Cost of Breadth-first Search (BFS)

- b – max branching factor (infinite?)
- d – depth to shallowest goal node
- m – max length of any path (infinite?)

- Analysis of space and time performance…
  - g(n) is O(f(n)) means that

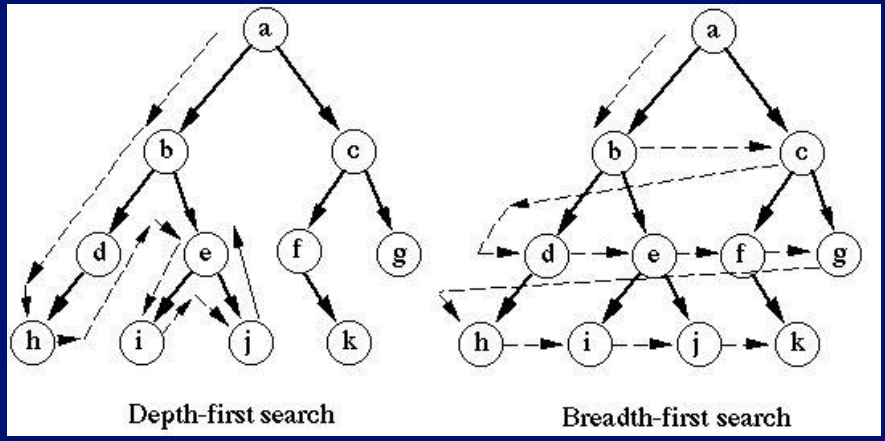
    there are positive constants c and n_0 such that

    $0 \leq g(n) \leq cf(n)$ for all $n \geq n_0$

Cost of Depth-first Search

- time complexity = $O(b^m)$
  - length of longest path anchors upper bound
- space complexity = O(bm)

- What's an example that highlights difference in performance between BFS and DFS?



Iterative Deepening

Essentially DFS with a depth limit

Why?

- Remember the space complexity, O(bm) and time complexity, $O(b^m)$ of DFS?
- So... limit m to be small
- But what if m is smaller than d?
  - we'll never find solution
- So… increment depth limit starting from 0

Bidirectional Search

Search from goal to start

Search from start to goal

Link – Procedure Graph

Bidirectional Search

- Do you always know the predecessors of a goal state?
- Do you always know the goal state?
  - 15-puzzle Link
  - Puzzle Game - Cannibals & Missionaries Link
  - Chess

Avoid Repeated States

How might you create repeated states in 8-puzzle?

How can you detect repeated states?

State Space

Dijkstra Algorithm

What about preserving lowest cost path among repeated states?

- uniform-cost search (shortest path) and

  BFS w/ constant step costs

Interesting problems

Exercise 3.9:

- 3 cannibals and 3 missionaries and a boat that can hold one or two people are on one side of the river. Get everyone across the river.
- 8-puzzle and 15-puzzle, invented by Sam Loyd in 1870s. Think about search space.
- Rubik's cube Link
- Traveling Salesman Problem (TSP) Link

Farmer, Fox, Goose, and Bag of Corn

Puzzle concerning a farmer, a fox, a goose, and a bag of corn.

The farmer wants to get all of this stuff across a river. His boat will only hold himself, and one of the other three items. The fox will eat the goose if the farmer is not present, and the goose will eat the corn if the farmer is not present. How can the farmer transport all three items across the river?

Chapter 4 – Informed Search

INFORMED?

- Uses problem-specific knowledge
  beyond the definition of the problem itself
  - selecting best lane in traffic

Best-first Search

Use an evaluation function to select node to expand

- f(n) = evaluation function = expected distance to goal
- select the node that minimizes f(n)
- but if we knew the ‘best’ node to explore it wouldn’t be search!!!
- Let’s use heuristics for our evaluation functions

Heuristics

A function, h(n), that estimates cost of cheapest path from node n to the goal

- h(n) = 0 if n == goal node

A* (A-star) Search

Combine two costs

- f(n) = g(n) + h(n)
  - g(n) = cost to get to n
  - h(n) = cost to get to goal from n

Minimize f(n)

A* Algorithm

Satisfies 2 conditions

- – h(n) is admissible (never overestimates the actual cost)
  – h(n) is monotonic [if there is an edge [n,n1], then h(n) is not greater than distance(n,n1)+ h(n1)]

- Proof of optimality?

-

A* is Optimal

We must prove that A* will not return a suboptimal goal or a suboptimal path to a goal

- Let G be a suboptimal goal node
  - f(G) = g(G) + h(G)
  - h(G) = 0 because G is a goal node
  - f(G) = g(G) > C* (because G is suboptimal)

A* is Optimal

We must prove that A* will not return a suboptimal goal or a suboptimal path to a goal

- Let G be a suboptimal goal node
  - f(G) = g(G) + h(G)
  - h(G) = 0 because G is a goal node
  - f(G) = g(G) > C* (because G is suboptimal)
- Let n be a node on the optimal path
  - because h(n) does not overestimate
  - f(n) = g(n) + h(n) <= C*
- Therefore f(n) <= C* < f(G)
  - node n will be selected before node G