

# Algorithm Design and Analysis

## Homework Assignment 1

**Group Name:** Static  
**Group Members:** Zhengjia Xu, JiaCheng Yang  
Zhuolin Yang, Wenda Qiu

8<sup>th</sup> Oct, 2016

### 1 Recursion and Dynamic Programming

**Exercise 1.1.** This complexity of the recursion process mainly depends on the additions. Let's define  $F_{i,j}$  as when such recursion process start at calculating  $\binom{i}{j}$ , the number of additions it will do.

Obvirusly,  $F_{i,j}$  consists of three parts below:

- $F_{i-1,j-1}$ : add the number of addition operations in  $\binom{i-1}{j-1}$  to the current answer.
- $F_{i-1,j}$ : add the  $\binom{i-1}{j-1}$ 's answer.
- 1: the number of additions which used to add  $\binom{i-1}{j-1}$  and  $\binom{i-1}{j}$  up.

Thus we have

$$F_{i,j} = F_{i-1,j-1} + F_{i-1,j} + 1 \quad (1)$$

This is a quite strange and Pascal-like equation. Consider

$$(F_{i,j} + 1) = (F_{i-1,j-1} + 1) + (F_{i-1,j} + 1)$$

We can easily find out that

$$F_{i,j} + 1 = \binom{i}{j} + \binom{i}{j-1} + \binom{i}{j+1}$$

Therefore

$$F_{i,j} = \binom{i}{j} + \binom{i}{j-1} + \binom{i}{j+1} - 1 \quad (2)$$

Hence, the total number of addition operation is

$$\binom{n}{k} + \binom{n}{k-1} + \binom{n}{k+1} - 1$$

The lower bound of the complexity of adding two numbers together being  $O(1)$  and the upper bound being  $O(n)$ , we can easily figure out that the lower bound of the complexity of this algorithm is

$$O\left(\binom{n}{k} + \binom{n}{k-1} + \binom{n}{k+1}\right) = O\left(\binom{n}{k}\right)$$

While the upper bound is

$$O\left(n\binom{n}{k}\right)$$

When  $n = 50, k = 25$ , this algorithm needs to do about  $1.3 \times 10^{16}$  additions, which is too slow. Hence this algorithm is inefficient.

**Exercise 1.2.** Since we use dynamic programming, the running time depends on how many binomial coefficients we calculate. Consider the formula

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

It's easy to see that the area we calculate is a parallelogram in Pascal's triangle. More specifically, in order to get the answer of  $\binom{n}{k}$ , there are exactly  $n - k$  numbers we calculate in the columns from 1 to  $k$ , so we need to do  $O(k \cdot (n - k))$  additions.

The time we need for a single addition depends on the length of the result, namely  $\log \binom{n}{k}$ . Since

$$\sum \binom{n}{k} = 2^n$$

We can easily see that  $\binom{n}{k}$  has a length up to  $n$ . So the upper bound of the running time is  $O(nk \cdot (n - k))$ , which is not very efficient.

**Exercise 1.3.** The additions we need is the same as Ex2,  $O(k \cdot (n - k))$ . But this time the addition can be done in  $O(1)$  time because we only need to keep the result modulo 2. So the running time is  $O(k \cdot (n - k))$ , which is quite efficient.

**Exercise 1.4.**

#### 1. Proof by induction

*Proof.* We can easily prove the lemma using induction:

##### (a) Basic step

When  $n = 2^1$ , the lemma obviously holds because

$$\binom{2}{1} = 0$$

##### (b) Induction step

Assume that the lemma holds where  $n$  equals to  $2^d$ , namely

$$\binom{n}{k} = 0$$

when  $1 \leq k \leq n - 1$ .

Then we will show that the lemma holds when  $n$  equals to  $2^{d+1}$ .

According to the hints, on the one hand, the number of paths to the  $n$ -th line and the  $k$ -th grid equals to

$$\binom{n}{k}$$

On the other hand, we can also calculate the number by Multiplication Principle and Addition Principle, namely

$$\binom{n}{k} = \sum_{i=0}^{n/2} \binom{n/2}{i} * \binom{n/2}{k-i} \quad (3)$$

Then our proof divides into three cases:

- When  $1 \leq k \leq n/2 - 1$ , it is obviously that

$$\binom{n/2}{k} = 0$$

and

$$\binom{n/2}{0} = \binom{n/2}{n/2} = 1$$

According to the equation (3), we have

$$\binom{n}{k} = \binom{n/2}{k-0} + \binom{n/2}{k-n/2}$$

By assumption, when  $1 \leq k < n/2$ , we have

$$\binom{n/2}{k} = 0$$

Notice that  $k - n/2 < n/2$ , so we have

$$\binom{n/2}{k-n/2} = 0$$

Therefore

$$\binom{n}{k} = 0$$

- When  $k = n/2$ , it is obviously that

$$\binom{n/2}{k} = \binom{n/2}{k-n/2} = 1$$

Therefore

$$\binom{n}{k} = (1 + 1) = 2 \equiv 0 \pmod{2}$$

- When  $n/2 \leq k < n$ , it is easy to see that

$$\binom{n/2}{k} = 0$$

And by assumption

$$\binom{n/2}{k-n/2} = 0$$

So that we have

$$\binom{n}{k} = 0$$

(c) Conclusion By induction, we have

$$\binom{n}{k} \equiv 0 \pmod{2}$$

where  $1 \leq k \leq n-1$  and  $n = 2^d$  ( $d \geq 1$ ).

□

## 2. Direct Proof

*Proof.* We can denote the binomial coefficient as  $2^p * q$ , namely

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = 2^p * q \quad (4)$$

where  $2 \nmid q$ ,  $n = 2^d$  ( $d \geq 1$ ) and  $1 \leq k \leq (n-1)$ .

We can easily calculate the exponent  $p$  with the formular below

$$p = \sum_{i=1}^d \left\lfloor \frac{n}{2^i} \right\rfloor - \sum_{i=1}^d \left\lfloor \frac{k}{2^i} \right\rfloor - \sum_{i=1}^d \left\lfloor \frac{n-k}{2^i} \right\rfloor$$

Specially when  $i = d$ , we have

$$\left\lfloor \frac{n}{2^d} \right\rfloor = 1 \text{ and } \left\lfloor \frac{k}{2^d} \right\rfloor = \left\lfloor \frac{n-k}{2^d} \right\rfloor = 0$$

According to the equation (4), we have

$$\begin{aligned} p &= 1 - 0 - 0 + \sum_{i=1}^{d-1} \left\lfloor \frac{n}{2^i} \right\rfloor - \left\lfloor \frac{k}{2^i} \right\rfloor - \left\lfloor \frac{n-k}{2^i} \right\rfloor \\ &\geq 1 + \sum_{i=1}^{d-1} \frac{n}{2^i} - \frac{k}{2^i} - \frac{n-k}{2^i} = 1 \end{aligned}$$

Hence we have

$$2 \mid \binom{n}{k}$$

That is

$$\binom{n}{k} \equiv 0 \pmod{2}$$

□

**Exercise 1.5.** We can easily come out an algorithm using the formular which is proved in Ex5.

---

**Algorithm 1** An efficient method calculating the binomial coefficient

---

```

1: function BINOM( $n, k, p$ )
2:   if  $m > n$  or  $m < 0$  then
3:     return 0
4:   end if
5:   if  $n < 2$  then :
6:     return 1
7:   end if
8:   if  $p = 0$  then :
9:      $p = 1$ 
10:    while  $p * 2 \geq n$  do
11:       $p \leftarrow p * 2$ 
12:    end while
13:  else
14:    while  $p \geq n$  do
15:       $p \leftarrow p / 2$ 
16:    end while
17:  end if
18:  return (BINOM( $n - p, m - p, p$ ) + BINOM( $n - p, m, p$ )) mod2
19: end function

```

---

The implementation in python is displayed below

```

1 def binom(n, m, p = 0):
2     if (m > n or m < 0):
3         return 0;
4     if (n < 2):
5         return 1;
6     if (p == 0):
7         p = 1;
8         while (p * 2 < n):
9             p = p * 2;
10    else:
11        while (p >= n):
12            p = p / 2;
13    return binom(n - p, m - p, p) ^ binom(n - p, m, p);

```

**Exercise 1.6.** Assume it will form a rho-like process. Considering  $i$  is the entrance of the “circle” of the proceses, we have

$$F'_i = F'_j \text{ and } F'_{i+1} = F'_{j+1}$$

But

$$F'_{i-1} \neq F'_{j-1}$$

If not, you can easily see that  $i - 1$  is the entrance of the “circle”.

But according to the formula

$$F'_{i+1} = F'_i + F'_{i-1}$$

We have

$$F'_i = F'_j \text{ and } F'_{i+1} = F'_{j+1} \Rightarrow F'_{i-1} = F'_{j-1}$$

which is a contradictory. Thus the assumption does not hold.

Consider every  $F'_i, F'_{i+1}$  can form a pair  $\{F'_i, F'_{i+1}\}$ . Because  $F'_i \in [0, k)$ , the number of such pair will not exceed  $k^2$ . According to the Pigeon's Theorem, there must be a cycle-like process whose length doesn't exceed  $k^2$ , so we can find  $j$  which satisfied  $F'_0 = F'_j$ ,  $F'_1 = F'_{j+1}$  and  $F'_n = F'_{n \bmod j}$  within the complexity  $O(k^2)$ .