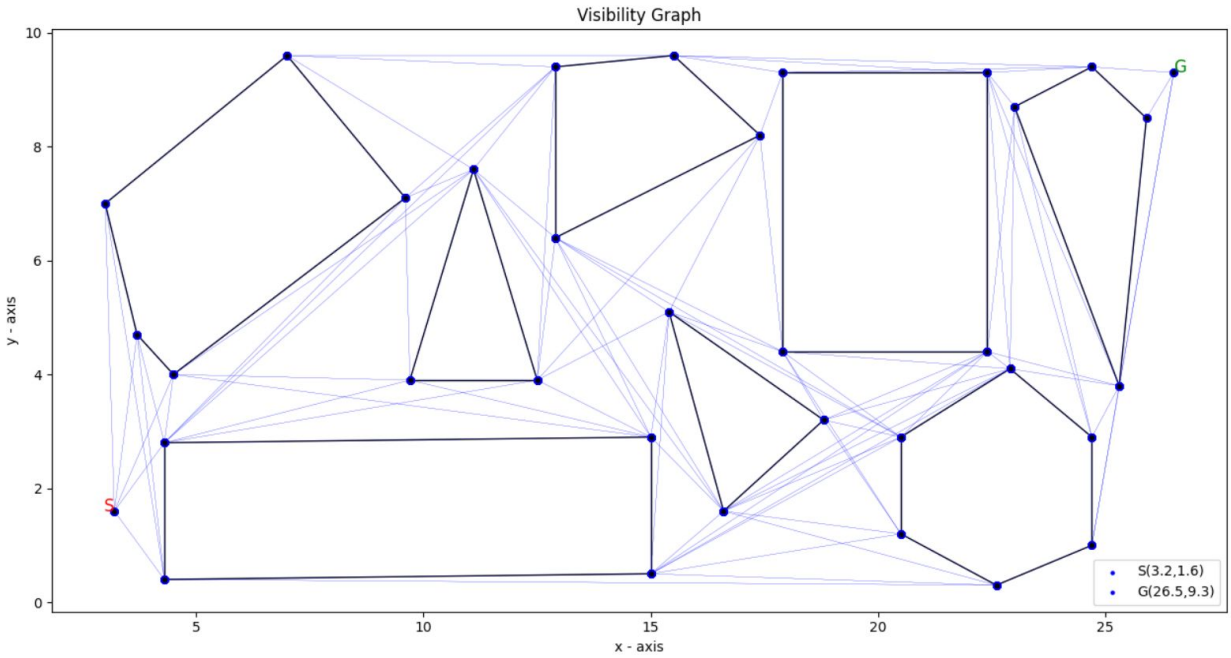
1. Breadth First Search (BFS)
2. Depth First Search (DFS)
3. Greedy Best First Search
4. A* Search

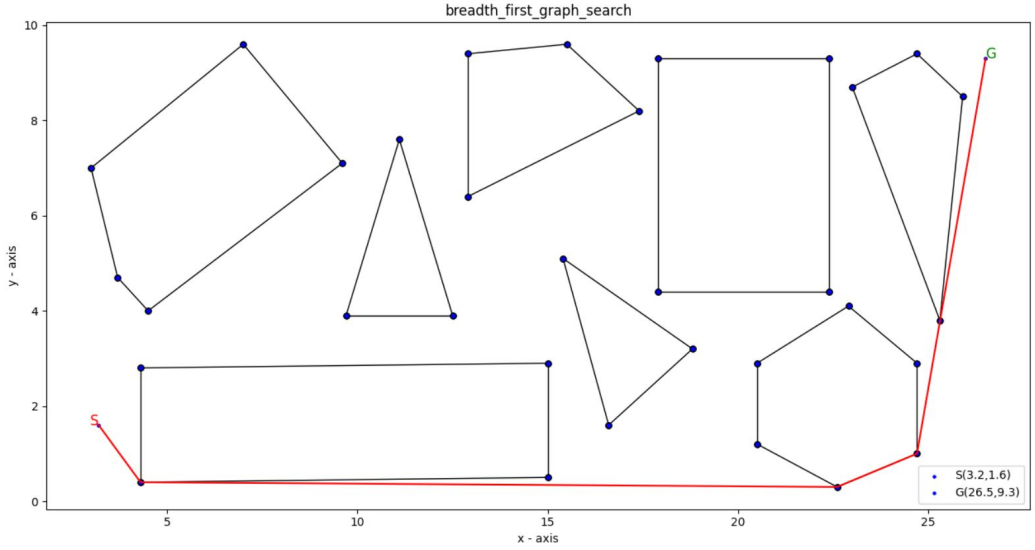
Analysis

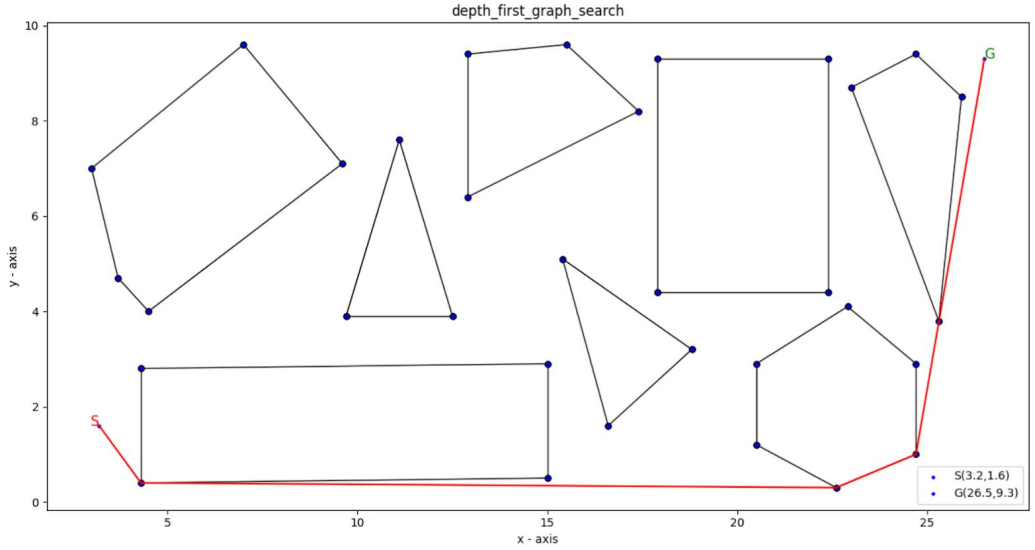
For 5 instances:

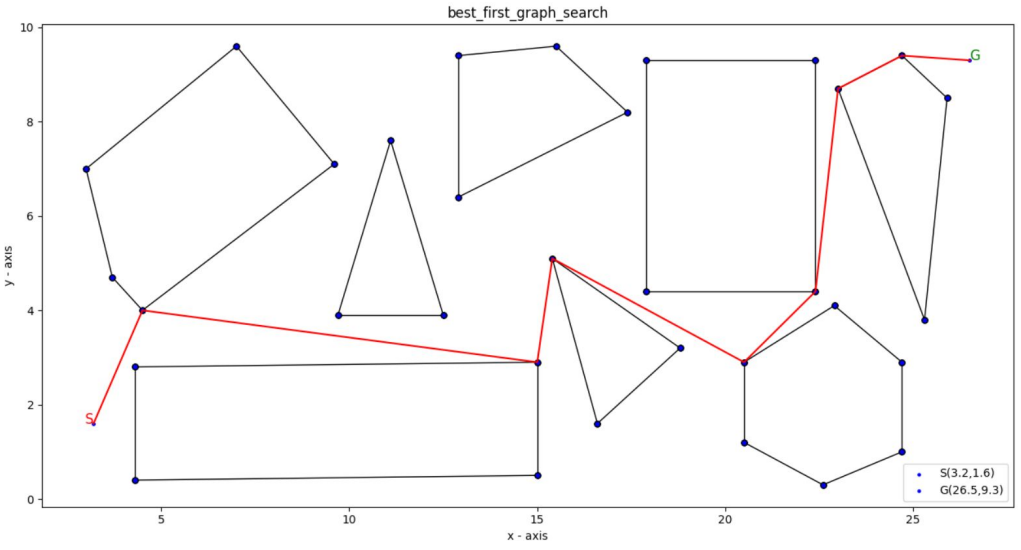
1. **Instance 1**

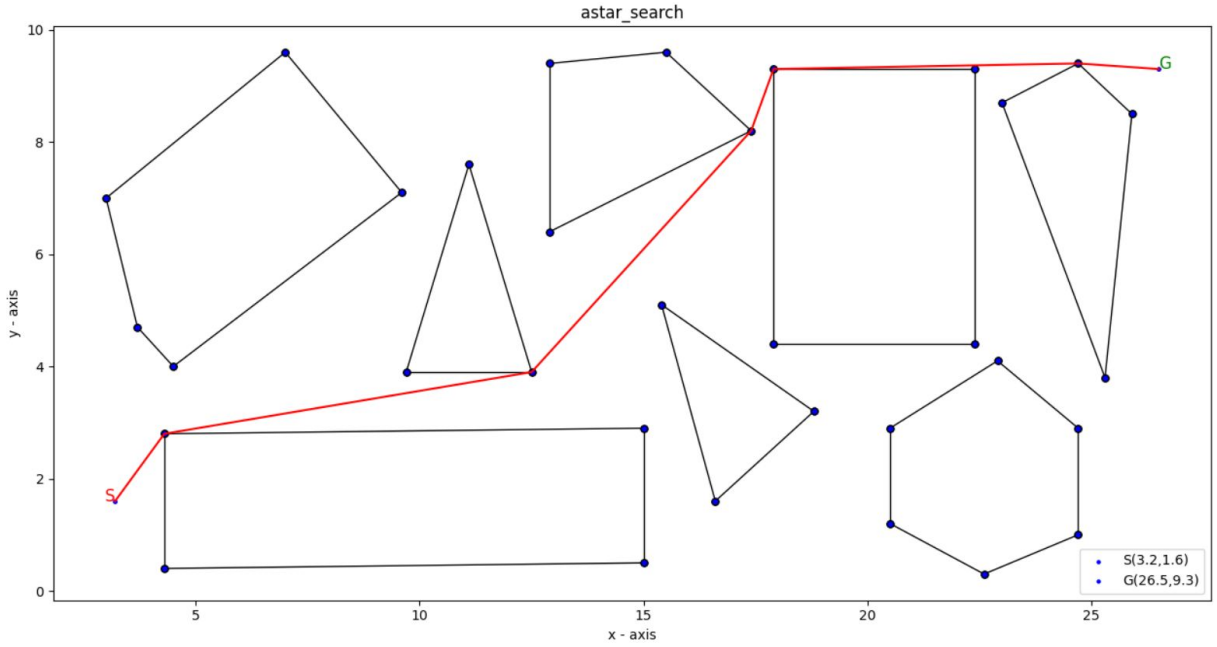





breadth_first_graph_search
y - axis
x - axis
S
G
S(3.2,1.6)
G(26.5,9.3)

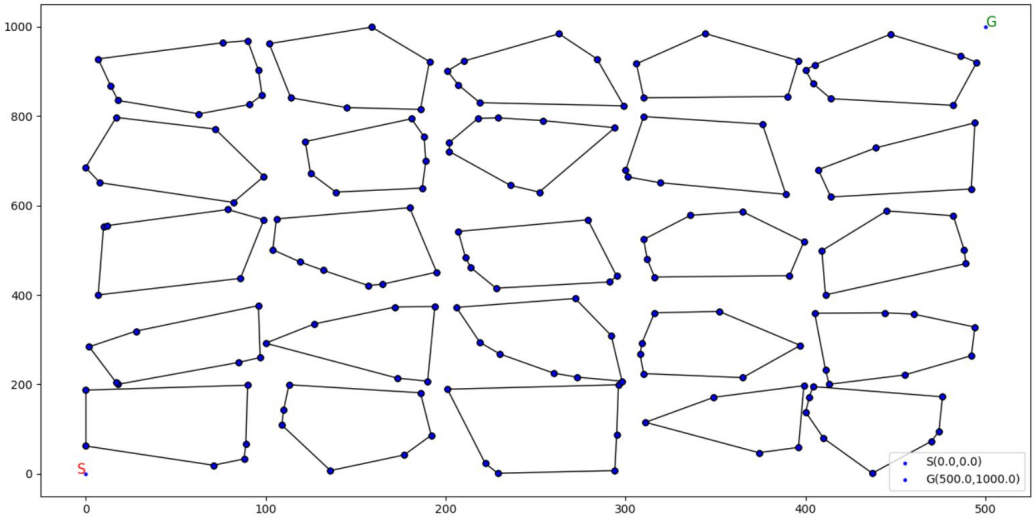

depth_first_graph_search
y - axis
x - axis
S
G
S(3.2,1.6)
G(26.5,9.3)

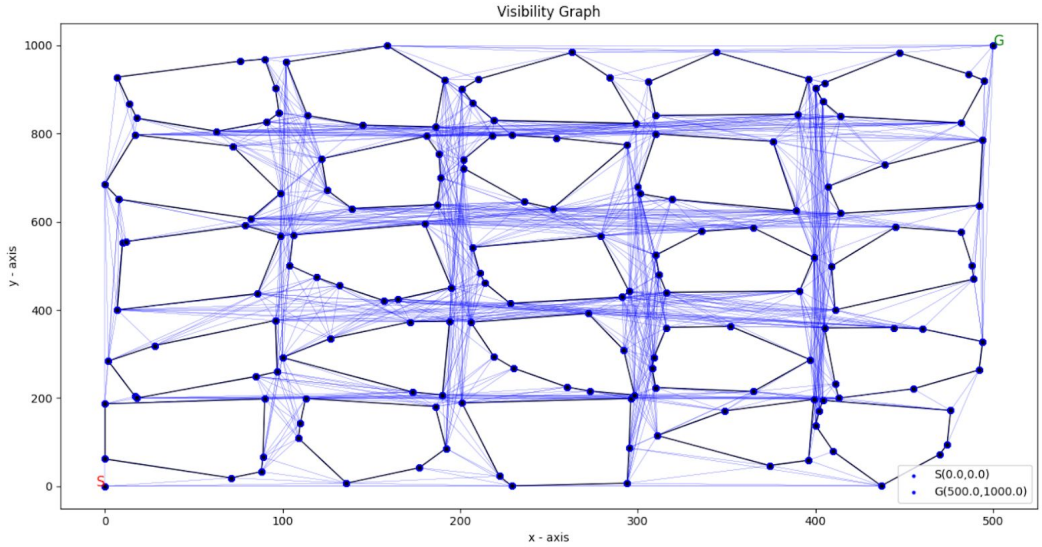

best_first_graph_search
y - axis
x - axis
S
G
S(3.2,1.6)
G(26.5,9.3)

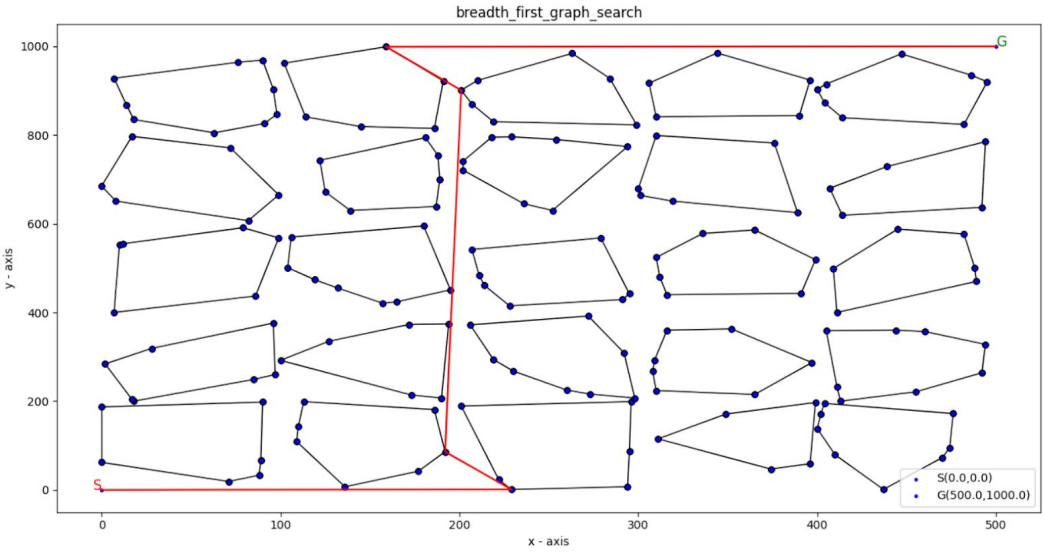


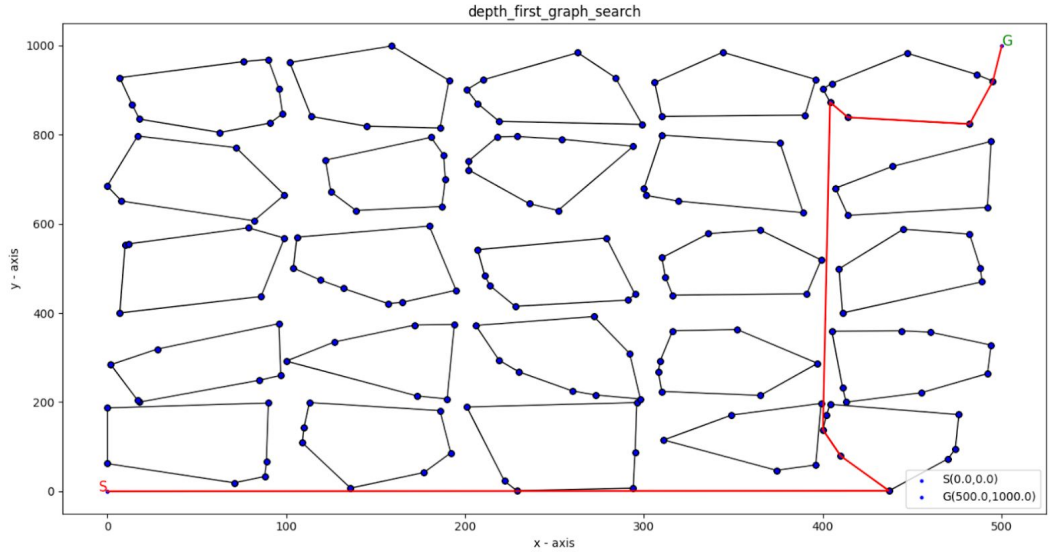

```
Time taken to create Visibility Graph:  0.8091857433319092
----------------------------------------------------------
breadth_first_graph_search
Path:  [<Node (3.2, 1.6)>, <Node (4.3, 0.4)>, <Node (22.6, 0.3)>, <Node (24.7, 1.0)>, <Node (26.5, 9.3)>]
Total Cost: 30.63
----------------------------------------------------------
----------------------------------------------------------
depth_first_graph_search
Path:  [<Node (3.2, 1.6)>, <Node (4.3, 0.4)>, <Node (22.6, 0.3)>, <Node (24.7, 1.0)>, <Node (26.5, 9.3)>]
Total Cost: 30.63
----------------------------------------------------------
----------------------------------------------------------
best_first_graph_search
Path:  [<Node (3.2, 1.6)>, <Node (4.5, 4.0)>, <Node (15.0, 2.9)>, <Node (15.4, 5.1)>, <Node (20.5, 2.9)>, <Node (22.4, 4
.4)>, <Node (23.0, 8.7)>, <Node (24.7, 9.4)>, <Node (26.5, 9.3)>]
Total Cost: 31.48
----------------------------------------------------------
----------------------------------------------------------
astar_search
Path:  [<Node (3.2, 1.6)>, <Node (4.3, 2.8)>, <Node (12.5, 3.9)>, <Node (17.4, 8.2)>, <Node (17.9, 9.3)>, <Node (24.7, 9
.4)>, <Node (26.5, 9.3)>]
Total Cost: 26.23
----------------------------------------------------------
Searcher                    Successors Goal Tests      States  Goal State      Time
breadth_first_graph_search     25         34            181    (26.5, 9.3)      0.0024
depth_first_graph_search        4          5             20    (26.5, 9.3)      0.0011
best_first_graph_search         8          9             60    (26.5, 9.3)      0.0029
astar_search                   22         23            159    (26.5, 9.3)      0.0279
```

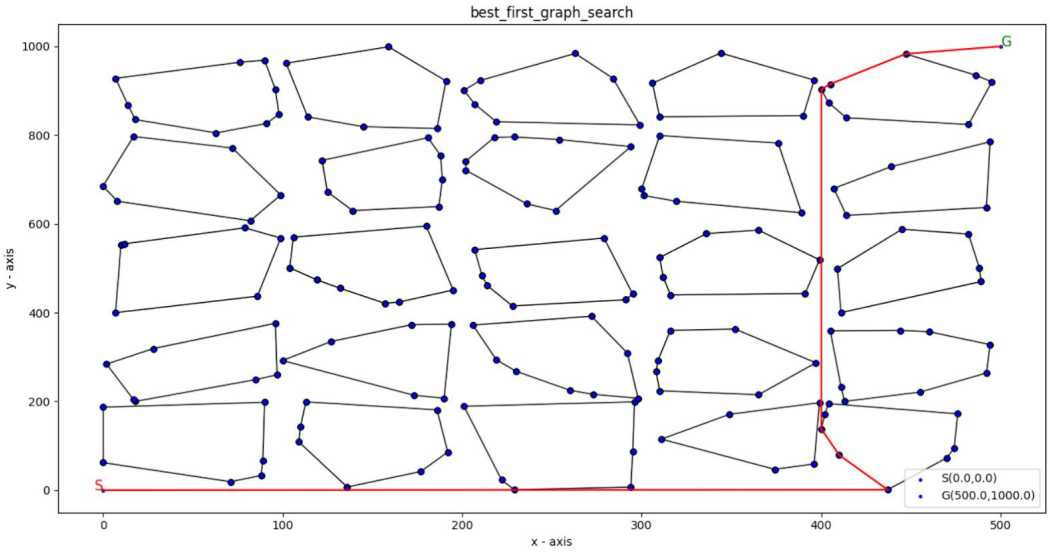
2. Instance 2

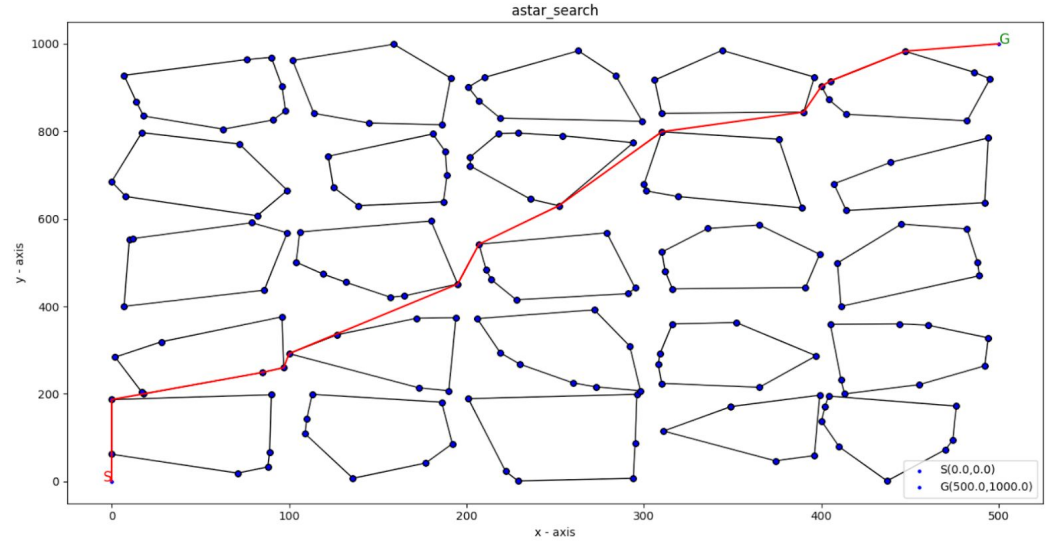

G
S
S(0.0,0.0)
G(500.0,1000.0)


Visibility Graph
G
S
S(0.0,0.0)
G(500.0,1000.0)
y - axis
x - axis


breadth_first_graph_search
G
S
S(0.0,0.0)
G(500.0,1000.0)
y - axis
x - axis

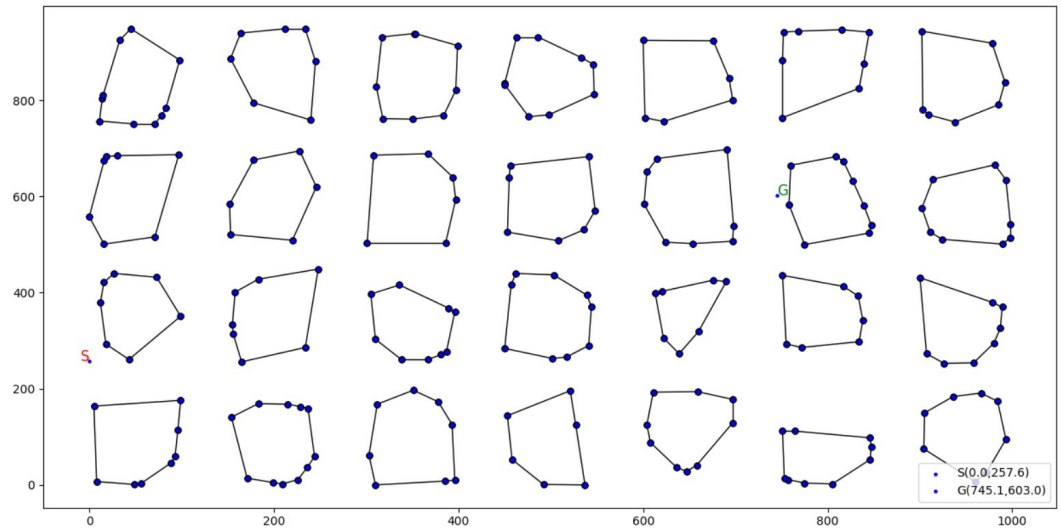

depth_first_graph_search
G
S
S(0.0,0.0)
G(500.0,1000.0)
y - axis
x - axis
0
200
400
600
800
1000
0
100
200
300
400
500

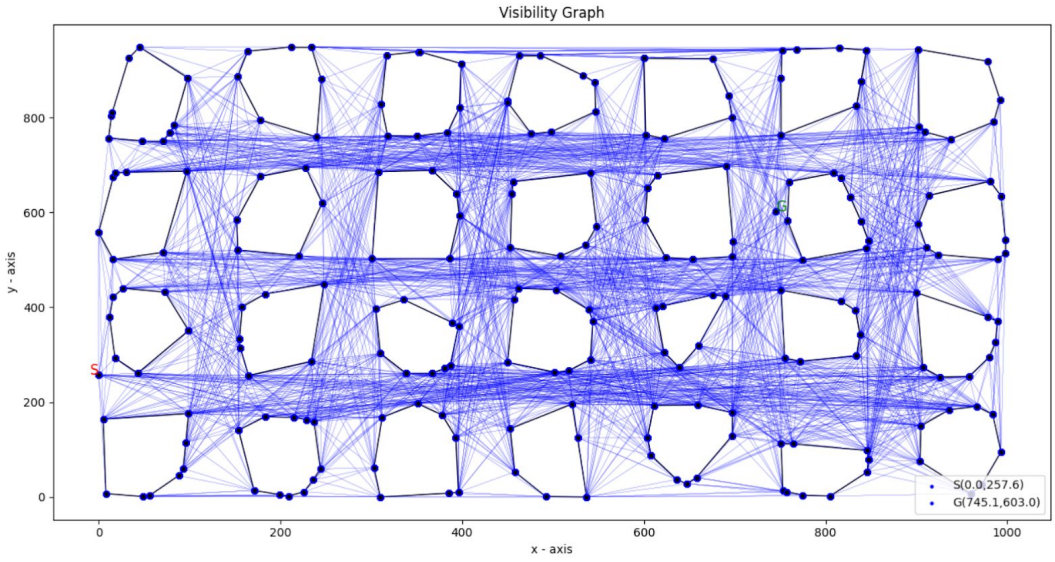

best_first_graph_search
G
S
S(0.0,0.0)
G(500.0,1000.0)
y - axis
x - axis
0
200
400
600
800
1000
0
100
200
300
400
500

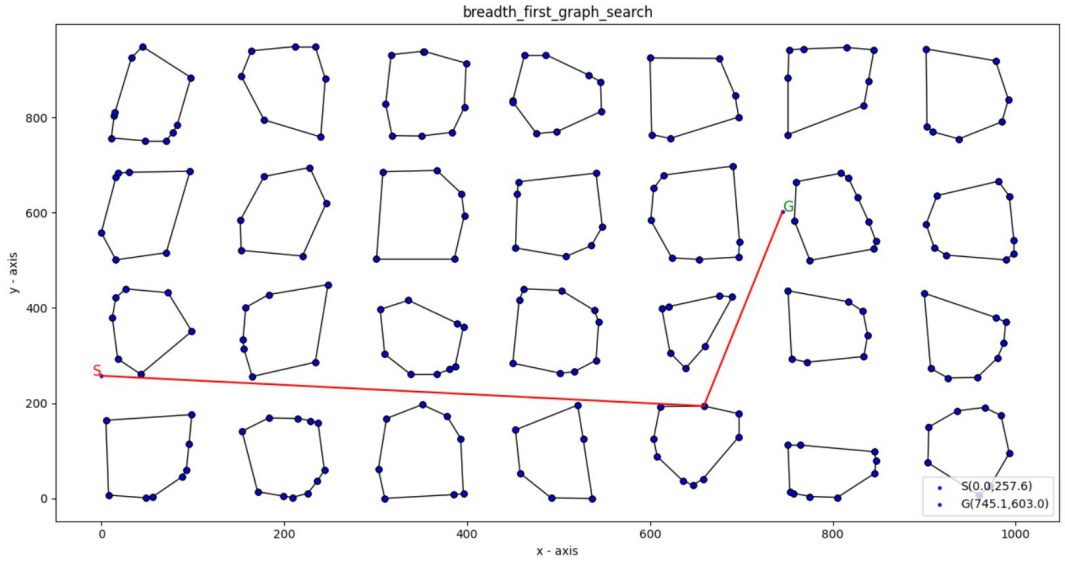

astar_search
G
S
S(0.0,0.0)
G(500.0,1000.0)
y - axis
x - axis
0
200
400
600
800
1000
0
100
200
300
400
500

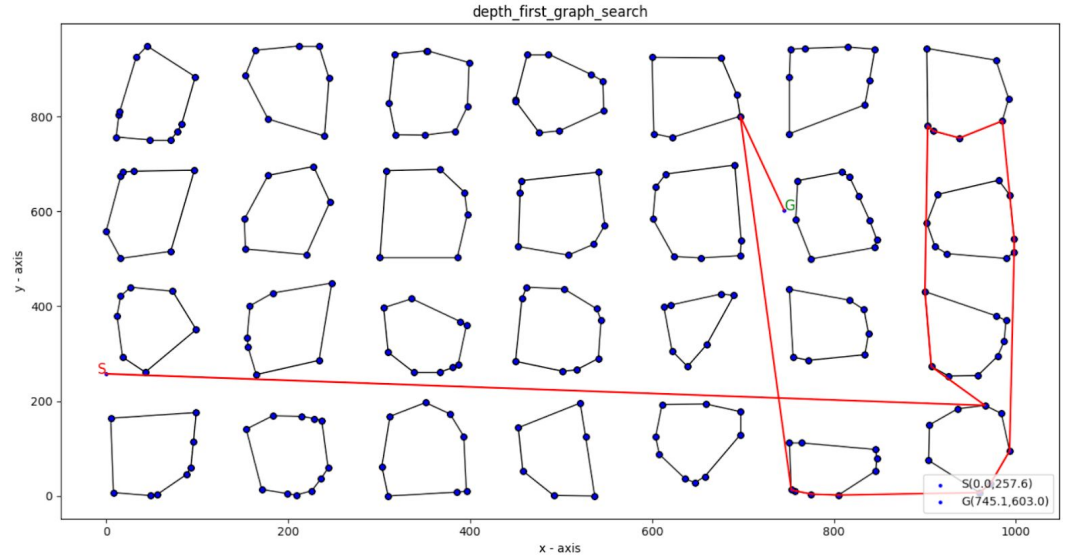
```
Time taken to create Visibility Graph:  35.55574917793274
----------------------------------------------------------
breadth_first_graph_search
Path:  [<Node (0.0, 0.0)>, <Node (229, 1)>, <Node (192, 86)>, <Node (201, 901)>, <Node (159, 999)>, <Node (500.0, 1000.0)>]
Total Cost: 1584.38
----------------------------------------------------------
----------------------------------------------------------
depth_first_graph_search
Path:  [<Node (0.0, 0.0)>, <Node (437, 1)>, <Node (410, 79)>, <Node (400, 138)>, <Node (404, 873)>, <Node (414, 839)>, <Node (482, 824)>, <Node (495, 920)>,
 <Node (500.0, 1000.0)>]
Total Cost: 1596.50
----------------------------------------------------------
----------------------------------------------------------
best_first_graph_search
Path:  [<Node (0.0, 0.0)>, <Node (437, 1)>, <Node (410, 79)>, <Node (400, 138)>, <Node (400, 903)>, <Node (405, 915)>, <Node (447, 983)>, <Node (500.0, 1000
.0)>]
Total Cost: 1492.97
----------------------------------------------------------
----------------------------------------------------------
astar_search
Path:  [<Node (0.0, 0.0)>, <Node (0, 62)>, <Node (0, 187)>, <Node (18, 200)>, <Node (85, 249)>, <Node (97, 260)>, <Node (100, 292)>, <Node (195, 451)>, <Nod
e (207, 542)>, <Node (252, 630)>, <Node (310, 799)>, <Node (390, 844)>, <Node (400, 903)>, <Node (405, 915)>, <Node (447, 983)>, <Node (500.0, 1000.0)>]
Total Cost: 1195.36
----------------------------------------------------------
Searcher                    Successors Goal Tests      States  Goal State      Time
breadth_first_graph_search    109        164           1448    (500.0, 1000.0)  0.0458
depth_first_graph_search        8          9             78    (500.0, 1000.0)  0.0010
best_first_graph_search         7          8             49    (500.0, 1000.0)  0.0043
astar_search                   69         70            973    (500.0, 1000.0)  0.1157
```

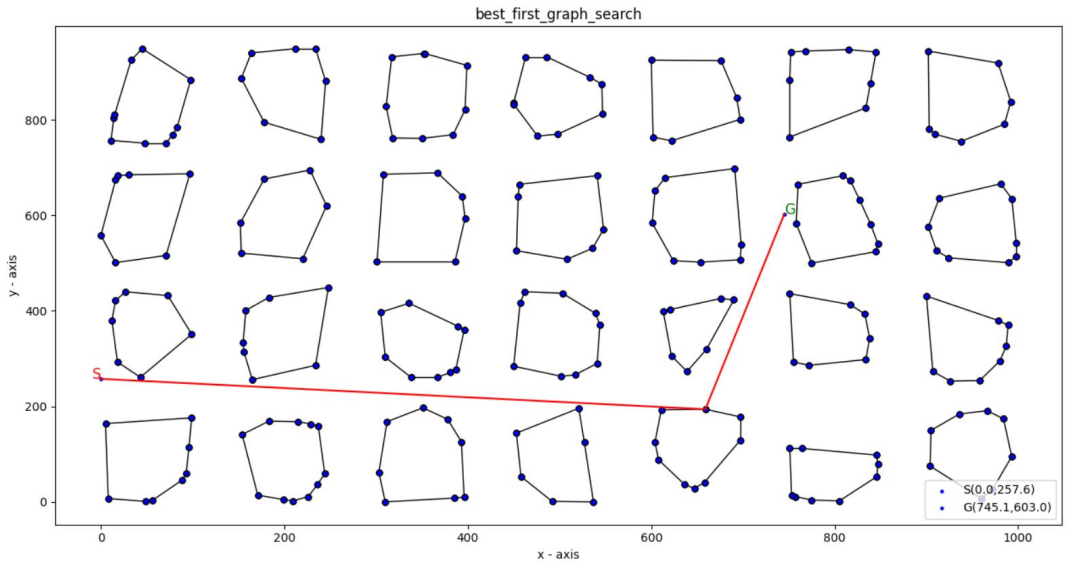
3. Instance 3

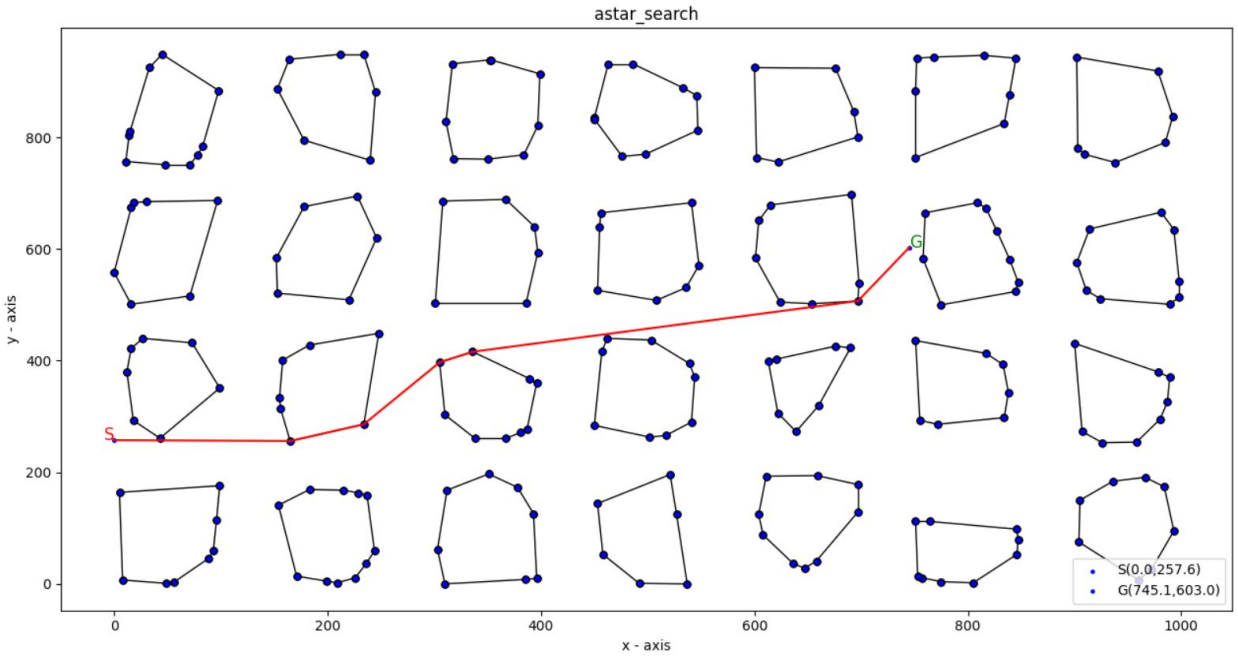





breadth_first_graph_search
y - axis
x - axis
S
G
S(0.0,257.6)
G(745.1,603.0)

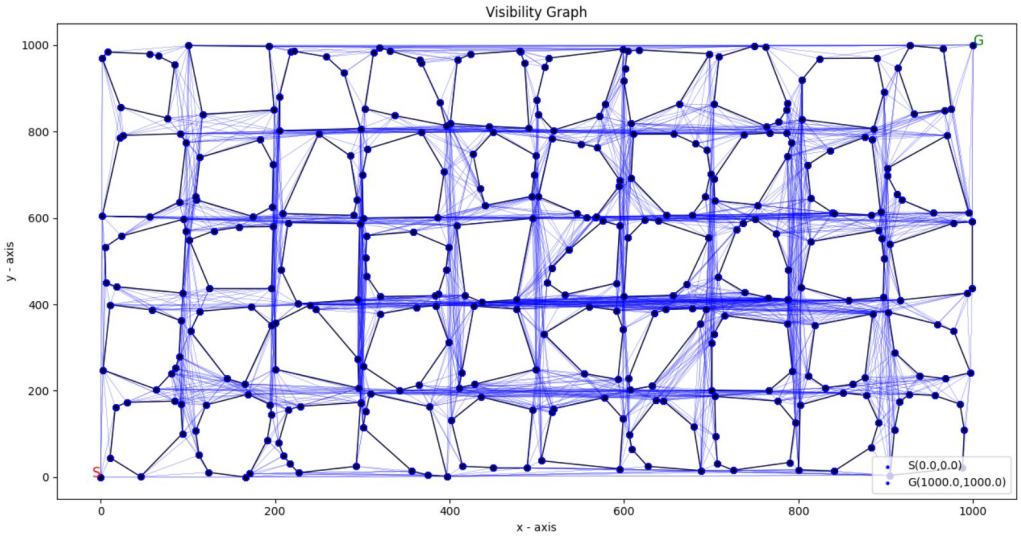

depth_first_graph_search
y - axis
x - axis
S
G
S(0.0,257.6)
G(745.1,603.0)

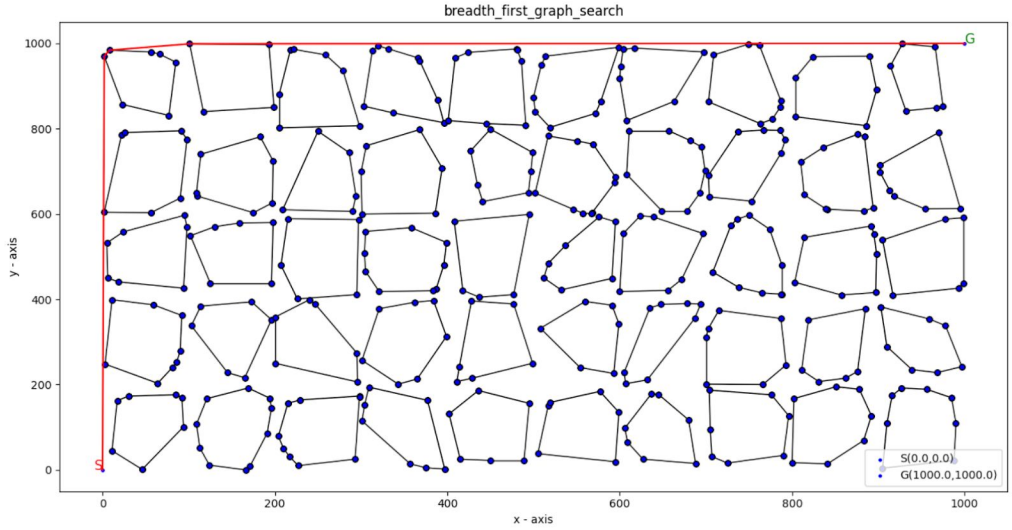

best_first_graph_search
y - axis
x - axis
S
G
S(0.0,257.6)
G(745.1,603.0)

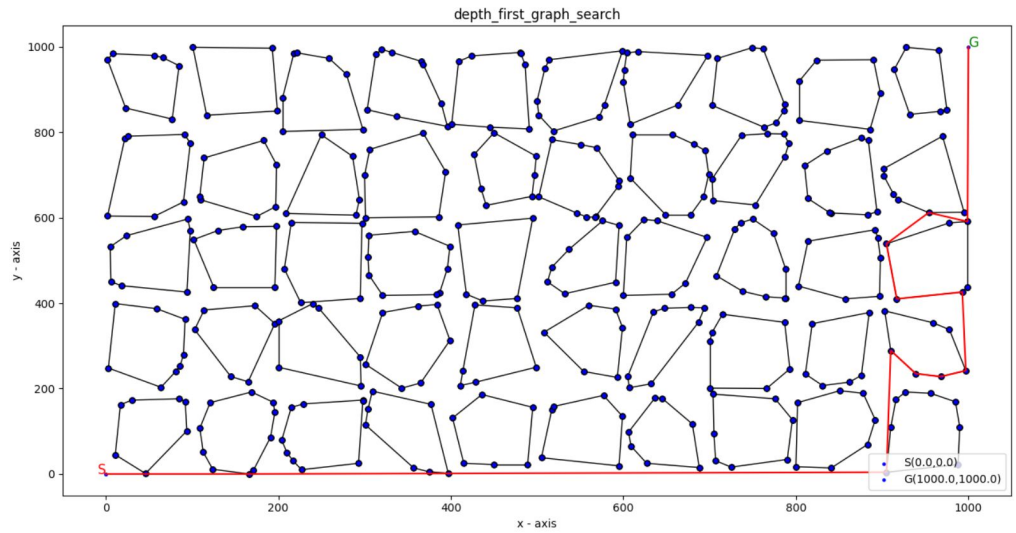


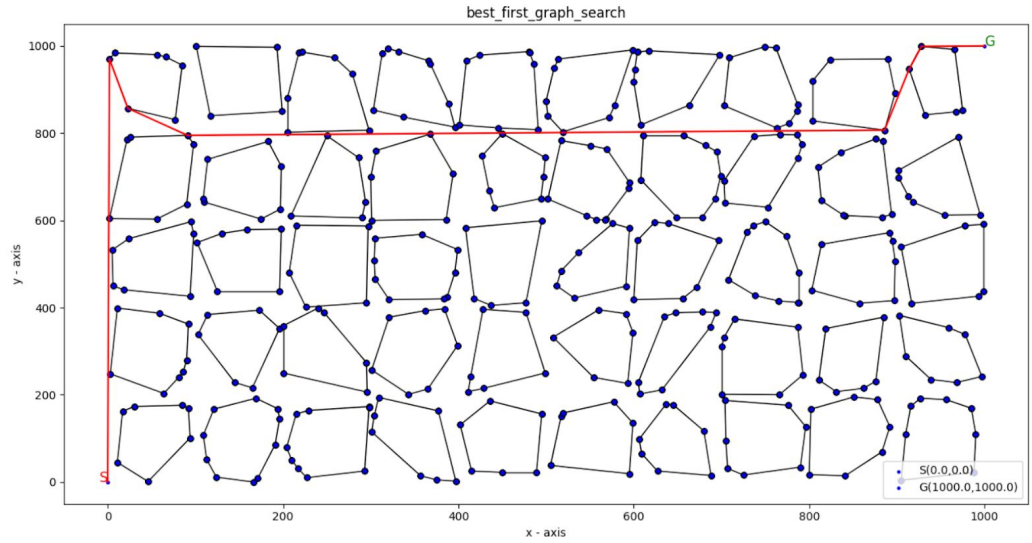

```
Time taken to create Visibility Graph:  82.92478370666504
----------------------------------------------------------
breadth_first_graph_search
Path:  [<Node (0.0, 257.6)>, <Node (659, 194)>, <Node (745.1, 603.0)>]
Total Cost: 1080.03
----------------------------------------------------------
----------------------------------------------------------
depth_first_graph_search
Path:  [<Node (0.0, 257.6)>, <Node (966, 191)>, <Node (907, 273)>, <Node (900, 431)>, <Node (903, 781)>, <Node (909, 770
)>, <Node (938, 755)>, <Node (985, 791)>, <Node (998, 542)>, <Node (998, 514)>, <Node (993, 95)>, <Node (972, 26)>, <Nod
e (960, 7)>, <Node (805, 2)>, <Node (775, 4)>, <Node (757, 11)>, <Node (753, 13)>, <Node (697, 801)>, <Node (745.1, 603.
0)>]
Total Cost: 3675.51
----------------------------------------------------------
----------------------------------------------------------
best_first_graph_search
Path:  [<Node (0.0, 257.6)>, <Node (659, 194)>, <Node (745.1, 603.0)>]
Total Cost: 1080.03
----------------------------------------------------------
----------------------------------------------------------
astar_search
Path:  [<Node (0.0, 257.6)>, <Node (165, 256)>, <Node (234, 286)>, <Node (305, 397)>, <Node (336, 416)>, <Node (697, 507
)>, <Node (745.1, 603.0)>]
Total Cost: 888.04
----------------------------------------------------------
Searcher                    Successors Goal Tests      States  Goal State      Time
breadth_first_graph_search     20         114             433    (745.1, 603.0)   0.0137
depth_first_graph_search       21          22             315    (745.1, 603.0)   0.0561
best_first_graph_search         2           3              58    (745.1, 603.0)   0.0310
astar_search                   33          34             795    (745.1, 603.0)   0.2269
```

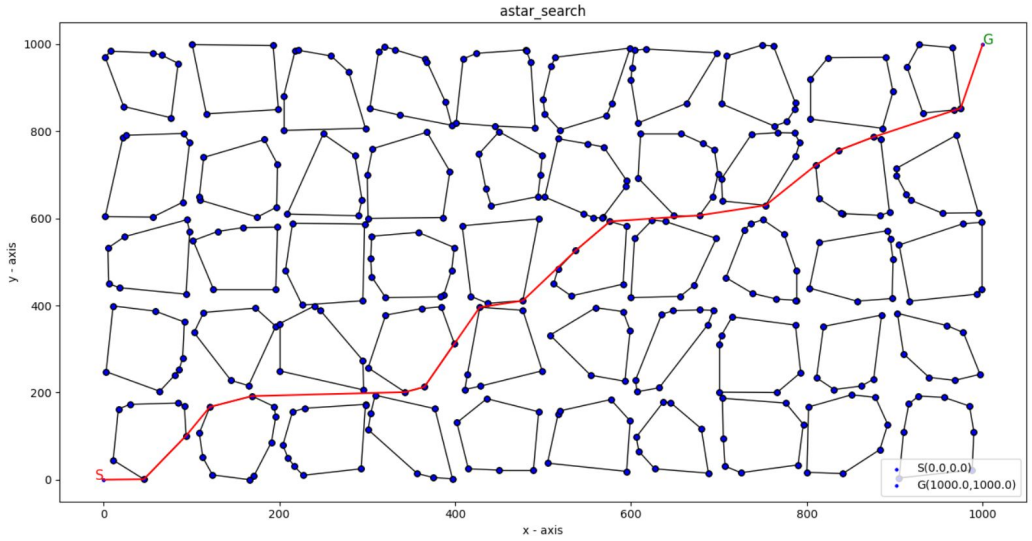
4. Instance 4


Visibility Graph
S
G
S(0.0,0.0)
G(1000.0,1000.0)
x - axis
y - axis


breadth_first_graph_search
S
G
S(0.0,0.0)
G(1000.0,1000.0)
x - axis
y - axis

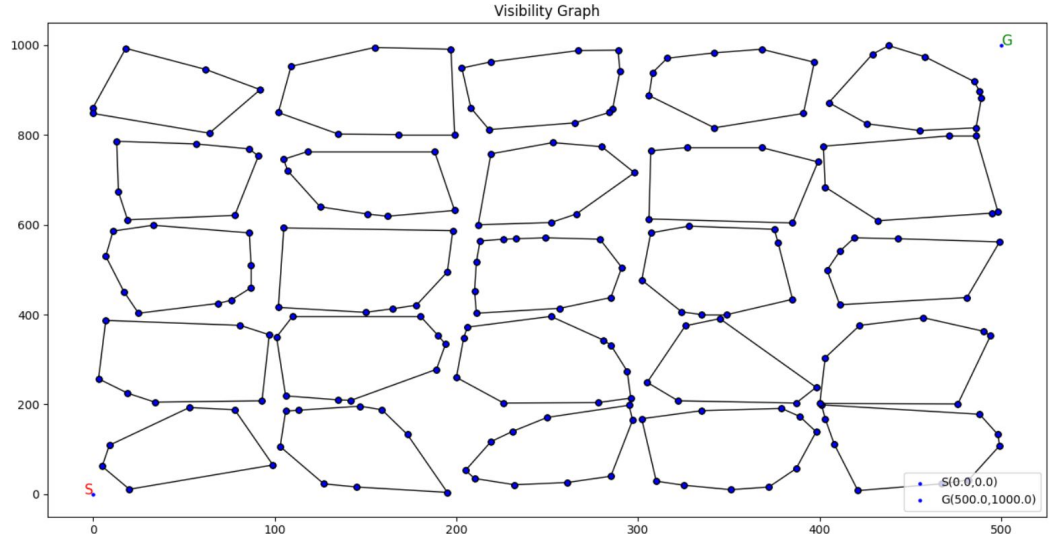

depth_first_graph_search
S
G
S(0.0,0.0)
G(1000.0,1000.0)
x - axis
y - axis

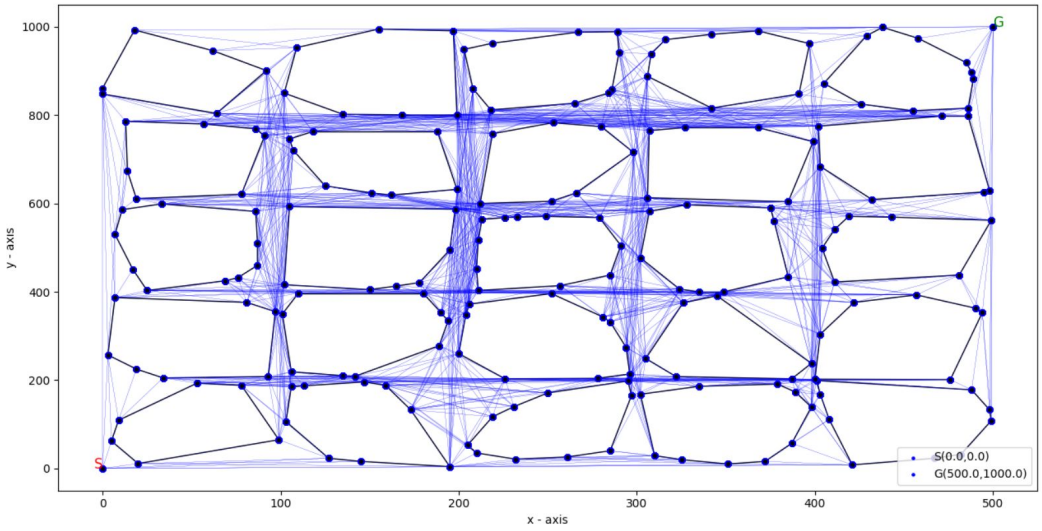


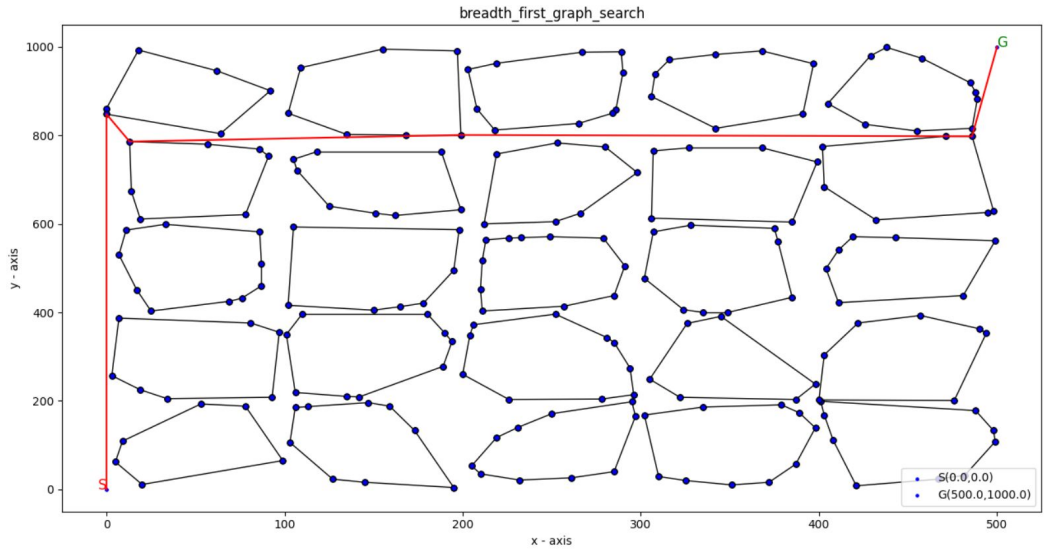


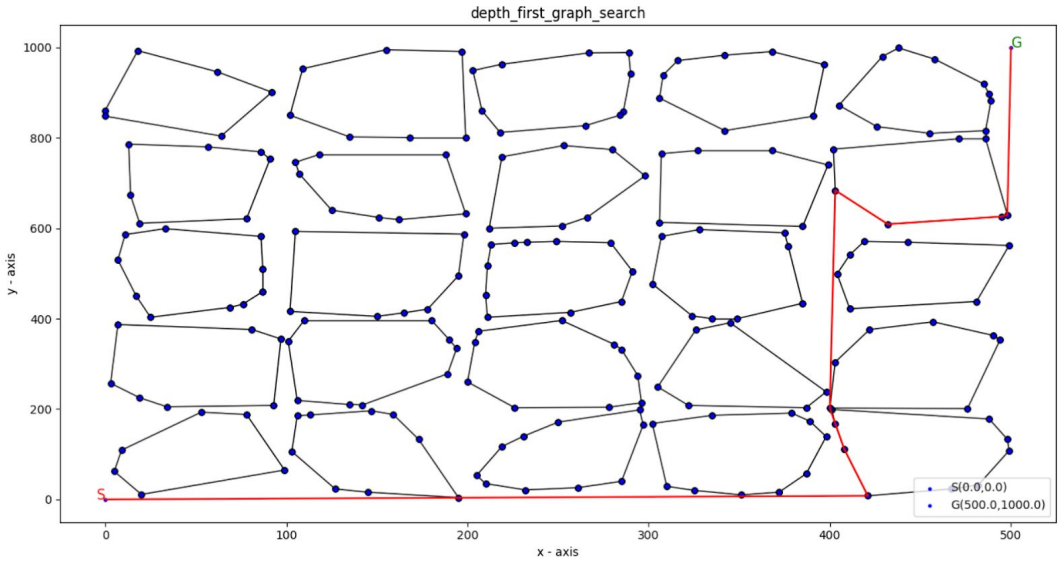

```
Time taken to create Visibility Graphs:  82.75644159317017

 breadth_first_graph_search
State:  1620
Successors:  115
Goal Tests:  242
Time: 0.02610
Cost: 1978.43610


 depth_first_graph_search
State:  179
Successors:  14
Goal Tests:  15
Time: 0.00649
Cost: 2251.23130


 best_first_graph_search
State:  84
Successors:  7
Goal Tests:  8
Time: 0.00447
Cost: 2240.68020


 astar_search
State:  2708
Successors:  181
Goal Tests:  182
Time: 0.25097
Cost: 1584.80920
```

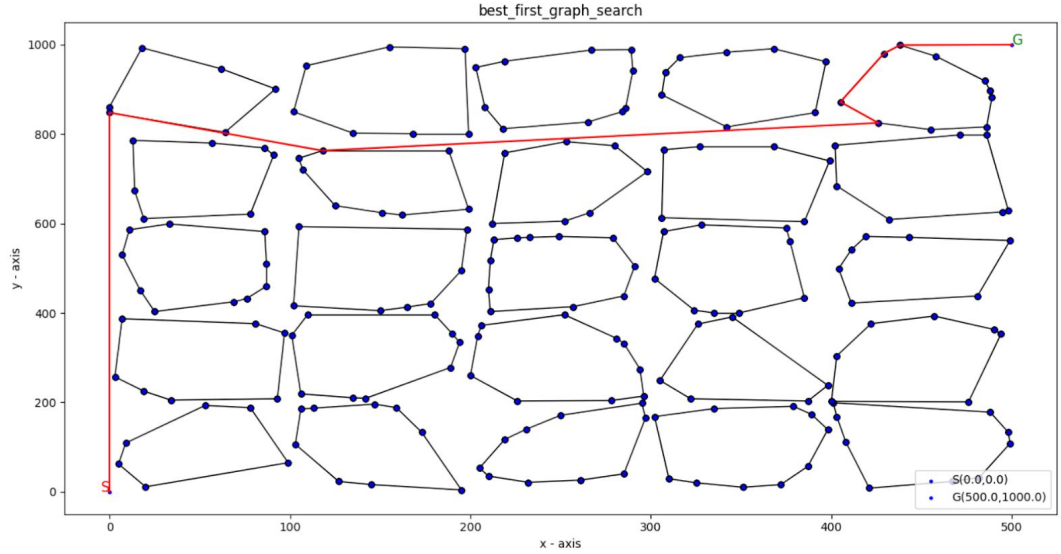
5. Instance 5

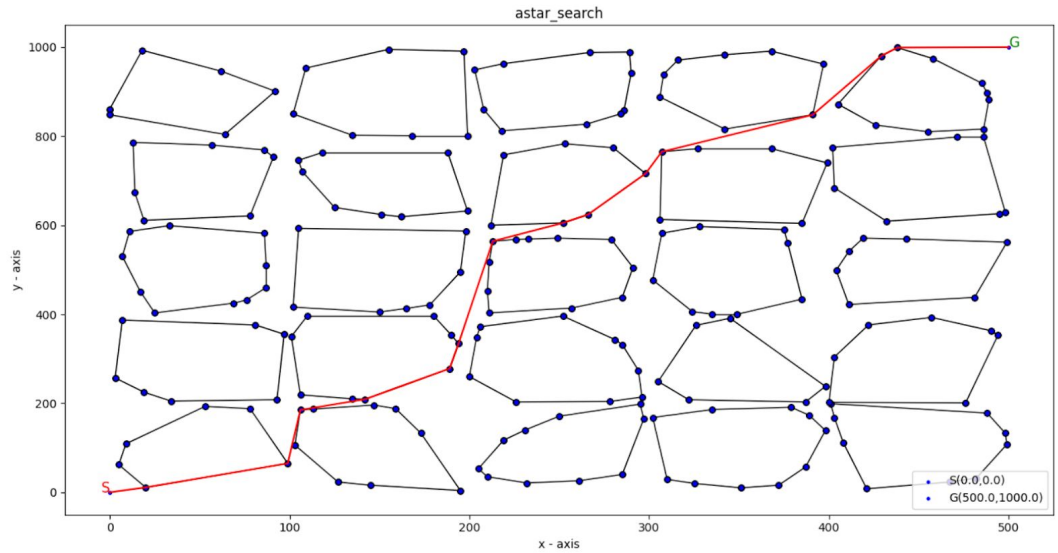

Visibility Graph
G
S
S(0.0,0.0)
G(500.0,1000.0)
0
100
200
300
400
500
0
200
400
600
800
1000


G
S
S(0.0,0.0)
G(500.0,1000.0)
y - axis
x - axis
0
100
200
300
400
500
0
200
400
600
800
1000


breadth_first_graph_search
G
S
S(0.0,0.0)
G(500.0,1000.0)
y - axis
x - axis
0
100
200
300
400
500
0
200
400
600
800
1000


depth_first_graph_search
y - axis
x - axis
S
G
S(0.0,0.0)
G(500.0,1000.0)


best_first_graph_search
y - axis
x - axis
S
G
S(0.0,0.0)
G(500.0,1000.0)


astar_search
y - axis
x - axis
S
G
S(0.0,0.0)
G(500.0,1000.0)

```
Time taken to create Visibility Graphs:  15.451735973358154

 breadth_first_graph_search
State:  2064
Successors:  154
Goal Tests:  197
Time: 0.03819
Cost: 1587.45240


 depth_first_graph_search
State:  83
Successors:  8
Goal Tests:  9
Time: 0.00119
Cost: 1618.92360


 best_first_graph_search
State:  71
Successors:  7
Goal Tests:  8
Time: 0.00898
Cost: 1552.74990


 astar_search
State:  1329
Successors:  92
Goal Tests:  93
Time: 0.09678
Cost: 1218.37560
```

For 100 Instances

- Average time taken to create **visibility graphs**: 20.0849s

Searcher	**States**	**Successors**	**Goal Tests**	**Time(s)**	**Cost**
BFS	1371	85	165	0.01303	1704.672
DFS	119	9	10	0.00141	1917.033
Greedy Best First Search	61	5	6	0.00143	1720.597
A* Search	1247	73	74	0.08892	1343.324

Note: All the instances used have 25 to 30 polygons. An instance with 50 polygons takes 80 - 100s to generate the visibility graph.

Conclusions

- ➔ **Path Cost**
  - ◆ A* search returns the optimal path with minimum path cost but it takes more time to find the path as compared to other algorithms.

- ➔ **Time**
  - ◆ DFS and Greedy takes the least amount of time as in greedy it chooses the nodes which are closest to the goal state and as a result it traverses very less number of states and DFS does not consider the path cost. But both of these algorithms return paths with a very high cost.