# Algorithm Design and Analysis

Not Strong Enough

March 7, 2020

## 1 Homework

**1: Problem 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?

**Algorithm 1** Caluculate Binomial Coefficient Using DP

Create a 2-dimension array $G$
**for** $i = 0$ to $n$ **do**
  $G[i][0] = 1$
**end for**
**for** $i = 0$ to $k$ **do**
  $G[i][i] = 1$
**end for**
**for** $i = 1$ to $n$ **do**
  **for** $j = 1$ to $\min(i-1, k)$ **do**
    $G[i][j] = G[i-1][j] + G[i-1][j-1]$
  **end for**
**end for**
**return** $G[n][k]$

```python
def calc_dp(n,k):
    arr = [[0 for i in range(k+1)] for j in range(n+1)]
    for i in range(n+1):
        arr[i][0]=1
    for i in range(k+1):
        arr[i][i]=1
```

```
    for i in range(1,n+1):
        for j in range(1,min(i-1,k)+1):
            arr[i][j]=arr[i-1][j]+arr[i-1][j-1]
return arr[n][k]
```

complexity analysis:

When we traverse the array and visit `arr[i][j]`, we actually perform the add operation $\binom{i}{j} = \binom{i-1}{j} + \binom{i-1}{j-1}$. So the operation cost O($\log\binom{i}{j}$) time. Using the Stirling Formula, we can estimate $\log\binom{i}{j} \approx (i+\frac{1}{2})\log i - (i-j+\frac{1}{2})\log(i-j) - (j+\frac{1}{2})\log j = \mathrm{O}(i)$. The number of nodes we visit is $\mathrm{O}(k*(2n-k)/2)$=$\mathrm{O}(kn)$. Since we will only visit `arr[i][j]` once, we can estimate the total complexity as below: the upper bound is $O(kn*n)$=$O(kn^2)$, the lower bound is $\Omega(kn)$. The algorithm is efficient, because there is no redundant calculation.