INFORMED SEARCH

David Kauchak
CS51A – Spring 2019

Admin

Assignment 9

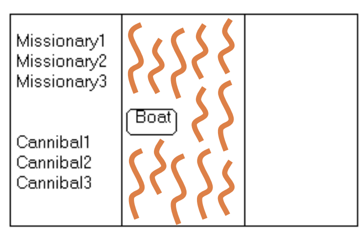
Missionaries and Cannibals

Three missionaries and three cannibals wish to cross the river. They have a small boat that will carry up to two people. Everyone can navigate the boat. If at any time the Cannibals outnumber the Missionaries on either bank of the river, they will eat the Missionaries. Find the smallest number of crossings that will allow everyone to cross the river safely.

What is the "state" of this problem (it should capture all possible valid configurations)?

Missionaries and Cannibals

Three missionaries and three cannibals wish to cross the river. They have a small boat that will carry up to two people. Everyone can navigate the boat. If at any time the Cannibals outnumber the Missionaries on either bank of the river, they will eat the Missionaries. Find the smallest number of crossings that will allow everyone to cross the river safely.



Missionaries and Cannibals

Three missionaries and three cannibals wish to cross the river. They have a small boat that will carry up to two people. Everyone can navigate the boat. If at any time the Cannibals outnumber the Missionaries on either bank of the river, they will eat the Missionaries. Find the smallest number of crossings that will allow everyone to cross the river safely.

MMMCCC B

MMCC B MC

MC B MMCC

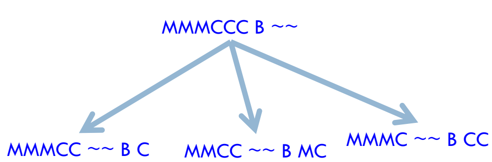
…

Searching for a solution

MMMCCC B ~~

What states can we get to from here?

Searching for a solution



Next states?

Code!

http://www.cs.pomona.edu/~dkauchak/classes/cs30/examples/cannibals.txt

Missionaries and Cannibals Solution

	Near side		Far side
0 Initial setup:	MMMCCC	B	-
1 Two cannibals cross over:	MMMC	B	CC
2 One comes back:	MMMCC	B	C
3 Two cannibals go over again:	MMM	B	CCC
4 One comes back:	MMMC	B	CC
5 Two missionaries cross:	MC	B	MMCC
6 A missionary & cannibal return:	MMCC	B	MC
7 Two missionaries cross again:	CC	B	MMMC
8 A cannibal returns:	CCC	B	MMM
9 Two cannibals cross:	C	B	MMMCC
10 One returns:	CC	B	MMMC
11 And brings over the third:	-	B	MMMCCC

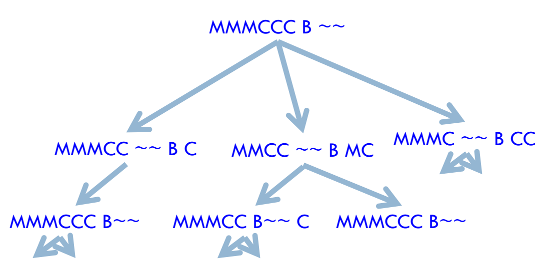
How is this solution different than the n-queens problem?

Missionaries and Cannibals Solution

	Near side		Far side
0 Initial setup:	MMMCCC	B	-
1 Two cannibals cross over:	MMMC	B	CC
2 One comes back:	MMMCC	B	C
3 Two cannibals go over again:	MMM	B	CCC
4 One comes back:	MMMC	B	CC
5 Two missionaries cross:	MC	B	MMCC
6 A missionary & cannibal return:	MMCC	B	MC
7 Two missionaries cross again:	CC	B	MMMC
8 A cannibal returns:	CCC	B	MMM
9 Two cannibals cross:	C	B	MMMCC
10 One returns:	CC	B	MMMC
11 And brings over the third:	-	B	MMMCCC

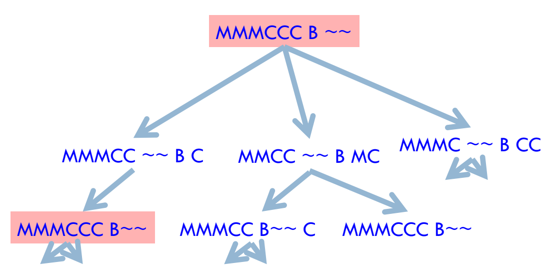
Solution is not a state, but a sequence of actions (or a sequence of states)

One other problem



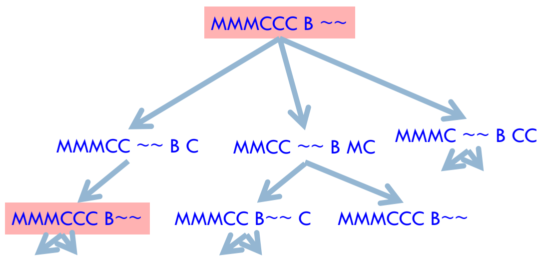
What would happen if we ran DFS here?

One other problem



If we always go left first, will continue forever!

One other problem

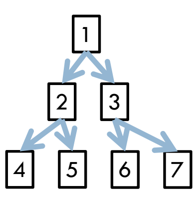


Does BFS have this problem? No!

DFS vs. BFS

Why do we use DFS then, and not BFS?

DFS vs. BFS

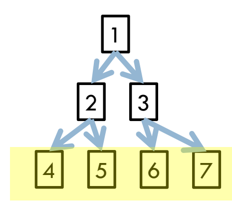


Consider a search problem where each state has two states you can reach

Assume the goal state involves 20 actions, i.e. moving between ~20 states

How big can the queue get for BFS?

DFS vs. BFS

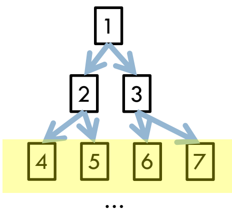


Consider a search problem where each state has two states you can reach

Assume the goal state involves 20 actions, i.e. moving between ~20 states

At any point, need to remember roughly a "row"

DFS vs. BFS

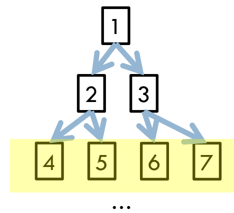


Consider a search problem where each state has two states you can reach

Assume the goal state involves 20 actions, i.e. moving between ~20 states

How big does this get?

DFS vs. BFS

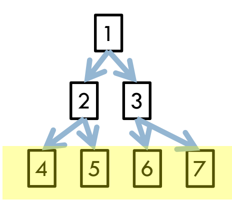


Consider a search problem where each state has two states you can reach

Assume the goal state involves 20 actions, i.e. moving between ~20 states

Doubles every level we have to go deeper.
For 20 actions that is 2^{20} = ~1 million states!

DFS vs. BFS

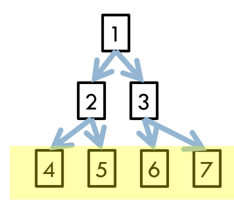


Consider a search problem where each state has two states you can reach

Assume the goal state involves 20 actions, i.e. moving between ~20 states

How many states would DFS keep on the stack?

DFS vs. BFS

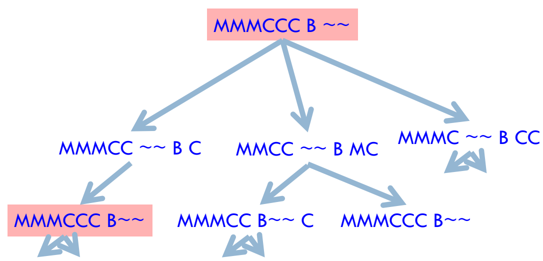


Consider a search problem where each state has two states you can reach

Assume the goal state involves 20 actions, i.e. moving between ~20 states

Only one path through the tree, roughly 20 states

One other problem



If we always go left first, will continue forever!

Solution?

DFS avoiding repeats

```
def dfs(state, visited):
    # note that we've visited this state
    visited[str(state)] = True

    if state.is_goal():
        return [state]
    else:
        result = []

        for s in state.next_states():
            # check if we've visited a state already
            if not(str(s) in visited):
                result += dfs(s, visited)

        return result
```

Other search problems

What problems have you seen that could be posed as search problems?

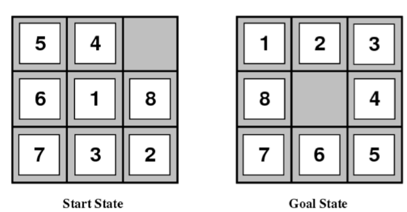
What is the state?

Start state

Goal state

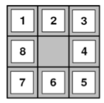
State-space/transition between states

8-puzzle



8-puzzle

goal


Goal State

state representation?

start state?

state-space/transitions?

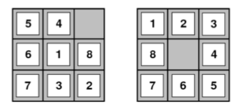
8-puzzle

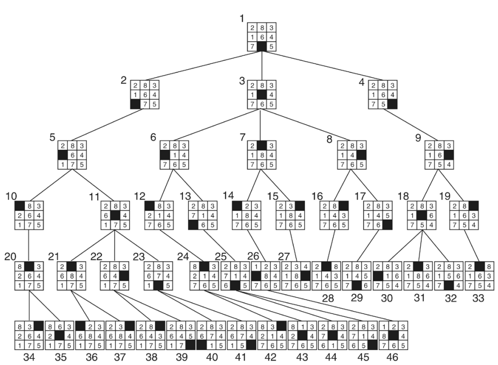
state:

- all 3 x 3 configurations of the tiles on the board

transitions between states:

- Move Blank Square Left, Right, Up or Down.
- This is a more efficient encoding than moving each of the 8 distinct tiles


Start State Goal State



Cryptarithmetic

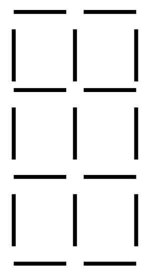
Find an assignment of digits (0, ..., 9) to letters so that a given arithmetic expression is true. examples:

SEND + MORE = MONEY

```
  FORTY
+  TEN
+  TEN
  -----
  SIXTY
F=2, O=9, R=7, etc.
```

Remove 5 Sticks

Given the following configuration of sticks, remove exactly 5 sticks in such a way that the remaining configuration forms exactly 3 squares.



Water Jug Problem

Given a full 5-gallon jug and a full 2-gallon jug, fill the 2-gallon jug with exactly one gallon of water.

Water Jug Problem

State = (x,y), where x is the number of gallons of water in the 5-gallon jug and y is # of gallons in the 2-gallon jug

Initial State = (5,2)

Goal State = (*,1), where * means any amount

Operator table

Name	Cond.	Transition	Effect
Empty5	–	(x,y)→(0,y)	Empty 5-gal. jug
Empty2	–	(x,y)→(x,0)	Empty 2-gal. jug
2to5	$x \le 3$	(x,2)→(x+2,0)	Pour 2-gal. into 5-gal.
5to2	$x \ge 2$	(x,0)→(x-2,2)	Pour 5-gal. into 2-gal.
5to2part	$y < 2$	(1,y)→(0,y+1)	Pour partial 5-gal. into 2-gal.

8-puzzle revisited

How hard is this problem?

1	3	8
4		7
6	5	2

8-puzzle revisited

The average depth of a solution for an 8-puzzle is 22 moves

An exhaustive search requires searching ~3^{22} = 3.1 x 10^{10} states

- BFS: 10 terabytes of memory
- DFS: 8 hours (assuming one million nodes/second)

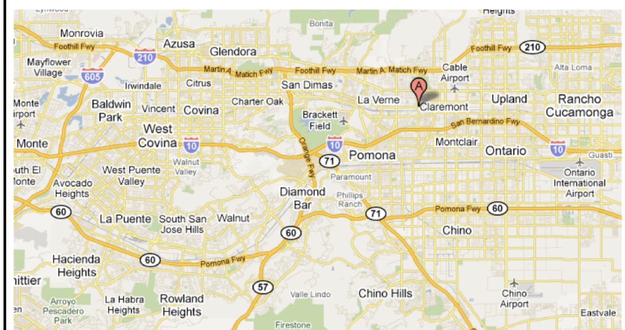
Can we do better?

Is DFS and BFS intelligent?

1	3	8
4		7
6	5	2

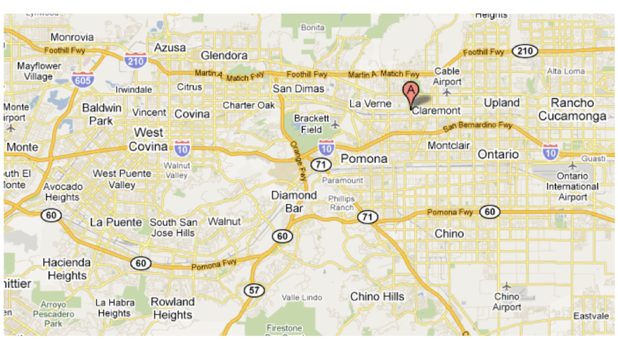
from: Claremont to:Rowland Heights

How do you think google maps does it?



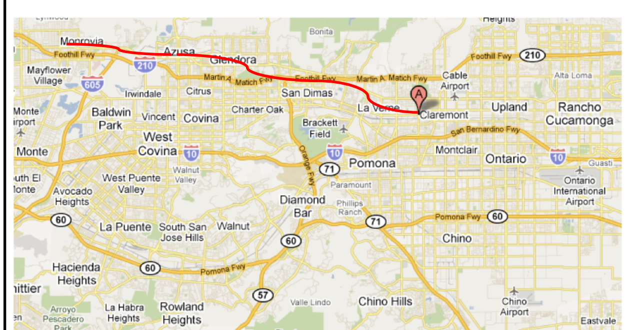
from: Claremont to:Rowland Heights

What would the search algorithms do?



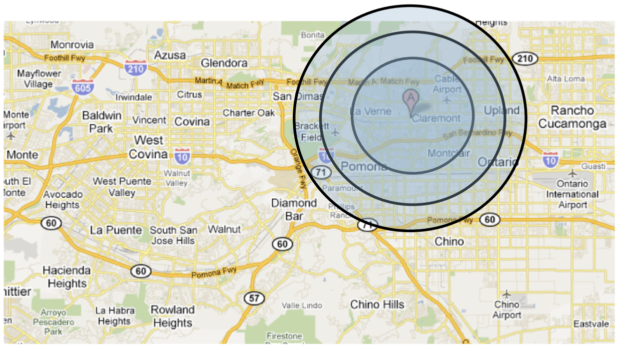
from: Claremont to:Rowland Heights

DFS



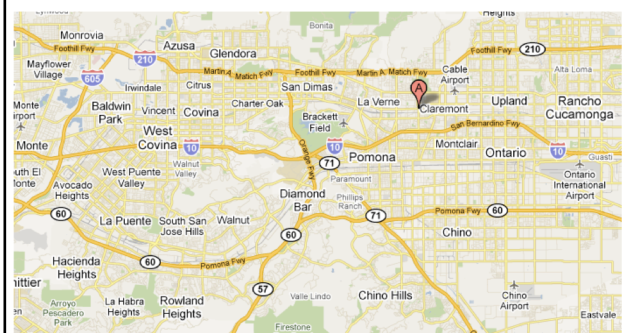
from: Claremont to:Rowland Heights

BFS



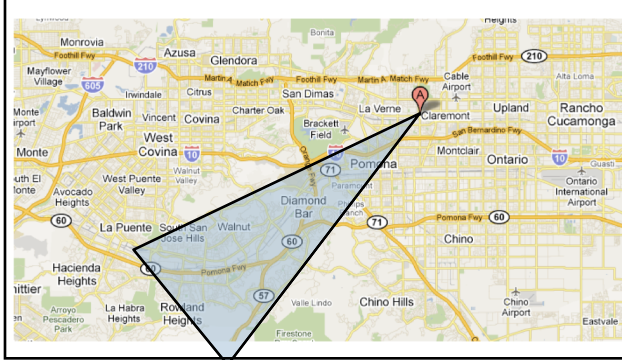
from: Claremont to: Rowland Heights

Ideas?



from: Claremont to: Rowland Heights

We'd like to bias search towards the actual solution



Informed search

Order to_visit based on some knowledge of the world that estimates how "good" a state is

- *h(n)* is called an evaluation function

Best-first search

- rank to_visit based on *h(n)*
- take the most desirable state in to_visit first
- different approaches depending on how we define *h(n)*

Heuristic

Merriam-Webster's Online Dictionary

Heuristic (pron. \hyu-ˈris-tik\): adj. [from Greek *heuriskein* to discover.] involving or serving as an aid to learning, discovery, or problem-solving by experimental and especially trial-and-error methods

The Free On-line Dictionary of Computing (2/19/13)

heuristic 1. Of or relating to a usually speculative formulation serving as a guide in the investigation or solution of a problem: "The historian discovers the past by the judicious use of such a heuristic device as the 'ideal type'" (Karl J. Weintraub).

Heuristic function: *h(n)*

An estimate of how close the node is to a goal

Uses domain-specific knowledge!

Examples

- Map path finding?
  - straight-line distance from the node to the goal ("as the crow flies")
- 8-puzzle?
  - how many tiles are out of place
  - sum of the "distances" of the out of place tiles
- Missionaries and cannibals?
  - number of people on the starting bank