0.1 Algorithm A Search

Algorithm A Search is different from other best first searches in two ways because it is interested in coming as close as possible to finding the *cheapest* path to a goal node.

- The heuristic function which it uses to take nodes off of the open list is:
  $f(n) = g(n) + h(n)$
  where:

  $g(n)$ is an estimate of the actual cost of the cheapest path from the start node to the node n – here we will use the minimal cost path found so far. So with will always be the case that $g\prime(n) \leq g(n)$, and

  $h(n)$ is an estimate of the actual cost of the cheapest path from n to a goal node – here we will use a heuristic function which we make up to estimate this cost.

- The algorithm itself must be capable of keeping track of the cheapest path found so far to a node. Note that the first path from start to a node may not be the cheapest. A better way may be found later on.

The algorithm itself follows. Before we get started, we will need to associate a number of properties with each node that we put in the graph:

parent – parent of the node along the path yielding the cheapest path from start.

g – value/cost of cheapest path from start found so far.

h – value of estimate of the cost of path from the node to the goal node (value of the state evaluation function).

f – g+h

children – list of successors – successors are in the form of a list of (child cost-of-going-from-parent-to-child) pairs.

Algorithm:
Start with S on Open – set up initial properties for S:

- parent = nil
- g = 0
- h = whatever the function returns for s
- f = g + h
- children = (not set yet)

Closed starts out empty

REPEAT until OPEN = empty (in which case we fail)

- choose a node from OPEN with lowest f value (call it BESTNODE)
- remove BESTNODE from OPEN
- put BESTNODE on CLOSED
- if BESTNODE is a goal – then succeed (you can return the path by traveling back through the parent pointers)
- Generate SUCCESSORS of BESTNODE (and install them on BESTNODE'S children list). You want to update each and add them to OPEN, but you must be careful since they may have already been generated – if they have, you must make sure that the path they are already on isn't better than the path they get from BESTNODE.
  FOR EACH SUCCESSOR

  1. if it has NOT been generated before (i.e., it is not already on OPEN or CLOSED) then add it to the graph:
     - put it on OPEN
     - set its parent=BESTNODE
     - set its g=g(BESTNODE) + cost of going from BESTNODE to the successor
     - set its h to whatever it is according to the state evaluation function
     - set its f to g+h
  2. if it HAS been generated before, then we have to see if its old g value is larger than the g value it gets from BESTNODE (= g(BESTNODE) + cost of going from BESTNODE to successor).

     If the value from BESTNODE is better, then
     - change successor's parent to BESTNODE
     - change its g value to (g(BESTNODE) + cost from BESTNODE to successor)
     - update its f value
     - repeat this step for each of the successor's children