Lecture 8
Uninformed search

CS 180 – Intelligent Systems

Dr. Victor Chen

Spring 2021

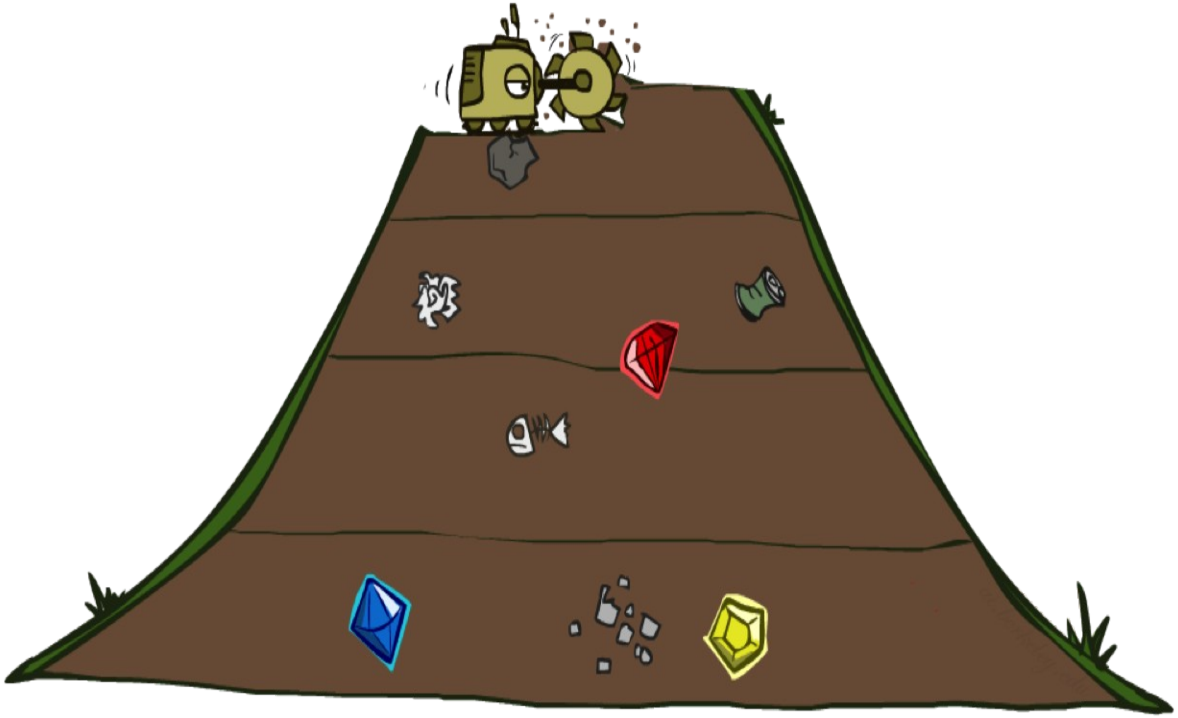
Uninformed Search

Uninformed search

In **uninformed search**, we use only the information available in the problem definition

- Breadth-first search
- Depth-first search
- Iterative deepening search
- Uniform-cost search

Breadth-First Search

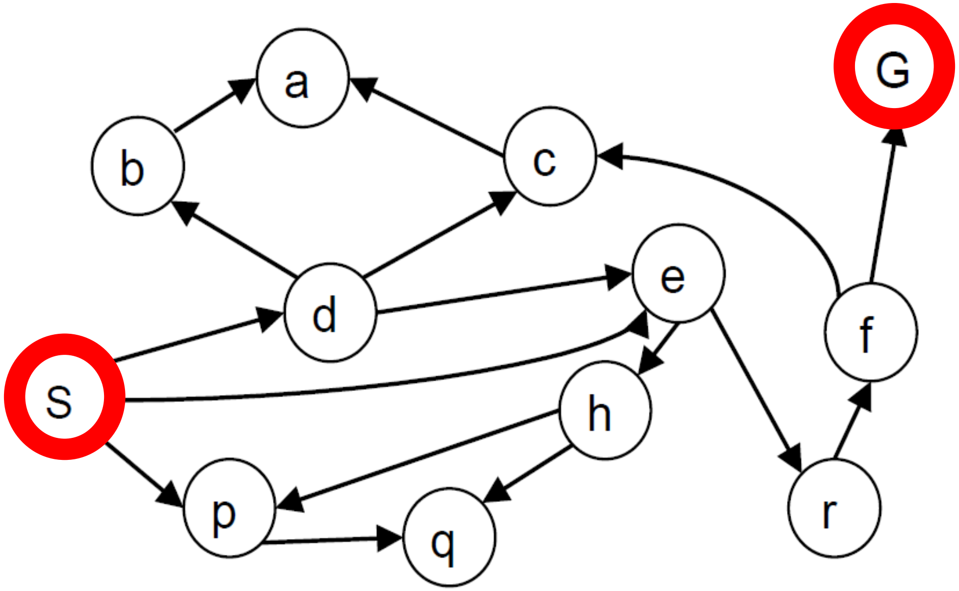


Breadth-First Search

Expand shallowest unexpanded node

- *Basic idea*: visit *all* your neighbors before your neighbor's neighbors
- Implementation: ***frontier* is a FIFO queue**

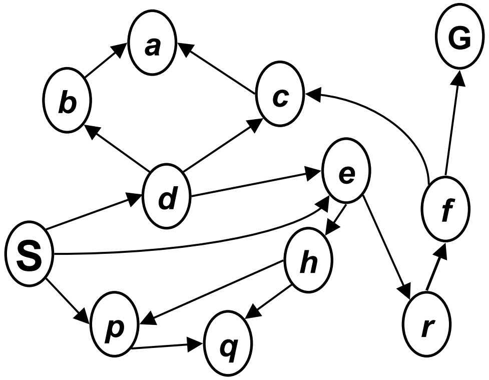
State space graph
for a search problem



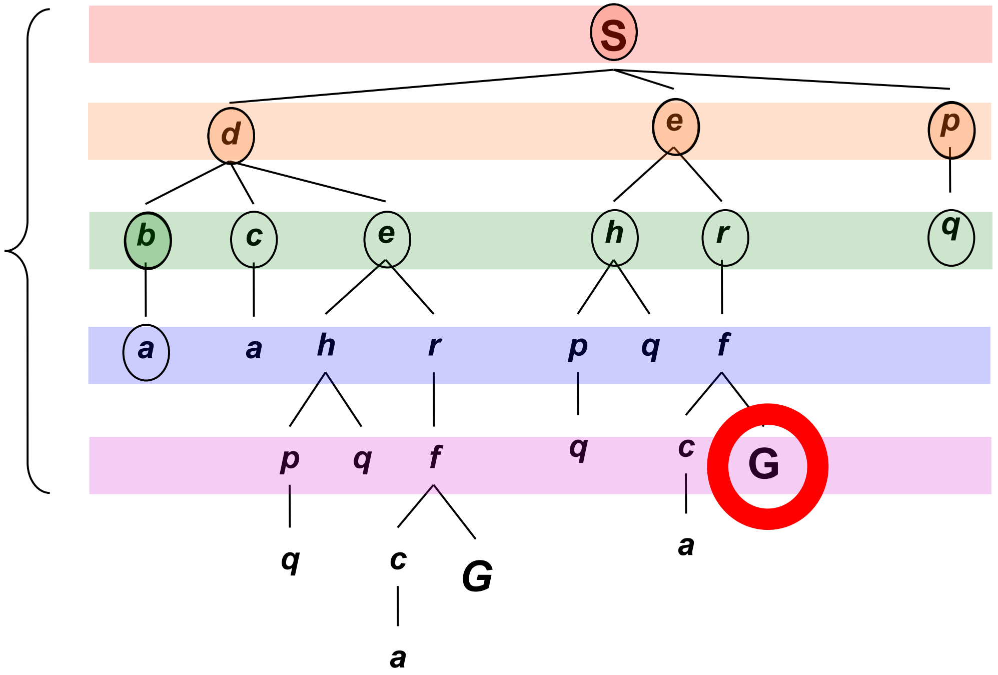
Breadth-First Search

***Strategy:* expand a shallowest node first**

Implementation:
Frontier is a FIFO queue



State space graph



Search tree

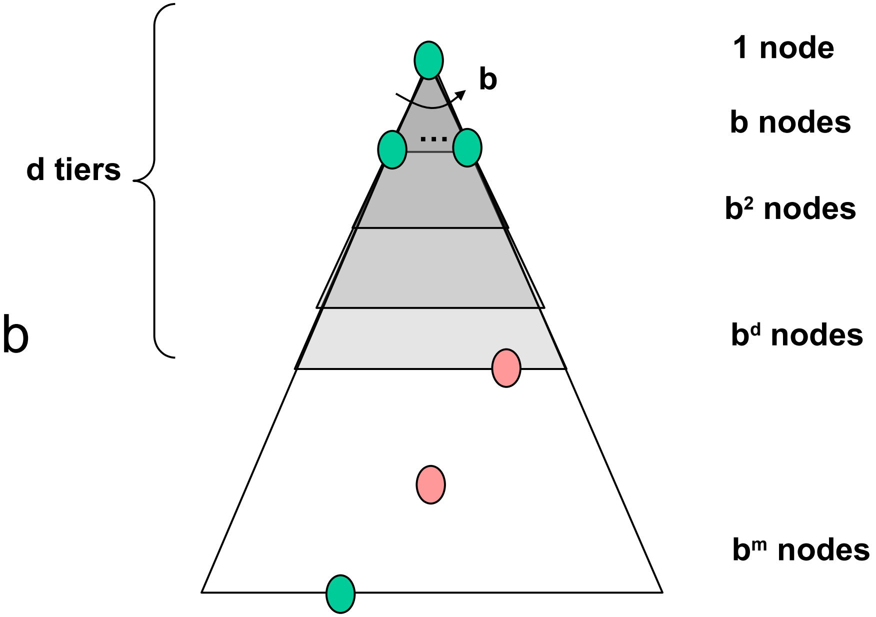
Breadth-First Search (BFS) Cost

What nodes does BFS expand?

- Expand all nodes above shallowest solution
- Let depth of shallowest solution be d
- Search takes time $O(b^d)$, where b is maximum branching factor of the search tree

How much space does the frontier take?

- Has roughly the last tier, so $O(b^d)$



Properties of breadth-first search

Complete?

If a solution exists, BFS will find it. So yes!

Optimal?

Yes, only if costs are all 1

Time cost?

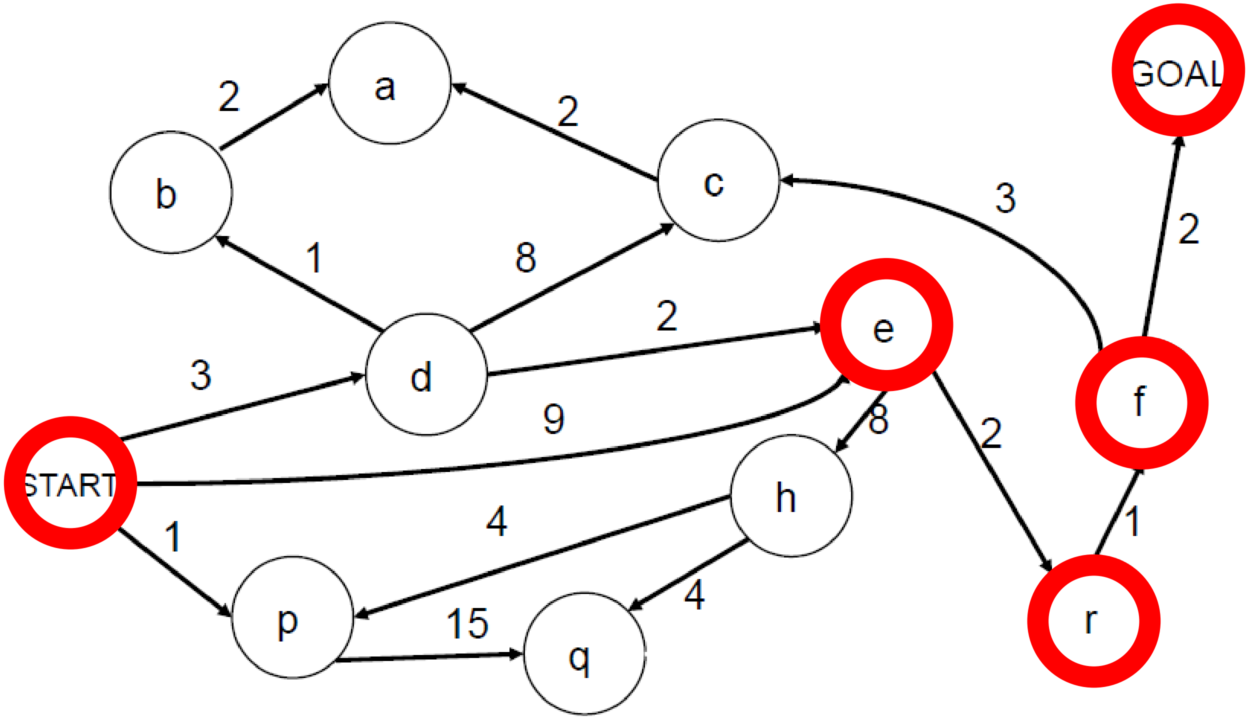
$O(b^d)$

(***b***: maximum branching factor and ***d*** is the depth of the optimal solution)

Space cost?

$O(b^d)$

BFS optimal? Yes but only if cost = 1 per step



BFS finds the solution with the fewest steps, but does not always find the shortest path (optimal solution)

Depth-first search

Depth-first search

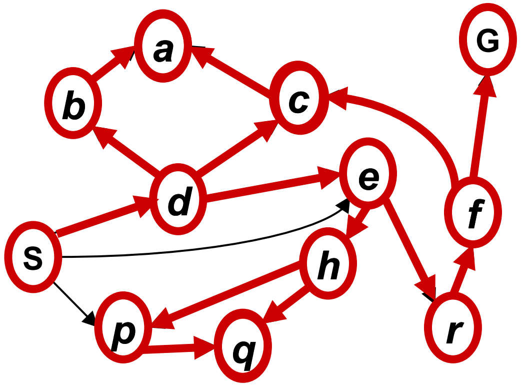
Expand deepest unexpanded node

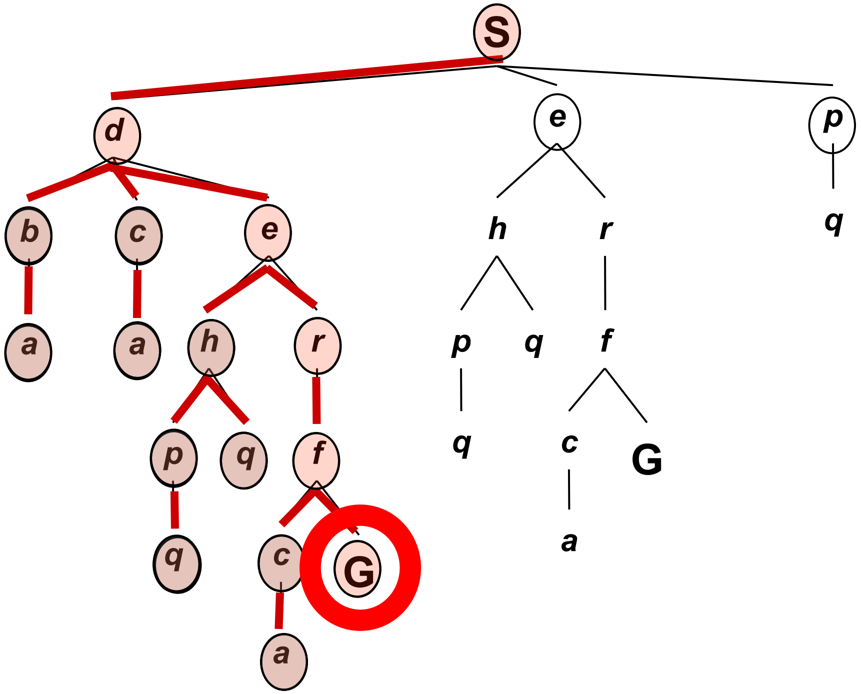
- Basic idea: visit nodes adjacent to the *last visited* node first
- Implementation: *frontier* is **a LIFO queue** (**a stack**)

Depth-First Search

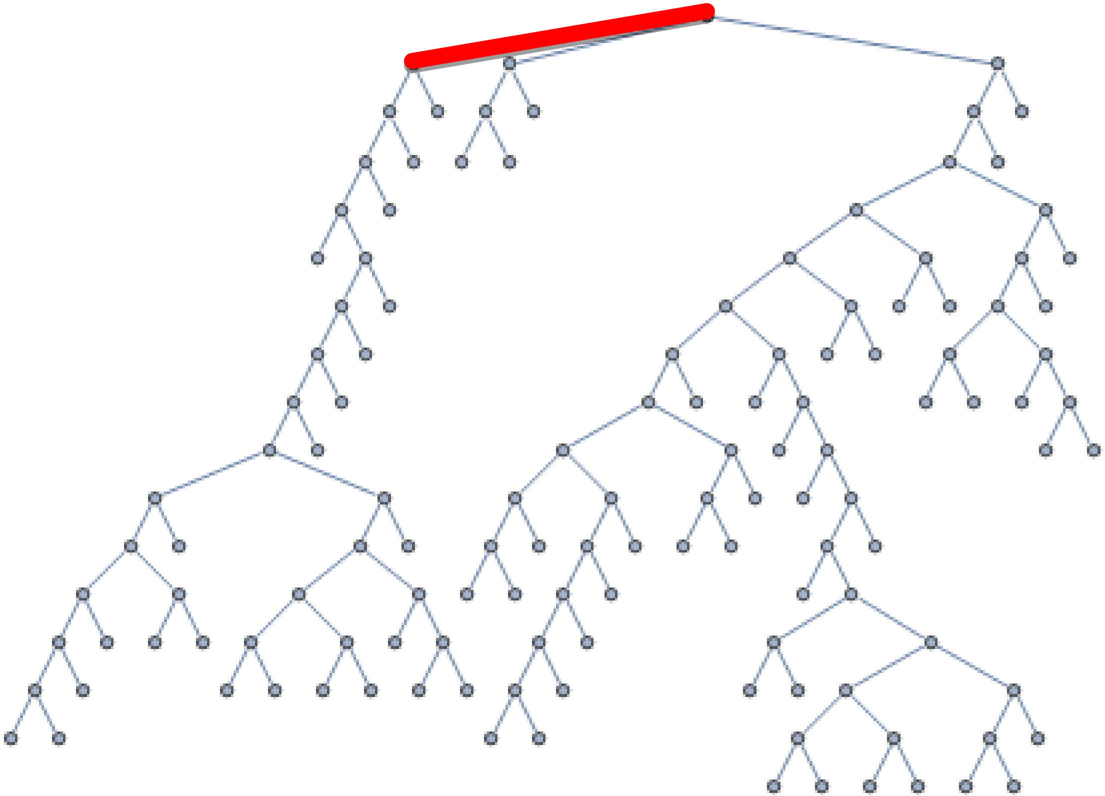
Strategy: expand a deepest node first

Implementation: Frontier is a LIFO stack

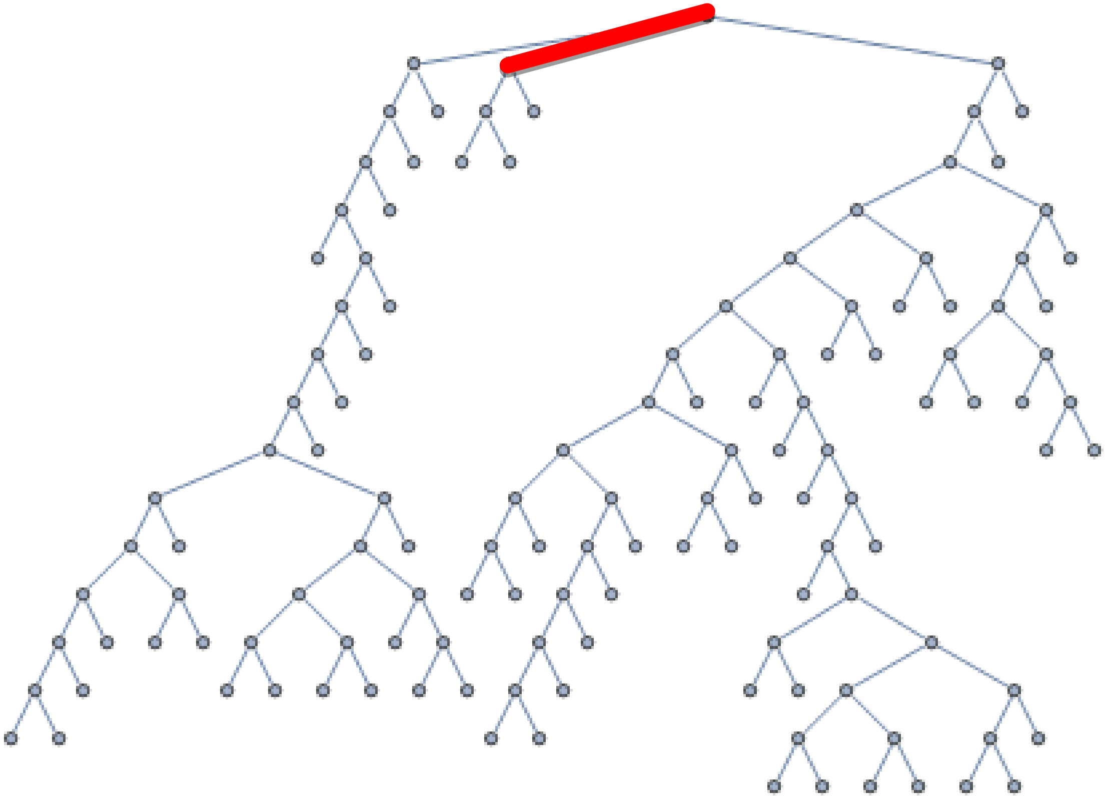




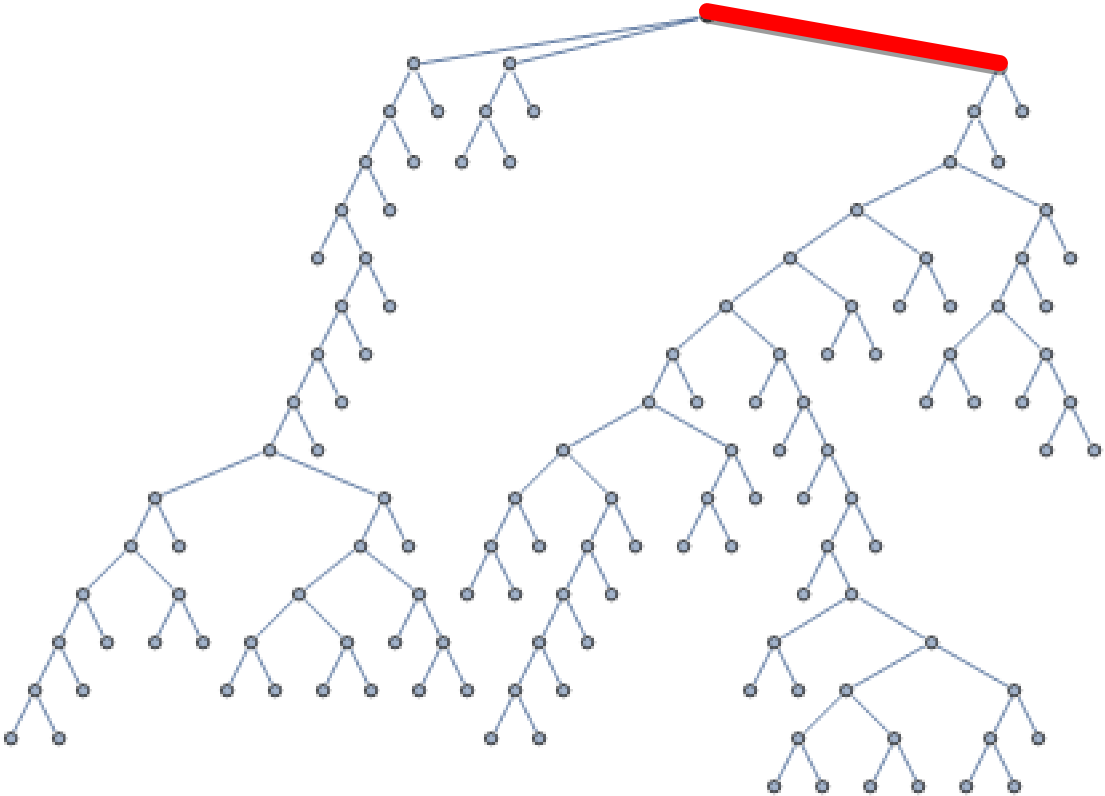
BFS vs. DFS



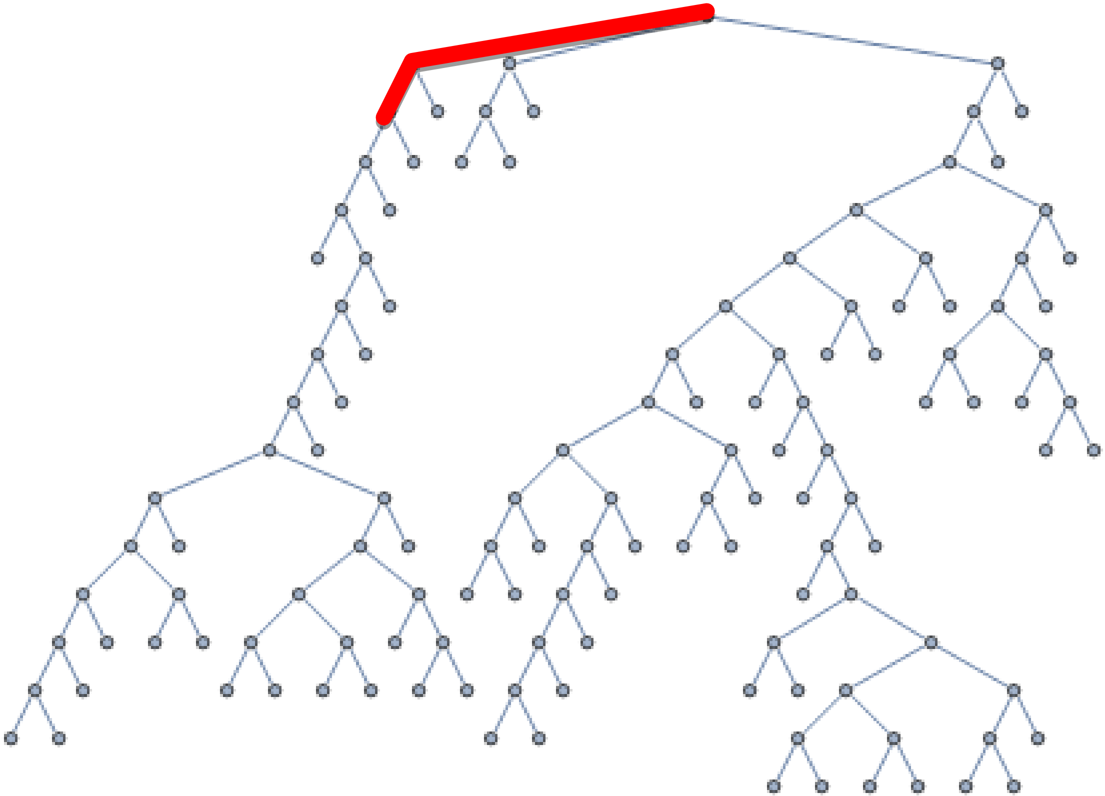
BFS vs. DFS



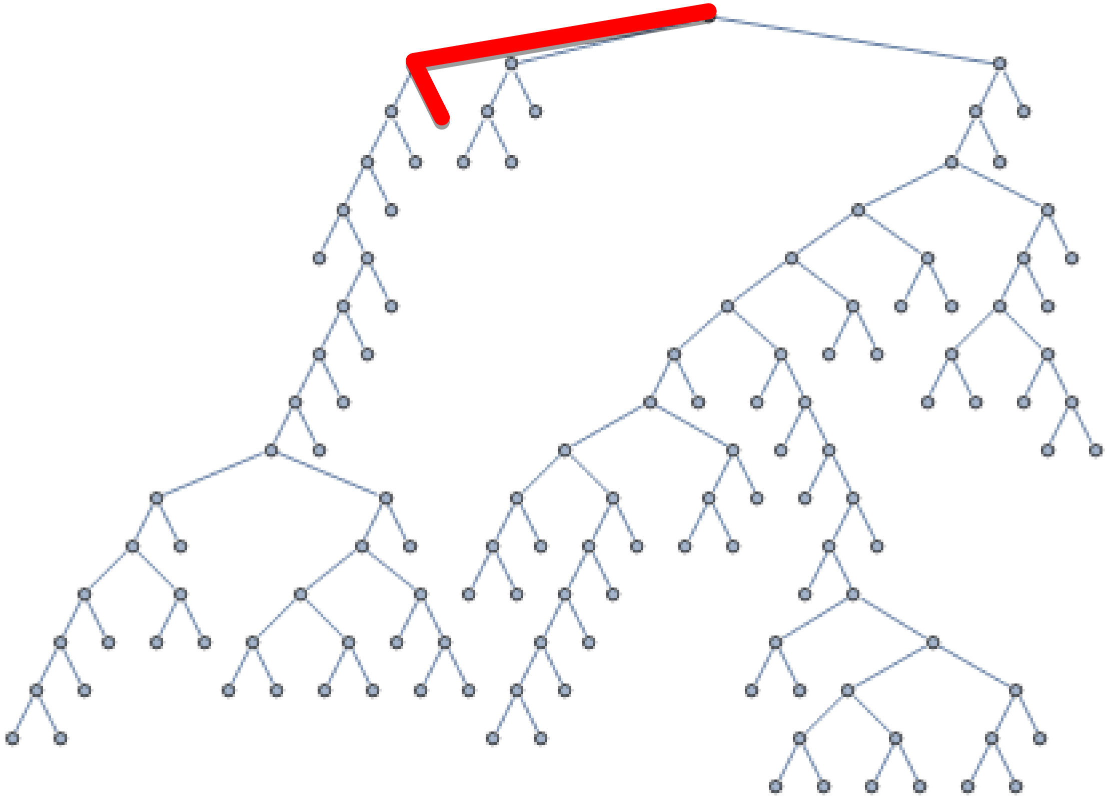
BFS vs. DFS



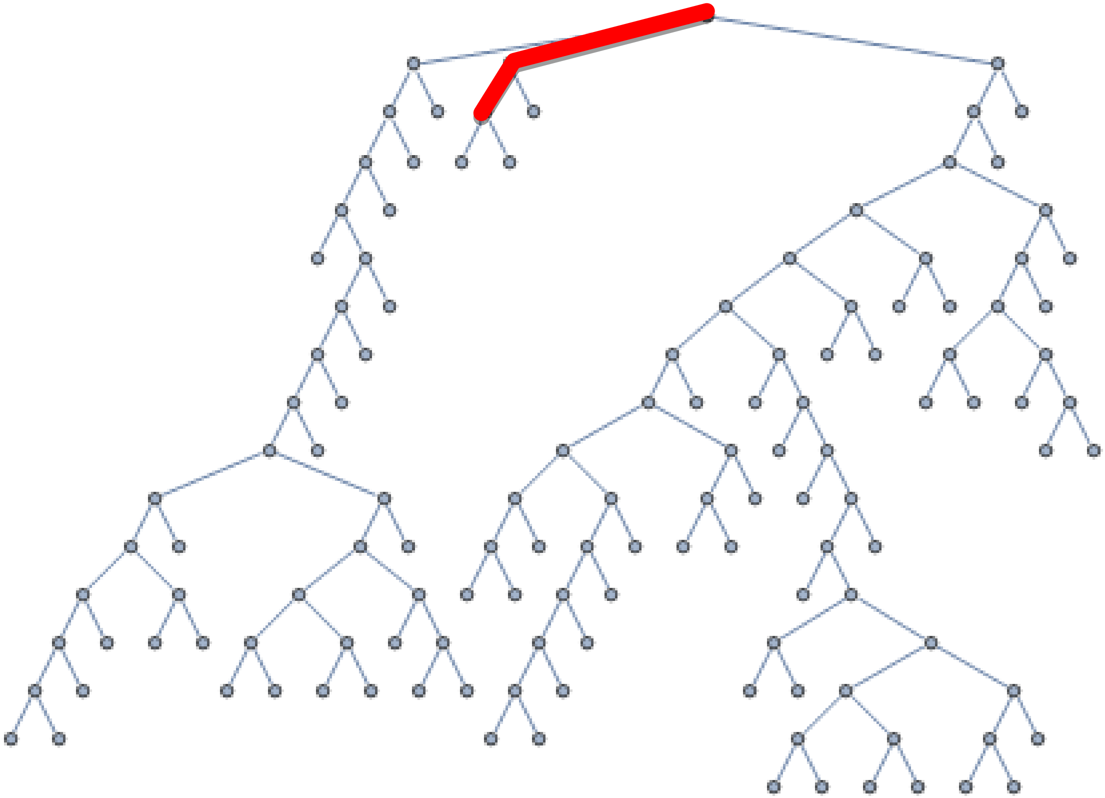
BFS vs. DFS



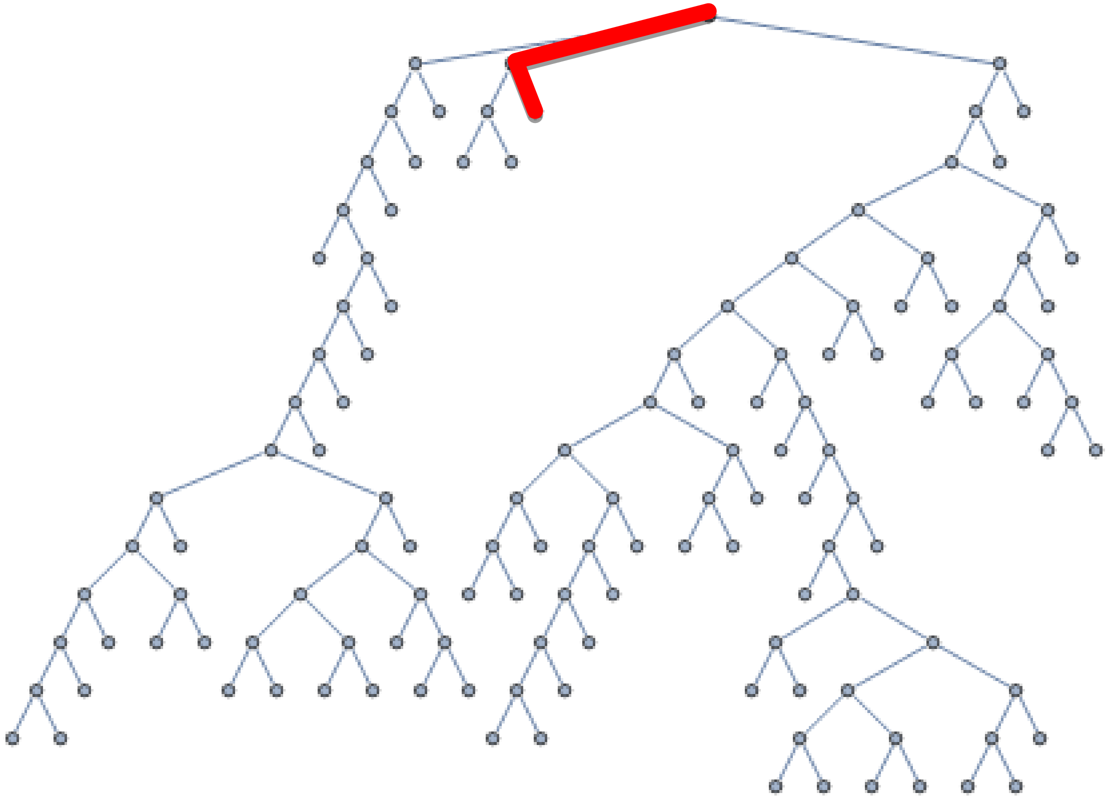
BFS vs. DFS



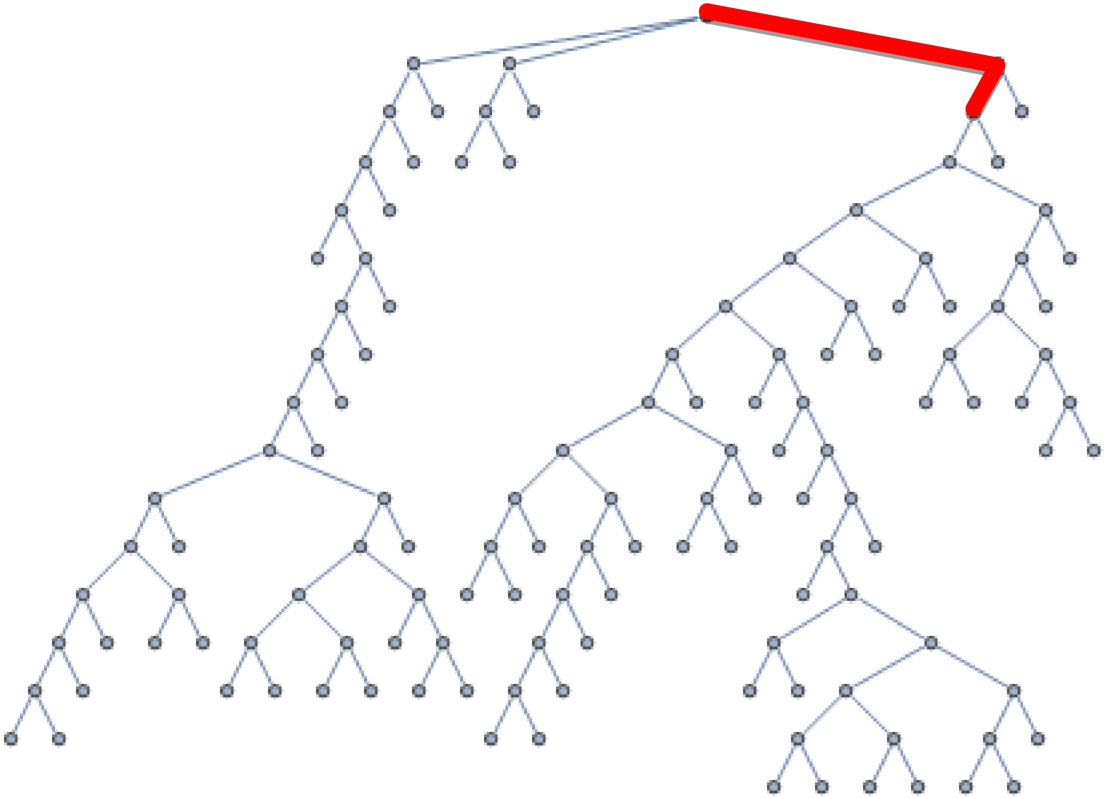
BFS vs. DFS



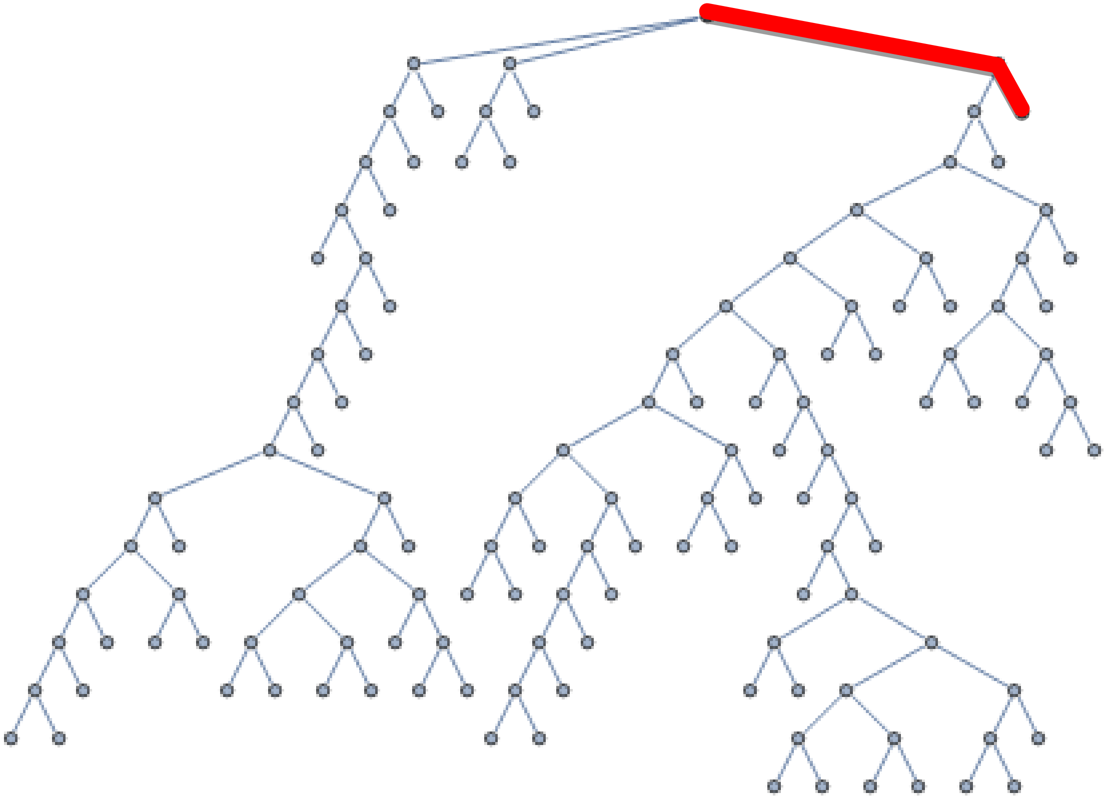
BFS vs. DFS



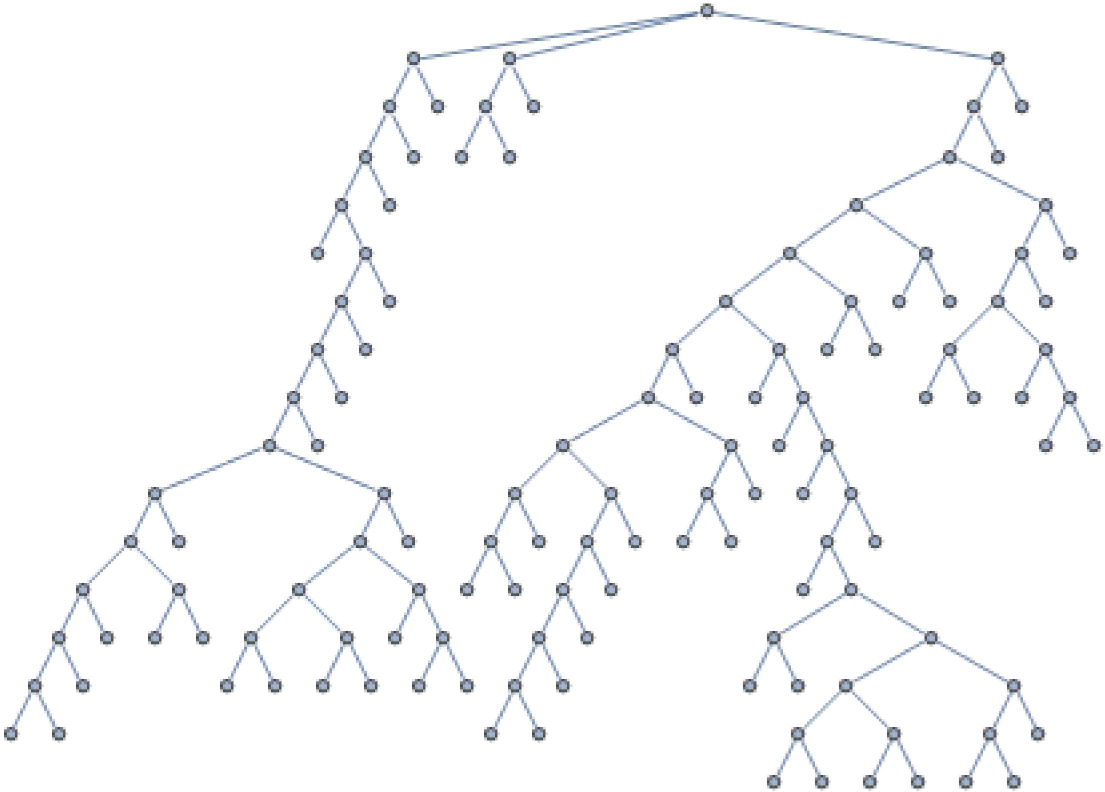
BFS vs. DFS



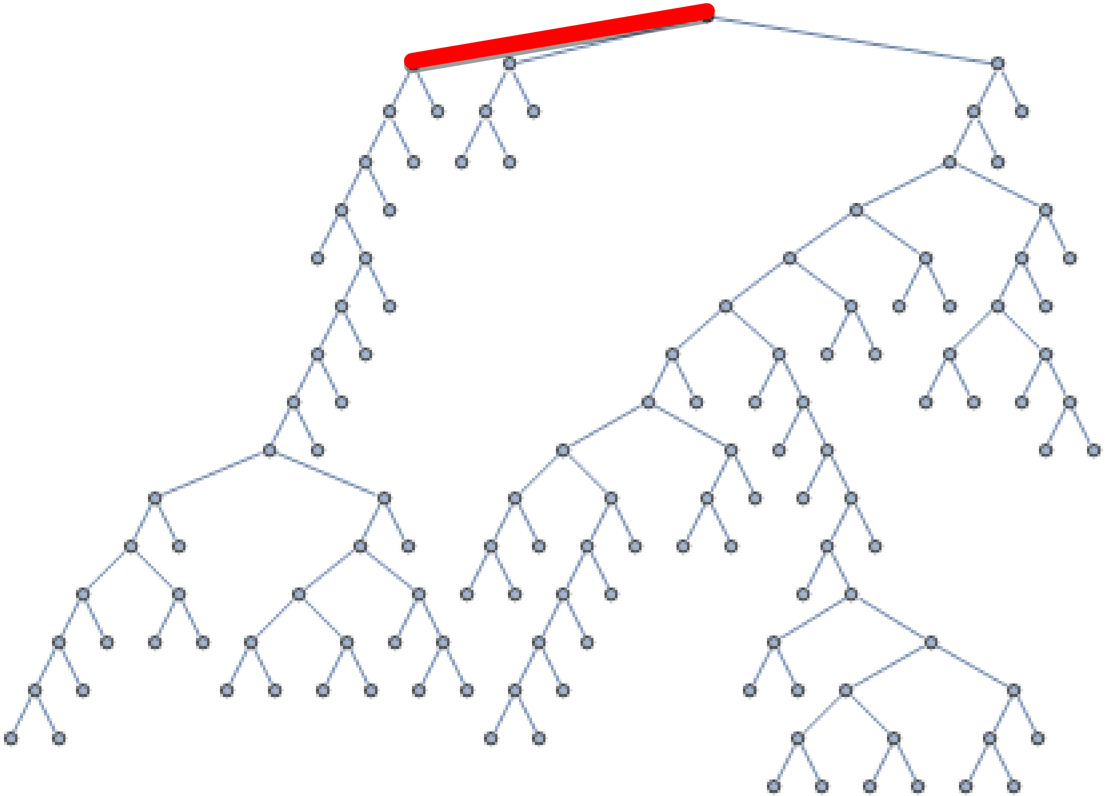
BFS vs. DFS



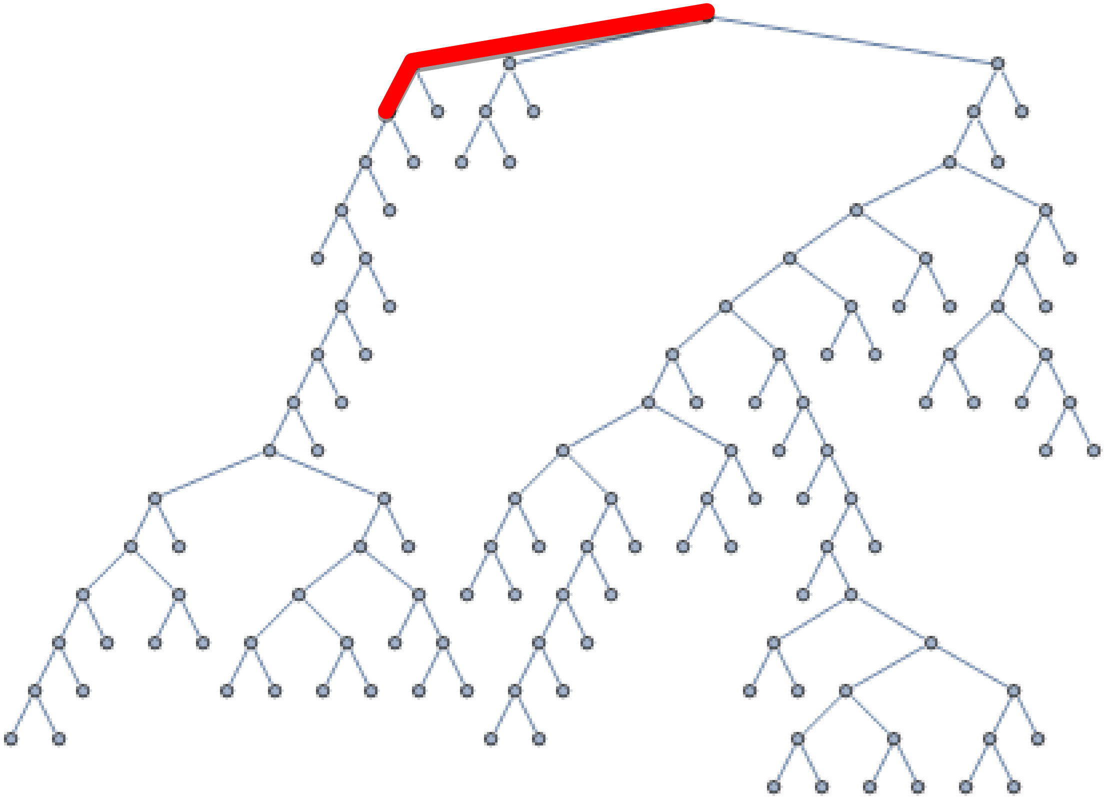
BFS vs. **DFS**



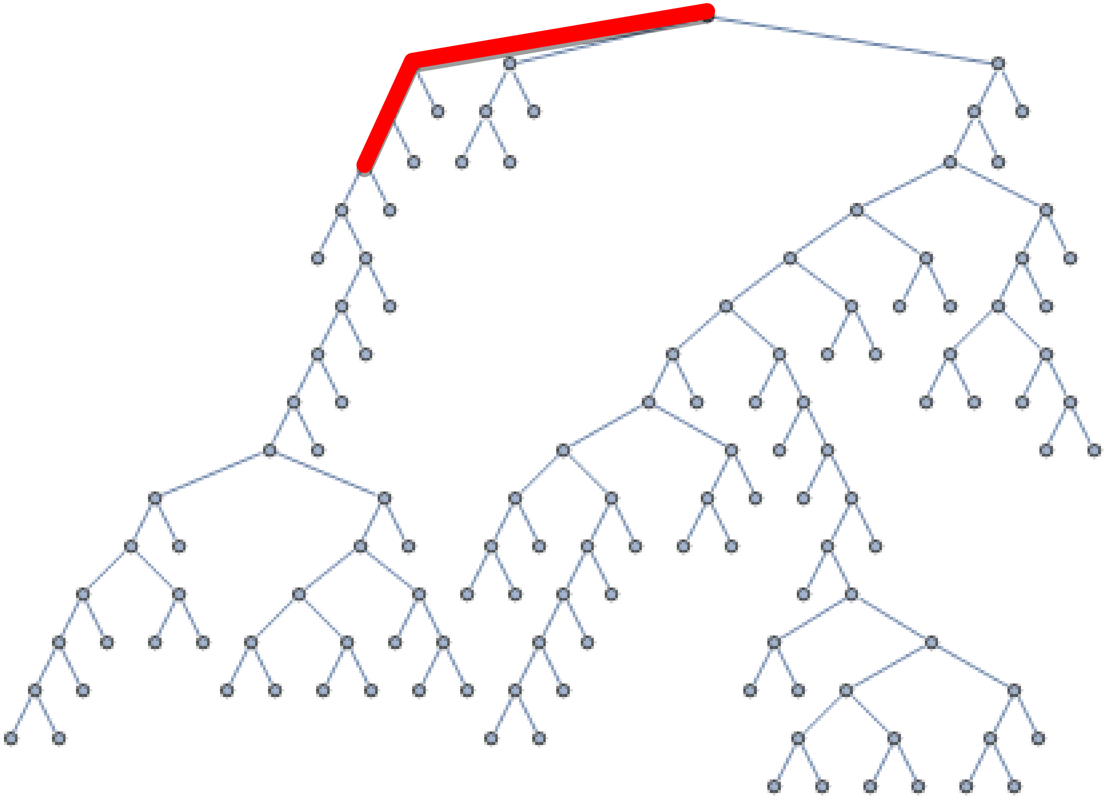
BFS vs. **DFS**



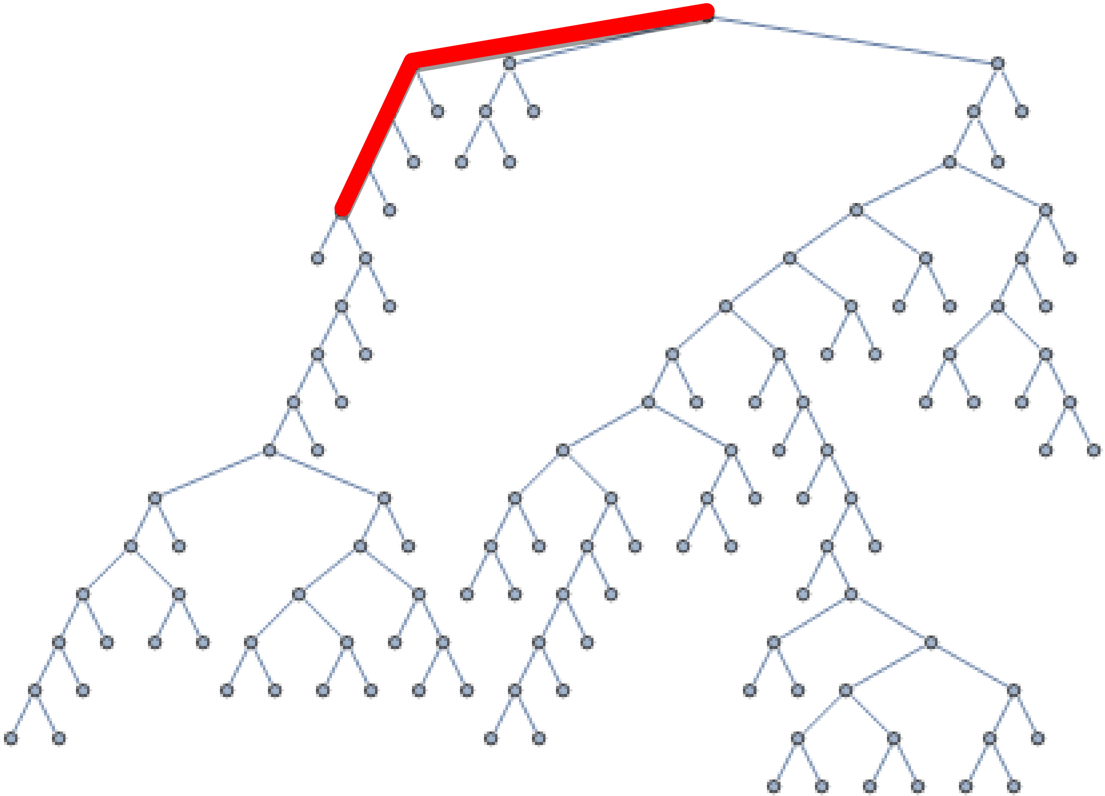
BFS vs. **DFS**



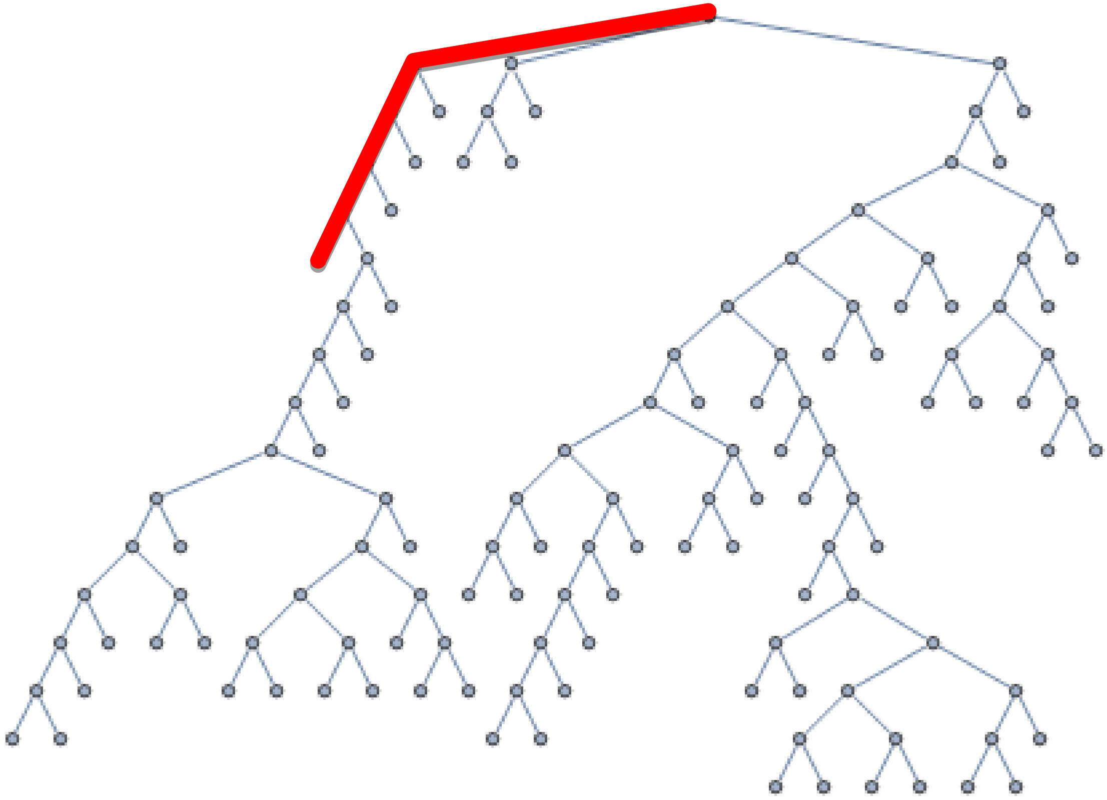
BFS vs. **DFS**



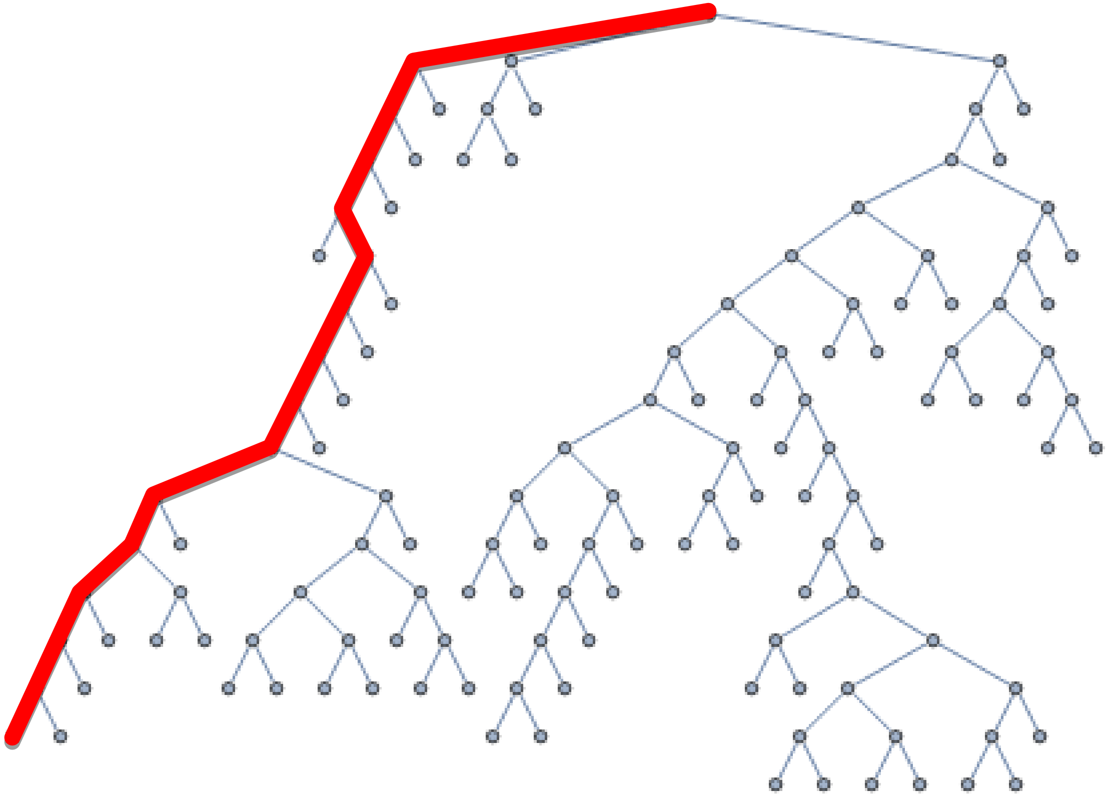
BFS vs. **DFS**



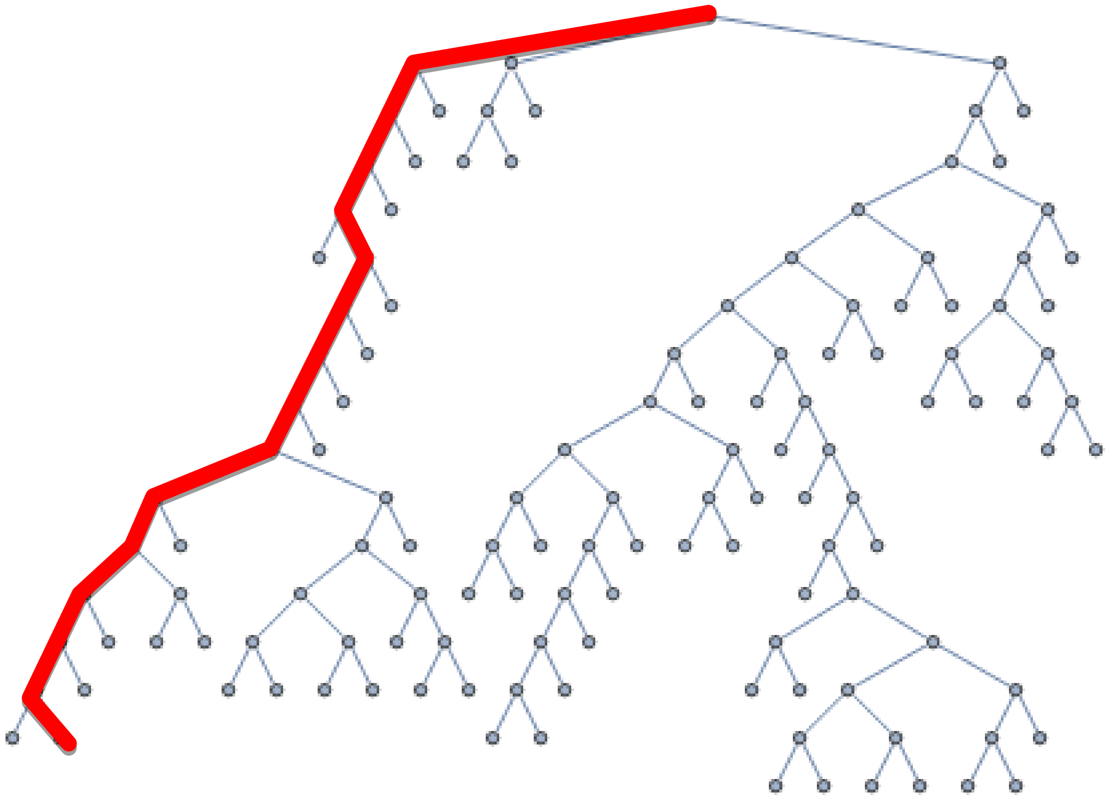
BFS vs. **DFS**



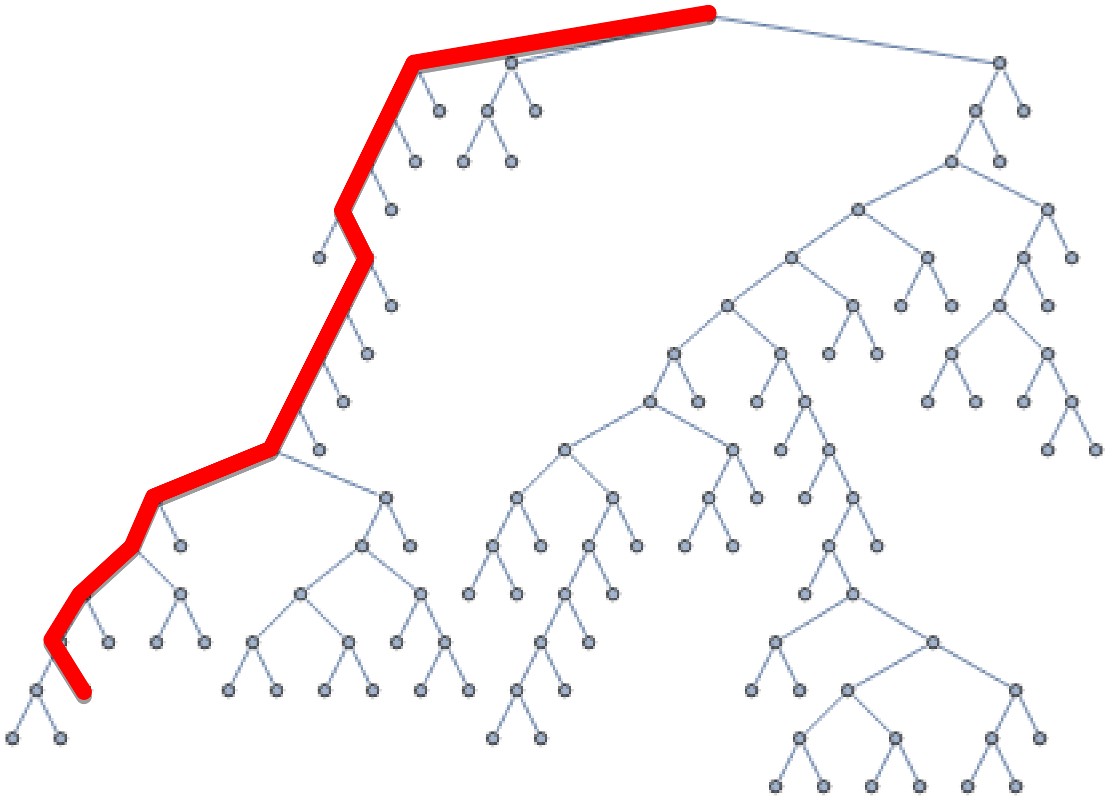
BFS vs. **DFS**



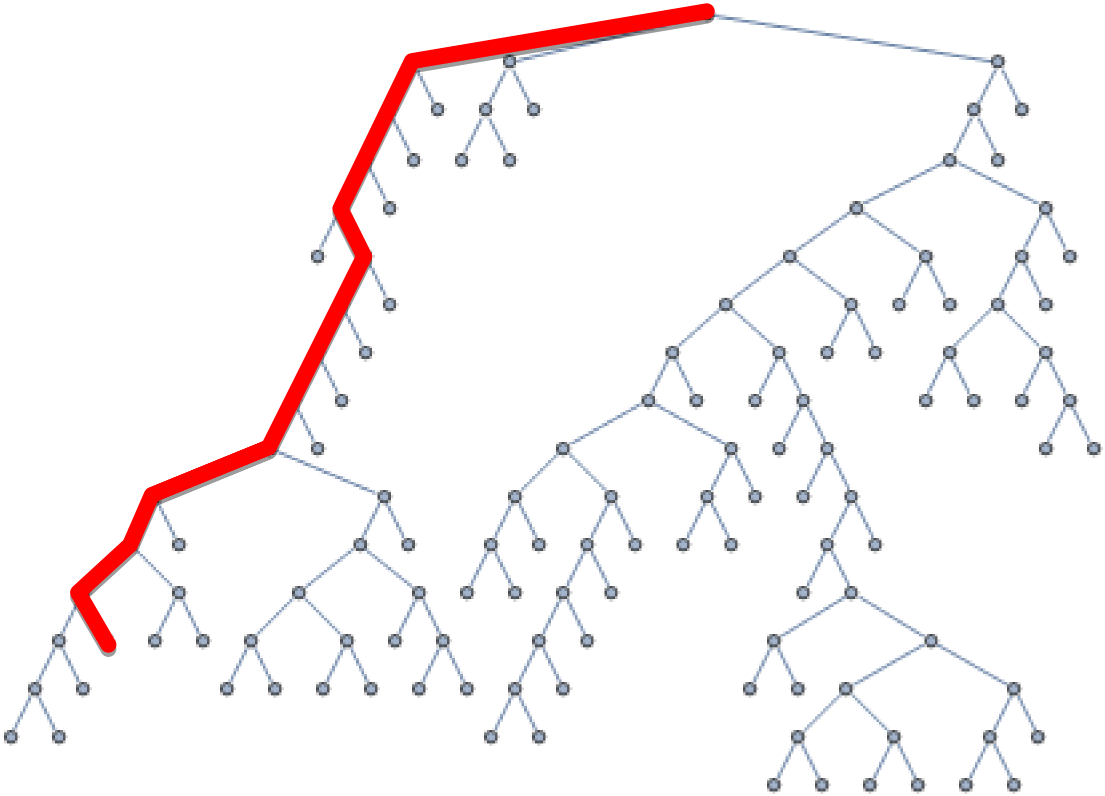
BFS vs. **DFS**



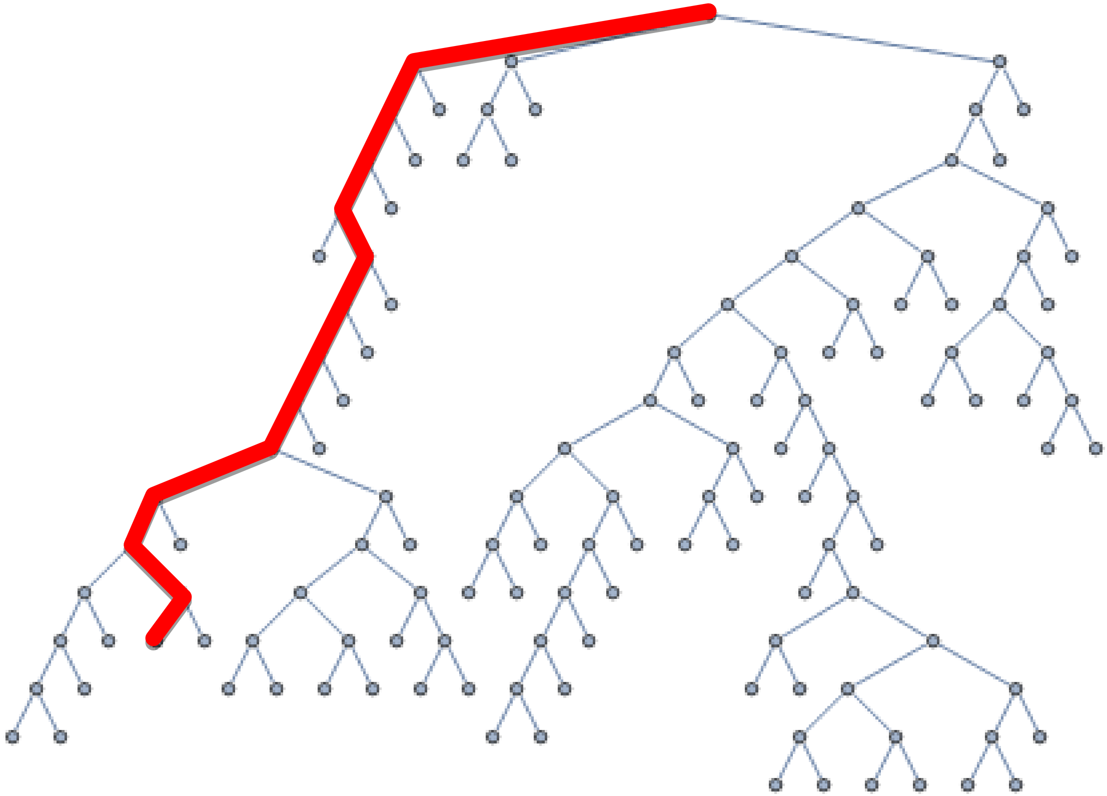
BFS vs. **DFS**



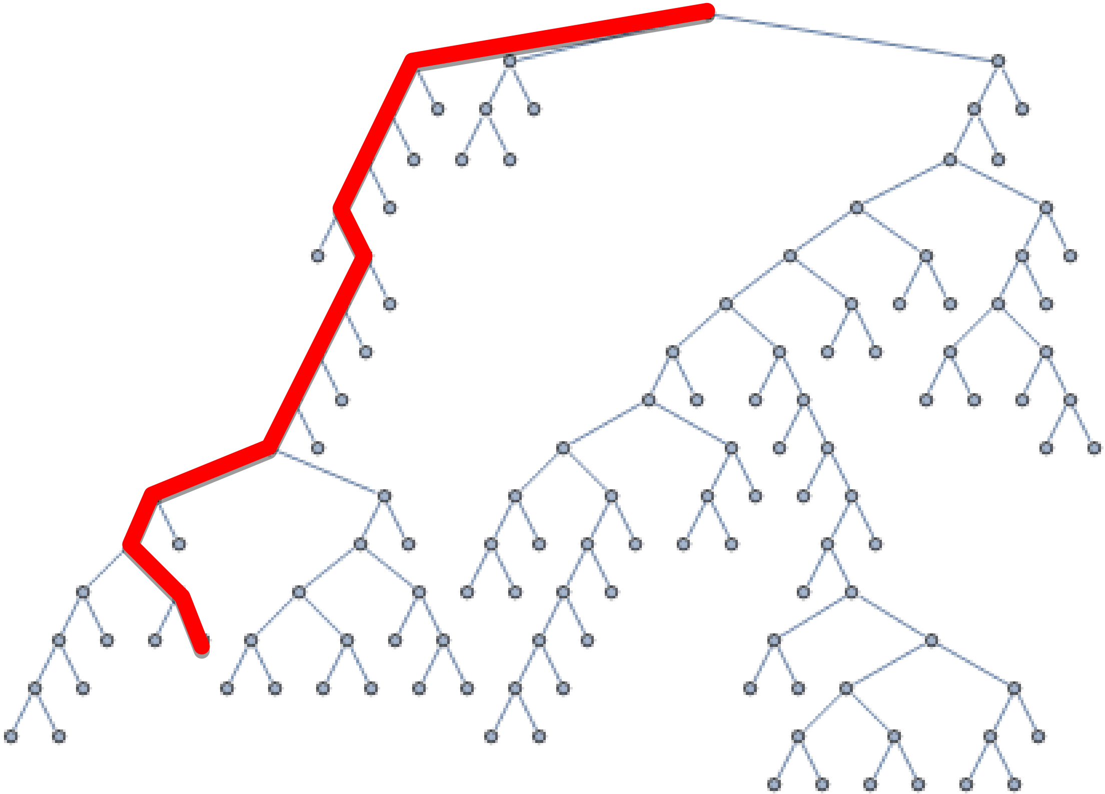
BFS vs. **DFS**



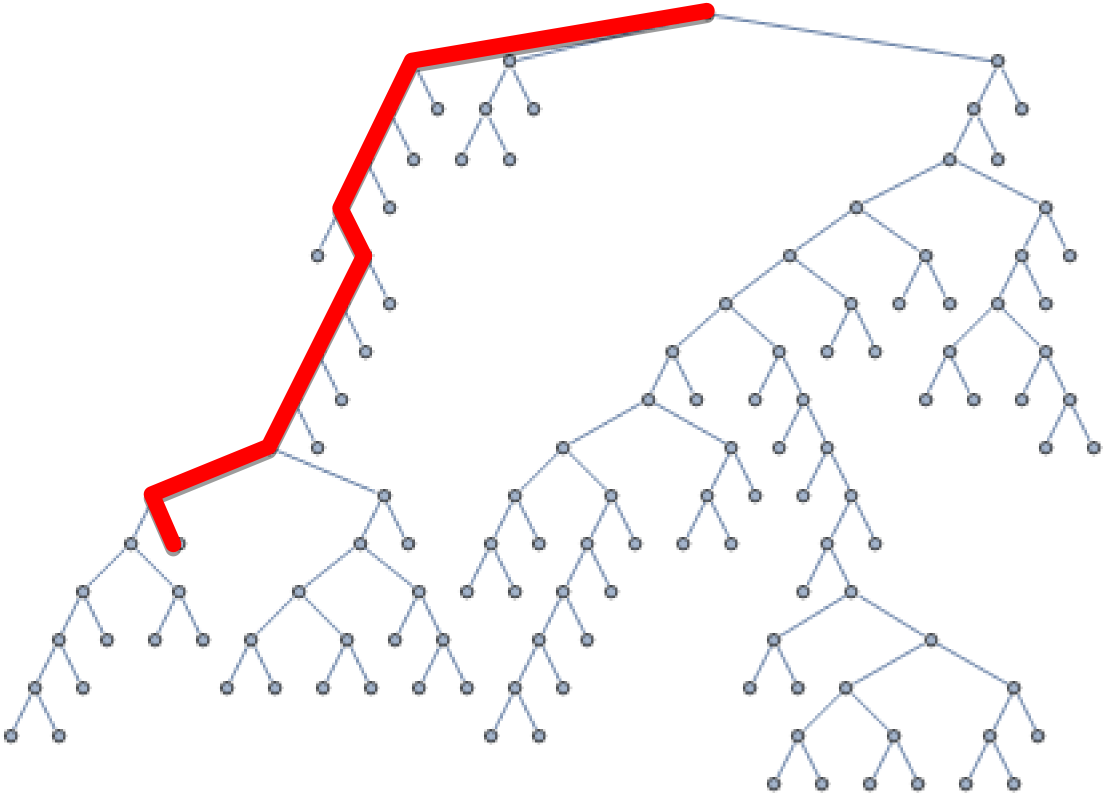
BFS vs. **DFS**



BFS vs. **DFS**



BFS vs. **DFS**



Properties of depth-first search

Complete?

Fails in infinite-depth tree, or states with loops

Optimal?

No – returns the first solution it finds

Time cost?

Worst case: $O(b^m)$ (***b**:* maximum branching factor and *m is the* depth of entire search tree)

Comparison with BFS:

$O(b^d)$ (***d*** is the depth of the optimal solution)

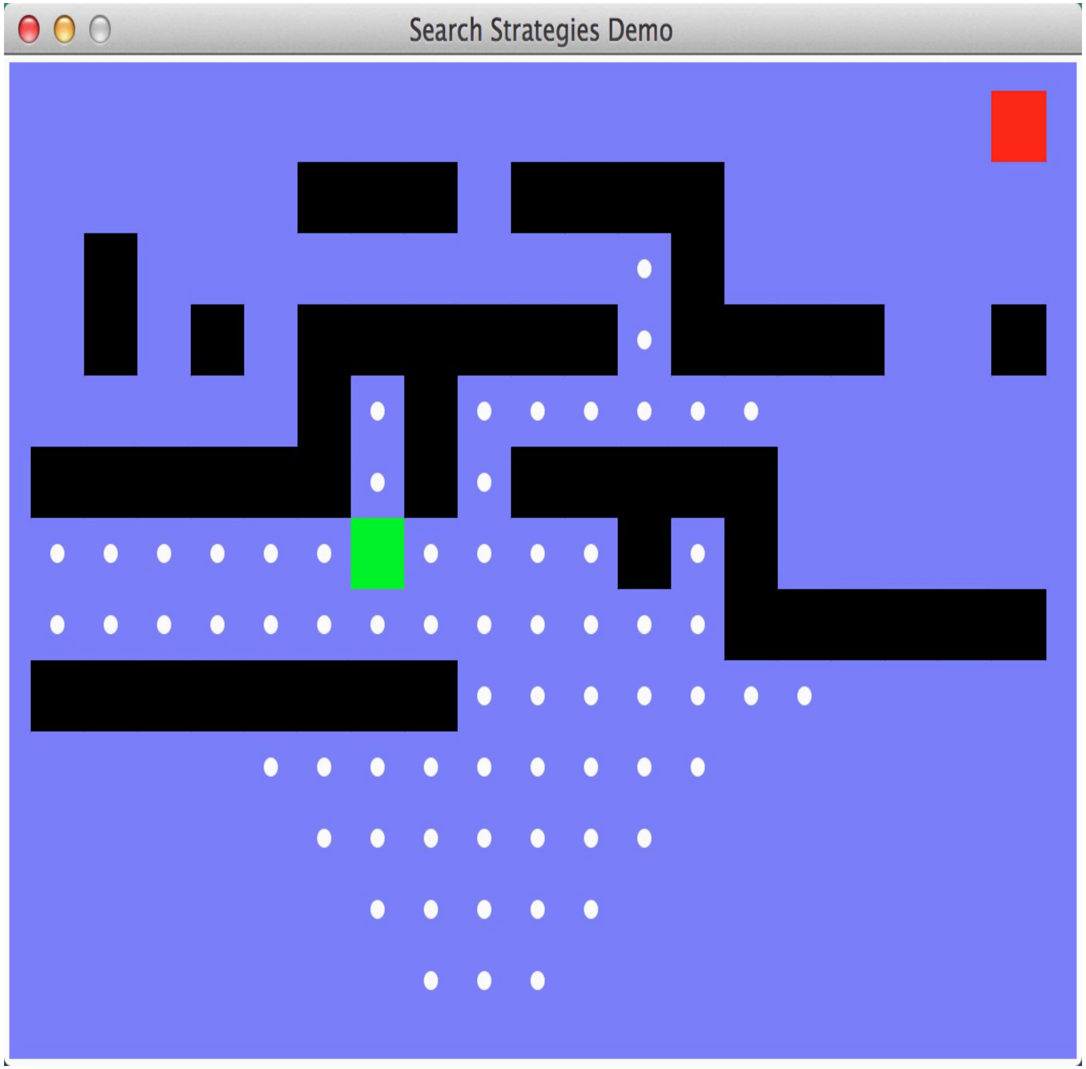
Terrible since *m* >> *d!*

But if there are lots of solutions, DFS may be much faster than BFS

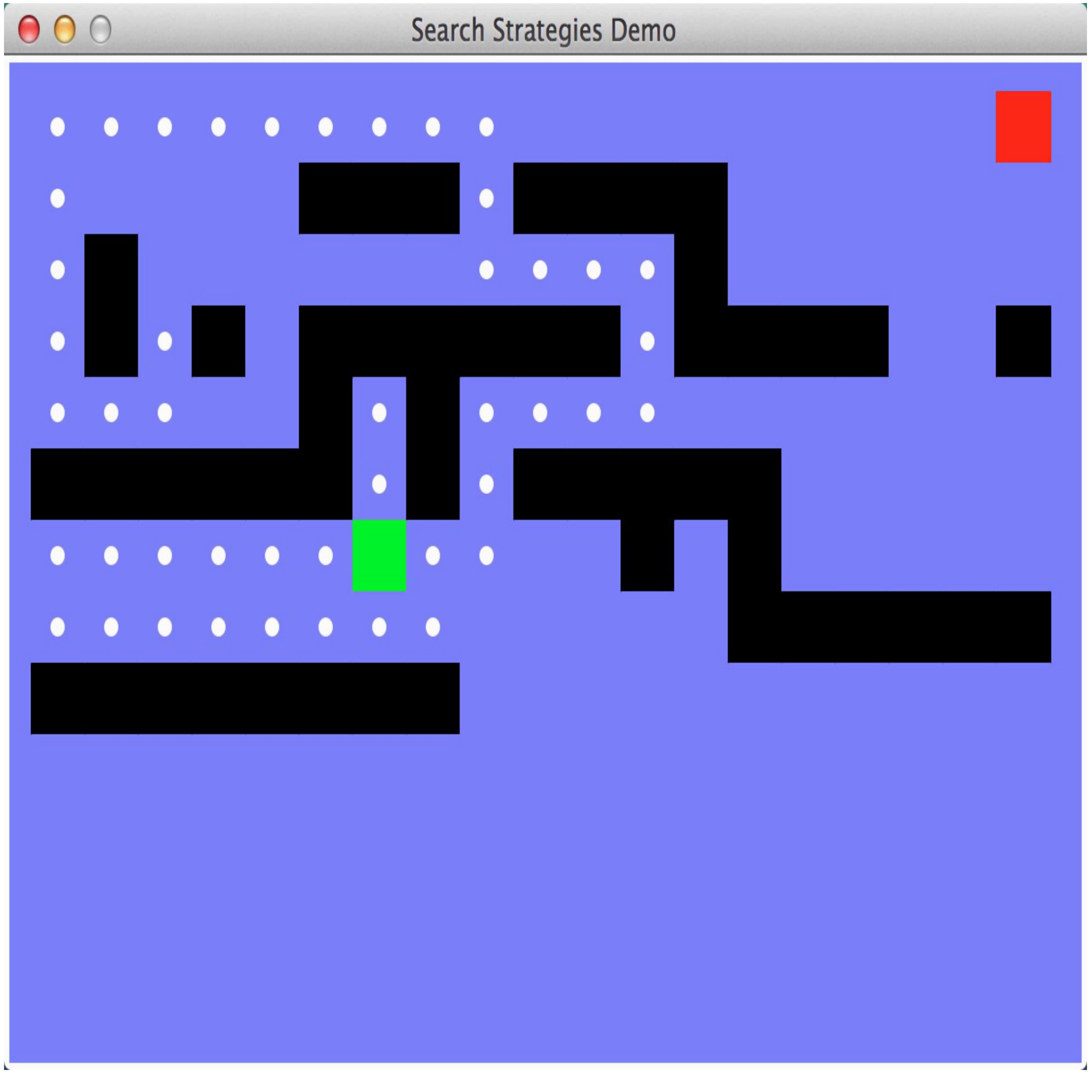
Space cost?

$O(bm)$, i.e., linear cost!

Demo Maze BFS



Demo Maze DFS



Can we have a search strategy that is optimal like BFS and also efficient like DFS?

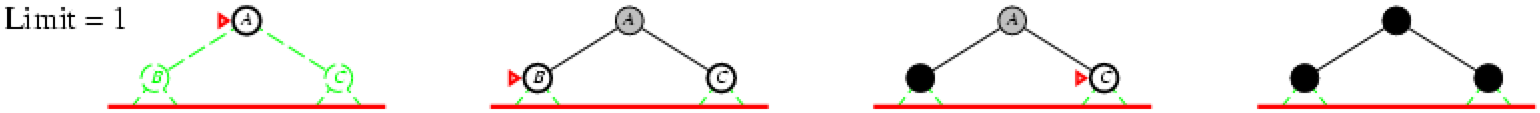
Iterative deepening search (IDS)

Use DFS as a subroutine

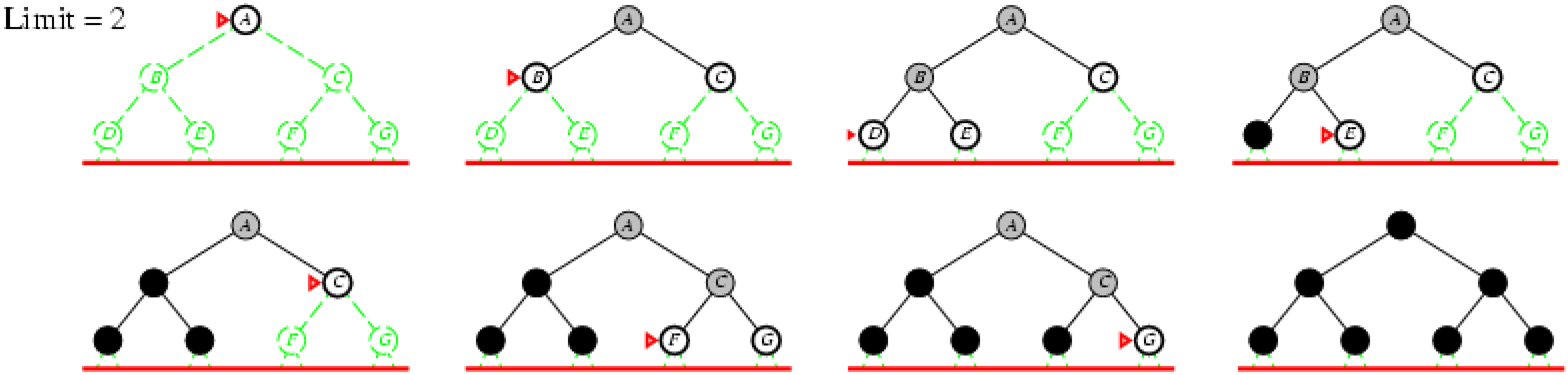
1. Check the root
2. Do a DFS search for a solution of length 1
3. If there is no solution of length 1, do a DFS search for a solution of length 2
4. If there is no solution of length 2, do a DFS search for a solution of length 3…

Iterative deepening search

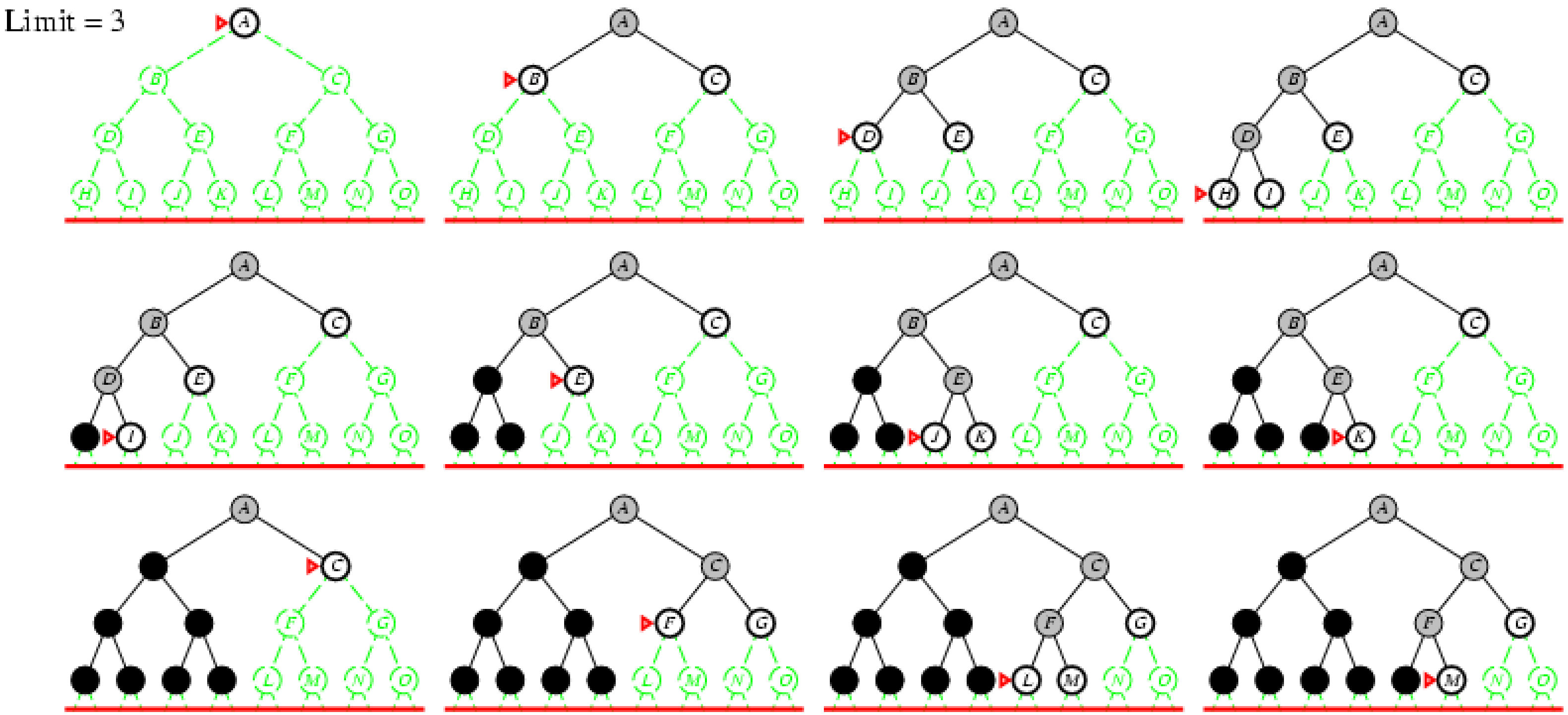
Iterative deepening search



Iterative deepening search



Iterative deepening search



Properties of iterative deepening search

Complete?

Yes

Optimal?

Yes, if step cost = 1

Time cost?

$O(b^d)$ (d is the depth of the optimal solution)

Like BFS

Space cost?

$O(bd)$

Linear cost (even better than DFS)

Uniform Cost Search

Uniform-cost search

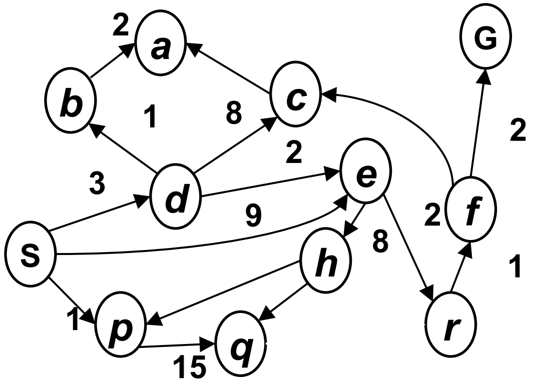
Rank all the nodes *on g(n)*, *the cost* from initial state to *n,* and expand the node with the lowest *g(n)*

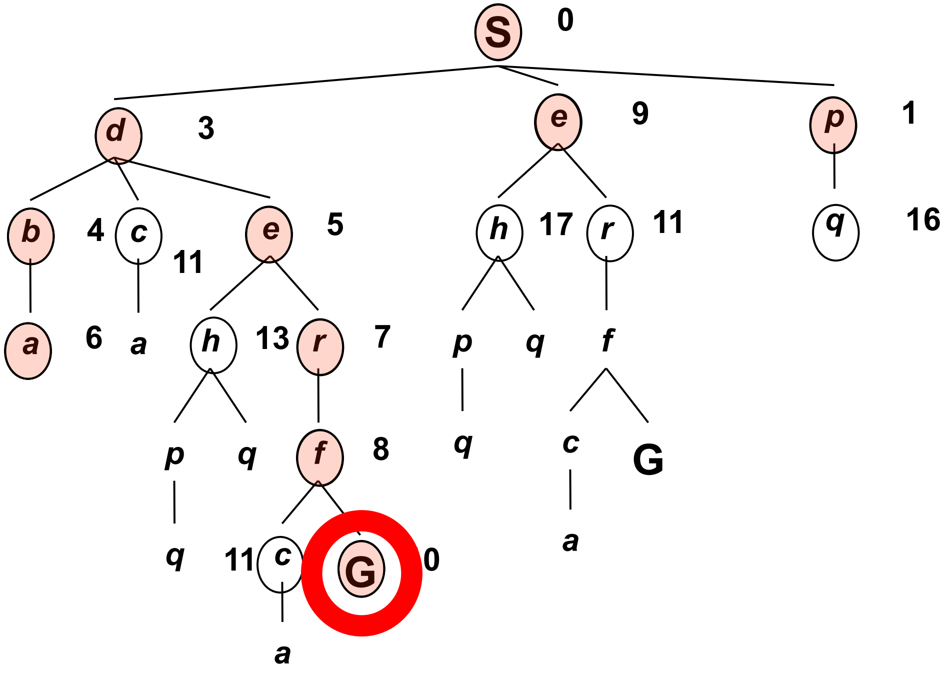
Implementation: *frontier* is a priority queue ordered by *g(n)*

Uniform-cost search example

***Strategy:* *expand a cheapest node first*:**

Frontier is a priority queue (priority: g-value (cumulative cost))





UCS vs BFS

- BFS is a special case of UCS when all step costs are equal

UCS == Dijkstra's algorithm

- UCS is equivalent to Dijkstra's algorithm except that Dijkstra's algorithm is used to find *shortest paths* *from initial node to every other nodes* in a graph, whereas UCS is only used to find the *shortest path* *from initial node to a goal node*.

Uniform Cost Search (UCS) Properties

Is it complete?

- Yes!

Is it optimal?

- Yes!

Time and space cost?

- Expand all nodes with cost less than optimal solution ($g(n) \leq C^*$)

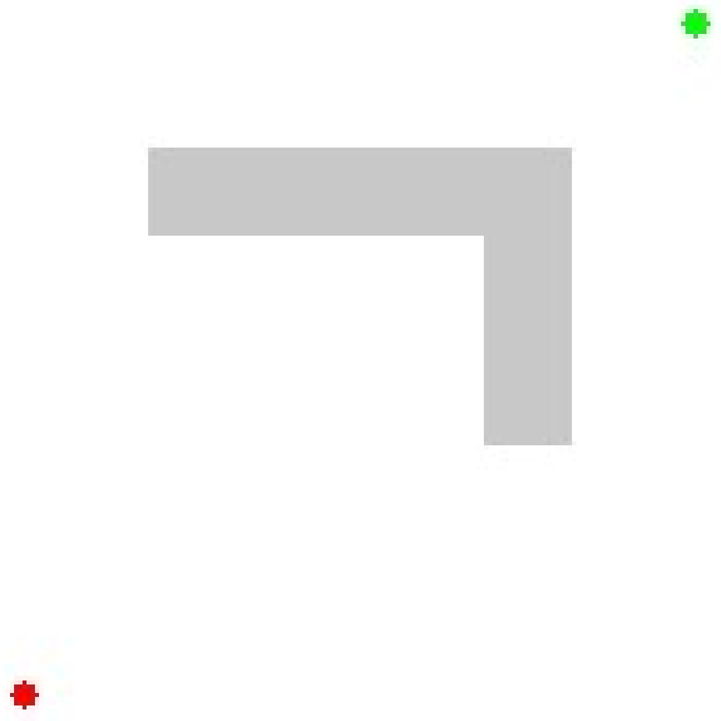
Pros and cons of uniform cost search

The good: UCS is complete and optimal!

The bad:

- Explores nodes in every “direction”
- Expensive

Uniform-cost search



Review: Uninformed search strategies

Algorithm	Complete?	Optimal?	Time complexity	Space complexity
BFS	Yes	If all step costs are equal	$O(b^d)$	$O(b^d)$
DFS	No	No	$O(b^m)$	$O(bm)$
IDS	Yes	If all step costs are equal	$O(b^d)$	$O(bd)$
UCS	Yes	Yes	Number of nodes with $g(n) \leq C^*$	

b: maximum branching factor of the search tree

d: depth of the optimal solution

m: maximum length of any path in the search tree

C*: cost of optimal solution

g(n): cost of path from initial state to node n

How to choose between uninformed search strategies

- Uniform Cost Search will reach the goal in the cheapest way possible.
- Breadth-First Search will reach the goal in the shortest way possible.
- Depth-First Search is not optimal but it may run much faster when multiple solutions exists.
- Iterative deepening search (IDS) mixed BFS with DFS

The One Queue for All Strategies

All search algorithms are the same except for frontier strategies

- Conceptually, all frontiers are **priority queues**.
- Practically, for DFS and BFS, you can avoid the overhead from an priority queue by using stacks and queues

Recap: All search strategies so far…

Algorithm	Complete?	Optimal?	Time complexity	Space complexity
BFS	Yes	If all step costs are equal	$O(b^d)$	$O(b^d)$
DFS	No	No	$O(b^m)$	$O(bm)$
IDS	Yes	If all step costs are equal	$O(b^d)$	$O(bd)$
UCS	Yes	Yes	Number of nodes with $g(n) \leq C^*$	
Greedy	No	No	Worst case: $O(b^m)$ Best case: $O(bd)$	
A*	Yes	Yes (if heuristic is admissible)	Number of nodes with $g(n)+h(n) \leq C^*$	

More demos

http://qiao.github.io/PathFinding.js/visual/

http://bryukh.com/labyrinth-algorithms/