Homework 6 sample solution

Due 09/13/16

September 8, 2016

1. Analyze the *worst-case* time complexity of the algorithm below. Please show all work. The $\lfloor \ \rfloor$ symbols represent the *floor* ("round down") function. You may assume that this function takes $\Theta(1)$ time for any input. You may also assume it takes a constant amount of time to determine whether an integer is odd.

   Note that figuring out what problem this algorithm solves is *irrelevant* to analyzing its complexity.

```
   Input: n: nonnegative integer
1  Algorithm: LoopMystery
2  sum = 0
3  t = 1
4  d = 1
5  k = n
6  while k > 1 do
7      for i = 1 to k do
8          t = t + d
9          sum = sum + t
10     end
11     if k is odd then
12         d = -d
13     end
14     k = ⌊k/2⌋
15 end
16 return sum
```

   Answer: Lines 2–5, 8, 9, 11, 12, 14, and 16 all take $\Theta(1)$ time. Thus, the body of the for loop takes a total of $\Theta(1)$ time. Since the for loop iterates k times, it must take $\Theta(k)$ time total. This complexity dominates the cost of each iteration of the while loop, so each iteration also takes $\Theta(k)$ time. In the worst case, $k = k/2$ (not $(k-1)/2$) in line 14, so the while loop will iterate $O(\lg n)$ times before $k = 1$. Since the time complexity of an iteration changes as k shrinks, we need to sum up the for loop iterations.

Since k is halved each iteration of the while loop, this sum is:

$$\begin{aligned}
O(n) + O(n/2) + O(n/4) + \ldots + O(1) &= O(n + n/2 + n/4 + \ldots + 1) \\
&= O(n(1 + 1/2 + 1/4 + \ldots + 1/n)) \\
&= O(n \sum_{i=1}^{\lg n} 1/2^i) \\
&= O(n(1)) \\
&= O(n)
\end{aligned}$$

As the return statement in line 16 takes $\Theta(1)$ time, the total time for LoopMystery is $O(n) + \Theta(1) = O(n)$.