A simple search exercise

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 \(\langle N, M \rangle\) as follows:

\[
\begin{align*}
\text{arc}(N, M, \text{Seed}) & :\quad M \text{ is } N \cdot \text{Seed}. \\
\text{arc}(N, M, \text{Seed}) & :\quad M \text{ is } N \cdot \text{Seed} + 1.
\end{align*}
\]

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, \(2 \times \text{Target}\), \(3 \times \text{Target}\), ...

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

E.g. Target = 13 yields goals nodes 13, \(2 \times 13\), \(3 \times 13\), etc.

2 Searching Graph(Seed) for Goal(Target)

The generic frontier search algorithm can be adapted to the case above of Seed, Target as follows:

\[
\begin{align*}
\text{search}([\text{Node}|\_], \_, \text{Target}) & :\quad \text{goal(\text{Node}, \text{Target})}. \\
\text{search}([\text{Node}|\text{FRest}], \text{Seed}, \text{Target}) & :\quad \\
& \quad \text{findall}(X, \text{arc(\text{Node}, X, \text{Seed})}, \text{Children}), \\
& \quad \text{add2frontier(Children, \text{FRest}, \text{FNew})}, \\
& \quad \text{search(\text{FNew}, \text{Seed}, \text{Target})}.
\end{align*}
\]

What remains to be specified is \text{add2frontier(Children, FRest, 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.)

\[
\begin{align*}
\text{search}([\text{Node}|\_], \_, \text{Target}, \text{Node}) & :\quad \text{goal(\text{Node}, \text{Target})}. \\
\text{search}([\text{Node}|\text{FRest}], \text{Seed}, \text{Target}, \text{Found}) & :\quad \\
& \quad \text{findall}(X, \text{arc(\text{Node}, X, \text{Seed})}, \text{Children}), \\
& \quad \text{add2frontier(Children, \text{FRest}, \text{FNew})}, \\
& \quad \text{search(\text{FNew}, \text{Seed}, \text{Target}, \text{Found})}.
\end{align*}
\]
2.1 Breadth-first search

Breadth-first search falls out if `add-to-frontier(Children,FRest,FNew)` forms `FNew` by appending `Children` to the end of `FRest`

```
add2frontier(C,[],C).
add2frontier(C,[N|Rest],[N|FNew]) :-
    add2frontier(C,Rest,FNew).
```

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

Lab

Make sure you understand why by tracing queries such as

```
| ?- search([1],3,13,Found).
```

(A useful debugging tool here is `spy(search)`, followed by 1’s.)

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 `add2frontier(Children,FRest,FNew)` forms `FNew` by appending `Children` in front of `FRest`

```
add2frontier([],F,F).
add2frontier([N|Children],F,[N|FNew]) :-
    add2frontier(Children,F,FNew).
```

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

Lab

Make sure you understand why by tracing queries such as

```
| ?- search([1],3,13,Found).
```

No node will be found as the search will be restricted to the nodes 3, 3², 3³, 3⁴, …

Questions.

1. Switch the order of the rules for `arc`. 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 `findall`?

2