Homework 5 (計算方法設計，Design and Analysis of Algorithms)

<u>**Due date: June 2, 2021**</u>

1. (40%) Suppose that among n identical-looking coins, one is fake, where $n \geq 3$. Moreover, it is unknown whether the fake coin is lighter or heavier than the genuine one. With a balance scale, you can compare any two sets of coins. That is, by tipping to the left, to the right, or staying even, the balance scale will tell you whether the two sets weight the same, or which of the two sets is heavier than the other but not by how much. Please use the prune and search strategy to design an efficient algorithm for detecting the fake coin (20%). Please also analyze the number of weighting needed by your algorithm in worst case (20%).

2. (60%) As mentioned in the class, the selection problem can be solved by the following two methods. The first method is to use the quick sort to sort the given n numbers in ascending order and pick out the k-th number from the sorted array. The second method is to utilize the prune and search strategy as introduced in the class. Use a programming language you familiar with to implement these two methods for solving the selection problem (40%) and compare their running time when the input size n is from 10000 to 30000 in steps of 1000 (20%). Note that you should implement the quick sort program by yourself, instead of using the built-in quick sort function in the programming language. Note also that for each n, you should generate three problem instances and average the running time of solving these three instances.

1. Algorithm

①

```
Detecting_fake_coin( n coins )
    n = the number of coins
    if (n==1) :
        coin is fake:
    else:
        Deviding coins into 3 groups:
            A group : ⌈n/3⌉ coins
            B group : ⌈n/3⌉ coins
            C group : n - 2 × ⌈n/3⌉ coins (Rest of the coins)
        Comparing A and B
        if( A.weight == B.weight ):
            Detecting_fake_coin( C group )
        else:
            Detecting_fake_coin( A group ∪ B group )
```

② $T(n) = T(\frac{2n}{3}) + \overset{f(n)}{O(1)}$ By Master Method, $a = 1$ $b = \frac{3}{2}$

$O(n^{\log_{\frac{3}{2}} 1}) = O(1) = f(n)$ ⇒ Case ② $T(n) = \Theta(\log_{\frac{3}{2}} n)$ #

Worst case: Deleting $\frac{1}{3}n$ Each Time

$n \cdot (\frac{2}{3})^k = 1$ $n = (\frac{3}{2})^k$ $\log n / \log(\frac{3}{2}) = k$ ⇒ $\frac{\log n}{\log(\frac{3}{2})}$ =次 #