BLG372E - Analysis of Algorithms, Spring 2013

Project 3

Handed Out: **18.04.2013** 23.30

Deadline: **04.05.2013** 23.30

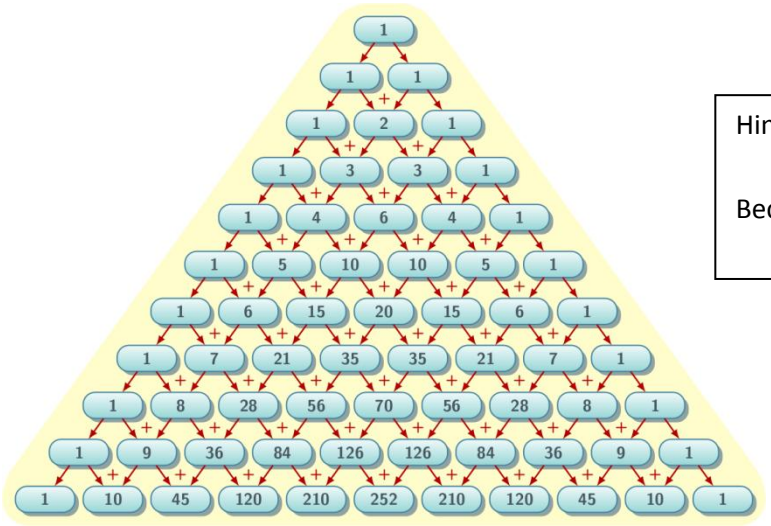
Overview

In this assignment you will calculate coefficient of each element in $(a + b)^n$ for various integer n values from n = 0, to n = *a specified value*.

There are two methods for computing those coefficients.

1) Use combinations. Find coefficient of $a^{n-k}b^k$ by calculating $\binom{n}{k}$, where 0 ≤ k ≤ n.

2) Use Pascal's triangle. n^{th} line of this triangle contains coefficients of $(a + b)^n$. k^{th} element of n^{th} line is coefficient of $a^{n-k}b^k$. Each line can be generated by using the line on top of it.



Hint: See that triangle is symmetric.

Because, $\binom{n}{k} = \binom{n}{n-k}$

This sample triangle has coefficients of each element in $(a+b)^n$, from n = 0 to n = 10. First line is n=0. In each line, leftmost elements are k=0.
Source: http://christopherolah.files.wordpress.com/2011/08/pascal-trinagle.png

This may sound simple, but when n is larger than 35, coefficients will not be able to fit in 32 bit unsigned integers. To handle with this problem, you will design and implement **BigUnsignedInteger** class that can hold and operate on unlimited sized unsigned integers.

With the help of **BigUnsignedInteger** class, you will implement Pascal's triangle and combinatorial solution for finding coefficients. You will write your code in C++ programming language. You are not allowed to use any library except standard C++ library. In your code, you will measure running times of both methods respectively (in milliseconds). In your report, you will compare and contrast their asymptotical bounds (space and time).

Code (60 points)

BigUnsignedInteger class (20 points)

This class must be able to handle with arbitrary digit numbers.

Store the value of big unsigned integer in an array of unsigned int.

Overloading +, -, *, / operators: 3 points for each (12).

Constructor(s), copy constructor, overloading =operator, destructor: 1 point for each (4).

Overloading <, ≤, == and *out stream* operators: 1 point for each (4).

All operators(except out stream) must be overloaded for both *BigUnsignedInteger* and unsigned int. For example, class must handle these kinds of different calls:

(*BigUnsignedInteger* + *BigUnsignedInteger*) and (*BigUnsignedInteger* + unsigned int).

CombinatorialSolution class (20 points)

- Implementation of algorithm for various n values: 15 points
- Optimizing use of resources: 5 points

Class will calculate each of the coefficients for: $(a + b)^0$, ... ,$(a + b)^{\text{maximum n}}$ and write length of each coefficient to "combinatorial.txt" file(see I/O section).

PascalsTriangle class (20 points)

- Implementation of algorithm for various n values: 15 points
- Optimizing use of resources: 5 points

Class will calculate each of the coefficients for: $(a + b)^0$, ... ,$(a + b)^{\text{maximum n}}$ and write length of each coefficient to "pascal.txt" file(see I/O section).

I/O (Important)

Your program will take two unsigned integers as arguments. First argument is maximum value of the n. Second argument is an integer between [0,n]. Before ending, you will write running times(in milliseconds) of Pascal's triangle and combinatorial solution to screen.

Your project will also be evaluated with your results. So, you **must** obey output format. Each line of output file will contain: n, k and length(number of binary digits) of coefficient of $a^{n-k}b^k$. When k is equal to second argument of program and n is equal to first argument of program, you will print exact value of coefficient of $a^{n-k}b^k$, instead of its length.

For example, first argument is 7 and second is 4. You will write length of each of the coefficients for: $(a + b)^0$, ... , $(a + b)^7$ respectively. But instead of its binary length, you will write exact value of a^3b^4's coefficient. See sample.txt for output file of this example.
While testing your code, you can cross check Pascal's triangle and combinatorial solution. They will generate same results.

Bonus (up to 25 points)

You can achieve faster asymptotical bounds for combinatorial solution by optimizing multiplication and division algorithms. If you do not report your analysis and explanations for bonus parts, you will not be able to get any bonus.

Karatsuba multiplication (15 points): Instead of simple multiplication (consequent additions), use Karatsuba multiplication for BigUnsignedInteger. But there is a problem that can make Karatsuba multiplication inefficient. If you are able to get through the problem and make combinatorial solution asymptotically faster than before, you will get 10 points. In your report, explain Karatsuba multiplication, explain how you got through the problem and show new asymptotical bounds for combinatorial solution (5 points). If you are not able to get through the problem, you will only be rewarded with your explanations of the algorithm in your report (up to 5 points).

Division (10 points): Instead of simple division (consequent subtractions), use any algorithm that makes division for BigUnsignedInteger asymptotically faster than before. If you are able to achieve a better bound you will get 5 points. In your report, explain the algorithm that you used, show new asymptotical bounds for combinatorial solution (5 points). If you are not able to make division asymptotically faster, you will only be rewarded with your explanations of the algorithm in your report (up to 5 points).

Penalties (All penalties are cumulative)

- Poor software design can cost up to -%20 of your points from code section. You should not cause any unnecessary computation overhead, optimize the code as much as you can, but **do not make it hard to understand**. Your code must be clean and well commented, your variables and methods must have meaningful names. For an excellent list about what you should not do while developing good software, check this link: http://goo.gl/Emwoq

- In your project; crashes, wrong results and violations of I/O will be counted as runtime errors. These errors can cost up to -%70 of your points from code section. I will try to correct your error(s), (each mistake will cost up to -%20) but if they are more than I can handle, or the problem is bad design, you will get -%70 from your entire code and **lose all bonus points from codes** (but explanations of bonus parts will be graded).

- Compilation errors can cost up to -%100 of your points from code section. If your code is unable to compile under ITU's Linux server (-15 points), you must specify your development environment and compilation way in your report. Normally, your code must be compiled with this generic command:
  g++ *.cpp

Report (40 points)

- Explanation of classes:
  - BigUnsignedInteger: 6 points.
  - PascalsTriangle: 6 points.
  - CombinatorialSolution: 6 points.
  - If you have other classes, they will be evaluated too (points will be shared).
- Show that, $\binom{n}{k-1} + \binom{n}{k} = \binom{n+1}{k}$ where **1 ≤ k ≤ n**. Write your opinions about why you are asked to prove this equality: 2 points.

- Comparative analysis of Pascal's triangle and combinatorial solution: 20 points. Compare and contrast time and space complexities of those two methods. Please pay attention to this section of your report; it is the most important part of your entire homework.

Submission

You should be aware that the Ninova system clock may not be synchronized with your computer, watch, or cell phone. Do not e-mail the teaching assistants or the instructors your submission after the Ninova site submission has closed. If you have submitted to Ninova once and want to make any changes to your report, you should do it before the Ninova submission system closes. Your changes will not be accepted by e-mail. Connectivity problems to the Internet or to Ninova in the last few minutes are not valid excuses for being unable to submit. You should not risk leaving your submission to the last few minutes. After uploading to Ninova, check to make sure that your project appears there.

Cheating

You may discuss the problem addressed by the project at an abstract level with your classmates, but you should not share or copy code from your classmates or from the Internet. You have to submit your own, individual project. Plagiarism and any other forms of cheating will have serious consequences, including failing the course.

IMPORTANT: Due to tight schedule of academic calendar, it is impossible to extend deadline of this homework.

If there is something unclear, please let me know (guclusa@itu.edu.tr)