## 0.1 VERSION 1: Simple Graph Search Algorithm

The following gives the simplest view of a search algorithm.

**INPUT** :

**START** -- initial state

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

**SUCCESSORS** -- returns a list of immediate successors of a state -- this function is responsible for applying the applicable operators. This function takes a state and for each operator it matches the preconditions of the operator against the state. If the match is successful, it generates another state by applying the associated action. Note that in general more than one operator may be applicable and a given operator may match the current state in more than one way.

**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

<space>

**EXPL** -- state we are currently exploring

**ALGORITHM** :

OPEN-LIST = (START)

EXPL = nil

LOOP until (EXPL is a GOAL-STATE (using GOAL-TEST predicate) and succeed) OR (OPEN is empty and fail)

- set EXPL to some member of the OPEN-LIST and remove it from OPEN

<space>

- compute the successors of EXPL and put them on the OPEN-LIST

<larger space>

- Go Loop