CSE 6140 Assignment 1
Due Sept. 15, 2016
Jingtian Zhang

1 Simple Complexity

1.
(a) $f \in \Theta(g)$
(b) $f \in \Theta(g)$
(c) $f \in \Omega(g)$
(d) $f \in O(g)$
(e) $f \in \Omega(g)$
(f) $f \in O(g)$

2.
Answer: The Big-O time complexity for this algorithm should be $nlog(n)$. Apparently, the outer while loop is increasing by 1 every time so that it will loop totally n times. For the inner while loop, it is kind of a Divide and Conquer so that it will loop logarithm of k. K at most is n. So in total, the time complexity should be $nlog(n)$.
As for the output, the format is: "i, 0" when 3 cannot divide i; and "i, $\log_3 k$". The log part for second case will only be presented as integer. For example, when i is 6, it will output "6, 1"

2 Greedy – Why is the pool always so busy anyway?

The time interval for each athlete can be treated as two major parts. One is the bottle-necked swimming section and the other is running plus biking. The running plus biking is an independent part, which means that the time consumption for this part will not be affected by the order of arrangement of athletes. While for the swimming part, the exact time consumption is time waiting for athletes ahead of him finishing swimming plus his own swimming time consumption. Therefore, the first athlete will have minimum time consumption in swimming part and last athlete will have maximum time consumption in swimming part. A general expression for time consumption for an athlete k is $a_k = (s_1 + s_2 + \cdots + s_k) + r_k + b_k$
The total finishing time should be $\max(a_1, \dots a_k, \dots a_n)$. Thus, my solution for this problem is:
Sort the time consumption for $r_i + b_i$ in descending order for the athletes. The swimming sequence should be following this order so that the result is optimum.

Proof.
The greedy solution is $A = \{a_1, a_2, \dots a_n\}$ from my solution and optimal solution is $O = \{O_1, O_2, \dots, O_n\}$.
Assume there are two athletes swimming sequence in O different with the greedy solution A (an inversion). Mark these two athletes as i, j.
Since there is an inversion in optimum solution O such that $r_i + b_i < r_j + b_j$. If we swap them, the j athlete is ahead of i athlete. Assume the total swimming time ahead of athlete i is s and swimming time between them is δs. The total time at the end of athlete j should be $r_j + b_j + s + \delta s + s_i + s_j$. The total finishing time after ij swapping is $r_i + b_i + s + \delta s + s_i + s_j$. The second

finishing time is earlier than first finishing time. This indicates that by swapping the inversion, we have no worse solution in greedy algorithm. So if we keep swapping the difference between solution O and greedy solution A, we will have no worsen solution and thus Greedy solution A is best solution.

3 Divide and Conquer

Mark all the tags in a sequence order such as 1,2…n. Assume they are put into an array like data structure. Here is the pseudo code below:

```
Function getMajority(a[1,2,3…n])
If n == 1
        Return a[1]
k = n/2
e1 = getMajority(a[1,…k])
e2 = getMajority(a[k+1,…n])
If e1 == null or e2 == null:
        Return null
If e1 == e2:
        Return e1
f1 = checkFrequency(a[1,..n],e1)
f2 = checkFrequency(a[1,..n],e2)
If f1 >= k+1:
        Return e1
Else if f2 >= k+1:
        Return e2
Else:
        Return null

Function checkFrequency(a[1,…n],e)
Count = 0
For i = 1:n:
        If a[i] == e:
                Count++
Return count
```

Explanation: The base case is when the array size is 1, then we return the element automatically. When we get a majority element candidate, we need to check the frequency of that element. If the element is majority, we return. If there is no majority element, we return null, which means there is no majority tag in this problem. Since everytime the K shrinks by 2 and checkFrequency takes O(N), the time complexity should be O(nlogn).