Artificial Intelligence
Basic Search

1 A scheme for specifying graphs and goals

1.1 The graph specified by Seed

The following scheme provides a convenient way to specify a graph. It does not capture all graphs. But it captures plenty — at least those with branching factor 2 (i.e. where every node has exactly two children).

A graph over the set of nodes \{1, 2, 3, \ldots\} can be specified by a positive integer Seed, with arcs \((N, M)\) as follows:

\[
\text{arc}(N, M, \text{Seed}) :- M \equiv N \times \text{Seed}.
\]

\[
\text{arc}(N, M, \text{Seed}) :- M \equiv N \times \text{Seed} + 1.
\]

E.g. Seed=3 yields arcs (1,3), (1,4), (2,6), (2,7), (3,9), (3,10), etc.

1.2 The goal nodes specified by Target

Let us agree that a positive integer Target specifies as goal nodes multiples of Target — i.e. Target, 2*Target, 3*Target, ...

\[
\text{is-goal}(N, \text{Target}) :- 0 \equiv N \mod \text{Target}.
\]

E.g. Target=13 yields goals nodes 13, 2*13, 3*13, etc.

2 Searching Graph(Seed) for Goal(Target)

The generic (frontier) search algorithm from chapter 4 of Computational Intelligence can be adapted to the case above of Seed, Target as follows:

\[
\begin{align*}
\text{search}([\text{Node}|\_],\_\_\_\_, \text{Target}) & :- \text{is-goal} (\text{Node}, \text{Target}). \\
\text{search}([\text{Node}|\text{FRest}],\text{Seed}, \text{Target}) & :- \\
& \text{setof} (X, \text{arc}(\text{Node}, X, \text{Seed}), \text{Children}), \\
& \text{add-to-frontier} (\text{Children}, \text{FRest}, \text{FNew}), \\
& \text{search} (\text{FNew}, \text{Seed}, \text{Target}).
\end{align*}
\]

Note that we need not worry about \text{arc}(\text{Node}, X, \text{Seed}) failing, so long as \text{Node} and \text{Seed} are positive integers.

What remains to be specified is \text{add-to-frontier}(\text{Children}, \text{FRest}, \text{FNew}). Before turning to that, let us modify \text{search} slightly in order to return the node found (as its last argument).

Question. How? (Try answering this before proceeding to the solution below.)

\[
\begin{align*}
\text{search}([\text{Node}|\_],\_\_\_\_, \text{Target}, \text{Node}) & :- \text{is-goal} (\text{Node}, \text{Target}). \\
\text{search}([\text{Node}|\text{FRest}],\text{Seed}, \text{Target}, \text{Found}) & :- \\
& \text{setof} (X, \text{arc}(\text{Node}, X, \text{Seed}), \text{Children}), \\
& \text{add-to-frontier} (\text{Children}, \text{FRest}, \text{FNew}), \\
& \text{search} (\text{FNew}, \text{Seed}, \text{Target}, \text{Found}).
\end{align*}
\]
2.1 Breadth-first search

Breadth-first search falls out if \texttt{add-to-frontier} forms \texttt{FNew} by appending \texttt{Children} to the end of \texttt{FRest}

\begin{verbatim}
add-to-frontier(C, [], C).
add-to-frontier(C, [N|Rest], [N|FNew]) :-
    add-to-frontier(C, Rest, FNew).
\end{verbatim}

Note that breadth-first treats the frontier as a “first-in, first out” queue.

\textbf{Lab}

Make sure you understand why by tracing queries such as

\begin{verbatim}
| ?- search([1],3,13,Found).
\end{verbatim}

(A useful debugging tool here is \texttt{spy(search)}, followed by 1’s.)

\textbf{Question}. At what rate does the frontier grow as the depth of a node searched increases?

2.2 Depth-first search

Depth-first search falls out if \texttt{add-to-frontier} forms \texttt{FNew} by appending \texttt{Children} in front of \texttt{FRest}

\begin{verbatim}
add-to-frontier([], F, F).
add-to-frontier([N|Children], F, [N|FNew]) :-
    add-to-frontier(Children, F, FNew).
\end{verbatim}

Note that depth-first is “last-in, first out” — i.e. a stack.

\textbf{Lab}

Make sure you understand why by tracing queries such as

\begin{verbatim}
| ?- search([1],3,13,Found).
\end{verbatim}

No node will be found as the search will be restricted to the nodes $3, 3^2, 3^3, 3^4, \ldots$.

\textbf{Questions}.

1. Switch the order of the rules for \texttt{arc}, and replace \texttt{setof} with \texttt{bagof} (returning an unsorted list, with as many copies as there are solutions). Now, note what branch is explored.

2. At what rate does the frontier grow as the depth of a node searched increases?

3. How can we do a depth-first search without resorting to either \texttt{setof} or \texttt{bagof}?