What would conceptually happen to Dijkstra's search if we replaced the priority queue with a queue (like FIFO) where it expanded cells in the order we discovered them ?

- ○ A) The algorithm would become faster but might miss the optimal path.
- ○ C) The algorithm would become faster but might never find a path.
- ○ D) The algorithm might fail on weighted cells map.
- ○ E) None of the above is correct
- ○ F) I don't know

Which of the following statements about Dijkstra's algorithm is correct?

- ○ A) The algorithm does not guarantee finding the shortest path.
- ○ B) The algorithm explores all cells simultaneously, creating redundant paths.
- ○ C) Some cells may have the same distance values, allowing multiple equivalent paths to the goal.
- ○ D) None of the above is correct
- ○ E) I don't know

What happens to A* when the heuristic function contributes almost nothing (for example, always 0 or nearly 0)?

- ○ A) It expands more nodes and might never find a path
- ○ B) It becomes incomplete and might never find a path.
- ○ C) It becomes greedy and may find a non-optimal path.
- ○ D) It behaves like Dijkstra's algorithm.
- ○ E) I don't know

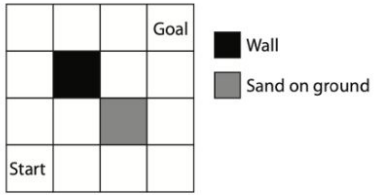
Which of the following statements about A* algorithm is correct?

- A) It has a better worst-case time than Dijkstra.
- B) It employs a heuristic -- which must be consistent with the task for solution optimality to be guaranteed.
- C) It employs a heuristic -- which counts the number of obstacles between the current node and the goal.
- D) It employs a heuristic -- which counts the number of steps between the current node and the start.
- E) None of the above is correct
- F) I don't know

In the context of the A* algorithm, what is an admissible heuristic function h(n)?

- A) A function that overestimates or equals the true cost to reach the goal from node.
- B) A function that underestimates or equals the true cost to reach the goal from node.
- C) A function that exactly calculates the true cost to reach the goal from node.
- D) A function that has no impact on the algorithm's performance.
- E) I don't know

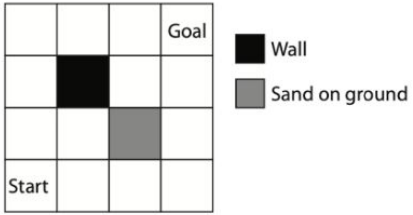
6. We apply Dijkstra's algorithm to the simple situation below to find the optimal path from start to goal. The wall cannot be passed, and the sand can be passed with some extra energy and time (cost of 2).



6.1) With a cost of 1 to move to any of the 4 neighbors (Manhattan) (excluding sand and walls), the number of optimal path(s) is/are:

- ◯ None
- ◯ 1 path
- ◯ 2 paths
- ◯ 3 paths
- ◯ 4 paths
- ◯ 5 paths
- ◯ I don't know

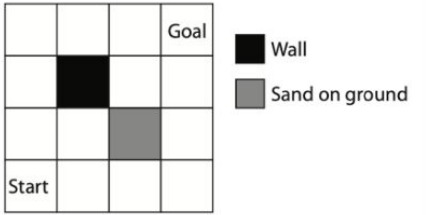
6. We apply Dijkstra's algorithm to the simple situation below to find the optimal path from start to goal. The wall cannot be passed, and the sand can be passed with some extra energy and time (cost of 2).



6.2) With a cost of 1 to move to any of the 8 neighbors (Euclidean) (excluding sand and walls), the number of optimal path(s) is/are:

- ○ None
- ○ 1 path
- ○ 2 paths
- ○ 3 paths
- ○ 4 paths
- ○ 5 paths
- ○ I don't know

6. We apply Dijkstra's algorithm to the simple situation below to find the optimal path from start to goal. The wall cannot be passed, and the sand can be passed with some extra energy and time (cost of 2).



6.3) If you implemented A* and Dijkstra on the situation with 4 neighbors (Manhattan), what would you notice?

- A) A* gives the same result as Dijkstra for both the number of cells explored and the path.
- B) A* gives the same result as Dijkstra only for the path.
- C) A* gives the same result as Dijkstra only for the number of cells explored.
- D) None of the above answers is correct.
- E) I don't know