323-34 Project 8: 8 puzzels **Java**

Student: Seratul Ambia

Project Due Date: 5/12/2021

<u>Source Code:</u>

```
package puzzle;

import java.io.*;
import java.util.*;
import java.util.stream.IntStream;


class PuzzleLinkedList {

        private int counter = 0;
        Node head;

        class Node {
                Object data;
                Node next;
                Node (Object d) {
                        data = d;
                }
        }

        private void decreamentCount() {
                counter -= 1;
        }


        private void increamentCount() {
                counter += 1;
        }

        public PuzzleLinkedList add(Object data) {

                Node newNode = new Node(data);
                newNode.next = null;
        if (head == null) {
            head = newNode;
        } else {

            Node last = head;

            while (last.next != null) {
                last = last.next;
            }


            last.next = newNode;
        }

        increamentCount();

        return this;
    }


    public PuzzleLinkedList add(int index, Object data) {

        Node newNode = new Node(data);
        newNode.next = null;
        Node current = head;
        Node prev = head;


        if (prev != null) {
            for (int i = 0; i <= index && current.next != null; i++) {
                prev = current;
                current = current.next;
            }

            newNode.next = prev;
            prev.next = newNode;
        } else {
            head = newNode;
        }

        increamentCount();

        return this;
}

public Node get(int index) {
        if (index < 0) {
                return null;
        }
        Node current = null;
        if (head != null) {
                current = head;
                for (int i = 0; i <= index; i++) {
                        if (i == index) {
                                return current;
                        }

                        if (current.next == null) {
                                return null;
                        }

                        current = current.next;
                }
        }

        return current;
}

public int indexOf(Object item) {
        int index = -1;

        if (head != null) {
                Node current = head;
                for (int i = 0; i < size(); i++, current = current.next) {
                        if (item == current.data) {
                                return i;
                        }
                }
```

```
            }


            return index;
    }

    public Node pop() {
             if (head == null) {
                      return null;
             }

             Node _head = head;
             head = head.next;

             decreamentCount();

             return _head;
    }

    public Node remove(int index) {
             if (head == null) {
                      return null;
             }

             Node temp = head;
             Node prev = head;

            if (index == 0) {
                     return pop();
            }

             for (int i = 0; temp != null && i < index; i++) {
                      prev = temp;
                      temp = temp.next;
             }

             if (temp == null) {
                      return null;
             }
```

```
                  prev.next = temp.next;
                  decreamentCount();
                  return temp;
           }

           public void printList() {
        Node currNode = head;

        System.out.print("LinkedList: ");
        while (currNode != null) {
           System.out.print(currNode.data + " ");
           currNode = currNode.next;
        }
    }

           public int size() {
        return counter;
    }
}


class AstarNode {

           protected int[] configuration;
           protected int gStar;
           protected int hStar;
           protected int fStar;
           AstarNode parent = null;

           public AstarNode(int[] configuration) {
                  this.configuration = configuration;
           }

           public void setGstar(int gStar) {
                  this.gStar = gStar;
           }

           public int getGstar() {
                  return gStar;
           }
```

```
	public void setHstar(int hStar) {
		this.hStar = hStar;
	}

	public int getHstar() {
		return hStar;
	}

	public void setFstar(int fStar) {
		this.fStar = fStar;
	}


	public int getFstar() {
		return fStar;
	}

	public String printNode() {
		String output = "<" + fStar +"::" + getConfiguration();

		if (parent != null) {
			output += "::" + parent.getConfiguration();
		}

		output += ">";

		return output;
	}

	public String getConfiguration() {
		StringJoiner sj = new StringJoiner(" ");
		IntStream.of(configuration).forEach(x -> sj.add(String.valueOf(x)));
		return sj.toString();
	}
}


public class main {
```

```
public static AstarNode startNode;
public static AstarNode goalNode;
public static PuzzleLinkedList Open;
public static PuzzleLinkedList Close;
public static PuzzleLinkedList childList;

public static int computeGstar(AstarNode n) {
        if (n.parent == null) {
                return 0;
        }

        return n.parent.getGstar() + 1;
}


public static int computeHstar(AstarNode n) {
        int temp = 0;

        String current = n.getConfiguration();
        String goal = goalNode.getConfiguration();
        for (int i = 0; i < current.length(); i++) {
                if (current.charAt(i) != goal.charAt(i)) {
                        temp += 1;
                }
        }
        return temp;
}


public static boolean match(String configuration1, String configuration2) {
        return configuration1.equals(configuration2);
}


public static boolean isGoalNode(AstarNode n) {

        return match(n.getConfiguration(), goalNode.getConfiguration());
}
```

```
public static void listInsert(AstarNode n) {
        Open.add(n);
}


public static PuzzleLinkedList.Node remove(PuzzleLinkedList OpenList) {
        return OpenList.remove(1);
}

public static boolean checkAncestors(AstarNode currentNode) {
        AstarNode parent = currentNode.parent;
        while (parent != null) {
                if (match(parent.getConfiguration(), currentNode.getConfiguration())) {
                        return true;
                }
                parent = parent.parent;
        }

        return false;
}

public static int[] translateConfiguration(String str) {
        String[] config = str.split(" ");
        int[] numConfig = new int[config.length];

        for (int i = 0; i < config.length; i++) {
                numConfig[i] = Integer.parseInt((config[i]));
        }

        return numConfig;
}


public static int[] rotateConfig(int[] arr) {
        // create temp array of size d
        int[] temp = new int[arr.length];
        int tempEl = arr[0];

        // copy first d element(s) in array temp
        for (int i = 0; i < arr.length - 1; i++) {
            temp[i] = arr[i + 1];
        }
        temp[arr.length - 1] = tempEl;

        return temp;
    }


    public static PuzzleLinkedList constructChildList(AstarNode currentNode) {
        PuzzleLinkedList childList = new PuzzleLinkedList();
        String nodeConfiguration = currentNode.getConfiguration();
        int[] nodeNumConfiguration = translateConfiguration(nodeConfiguration);
        int configLength = nodeNumConfiguration.length;
        int[] nextConfig = nodeNumConfiguration;


        for (int i = 1; i <= configLength; i++) {
            nextConfig = rotateConfig(nextConfig);

            AstarNode newNode = new AstarNode(nextConfig);
            newNode.parent = currentNode;

            if (nextConfig[0] != 0 && checkAncestors(newNode) == false) {
                childList.add(newNode);
            }
        }

        return childList;
    }


    public static void printList(PuzzleLinkedList.Node listHead, FileWriter outFile1) {
        PuzzleLinkedList.Node _next = listHead;
        while (_next != null) {
            AstarNode actualNode = (AstarNode) _next.data;
            try {
                outFile1.write(actualNode.printNode() + "\n");
                _next = _next.next;
            } catch (IOException e) {
```

```
                        System.out.println("Something went wrong");
                    }
            }
    }


    public static void printSolution(AstarNode currentNode, FileWriter outFile2) {
             try {
                     outFile2.write(currentNode.printNode() + "\n");
             } catch (IOException e) {
                     System.out.println("Something went wrong!");
             }
    }


    public static void printToFile(
                    FileWriter wr,
                    PuzzleLinkedList Open,
                    PuzzleLinkedList Close
                    ) {
            try {
                    wr.write("This is Open list:\n");
                    printList(Open.head, wr);
                    wr.write("This is CLOSE list:\n");
                    printList(Close.head, wr);
            } catch (IOException e) {
                    System.out.println("Something went wrong");
            }
    }


    public static int[] readFile(String filename) throws IOException {
            int j = 0;
            File file = new File(filename);
            FileInputStream fileInputStream;

            fileInputStream = new FileInputStream(file);
            byte[] value = new byte[(int) file.length()];
            fileInputStream.read(value);
            fileInputStream.close();
```

```
        String fileContent = new String(value, "UTF-8");

        String[] stringConfig = fileContent.split("", 0);
        int[] intConfig = new int[stringConfig.length];

        for (int i = 0; i < stringConfig.length; i++) {
                try {
                intConfig[j] = Integer.parseInt(stringConfig[i]);
                }
                catch (NumberFormatException nfe) {
             continue;
          }
                j++;
        }

        return intConfig;
}

public static void main(String[] args) {
        try {
                // Step0
                int[] inFile1 = readFile(args[0]);
                int[] inFile2 = readFile(args[1]);
                int[] dummyNodeConfig = {-1, -1, -1, -1, -1, -1, -1, -1, -1};
                // Flag to determine goal reach status
                boolean goalReached = false;

                FileWriter outFile1 = new FileWriter(args[2]);
                FileWriter outFile2 = new FileWriter(args[3]);

                startNode = new AstarNode(inFile1);
                goalNode = new AstarNode(inFile2);

                Open = new PuzzleLinkedList();
                Open.add(new AstarNode(dummyNodeConfig));
                Close = new PuzzleLinkedList();
                Close.add(new AstarNode(dummyNodeConfig));

                // Step1
```

```
startNode.setGstar(computeGstar(startNode));
startNode.setHstar(computeHstar(startNode));
startNode.setFstar(startNode.getGstar() + startNode.getHstar());

listInsert(startNode);

/**
 * Remove this loop variable and its reference at the end
 * of the loop
 */
int loops = 0;

while (Open.size() > 0 && loops < 5) {

        //Step2
        PuzzleLinkedList.Node _node = remove(Open);

        if (_node == null) {
                break;
        }

        AstarNode currentNode = (AstarNode)_node.data;

        // Step3
        if (isGoalNode(currentNode)) {
                printSolution(currentNode, outFile2);
                goalReached = true;
                break;
        }


        // Step4
        childList = constructChildList(currentNode);

        while (childList.size() > 0) {
                // Step5
                AstarNode child = (AstarNode) childList.pop().data;

                // Step6
                child.setGstar(computeGstar(child));
```

```
                                                child.setHstar(computeHstar(child));
                                                child.setFstar(child.getGstar() + child.getHstar());

                                                // Step7
                                                int inOpen = Open.indexOf(child);
                                                int inClose = Close.indexOf(child);


                                                // Not in both lists
                                                if (inOpen == -1 && inClose == -1) {
                                                        Open.add(child);
                                                } else {
                                                        if (inOpen != -1) {
                                                                AstarNode oldNode = (AstarNode)
Open.get(inOpen).data;
                                                                if (child.getFstar() < oldNode.getFstar()) {
                                                                        // replace child with the old child
open
                                                                        Open.remove(inOpen);
                                                                        Open.add(inOpen, child);
                                                                }
                                                        }

                                                        if (inClose != -1) {
                                                                AstarNode oldNode = (AstarNode)
Close.get(inClose).data;
                                                                if (child.getFstar() < oldNode.getFstar()) {
                                                                        // Remove from Close and add to
Open
                                                                        PuzzleLinkedList.Node rNode =
Close.remove(inClose);

                                                                        Open.add(rNode);
                                                                }
                                                        }

                                                }
                                        }

                                        // Step9: Add currentNode to Close
```

```
                        Close.add(currentNode);

                        // Step10
                        printToFile(outFile1, Open, Close);

                        loops++;
                    }

                    // Step12
                    if (goalReached != true) {
                        outFile1.write("no output can be found in the search!");
                    }

                    // Step13: Close files
                    outFile1.close();
                    outFile2.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
        }
}
```

OUTPUT:

```
This is Open list:
<0::-1 -1 -1 -1 -1 -1 -1 -1 -1>
<11::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<12::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<15::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<16::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
This is CLOSE list:
<0::-1 -1 -1 -1 -1 -1 -1 -1 -1>
<6::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
This is Open list:
<0::-1 -1 -1 -1 -1 -1 -1 -1 -1>
<12::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
```

```
<13::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<15::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<16::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<16::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<17::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
This is CLOSE list:
<0::-1 -1 -1 -1 -1 -1 -1 -1 -1>
<6::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<11::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
This is Open list:
<0::-1 -1 -1 -1 -1 -1 -1 -1 -1>
<13::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<15::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<16::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<16::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<17::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<14::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<14::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<16::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<17::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<12::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
This is CLOSE list:
<0::-1 -1 -1 -1 -1 -1 -1 -1 -1>
<6::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<11::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<12::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
This is Open list:
<0::-1 -1 -1 -1 -1 -1 -1 -1 -1>
```

```
<13::7 2 6 4 0 0 0 0 0 0 0 0 0 1 5 8 3 0::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0>
<13::2 6 4 0 0 0 0 0 0 0 0 0 1 5 8 3 0 7::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0>
<15::6 4 0 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0>
<16::4 0 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0>
<13::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1>
<14::3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5 8::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1>
<14::7 2 6 4 0 0 0 0 0 0 0 0 0 1 5 8 3 0::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1>
<14::2 6 4 0 0 0 0 0 0 0 0 0 1 5 8 3 0 7::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1>
<16::6 4 0 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1>
<17::4 0 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1>
<14::3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5 8::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5>
<14::7 2 6 4 0 0 0 0 0 0 0 0 0 1 5 8 3 0::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5>
<14::2 6 4 0 0 0 0 0 0 0 0 0 1 5 8 3 0 7::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5>
<16::6 4 0 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5>
<17::4 0 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5>
<12::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5>
<14::7 2 6 4 0 0 0 0 0 0 0 0 0 1 5 8 3 0::3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5 8>
<14::2 6 4 0 0 0 0 0 0 0 0 0 1 5 8 3 0 7::3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5 8>
<16::6 4 0 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5 8>
<17::4 0 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5 8>
<12::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1::3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5 8>
<13::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5::3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5 8>
This is CLOSE list:
<0::-1 -1 -1 -1 -1 -1 -1 -1 -1>
<6::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0>
<11::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0>
<12::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0>
<13::3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5 8::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0>
This is Open list:
<0::-1 -1 -1 -1 -1 -1 -1 -1 -1>
<13::2 6 4 0 0 0 0 0 0 0 0 0 1 5 8 3 0 7::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0>
<15::6 4 0 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0>
<16::4 0 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0>
<13::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1>
<14::3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5 8::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1>
<14::7 2 6 4 0 0 0 0 0 0 0 0 0 1 5 8 3 0::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1>
<14::2 6 4 0 0 0 0 0 0 0 0 0 1 5 8 3 0 7::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1>
<16::6 4 0 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1>
<17::4 0 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1>
<14::3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5 8::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 0 1 5>
```

```
<14::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<14::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<16::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<17::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<12::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<14::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8>
<14::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8>
<16::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8>
<17::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8>
<12::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8>
<13::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8>
<14::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0>
<16::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0>
<17::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0>
<12::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0>
<13::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0>
<14::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0>
This is CLOSE list:
<0::-1 -1 -1 -1 -1 -1 -1 -1 -1>
<6::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<11::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<12::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
no output can be found in the search!

This is Open list:
<0::-1 -1 -1 -1 -1 -1 -1 -1 -1>
<11::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<12::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<15::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<16::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
This is CLOSE list:
<0::-1 -1 -1 -1 -1 -1 -1 -1 -1>
<6::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
This is Open list:
<0::-1 -1 -1 -1 -1 -1 -1 -1 -1>
```

```
<12::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<15::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<16::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<16::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<17::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
This is CLOSE list:
<0::-1 -1 -1 -1 -1 -1 -1 -1 -1>
<6::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<11::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
This is Open list:
<0::-1 -1 -1 -1 -1 -1 -1 -1 -1>
<13::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<15::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<16::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<16::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<17::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<14::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<14::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<16::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<17::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<12::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
This is CLOSE list:
<0::-1 -1 -1 -1 -1 -1 -1 -1 -1>
<6::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<11::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<12::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
```

```
This is Open list:
<0::-1 -1 -1 -1 -1 -1 -1 -1 -1>
<13::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<15::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<16::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<16::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<17::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<14::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<14::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<16::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<17::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<12::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<14::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8>
<14::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8>
<16::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8>
<17::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8>
<12::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8>
<13::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8>
This is CLOSE list:
<0::-1 -1 -1 -1 -1 -1 -1 -1 -1>
<6::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<11::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<12::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
This is Open list:
<0::-1 -1 -1 -1 -1 -1 -1 -1 -1>
<13::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<15::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<16::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<16::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
```

```
<17::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1>
<14::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<14::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<14::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<16::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<17::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<12::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5>
<14::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8>
<14::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8>
<16::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8>
<17::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8>
<12::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8>
<13::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8>
<14::2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0>
<16::6 4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0>
<17::4 0 0 0 0 0 0 0 0 1 5 8 3 0 7 2 6::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0>
<12::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0>
<13::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0>
<14::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0>
This is CLOSE list:
<0::-1 -1 -1 -1 -1 -1 -1 -1 -1>
<6::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<11::5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<12::8 3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::3 0 7 2 6 4 0 0 0 0 0 0 0 0 1 5 8::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
<13::7 2 6 4 0 0 0 0 0 0 0 0 1 5 8 3 0::1 5 8 3 0 7 2 6 4 0 0 0 0 0 0 0 0>
no output can be found in the search!
```