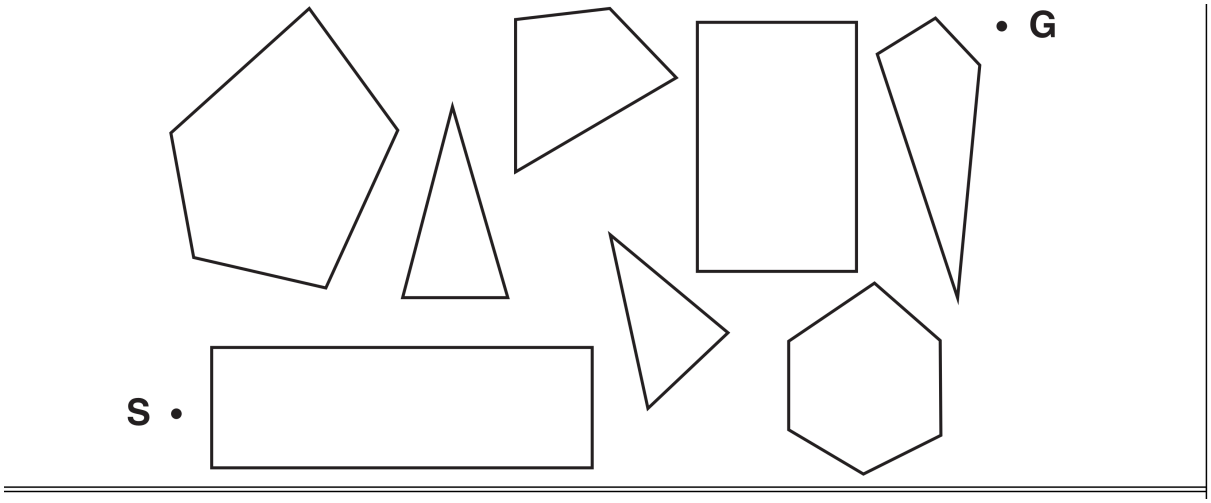
4. aConsider an autonomous mobile robot in a crowded environment that needs to find an efficient path from its current location S to a desired location G. As an idealization of the situation, assume that the obstacles (whatever they may be) are abstracted by polygons. The problem now reduces to finding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

(c) Formulate this problem in Python by subclassing the Problem class in “search.py” of the reference implementation.

(d) Define your evaluation function to evaluate the goodness or badness of a state

(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?