KR-IST - Lecture 3a: Problem solving in Java

Chris Thornton

October 30, 2014

Introduction

This lecture will look at a java program for solving the water-jugs problem.

We have two jugs X and Y.

X can hold 4 pints.

Y can hold 3.

Permissable moves are to fill X from Y, to fill Y from X, and to empty or fill either jug *completely*.

The aim is to find a sequence of actions, starting from empty jugs, which achieves a state in which Y contains exactly two pints.

Preliminaries

Initial questions

Preliminaries

Initial questions

- What are the essential features of a state?

Preliminaries

Initial questions

- What are the essential features of a state?
- What is the best way to represent a state?

Preliminaries

Initial questions

- What are the essential features of a state?
- What is the best way to represent a state?
- Given a chosen state representation, how easy will it be to generate successors?

Preliminaries

Initial questions

- What are the essential features of a state?
- What is the best way to represent a state?
- Given a chosen state representation, how easy will it be to generate successors?

Produce a sketch of the successor method

Represent jug contents by numbers.

State representation is a data structure containing two numbers.

Successor generation will involve looking at two numbers and finding out which actions are possible.

Is it possible to transfer from one jug to another?

Is it possible to empty a jug?

Is it possible to fill a jug?

How will these actions be applied within the chosen state representation?

Node class

```
import java.util.*;

class Node {
   int x = 0, y = 0; /* state variables */
   Node parent = null; /* parent link */

   Node (int x, int y, Node parent) {
      this.x = x;
      this.y = y;
      this.parent = parent;
   }

   public String toString() {
      return(x + " " + y);
   }

   public boolean equals(Object node) { /* argument has to be an Obje
      return(((Node)node).x == x && ((Node)node).y == y);
   }
```

Node class cont.

NB. Use Vector instead of ArrayList because they allow ‘insert’ mutations.

```
    Vector&lt;Node&gt; getPath(Vector&lt;Node&gt; v) {
       v.insertElementAt(this, 0);
       if (parent != null) v = parent.getPath(v);
       return(v);
    }

    Vector&lt;Node&gt; getPath() { return(getPath(new Vector&lt;Node&g
 }
```

WaterJugsSearch class (successor function)

```
public class WaterJugsSearch {

   boolean isGoal(Node node) {
      return(node.y == 2);
   }
```

WaterJugsSearch class cont.

```
Vector&lt;Node&gt; getSuccessors(Node parent) {
   int x = parent.x, y = parent.y;
   Vector&lt;Node&gt; successors = new Vector&lt;Node&gt;();
   if (x &lt; 4 && y &gt; 0) { /* transfer amount z from y to x */
      int z = Math.min(y, 4-x);
      successors.add(new Node(x+z, y-z, parent)); }
   if (y &lt; 3 && x &gt; 0) { /* transfer amount z from x to y */
      int z = Math.min(x, 3-y);
      successors.add(new Node(x-z, y+z, parent)); }
   if (x &gt; 0) { /* empty x */
      successors.add(new Node(0, y, parent)); }
   if (y &gt; 0) { /* empty y */
      successors.add(new Node(x, 0, parent)); }
   if (x &lt; 4) { /* fill x from tap */
      successors.add(new Node(4, y, parent)); }
   if (y &lt; 3) { /* fill y from tap */
      successors.add(new Node(x, 3, parent)); }
   return(successors);
}
```

Main loop

```
void run() {
   Vector&lt;Node&gt; open = new Vector&lt;Node&gt;();
   open.add(new Node(0, 0, null));

   while (open.size() &gt; 0) {
      Node node = open.remove(0);
      if (isGoal(node)) {
         System.out.println("Solution: " + node.getPath()); }
      else {
         Vector&lt;Node&gt; successors = getSuccessors(node);
         for (int i = 0; i &lt; successors.size(); i++) {
            Node child = successors.get(i);
            if (!node.getPath().contains((Object)child)) {
               open.add(child); }
         }
      }
   }
}
```

main method

```
   public static void main(String args[]) { // do the search
      new WaterJugsSearch().run();
   }
}
```

Output generated

There are 12 distinct solutions:

```
Solution: [0 0, 0 3, 3 0, 3 3, 4 2]
Solution: [0 0, 4 0, 1 3, 0 3, 3 0, 3 3, 4 2]
Solution: [0 0, 4 0, 4 3, 0 3, 3 0, 3 3, 4 2]
Solution: [0 0, 4 0, 1 3, 4 3, 0 3, 3 0, 3 3, 4 2]
Solution: [0 0, 4 0, 1 3, 1 0, 0 1, 4 1, 2 3, 2 0, 0 2]
Solution: [0 0, 4 0, 1 3, 1 0, 0 1, 0 3, 3 0, 3 3, 4 2]
Solution: [0 0, 4 0, 1 3, 1 0, 0 1, 4 1, 2 3, 0 3, 3 0, 3 3, 4 2]
Solution: [0 0, 4 0, 1 3, 1 0, 0 1, 4 1, 4 3, 0 3, 3 0, 3 3, 4 2]
Solution: [0 0, 0 3, 3 0, 4 0, 1 3, 1 0, 0 1, 4 1, 2 3, 2 0, 0 2]
Solution: [0 0, 0 3, 4 3, 4 0, 1 3, 1 0, 0 1, 4 1, 2 3, 2 0, 0 2]
Solution: [0 0, 4 0, 1 3, 1 0, 0 1, 4 1, 2 3, 4 3, 0 3, 3 0, 3 3, 4 2
Solution: [0 0, 0 3, 3 0, 3 3, 4 3, 4 0, 1 3, 1 0, 0 1, 4 1, 2 3, 2 0
```

.

Summary

Summary

- Always start by choosing a state representation

Summary

- Always start by choosing a state representation
- Node class

Summary

- Always start by choosing a state representation
- Node class
- WaterJugsSearch class with successor function

Summary

- Always start by choosing a state representation
- Node class
- WaterJugsSearch class with successor function
- Main loop

Summary

- Always start by choosing a state representation
- Node class
- WaterJugsSearch class with successor function
- Main loop
- main method

Summary

- Always start by choosing a state representation
- Node class
- WaterJugsSearch class with successor function
- Main loop
- main method
- Output generated

Summary

- Always start by choosing a state representation
- Node class
- WaterJugsSearch class with successor function
- Main loop
- main method
- Output generated

Questions

Questions

- What is the first task when implementatating a search program?

Questions

- What is the first task when implementating a search program?
- How soon should you start writing code?

Questions

- What is the first task when implementating a search program?
- How soon should you start writing code?
- How can we resolve the main drawback with the Vector data structure?

Questions

- What is the first task when implementating a search program?
- How soon should you start writing code?
- How can we resolve the main drawback with the Vector data structure?
- Writing the code for successor generation is the key task in implementing a search program? What is the best way to go about this task?

Questions

- What is the first task when implementating a search program?
- How soon should you start writing code?
- How can we resolve the main drawback with the Vector data structure?
- Writing the code for successor generation is the key task in implementing a search program? What is the best way to go about this task?
- What strategy does the WaterJugsSearch program use for generating solution paths?

Questions

- What is the first task when implementating a search program?
- How soon should you start writing code?
- How can we resolve the main drawback with the Vector data structure?
- Writing the code for successor generation is the key task in implementing a search program? What is the best way to go about this task?
- What strategy does the WaterJugsSearch program use for generating solution paths?
- What other strategies might be used for keeping track of and printing out solution paths?

Questions

- What is the first task when implementating a search program?
- How soon should you start writing code?
- How can we resolve the main drawback with the Vector data structure?
- Writing the code for successor generation is the key task in implementing a search program? What is the best way to go about this task?
- What strategy does the WaterJugsSearch program use for generating solution paths?
- What other strategies might be used for keeping track of and printing out solution paths?
- Modify the program so that it implements an iterative-deepening search strategy.

Questions

- What is the first task when implementating a search program?
- How soon should you start writing code?
- How can we resolve the main drawback with the Vector data structure?
- Writing the code for successor generation is the key task in implementing a search program? What is the best way to go about this task?
- What strategy does the WaterJugsSearch program use for generating solution paths?
- What other strategies might be used for keeping track of and printing out solution paths?
- Modify the program so that it implements an iterative-deepening search strategy.

Exercises

Implement a version of the WaterJugsSearch program which searches for a debt-minimising sequence of card transfers (as detailed in the previous lecture).