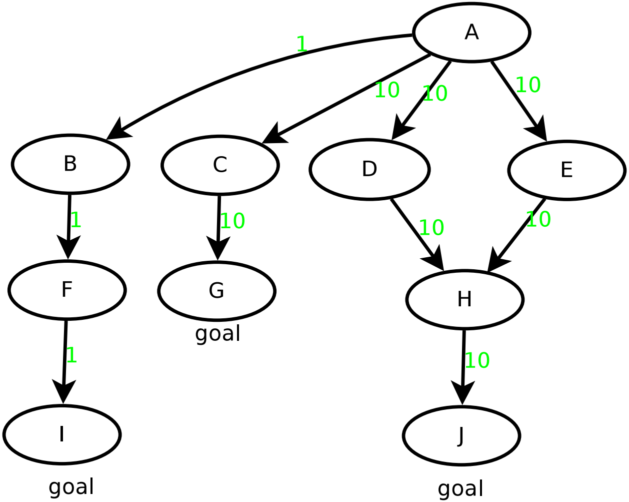
CS444

Nathan Sprague

September 5, 2012

Consider This Problem...

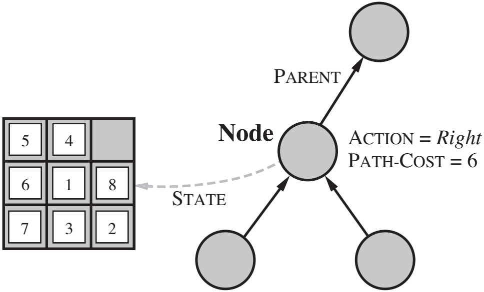


Uniform Cost Search

```
function UNIFORM-COST-SEARCH(problem) returns a solution, or failure
  node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  frontier ← a priority queue ordered by PATH-COST, with node as the only element
  explored ← an empty set
  loop do
     if EMPTY?(frontier) then return failure
     node ← POP(frontier)  /* chooses the lowest-cost node in frontier */
     if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
     add node.STATE to explored
     for each action in problem.ACTIONS(node.STATE) do
        child ← CHILD-NODE(problem, node, action)
        if child.STATE is not in explored or frontier then
           frontier ← INSERT(child, frontier)
        else if child.STATE is in frontier with higher PATH-COST then
           replace that frontier node with child
```

Figure 3.13 Uniform-cost search on a graph. The algorithm is identical to the general graph search algorithm in Figure **??**, except for the use of a priority queue and the addition of an extra check in case a shorter path to a frontier state is discovered. The data structure for *frontier* needs to support efficient membership testing, so it should combine the capabilities of a priority queue and a hash table.

Reminder... Search Nodes



What We Really Want...

$f(n) = g(n) + r(n)$
Where,
$g(n)$ = cost to reach node n.
$r(n)$ = minimum cost to reach the goal, starting at n.

What We Settle For...

$f(n) = g(n) + h(n)$
Where,
$g(n)$ = cost to reach node n.
$h(n)$ = *Estimate* of the minimum cost to reach the goal, starting at n.

Using $f(n)$ to select nodes from the frontier gives us A^*.

A*

- UCS
  - Complete
  - Optimal
- A*
  - Complete
  - Optimal (If h(n) meets certain conditions)
  - Potentially much faster than UCS

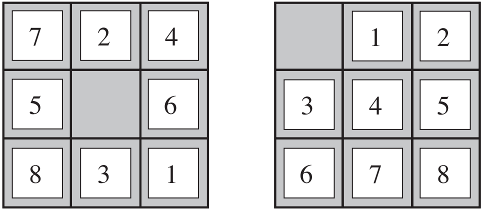
A*

- A* is optimal for TREE-SEARCH if $h(n)$ is admissible.
- A* is optimal for GRAPH-SEARCH if $h(n)$ is consistent.

- Admissible
  - $h(n)$ never overestimates the true cost to goal.
- Consistent
  - $h(n) <= c(n, a, n') + h(n')$

Examples...

8-Puzzle



Start State   Goal State