Islamic University of Technology

Artificial Intelligence Lab

CSE 4712

Lab 2

Author:
Ishmam Tashdeed
CSE, IUT

Contents

1 Tasks

In the last lab you learned about uninformed search algorithms such as: BFS, DFS, and UCS. In this lab, you will learn how to formulate new problems as search problems and solve them using those algorithms. You are already given a complete implementation of these algorithms. Your job is **NOT** to modify the search logic but to design new problems that use the search engine such as:

```
uninformed_search(
    start_state,
    goal_test_fn,
    successor_fn,
    cost_fn,
    strategy='bfs'
)
```

Here, you will have to define the `start_state`, `goal_test_fn`, `successor_fn`, `cost_fn`. After formulating each of the problems, you need to compare the performance of BFS, DFS, and UCS on each problem.

1.1 Task 1

You will be given a graph and you will have to formulate:

- State representation (how to describe each state)
- Successor function (returns all neighbors of a node)
- Goal test
- Start state
- Cost function (usually a constant value *i.e.* 1, unless you choose to use UCS with varying costs, in which case the number of nodes that need to be visited before reaching a certain node is the cost of that node)

The graph will be given as a `Dictionary` where the keys are parent nodes and values are children. Your implementation should output the **sequence of nodes** that are in the path from the start to the goal node.

Sample Input:

```
graph = {
    'A': ['B', 'C'],
    'B': ['D', 'E'],
    'C': ['F'],
    'D': [],
    'E': ['F'],
    'F': []
}
```

Sample Output (BFS from node 'A'):

```
A B C D E F
```

1.2 Task 2

Suppose there in a robot in an $M \times N$ grid who has k piles of trash to clean. The robot wants to clean the trash piles with minimal traversal of the grids to conserve battery while avoiding obstacles. Formulate this scenario as a search problem where you will state:

- State representation (how to describe each state)
- Successor function (returns all neighbors of a node)
- Goal test
- Start state (any grid from the collection of grids)
- Cost function (usually a constant value *i.e.* 1, unless you choose to use UCS with varying costs, in which case the number of grid positions that need to be visited before reaching a certain position is the cost of that position)

The world will be given as a `String` where `'R'` denotes the starting position of the robot, `'T'` denotes a trash pile, `'.'` denotes an empty grid, and `'#'` denotes an obstacle. Your implementation should output the **sequence of actions** that are in the path from the start to the goal node.

Sample Input:

```
########
#R...T.#
#..##..#
#..#..T#
########
```

Sample Output:

```
Actions: right, right, right, right, down, down, right
```

1.3 Task 3

Design a function that can estimate how close you are to your goal for both of the previous tasks. This function does not need to output the exact cost needed to reach the goal (how would you know the exact cost before reaching the goal?), rather a rough estimation of the cost. The function you will design should:

- give an estimate of "how close" the state is to the goal
- be fast to compute (faster than just running a search algorithm and reaching the goal state)

Design such a function for both Task 1 and Task 2.