CS 217 – Algorithm Design and Analysis

Shanghai Jiaotong University, Spring 2021

Handed out on 2021-02-25
First submission and questions due on 2021-03-04
You will receive feedback from the TA.
Final submission due on 2021-03-11 before class

1 Bit Complexity, Recursion, and Dynamic Programming

1.1 Bit Complexity of Euclid's Algorithm

We have proved that Euclid's algorithm for computing $\gcd(a, b)$ makes at most $O(\log a)$ iterations. What is the overall running time? Each iteration computes $u \mod v$ for some integers. This can be done by integer division. What is its running time? There are very sophisticated algorithms, but python probably does not come with them. Recall the "school method" for dividing integers. Have a look at the pdf slides on the webpage for an illustration of the school method. It is especially simple if we are dealing with binary numbers. If a and b have at most n bits, then the school method has complexity $O(n^2)$.

Exercise 1. Show the following, more precise bound of the school method for integer division: If a has n bits and b has k bits, then the school method can be implemented to run in $O(k(n - k))$ operations.

Exercise 2. Show that the bit complexity of Euclid's algorithm, using the school method to compute $a \mod b$, is $O(n^2)$. That is, if a and b have at

most n bits, then $\gcd(a, b)$ makes $O(n^2)$ bit operations.

In order to do so, here is python code of the Euclidean algorithm:

```
def euclid(a,b):
  while (b > 0):
      r = a % b # so a = bu+r
      if (r == 0):
          return b
      s = b % r # so b = rv + s
      a = r
      b = s
  return a
```

Don't be afraid to introduce notation! I recommend to let n_0 denote the number of bits of a. Take some other letters for the number of bits in b and so on.

1.2   Computing the Binomial Coefficient

Next, we will investigate the binomial coefficient $\binom{n}{k}$, which you might also know by the notation C_n^k. The number $\binom{n}{k}$ is defined as the number of subsets of $\{1, \ldots, n\}$ which have size exactly k. This immediately shows that $\binom{n}{k}$ is 0 if k is negative or larger than n. You might have seen the following recurrence:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \text{ if } n, k \geq 0 \; .$$

Exercise 3. [A Recursive Algorithm for the Binomial Coefficient] Using pseudocode, write a recursive algorithm computing $\binom{n}{k}$. Implement it in python! What is the running time of your algorithm, in terms of n and k? Would you say it is an efficient algorithm? Why or why not?

Exercise 4. [A Dynamic Programming Algorithm for the Binomial Coefficient] Using pseudocode, write a dynamic programming algorithm computing $\binom{n}{k}$. Implement it in python! What is it running time in terms of n and k? Would you say your algorithm is efficient? Why or why not?

Exercise 5. [Binomial Coefficient modulo 2] Suppose we are only interested in whether $\binom{n}{k}$ is even or odd, i.e., we want to compute $\binom{n}{k} \mod 2$. You could do this by computing $\binom{n}{k}$ using dynamic programming and then taking the result modulo 2. What is the running time? Would you say this algorithm is efficient? Why or why not?

Exercise 6. Let $k \geq 2$ be a fixed integer. Find an efficient algorithm that takes as input an integer n and computer $F_n \mod k$. Be careful: your input size is roughly $\log(n)$ and your output size is roughly $\log(k) = O(1)$. So a running time of $O((\log(n))^3)$ would be considered *efficient*, since it is polynomial in the size of input + output, but a running time of $O(n)$ would be considered *inefficient*, since it is *exponential* in the input size: $O\left(2^{\text{bit size of input}}\right)$.

Python. Please write your code in python. It is a very simple programming language. If you do not know any python, I can put same example code online and you can learn by example. Also, for this homework you definitely need a Big Integer class since numbers with ten thousand digits do not fit into any `long long int` or similar. Python automatically supports Big Integer, so there is no problem here.