Maze Algorithms

**Algorithm Maze** \((maze, \text{start})\)
Find the goal in a maze using a stack (depth first search)

create an empty stack named \(S\);
push \(\text{start}\) onto \(S\);
while \(S\) is not empty do
  \(\text{current} \leftarrow \text{pop from } S\);
  if \(\text{current}\) is the goal then
    output “Success!”;
    \(S \leftarrow \text{empty stack};\) \{to end the loop\}
  else if \(\text{current}\) is not a wall and \(\text{current}\) is not marked as visited then
    mark \(\text{current}\) as visited;
    push onto \(S\) the point to the right of \(\text{current}\);
    push onto \(S\) the point to the left of \(\text{current}\);
    push onto \(S\) the point above \(\text{current}\);
    push onto \(S\) the point below \(\text{current}\);
  end if
end while

**Algorithm Maze** \((maze, \text{start})\)
Find the goal in a maze using a queue (breadth first search)

create an empty queue named \(Q\);
enqueue \(\text{start}\) in \(Q\);
while \(Q\) is not empty do
  \(\text{current} \leftarrow \text{dequeue from } Q\);
  if \(\text{current}\) is the goal then
    output “Success!”;
    \(Q \leftarrow \text{empty queue};\) \{to end the loop\}
  else if \(\text{current}\) is not a wall and \(\text{current}\) is not marked as visited then
    mark \(\text{current}\) as visited;
    enqueue in \(Q\) the point to the right of \(\text{current}\);
    enqueue in \(Q\) the point to the left of \(\text{current}\);
    enqueue in \(Q\) the point above \(\text{current}\);
    enqueue in \(Q\) the point below \(\text{current}\);
  end if
end while