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!

Admin
Assignment 8
Solve the maze!

How did you figure it out?

One approach

What now?
One approach

Three choices

Pick one!
What now?

Still three options!
Which would you explore/pick?

Most people go down a single path until they realize that it’s wrong
One approach

Keep exploring

What now?

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

How many more states are there?
Now what?
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

How do we start?

Add start 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
  - 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

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
- 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

Dead-end. What do we do now?

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

- 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
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
- repeat

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

DFS: 1, 4, 3, 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

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

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

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?

**Depth first search (DFS):**
- DFS: 1, 4, 3, 8, 7, 6, 9, 2, 5
- to_visit is a stack

**Breadth first search (BFS):**
- BFS: 1, 2, 3, 4, 5
- to_visit is a queue

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

BFS:
```
1, 2, 3, 4, 5
```

**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

**Defining functions**

```python
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)

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

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 is None:
                return result
            return
```

```
1, 2, 3, 4, 5
```

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

```
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)
```

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

```
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)
```

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

```
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)
```

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

What order would this variant visit the states?

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

What search algorithm is this?

DFS!