SEARCH

Joe Osborn

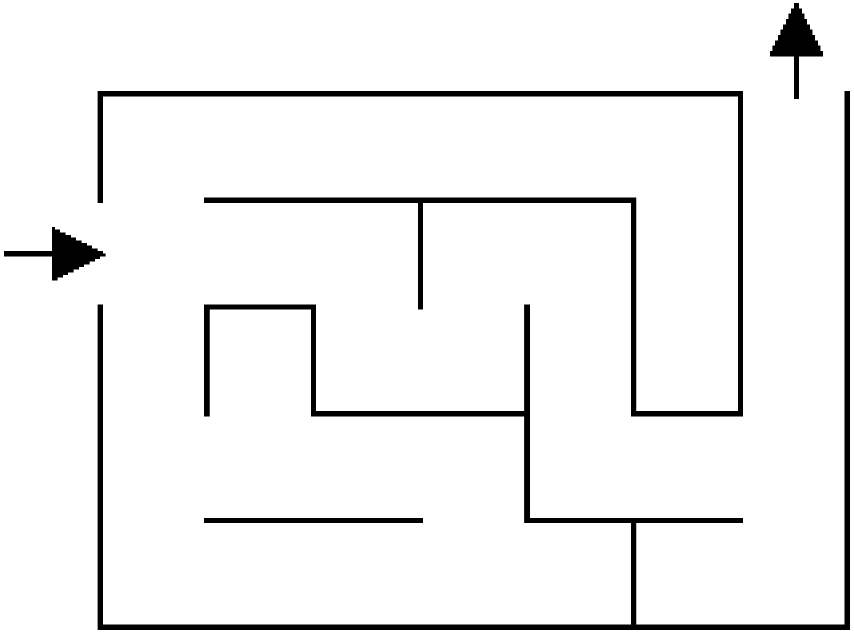
CS51A – Spring 2020

What is AI?

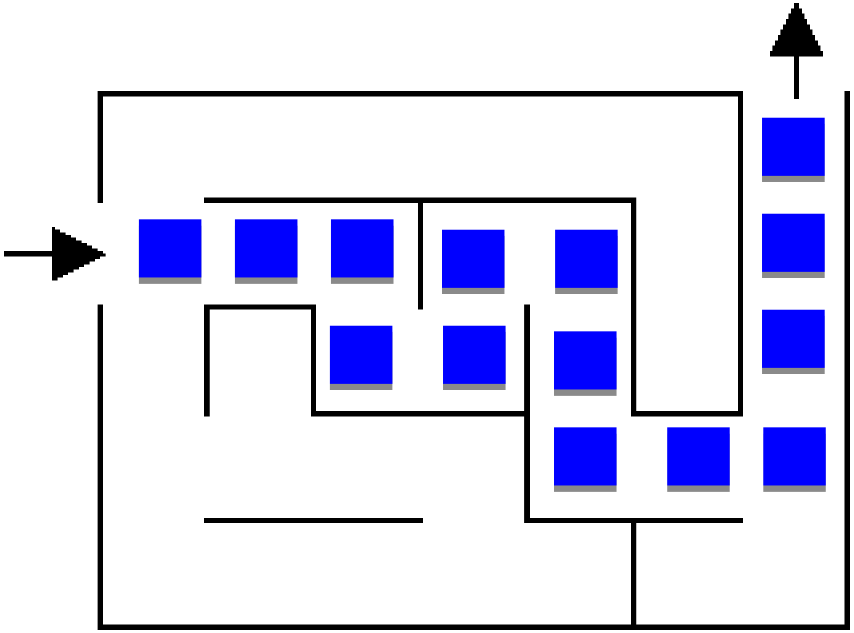
Think like a human Cognitive Modeling	**Think rationally** Logic-based Systems
Act like a human Turing Test	**Act rationally** Rational Agents

Next couple of weeks

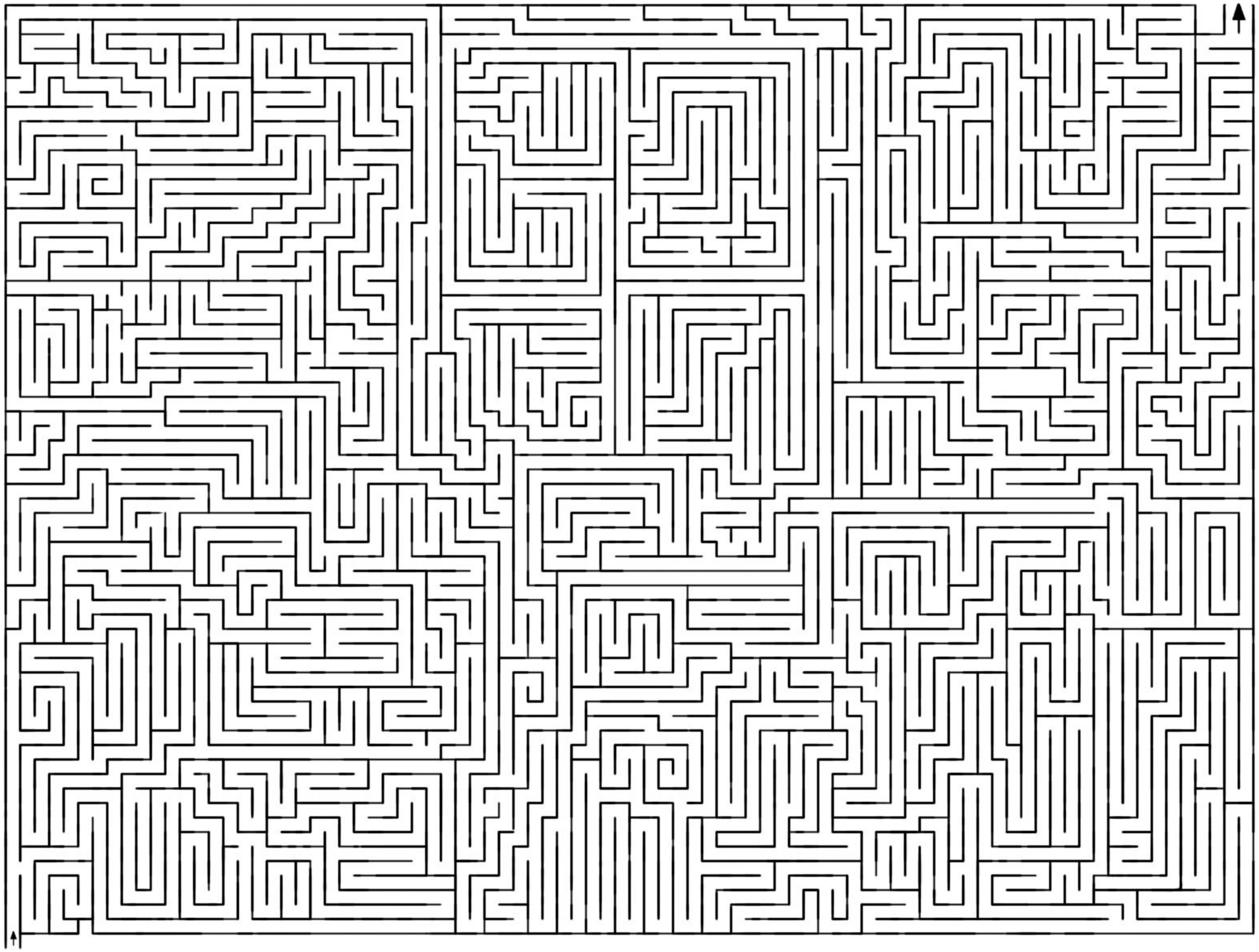
Solve the maze!



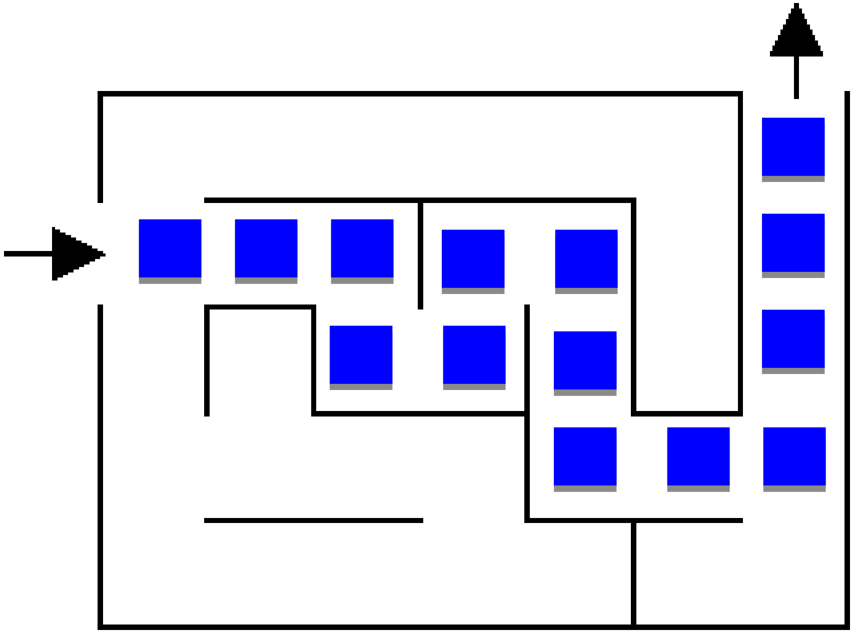
Solve the maze!



Solve the maze!

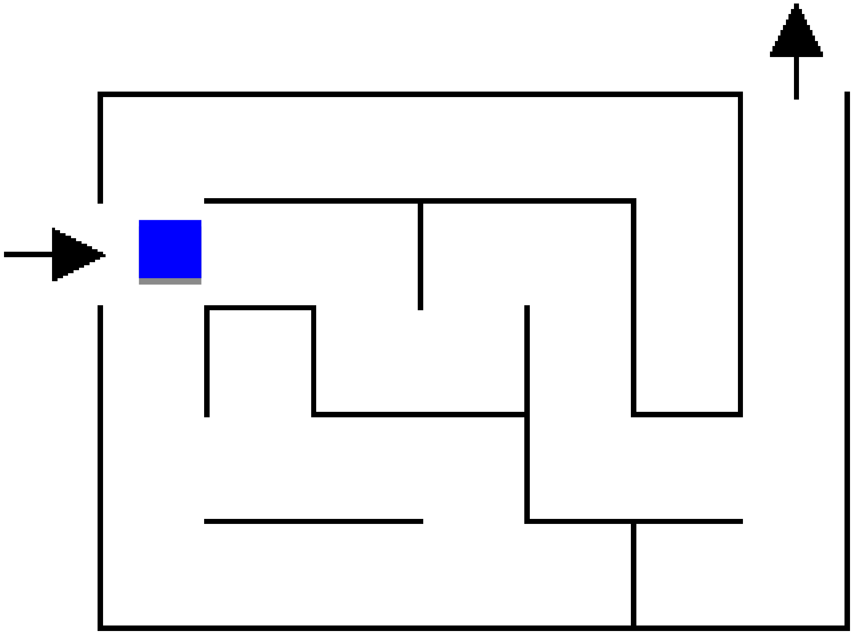


Solve the maze!



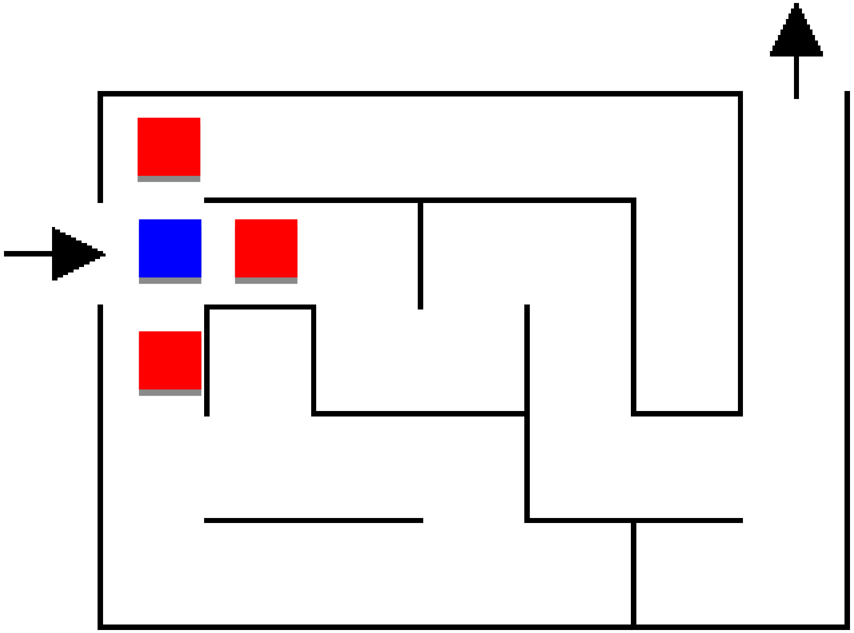
How did you figure it out?

One approach



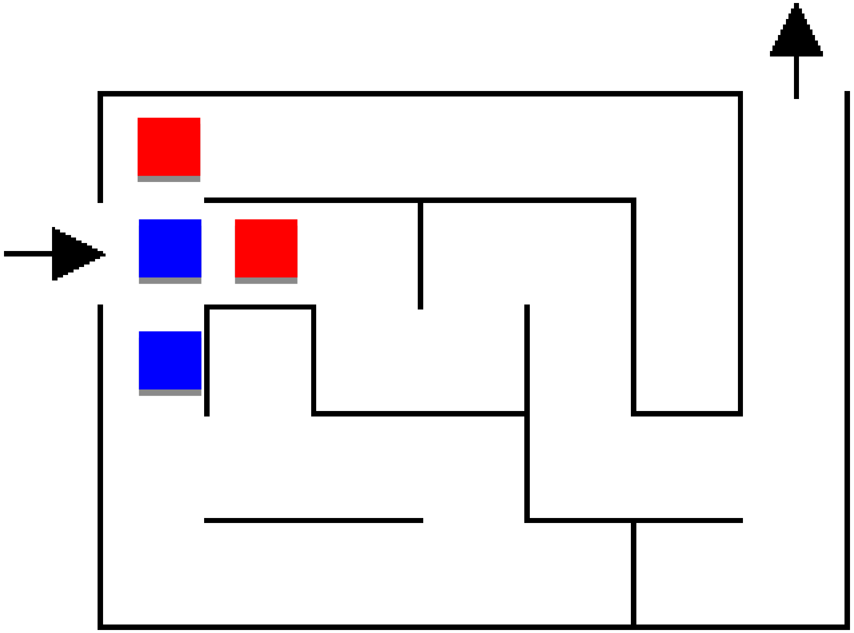
What now?

One approach



Three choices

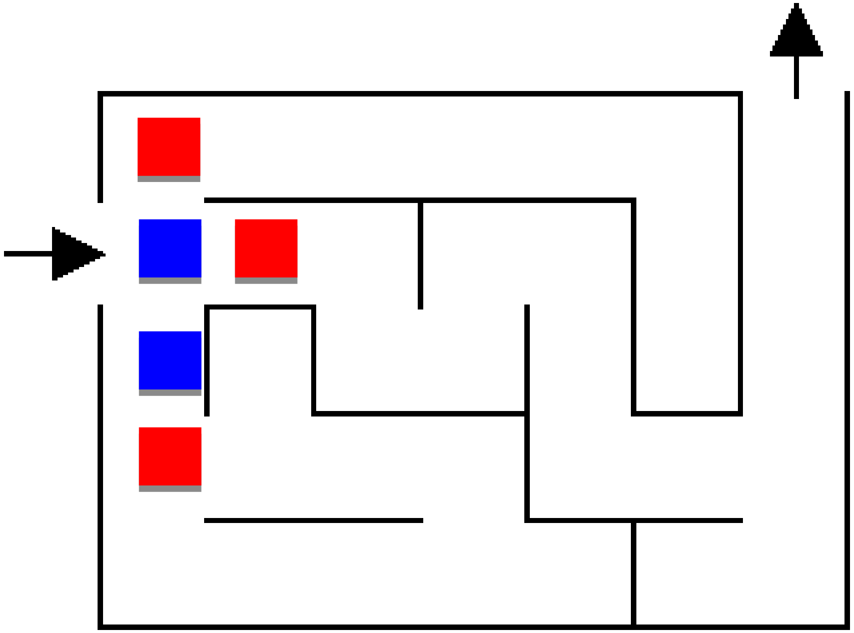
One approach



Pick one!

What now?

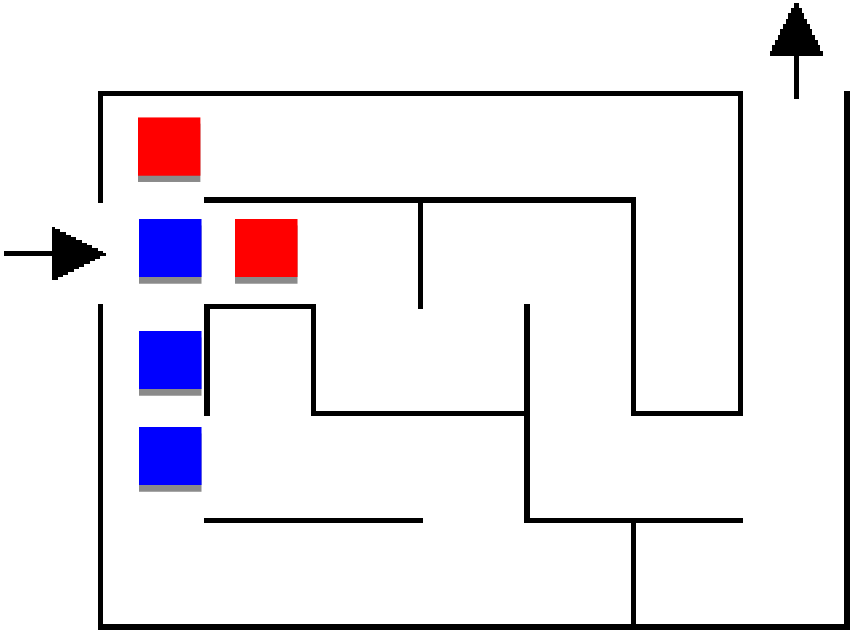
One approach



Still three options!

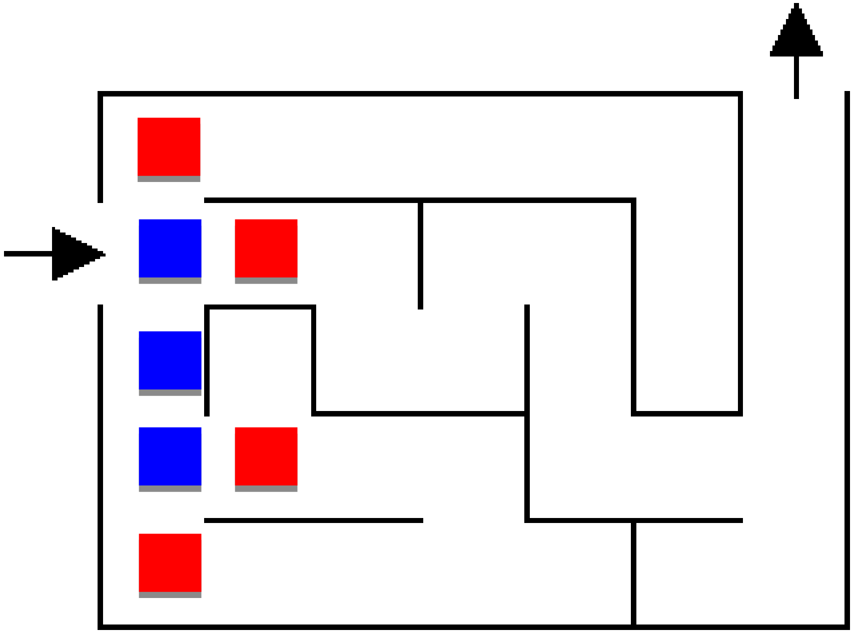
Which would you explore/pick?

One approach



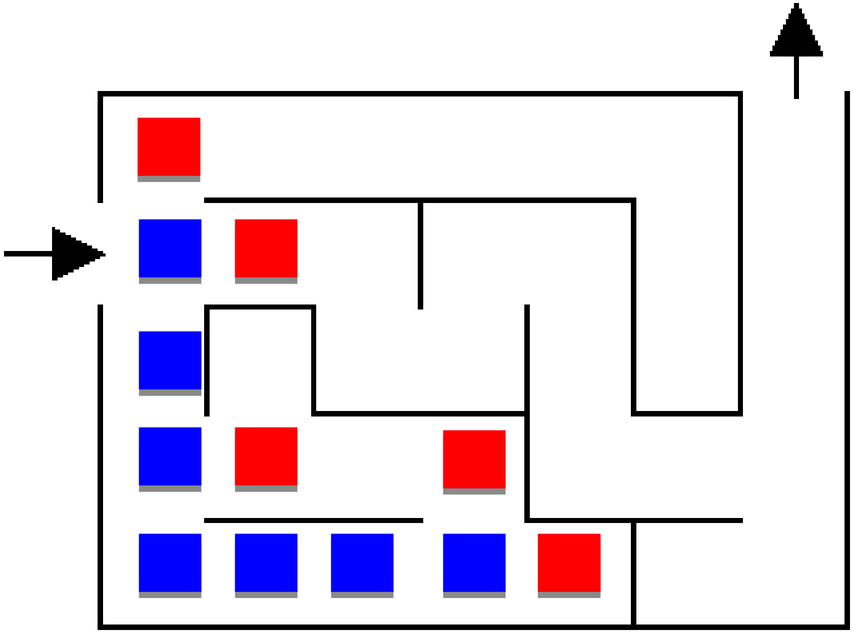
Most people go down a single path until they realize that it's wrong

One approach



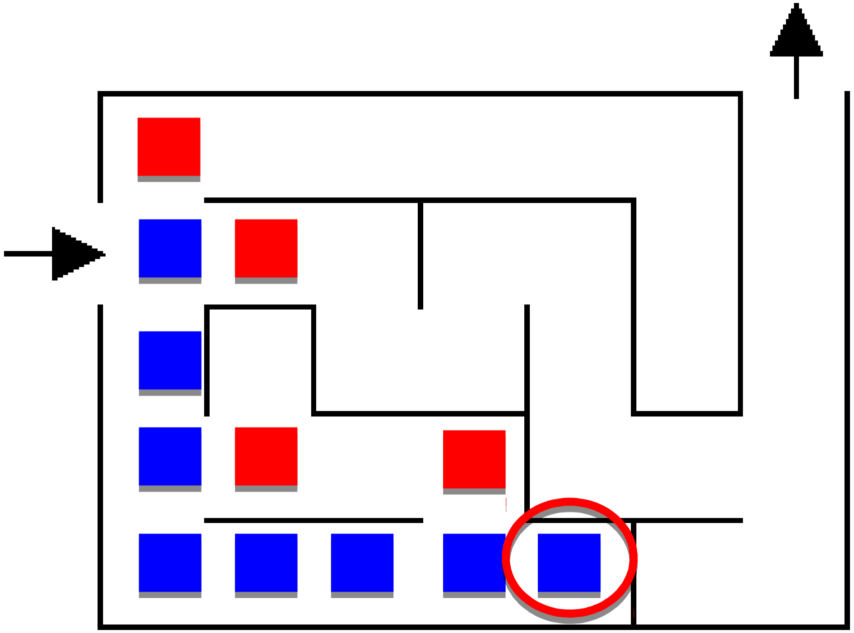
Keep exploring

One approach



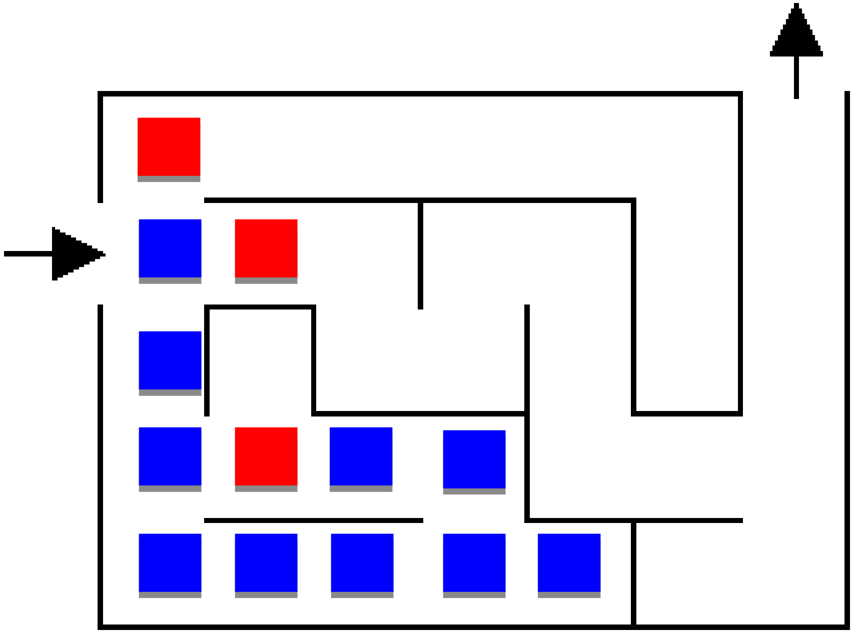
Keep exploring

One approach



What now?

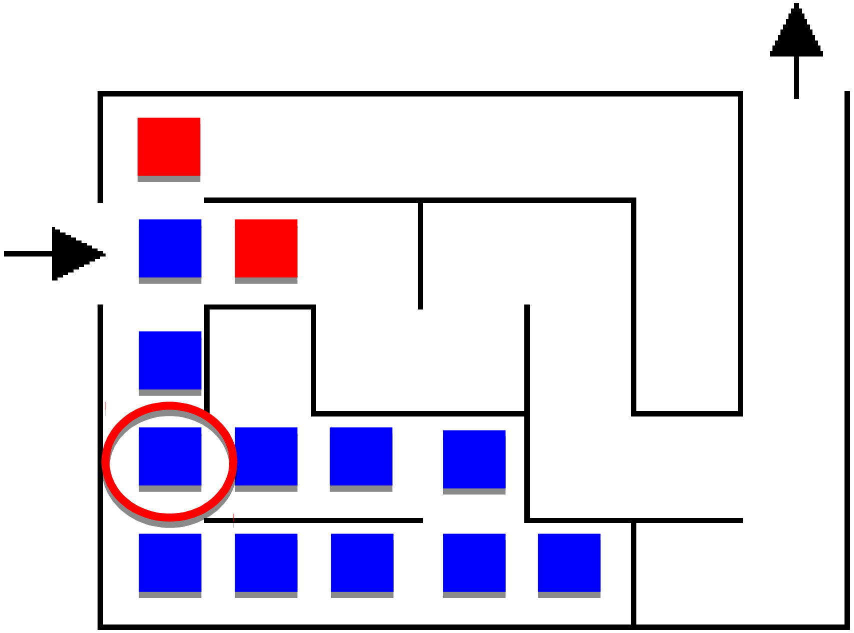
One approach



Are we stuck?

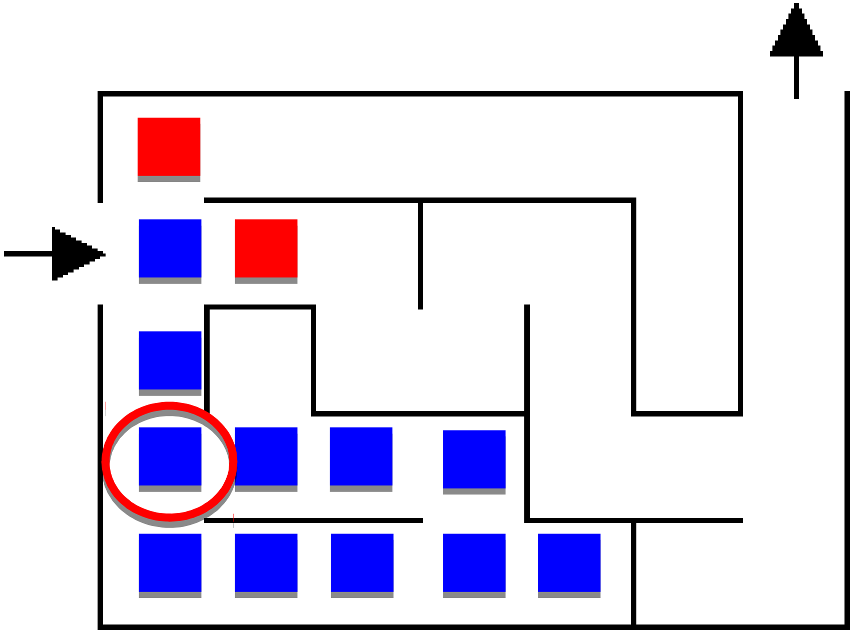
No. Red positions are just possible options we haven’t explored

One approach



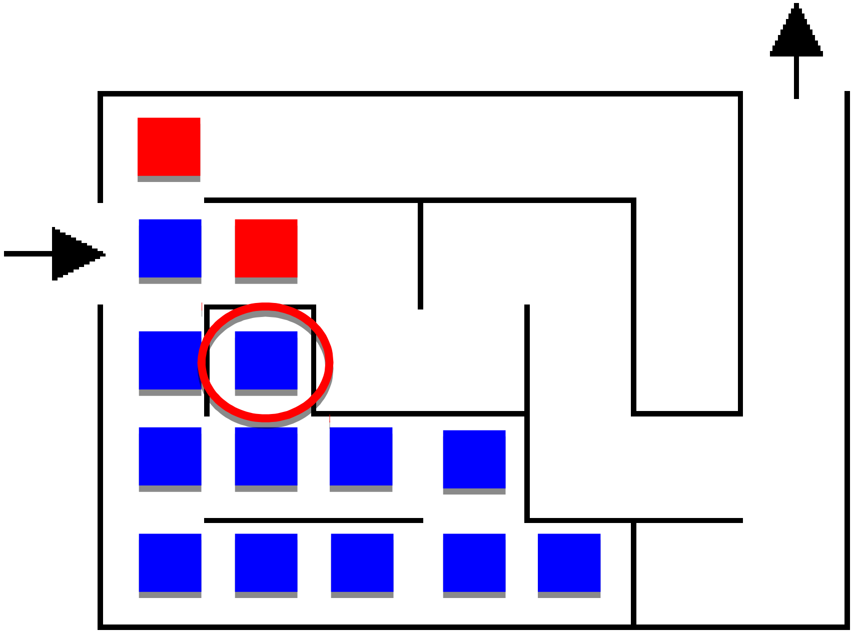
How do we know not to go left?

One approach



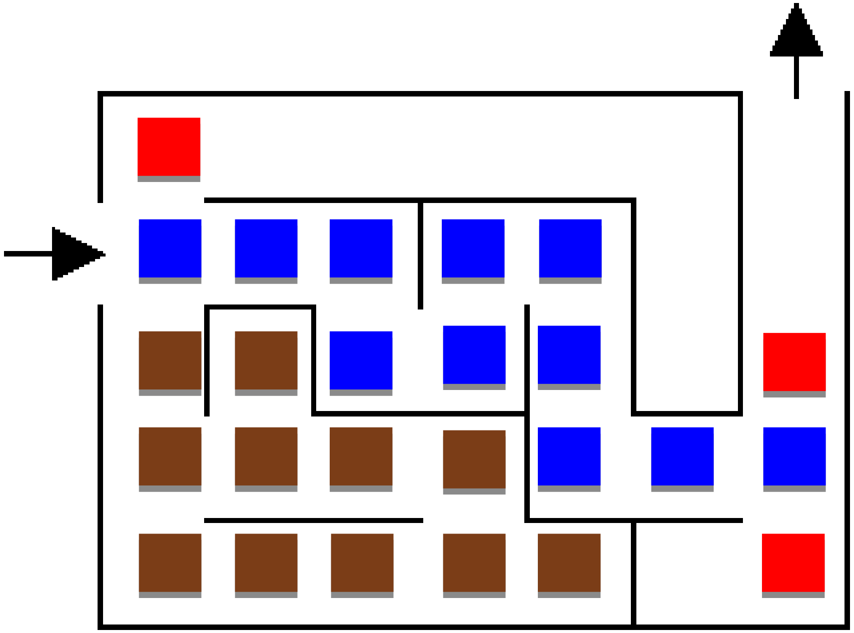
Have to be careful and keep track of where we’ve been if we can loop

One approach



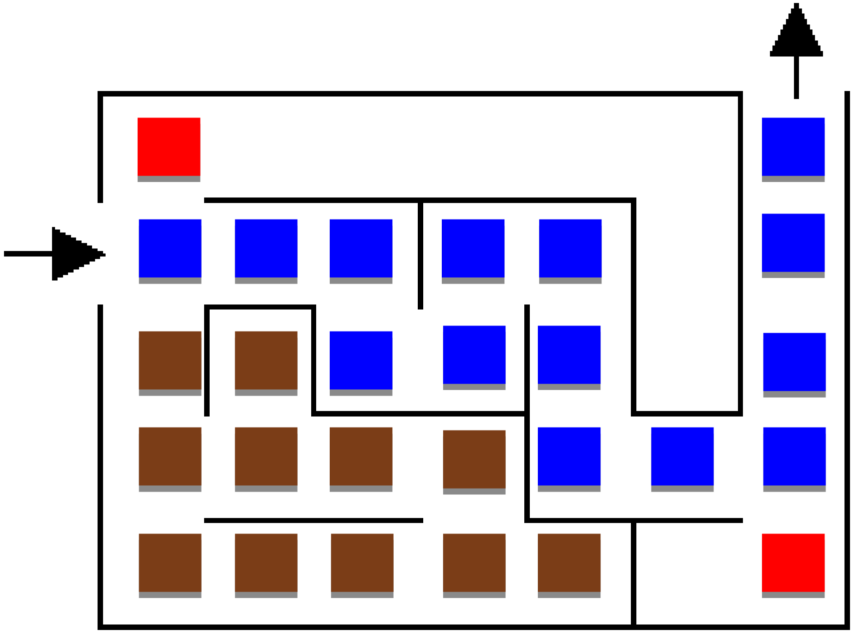
Now what?

One approach

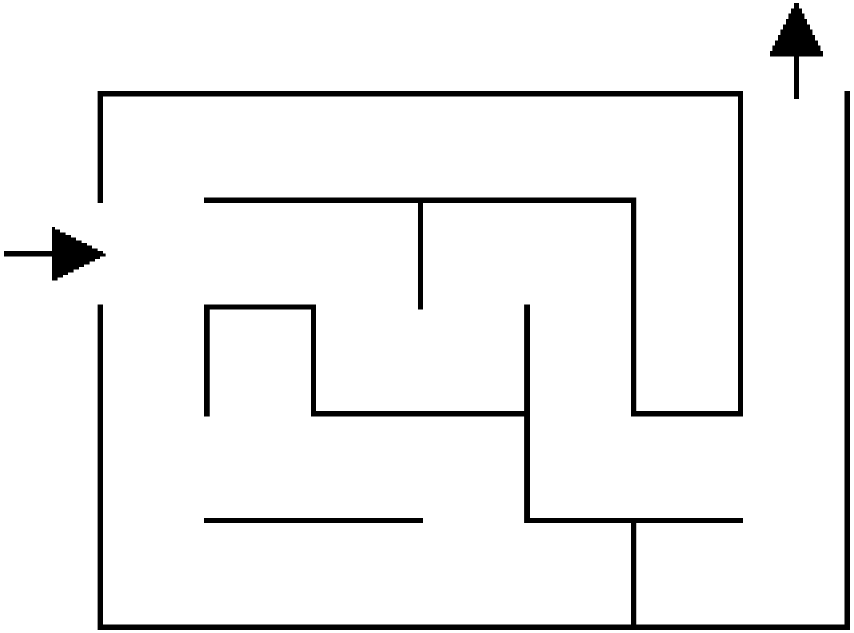


Now
what?

One approach



Search problems



What information do we need to figure out a solution?

Search problems

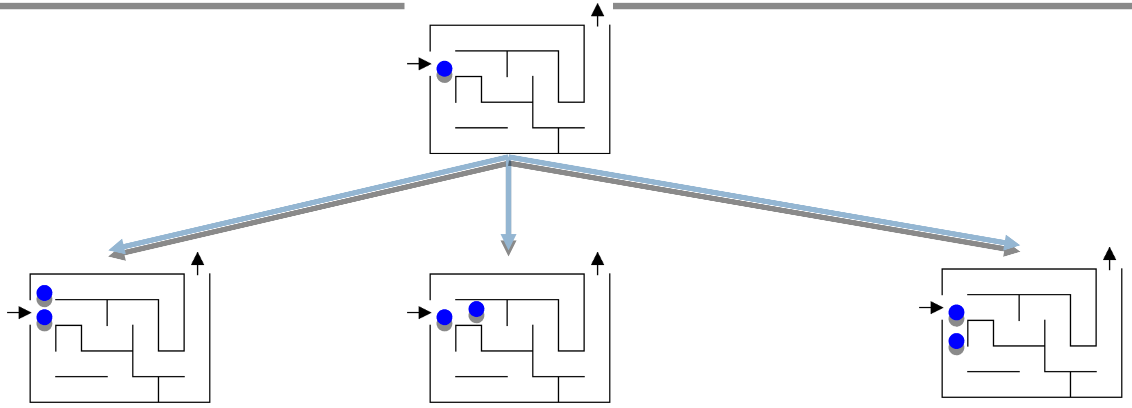
Where to start

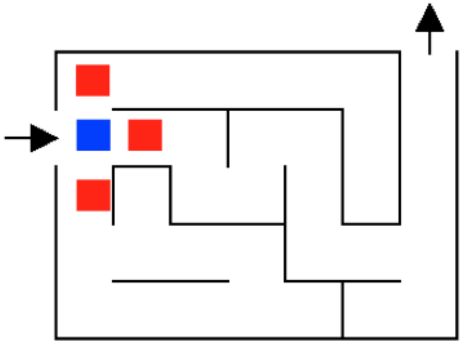
Where to finish (goal)

What the "world" (in this case a maze) looks like

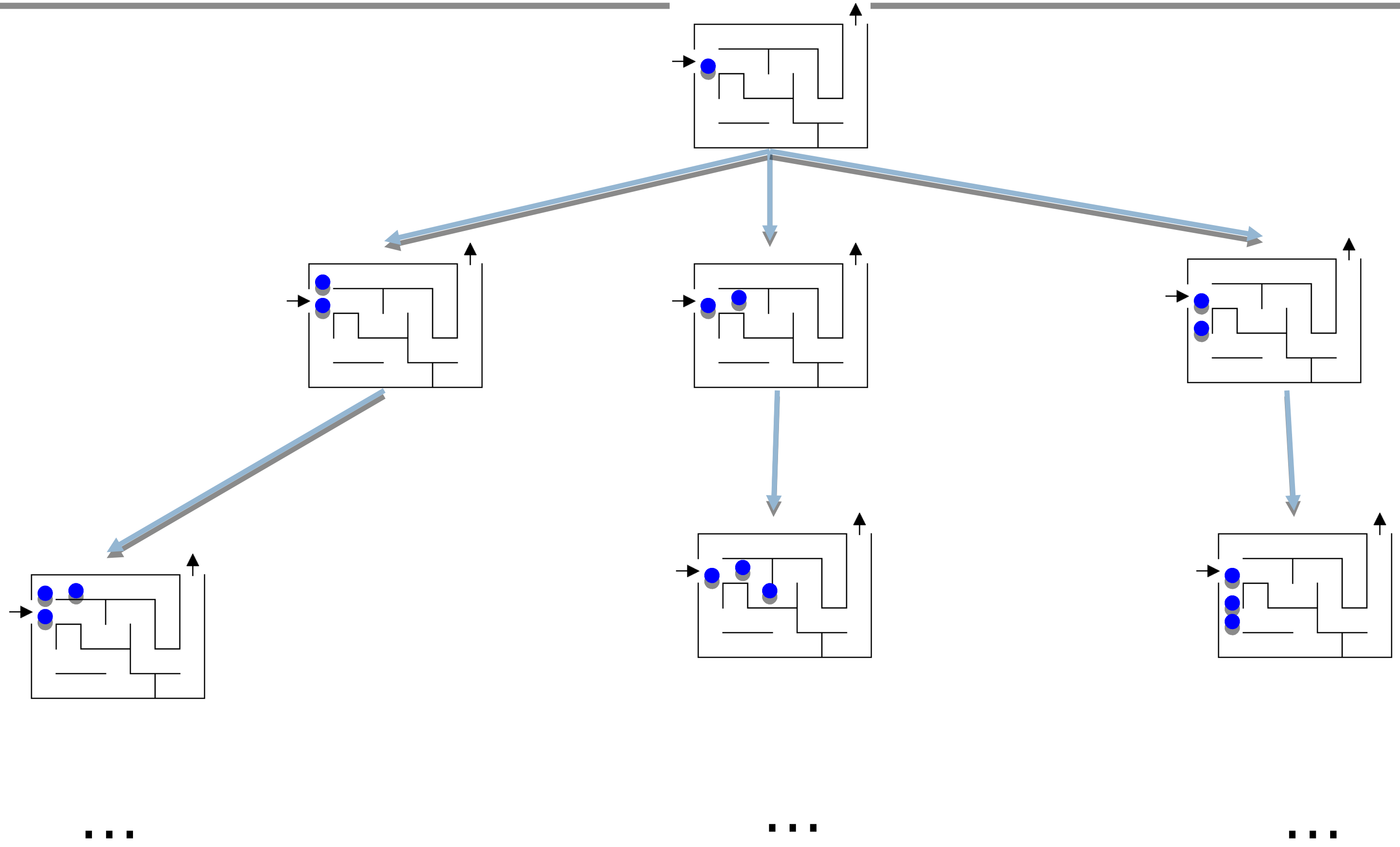
- We'll define the world as a collection of discrete states
- States are connected if we can get from one state to another by taking a particular action
- This is called the "state space"

State space example

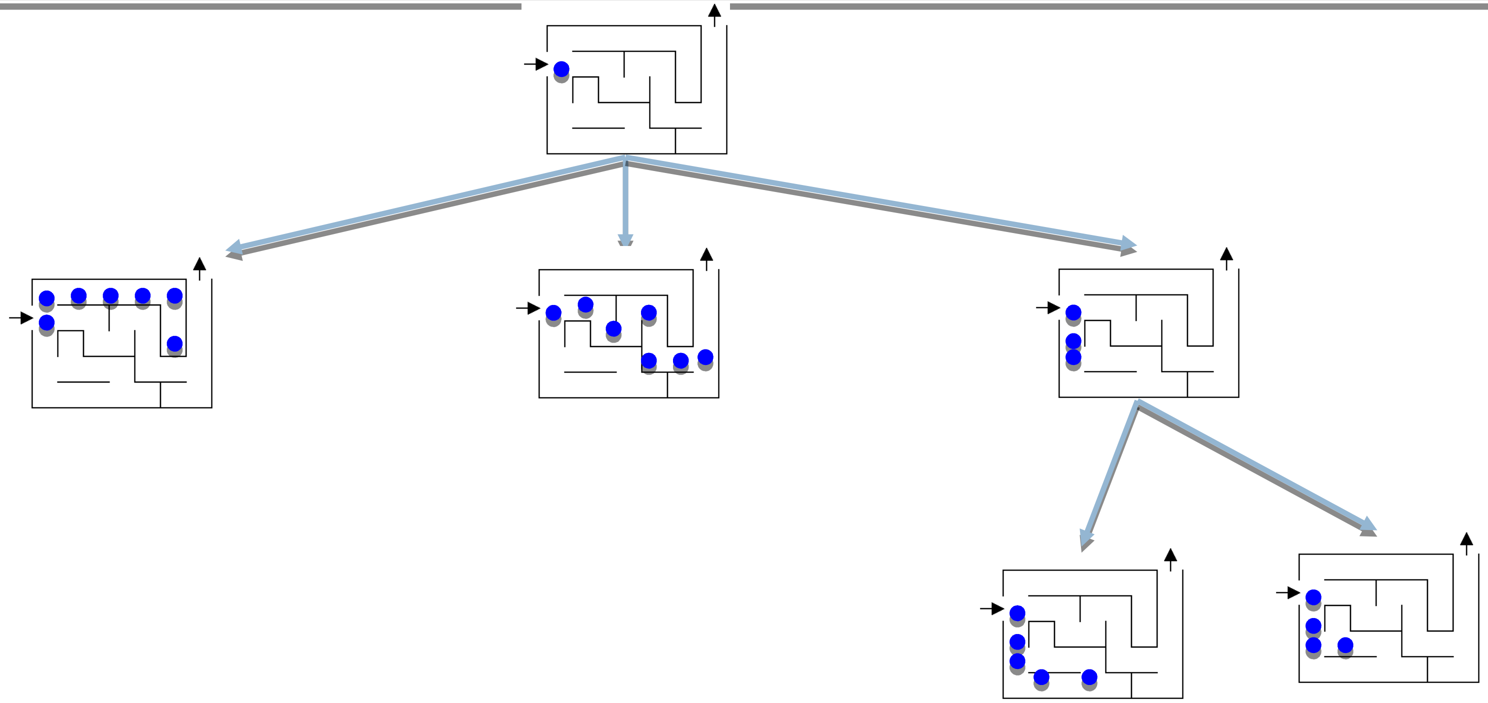




State space example



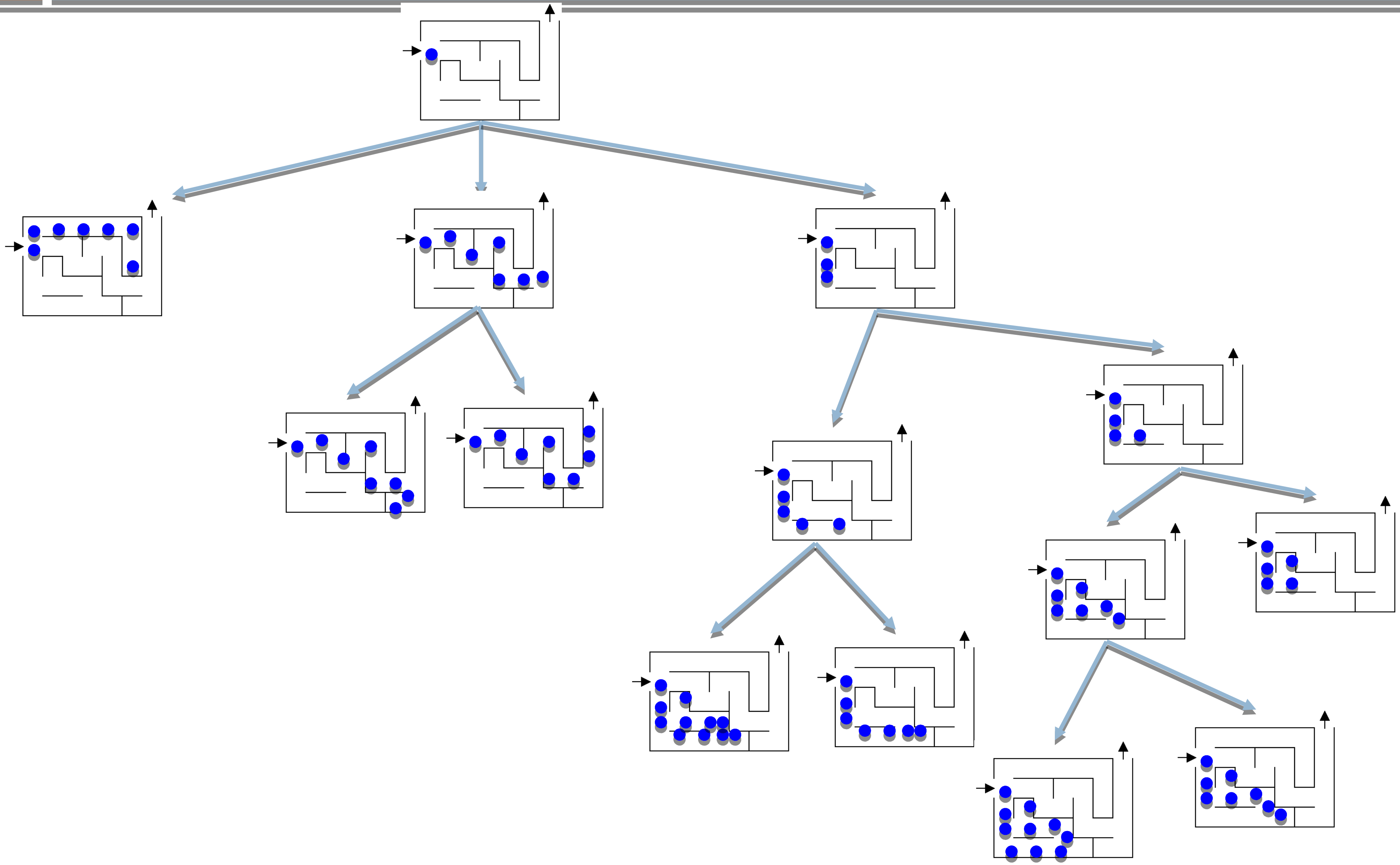
State space example



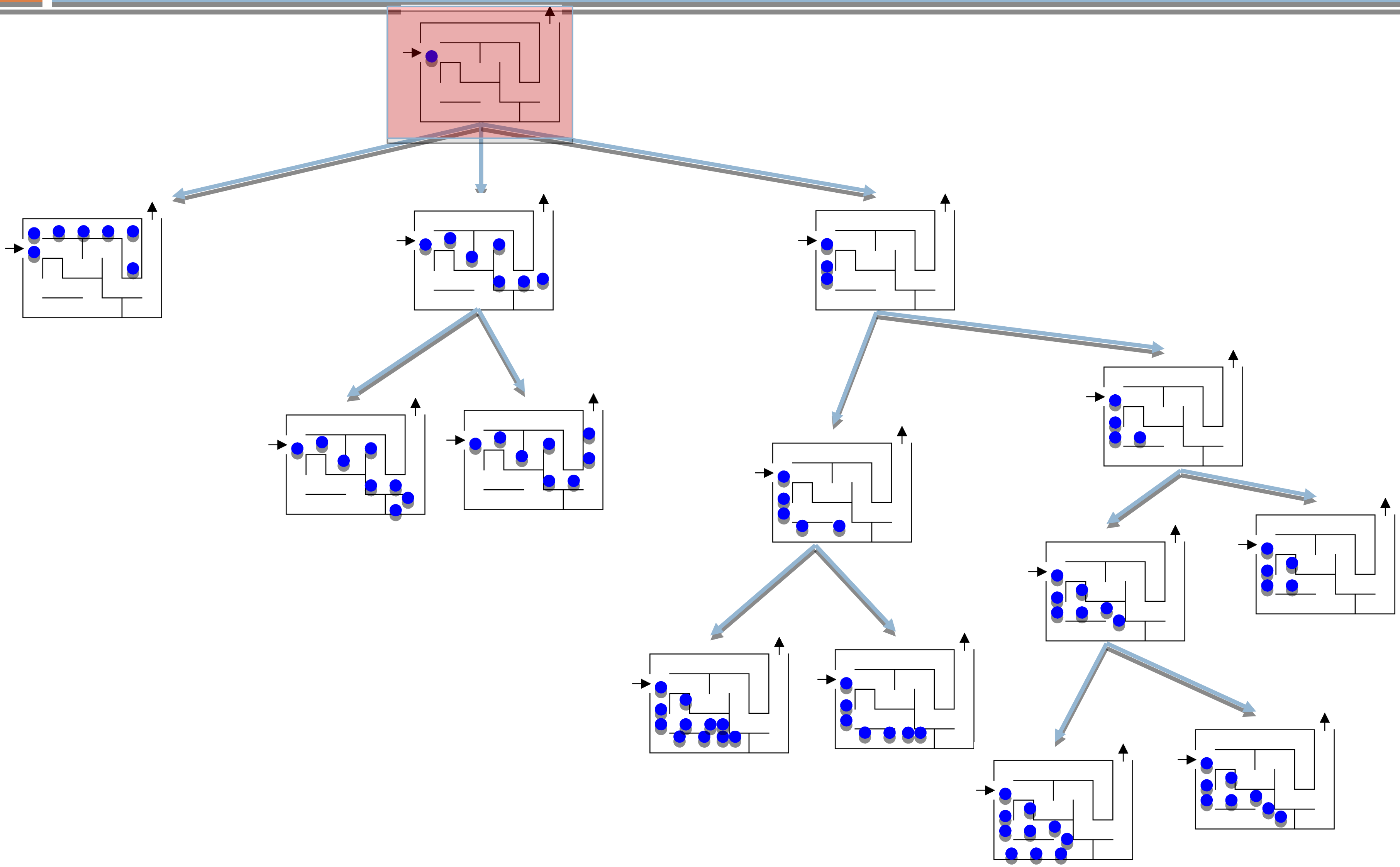
For a given problem, still could have different state-spaces

How many more states are there?

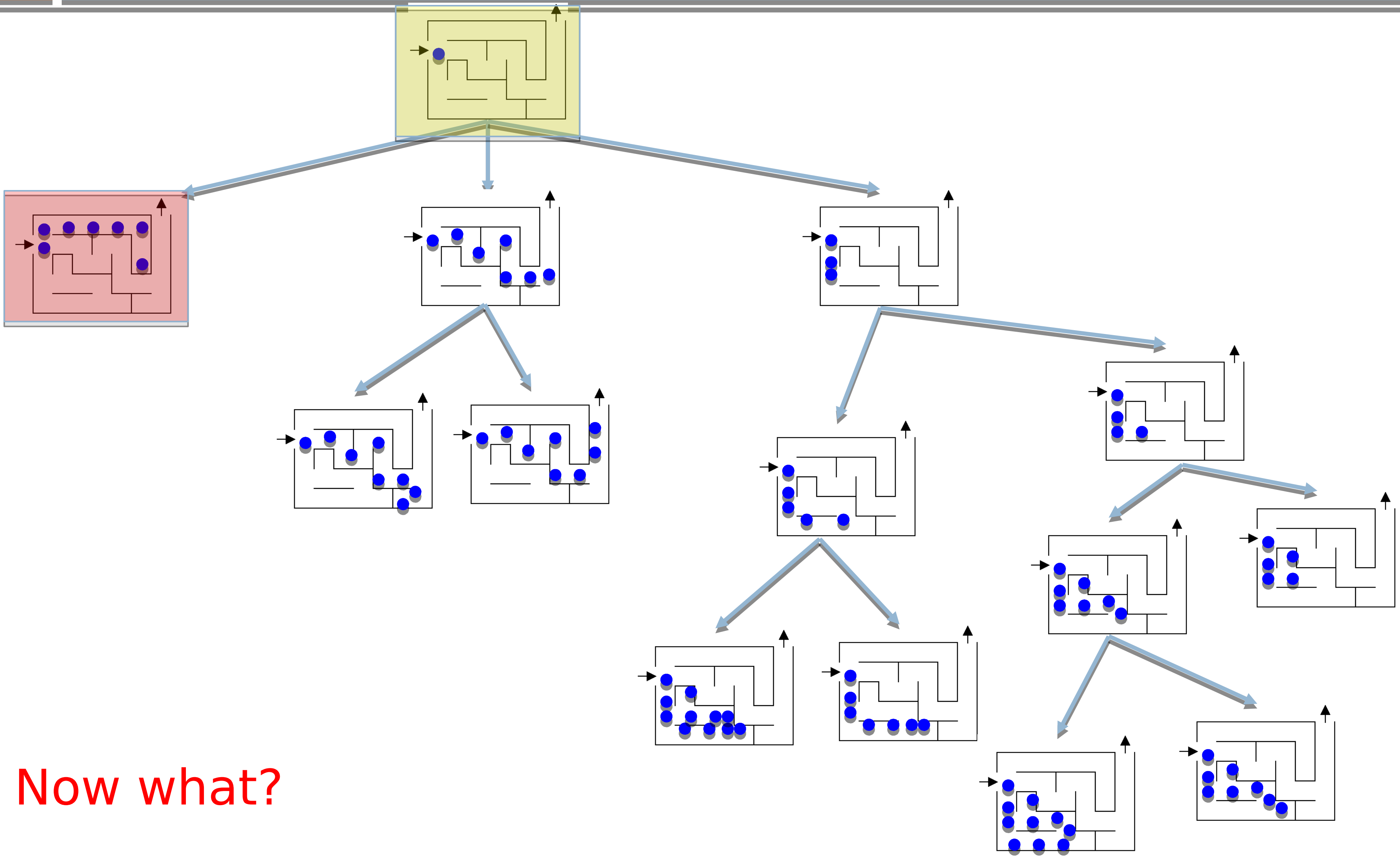
State space example



State space example

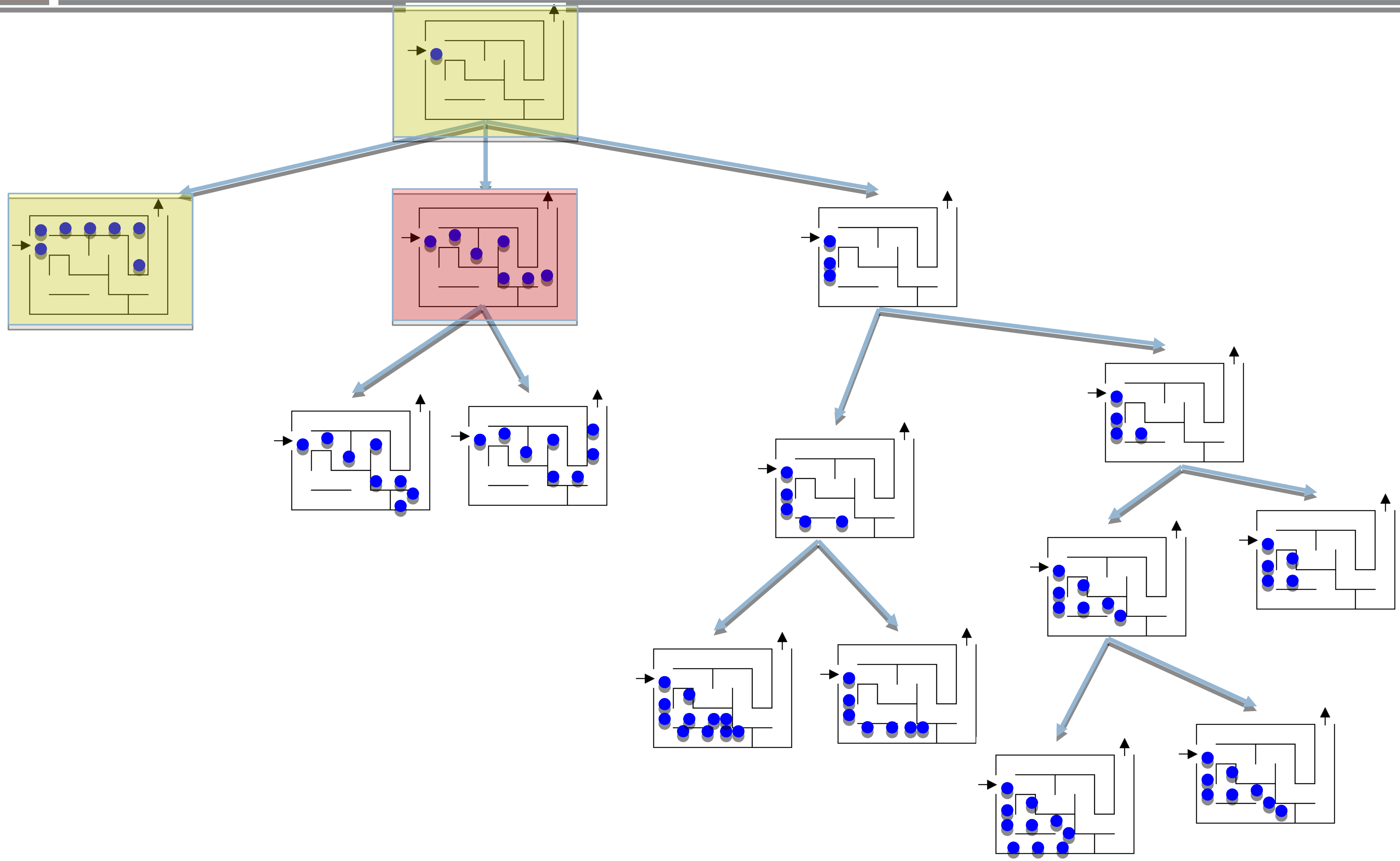


State space example

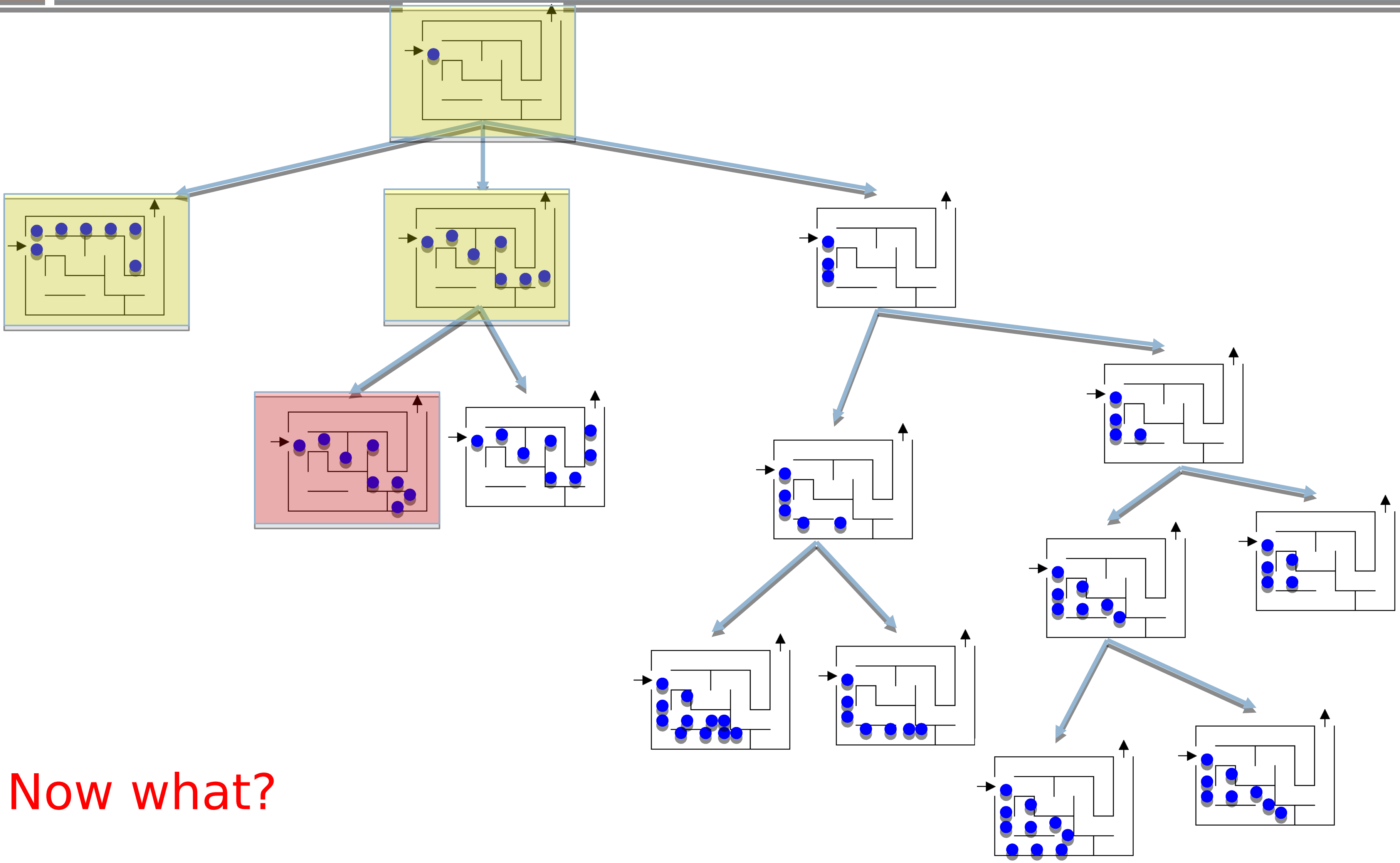


Now what?

State space example

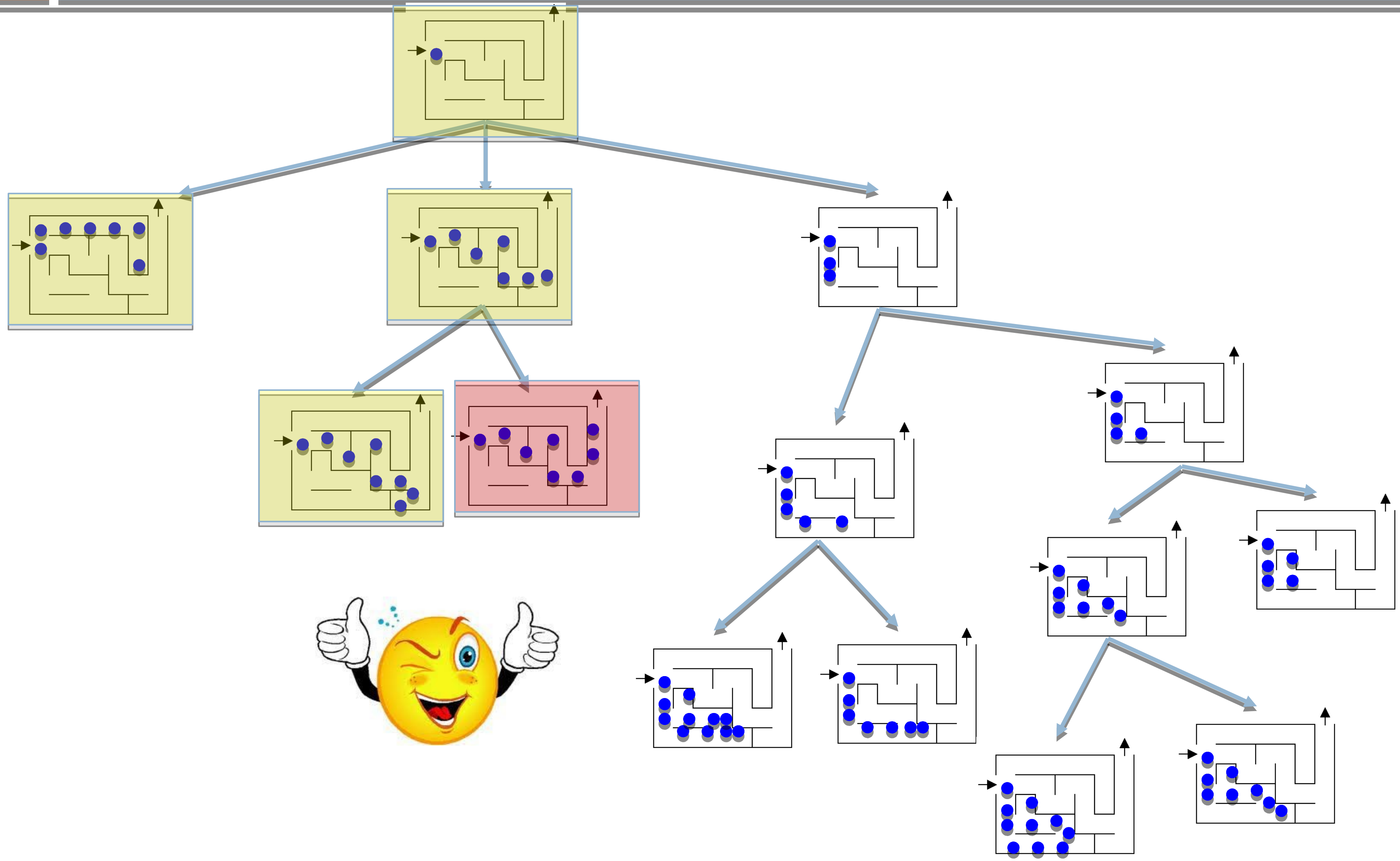


State space example

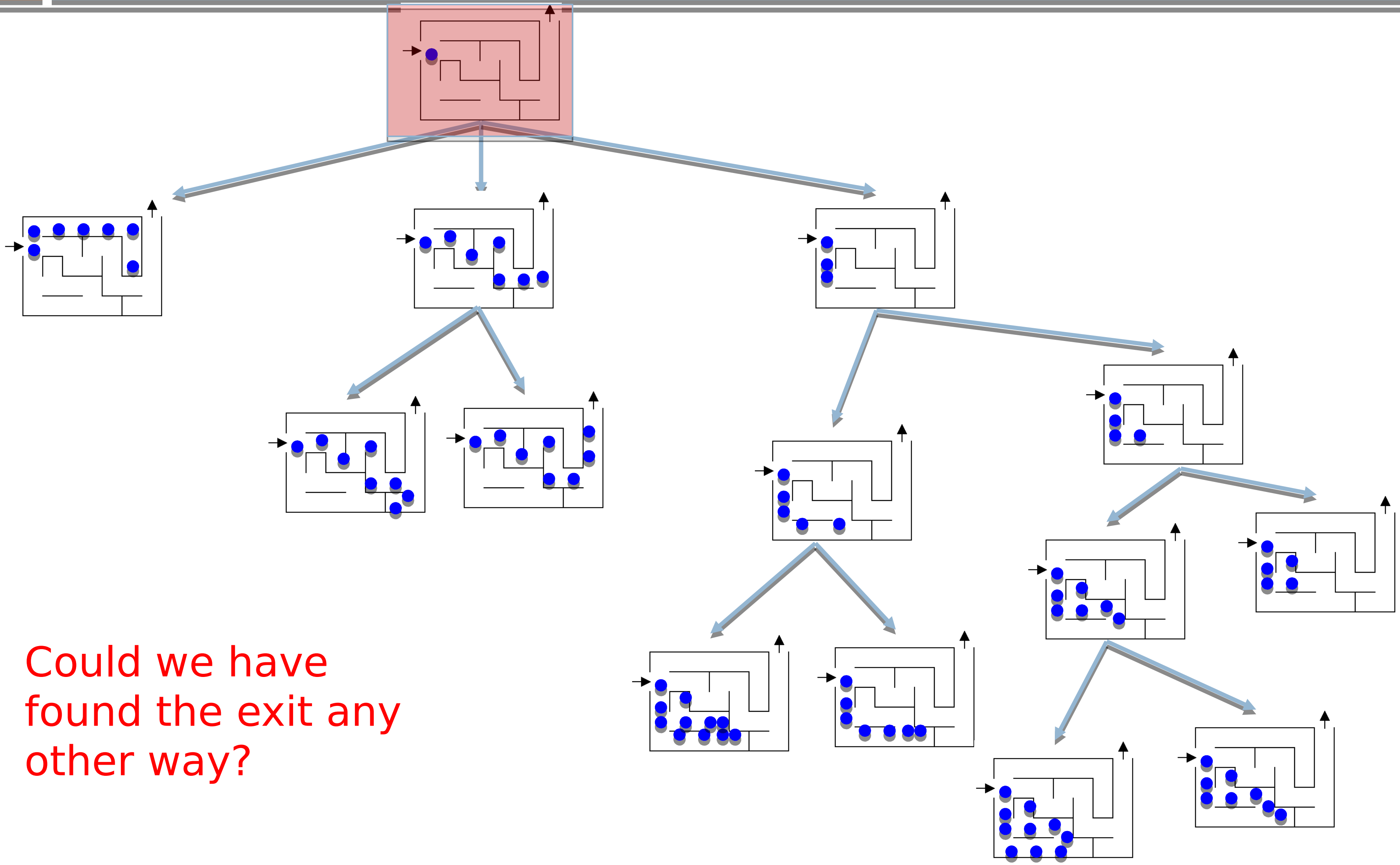


Now what?

State space example



State space example



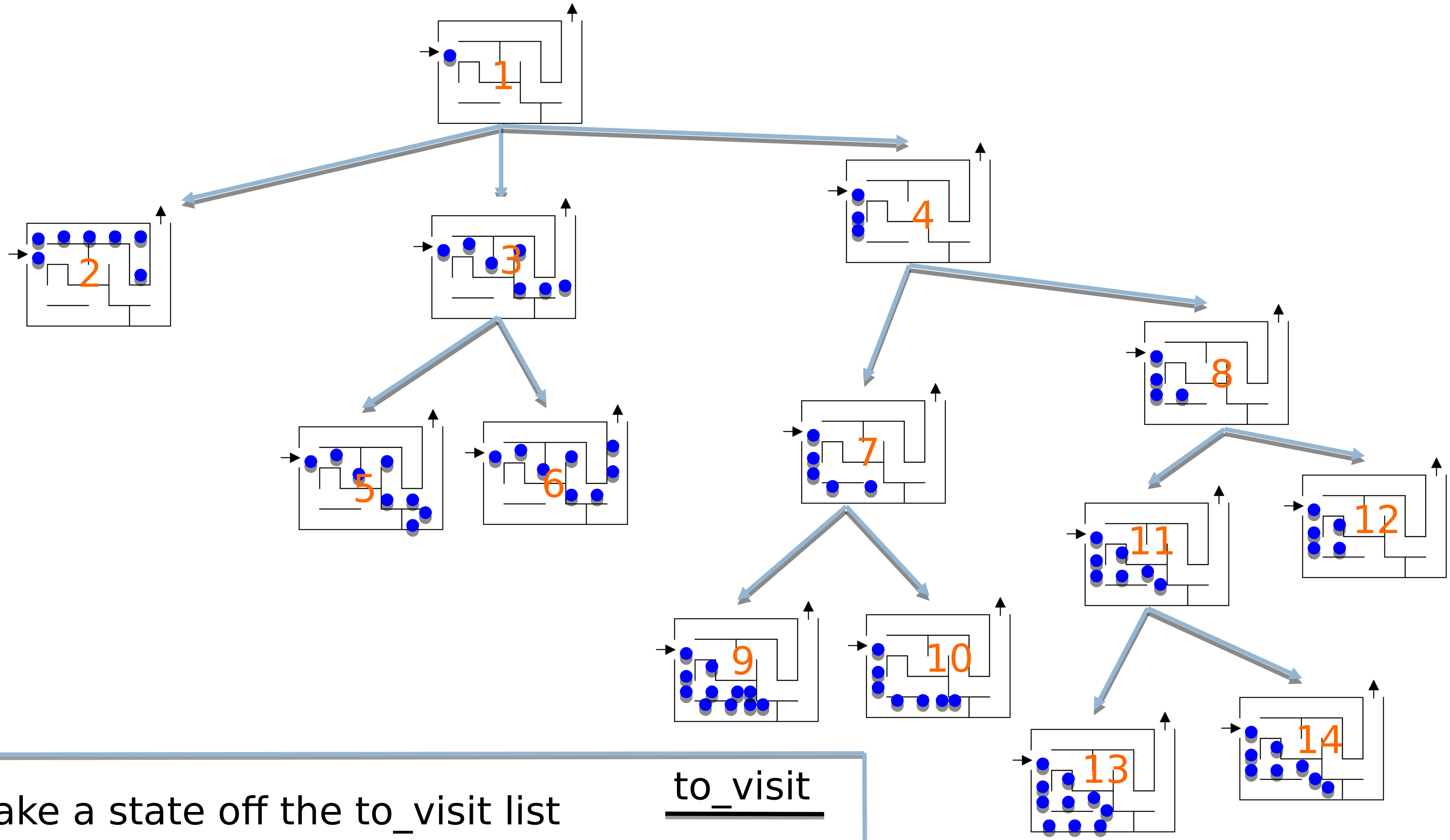
Could we have found the exit any other way?

Search algorithm

Keep track of a list of states that we *could* visit, we’ll call it “to_visit”

General idea:

- take a state off the to_visit list
- if it’s the goal state
  - we’re done!
- if it’s not the goal state
  - Add all of the next states to the to_visit list
- repeat



to_visit

- take a state off the to_visit list
- if it’s the goal state

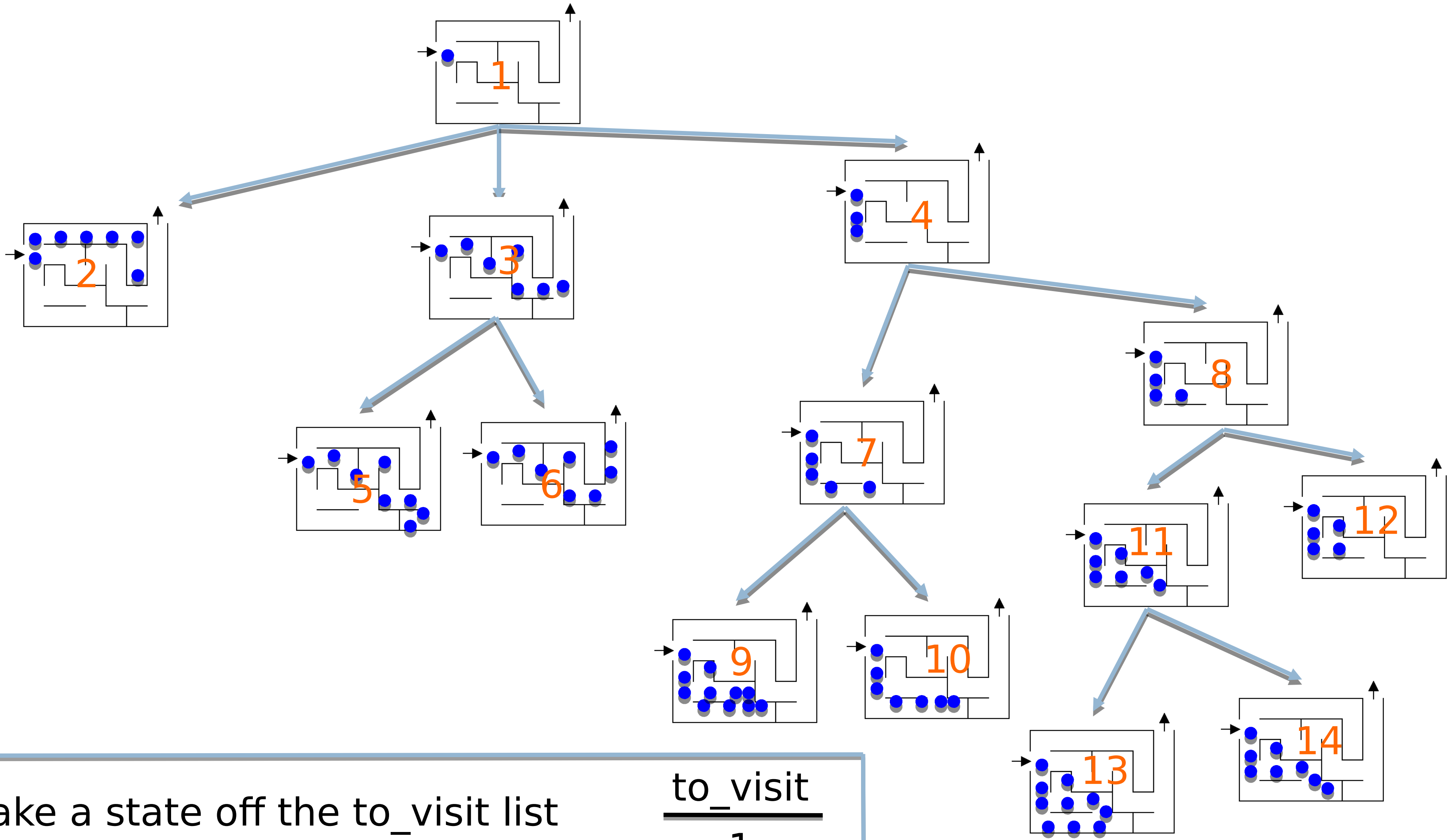
  we’re done!

  - if it’s not the goal state

  Add all of the next states to the to_visit list

  - repeat

How do we start?

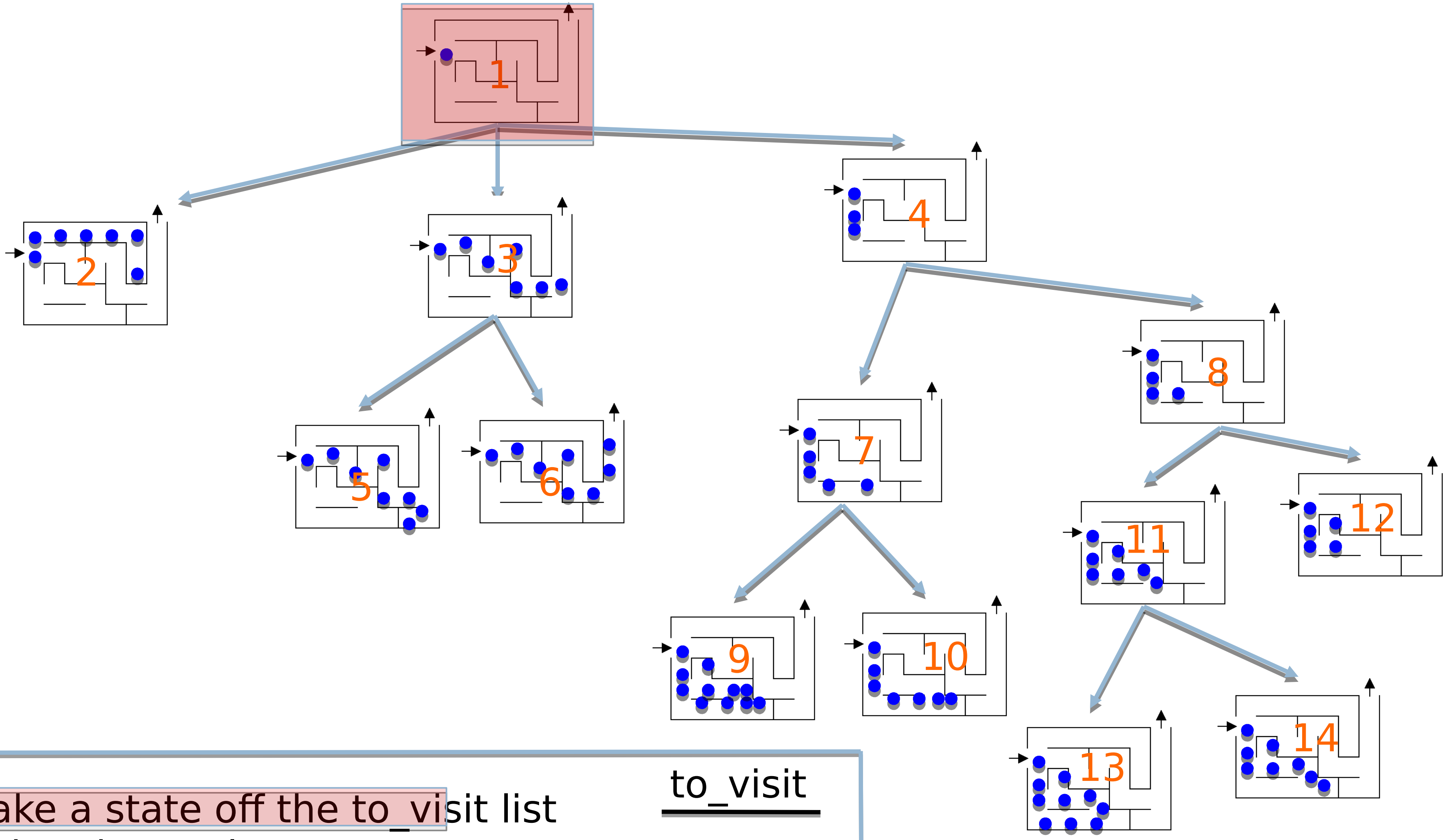


- take a state off the to_visit list
- if it’s the goal state
    we’re done!
    - if it’s not the goal state
    Add all of the next states to
        the to_visit list
    - repeat

to_visit

1

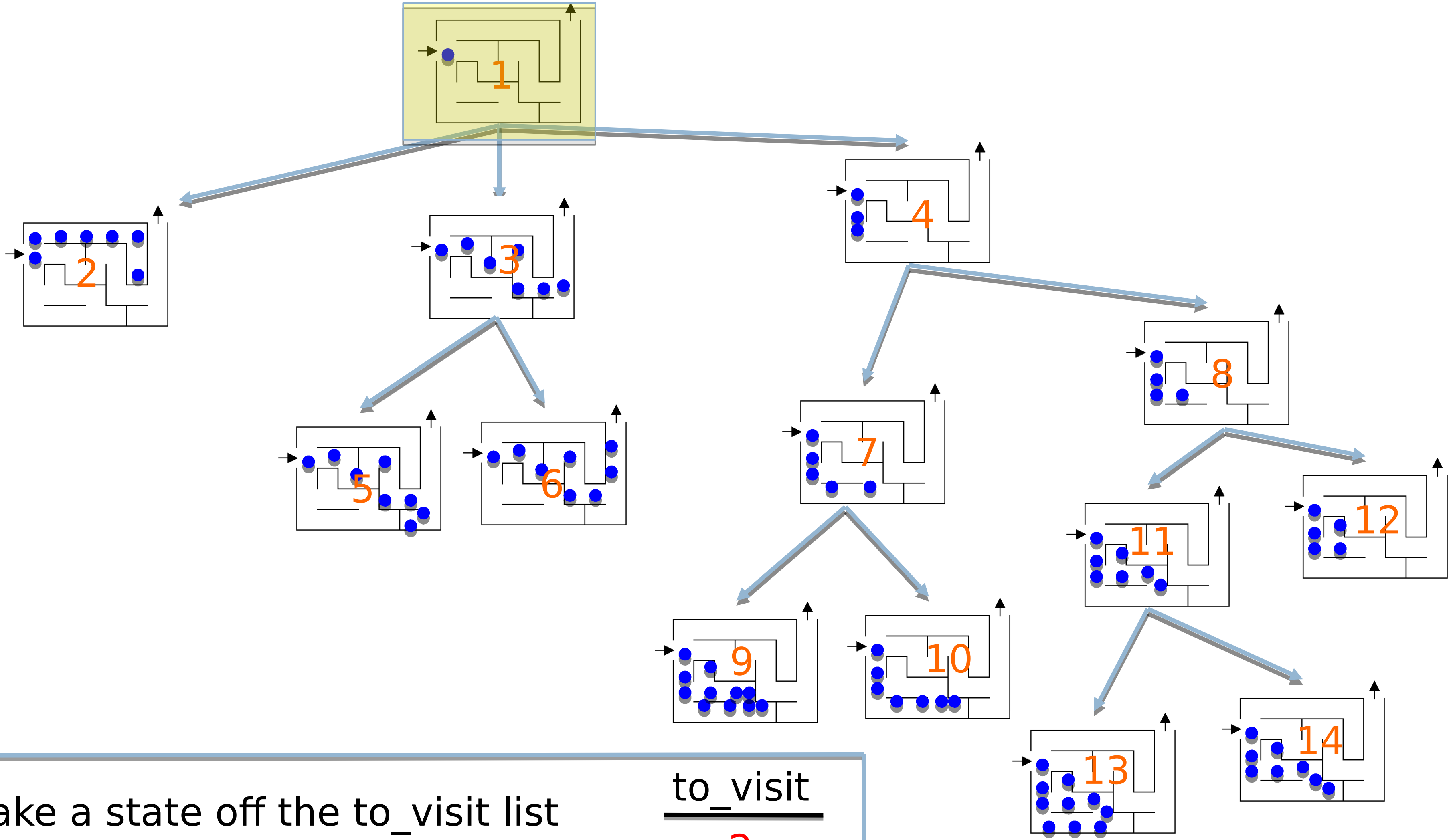
Add start to to_visit



to_visit

- take a state off the to_visit list
- if it’s the goal state
  we’re done!
  - if it’s not the goal state
  Add all of the next states to
    the to_visit list
  - repeat

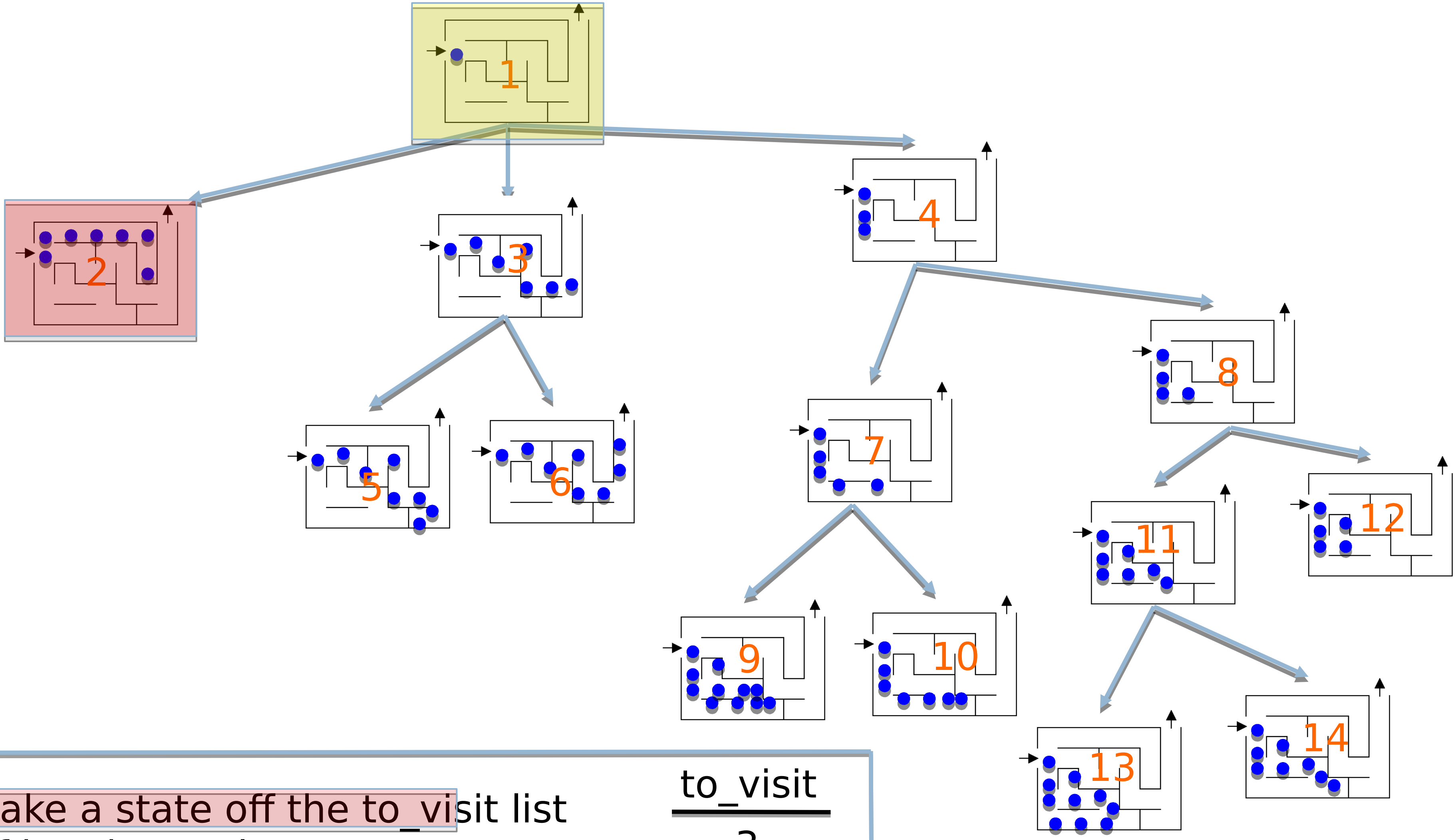
Is it a goal state?



- take a state off the to_visit list
- if it's the goal state
  we're done!
  - if it's not the goal state
  Add all of the next states to
  the to_visit list
  - repeat

to_visit

2
3
4

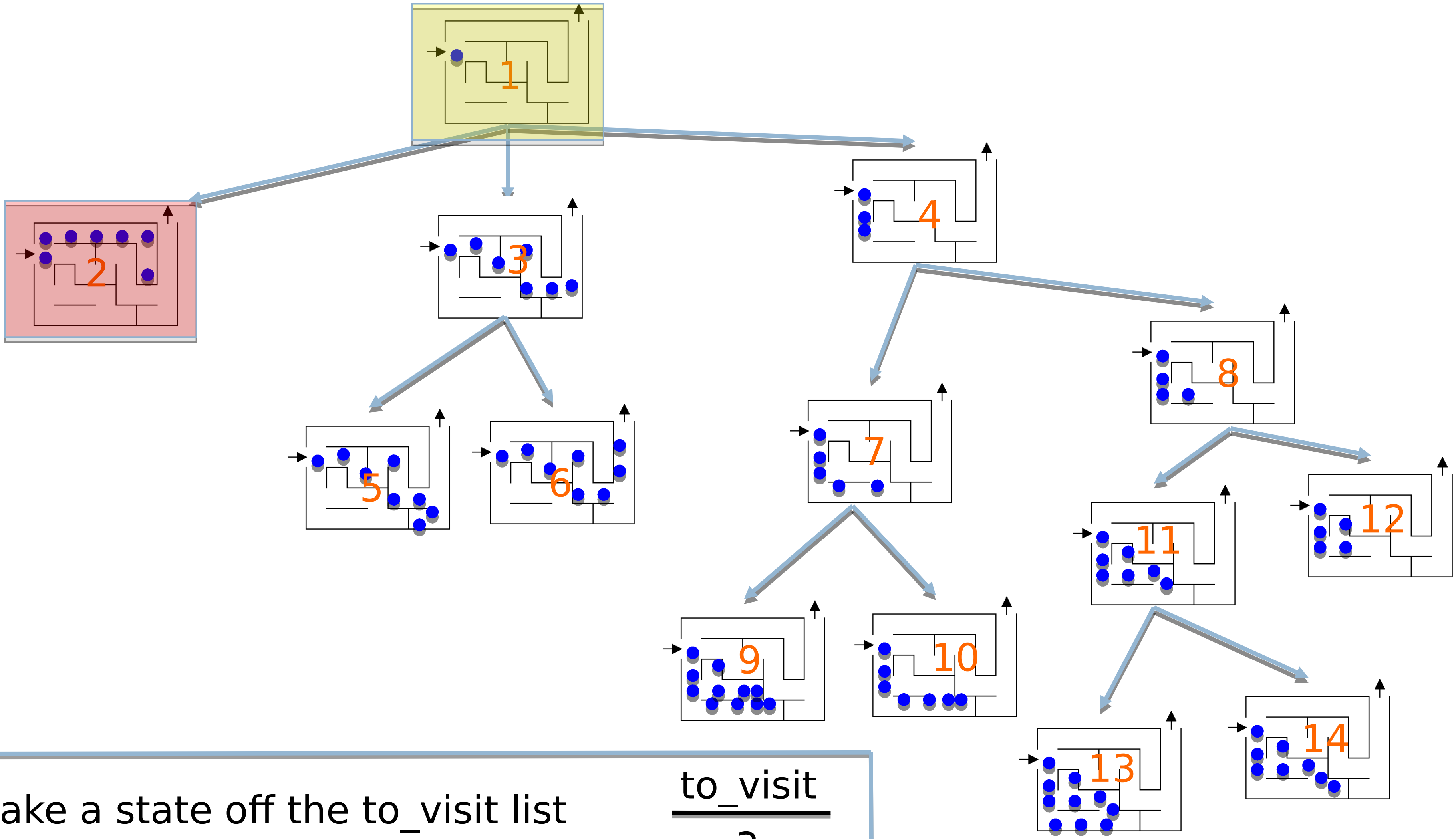


- take a state off the to_visit list
- if it’s the goal state
  we’re done!
  - if it’s not the goal state
  Add all of the next states to
    the to_visit list
  - repeat

to_visit

3
4

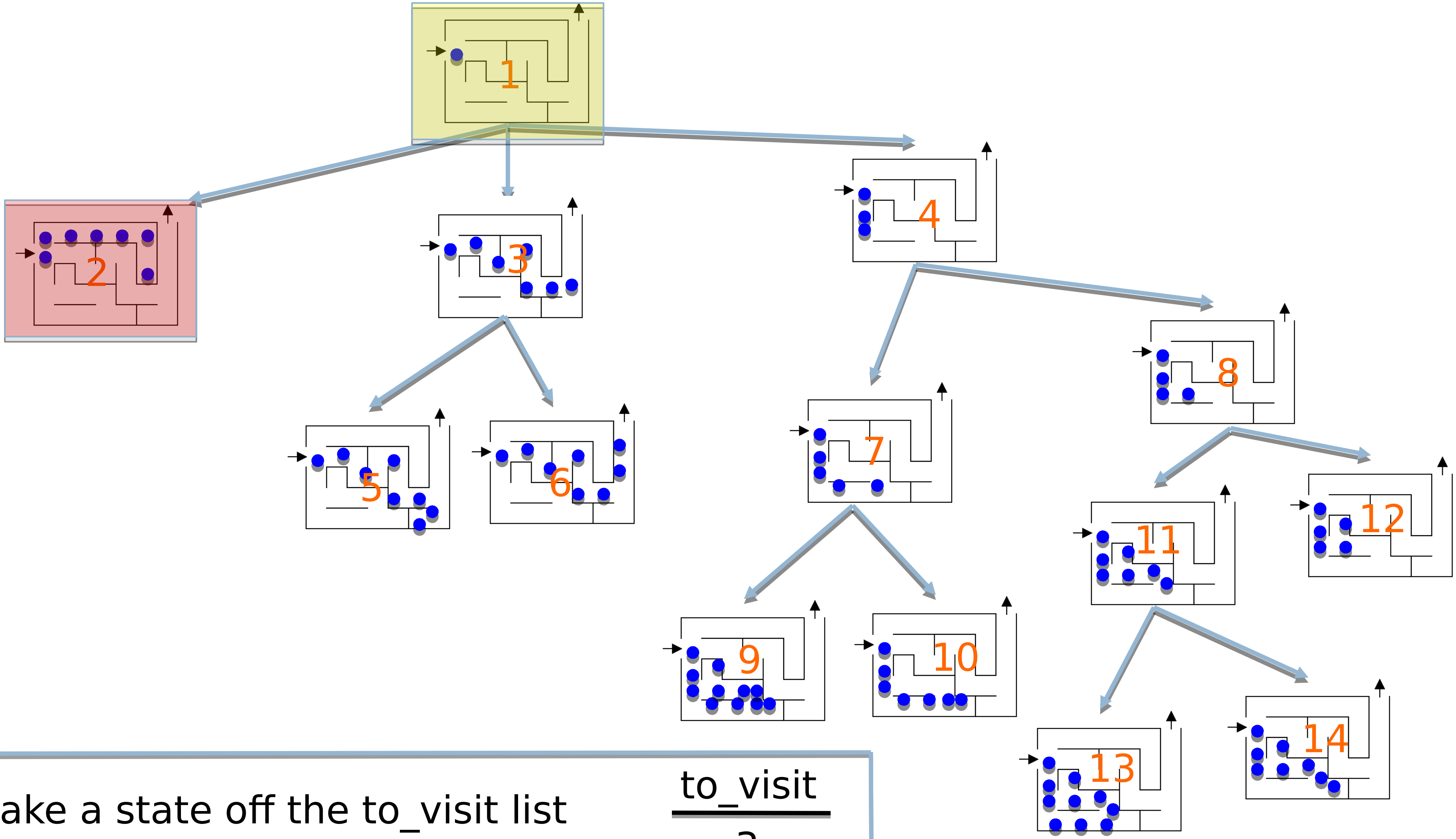
Is it a goal state?



to_visit

3
4

- take a state off the to_visit list
- if it’s the goal state
  we’re done!
  - if it’s not the goal state
  Add all of the next states to
  the to_visit list
  - repeat

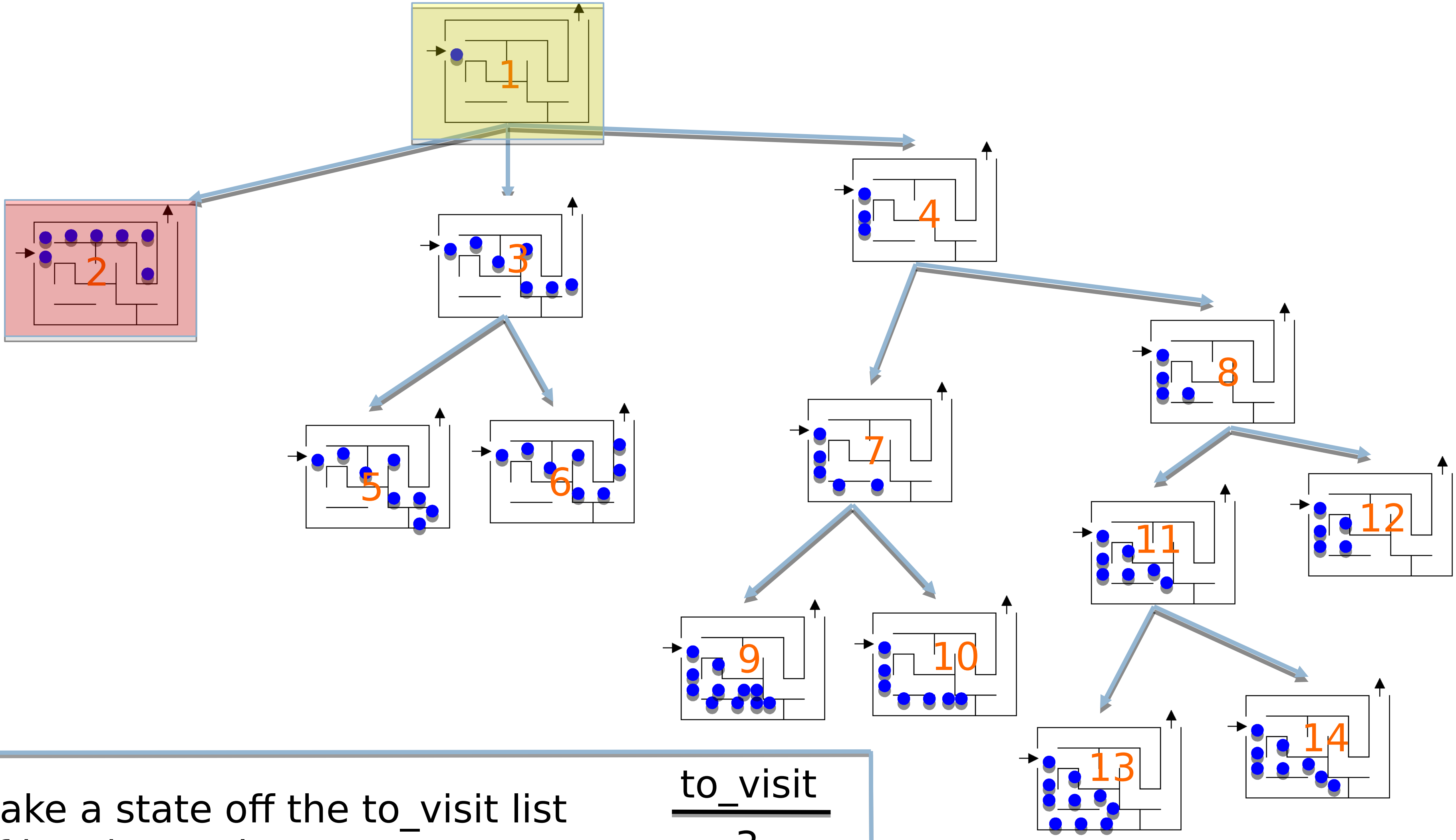


- take a state off the to_visit list
- if it's the goal state
    we're done!
    - if it's not the goal state
    Add all of the next states to
      the to_visit list
    - repeat

to_visit

3
4

Dead-end. What do we do now?

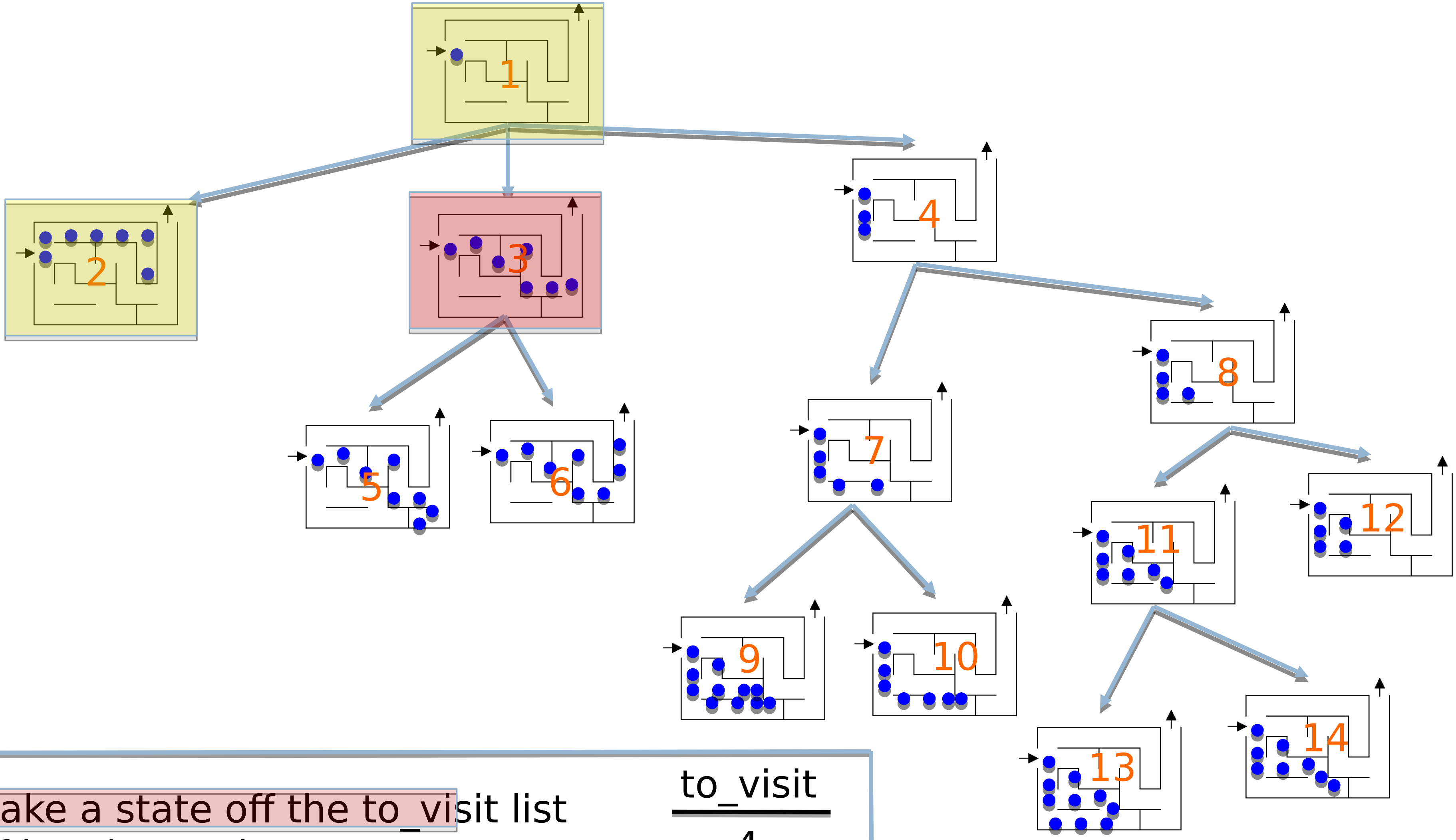


- take a state off the to_visit list
- if it's the goal state
  - we're done!
  - if it's not the goal state Add all of the next states to the to_visit list
  - repeat

to_visit

3
4

list keeps track of where to go next, i.e. the states we know about but haven't explored



- take a state off the to_visit list
- if it’s the goal state

  we’re done!

  - if it’s not the goal state

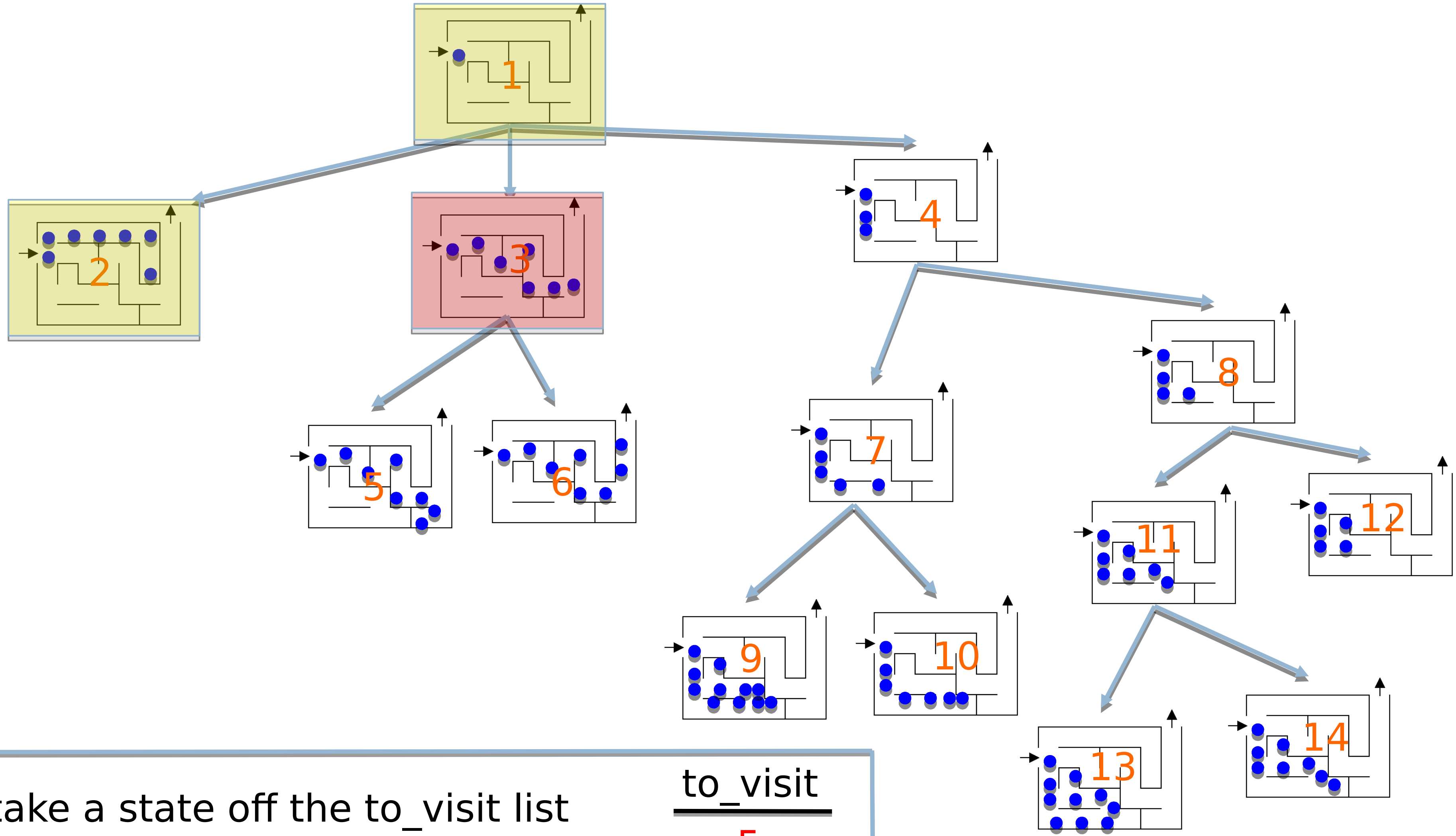
  Add all of the next states to
  the to_visit list

  - repeat

to_visit

4

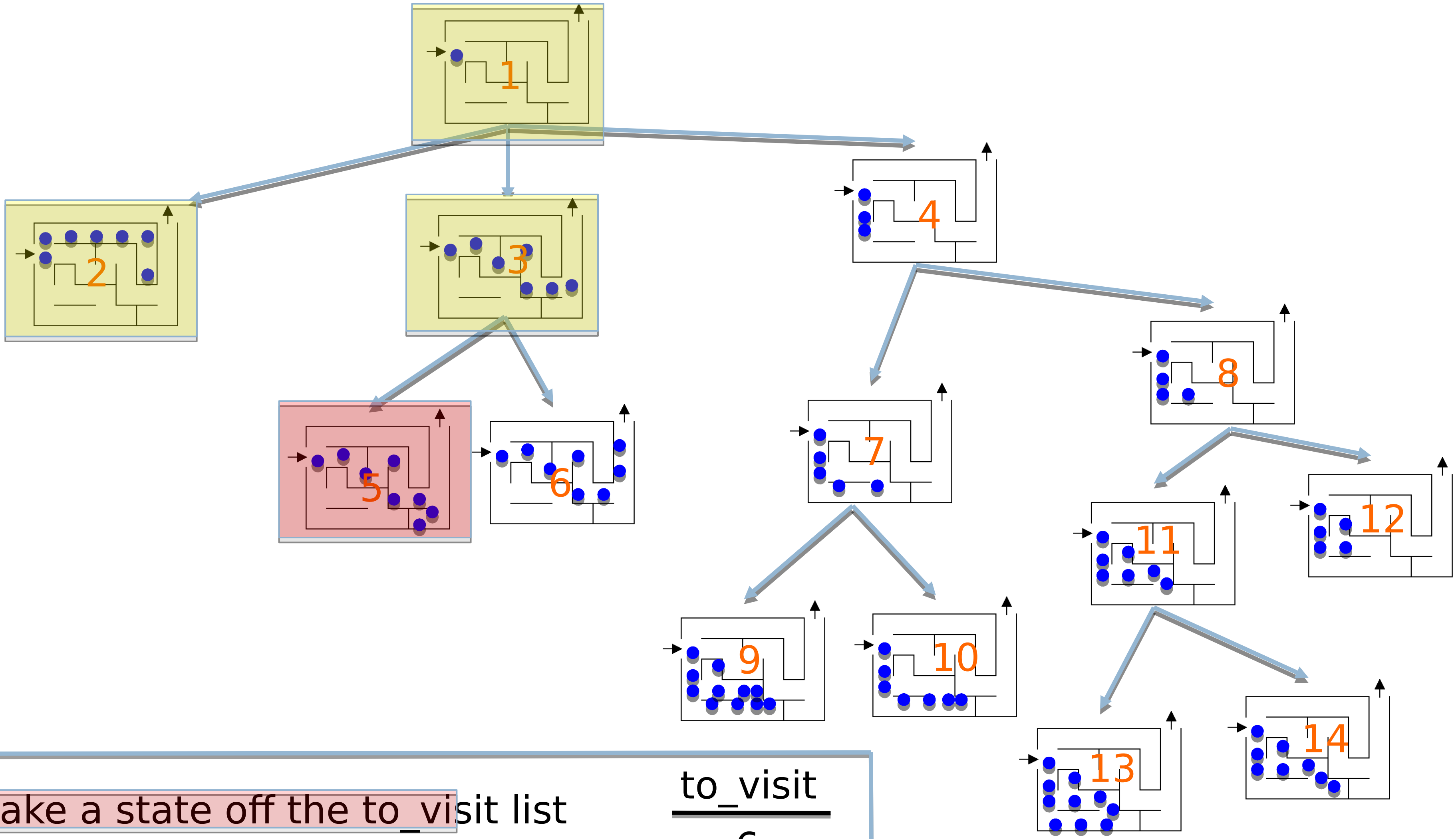
Is it a goal state?



to_visit

5
6
4

- take a state off the to_visit list
- if it’s the goal state
    - we’re done!
    - if it’s not the goal state Add all of the next states to the to_visit list
    - repeat

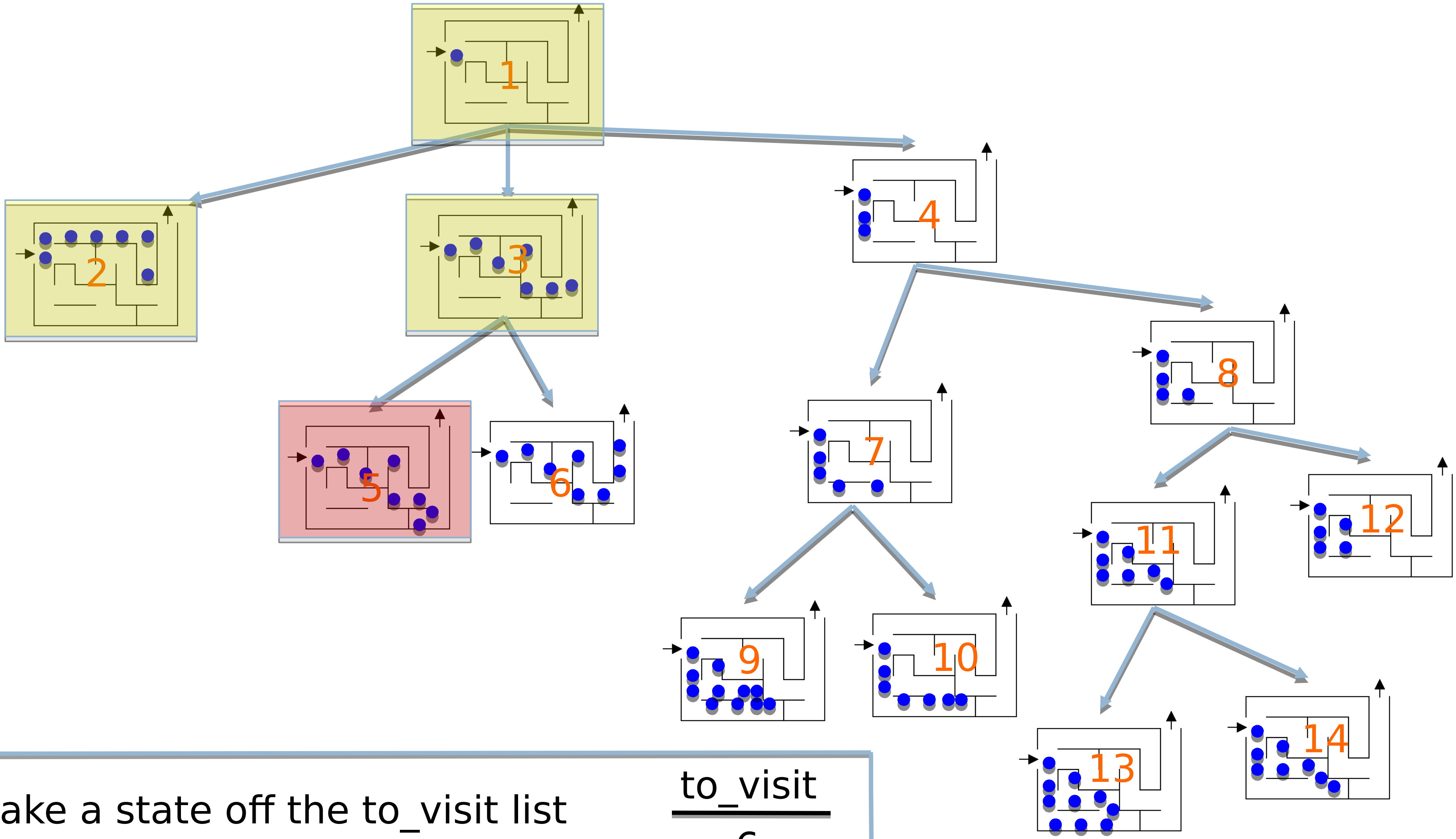


- take a state off the to_visit list
- if it's the goal state
  - we're done!
  - if it's not the goal state
    Add all of the next states to
    the to_visit list
  - repeat

to_visit

6
4

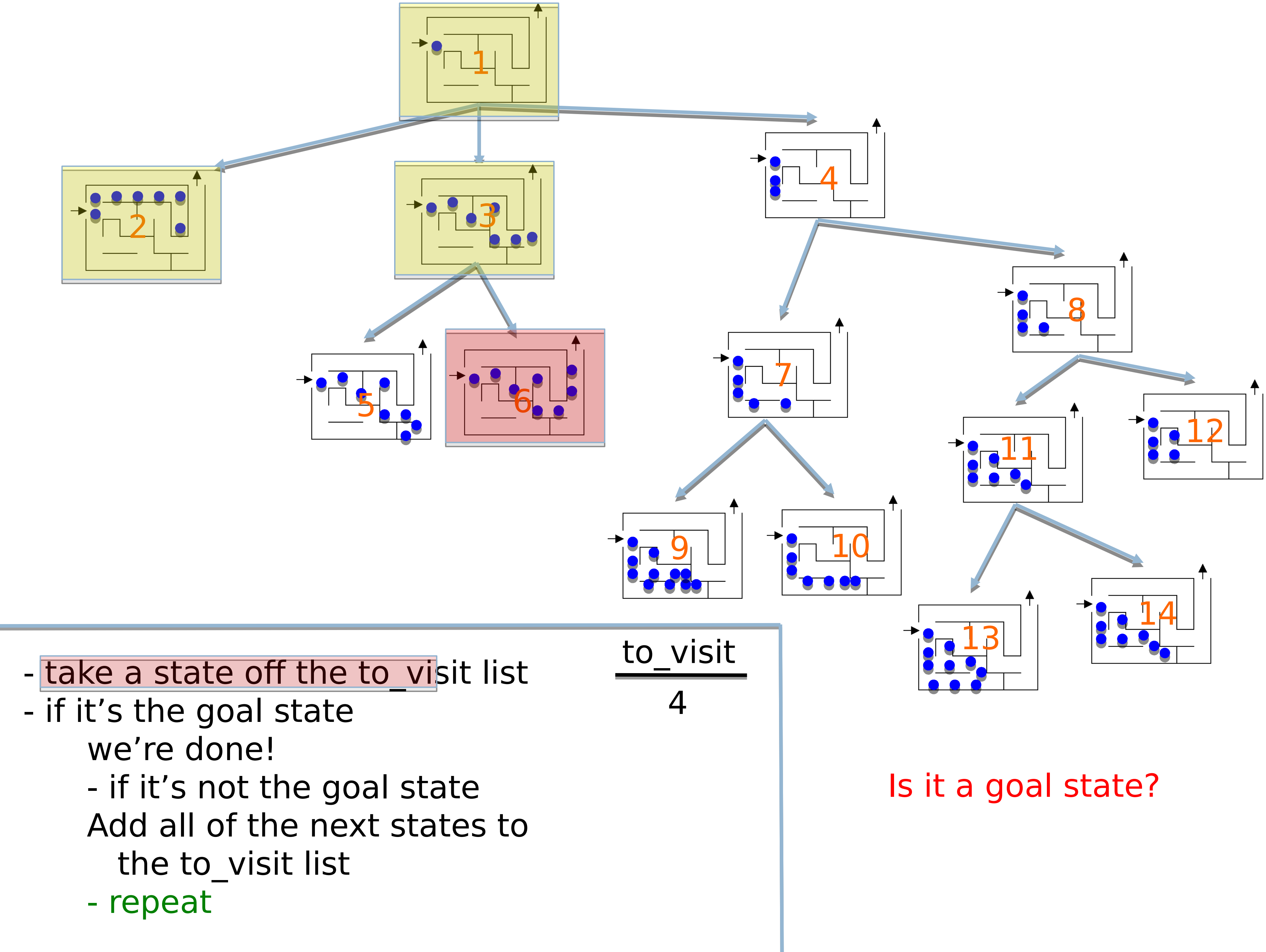
Is it a goal state?



- take a state off the to_visit list
- if it’s the goal state
    we’re done!
    - if it’s not the goal state
    Add all of the next states to
      the to_visit list
    - repeat

to_visit

6
4

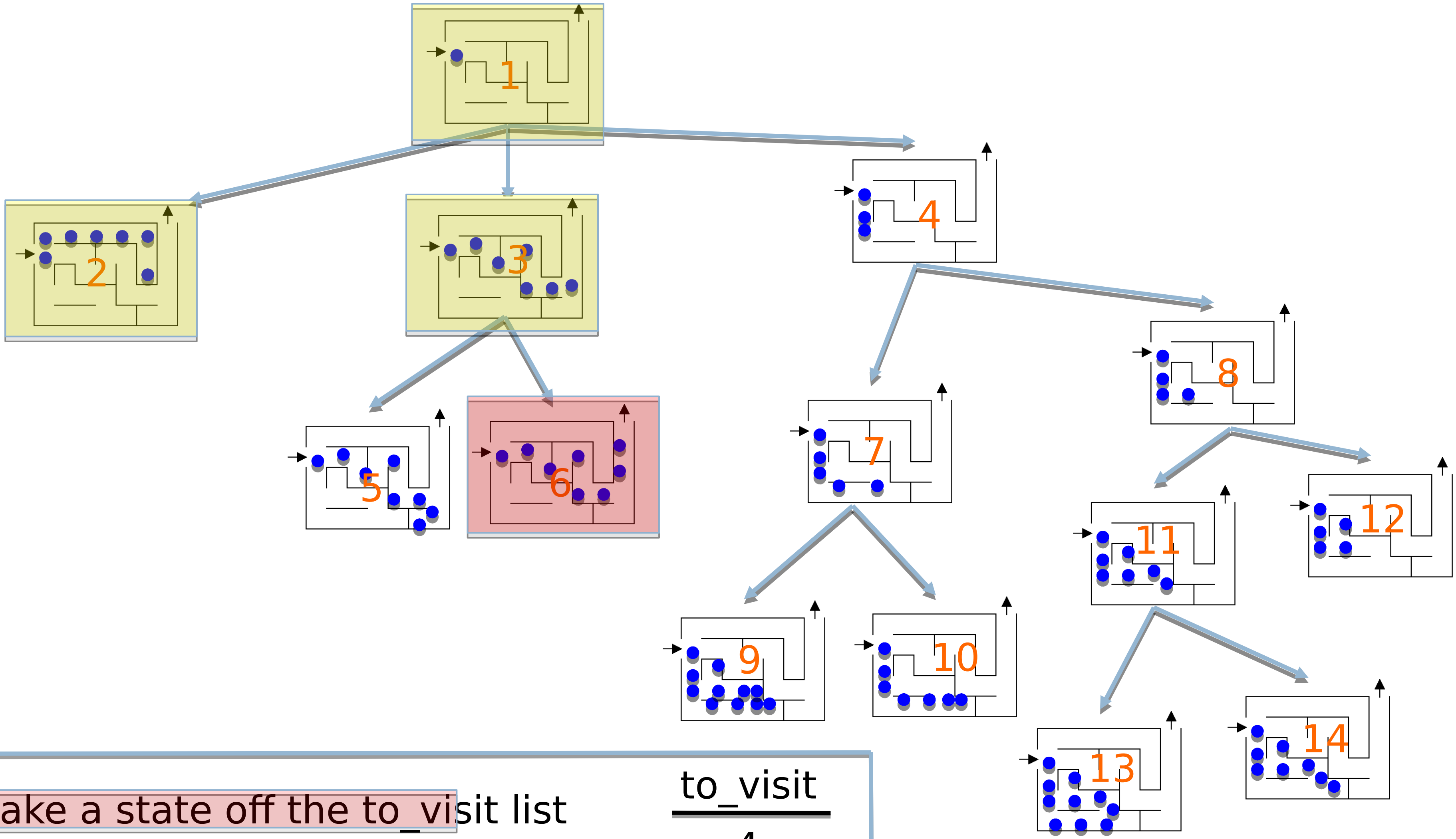


- take a state off the to_visit list
- if it’s the goal state
  we’re done!
  - if it’s not the goal state
  Add all of the next states to
  the to_visit list
  - repeat

to_visit

4

Is it a goal state?



- take a state off the to_visit list
- if it’s the goal state

    we’re done!

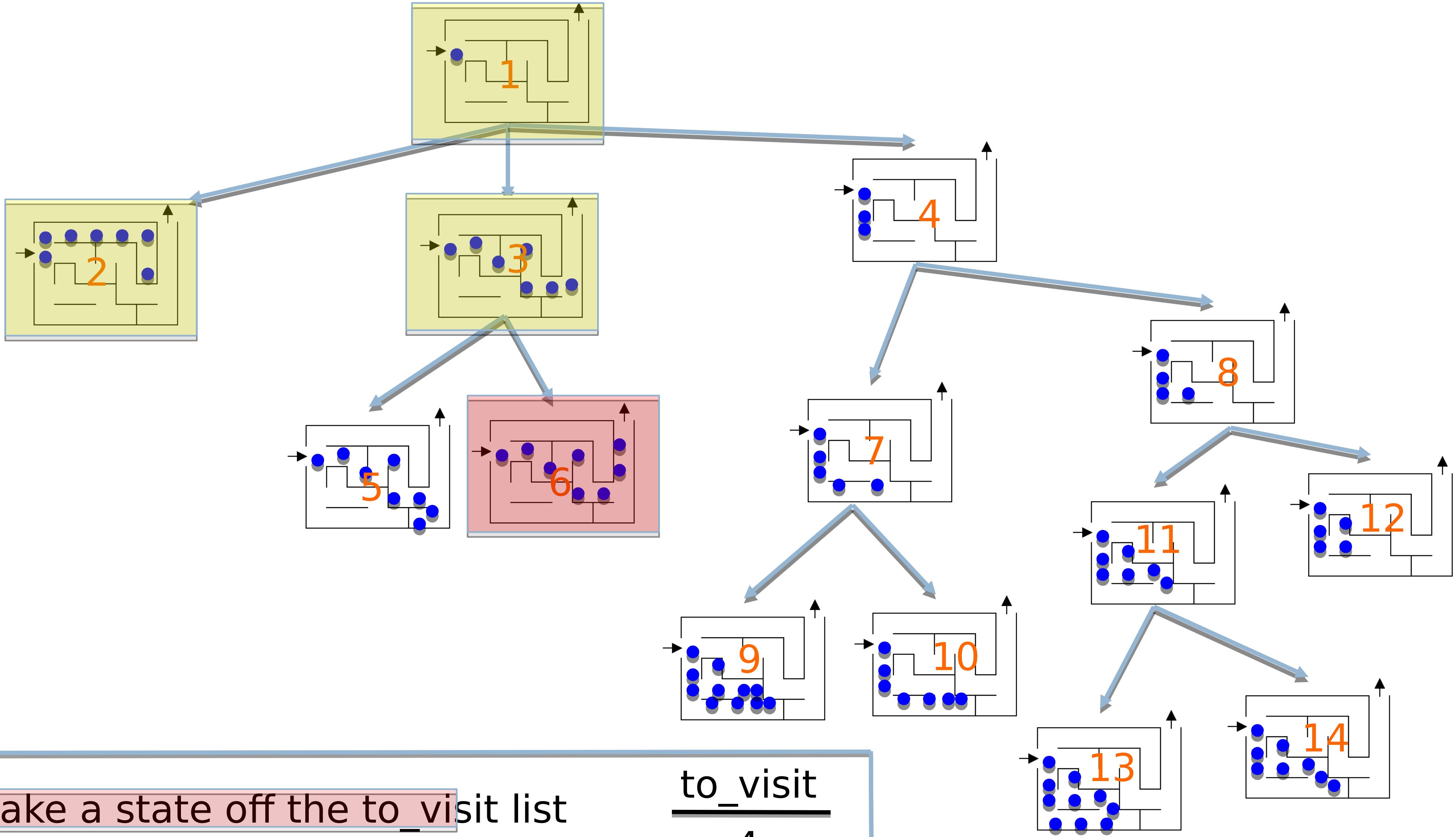
    - if it’s not the goal state

    Add all of the next states to
      the to_visit list

    - repeat

to_visit

4



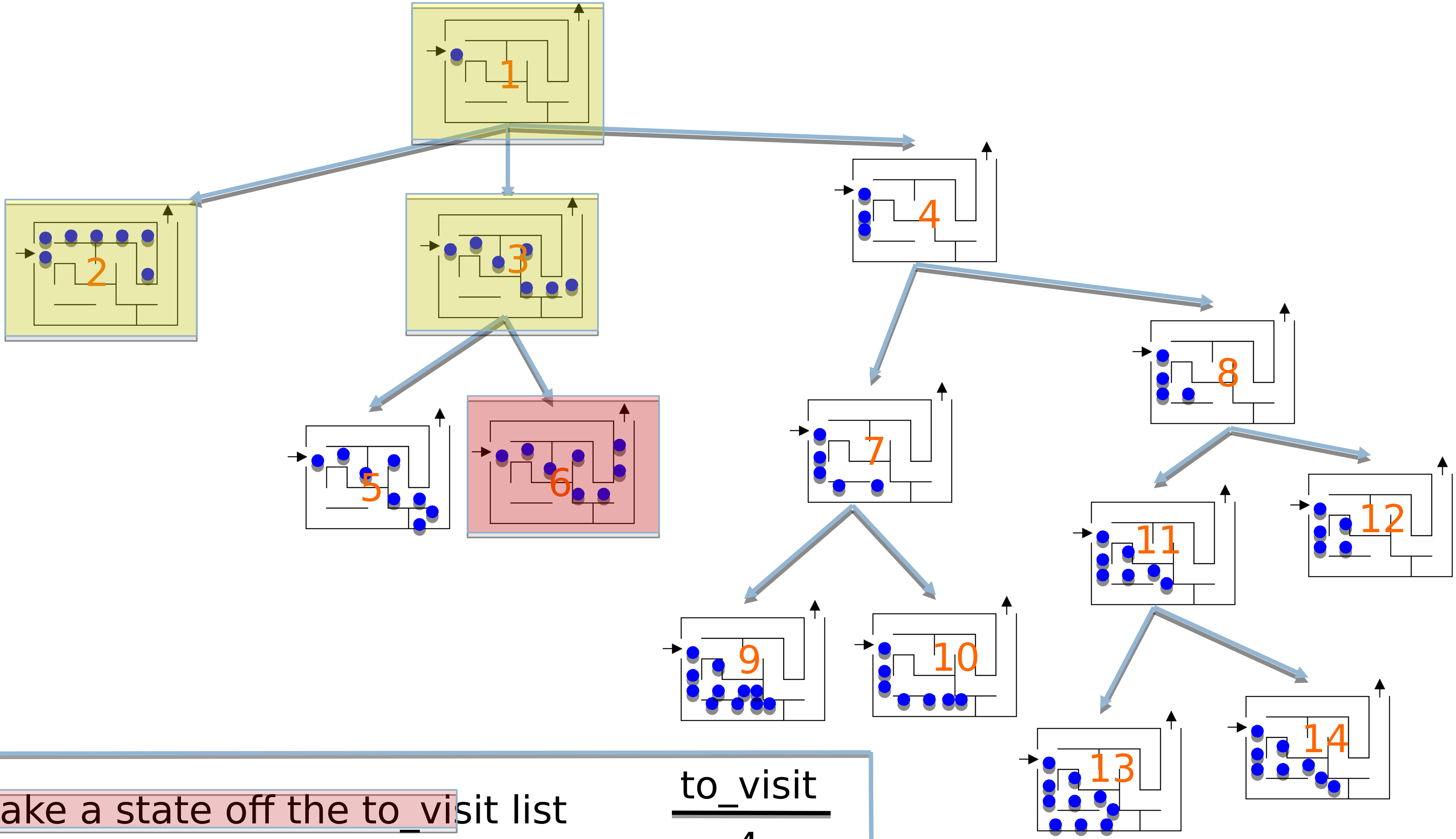
- take a state off the to_visit list
- if it's the goal state
    we're done!
    - if it's not the goal state
    Add all of the next states to
        the to_visit list
    - repeat

to_visit

4

How was the to_visit list organized in this example, i.e., what order?

It's a stack!!! (LIFO)



- take a state off the to_visit list
- if it’s the goal state
  we’re done!
  - if it’s not the goal state
  Add all of the next states to
    the to_visit list
  - repeat

to_visit

4

What would happen
if we used a queue?

Search algorithms

add the start state to to_visit

Repeat

- take a state off the to_visit list
- if it’s the goal state
    - we’re done!
- if it’s not the goal state
    - Add all of the next states to the to_visit list

Search algorithms

add the start state to to_visit

Repeat

- take a state off the to_visit list
- if it’s the goal state
  - we’re done!
- if it’s not the goal state
  - Add all of the next states to the to_visit list

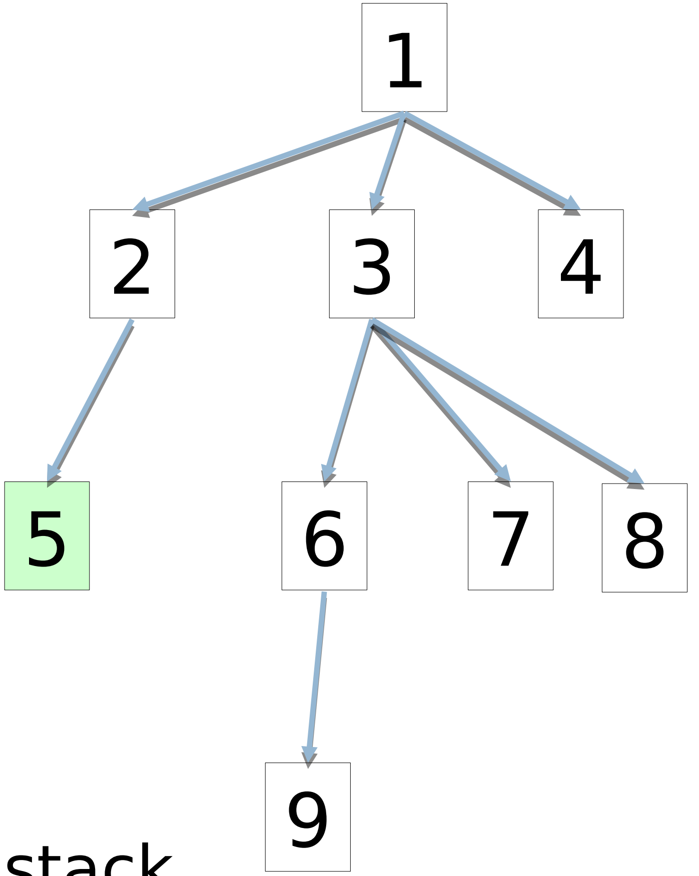
Depth first search (DFS): to_visit is a stack
Breadth first search (BFS): to_visit is a queue

What order will BFS and DFS visit the states assuming states are added to to_visit left to right?

add the start state to to_visit

Repeat

- take a state off the to_visit list
- if it's the goal state
  - we're done!
- if it's not the goal state
  - Add all of the successive states to the to_visit list

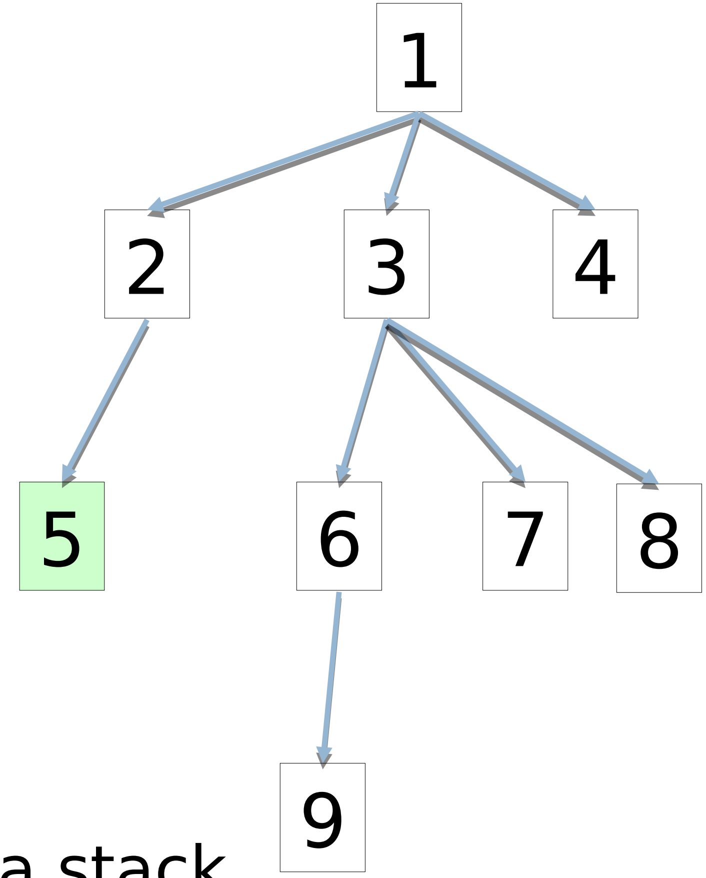


Depth first search (DFS): to_visit is a stack
Breadth first search (BFS): to_visit is a queue

What order will BFS and DFS visit the states?

DFS: 1, 4, 3, 8, 7, 6, 9, 2, 5

Why not 1, 2, 5?

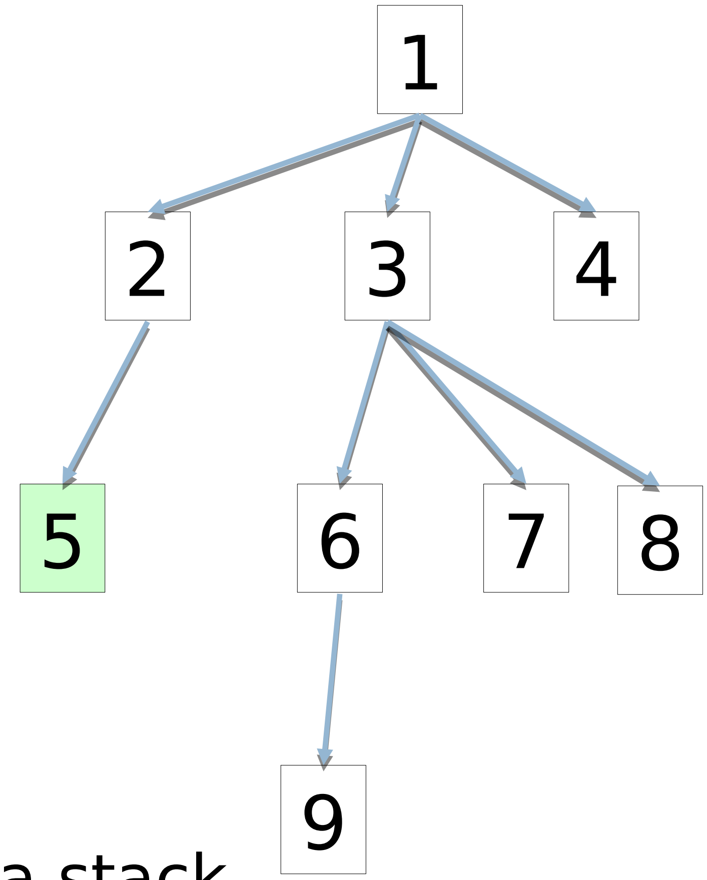


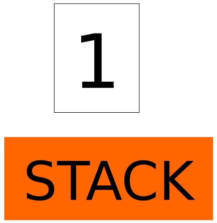
Depth first search (DFS): to_visit is a stack

Breadth first search (BFS): to_visit is a queue

What order will BFS and DFS visit the states?

DFS: 1, 4, 3, 8, 7, 6, 9, 2, 5



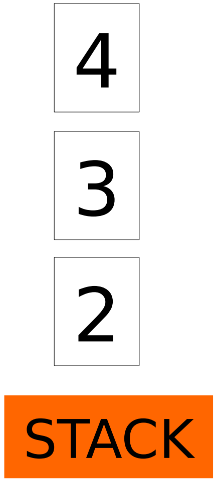


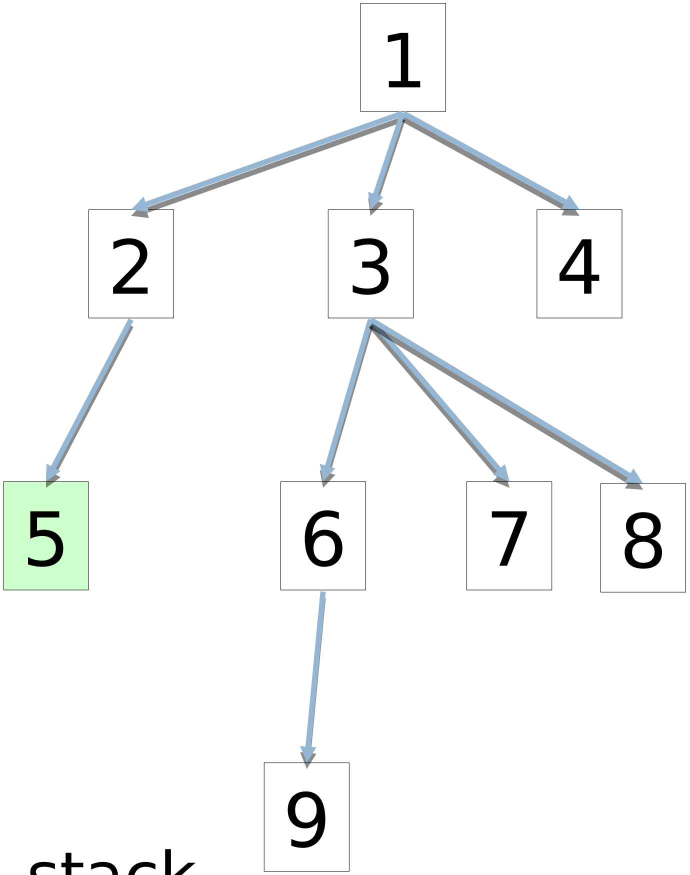
Depth first search (DFS): to_visit is a stack

Breadth first search (BFS): to_visit is a queue

What order will BFS and DFS visit the states?

DFS: 1, 4, 3, 8, 7, 6, 9, 2, 5

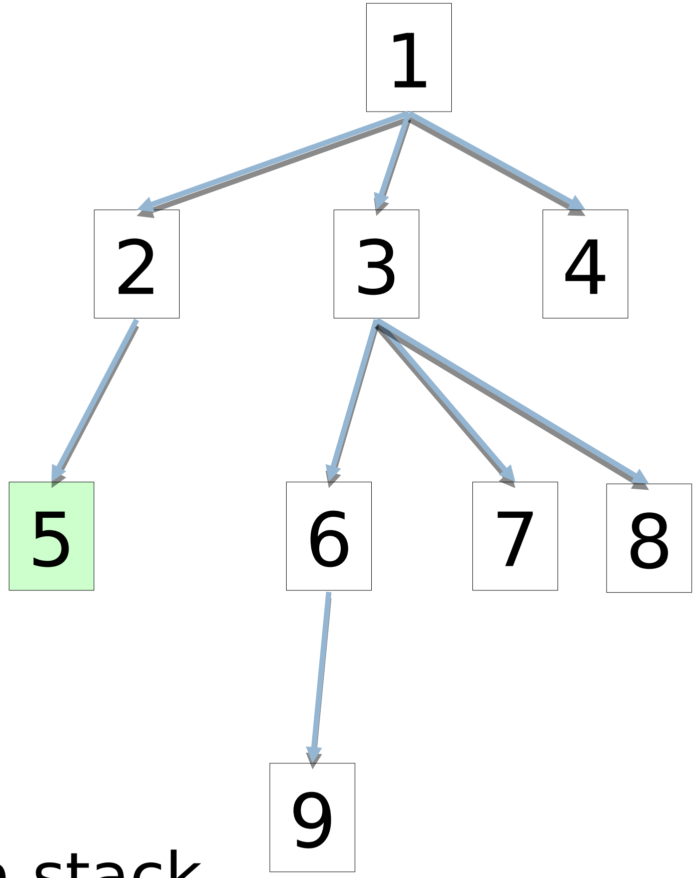


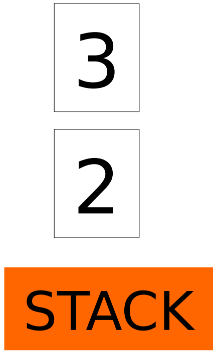


Depth first search (DFS): to_visit is a stack
Breadth first search (BFS): to_visit is a queue

What order will BFS and DFS visit the states?

DFS: 1, 4, 3, 8, 7, 6, 9, 2, 5



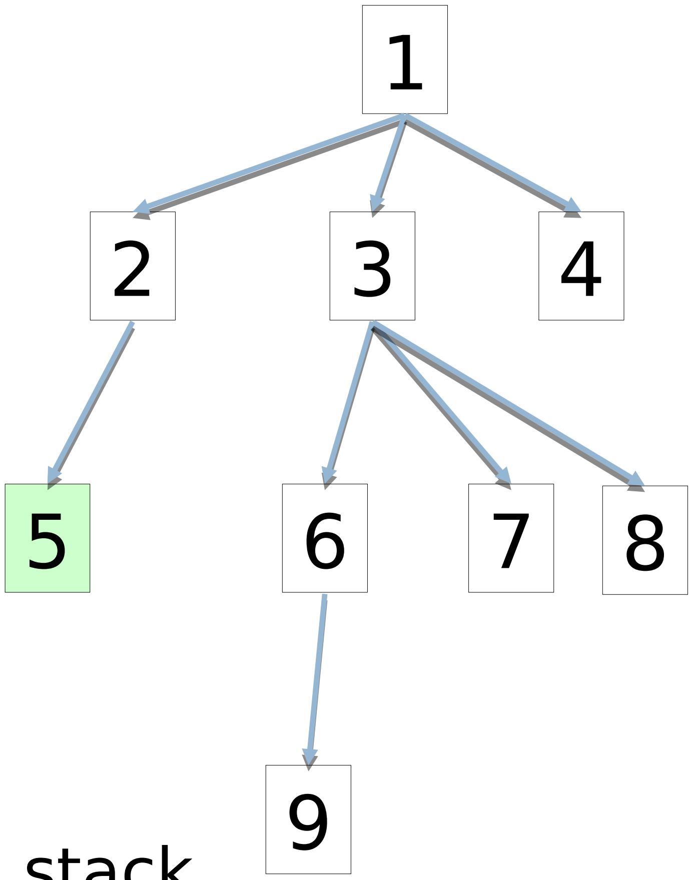


Depth first search (DFS): to_visit is a stack
Breadth first search (BFS): to_visit is a queue

What order will BFS and DFS visit the states?

DFS: 1, 4, 3, 8, 7, 6, 9, 2, 5

BFS:1, 2, 3, 4, 5



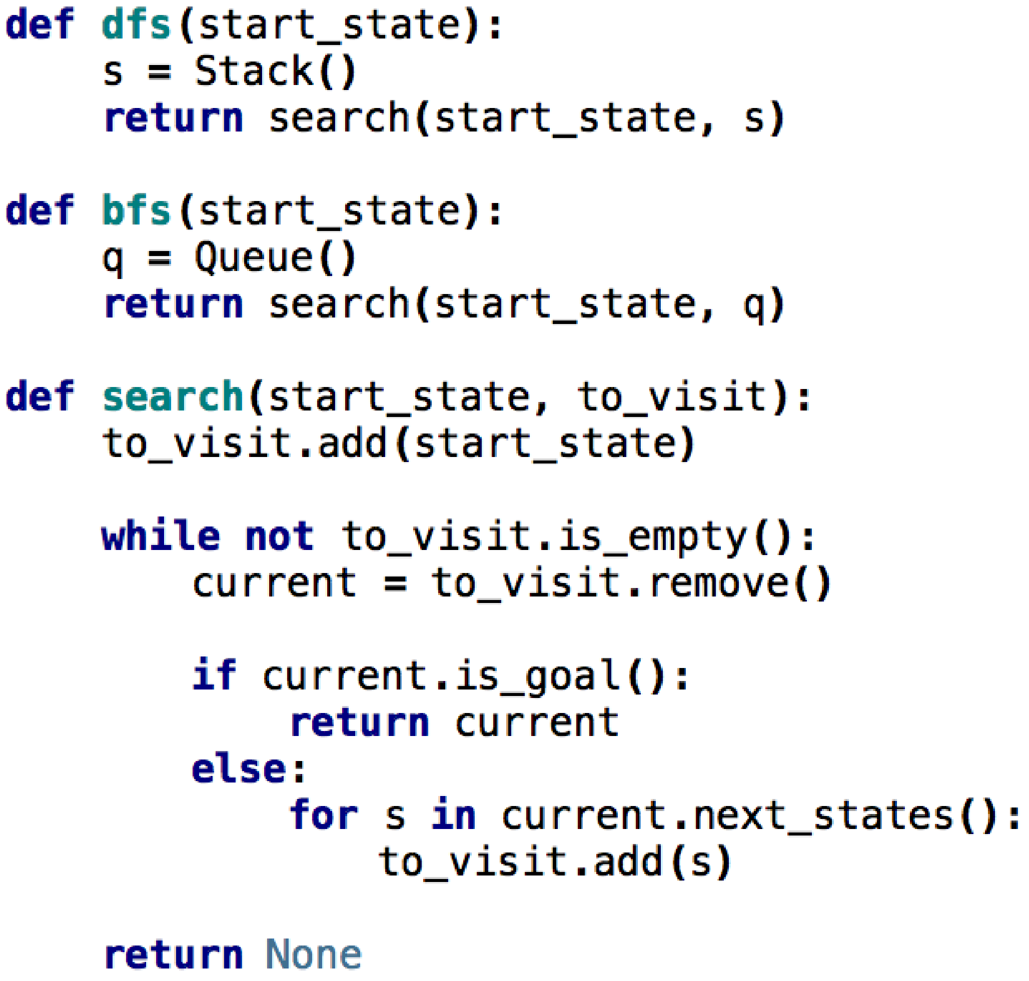
Depth first search (DFS): to_visit is a stack
Breadth first search (BFS): to_visit is a queue

Search variants implemented

add the start state to to_visit

Repeat

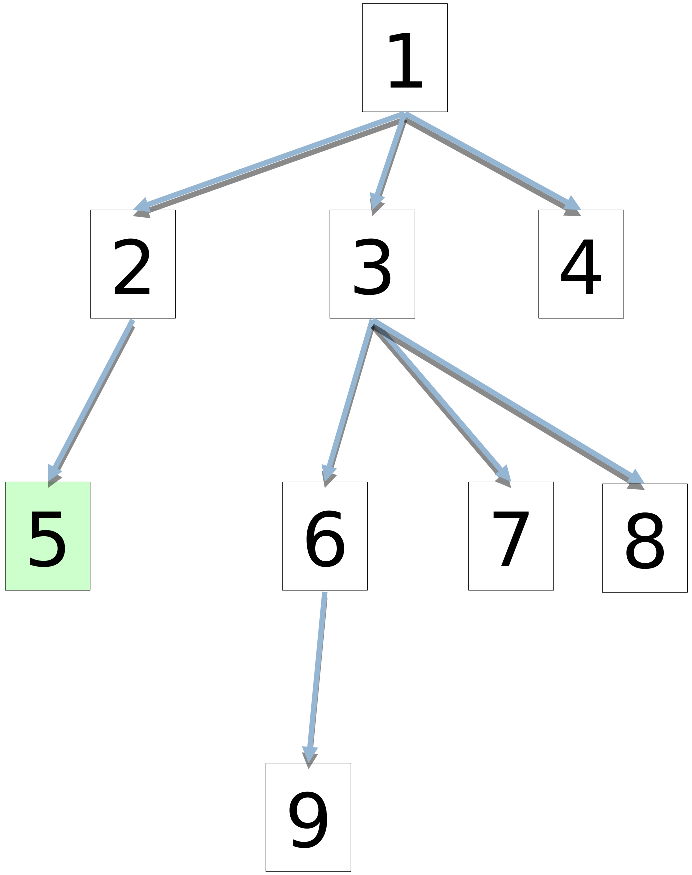
- take a state off the to_visit list
- if it’s the goal state
  - we’re done!
- if it’s not the goal state
  - Add all of the successive states to the to_visit list



```
def dfs(start_state):
    s = Stack()
    return search(start_state, s)

def bfs(start_state):
    q = Queue()
    return search(start_state, q)

def search(start_state, to_visit):
    to_visit.add(start_state)

    while not to_visit.is_empty():
        current = to_visit.remove()

        if current.is_goal():
            return current
        else:
            for s in current.next_states():
                to_visit.add(s)

    return None
```

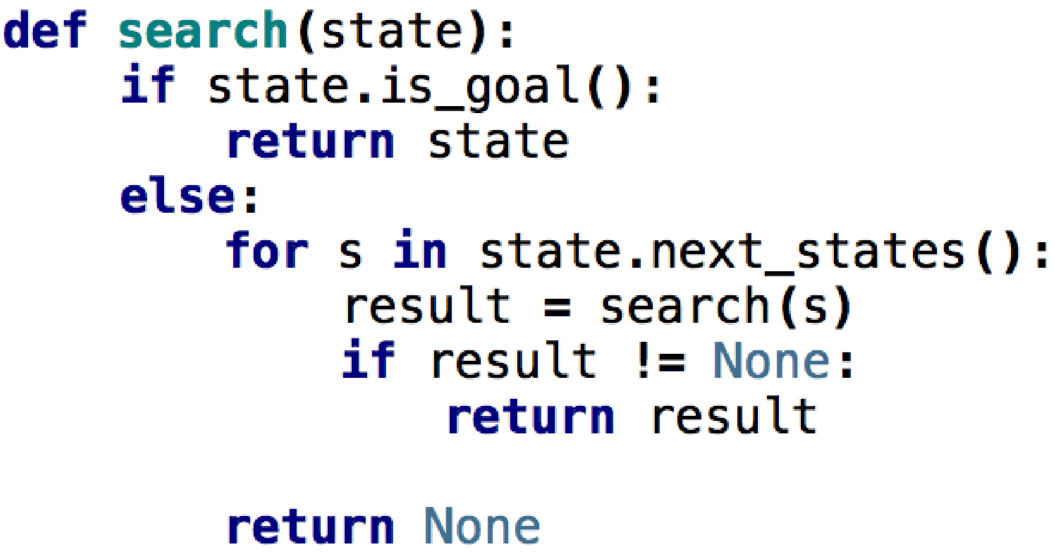
What order would this variant visit the states?

```
def search(state):
    if state.is_goal():
        return state
    else:
        for s in state.next_states():
            result = search(s)
            if result != None:
                return result

        return None
```

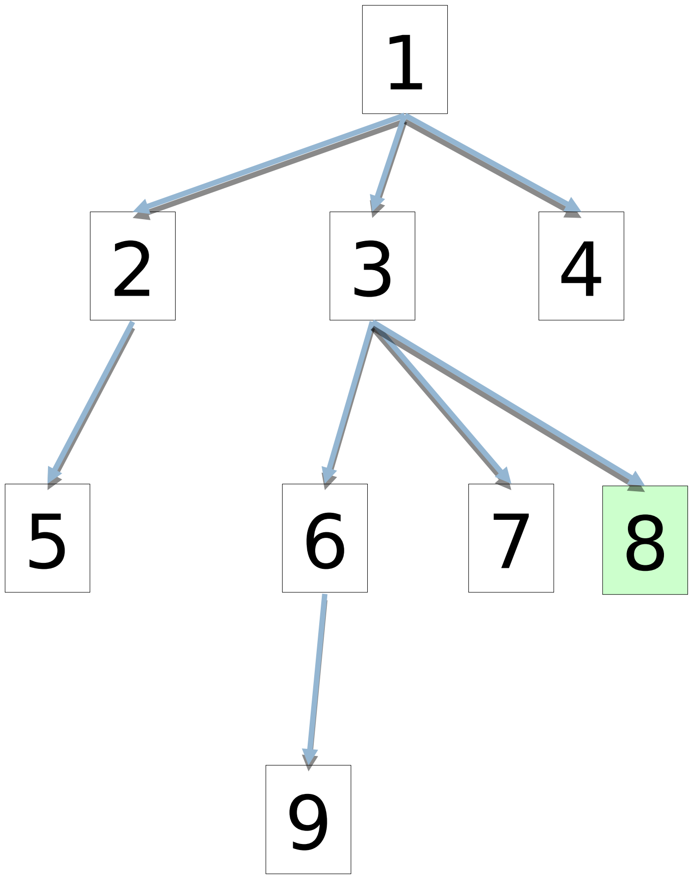


1, 2, 5

What order would this variant visit the states?



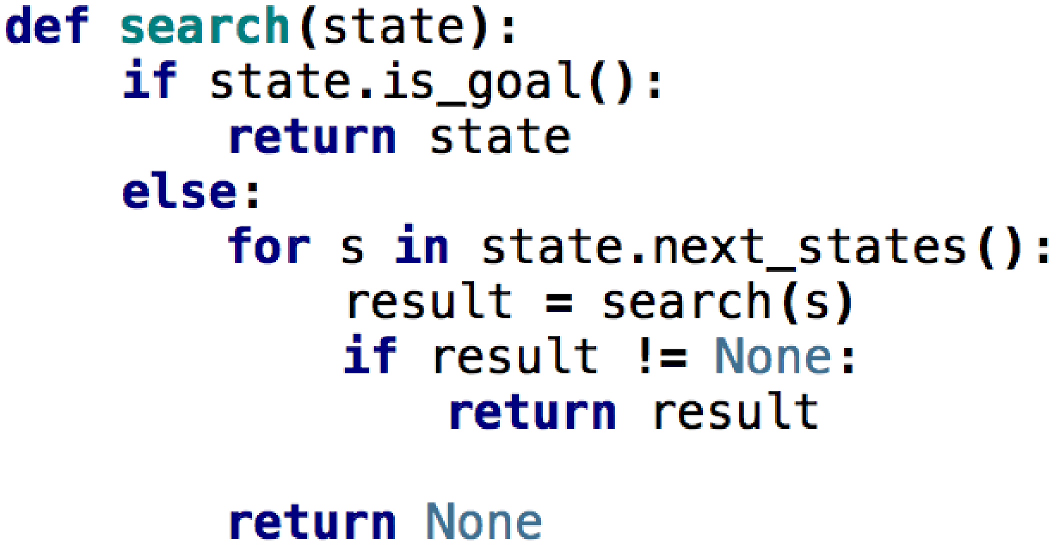
```
def search(state):
    if state.is_goal():
        return state
    else:
        for s in state.next_states():
            result = search(s)
            if result != None:
                return result

        return None
```



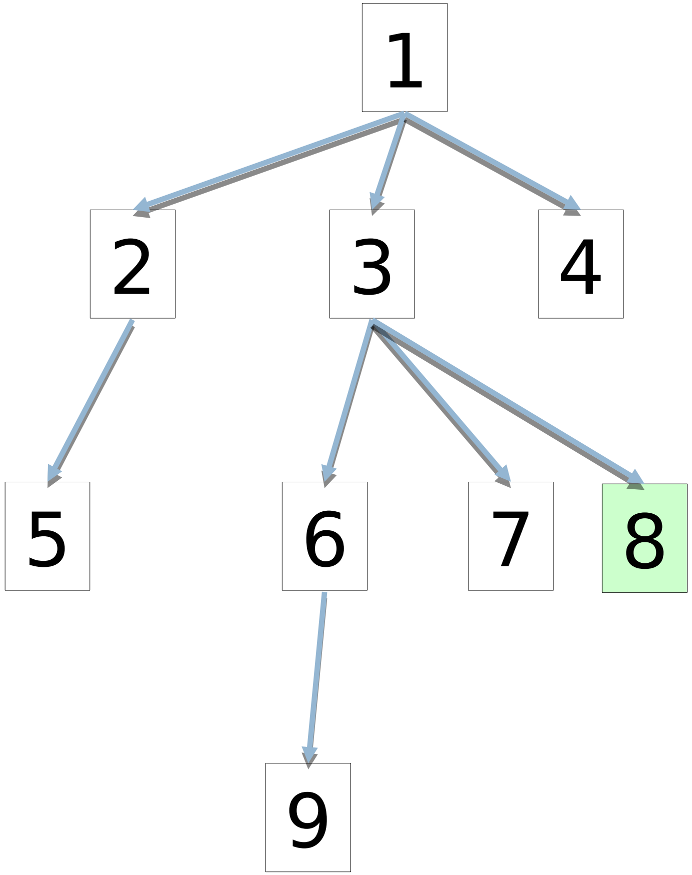
1, 2, 5, 3, 6, 9, 7, 8

What search algorithm is this?

What order would this variant visit the states?



```
def search(state):
    if state.is_goal():
        return state
    else:
        for s in state.next_states():
            result = search(s)
            if result != None:
                return result

        return None
```



1, 2, 5, 3, 6, 9, 7, 8

DFS!