VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY

UNIVERSITY OF SCIENCE

REPORT PATH FINDING

COURSE: ARTIFICIAL INTELLIGENCE

TEACHER : Nguyen Ngoc Thao

ASSISTANT TEACHING: Nguyen Thai Vu

LIST OF STUDENT:

Hoang Huu Minh An	20127102
Nguyen Vo Minh Tri	20127364
Pham Minh Xuan	20127395

TABLE OF CONTENTS

I. STUDENT'S INFORMATION

ID	Name	Mail
20127102	Hoàng Hữu Minh An	20127102@student.hcmus.edu.vn
20127364	Võ Nguyễn Minh Trí	20127364@student.hcmus.edu.vn
20127395	Phan Minh Xuân	20127395@student.hcmus.edu.vn

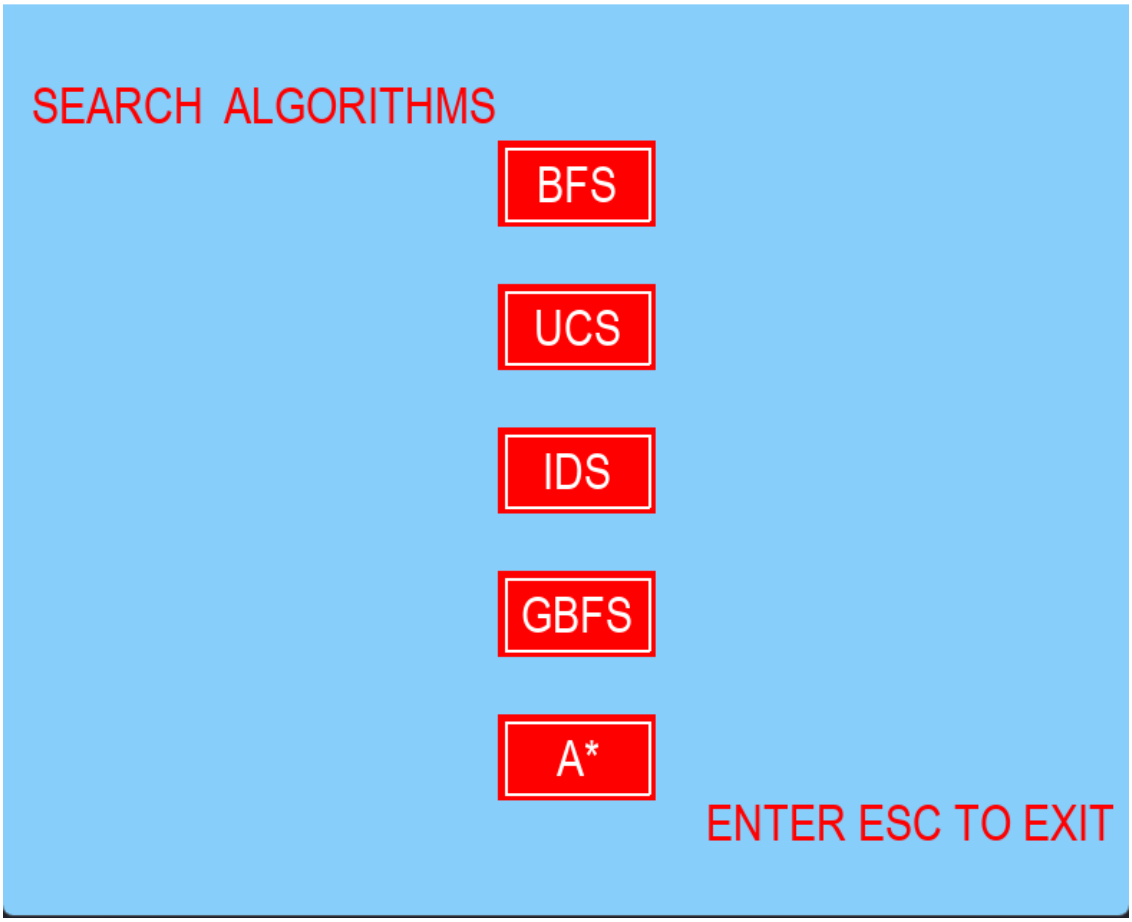
II. ANALYSTS:

✧ **Input:**

22 18 **2 2 19 16** **0**	**22 18** **2 2 19 16** **3** **4 4 5 9 8 10 9 5** **8 12 8 17 13 12** **11 1 11 6 14 6 14 1**

- The format of the input file:
  - First line: the size of the maze width, height.
  - Second line: the position of the Source and Goal. For example: 2 2 19 16 meaning source point is (2, 2) and goal point is (19, 16).
  - Third line: the number of the obstacles in the maze.
  - The next following line, defining the obstacle by the rule:
- The obstacle is a Convex polygon.
- Polygon is a set of points that are next to each other clockwise. The last point will be implicitly concatenated to the first point to form a valid convex polygon.

❖ **The program is running:**



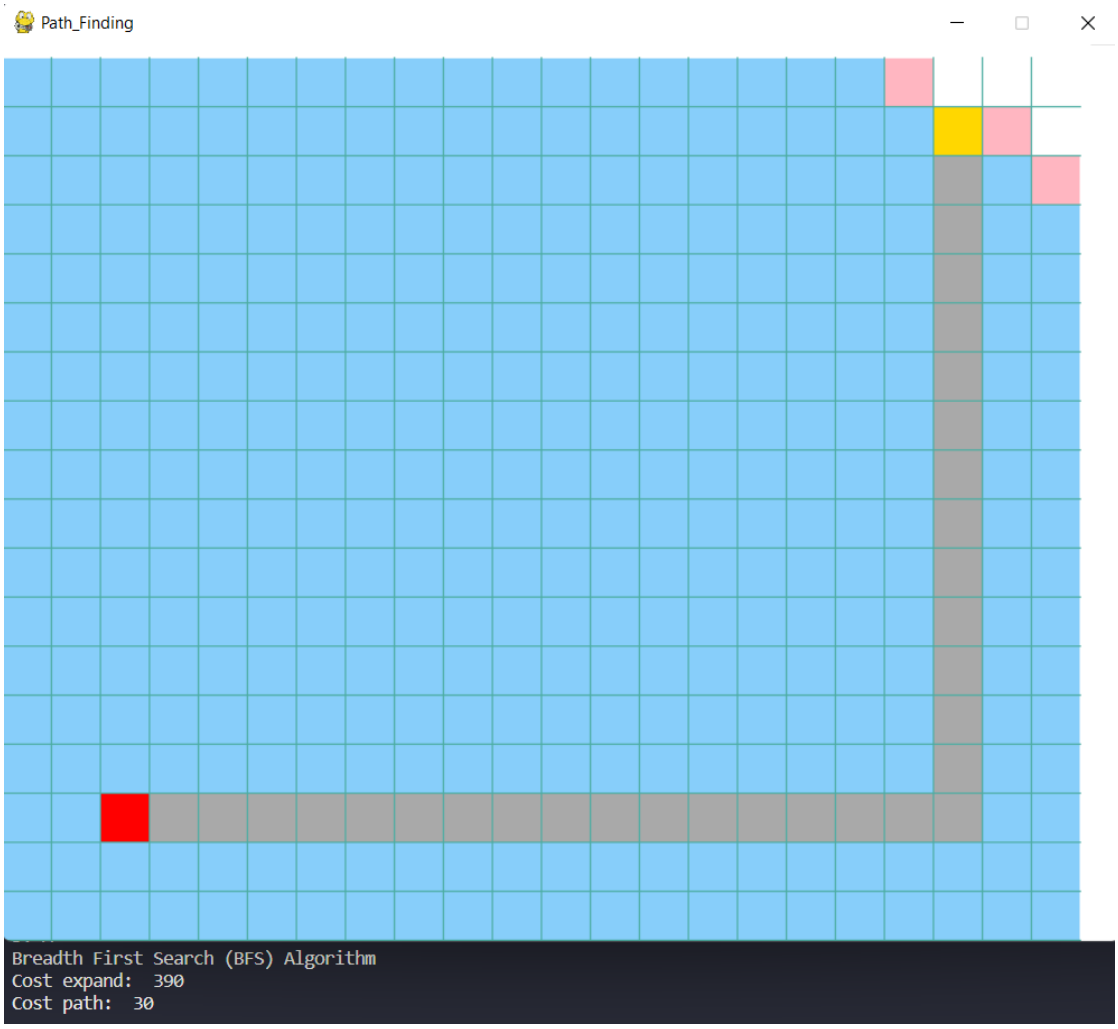
A. BFS (Breadth-first search)

1. Idea:

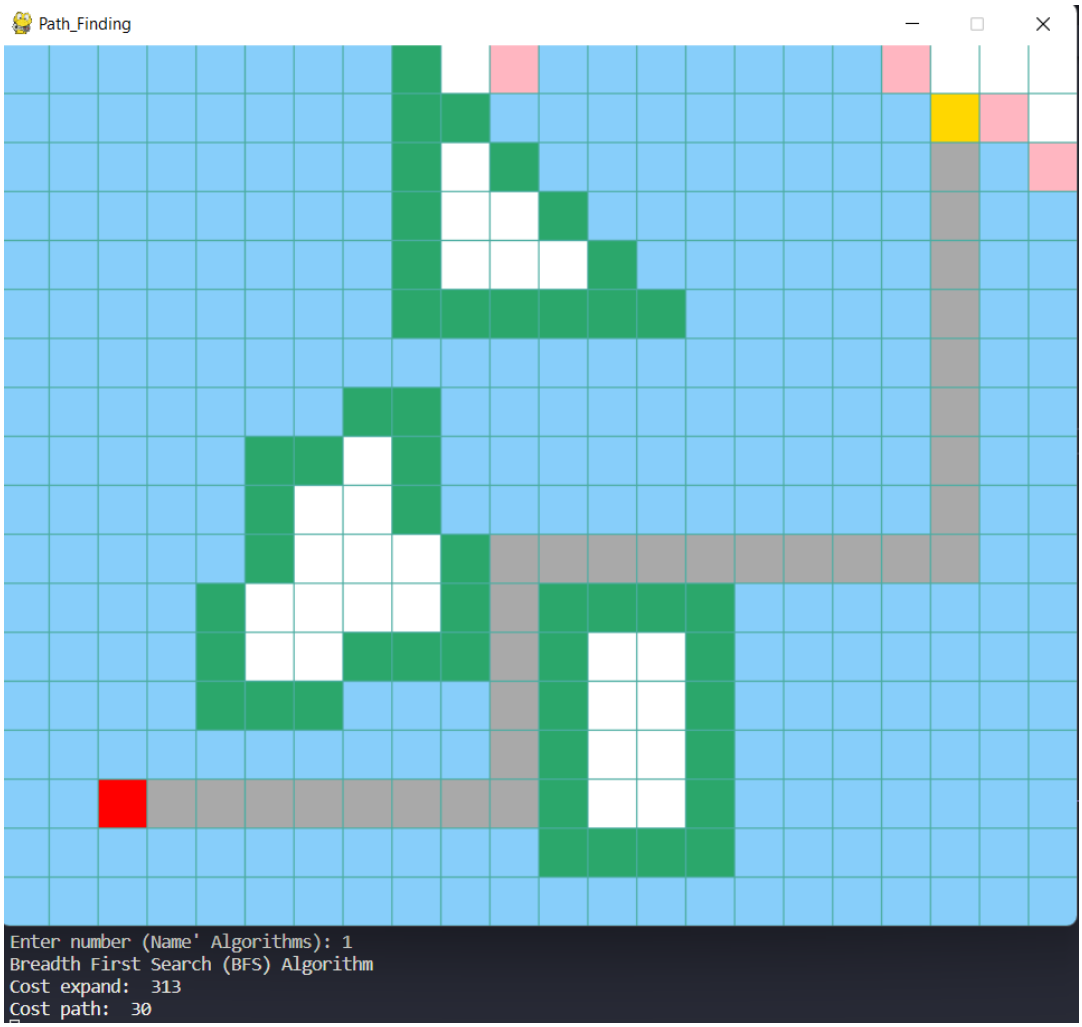
From a vertex, we find adjacent vertices and then traverse these vertices. Continue to find the adjacent vertices of the vertex just considered and then continue to traverse until all vertices can go.

2. Example

❖ **No obstacle**


Path_Finding
Breadth First Search (BFS) Algorithm
Cost expand: 390
Cost path: 30

❖ **Three obstacles**



3. Conclusion

✓ **Pros**

- If there is a solution, it will be offered by BFS.
- If there are multiple solutions to a certain issue, BFS will offer the one that has the fewest steps.

× **Cons**

- Use a lot of memory.
- BFS takes a long time if the solution is far from the node root.

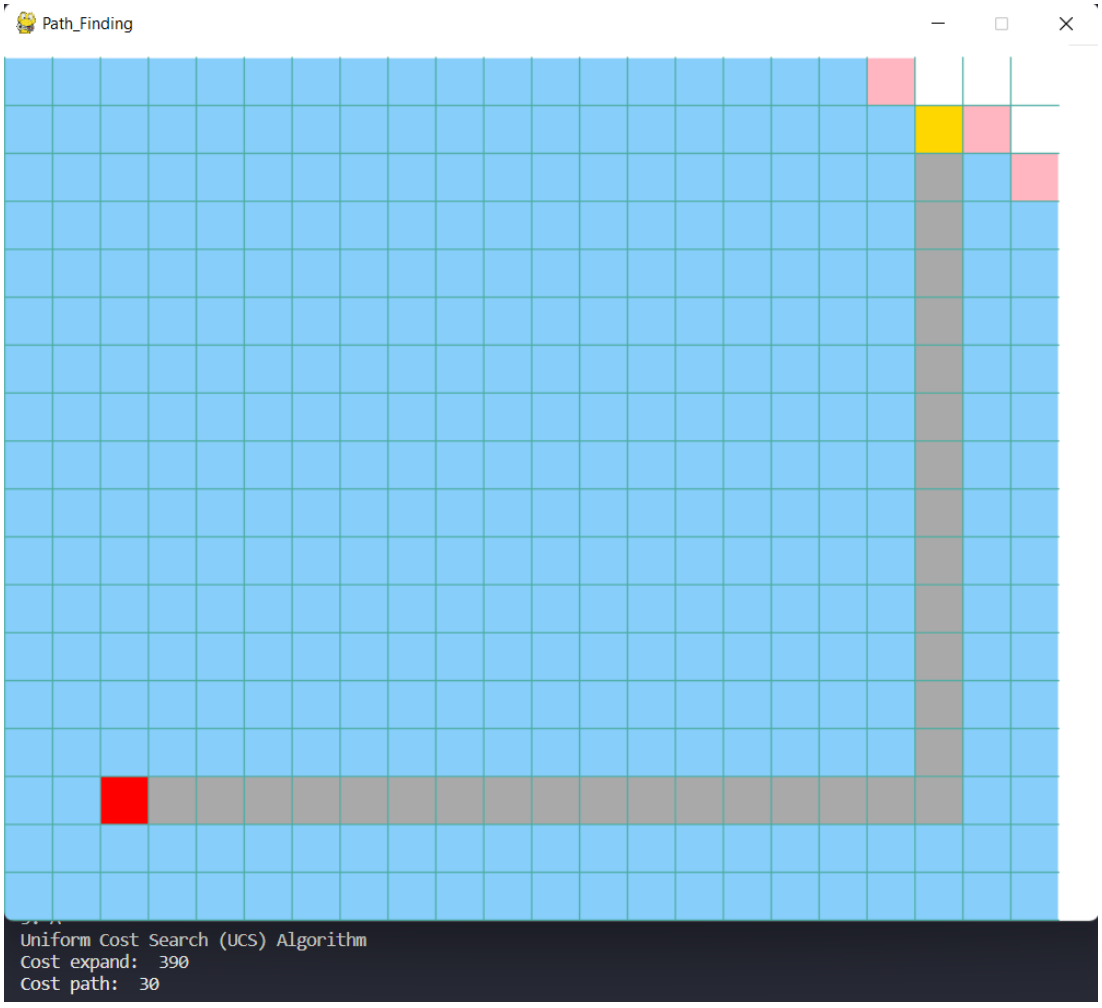
B. UCS(Uniform-cost search)

1. Idea:

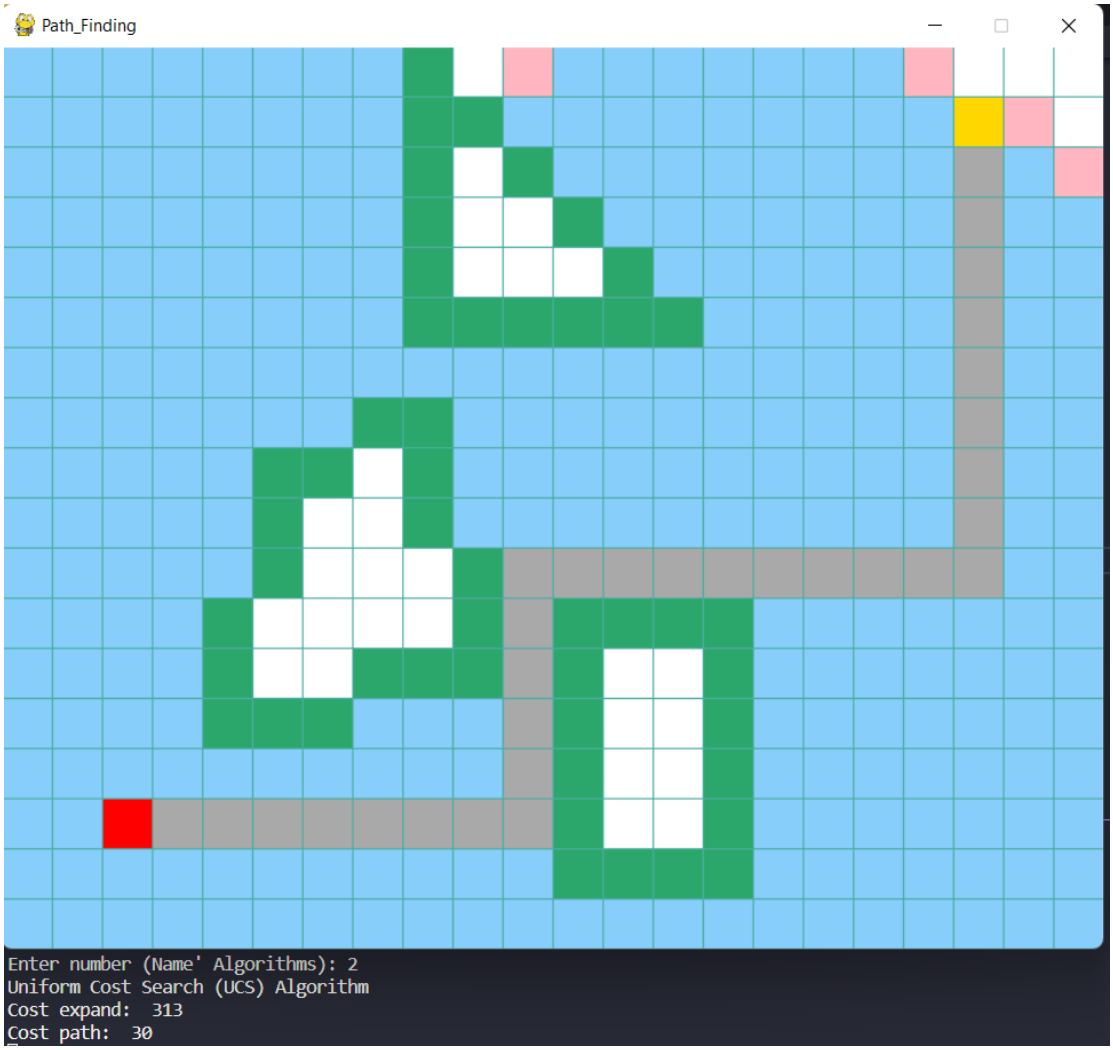
The UCS algorithm is a traversal algorithm that searches for a tree structure or a weighted graph . Using Priority Queue, the search starts at the root node and continues through the next nodes with the least weight or cost from the root node.

2. Example

❖ No obstacle



❖ **Three obstacles**



3. Conclusion

✓ **Pros**

- Find optimally path.
- Uniform expense search is coming first in light of the fact that in all expresses the picked minimal expense subpath.

× **Cons**

- It doesn't care about the number of steps involved in the search and only cares about the path cost. Hence this algorithm can get stuck in an infinite loop.

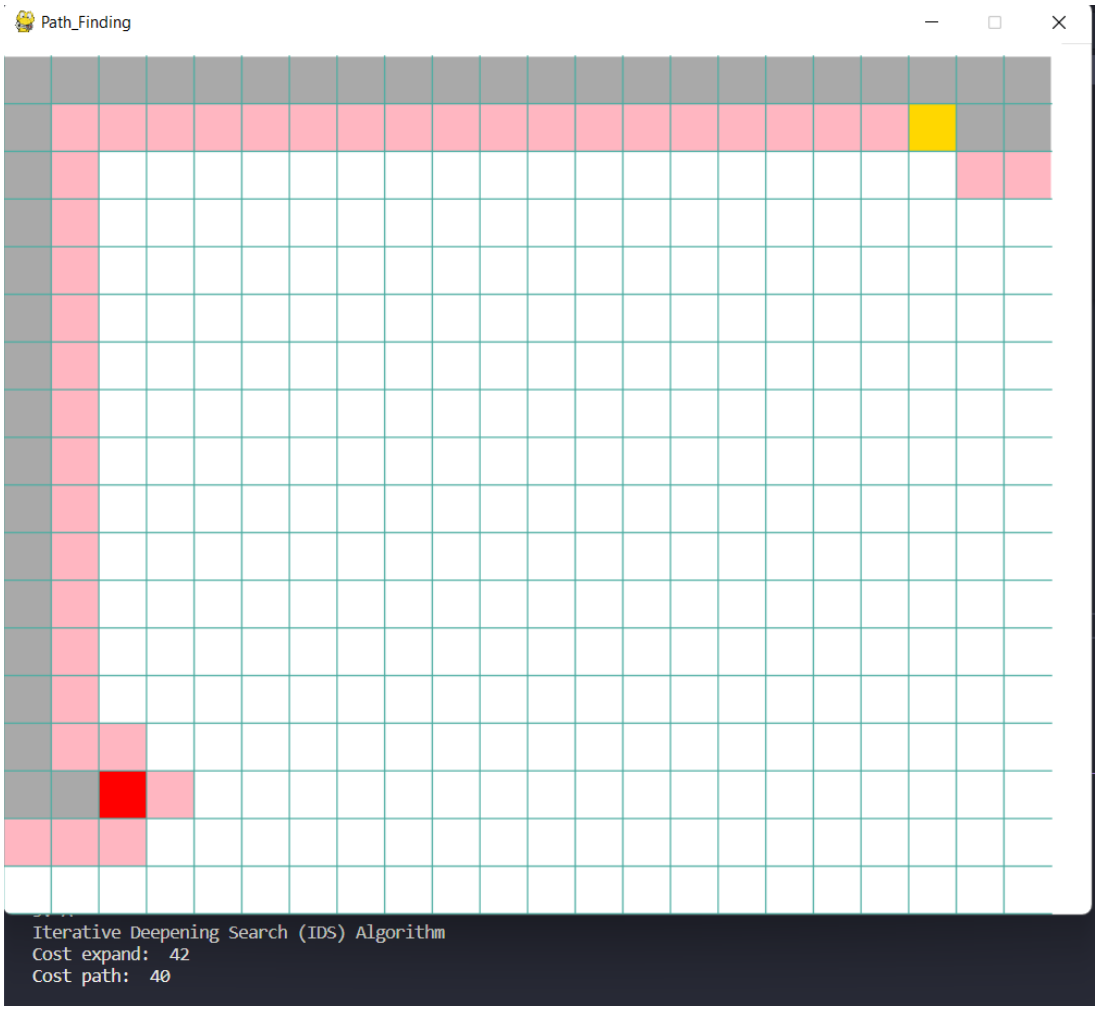
C. IDS(Iterative deepening search)

1. Idea:

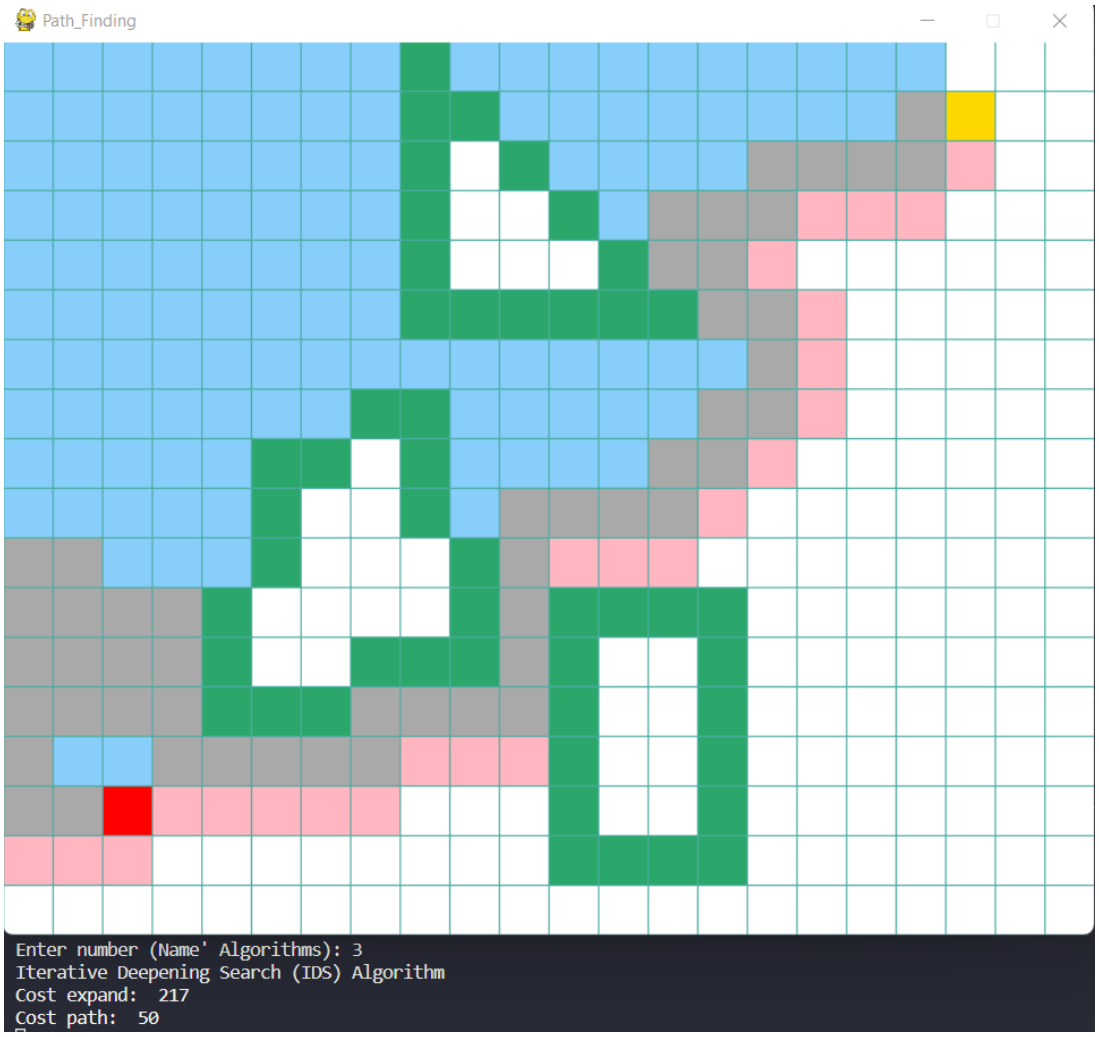
The DFS and BFS algorithms are combined to create the IDS algorithm. The best depth limit is determined by this search method, which then uses it by gradually raising the limit until the target is located.

2. Example

❖ **No obstacle**



❖ **Three obstacles**



3. Conclusion

✓ **Pros**

- It combines the benefits of BFS and DFS search algorithms in terms of fast searching.

× **Cons**

- Repeat all the work of the previous stage.
- Determination of depth until the search has proceeds.

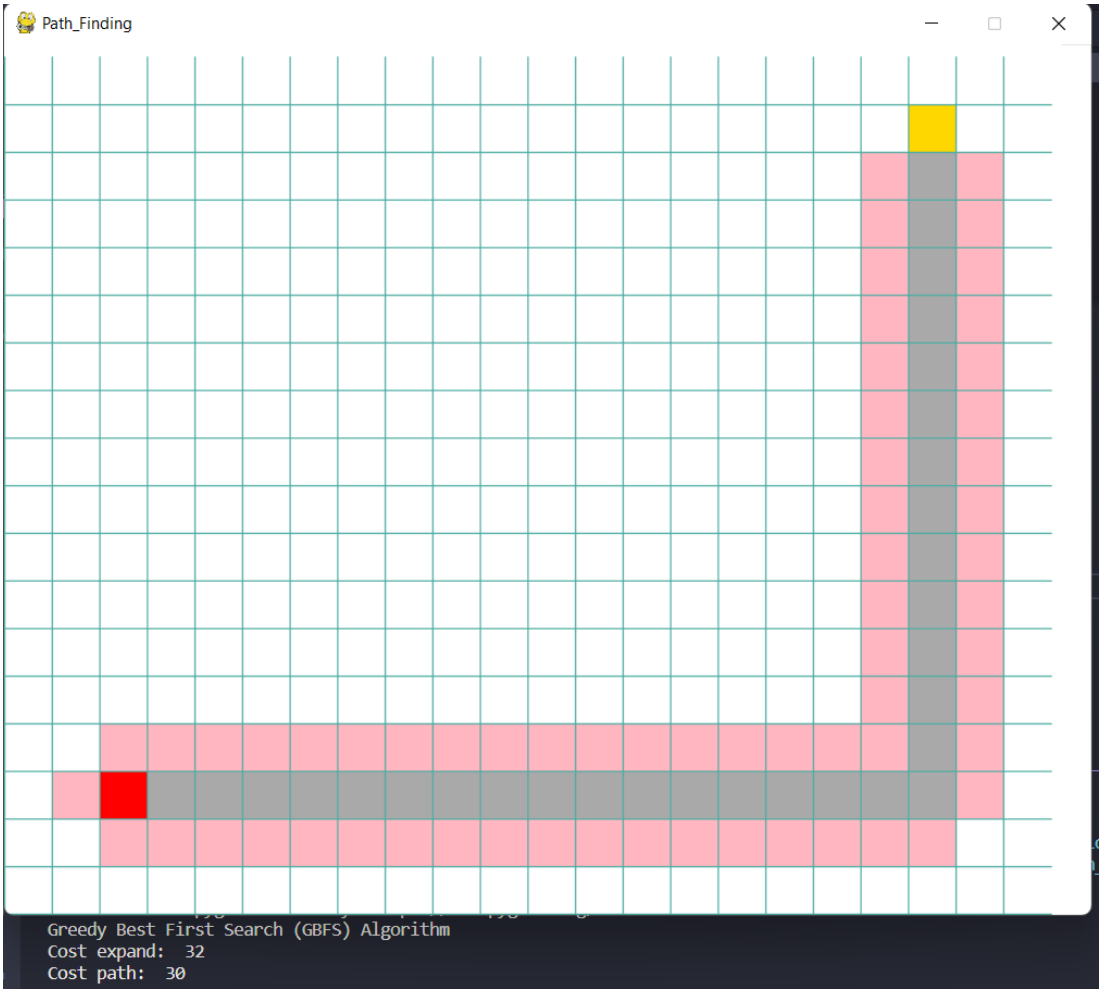
D. GBFS(Greedy-best first search)

1. Idea:

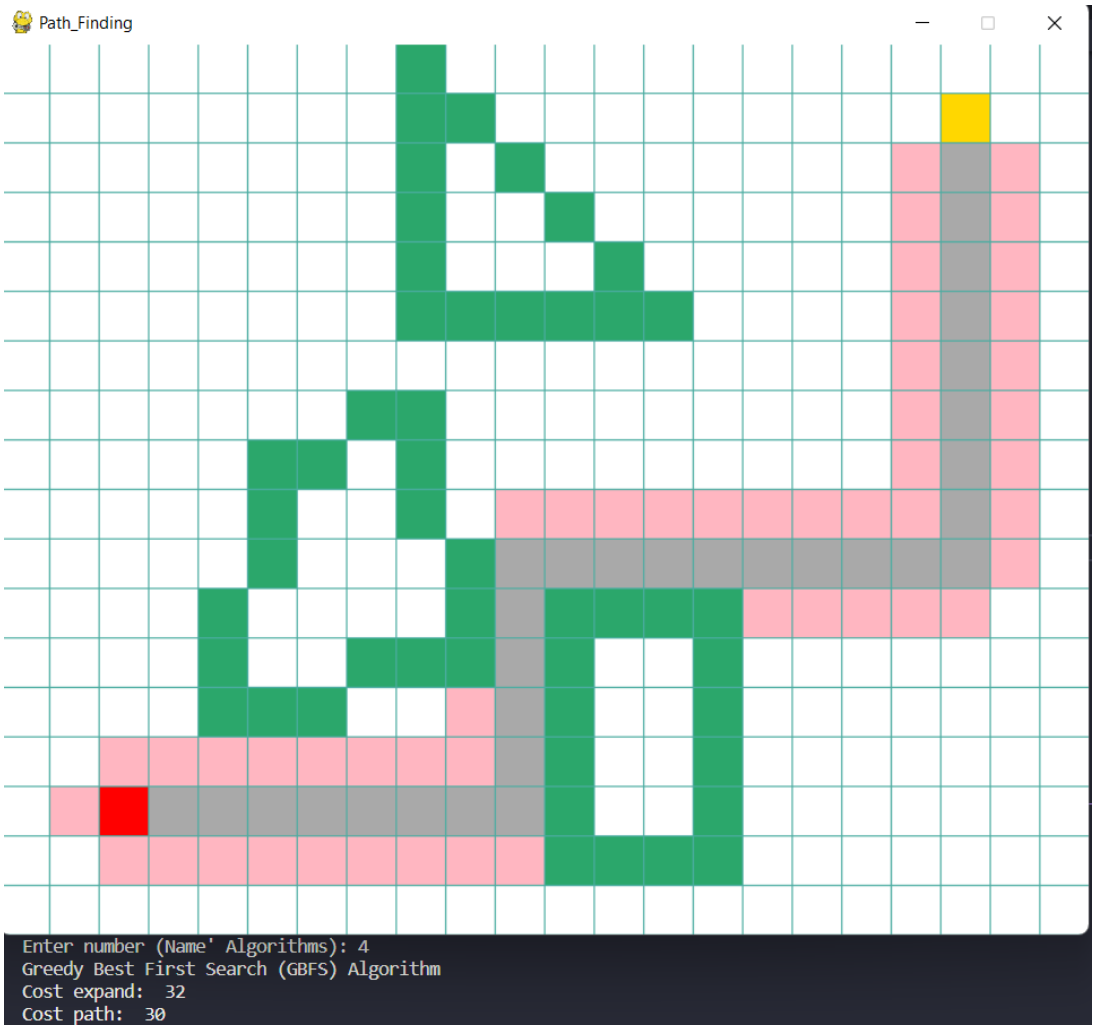
GBFS algorithm is that the search starts at the root node and continues by traversing the next nodes with the lowest heuristic value compared to the remaining nodes in the priority queue.

2. Example

❖ No obstacle



❖ **Three obstacles**



3. Conclusion

✓ **Pros**

- Can switch among BFS and DFS by acquiring the upsides of both.
- More productive than BFS and DFS.

× **Cons**

- It isn't optimal.
- It can get stuck in a loop as DFS

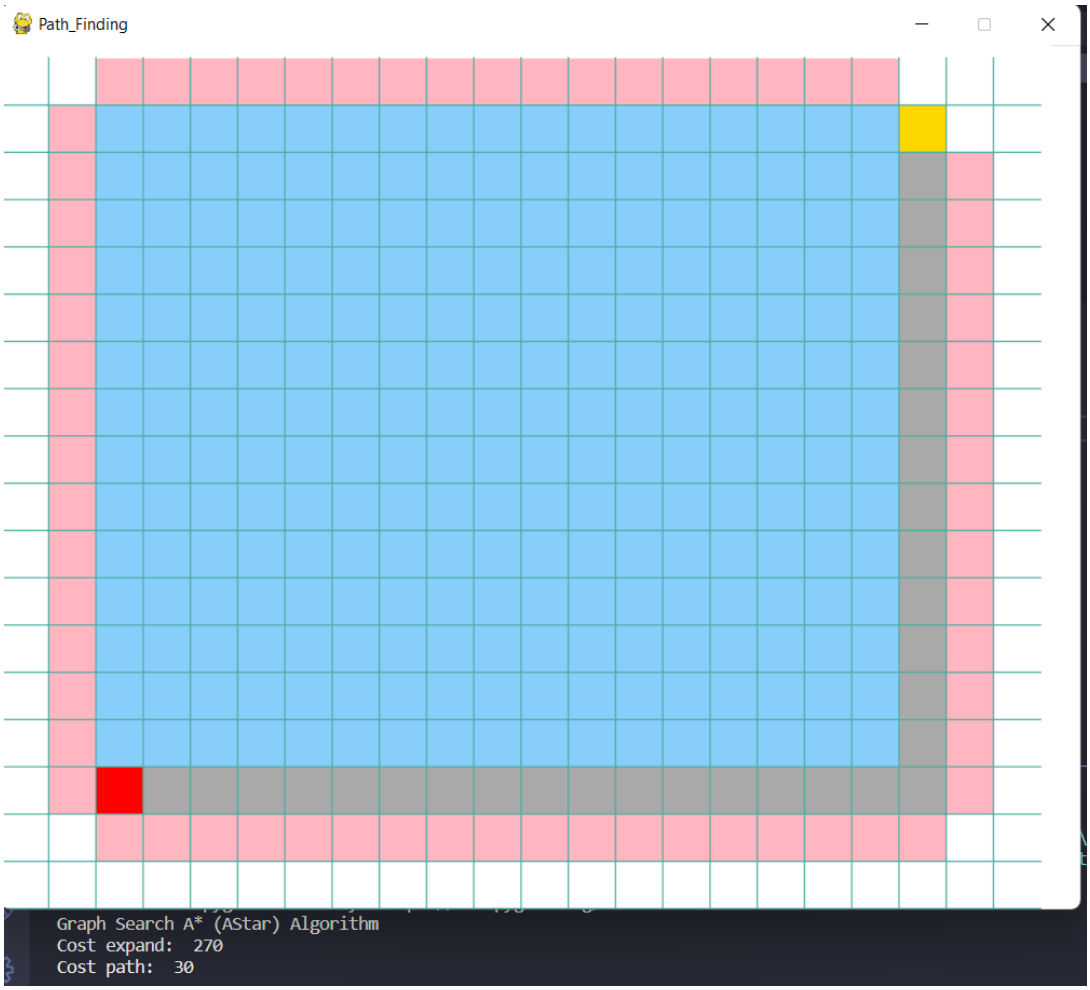
E. A*(Graph-search A*)

1. Idea:

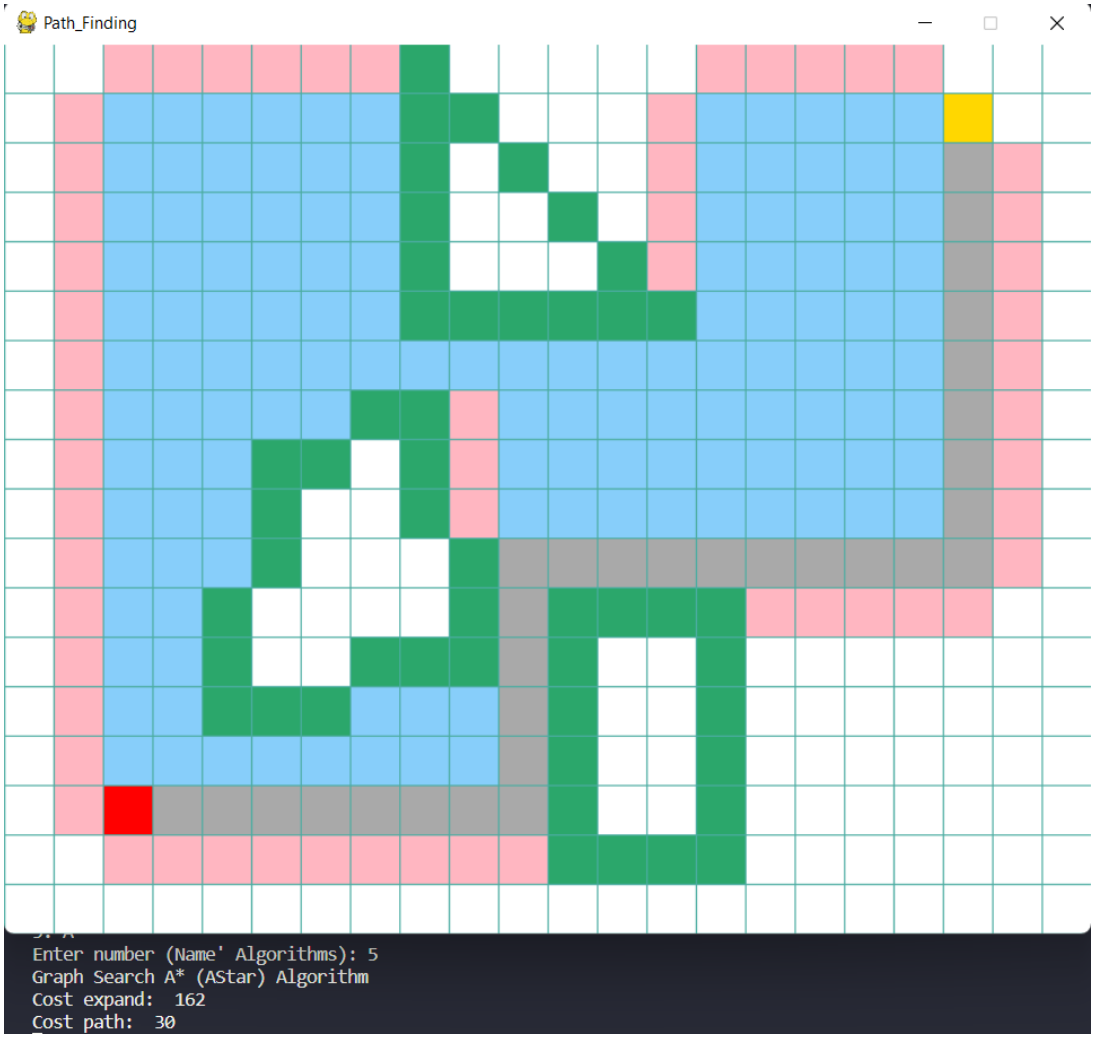
Graph search algorithm A*. This algorithm establishes a route between a starting node and a destination node. This technique ranks each node according to an estimate of the optimum path via that node using a "heuristic evaluation." By using the formula f=cost hueristic + weight graph, this method explores the nodes in the order of this heuristic evaluation.

2. Example

❖ No obstacle



❖ **Three obstacles**



3. Conclusion

✓ **Pros**

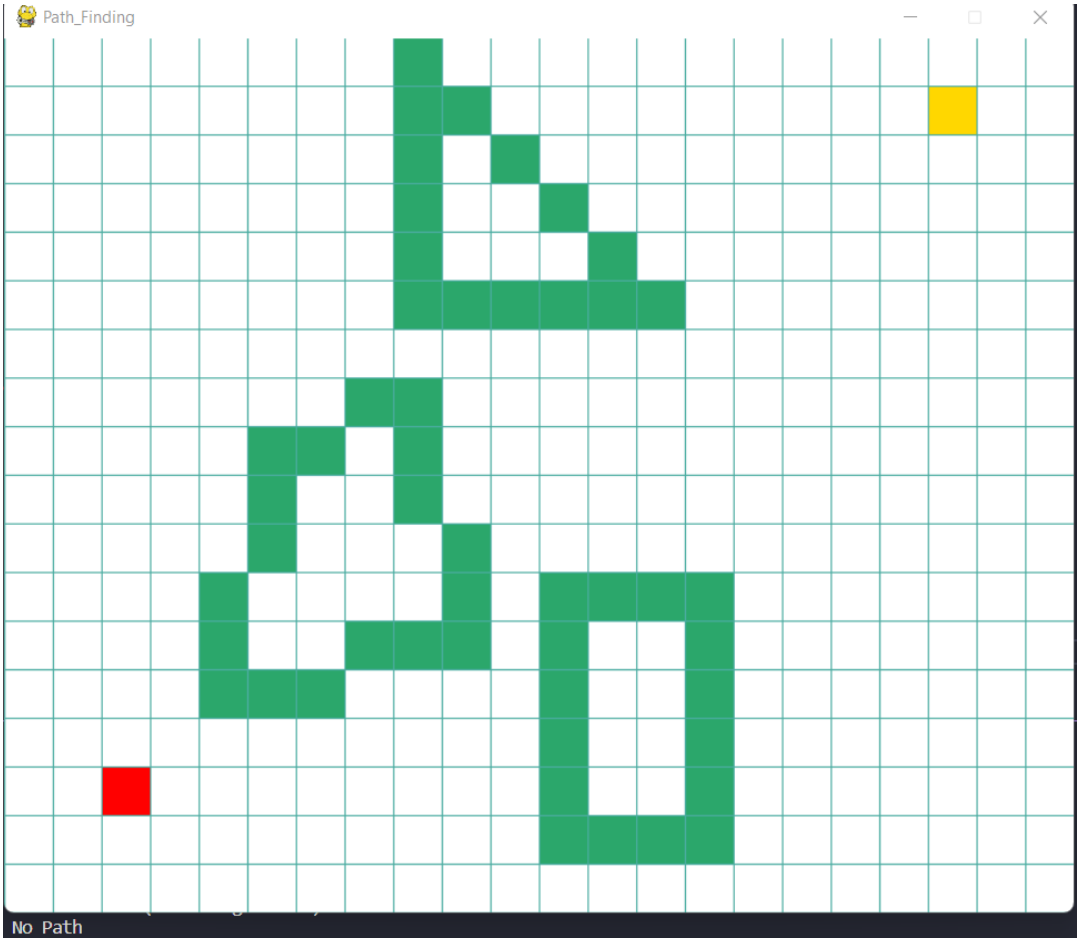
- Quickly find the solution with the orientation of the Heuristic function.
- It is used to solve complex search problems.

× **Cons**

- Use a lot of memory.
- It isn't optimal if the space of states is infinite.
- Performance is dependent on the accuracy of the heuristic algorithm used to compute.

❖ Opposite if it can't find the way:

- It will notify "**No Path**"



❖ NOTE:

Color in Path_Finding window:

- Path : gray color
- Expanded: blue color
- Unexpand : pink color
- Obstacle: green color
- Start: red color
- Goal: yellow color

III. COMPLEXITY

Algorithms	
Breadth First Search	Let the shallowest solution's depth be d. Complexity $O(b^d)$.
Uniform Cost Search	Let $C*$ be the cost of the optimal solution and assume that every action costs at least ε. Complexity $O(b^{1+ [C_*/\varepsilon]})$
Iterative Deepening Search	If the maximum depth m is finite. Complexity $O(b^m)$
Greedy Best First Search	If it is a graph-search instance in finite state spaces. Complexity $O(b^m)$
Graph Search A*	$O(b^d)$

IV. COMPLETION

Algorithms	Complete
Breadth First Search	100%
Uniform Cost Search	100%
Iterative Deepening Search	100%
Greedy Best First Search	100%
Graph Search A*	100%

V. REFERENCES

Greedy Best First Seach
https://en.wikipedia.org/wiki/Best-first_search
AStar
https://en.wikipedia.org/wiki/A*_search_algorithm
Course Material
https://drive.google.com/drive/folders/1OmW_a959zfyCfWP__agipFzfEZX_eV1S
Use Pygame
https://codelearn.io/sharing/lap-trinh-game-co-ban-voi-pygame