A Milk (chairman -> Chair, so milkman -> Milk) has three jugs of milk, one carrying 8 liters, one 5 liters,
and one 3 liters.The 8 liter jug is full, the other two are empty.Can they measure 4 liters of milk?
This progam takes any number of jugs with different sizes.

```
Moves::usage =
  "Moves[pstates_List, kstates_List, fixedstate_List, output_List:{}].\n pstates
    >> This variable is for processing the states.It grows and eventually
    becomes empty. \n kstates >> This varialble helps to avoid duplicates. \n
    fixedstate >> Holdes the sizes of the jugs.\n output >>  The final output";
```

```
? Moves
```

```
Moves[pstates_List, kstates_List, fixedstate_List, output_List:{}].
 pstates >> This variable is for
  processing the states.It grows and eventually becomes empty.
 kstates >> This varialble helps to avoid duplicates.
 fixedstate >> Holdes the sizes of the jugs.
 output >>  The final output
```

```
Moves[{}, kstates_List, fixedstate_List, output_List] := Return[output];
Moves[pstates_List, kstates_List, fixedstate_List, output_List: {}] :=
 Module[{n, i , j , pst, fst, kst, tmp, diff, ou, gph = {}},
  pst  = pstates; kst = kstates; fst = fixedstate;
  If[Length[pst[[1]]] ≠ Length[fst],
   Print["Error"];
   Return[];
  ];
  n = Length[fst];
  For[i = 1, i ≤ n, i++,
   For[j = 1, j ≤ n, j++,
    If[i ≠ j,
      tmp = pst[[1]];
      If[pst[[1, i]] > 0,
       If[pst[[1, i]] >  fst[[j]] - pst[[1, j]],
         tmp[[i]] = pst[[1, i]] + pst[[1, j]] - fst[[j]];
         tmp[[j]] = fst[[j]];
         If[MemberQ[kst, tmp] === False,
          AppendTo[kst, tmp]; AppendTo[pst, tmp]; AppendTo[gph, pst[[1]] → tmp];
         ], (* Else*)
         tmp[[i]] = 0;
         tmp[[j]] =  pst[[1, i]] + pst[[1, j]];
         If[MemberQ[kst, tmp] === False,
          AppendTo[kst, tmp];  AppendTo[pst, tmp];
          AppendTo[gph, pst[[1]] → tmp]
         ];
```

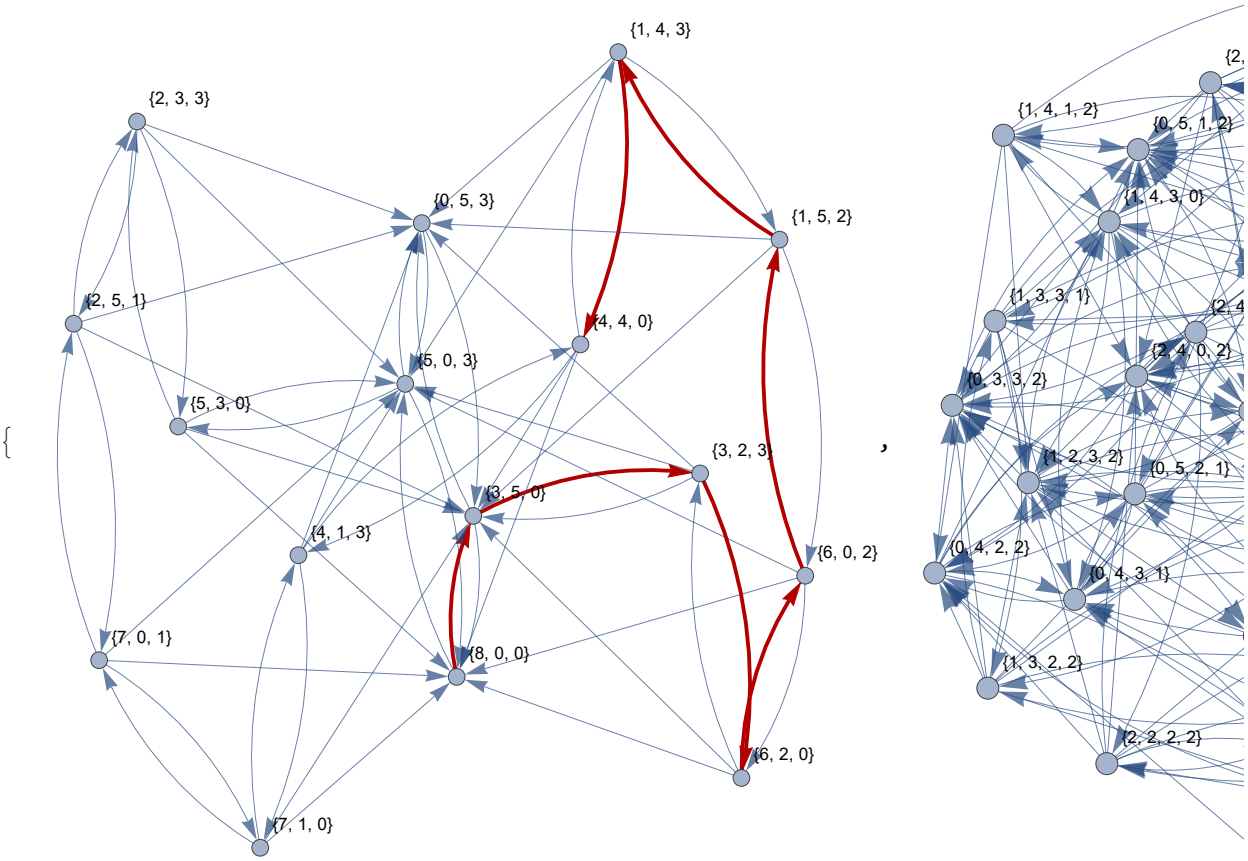
```
        ];
       ];

       (*Back tracking*)
       If[pst[[1, i]] < fst[[i]],
        diff = fst[[i]] - pst[[1, i]];
        If[pst[[1, j]] ≤ diff,
         tmp[[i]] = pst[[1, i]] + pst[[1, j]];
         tmp[[j]] = 0;
         If[pst[[1]] ≠ tmp,
          AppendTo[gph, pst[[1]] → tmp];
         ]
        ];
        If[pst[[1, j]] ≥ diff,
         tmp[[j]] = pst[[1, j]] - diff;
         tmp[[i]] = pst[[1, i]] + diff;
         If[pst[[1]] ≠ tmp,
          AppendTo[gph, pst[[1]] → tmp];
         ]
        ]
       ]
      ];
    ](* End of second For loop*)
   ]; (* End of first For loop*)
   ou = Union[output, gph];
   Moves[Delete[pst, 1], kst, fst, ou]
  ]

sts1 = Moves[{{8, 0, 0}}, {{8, 0, 0}}, {8, 5, 3}];
g1 = Graph[sts, VertexLabels → "Name", ImageSize → 450];
m1 = HighlightGraph[g1, PathGraph[FindShortestPath[sts, {8, 0, 0}, {4, 4, 0}],
     DirectedEdges → True], GraphHighlightStyle → "Thick"];

(*More jugs*)
sts2 = Moves[{{8, 0, 0, 0}}, {{8, 0, 0, 0}}, {8, 5, 3, 2}];
g2 = Graph[sts2, VertexLabels → "Name", ImageSize → 450];
m2 = HighlightGraph[g2, PathGraph[FindShortestPath[sts2, {8, 0, 0, 0}, {4, 4, 0, 0}],
     DirectedEdges → True], GraphHighlightStyle → "Thick"];
```

```
{m1, m2}
```


{1, 4, 3}
{2, 3, 3}
{0, 5, 3}
{1, 5, 2}
{2, 5, 1}
{4, 4, 0}
{5, 0, 3}
{5, 3, 0}
{3, 2, 3}
{3, 5, 0}
{4, 1, 3}
{6, 0, 2}
{7, 0, 1}
{8, 0, 0}
{6, 2, 0}
{7, 1, 0}
{1, 4, 1, 2}
{0, 5, 1, 2}
{1, 4, 3, 0}
{1, 3, 3, 1}
{2, 4, 0, 2}
{0, 3, 3, 2}
{1, 2, 3, 2}
{0, 5, 2, 1}
{0, 4, 2, 2}
{0, 4, 3, 1}
{1, 3, 2, 2}
{2, 2, 2, 2}