SSN College of Engineering

Department of Computer Science and Engineering

CS1504 — Artificial Intelligence

2020 – 2021

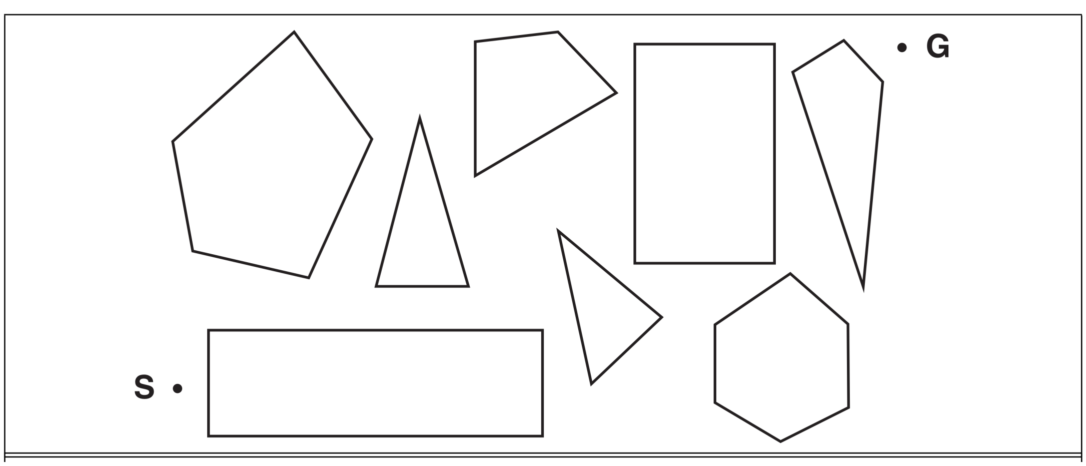
Session — 03

September 12, 2020

- This homework is due by 11:30pm on September 16, 2020
- Grace period may be given up to midnight on September 16, 2020
- Coding should be done using Python programming language — you may use online coding platforms such as `https://repl.it/`
- Reference code available at `https://github.com/aimacode/aima-python` may be used
- Plagiarism is strictly prohibited and strict academic actions may be taken against those who violate
- You can upload only one ZIP file
- The naming convention is "<Your first name (first letter capital and all the other letters small)>-CS1504-S02.zip"

---

1. 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.



(a) How do we formulate the state-space? How many states are there? How many paths are there to the goal? Think carefully to define a good state-space. Justify your decisions.

(b) How do we define the actions in your formulation of the state-space?

(c) Formulate this problem in Python by subclassing the Problem class in "search.py" of the reference implementation. Take extra care to implement ACTIONS to find successor states of a given state.

(d) Define your evaluation function to evaluate the goodness or badness of a state using an admissible (and, preferable consistent as well) heuristics function (it should be easy to find a heuristics for this problem!)

(e) Create several instances (at least 100) of this problem by randomly generating planes with random start and goal points and random polygons as obstacles.

(f) Solve all the instances using the following search strategies:

   - Any basic strategy of your choice (DFS/BFS/IDS)
   - Best-first greedy search
   - A^* search

   You may use the reference Python code to implement these search algorithms.

(g) Perform an empirical analysis in terms of number of nodes generated, expanded, actual time taken, completeness, optimality, etc. Which algorithm performs better, in general, on all the instances?