Best-First

Inputs	search problem $\langle X, S, G, \mathcal{T}, c\rangle$
	evaluation function e
Output	(path to) goal node
Initialize	$O = S$ is the list of open nodes

```
while (O is not empty) do
```

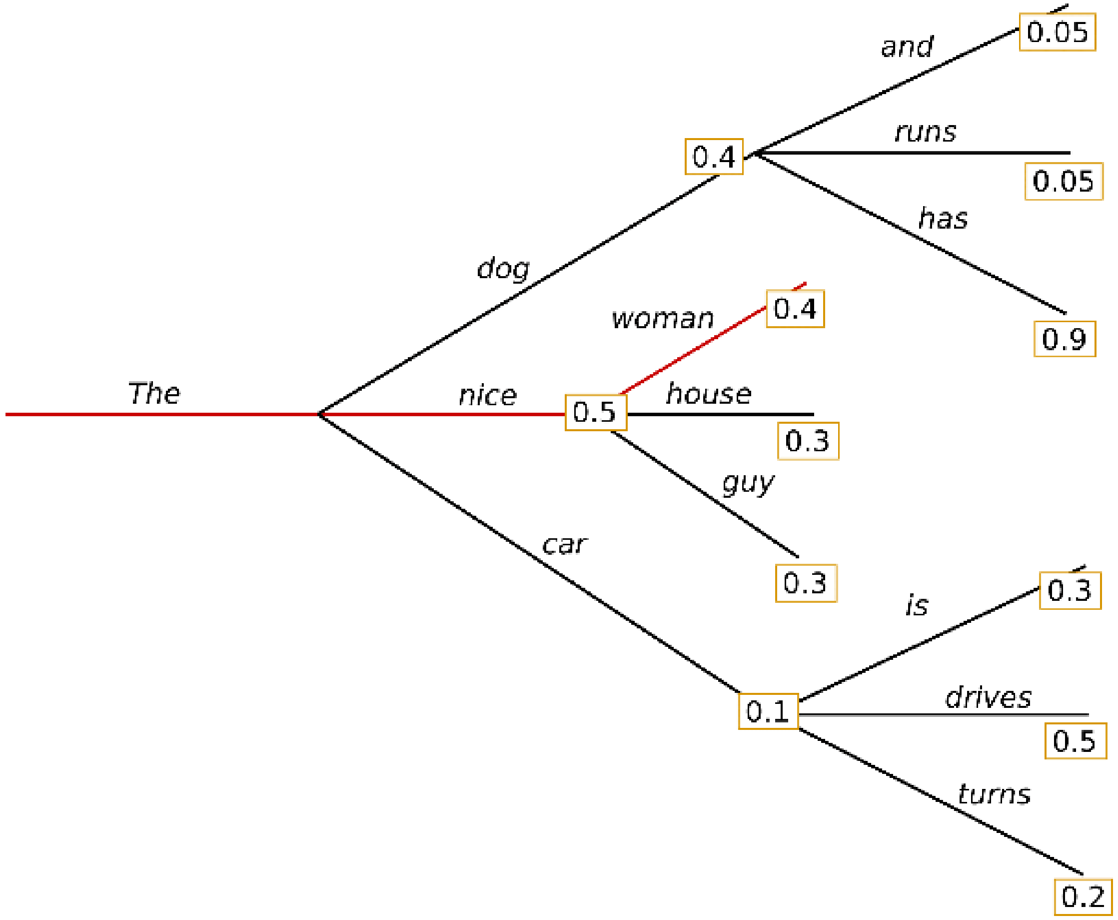
1. delete node $n \in O$ $s.t.$ $e(n)$ is minimal
2. if $n \in G$, return (path to) n
3. for all $m \in \mathcal{T}(n)$
   (a) compute $e(m)$
   (b) insert m into O with priority $e(m)$

```
fail
```

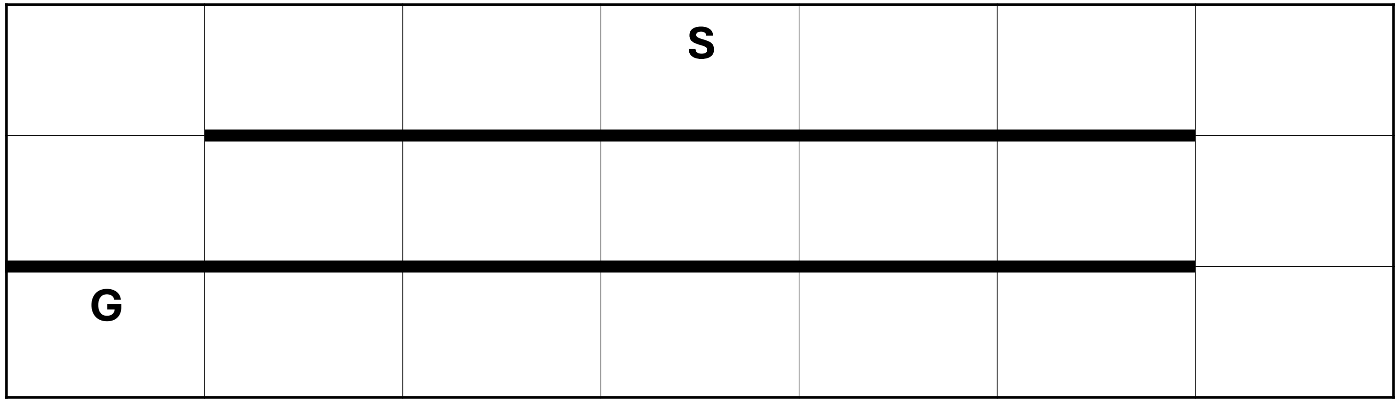
Language Modelling

Standard Language Modelling selects "top-p" at each node (i.e., greedily selects the most likely next word)

Can we find a more likely continuation?



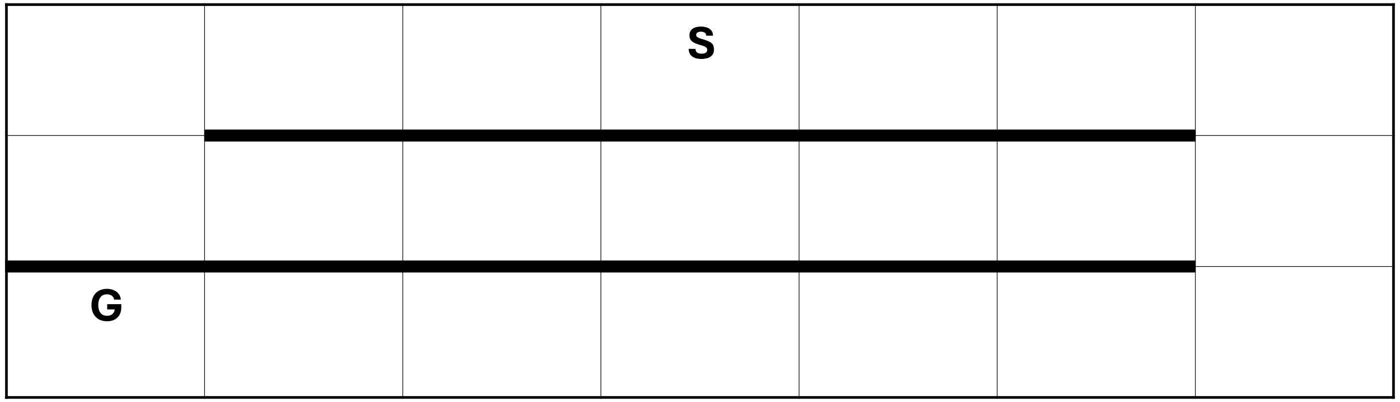
What is g(n) for every node?
What order will best-g explore the maze?



What is g(n) for every node?
What order will best-g explore the maze?

3	2	1	**S**	1	2	3
4	5	6	7	6	5	4
G	10	9	8	7	6	5

If we use the Manhattan distance as h,
What is h(n) for each node n?
What order will best-h explore the maze?

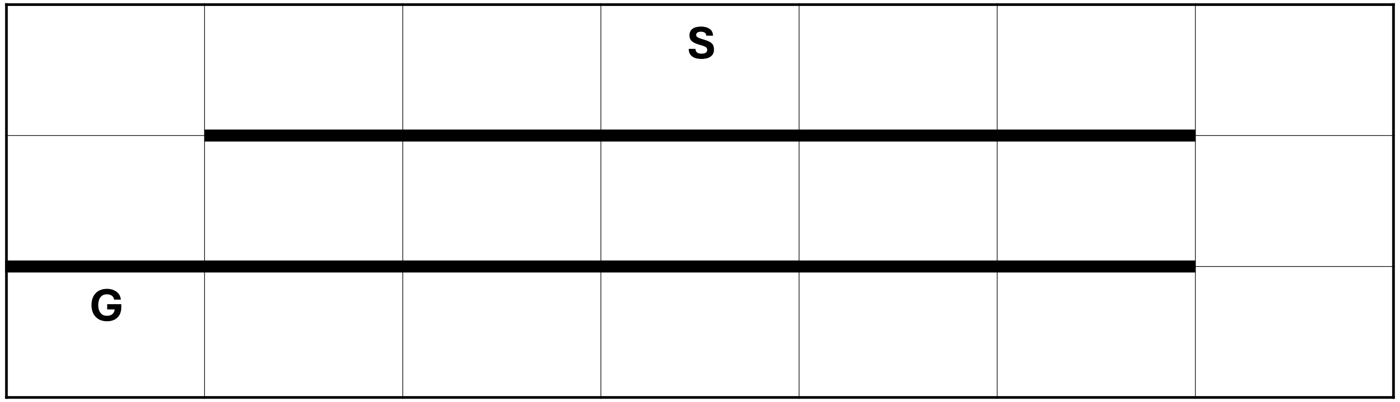


If we use the Manhattan distance as h,
What is h(n) for each node n?
What order will best-h explore the maze?

2	3	4	**S**	6	7	8
1	2	3	4	5	6	7
G	1	2	3	4	5	6

How will A* search, using Manhattan distance as a heuristic explore the maze?

What is f(n) at each node n?



How will A* search, using Manhattan distance as a heuristic explore the maze?

What is f(n) at each node n?

g(n)

3	2	1	**S**	1	2	3
4	5	6	7	6	5	4
G	10	9	8	7	6	5

h(n)

2	3	4	**S**	6	7	8
1	2	3	4	5	6	7
G	1	2	3	4	5	6

f(n)

5	5	5	**S**	7	9	11
5	7	9	11	11	11	11
G	11	11	11	11	11	11

How will A* search, using Manhattan distance as a heuristic explore the maze?

What is f(n) at each node n?

g(n)

3	2	1	**S**	1	2	3
4	5	6	7	6	5	4
G	10	9	8	7	6	5

h(n)

2	3	4	**S**	6	7	8
1	2	3	4	5	6	7
G	1	2	3	4	5	6

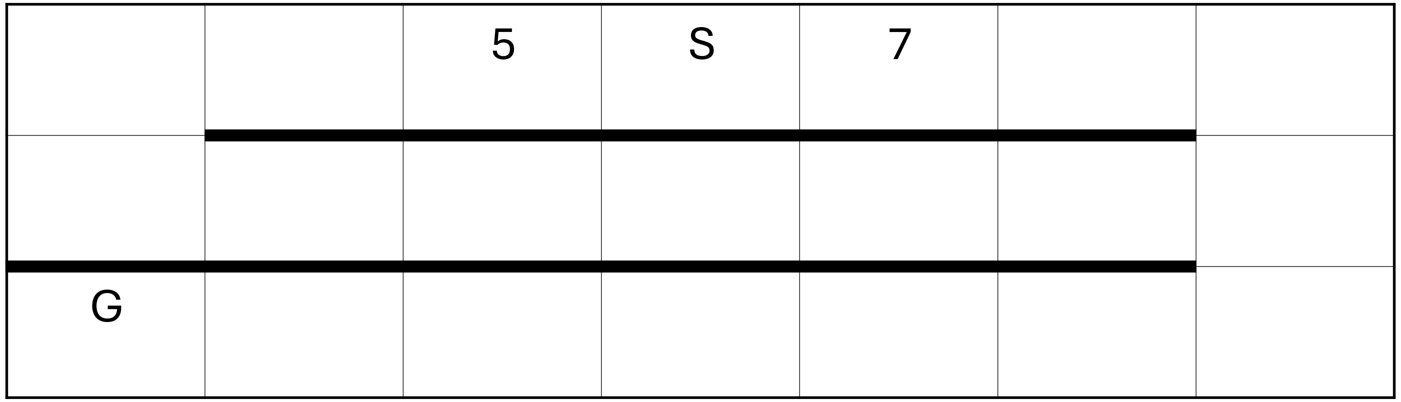
What happened here?

f(n)

5	5	5	**S**	7	9	11
5	7	9	11	11	11	11
G	11	11	11	11	11	11

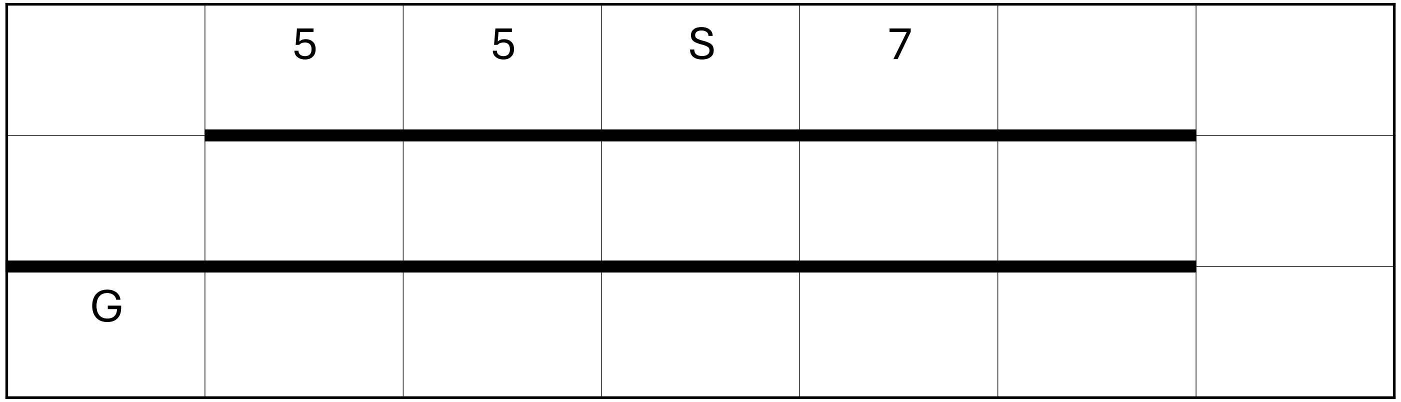
How will A* search, using Manhattan distance as a heuristic explore the maze?

What is f(n) at each node n?



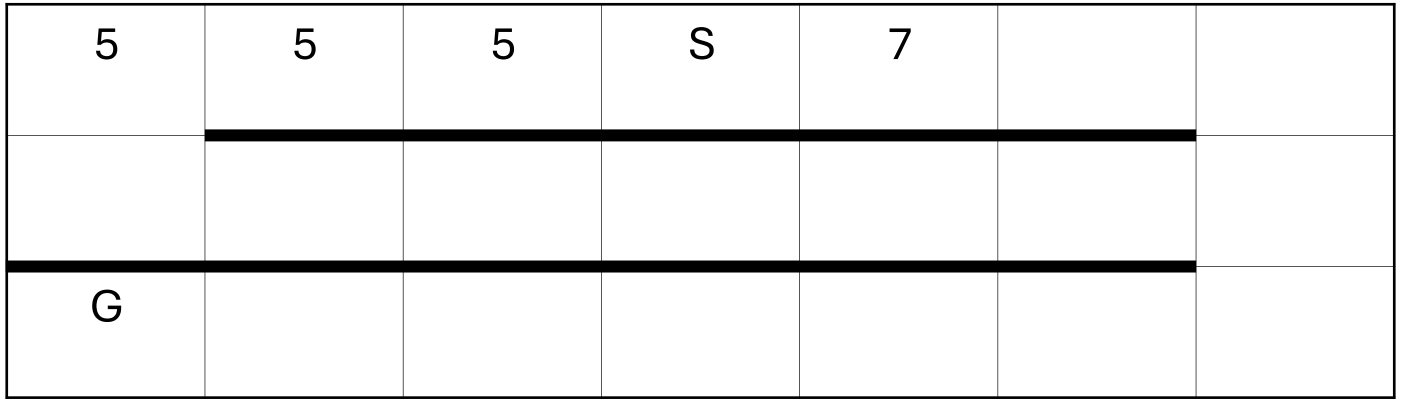
How will A* search, using Manhattan distance as a heuristic explore the maze?

What is f(n) at each node n?



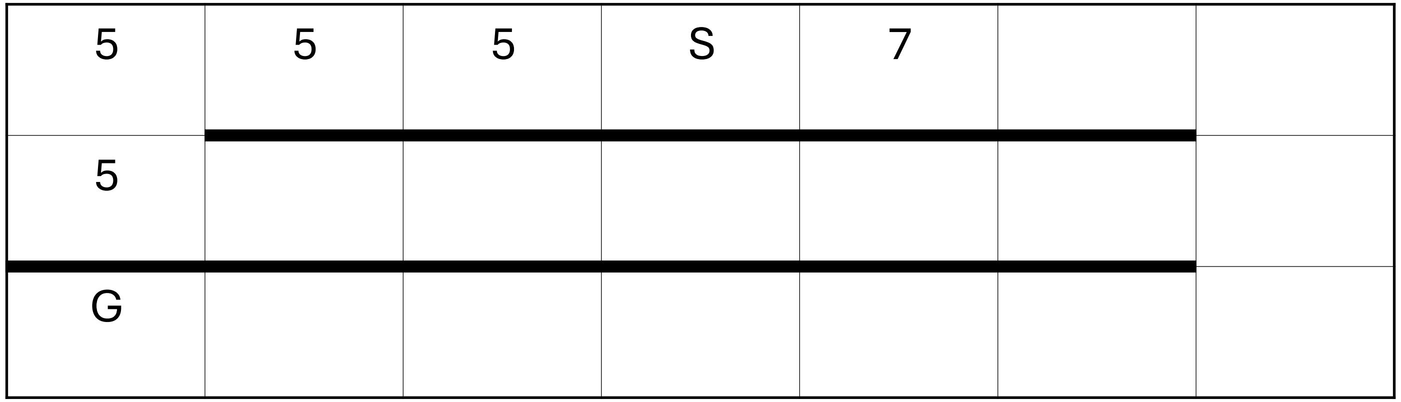
How will A* search, using Manhattan distance as a heuristic explore the maze?

What is f(n) at each node n?



How will A* search, using Manhattan distance as a heuristic explore the maze?

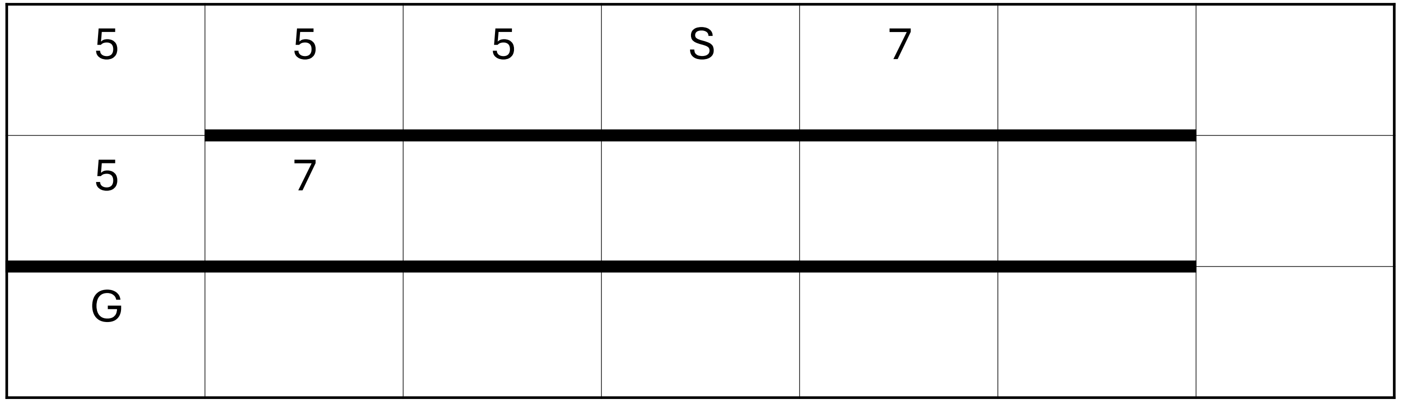
What is f(n) at each node n?



How will A* search, using Manhattan distance as a heuristic explore the maze?

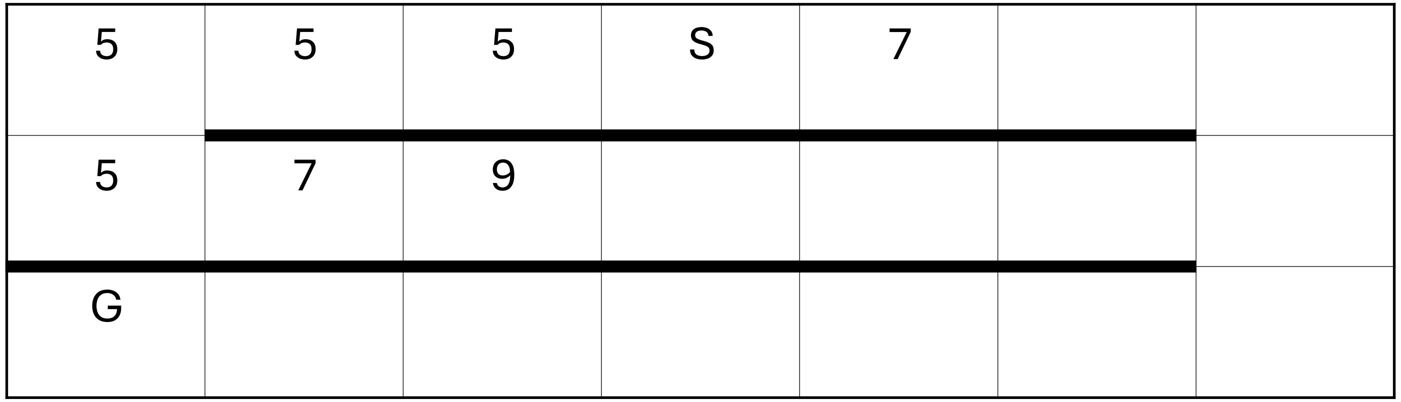
What is f(n) at each node n?

Tie, either node can be expanded first



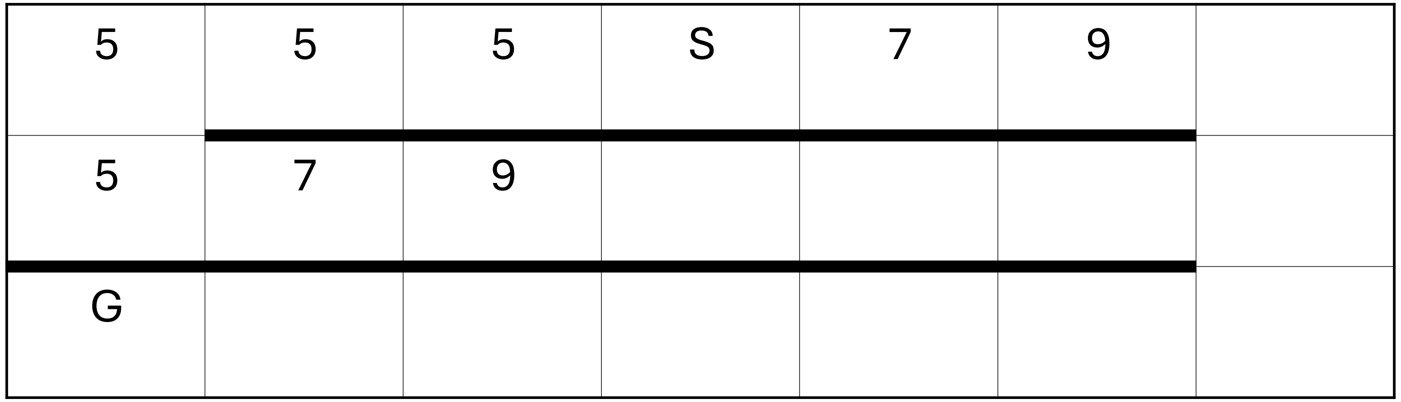
How will A* search, using Manhattan distance as a heuristic explore the maze?

What is f(n) at each node n?



How will A* search, using Manhattan distance as a heuristic explore the maze?

What is f(n) at each node n?



How will A* search, using Manhattan distance as a heuristic explore the maze?

What is f(n) at each node n?

5	5	5	S	7	9	
5	7	9	11			
G						

How will A* search, using Manhattan distance as a heuristic explore the maze?

What is f(n) at each node n?

5	5	5	S	7	9	11
5	7	9	11			
G						

How will A* search, using Manhattan distance as a heuristic explore the maze?

What is f(n) at each node n?

5	5	5	S	7	9	11
5	7	9	11	13		
G						

How will A* search, using Manhattan distance as a heuristic explore the maze?

What is f(n) at each node n?

5	5	5	S	7	9	11
5	7	9	11	13		11
G						

How will A* search, using Manhattan distance as a heuristic explore the maze?

What is f(n) at each node n?

5	5	5	S	7	9	11
5	7	9	11	13	11	11
G						

How will A* search, using Manhattan distance as a heuristic explore the maze?

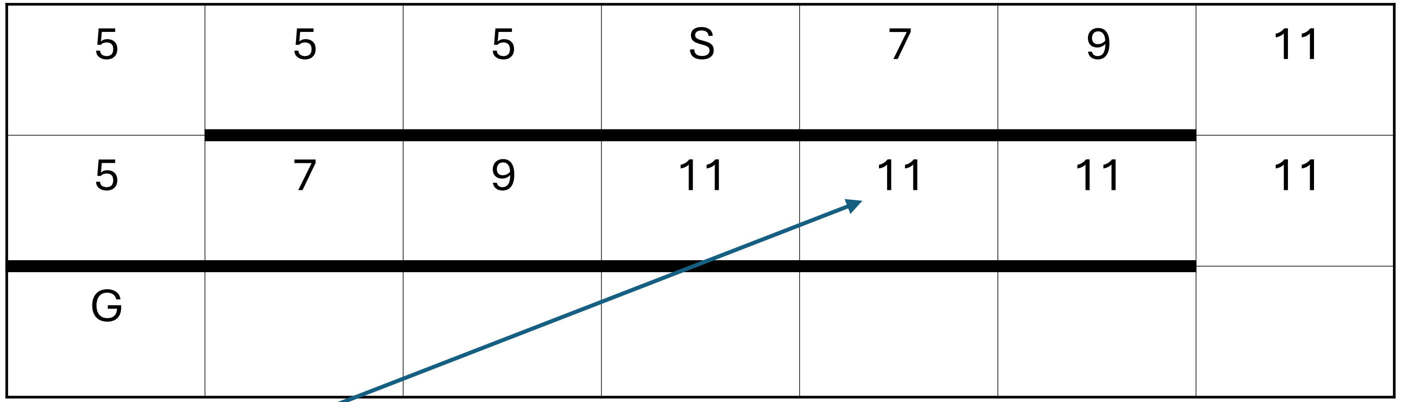
What is f(n) at each node n?

5	5	5	S	7	9	11
5	7	9	11	13	11	11
G						

Old g(n)=8, new g(n)=6

How will A* search, using Manhattan distance as a heuristic explore the maze?

What is f(n) at each node n?



Old g(n)=8, new g(n)=6

How will A* search, using Manhattan distance as a heuristic explore the maze?

What is f(n) at each node n?

- Each node needs to keep track of its predecessor to reconstruct the path.
- You must also keep track of g(n) for each node.
- If n is multiple times and g(n) is lower than the first time, you must update g(n) and its predecessor)

5	5	5	S	7	9	11
5	7	9	11	11	11	11
G	11	11	11	11	11	11