

CS375 Assignment 6

Due October 6th 10PM ET

A general note for CS375: When writing up your homework, please write neatly and explain your answers clearly, giving all details needed to make your answers easy to understand. Graders may not award full credit to incomplete or illegible solutions.

You are encouraged to discuss the problems with your classmates, **but your submission should be your own**. When writing your solutions up, you should not be looking at any other student's submission.

Clear communication **is the point**, on every assignment. In general in CS375, unless explicitly specified otherwise, answers should be accompanied by explanations. Answers without explanations may not receive full credit. Please feel free to ask me any questions about explanations that might come up!

Necessary acknowledgements: many of these questions are reproductions or adaptations of questions of Eric Aaron's devising. Others are from our course texts as listed on the syllabus.

1. To warm up, a couple things we needed from lecture:

- (a) Show via induction that $\sum_{i=0}^n r^i = \frac{r^{n+1}-1}{r-1}$ (we used this sum in class today)
- (b) Show that $a^{\log_b(c)} = c^{\log_b(a)}$. We didn't use this today, but it'll simplify things going forward. Recall we had to deal with $3^{\log_2(n)}$ - by applying this identity, we get $3^{\log_2(n)} = n^{\log_2(3)}$, which is faster than how we went about it.

2. (a)

```
def InsertionSort(A[1, ...], n):
    if n == 1:
        return

    InsertionSort(A, n - 1)
    j = n
    while j > 1 and A[j] < A[j - 1]:
        swap A[j] and A[j - 1]
        j -= 1
```

- i. Explain why this algorithm is **exactly** the same as the algorithm we wrote in class for InsertionSort.
 - ii. **By means of induction**, explain why this algorithm is correct (that it correctly sorts the list).
 - iii. Explain why the runtime of this function is given by $T(1) = \Theta(1)$ and $T(n) \leq T(n-1) + \Theta(n)$.
 - iv. Use the expansion method we used in class to determine the runtime of $T(n)$.
- (b) For this one, you will need to know what the Tower of Hanoi game is. If you don't know the game, Erickson has a good section on it in the reading tonight :)

```
def TowerOfHanoi(n, source, destination, spare):
    if n == 1:
        move disk 1 from the source to the destination
        return

    TowerOfHanoi(n - 1, source, spare, destination)
    move disk n from source to destination
    TowerOfHanoi(n - 1, spare, destination, source)
```

- i. Explain why this algorithm never moves a larger tile on top of a smaller one (note that the original state is all the tiles on the left peg, arranged from biggest to smallest with the biggest on the bottom).
- ii. Explain why the number of moves made by this algorithm is given by $T(1) = 1$ and $T(n) \leq 2T(n-1) + 1$.
- iii. Use the expansion method we used in class to determine $T(n)$.