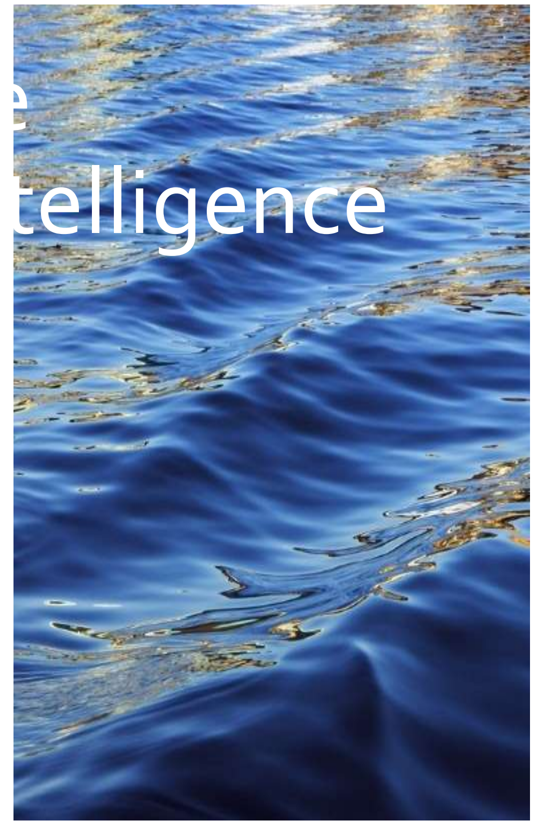
CS 4810 Artificial Intelligence
CS 6810 Topics in Artificial Intelligence





Uninformed search strategies (Section 3.4)

Uninformed search strategies

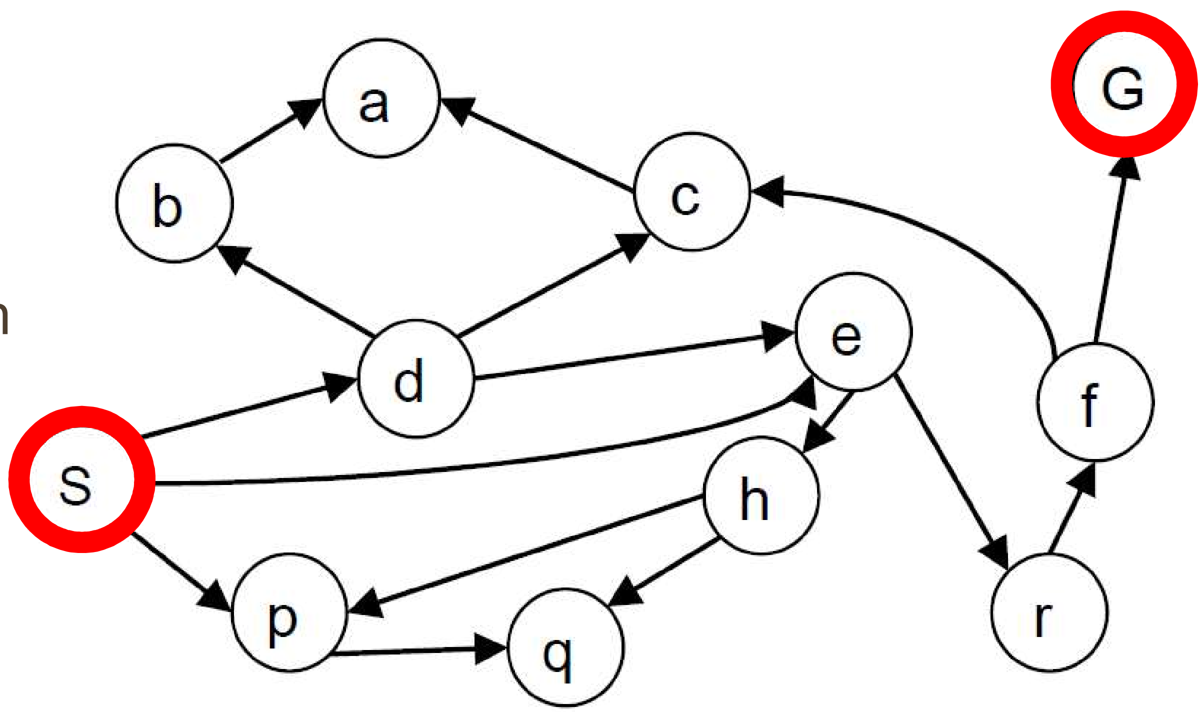
- A **search strategy** is defined by picking the order of node expansion
- **Uninformed** search strategies use only the information available in the problem definition
  - Breadth-first search
  - Depth-first search
  - Iterative deepening search
  - Uniform-cost search

Breadth-First Search

Breadth-First Search

- Expand shallowest unexpanded node
- Implementation: *frontier* is a FIFO queue

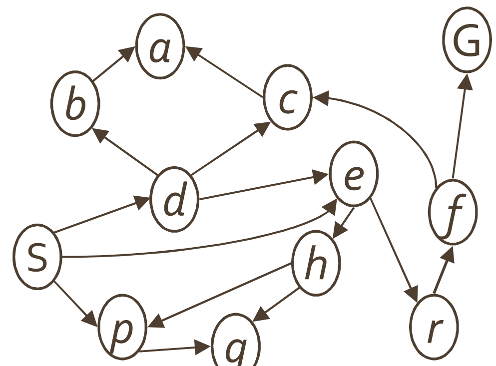
Example state space graph for a tiny search problem



Breadth-First Search

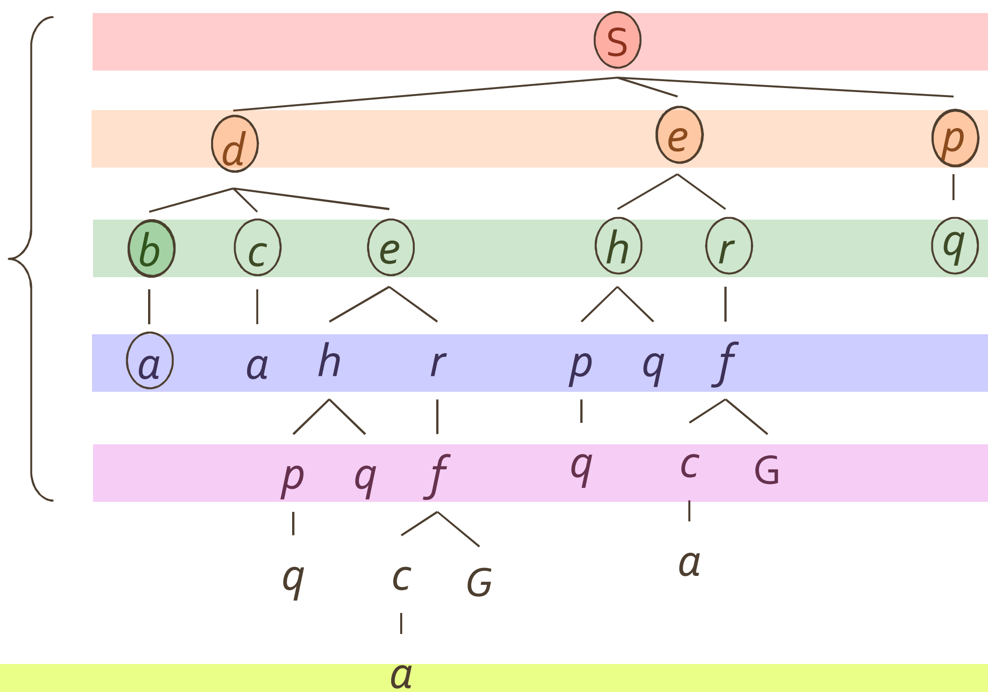
Strategy: expand a shallowest node first

Implementation: Frontier is a FIFO queue



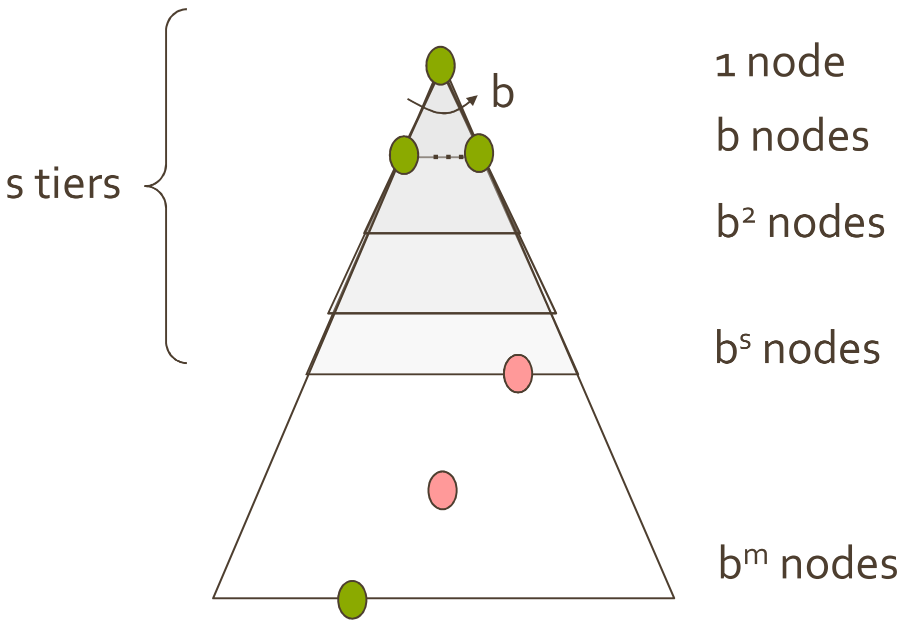
Search

Tiers



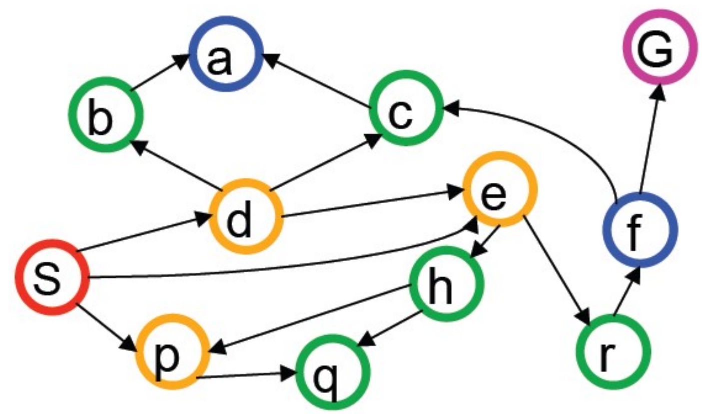
Breadth-First Search (BFS) Properties

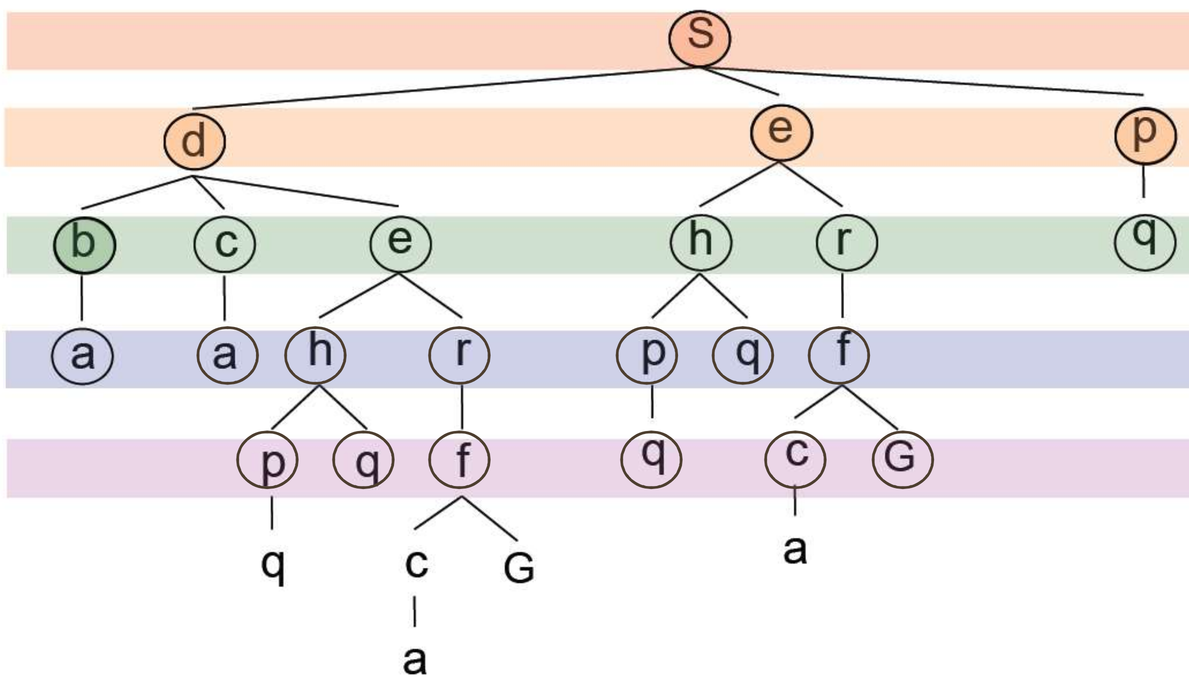
- What nodes does BFS expand?
  - Processes all nodes above shallowest solution
  - Let depth of shallowest solution be s
  - Search takes time $O(b^s)$
- How much space does the frontier take?
  - Has roughly the last tier, so $O(b^s)$
- Is it complete?
  - s must be finite if a solution exists, so yes!
- Is it optimal?
  - Only if costs are all 1 (more on costs later)



Breadth-first search

- Expansion order:
  (S,d,e,p,b,c,e,h,r,q,a,a,
  h,r,p,q,f,p,q,f,q,c,G)



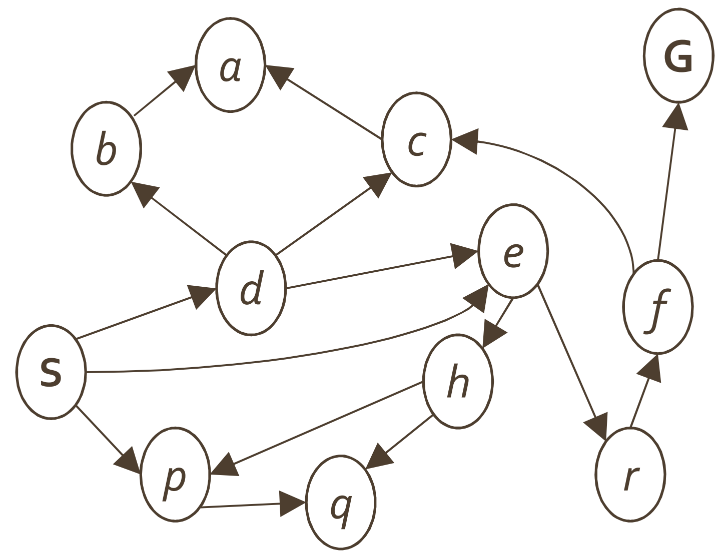


State Space Graphs

- State space graph: A mathematical representation of a search problem
  - Nodes are (abstracted) world configurations
  - Arcs represent successors (action results)
  - The goal test is a set of goal nodes (maybe only one)

- In a search graph, each state occurs only once!

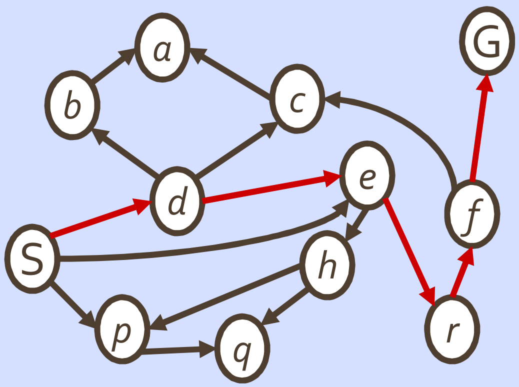
- We can rarely build this full graph in memory (it's too big), but it's a useful idea



Tiny search graph for a tiny search problem

State Space Graphs vs. Search Trees

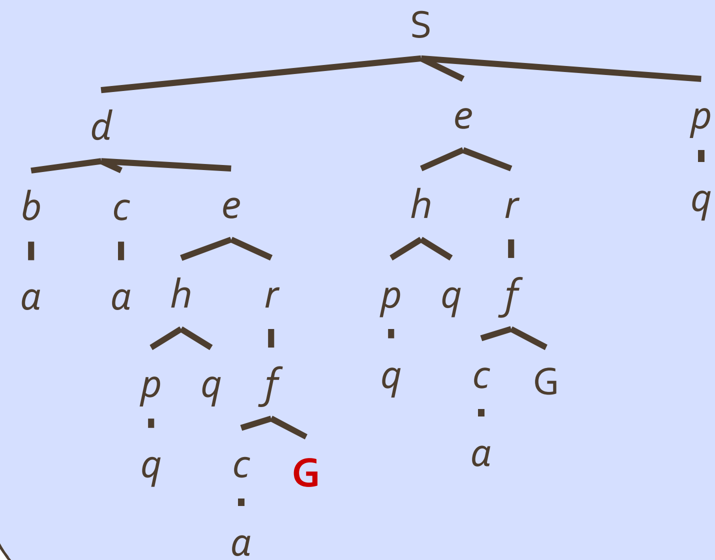
State Space Graph



Each NODE in in the search tree is an entire PATH in the state space graph.

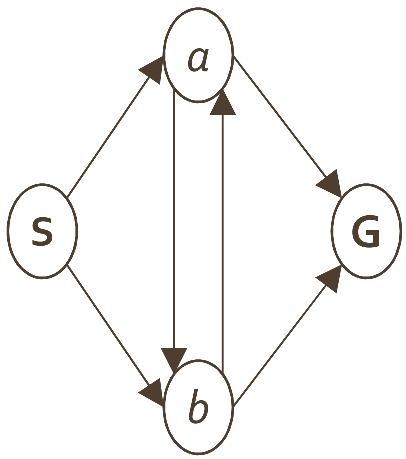
We construct both on demand – and we construct as little as possible.

Search Tree



Quiz: State Space Graphs vs. Search Trees

Consider this 4-state graph:



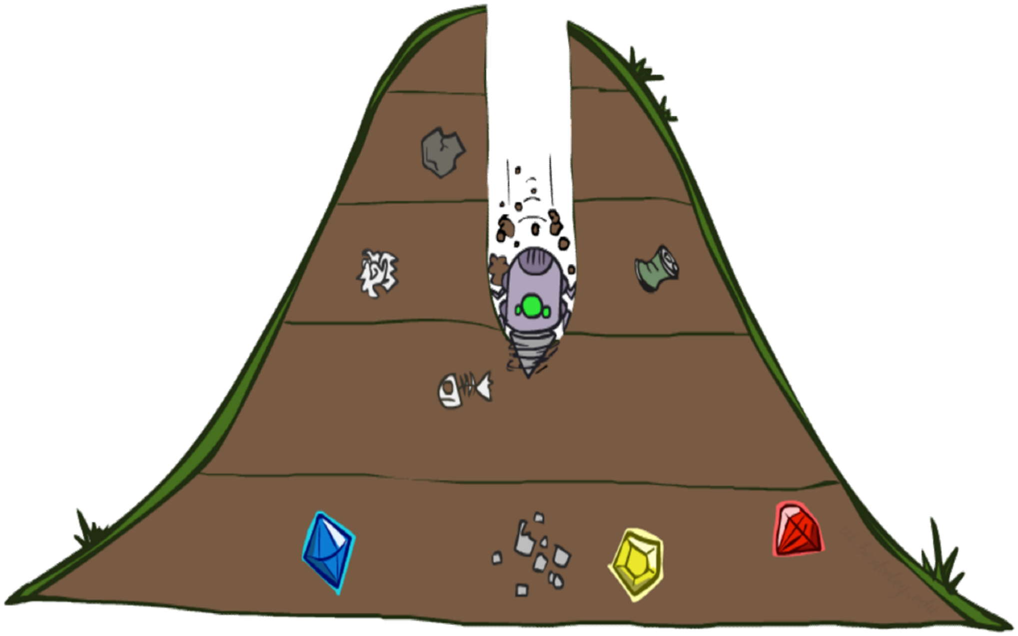
How big is its search tree (from S)?

$$\infty$$

Important: Lots of repeated structure in the search tree!

Depth-first search

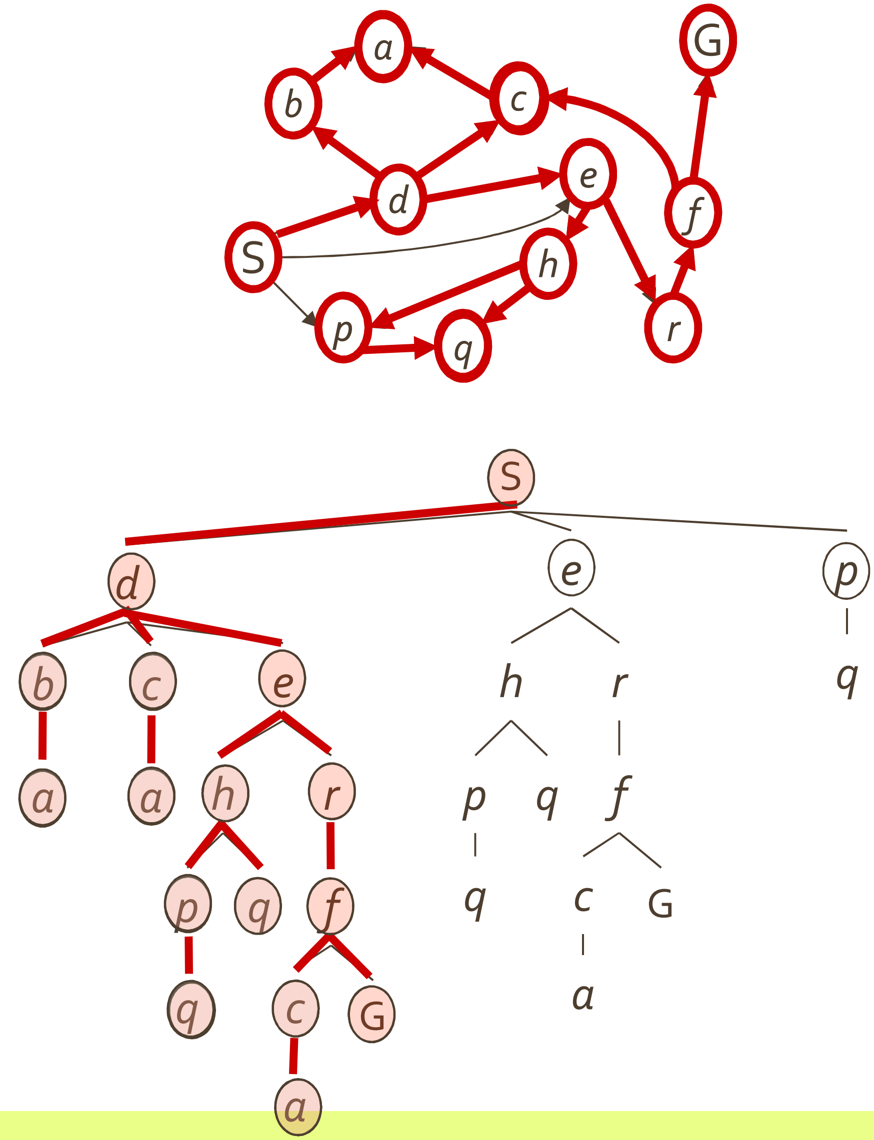
- Expand deepest unexpanded node
- Implementation: *frontier* is a LIFO queue



Depth-First Search

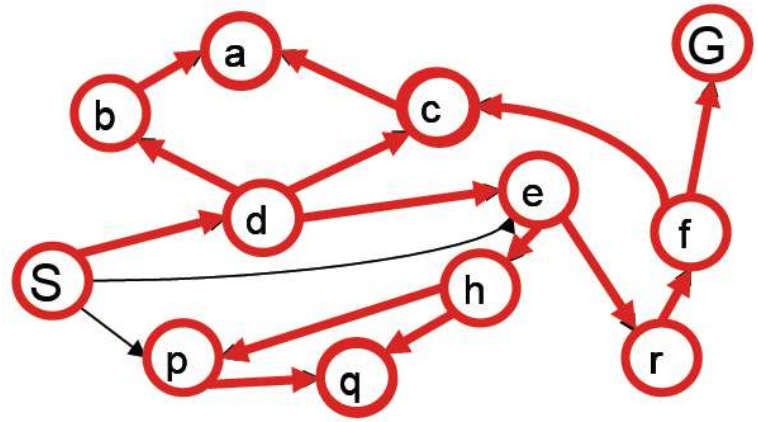
Strategy: expand a deepest node first

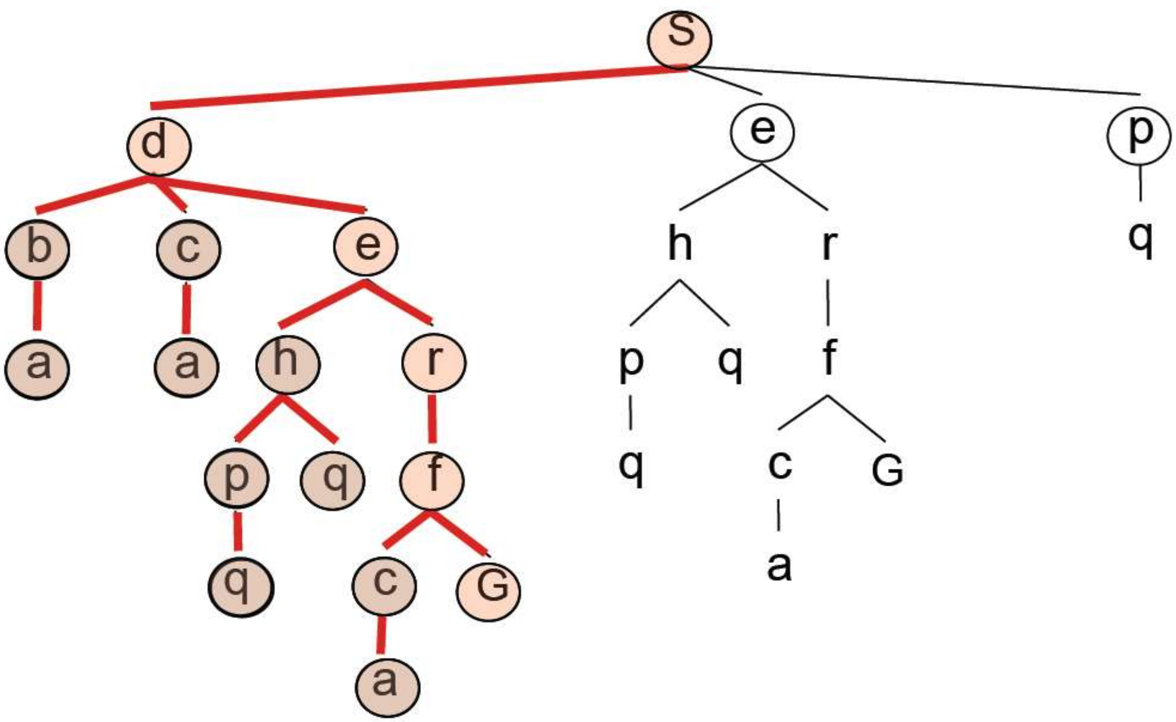
Implementation: Frontier is a LIFO stack

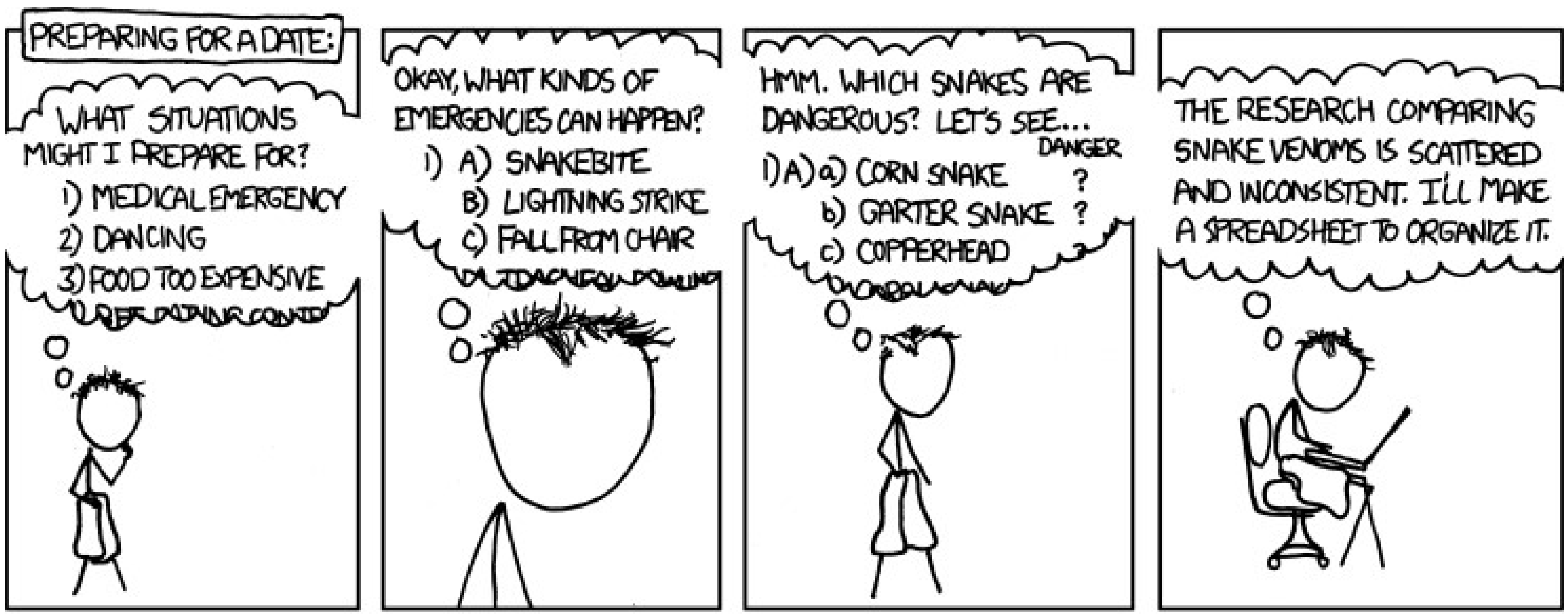


Depth-first search

- Expansion order:
  (d,b,a,c,a,e,h,p,q,q, r,f,c,a,G)

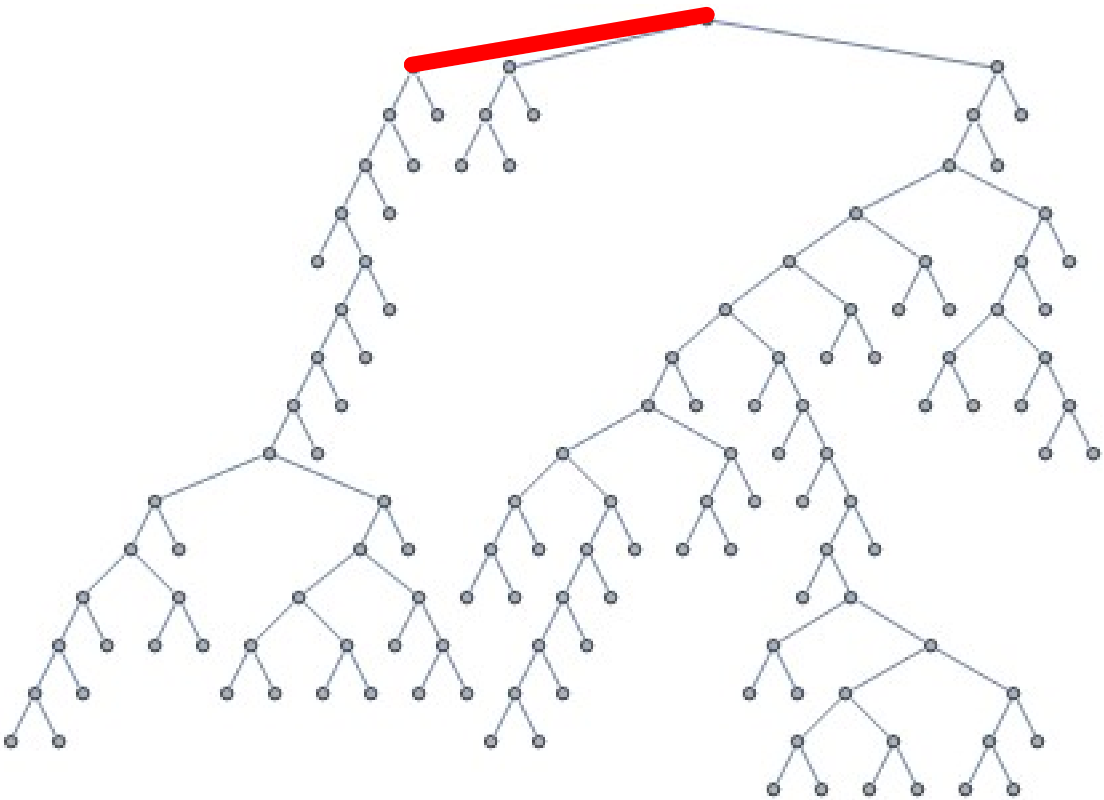




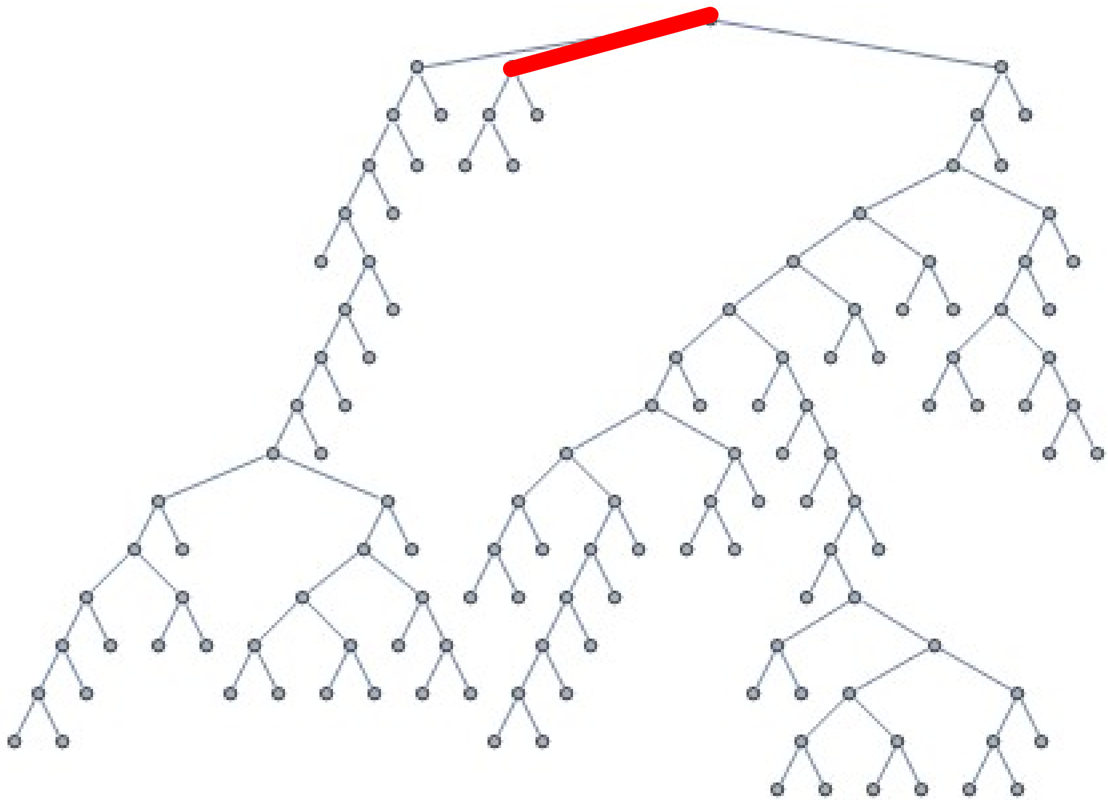


http://xkcd.com/761/

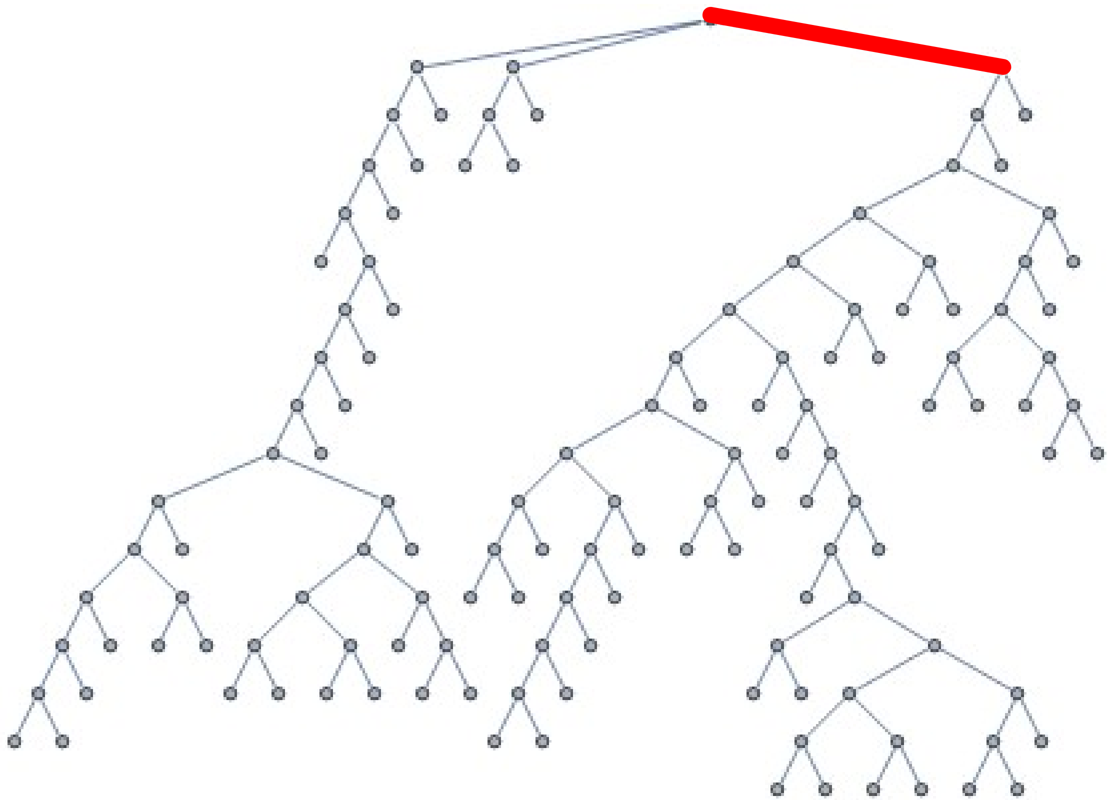
BFS vs. DFS



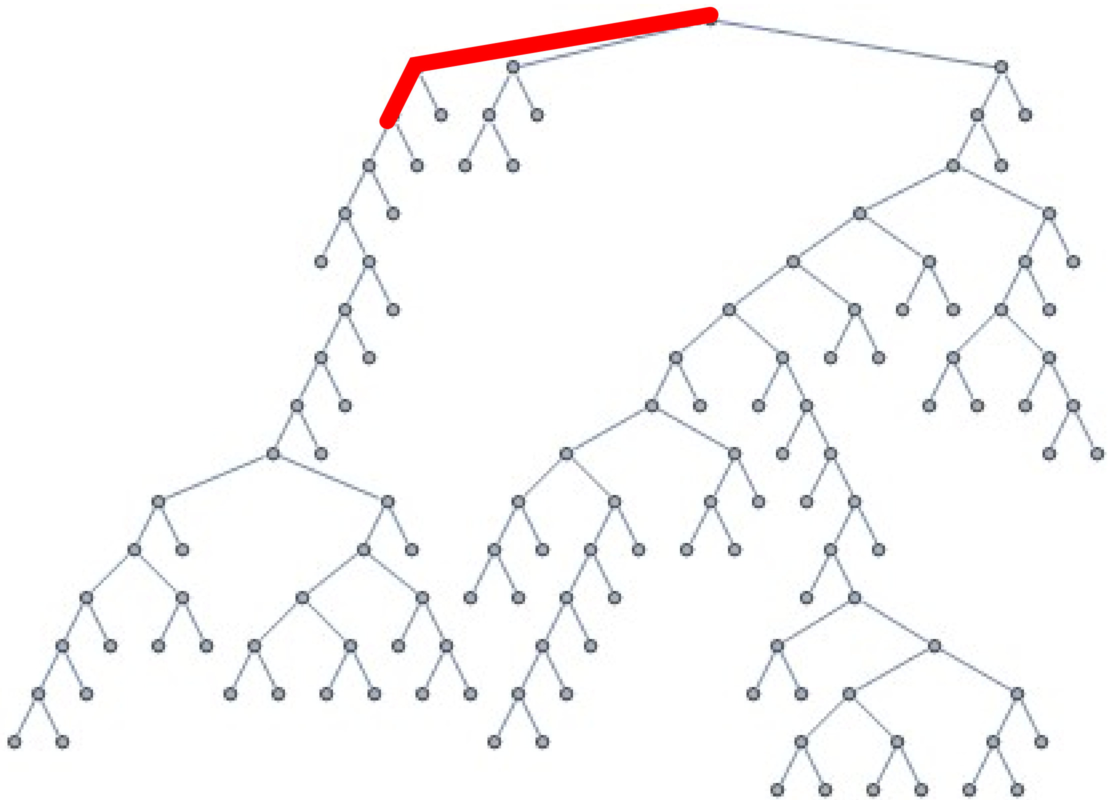
BFS vs. DFS



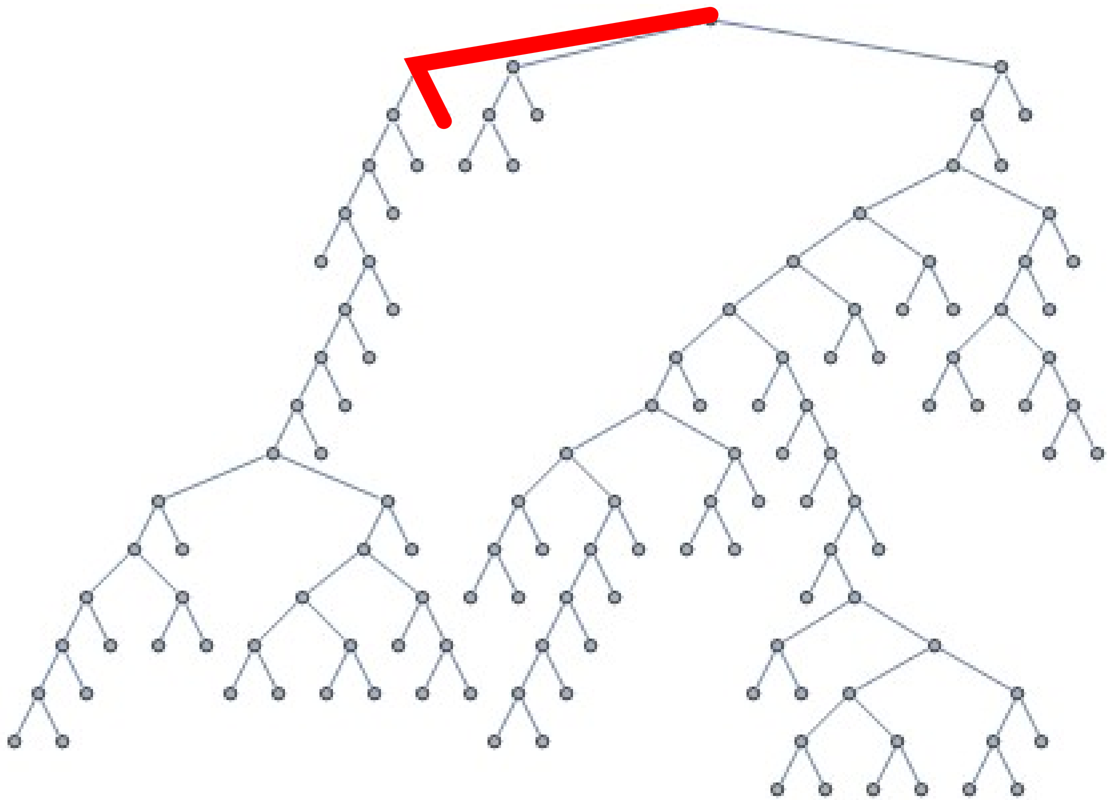
BFS vs. DFS



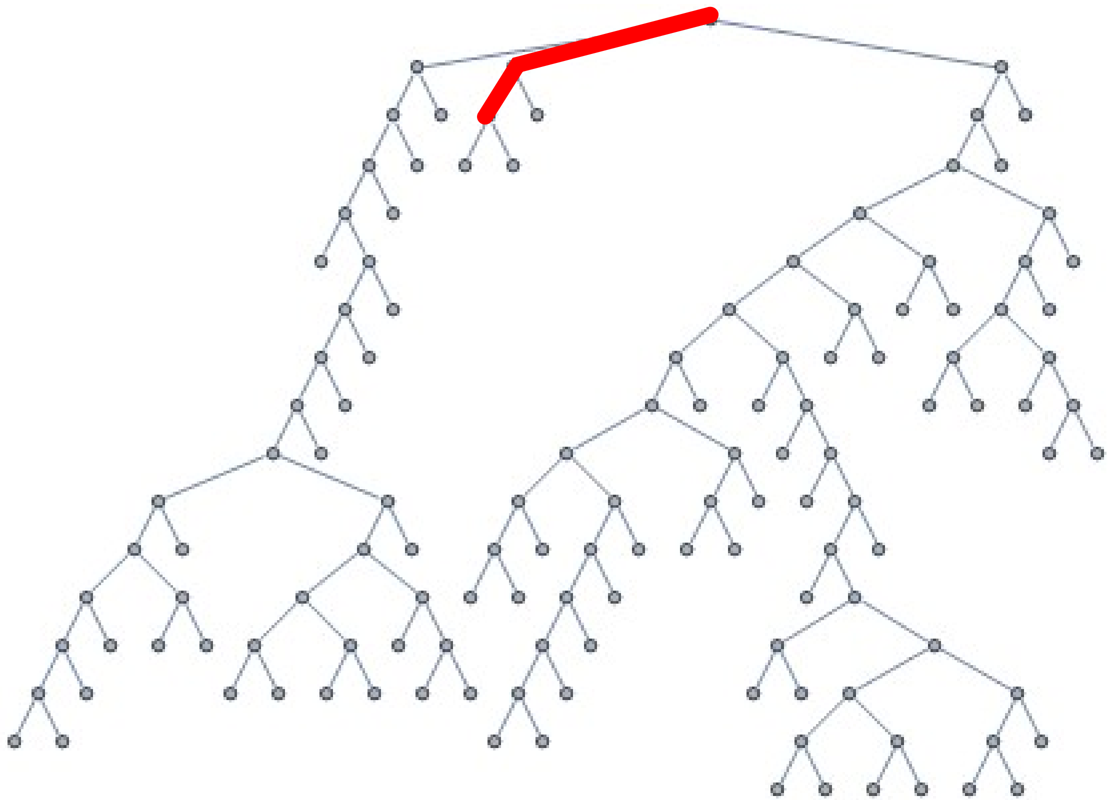
BFS vs. DFS



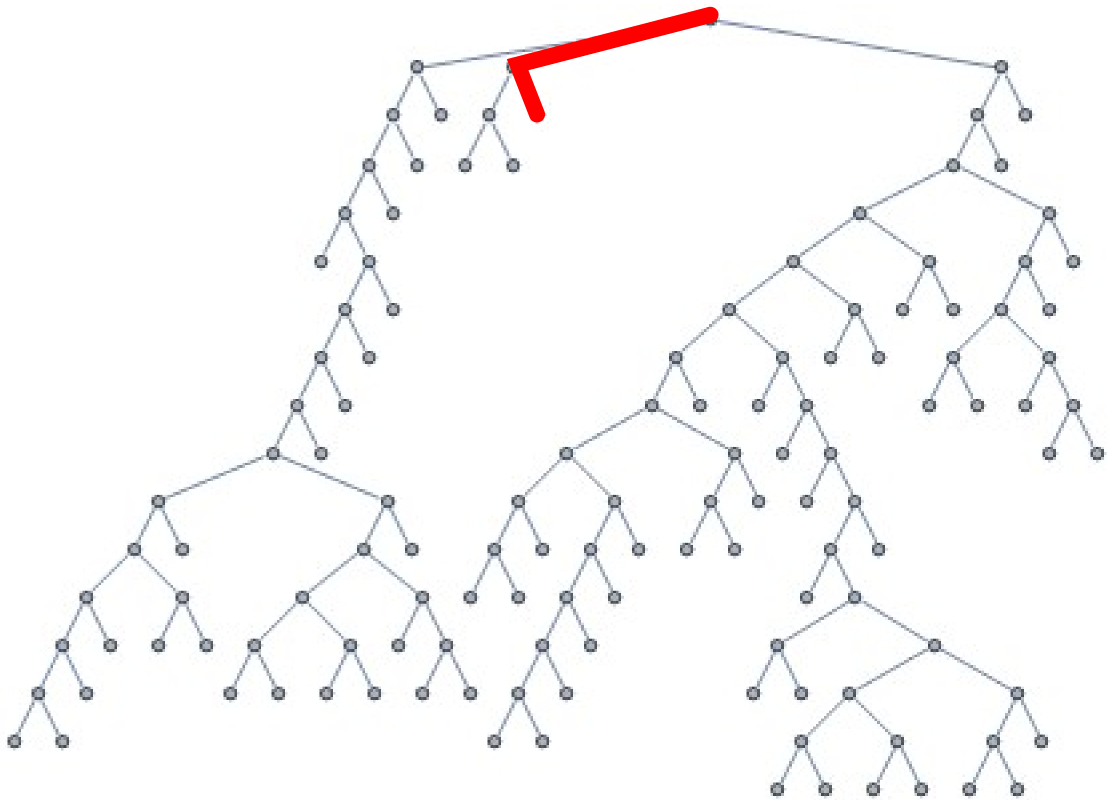
BFS vs. DFS



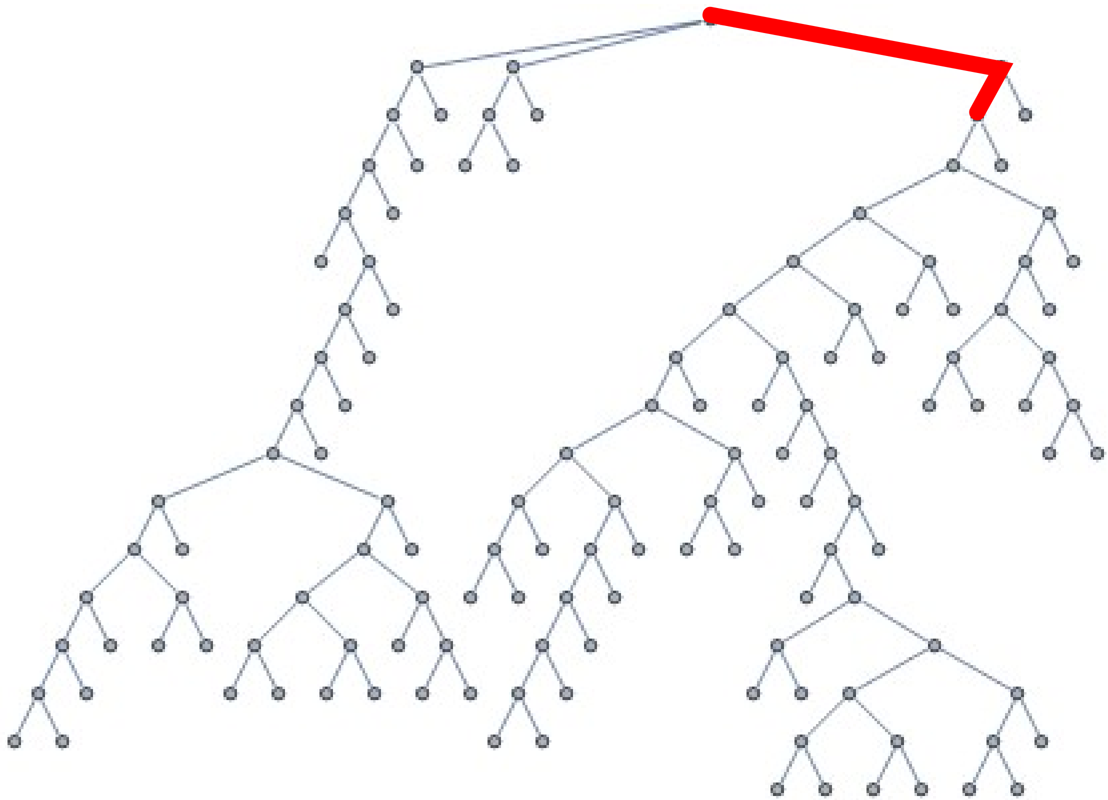
BFS vs. DFS



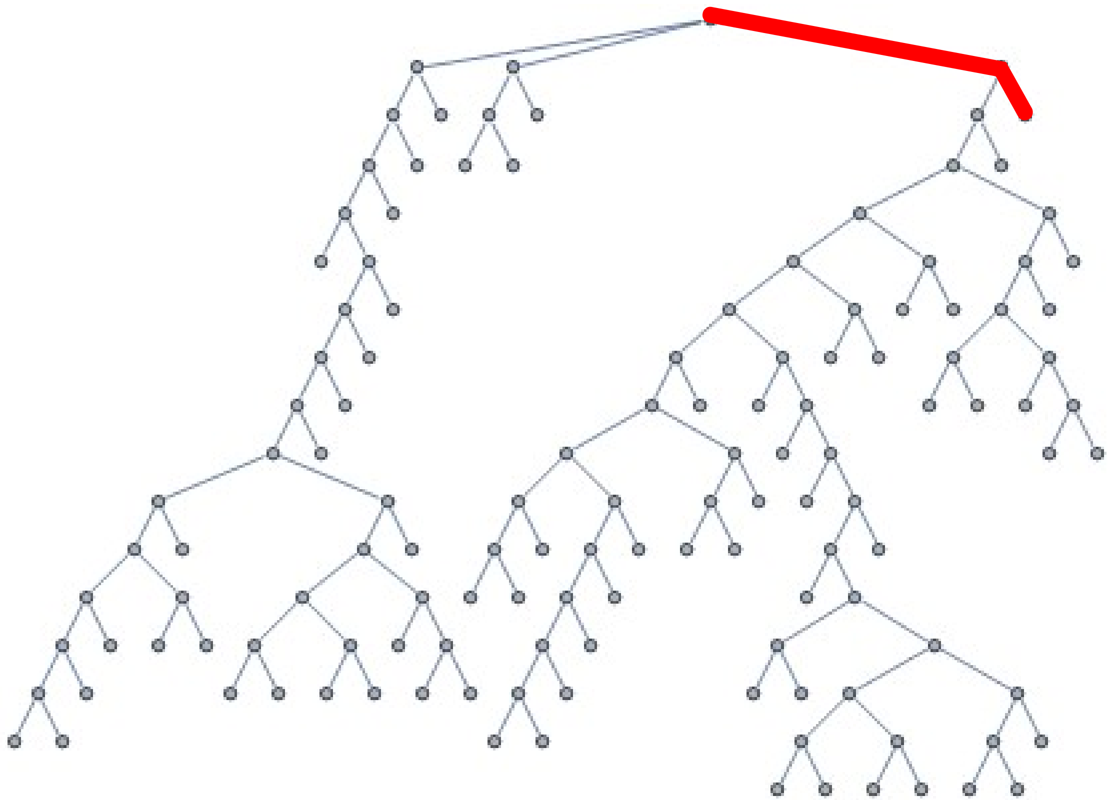
BFS vs. DFS



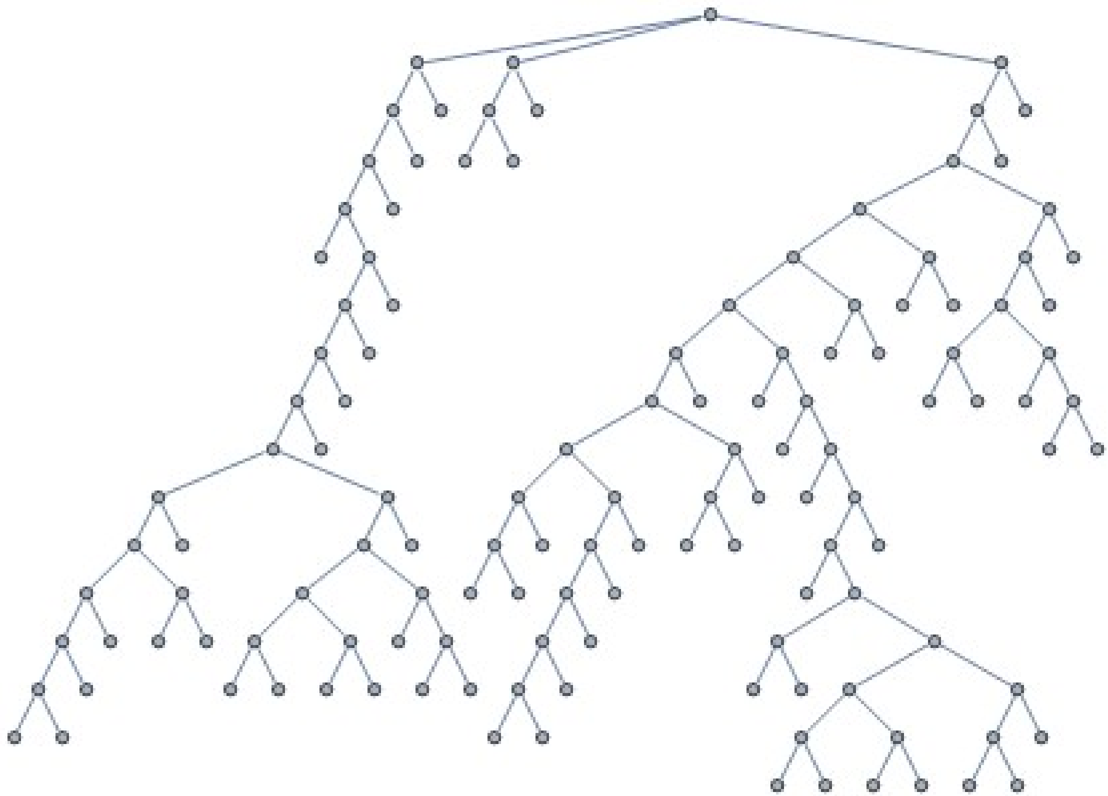
BFS vs. DFS



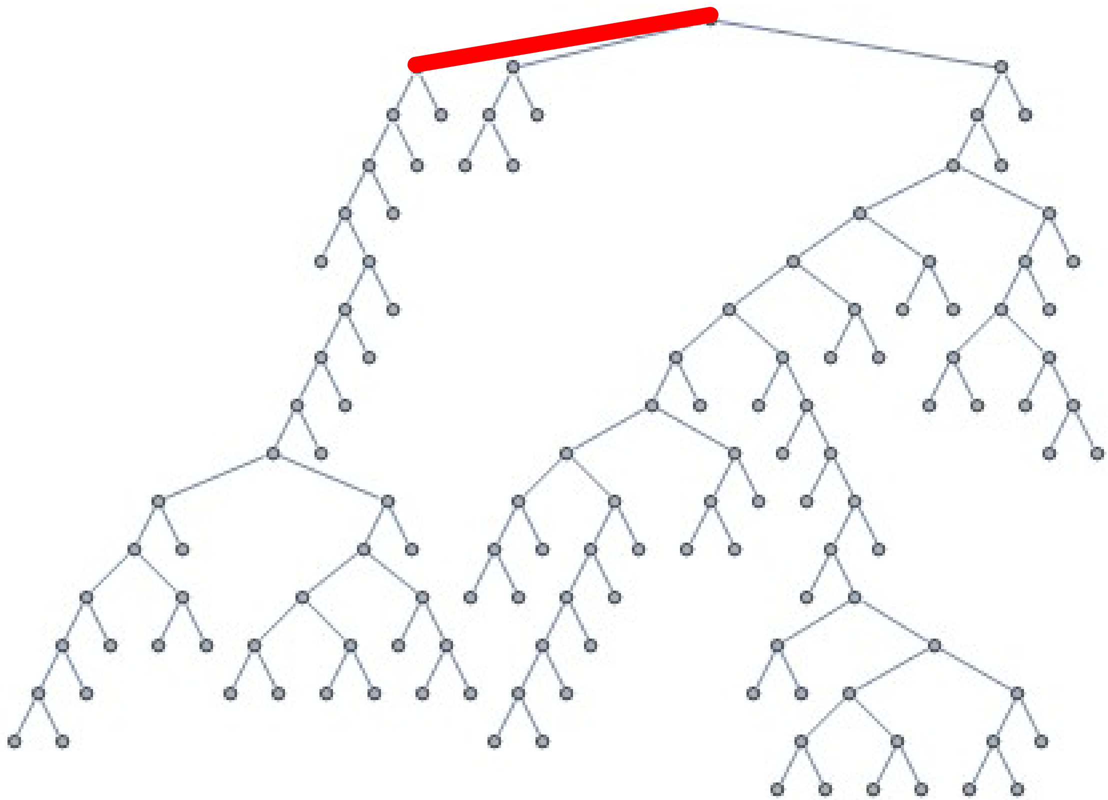
BFS vs. DFS



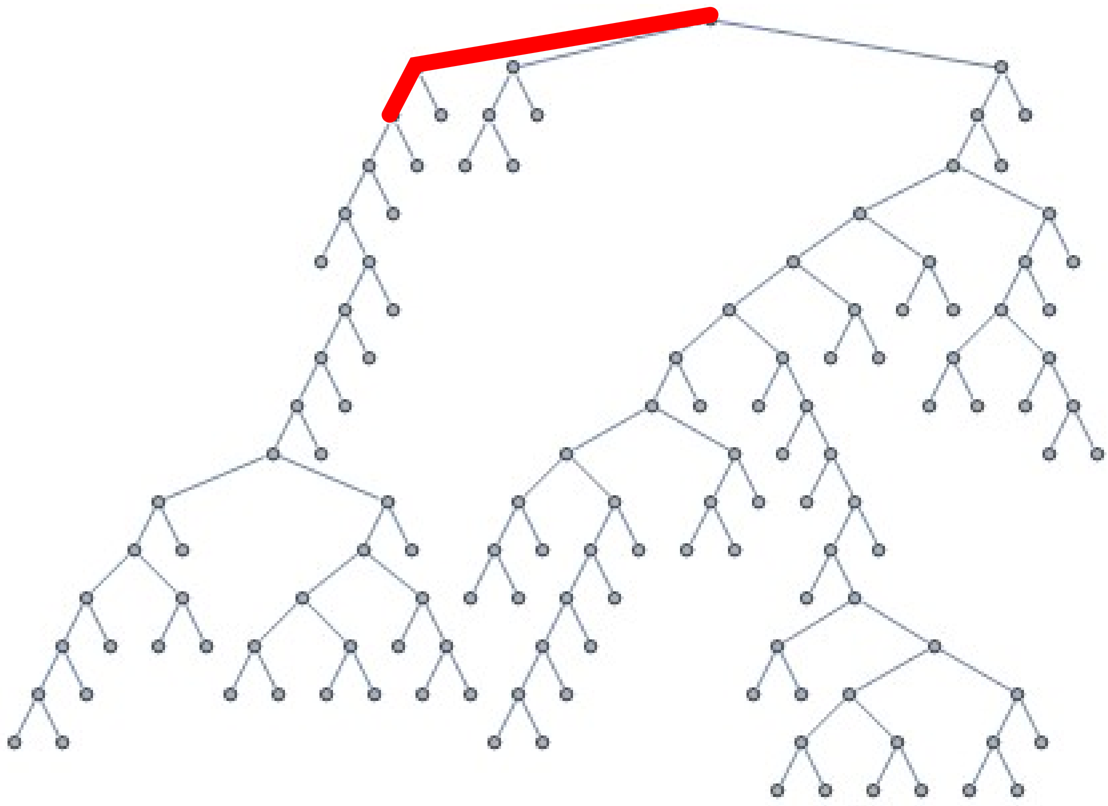
BFS vs. **DFS**



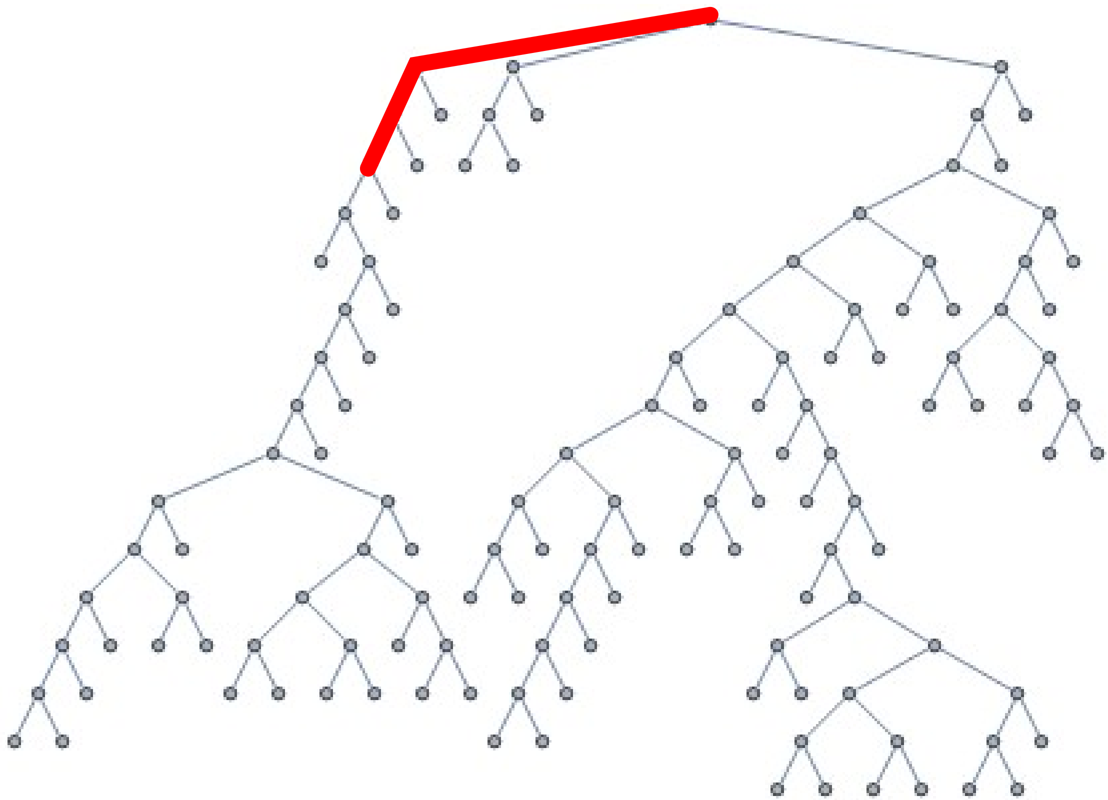
BFS vs. **DFS**



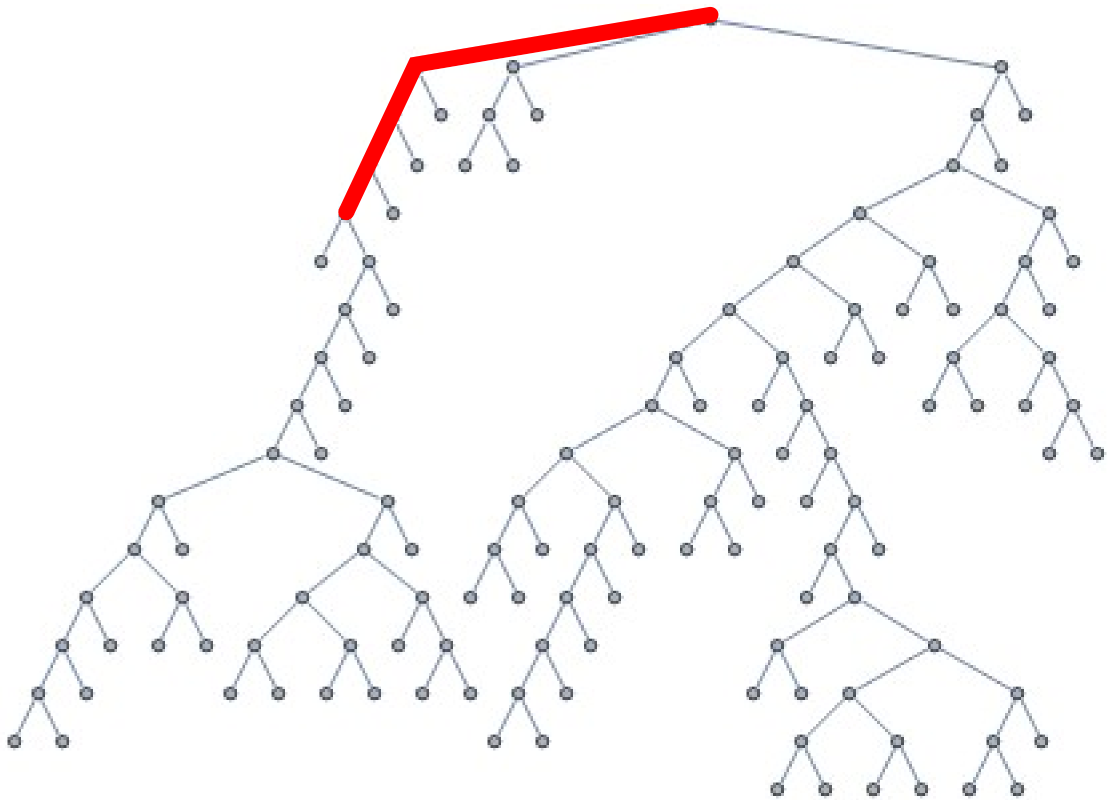
BFS vs. **DFS**



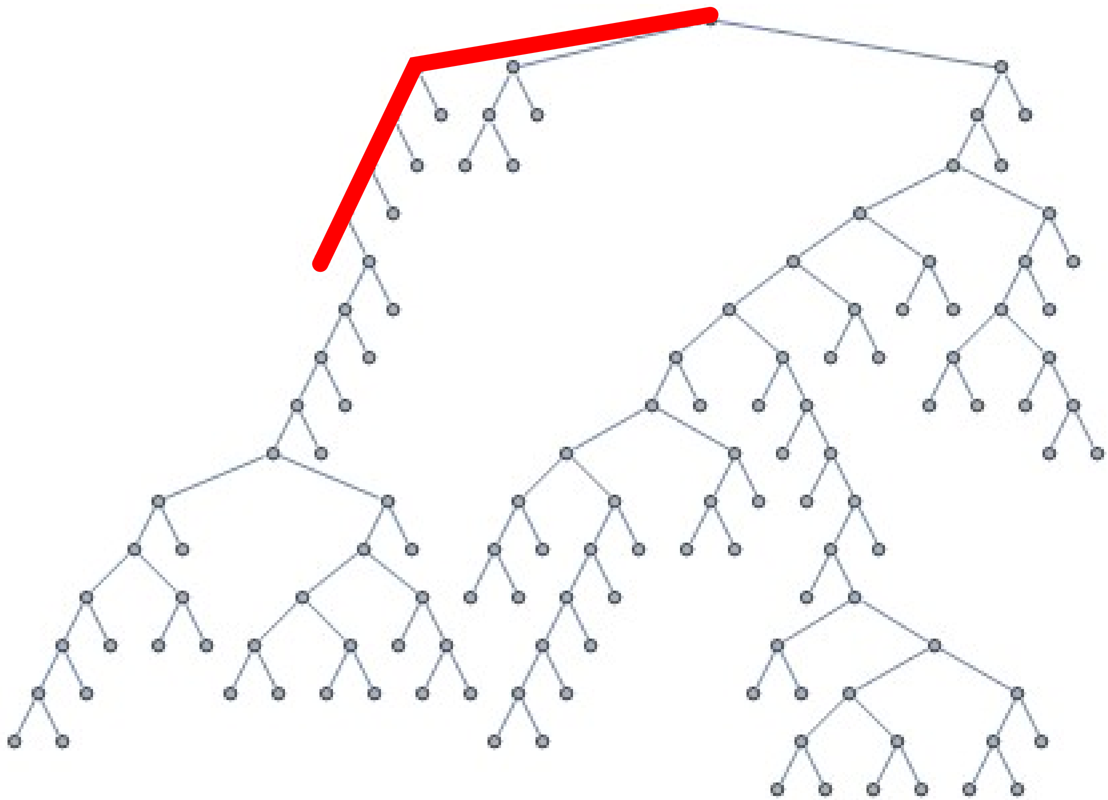
BFS vs. **DFS**



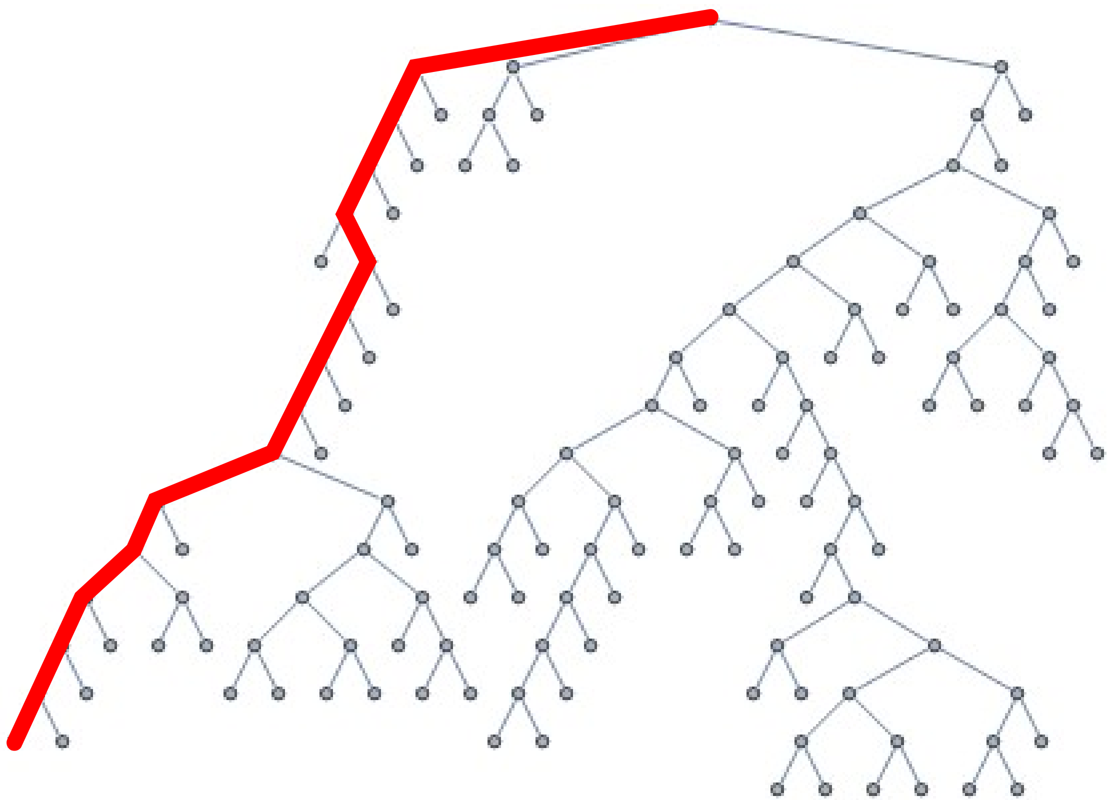
BFS vs. **DFS**



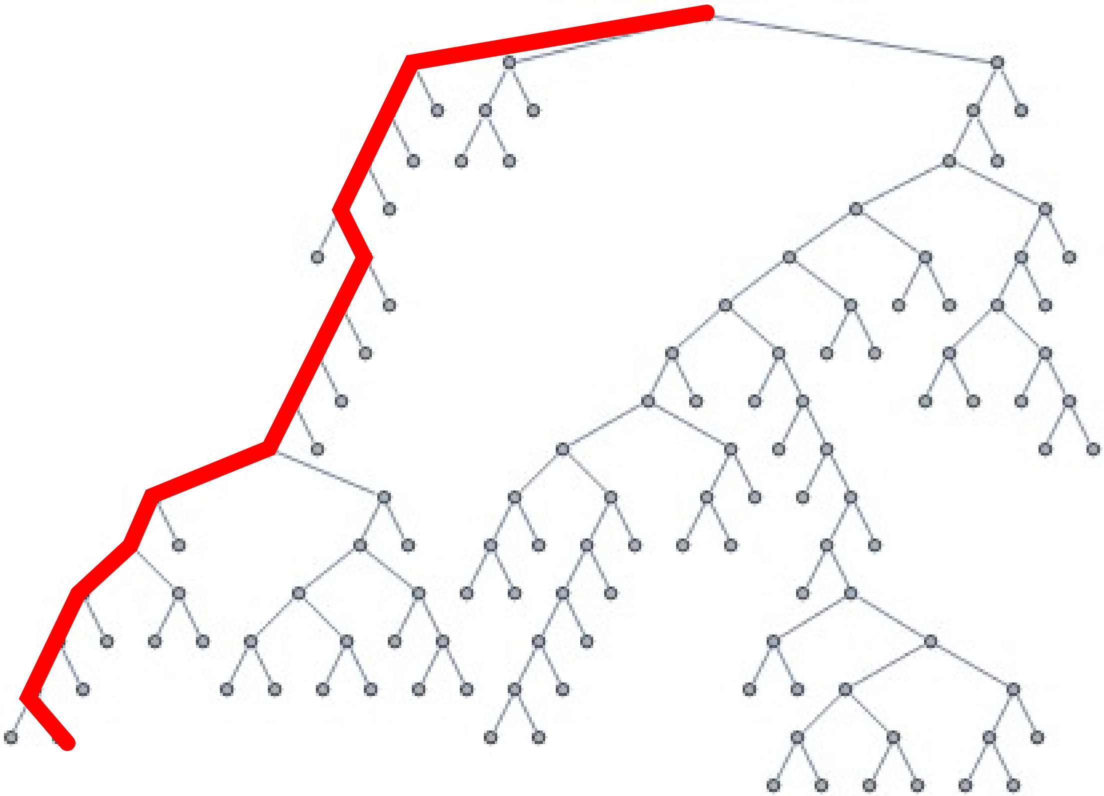
BFS vs. **DFS**



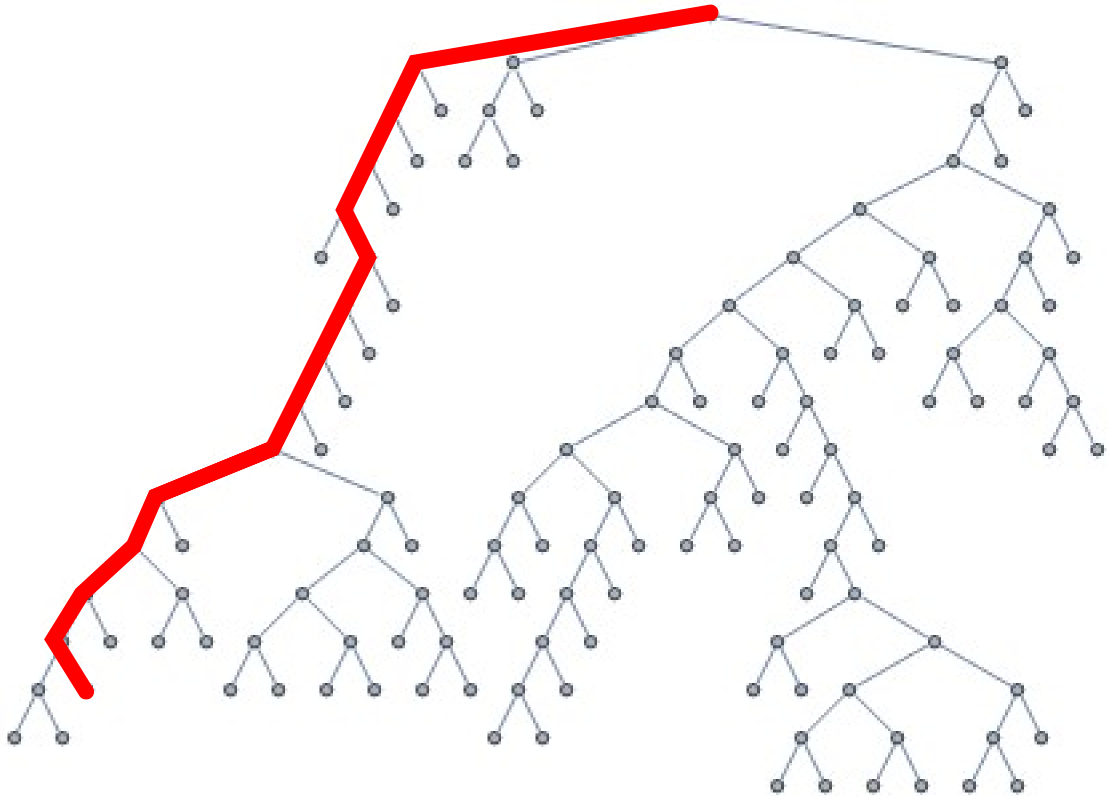
BFS vs. **DFS**



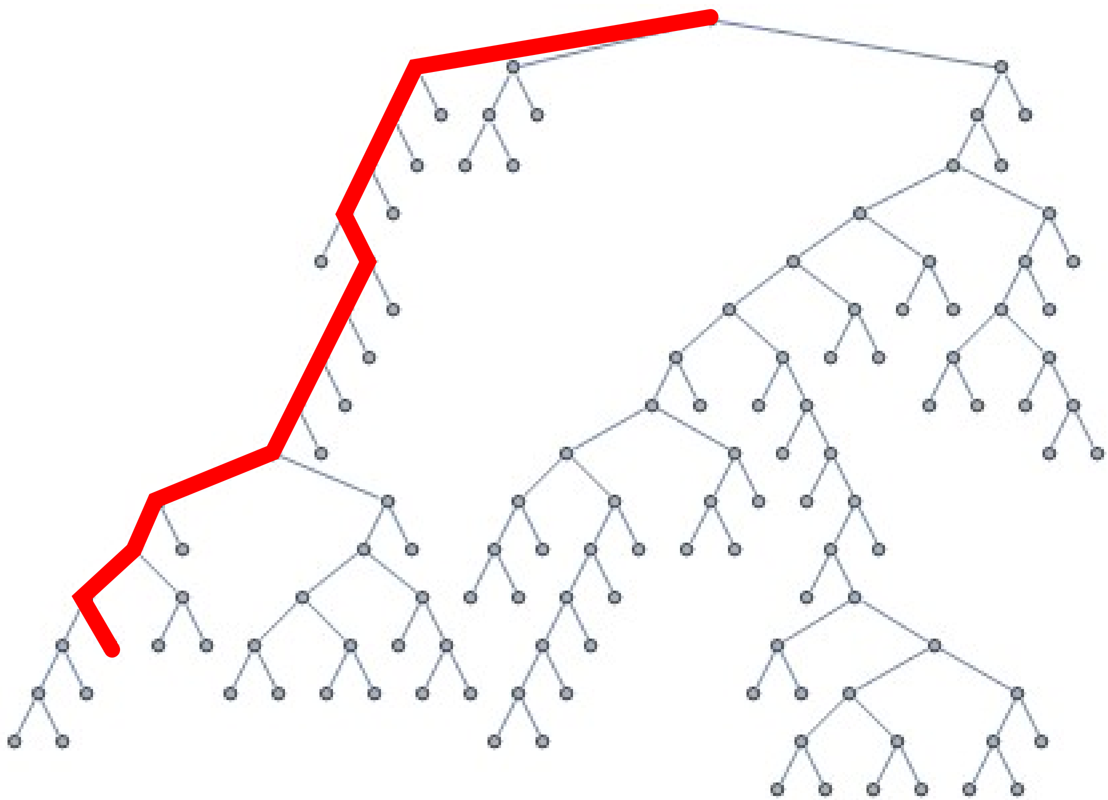
BFS vs. **DFS**



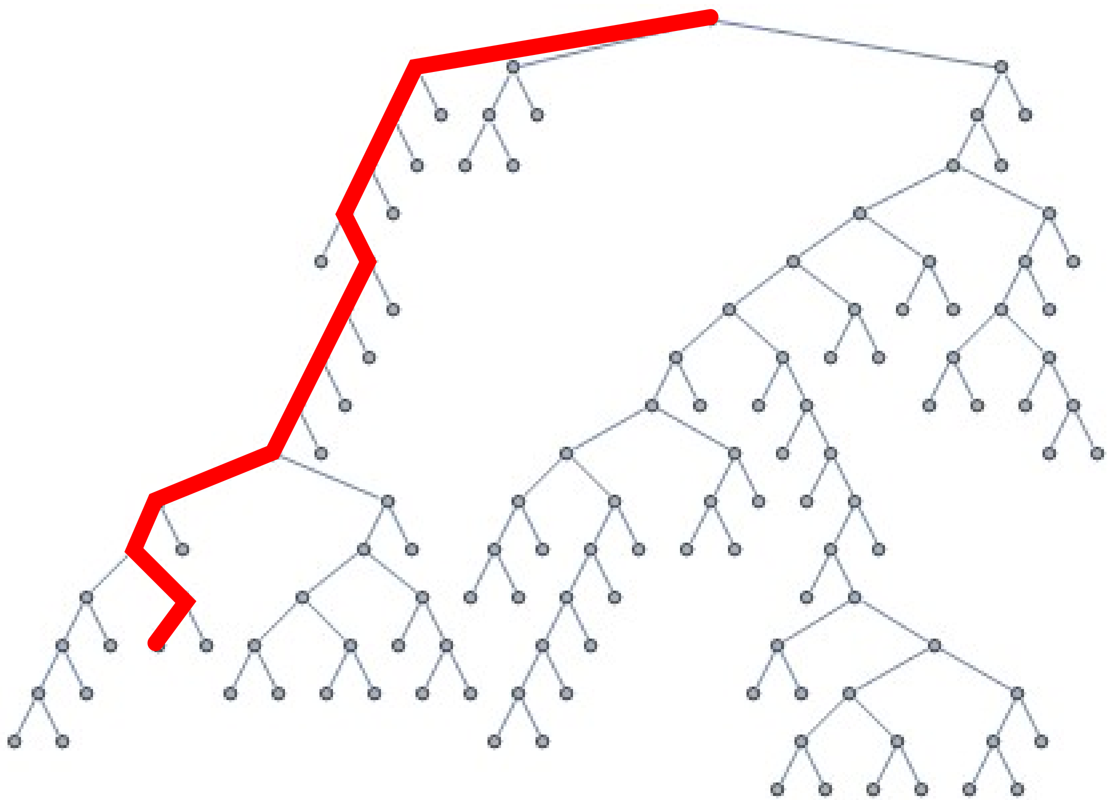
BFS vs. **DFS**



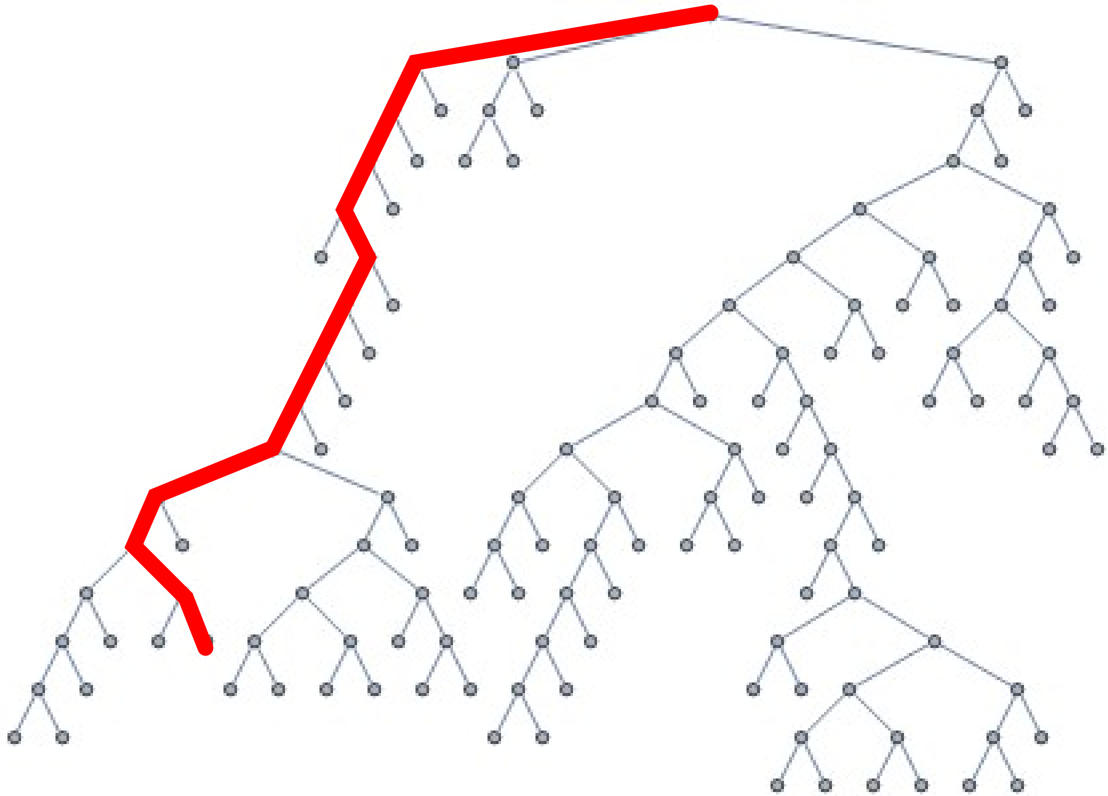
BFS vs. **DFS**



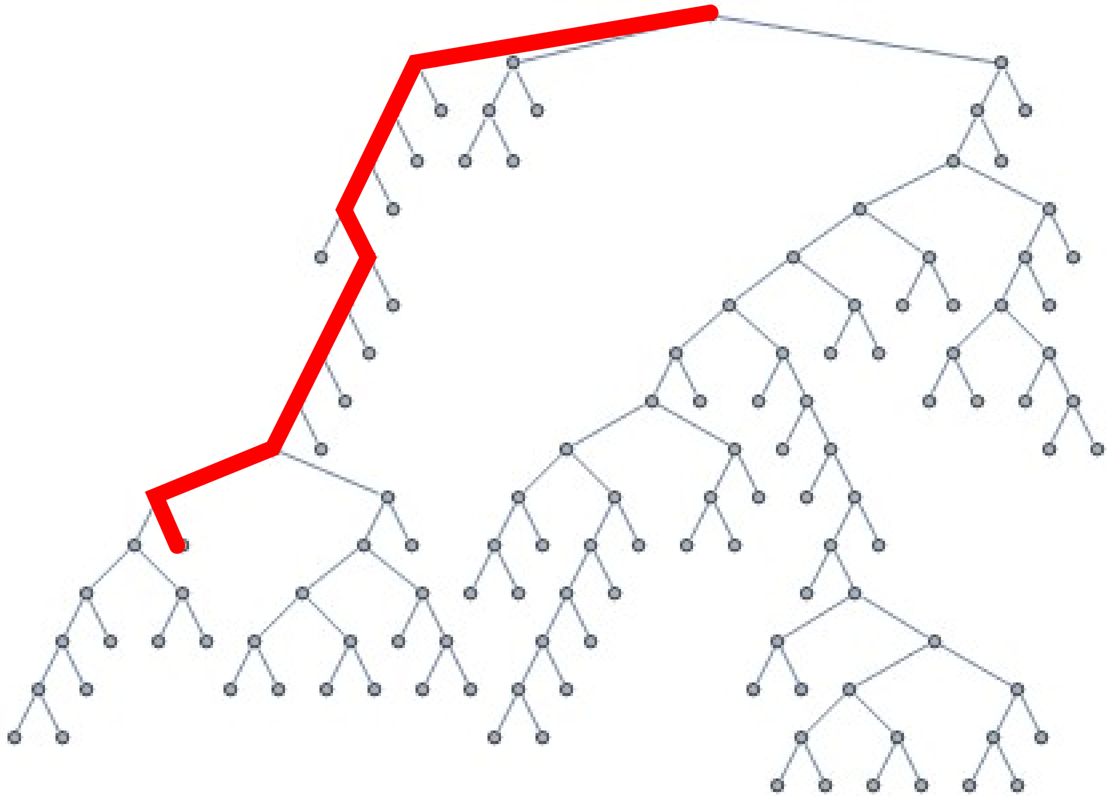
BFS vs. **DFS**



BFS vs. **DFS**



BFS vs. **DFS**



Analysis of search strategies

- Strategies are evaluated along the following criteria:
  - **Completeness:** does it always find a solution if one exists?
  - **Optimality:** does it always find a least-cost solution?
  - **Time complexity:** number of nodes generated
  - **Space complexity:** maximum number of nodes in memory
- Time and space complexity are measured in terms of
  - ***b****:* maximum branching factor of the search tree
  - ***d****:* depth of the optimal solution
  - ***m***: maximum length of any path in the state space (may be infinite)

Properties of breadth-first search

- **Complete?**
  Yes (if branching factor b is finite)
- **Optimal?**
  Yes – if cost = 1 per step
- **Time?**
  Number of nodes in a b-ary tree of depth d: $O(b^d)$
  (d is the depth of the optimal solution)
- **Space?**
  $O(b^d)$

- Space is the bigger problem (more than time)

Properties of depth-first search

- **Complete?**

  Fails in infinite-depth spaces, spaces with loops

  Modify to avoid repeated states along path

  → complete in finite spaces

- **Optimal?**

  No – returns the first solution it finds

- **Time?**

  Could be the time to reach a solution at maximum depth m: $O(b^m)$

  Terrible if m is much larger than d

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

- **Space?**

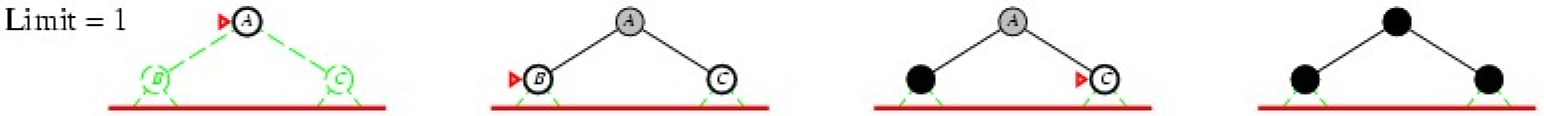
  $O(bm)$, i.e., linear space!

Iterative deepening search

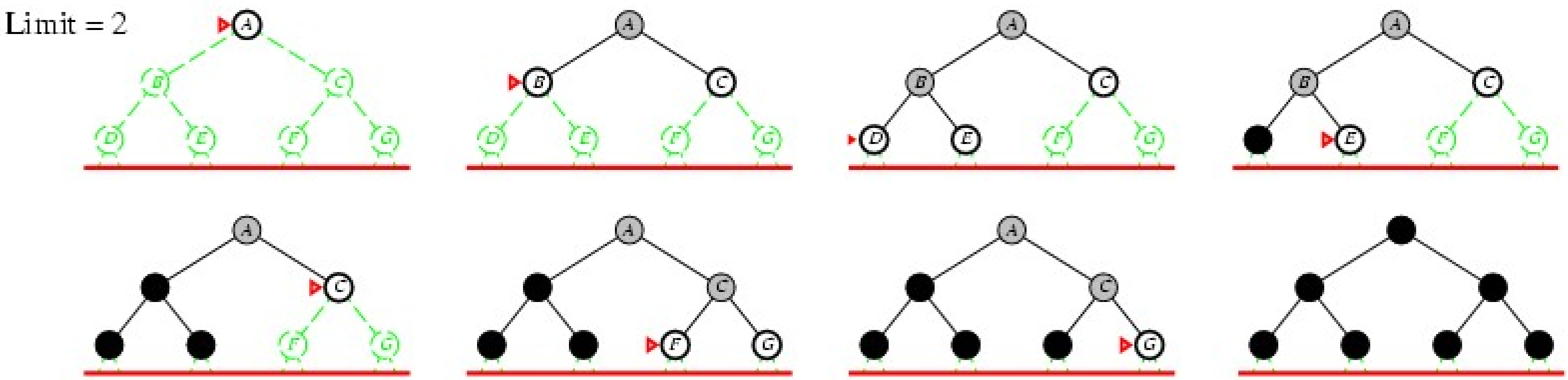
- Use DFS as a subroutine
  1. Check the root
  2. Do a DFS searching for a path of length 1
  3. If there is no path of length 1, do a DFS searching for a path of length 2
  4. If there is no path of length 2, do a DFS searching for a path of length 3…

Iterative deepening search

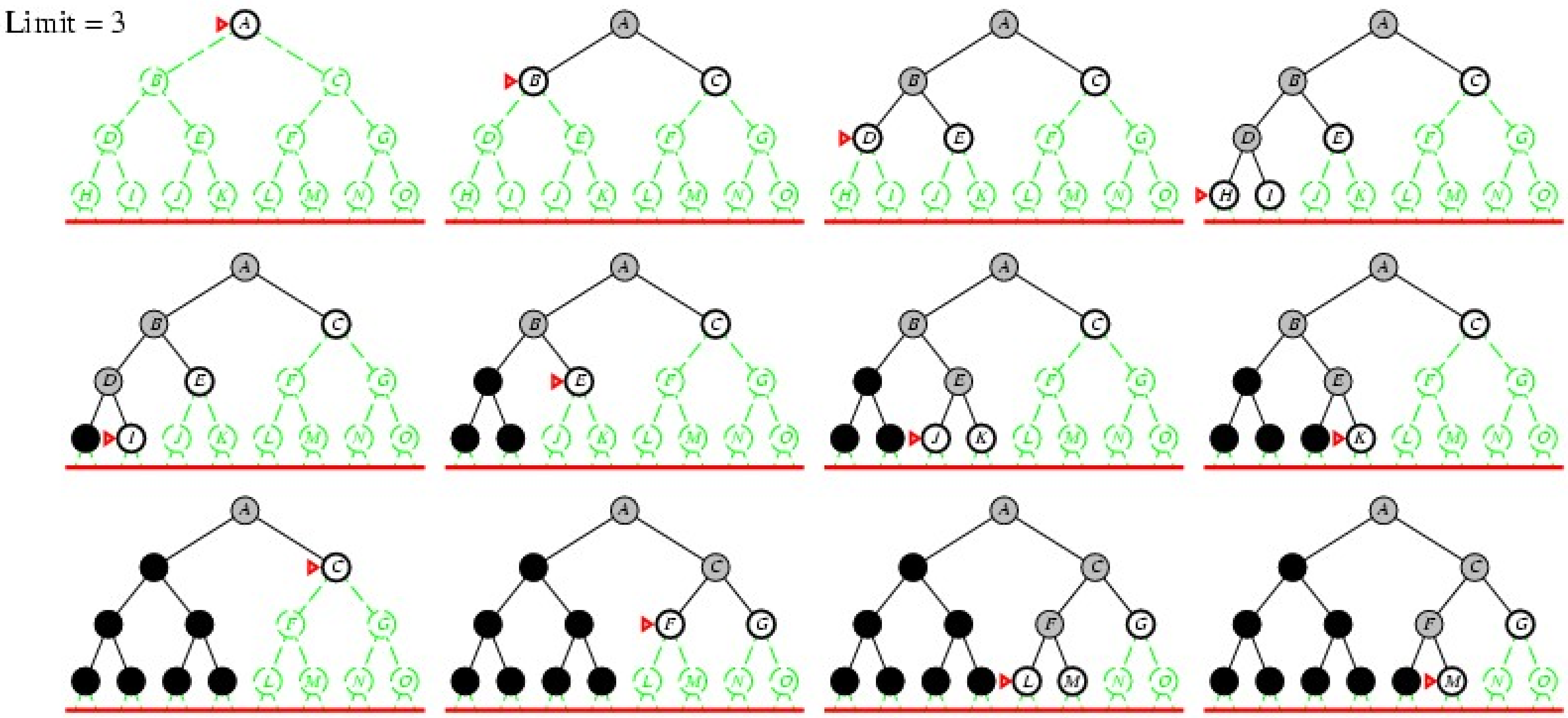
Iterative deepening search



Iterative deepening search



Iterative deepening search



Properties of iterative deepening search

- **Complete?**
  Yes
- **Optimal?**
  Yes, if step cost = 1
- **Time?**
  $(d+1)b^0 + d\, b^1 + (d-1)b^2 + \ldots + b^d = O(b^d)$
- **Space?**
  $O(bd)$