2016 Fall CS300 Homework #2

TA in charge: Chungkwon Ryu (ryuch91@kaist.ac.kr)

Due: October 13, AM 10:30 on classroom

1. (20pts) **Quick Sort**

```
RANDOMIZED-PARTITION(A, p, r)
1 i = RANDOM(p,r)
2 exchange A[r] with A[i]
3 return PARTITION(A,p,r)
```

```
RANDOMIZED-QUICKSORT(A, p, r)
1 If p < r
2       q = RANDOMIZED-PARTITION(A, p, r)
3       RANDOMIZED-QUICKSORT(A, p, q-1)
4       RANDOMIZED-QUICKSORT(A,q+1,r)
```

(a) Show that quicksort's best-case running time is $\Omega(n \lg n)$.

(b) Show that RANDOMIZED-QUICKSORT's expected running time is $\Omega(n \lg n)$.

2. (40pts) **Divide and Conquer**

Suppose there are three alternatives for dividing a problem of size n into sub-problems of smaller size: if you solve 3 sub-problems of size n/2, then the cost for combining the solutions of the sub-problems to obtain a solution for the original problem is $\Theta(n^2\sqrt{n})$; if you solve 4 sub-problems of size n/2, then the cost for combining the solutions is $\Theta(n^2)$; if you solve 5 sub-problems of size n/2, then the cost for combining the solutions is $\Theta(n \log n)$. Which alternative do you prefer and why?

3. (40pts) **Quick Sort**

(In practice) The running time of quicksort is able to be improved by taking advantage of the fast running time of insertion sort when its input is "nearly" sorted. When quicksort is called on a subarray with fewer than k elements, let it simply return without sorting the subarray. After the top-level call to quicksort returns, run insertion sort on the entire array to finish the sorting process. Argue that this sorting algorithm runs in $O(nk+n\log(n/k))$ expected time. How should k be picked, both in theory and in practice? (*In practice, consider constant factors in each sort.)