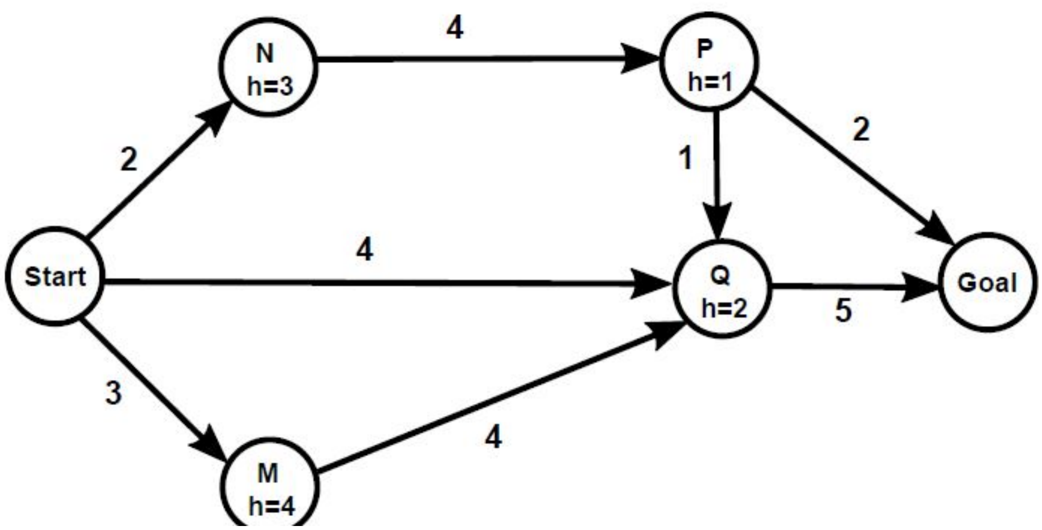
Assignment 1

Due Feb 10 by 11:59pm **Points** 80 **Submitting** a file upload
Available after Jan 26 at 12am

```
Submit the assignment as a PDF document. You must typeset the answers; u
se your favorite text/doc editor.
Submit the code separately (e.g., .java/.py/.cpp files). Make sure to at
tach the screenshots of your results for the coding question.
All the files should be zipped together and submitted to Canvas.
```

Answer the following questions (no programming required -- show complete calculations)

- Consider the n-queens problem using the “efficient” incremental formulation given on page 72. Explain why the state space has at least $\sqrt[3]{n!}$ states and estimate the largest n for which exhaustive exploration is feasible. [10 points]
  - Hint: Derive a lower bound on the branching factor by considering the maximum number of squares that a queen can attack in any column.
- Compute the order in which states of the above graph are expanded and the the returned path for each of these graph search methods:
  - Depth-first search
  - Breadth-first search
  - Uniform cost search
  - Greedy search using the heuristic h shown on the graph. [4 * 5 = 20 points]



Write a program to solve the following problem.

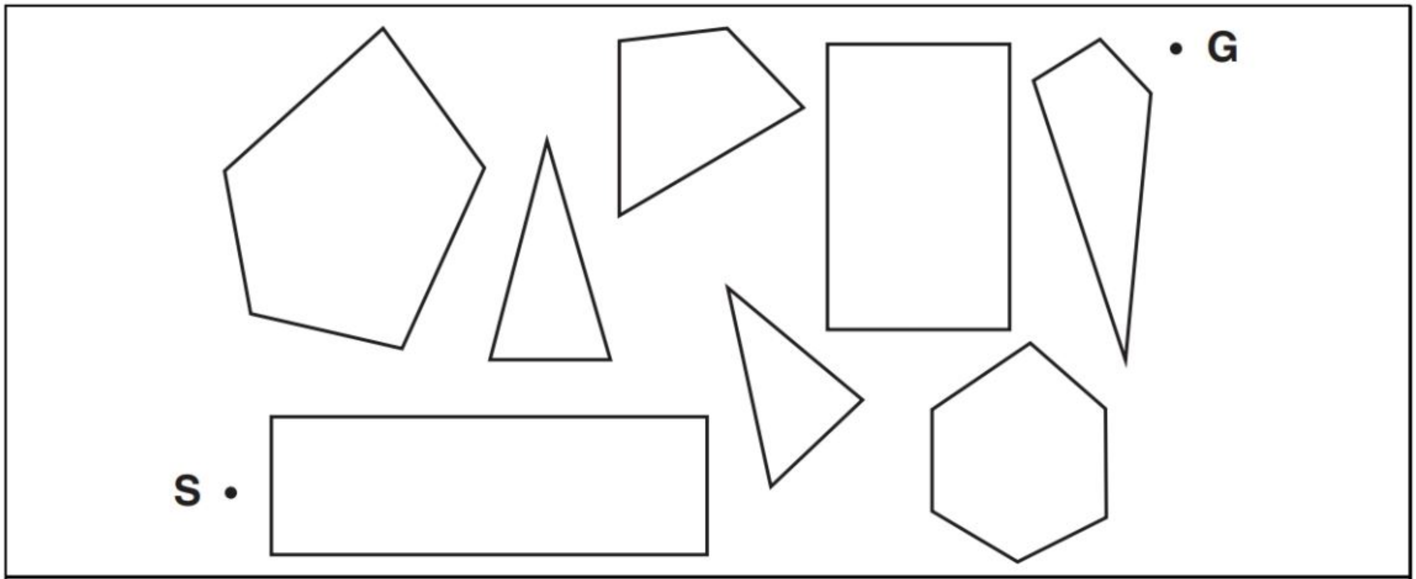


Figure 3.31 A scene with polygonal obstacles. S and G are the start and goal states.

Consider the problem of finding the shortest path (e.g., using A* algorithm) between two points on a plane that has convex polygonal obstacles as shown in Figure 3.31. This is an idealization of the problem that a robot has to solve to navigate in a crowded environment. [50 points]

- Define the necessary functions to implement the search problem, including an ACTIONS function that takes a *vertex* as input and returns a set of vectors, each of which maps the current vertex to one of the vertices that can be reached in a straight line. (Do not forget the neighbors on the same polygon.)
  - Use the straight-line distance for the heuristic function.
- Write a program (*language no bar*) to solve the shortest pathfinding problem for such an environment.
  - Input: an environment like the one shown in Fig. 3.31 in your textbook, a start, and a goal location.
  - Output: the shortest path that the robot has to follow to reach from the start to the goal (Shown visually on screen).
    - the screenshots of the output should be added to the PDF.