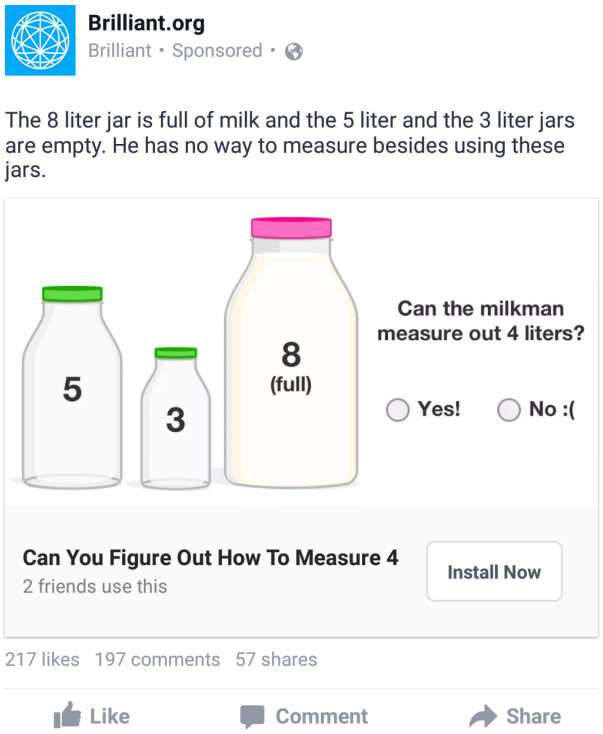
The 8-5-3 Milk Jug Problem

Charlene Chu - November 23, 2017

Problem



Challenge. Draw the state graph of this problem (with spilling allowed).

Solution

Assuming that spilling is allowed, here are all of the possible moves we can make. A move {*x*, *y*} means we are pouring milk from jug *x* to *y*, where jug 1, 2, 3 represents the 8-, 5-, 3-litre jug, respectively, and jug 4 is the spill/garbage jug. Note that we cannot pour back from the spill/garbage jug. That is, think of the spill/garbage jug as a sink, where once you pour milk into it, it goes down the drain and you can never retrieve it.

```
moves = Select[Join @@ Table[{x, y}, {x, 3}, {y, 4}], #[[1]] ≠ #[[2]] &]
```

```
{{1, 2}, {1, 3}, {1, 4}, {2, 1}, {2, 3}, {2, 4}, {3, 1}, {3, 2}, {3, 4}}
```

```
(* moves=
 Select[Join[Subsets[{1,2,3,4},{2,2}],Reverse[Subsets[{1,2,3,4},{2,2}],2]],#[[1]]≠4 &] *)
```

Here are all of the possible states. A state {*a*, *b*, *c*, *d*} means that there are *a*, *b*, *c*, *d* litres of milk in the 8-, 5-, 3-litre, spill jug, respectively. Having a spill/garbage jug is like having another 8-litre jug in which we can empty the contents of the other jugs into it, since we are starting with 8 litres of milk.

```
states = Solve[{a + b + c + d == 8, 0 ≤ a ≤ 8, 0 ≤ b ≤ 5, 0 ≤ c ≤ 3, 0 ≤ d ≤ 8},
   {a, b, c, d}, Integers] /. (_ → α_) :↦ α;
states // Short
```

```
{{0, 0, 0, 8}, {0, 0, 1, 7}, {0, 0, 2, 6}, {0, 0, 3, 5},
 {0, 1, 0, 7}, <<111>>, {7, 0, 0, 1}, {7, 0, 1, 0}, {7, 1, 0, 0}, {8, 0, 0, 0}}
```

Given a move and the current state, *f*[move, state] returns the state after making a move.

```
f[move_, state_] := Module[{x, y, max = {8, 5, 3, 8}, m, α, β, temp},
   x = move[[1]];
   y = move[[2]];
   m = max[[y]];
   α = state[[x]];
   β = state[[y]];
   temp = state;
   If[α ≠ 0 ∧ β ≠ m, If[α + β ≥ m,
     {temp[[x]], temp[[y]]} = {α + β - m, m}, {temp[[x]], temp[[y]]} = {0, α + β}], temp = Null];
   temp
  ];
```

We can create an edge from one state to another when we make a move.

```
createEdge[move_, state_] := state → f[move, state];
```

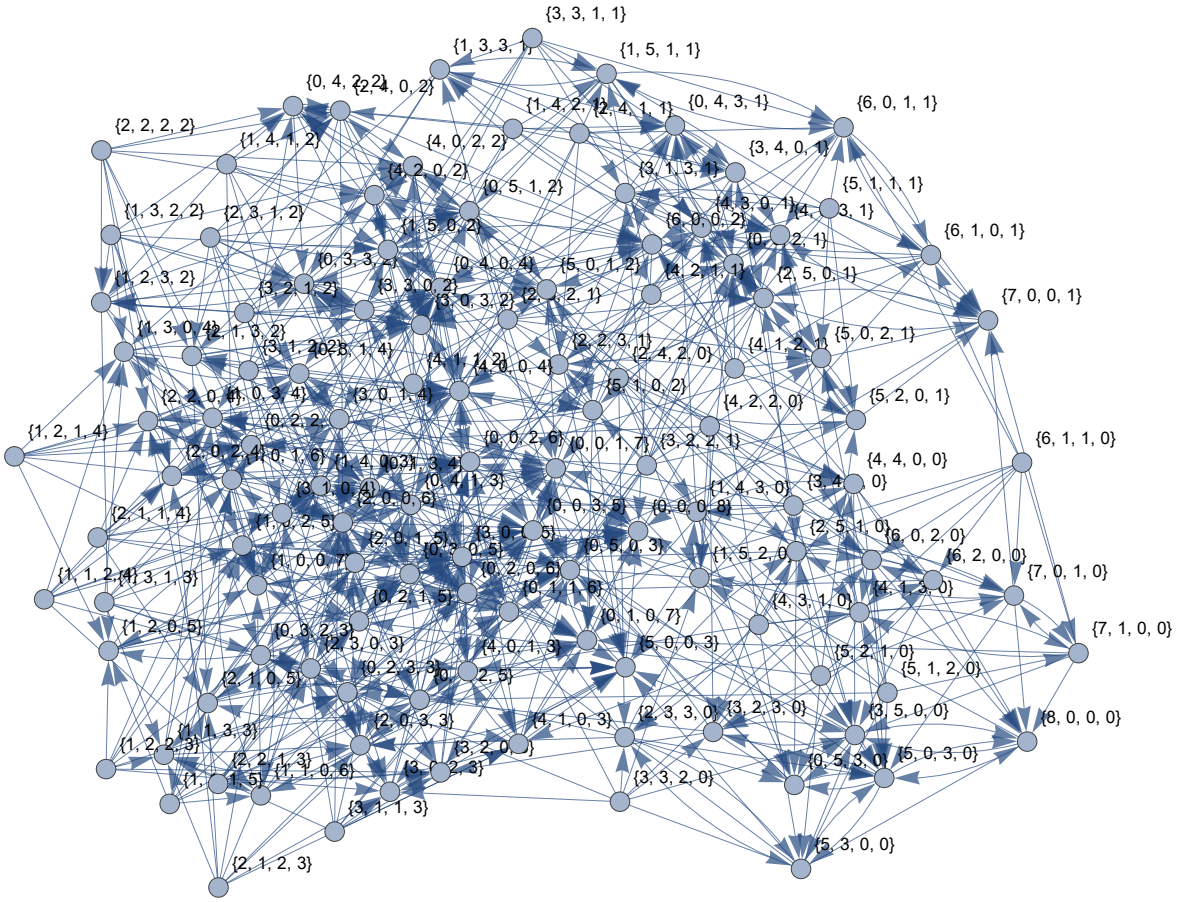
Here are all of the possible edges.

```
edges = Select[Flatten[Table[createEdge[move, state], {move, moves}, {state, states}]],
  ! MemberQ[#, Null] &];
edges // Short
```

```
{{1, 0, 0, 7} → {0, 1, 0, 7}, {1, 0, 1, 6} → {0, 1, 1, 6},
 <<755>>, {6, 1, 1, 0} → {6, 1, 0, 1}, {7, 0, 1, 0} → {7, 0, 0, 1}}
```

Here is the graph with all of the possible states.

```
graph = Graph[edges, VertexLabels → "Name", ImageSize → 620]
```



However, we are only interested in measuring 4 litres starting from the state {8, 0, 0, 0}. So, consider the start state and the end states. The end state is only when we have 4 litres in one of the first three jugs. The spill/garbage state does not count.

```
start = {{8, 0, 0, 0}}
```

```
{{8, 0, 0, 0}}
```

```
end = Select[states, MemberQ[Most[#], 4] &]; end // Short
```

```
{{0, 4, 0, 4}, {0, 4, 1, 3}, {0, 4, 2, 2}, {0, 4, 3, 1},
 {1, 4, 0, 3}, <<18>>, {4, 2, 2, 0}, {4, 3, 0, 1}, {4, 3, 1, 0}, {4, 4, 0, 0}}
```

Note that some of the end states cannot be achieved when we start at {8, 0, 0, 0}. We can use FindShortestPath to determine which states we need and which states we don't.

```
paths = Join @@ Table[FindShortestPath[graph, s, e], {s, start}, {e, end}];
paths // Short
```

```
{{{8, 0, 0, 0}, {3, 5, 0, 0}, {3, 2, 3, 0}, {6, 2, 0, 0},
  {6, 0, 2, 0}, {1, 5, 2, 0}, {0, 5, 2, 1}, {0, 4, 3, 1}, {0, 4, 0, 4}}, <<26>> }
```

Here, we can see that some paths are empty. This means that we cannot reach those end states. So, let's remove them from the set of end states.

```
newEnd = Delete[end, Position[paths, {}]]; newEnd // Short
```

```
{{0, 4, 0, 4}, {0, 4, 1, 3}, {0, 4, 2, 2}, {0, 4, 3, 1},
 {1, 4, 0, 3}, <<8>>, {4, 1, 3, 0}, {4, 2, 0, 2}, {4, 3, 0, 1}, {4, 4, 0, 0}}
```

Since we know the shortest path between the start state to the end states, we can create edges between each node in the path.

```
newEdges = DeleteDuplicates[
  Flatten[Table[Table[path[[i]] → path[[i + 1]], {i, Length[path] - 1}], {path, paths}]]];
newEdges // Short
```

```
{{8, 0, 0, 0} → {3, 5, 0, 0}, {3, 5, 0, 0} → {3, 2, 3, 0},
 <<37>>, {4, 0, 3, 1} → {4, 3, 0, 1}, {1, 4, 3, 0} → {4, 4, 0, 0}}
```

We are not interested in the contents of the spill/garbage jug, so let's clean up the states to exclude the spill jug. s

```
cleanedStart = start /. {a_, b_, c_, d_} :→ {a, b, c}
```

```
{{8, 0, 0}}
```

```
cleanedEnd = newEnd /. {a_Integer, b_Integer, c_Integer, d_Integer} :→ {a, b, c};
cleanedEnd // Short
```

```
{{0, 4, 0}, {0, 4, 1}, {0, 4, 2}, {0, 4, 3}, {1, 4, 0}, {1, 4, 3},
 <<5>>, {4, 0, 3}, {4, 1, 0}, {4, 1, 3}, {4, 2, 0}, {4, 3, 0}, {4, 4, 0}}
```

```
cleanedEdges = newEdges /. ({a_, b_, c_, d_} → {e_, f_, g_, h_}) :→ ({a, b, c} → {e, f, g});
cleanedEdges // Short
```

```
{{8, 0, 0} → {3, 5, 0}, {3, 5, 0} → {3, 2, 3},
 {3, 2, 3} → {6, 2, 0}, <<36>>, {4, 0, 3} → {4, 3, 0}, {1, 4, 3} → {4, 4, 0}}
```

Here is the graph for the 8-5-3 milk jug problem where the green node is a start state and the red nodes are end states.

```
startStyle = cleanedStart /. {a_Integer, b_Integer, c_Integer} :> ({a, b, c} → Green);
endStyle = cleanedEnd /. {a_Integer, b_Integer, c_Integer} :> ({a, b, c} → Red);
newGraph = Graph[cleanedEdges, VertexLabels → "Name",
  VertexStyle → Join[startStyle, endStyle], ImageSize → 620]
```