Fundamentals of Algorithms

Codelab 2: (100 points) Due on Monday, September 4th by 1:59 pm

Learning Outcomes

- Create a naive algorithm using brute force
- Design an efficient algorithm using divide-and-conquer
- Analyze the complexity of different algorithms

Instructions

In this codelab, you can use any development environment that you are most comfortable with. A much easier browser-based C++ editor is available at repl.it so that might be a good place to write your code.

Description

In this codelab, you will develop algorithms to multiply two n-digit numbers x and y where n is very large and thus must be represented using a data structure such as a list or array where each element holds a single digit. Suppose that in one instruction you can only multiply, add or subtract two 1-digit numbers. Your task is to design an algorithm to multiply two n-digit numbers using only basic arithmetic operations on 1 digit numbers.

Useful Observations: (1) Multiplying a number by a power of 10 can be implemented as a shift operation, (2) The addition or subtraction of two $\Theta(n)$ digit numbers can be performed in $\Theta(n)$ time (by breaking it down into $\Theta(n)$ additions of two one digit numbers).

Part 1 - The Naive Approach

Consider the algorithm you would use to multiply two n-digit numbers by hand and then argue that this method has time complexity Ω(n^2)? That is you are showing a lower bound on the time complexity of this naive algorithm. If you don’t believe your algorithm has time complexity Ω(n^2) let me know.

In this part of the lab, you will write a program with the following specifications:

1. Takes two numbers from user input (std::cin), check out the starter code that this link https://github.com/issao/hw-algorithms/blob/master/lab2/lab2_sample.cc
2. Stores each digit from the two numbers in as items in two integer arrays. For example, if the user inputted numbers 13, 45, then they would be stored as a[0] = 1, a[1] = 3, b[0] 4, b[1] = 5.
3. Create a shift-left function that takes two integer arrays and an integer as arguments where the first integer array is the input number, the second array is the output number, and the integer is the number of positions to shift left.
4. Create an addition function that takes three integer arrays as arguments, where the first two arrays are used to store the two input numbers and the third array is used to store the output number.
5. Create a subtraction function that takes three integer arrays as arguments, where the first two arrays are used to store the two input numbers and the third array is used to store the output number.
6. Create a multiply function that takes three integer arrays as arguments, where the first two arrays are used to store the two input numbers and the third array is used to store the output number.
7. Performs n-digit multiplication by using only basic arithmetic operations on elements on the array by using the functions implemented above.
8. Output the result to the user (std::cout).
9. Save your work in a file called lab2_part1.cc.

Part 2 - The Divide-and-Conquer Approach

Let’s now design a divide-and-conquer algorithm for this problem. Divide x into xl and xr where xl is the most significant n/2 digits of x and xr is the least significant n/2 digits of x. So, for example, if x = 783621 then xl = 783 and xr = 621. In the same manner split y into yl and yr. Using the fact that x = xl * 10^(n/2) + xr and y = yl * 10^(n/2) + yr, complete the design of this algorithm in code. *Hints: Multiply out the formulas given above for x and y. Could you make use of the product of any n/2 digit numbers?*

In part 2 of the lab, you will update your code with the following specifications:

1. Split each of the two numbers into two halves (e.g. 7834 turns into 78, 34)
2. Determine the equation to use the four numbers to obtain the product of the two numbers provided by the user.
3. Re-use the shift-left, addition, and multiply functions created in part 1 to solve the

previous equation.

4. Output the result to the user (std::cout).
5. Save your work in a file called lab2_part2.cc.

Part 3 - The Recursive Divide-and-Conquer Approach

In part 2, you were able to split the original problem into subproblems by breaking each number into two halves and figuring out the correct equation to obtain the product of the original two numbers. Now apply this optimization recursively by breaking each subproblem into smaller subproblems recursively. Update your code to reflect this change.

In part 3 of the lab, you will update your code with the following specifications:

1. Recursively perform the operations from part 2
   a. split each number into halves
   b. use the equation to determine their product
   c. re-use the math operation functions from part 1 to find the product
2. Remember to identify the base case for your recursive function.
3. Output the result to the user (std::cout).
4. Save your work in a file called lab2_part3.cc.

Questions

Answer the questions below in a word document and save it as a PDF. Go to the course website at classroom.google.com and upload your answers as well as your C++ files under Codelab 2.

1. For part 1, write a formula with the variable n (where n is the number of digits that captures the number of multiplications, additions, and subtractions. (10 points)
2. For part 2, write a formula with the variable n (where n is the number of digits) that captures the number of multiplications, additions, and subtractions. (10 points)
3. How many subproblems of size roughly n/2 do you now need to solve in order to then combine to solve the original problem? (10 points)
4. Write the recurrence equation and asymptotic time complexity of part 3. (10 points)
5. Which of the three algorithms is the fastest and why? (10 points)
6. Upload your code for all three parts to classroom.google.com (50 points)