Quiz 13

Name: Zian Wang

Program Description:

(Here are test cases.)

I use a backtracking algorithm for this problem. I use breadth-first search, the root of the tree is [1], then we visit all its children: [1, 1], [1, 2] and [1, 3], then visit the children of [1, 1]: [1, 1, 1], [1, 1, 2] and [1, 1, 3], then visit the only child of [1, 2]: [1, 2, 3], and [1, 3] has no child, then we will visit the children of [1, 1, 1], and so on. When we create the children, we limit the next step must no smaller than the last step to make all results legal. Once a node's solution is complete, which means we sum up all lengths of all steps that stored this node, it equals to n, we will print this solution. Once the solution's length is larger than n, we will make the first step plus 1 and delete all following steps, and put this new node into the queue, this is like we make a new root and put it into the queue.

Node.java defines the nodes in the tree, it has an array: steps[], step[i] = k means the ith step of the robot is k meter. The other member in the Node class is a pointer that points to the next node in the queue.

Queue.java defines the queue, it is not priority queue, just a queue.

Quiz13.java is the main file. Here are the functions in these 3 files:

public static boolean complete(Node v, int n)
Inputs: the node v, the length n.
Outputs: whether all this node's steps' lengths sum up is equal to n.
Function: judge whether the node's solution is a complete solution, which means we sum up this node's all steps and it equals to n.

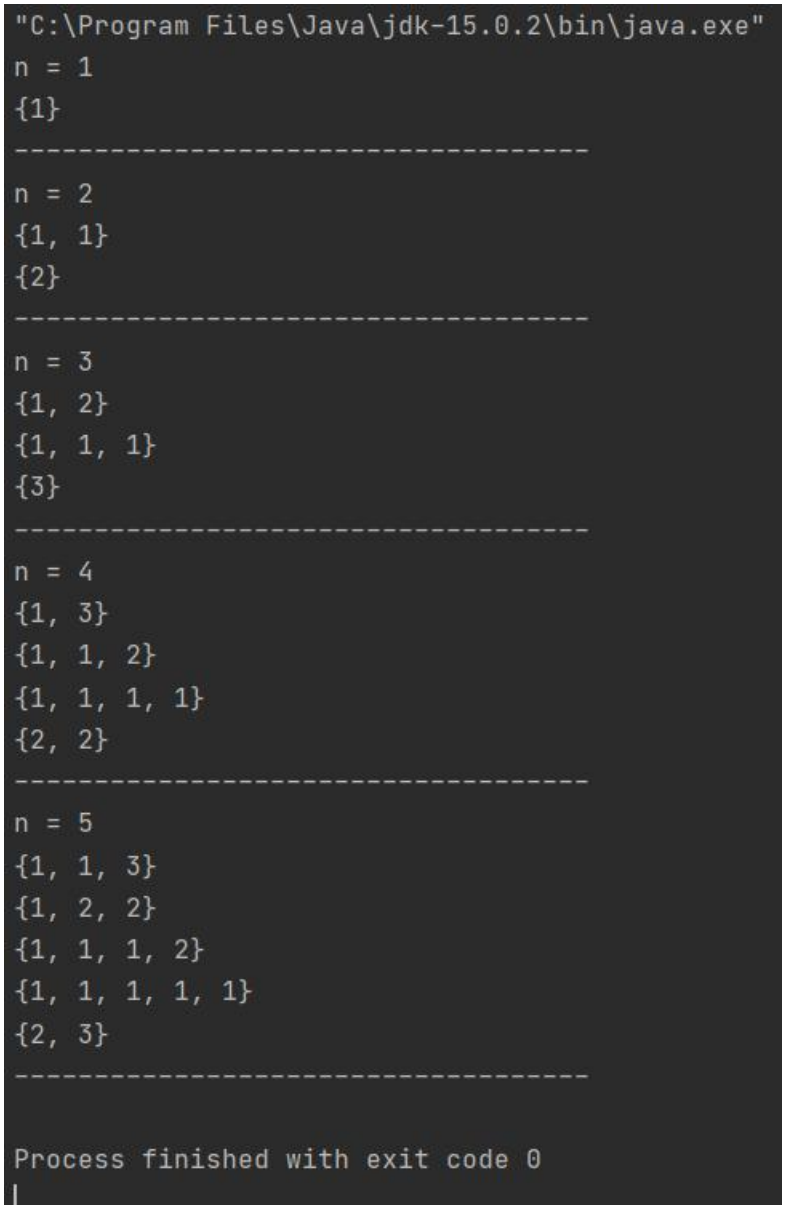
public static boolean exceed(Node v, int n)
Inputs: the node v, the length n.
Outputs: whether all this node's steps' lengths sum up is larger than n.
Function: judge whether the solution stored in this node exceeds the length n.

public static void robotStep(int n)
Inputs: length n.
Outputs: none.
Function: print all legal complete solutions of length n.

In queue.java, there is a function:

public boolean noExist(int n)
Input: int n.
Output: whether in this queue exist the steps that begins from n.
Function: to make sure we do not make multiple roots of the first step is n.

Test Cases:


"C:\Program Files\Java\jdk-15.0.2\bin\java.exe"
n = 1
{1}
---
n = 2
{1, 1}
{2}
---
n = 3
{1, 2}
{1, 1, 1}
{3}
---
n = 4
{1, 3}
{1, 1, 2}
{1, 1, 1, 1}
{2, 2}
---
n = 5
{1, 1, 3}
{1, 2, 2}
{1, 1, 1, 2}
{1, 1, 1, 1, 1}
{2, 3}

Process finished with exit code 0