Programming Assignment 1

CMPE 250, Data Structures and Algorithms, Fall 2011

Instructor: A. T. Cemgil
TA's: Barış Kurt, Murat Arar, Zeynep Gözen Sarıbatur

Due: 26 Oct Wednesday, 14:59, sharp

Problem Definition

We count the number of binary trees with N nodes, denoted by $T(N)$.

The key observation is that the given the root node, the left and right subtrees are (possibly empty) binary trees. So for a tree with $N+1$ nodes, we need to choose a left subtree of k nodes and a right subtree of $N-k$ nodes. So the number of trees satisfies the recursion

$$\begin{aligned} T(N+1) &= \sum_{k=0}^{N} T(k)T(N-k) \\ T(0) &= 1 \end{aligned}$$

This type of recursion is called a convolution. One can solve this for $T(N)$ using the method of generating functions (known also as the z transform) but this is not our topic here.

We verify this recursion

$$\begin{aligned} T(0) &= 1 \\ T(1) &= T(0)T(0) = 1 \\ T(2) &= T(0)T(1) + T(1)T(0) = 1 + 1 = 2 \\ T(3) &= T(0)T(2) + T(1)T(1) + T(2)T(0) = 2 + 1 + 2 = 5 \\ T(4) &= T(0)T(3) + T(1)T(2) + T(2)T(1) + T(3)T(0) = 5 + 2 + 2 + 5 = 14 \end{aligned}$$

And we program this recursion to get

```
0   1
1   1
2   2
3   5
4   14
5   42
6   132
7   429
8   1430
9   4862
10   16796
```

```
11  58786
12  208012
13  742900
14  2674440
15  9694845
16  35357670
17  129644790
18  477638700
19  1767263190
20  6564120420
...
99  227508830794229349661819540395688853956041682601541047340
```

An 32 bit integer can not store the number of trees when $N > 19$, so we need a large integer class.

Assignment

- Write an ANSI C++ program that calculates the number of binary trees up to 200 nodes.

Important Points

- Within your code, implement a LargeInteger class that can do addition and multiplication with large integers.
- You are expected to use C++ with as many related features as possible. This can (but does not have to) include operator overloading, classes, dynamic memory management, templates.
- Make sure that your output format is exactly as above example, that is each line i for $i \in [0, 200]$ should have i $T(i)$, and only that. This is important for the automated testing of correctness of your program.

Bonuses and Beyond

- Bonus: Write a program to draw all 429 binary trees with $N = 7$ nodes. (Deliver the drawings as one separate postscript file.)
  - How to draw trees in C++ using postscript? This will be covered in the problem session and we will provide more information next week on the course web page.
- Bonus: Derive an explicit formula for $T(N)$ (Hard). (Deliver your derivation as a separate pdf file (scanned or latex))
- For the Connoisseur: Show or disprove that $T(N)$ that is odd if and only if $N = 2^k - 1$ for positive integer k.

Helpful Links

- en.wikipedia.org/wiki/Computational_complexity_of_mathematical_operations
- en.wikipedia.org/wiki/Arbitrary-precision_arithmetic
- http://en.wikipedia.org/wiki/The_C_Programming_Language_%28book%29

Submission & Grading

- **What, How and When to submit**
  - You should submit your project in electronic form.
  - You should compress your source code (.cpp and .h files) in a zip file, name it as [pr]_[#]_[Student ID] (e.g. pr_1_2000700803.zip), and email to bucmpe250@gmail.com.
  - Your zip file should NOT contain any executable file (.exe) or any folder. If you sent multiple e-mails, only the last one will be taken into account.
  - The deadline is 26 Oct Wednesday, 14:59, sharp. Emails tagged later won't be considered.
- **Grading**
  - Your program will be graded based the correctness of your output and the clarity of the source code. Correctness of your output will be tested automatically so make sure you stick with the format described above.
  - There are several issues that makes a code piece 'quality'. In our case, you are expected to use C++ as powerful, steady and flexible as possible. Use mechanisms that affects these issues positively.
  - Make sure you document your code with necessary inline comments, and use meaningful variable names. Do not over-comment, or make your variable names unnecessarily long.
  - Try to write as efficient (both in terms of space and time) as possible. Informally speaking, try to make sure that your program completes under 20 seconds.