

COMP 482 Project 3: Number of BSTs containing 1-N

Due: 2355 Tuesday July 12, 2022

Points: 30 points possible

Overview: Use Dynamic Programming to determine the number of different Binary Search Trees that contain the values 1, 2, 3, ..., N .

Details: The input will come from a file called input.txt which will be placed in the same directory as your java file. The first and only line of the file will have a single integer value N . Your program will output the number of different Binary Search Trees which contain the values from 1, 2, ..., N . Given the fact that these values grow very rapidly, you will probably want to use the java class BigInteger.

You must calculate the value using dynamic programming. If you either hardcode the solution or use pure recursion (without using memoization or tabularization) expect to get almost no points. A dynamic programming solution can calculate the answer for $N = 1000$ in less than 10 seconds on a modest computer (I used a 2016 MacBook) while a recursive solution will be extremely slow when $N = 25$ (ish - depends a bit upon the power of the computer used and how you define extremely).

You can discuss the algorithm to be used with anyone and consult any source (books, internet, etc). However, for this project, you are expected to write the code on your own with limited or no assistance from the professor, no assistance from others, and limited or no assistance from other sources (books, internet, etc). To clarify, you can seek assistance in understanding the task, but “your code” should be written by you: not written by others, not copied from others, not copied from books/internet.

Picky, but required specifications: Your project must:

- be submitted via canvas.
- consist of 1 or more dot-java files (no class files, zip files, input files or other files should be submitted). Each file must have your name and which project you are submitting as comments on the first 2 lines.
- not be placed into any package (for the java pedants, it must be in the default package).
- be designed and formatted reasonably (correct indentation, no excessively long lines, no excessively long methods, has useful method/variable names, etc)
- have one file called Project3.java.
- compile using the command ‘javac Project3.java’.
- run using the command ‘java Project3’.
- accept input from a file called input.txt in the same directory as the java file(s) formatted precisely as described above.
- accomplishes the goal of the project. In other words, the output should be the correct answer, computed in a valid way, formatted correctly.
- be submitted on time (early and multiple times is fine).

For each listed item that you fail to follow, expect to lose at least 5 points. In particular, submitting via anything other than canvas will result in a 0 and submitting more than a week late will also result in a 0.

Sample execution: If input.txt contains

0

then the output should be

1

If input.txt contains

5

then the output should be

42

If input.txt contains

20

then the output should be

6564120420

If input.txt contains

100

then the output should be

896519947090131496687170070074100632420837521538745909320

Stray Thoughts:

I suggest you finish you finish and submit your project at least a day or 2 in advance. This way you have time and opportunity to ask any last questions and verify that what you upload satisfies the requirements. There is nothing wrong with working on the project for a day and uploading your code, working for another day and uploading your improved code, ..., working for another day and uploading your final version. In fact there are advantages: you have a fairly reliable place that keeps your versions and even if you get busy at the last moment you have still uploaded your best version. Do not be concerned that canvas will append a -1, -2, -3, ... to your submission.

While I am likely to test your project on the input above, I am likely to test your projects on many other inputs as well.

Your project should be written and understood by you. Helping or receiving help from others to figure out what is allowed/required is fine, but copying code is not. Significant shared source code indicates that you either did not write/understand what you submitted or you assisted another in submitting code they did not write/understand.