ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
---------------o0o---------------

FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE

LAB 1: SEARCH

Lecture: **Bùi Tiến Lên**

TA: **Nguyễn Thái Vũ**

INDEX

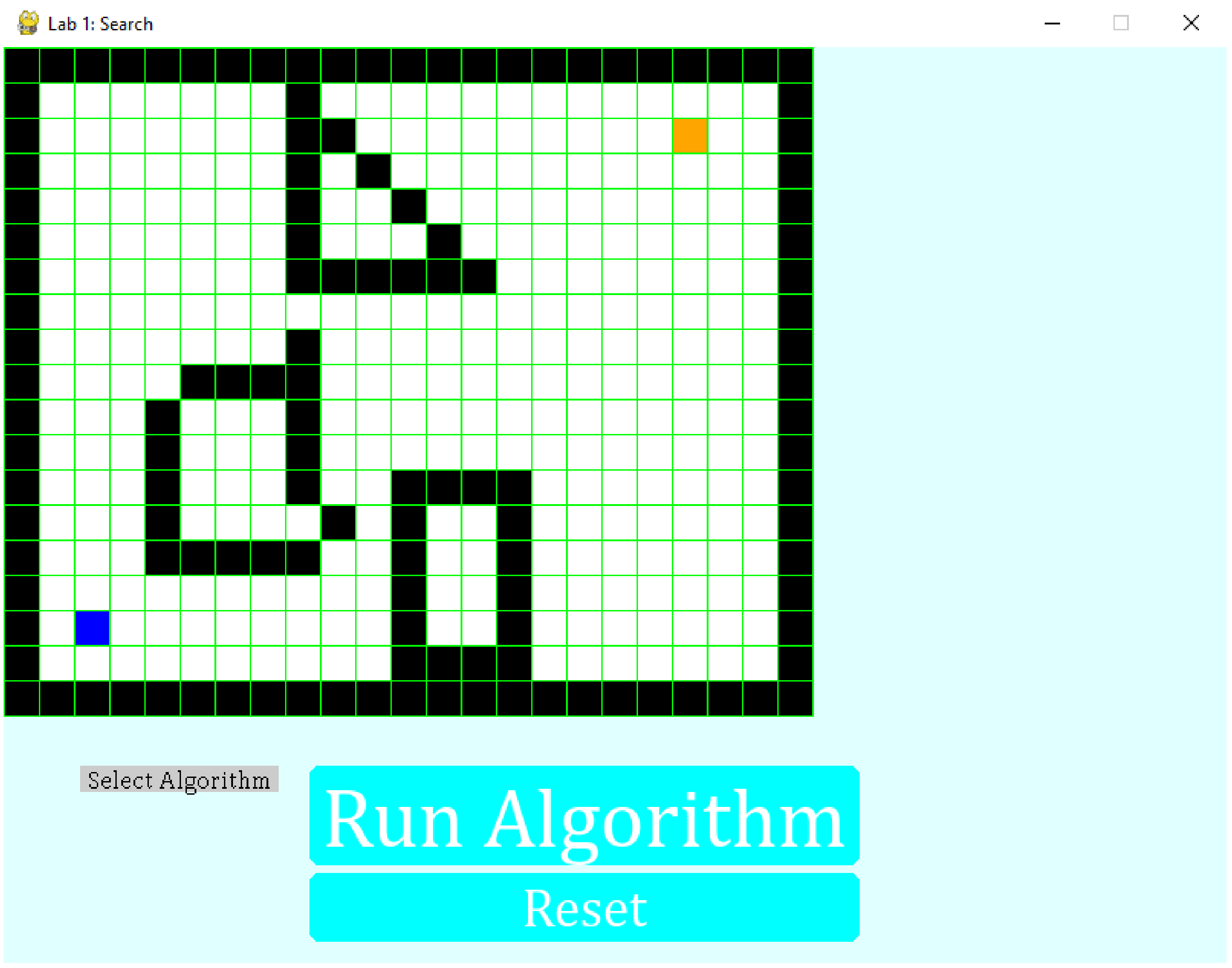
Content

A. Student’s information:

- Name: Đặng Ngọc Tiến
- ID: 20127641

B. Search algorithms, pros and cons, example:

Programming language: Python



1. Breadth First Search:

- Idea:

- Breadth-first search is the most common search strategy for traversing a tree or graph. This algorithm searches breadthwise in a tree or graph, so it is called breadth-first search.
- BFS algorithm starts searching from the root node of the tree and expands all successor node at the current level before moving to nodes of next level.
- The breadth-first search algorithm is an example of a general-graph search algorithm.
- Breadth-first search implemented using FIFO queue data structure.

- Pros:
  - BFS will provide a solution if any solution exists.
  - If there are more than one solutions for a given problem, then BFS will provide the minimal solution which requires the least number of steps
- Cons:
  - It requires lots of memory since each level of the tree must be saved into memory to expand the next level.
  - BFS needs lots of time if the solution is far away from the root node.
- Input/ output:
  - Input: start node
  - Output: path and cost

2. Uniform Cost Search:

- Idea:

  Uniform-cost search is a searching algorithm used for traversing a weighted tree or graph. This algorithm comes into play when a different cost is available for each edge. The primary goal of the uniform-cost search is to find a path to the goal node which has the lowest cumulative cost. Uniform-cost search expands nodes according to their path costs form the root node. It can be used to solve any graph/tree where the optimal cost is in demand. A uniform-cost search algorithm is implemented by the priority queue. It gives maximum priority to the lowest cumulative cost. Uniform cost search is equivalent to BFS algorithm if the path cost of all edges is the same.
- Pros:
  - Uniform cost search is optimal because at every state the path with the least cost is chosen.

- Cons:
  - It does not care about the number of steps involve in searching and only concerned about path cost. Due to which this algorithm may be stuck in an infinite loop.
- Input/ output:
  - Input: start node
  - Output: path and cost

3. Iterative Deepening Search:

- Idea:
  - The iterative deepening algorithm is a combination of DFS and BFS algorithms. This search algorithm finds out the best depth limit and does it by gradually increasing the limit until a goal is found.
  - This algorithm performs depth-first search up to a certain "depth limit", and it keeps increasing the depth limit after each iteration until the goal node is found.
  - This Search algorithm combines the benefits of Breadth-first search's fast search and depth-first search's memory efficiency.
  - The iterative search algorithm is useful uninformed search when search space is large, and depth of goal node is unknown.
- Pros:
  - Itcombines the benefits of BFS and DFS search algorithm in terms of fast search and memory efficiency.
- Cons:
  - The main drawback of IDDFS is that it repeats all the work of the previous phase
- Input/ output:
  - Input: start node and depth limit
  - Output: path and cost

4. A* Search:

- Idea:

A* search is the most commonly known form of best-first search. It uses heuristic function h(n), and cost to reach the node n from the start state g(n). It has combined features of UCS and greedy best-first search, by which it solve the problem efficiently. A* search algorithm finds the shortest path through the search space using the heuristic function. This search algorithm expands less search tree and provides optimal result faster. A* algorithm is similar to UCS except that it uses g(n)+h(n) instead of g(n).

- Pros:
  - A* search algorithm is the best algorithm than other search algorithms.
  - A* search algorithm is optimal and complete.
  - This algorithm can solve very complex problems.
- Cons:
  - It does not always produce the shortest path as it mostly based on heuristics and approximation.
  - A* search algorithm has some complexity issues.
  - The main drawback of A* is memory requirement as it keeps all generated nodes in the memory, so it is not practical for various large-scale problems.
- Input/ output:
  - Input: start node and end node
  - Output: path and cost

5. Greedy Best First Search:

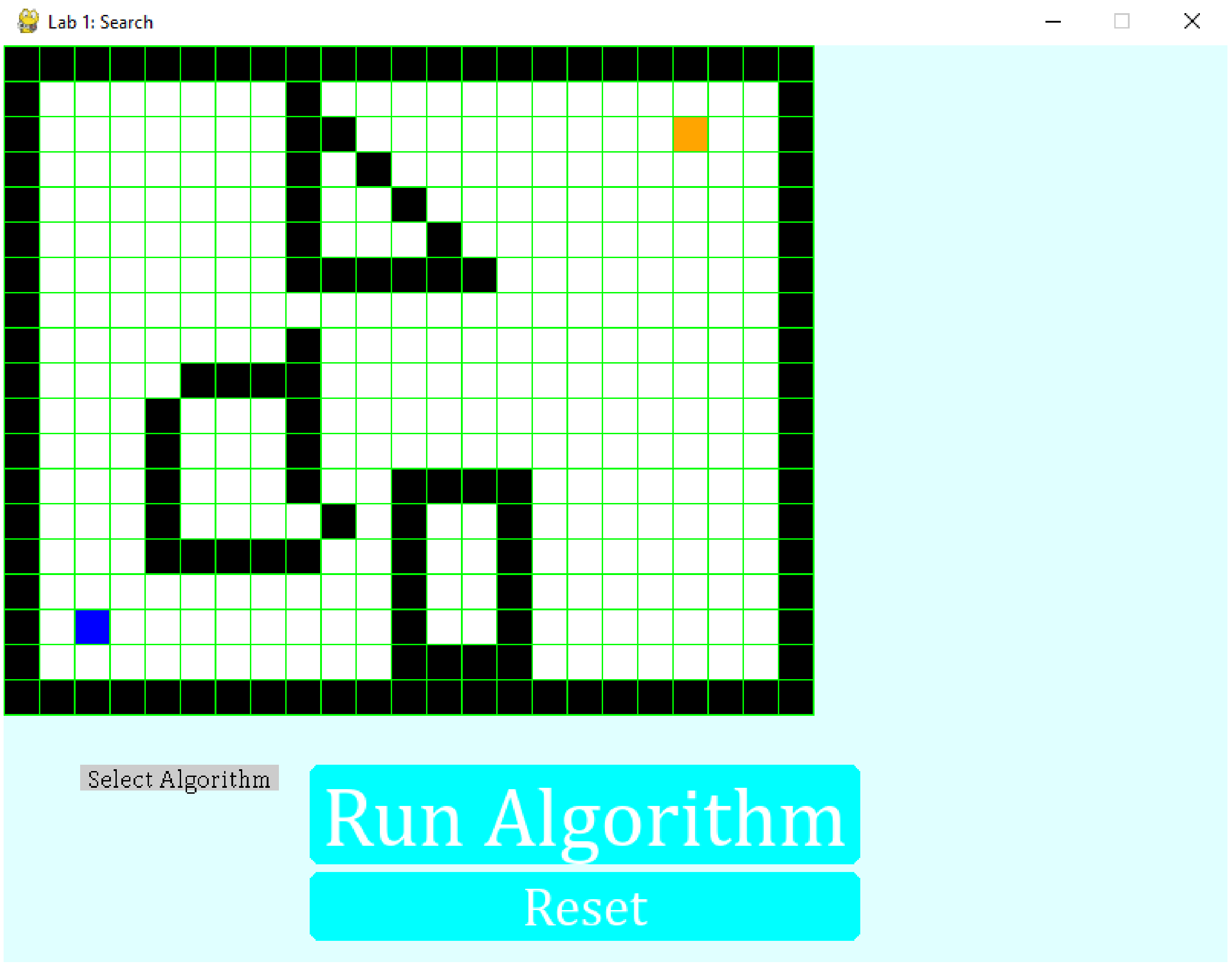
- Idea:

Greedy best-first search algorithm always selects the path which appears best at that moment. It is the combination of depth-first search and breadth-first search algorithms. It uses the heuristic function and search. Best-first search allows us to take the advantages of both algorithms. With the help of best-first search, at each step, we can choose the most promising node. In the best first search algorithm, we expand the node which is closest to the goal node and the closest cost is estimated by heuristic function:

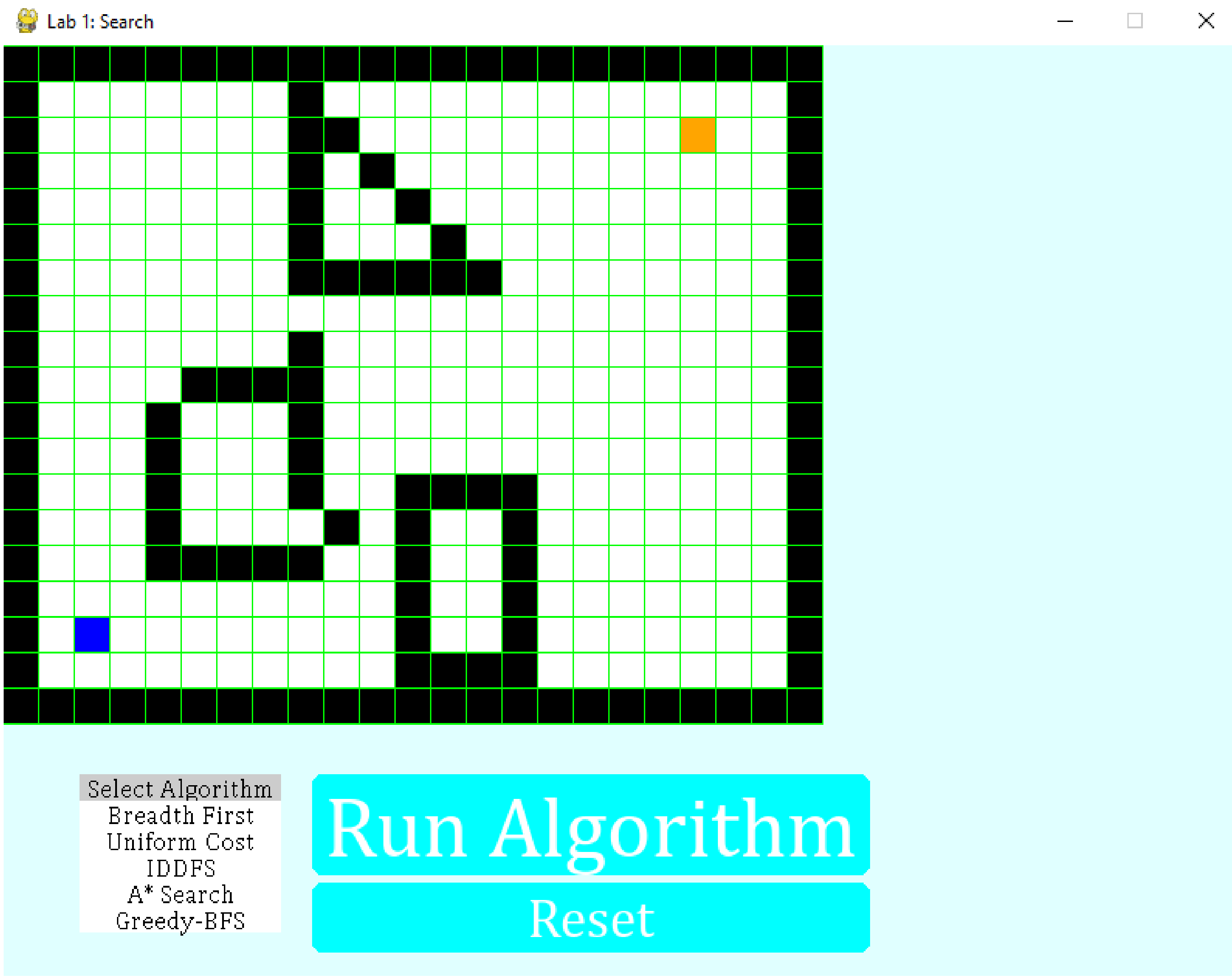
- Pros:

    - Best first search can switch between BFS and DFS by gaining the advantages of both the algorithms.
    - This algorithm is more efficient than BFS and DFS algorithms.
- Cons:
    - It can behave as an unguided depth-first search in the worst-case scenario.
    - It can get stuck in a loop as DFS.
    - This algorithm is not optimal.
- Input/ output:
    - Input: start node and end node
    - Output: path and cost

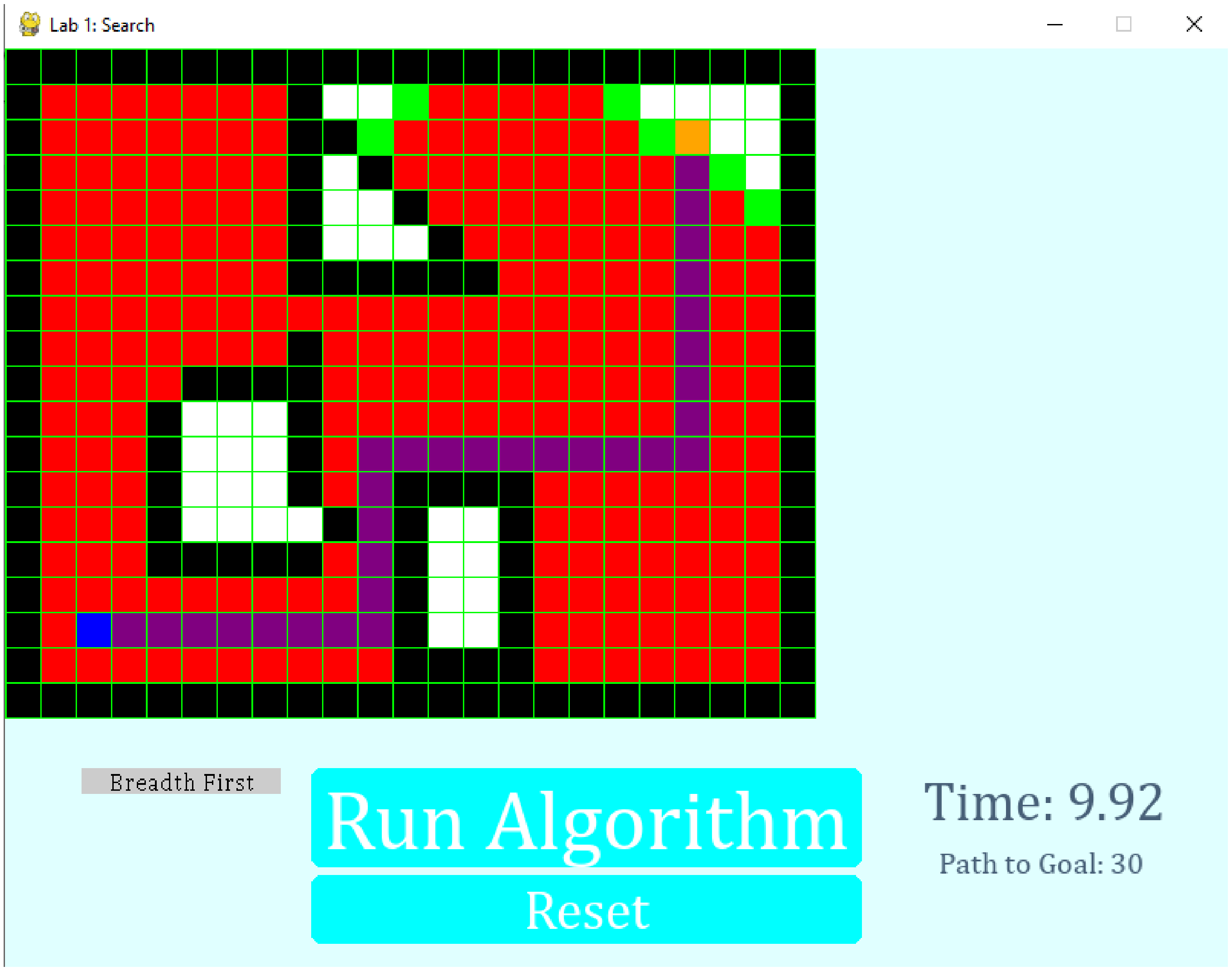
B. Demo:



- Select algorithm:



- Run algorithm:



Time: 9.92

Path to Goal: 30