```
DepthLimitedSearch( initialState, cutOff )
    stateQueue ← MakeQueue( initialState )
    loop until stateQueue is empty
        state ← pop stateQueue
        if GoalStack( state ) == NIL or GotCutOff( state, cutOff )
            OutputProof( state )
        else
            stateQueue ← Append( stateQueue, Expand( state ) )  ;; Notice they are APPEND'ed

GotCutOff( state, cutoff )
    application of construction op exceeds cutOff.construction
    depth of search exceeds cutOff.searchDepth

Expand( state )
    ExhaustiveForwardChain( state )
    if brute force flag is true
        BruteForceConstruction( state )
    else
        Construction( state )
        BackwardChain( state )

BackwardChain( state )
    successors ← NIL
    if First(GoalStack( state ) unifies with a fact in state
        newState ← Copy( state )
        pop GoalStack( state )
        add newState to successors
    for each diagrammatic schema DS such that
      its configuration totally matches problem configuration in state
        for each deduction in DS such that
          its conclusion unifies First(GoalStack( state )) and
          all conditions are quantitatively true in state
        newState ← Copy( state )
        pop GoalStack( newState )
        push conditions to GoalStack( newState )
        add newState to successors
    return successors
```

```
CONSTRUCTION( state )
    successors ← NIL
    for each usefulDS wrt FIRST(GOALSTACK( state ))
        for each partialMatch for usefulDS
            for each tmpState in GENERALCONSTRUCTION( state, partialMatch, 1 )
                for each deduction in usefulDS such that
                  all conditions of deduction are quantitatively true in tmpState
                    newState ← Copy( tmpState )
                    pop GoalStack( newState )
                    push conditions to GOALSTACK( newState )
                    add tmpState to successors
    return successors

GENERALCONSTRUCTION( state, partialMatch, numConstructionOp )
    constructions ← NIL
    for each application of constructionOp that reduces the number of unbound points
        create a new state tmpState that has new problem configuration,
          the same facts, the same goal stack,
          and constructionOp cons'ed
        if any of partialMatch still remains unbound then
            unless numConstructionOp == MAX_CONSTRUCTION_OP
                add GENERALCONSTRUCTION( tmpState, partialMatch, numConstructionOp +1 )
                  to constructions
        else
            add tmpState to constructions
    return constructions

BRUTEFORCECONSTRUCTION( state )
    successors ← NIL
    for each application of constructionOp
        create a new state tmpState that has new problem configuration,
          the same facts, the same goal stack,
          and constructionOp cons'ed
        add tmpState to successors
    return successors
```