**4) Java implementation of the Water-Jug problem**

**Solution:**

**1) pour 5 litres from the 8-litre to the 5-litre bowl,**

**2) pour 3 litres from the 5-litre to the 3-litre bowl,**

**3) pour these 3 litres back to the 8-litre bowl,**

**4) pour the remaining 2 litres from the 5-litre to the 3-litre bowl,**

**5) pour 5 litres from the 8-litre to the 5-litre bowl,**

**6) pour the missing 1 litre from the 5-litre to the 3-litre bowl (there should be 4 litres left in the 5-litre bowl),**

**7) pour the 3 litres back from the 3-litre to the 8-litre bowl (and that’s it – in 8-litre bowl 4 litres).**

**import java.util.ArrayList;**

**import java.util.List;**

**public class State**

**{**

**int a=0;// 3 litre**

**int b=0;//5 litre**

**int c=8;//8 litre**

**public State(int a, int b)**

**{**

**this.a=a;**

**this.b=b;**

**this.c=8-a-b;**

**}**

**public boolean isGoal()**

**{**

**return (b==4 && c==4);**

**}**

**public boolean equals(Object xx)**

**{**

**State x = (State) xx;**

**if(this.a==x.a && this.b==x.b && this.c==x.c)**

**{**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**public int hashCode()**

**{**

**return 8;**

**}**

**public List<State> getChildren()**

**{**

**List<State> children = new ArrayList<State>();**

**// a -> b**

**if(a!=0 && b!=5)// if a is not empty**

**{**

**if(a+b<=5)**

**{**

**children.add(new State(0, a+b));**

**}**

**else**

**{**

**children.add(new State(a+b-5,5));**

**}**

**}**

**//a->c**

**if(a!=0 && c!=8)**

**{**

**// We are pouring completely from a to c**

**// a will be 0**

**// b will be 8-a-c**

**// c will be a+c**

**children.add(new State(0, 8-a-c));**

**}**

**//b->a**

**if(b!=0 && a!=3)**

**{**

**if(a+b<=3)**

**{**

**children.add(new State(a+b, 0));**

**}**

**else**

**{**

**children.add(new State(3, a+b-3));**

**}**

**}**

**// b->c**

**if(b!=0 && c!=8)**

**{**

**// We are pouring completely from b to c**

**// a will be 8-b-c**

**// b will be 0**

**// c will be b+c**

**children.add(new State(8-b-c, 0));**

**}**

**//c->a**

**if(c!=0 && a!=3)**

**{**

**if(c+a<=3)**

**{**

**children.add(new State(c+a, 8-c-a));**

**}**

**else**

**{**

**// a will be full i.e. 3 liters**

**// b will be 8-c-a**

**// c will be c+a-3**

**children.add(new State(3, 8-c-a));**

**}**

**}**

**// c->b**

**if(c!=0 && b!=5)**

**{**

**if(c+b<=5)**

**{**

**children.add(new State(8-c-b , c+b));**

**}**

**else**

**{**

**children.add(new State(8-c-b, 5));**

**}**

**}**

**return children;**

**}**

**@Override**

**public String toString()**

**{**

**return "{"+a+","+b+","+c+"}";**

**}**

**}**

**public class BFSThreeJugs**

**{**

**private static List<State> visitedStates=new ArrayList<State>();**

**private static Queue<State> stateQueue = new LinkedList<State>();**

**public static void main(String[] args) throws NotSupportedException**

**{**

**State currentState = new State(0,0);**

**// Add current state to state Queue.**

**stateQueue.add(currentState);**

**do**

**{**

**// Get the first Element from Queue.**

**State firstElementInQueue = stateQueue.peek();**

**// If the first Element is the Goal**

**// We are done.**

**if(firstElementInQueue.isGoal())**

**{**

**for(State p : visitedStates)**

**{**

**System.out.println(p.toString());**

**}**

**// There is no recursion here, so simple return would do.**

**return;**

**}**

**else**

**{**

**// Add firstElement to visited States**

**visitedStates.add(firstElementInQueue);**

**// Get the children of first element**

**List<State> children = firstElementInQueue.getChildren();**

**for(State v : children)**

**{**

**// if children has not already been visited.**

**if(!visitedStates.contains(v))**

**{**

**// add the child to state Queue.**

**stateQueue.add(v);**

**}**

**}**

**// Remove the first element from state queue.**

**stateQueue.remove(firstElementInQueue);**

**}**

**// do this till state queue is empty.**

**}while(!stateQueue.isEmpty());**

**}**

**}**

**Reference:**

**I took reference for the implementation from**

**a) https://gist.github.com/oktapodi/5443952**

**b) http://projectsgeek.com/2011/09/water-jug-problem-artificial-intelligence.html**

**c) http://javakafunda.blogspot.com/2013/09/the-water-jug-problem.html**

**d) http://www.iaeng.org/publication/IMECS2015/IMECS2015\_pp138-140.pdf**

**e) https://www.youtube.com/watch?v=uv9Mgs-cUA0**