0.1 General Graph Search Algorithm

INPUT :

START -- initial state

GOAL-TEST -- a predicate that taskes a state and returns non-nil if it is a goal state

SUCCESSORS -- returns a list of immediate successors of a state -- this function returns a set of triples: (successor operator cost) where

successor is the immediate successor

operator is the operator which generates successor from the given state

cost is the cost of generating successor

LOCAL :

OPEN LIST -- a list of states that have not been explored/ looked at/expanded -- these are states which have been entered into the graph, but have not been checked

CLOSED LIST -- a list of states that have already been expanded I.e., they have been entered into the graph and their successors have been generated

EXPL -- state we are currently exploring

PROPERTIES ASSOCIATED WITH STATES : Each state has

Parent : pointer to state the state is a successor of

Path-Cost : cost of path from start to state (going through parent)

Operator : operator applied to parent to generate state

Children : the list returned from successor function when it was applied to state -- contains a triple for each immediate successor containing the successor, the operater used to generate successor, and the cost involved in generating successor

ALGORITHM :

```
OPEN-LIST = (START)

Initialize Parent(START) = nil; Path-cost(START) = 0

EXPL = START

LOOP until (EXPL is a GOAL-STATE (using GOAL-TEST predicate) and
succeed returning the path by reading back through the parent property
of each state) OR (OPEN is empty and fail)

  1. remove EXPL from OPEN, and put it on the CLOSED-LIST
  2. compute the successors of EXPL and set the children(EXPL) to be the
     list returned from the successors function
  3. For each successor NOT already on OPEN or CLOSED
     (a) Put the successor on OPEN and call the triple associated with
         the current successor (s1 o1 c1)
     (b) set Parent(s1) to EXPL
     (c) set the Operator(s1) to o1
     (d) set the Path-cost(s1) to (Path-cost(EXPL) + c1)
  4. For each successor already on OPEN or CLOSED
     (a) call the triple associated with the current successor (s1 o1
         c1)
     (b) If Path-cost(s1) $>$ (Path-cost(EXPL) + c1)
         % need to update to point to new parent and new path cost
           i. set Parent(s1) to EXPL
          ii. set the Operator(s1) to o1
         iii. set the Path-cost(s1) to (Path-cost(EXPL) + c1)
          iv. If children(s1) $\neq$ nil, for each element of children(s1) do
              % potentially need to update all children and their
              children....
              A. If Parent(child) = s1 then
                 • update its Path-cost by recalculating it based on new
                   Path-cost(s1)
                 • update its children if they exist
              B. Else child is pointing to another parent, check to see if
                 you just found a cheaper way, if so, update the child and
                 all of its children just as was done for s1 with EXPL as a
                 parent
  5. set EXPL to some member of the OPEN-LIST and Go Loop
```