CS 214 – Algorithms and Complexity

Shanghai Jiaotong University, Fall 2016

handed out on 2016-09-30, due on 2016-10-09 by email

We will pose challenging homework problems in this course and have a feedback phase for each homework. After you hand in your solution, we will get back to you with comments like "proof of Exercise Bla not correct" or "please make Definition... more precise". Then you can have another go at it. Therefore, not every due date will fall on a day with class, so please hand in your solution by email.

Your solution should be written in a way that a third person (for example a colleague of you in the same class) could understand it.

Please solve the homework assignment in your group and hand in *one* solution per group. Do not copy solutions from other groups! If you are completely stuck, you may ask me for advice!

1 Recursion and Dynamic Programming

In this homework we will investigate the binomial coefficient $\binom{n}{k}$, which you might also know by the notation C_k^n. 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:
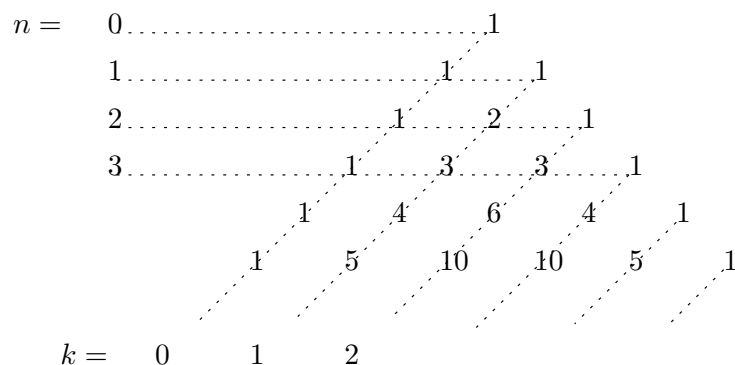
$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \text{ if } n, k \geq 0 .$$

Exercise 1. [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 2. [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 3. [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?

You might have heard that the numbers $\binom{n}{k}$ can be arranged to form Pascal's triangle[1]. Here is its top part:

$$
\begin{array}{ccccccccccc}
n = & 0 & & & & & & & 1 \\
& 1 & & & & & & 1 & & 1 \\
& 2 & & & & & 1 & & 2 & & 1 \\
& 3 & & & & 1 & & 3 & & 3 & & 1 \\
& & & 1 & & 4 & & 6 & & 4 & & 1 \\
& & 1 & & 5 & & 10 & & 10 & & 5 & & 1 \\
\\
k = & & 0 & & 1 & & 2
\end{array}
$$

If we draw a black dot for every odd number in this triangle (and nothing for an even number), we get the *Sierpinski triangle*:

---

[1] Also known as Yanghui triangle or Khayyam triangle, after the Chinese (respectively Iranian) mathematician who described it. Read the Wikipedia page!
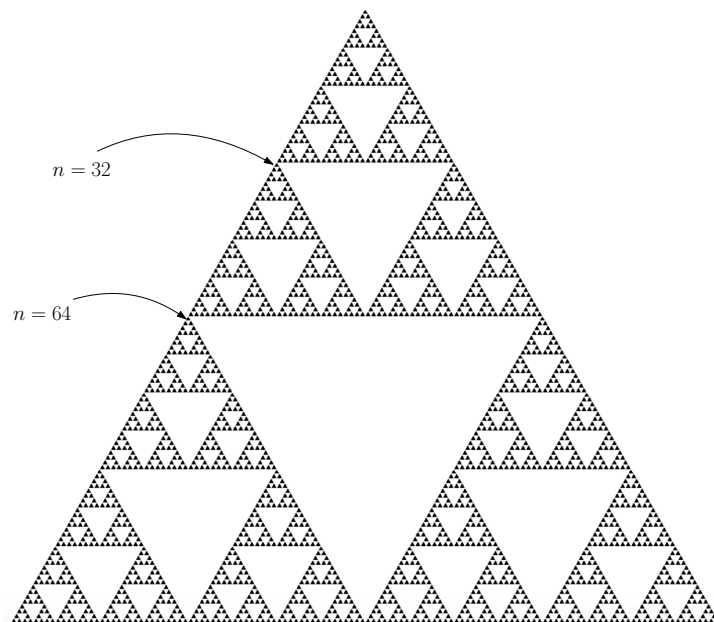
Notice its strong recursive structure. In particular, if $n = 2^d$ is a power of 2, then the nth row of the triangle is completely white, except for the two endpoints. More formally, we have the following lemma:

**Lemma 4.** *If $n = 2^d$ is a power of 2, then $\binom{n}{k} = 0$ for all $1 \le k \le n - 1$.*

Exercise 5. [Binomial Coefficient modulo 2, Recursive] Give a recursive algorithm for computing $\binom{n}{k}$ mod 2 using the above lemma. Your algorithm should not have problems handling numbers n and k with tens of thousands of digits. Implement your algorithm in python!

Hint. Look closely at the structure in the picture above. If you want to reason more formally, here is a good way to think of $\binom{n}{k}$: it is the number of paths leading from the location of $\binom{n}{k}$ to the top of Pascal's triangle, if you are only allowed left-up and right-up moves, as shown in this picture:
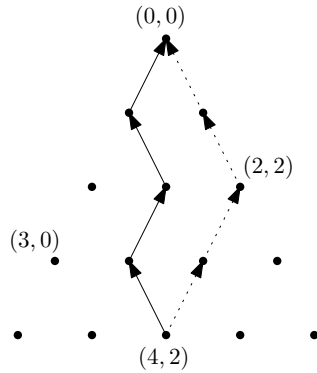
two of the $\binom{4}{2} = 6$ paths leading from $(4, 2)$ to $(0, 0)$.

Exercise 6. Remember the "period" algorithm for computing $F'_n := (F_n \bmod k)$ discussed in class: (1) find some i, j between 0 and k^2 for which $F'_i = F'_j$ and $F'_{i+1} = F'_{j+1}k$. Then for $d := j - i$ the sequence F'_n will repeat every d steps, as there will be a cycle. This cycle can either be a "true cycle" or a "lasso":
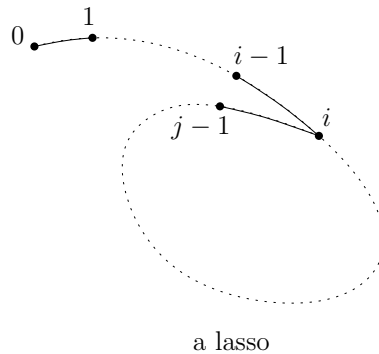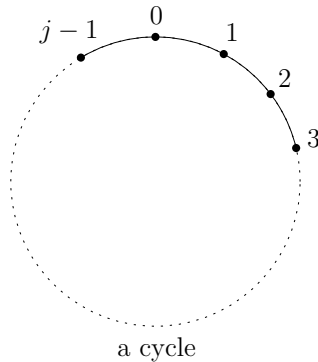
Show that a lasso cannot happen. That is, show that the smallest i for which this happens is 0, i.e, for some j we have $F'_0 = F'_j$ and $F'_1 = F'_{j+1}$ and thus $F'_n = F'_{n \bmod j}$.

Exercise 7. Come up with a question related to the topic of this homework or the last lectures.

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.