Tree/Graph Search

Basis of the search

Order in which nodes are evaluated and expanded

Determined by Two Lists

Open List
List of unexpanded nodes

CLOSED
List of expanded nodes

Depth First Search

1. Put start node on OPEN
2. If OPEN empty, exit with failure
3. Remove the **topmost** node, n, from OPEN and put it on CLOSED
4. If depth has reached the depth bound, go back and choose another node
5. Expand n:
   (a) Put all successors (in no particular order) on the **top** of OPEN
   (b) Provide a pointer back to n
   (c) If any is a solution, note it as such
   (d) If any successor node is a ***dead end*** remove it from OPEN
6. Repeat

Depth First Search

The placement of the successors on the top:
Uninformed Search

The search proceeds depth first because
the first nodes to go on the stack
are the first to be expanded
First In First Out (FIFO)

Infinite loops are prevented by: **Depth Bound**

If the node is examined and it is known that it can't lead to a solution: **Dead End**

Relation to Backtracking

Backtracking is Depth First Search

The only difference is that
instead of generating all the nodes at once,
only one of the successors is generated

As soon as a dead end or depth limit is reached
its parent is looked at
and its next successor generated

Relation to Breadth First

1. Put start node on OPEN
2. If OPEN empty, exit with failure
3. Remove the **topmost** node, n, from OPEN and put it on CLOSED
4. If depth has reached the depth bound, go back and choose another node
5. Expand n:
   (a) Put all successors (in no particular order) on the **bottom** of OPEN
   (b) Provide a pointer back to n
   (c) If any is a solution, note it as such
   (d) If any successor node is a *dead end* remove it from OPEN
6. Repeat

Relation to Breadth First

The only difference is
that instead of putting the successors at the top of OPEN
they are put at the bottom

The first nodes to be expanded
are the last ones to be expanded
First In Last Out (FILO)

This means that all the nodes are expanded
before their successors are evaluated

Depth First: Relation to Clips Program

The **Agenda** corresponds to elements on the OPEN list

The default strategy is **Depth First**

So, given the same Salience, new rules are placed on top of the agenda

Without any further control mechanism, an infinite search easily results

How could the list of rules be modified so that they would find a solution

Add a depth limit

Breadth First: Relation to Clips Program

The search was made breadth first by adding the requirement that all nodes of a given depth be expanded first.

All successors are of the next depth
This means that all successors come afterwards

The adgenda represents those nodes on the OPEN list which are on the given depth

The other nodes on the OPEN list
(Those of next depth)
Do not appear on the agenda until that depth is to be evaluated

The order evaluation at a given depth
is arbitrary
(Thus can use the original depth first strategy)

The extra mechanism of the clips program
(i.e. the extra rules)
enabled controling the order of firing of rules

Best First Search

A Comprimise between

Breadth First and Depth First

Heuristic Evaluation Function

$f(n)$

The promise of the node

Includes Knowledge of the System

Nodes are selected on the basis of f(n)

The lowest f(n) is evaluated first

Best First Algorithm

1. Put Start node on OPEN list
2. If OPEN is empty, exit with failure; No solution exists
3. Remove *best* node,n, from OPEN determined by f(n).
4. Place n on CLOSED list and expand, generating all its successors; Each successor has a pointer back to n.
5. If any successors are goal nodes:
   (a) Trace Solution back through parents
6. For Every Successor, n':
   (a) Calculate $f(n')$
   (b) If n' not on the OPEN or CLOSED
      i. Add to OPEN list
      ii. Attach Pointer from n to n', Assign $f(n')$ to n'
   (c) If n' is on OPEN or CLOSED
      i. Compare $f(n')$ of each
      ii. If worse, discard
      iii. Otherwise, replace.
      iv. If on CLOSED, move to OPEN
7. Repeat