EECS340 - Algorithms - HW#5

Mark Schultz - mxs802

October 10, 2010

7.2-5

For the minimum depth, the smaller part of the partition is always taken. Therefore, i iterations reduce the number of elements to $\alpha^i n$. If m is the minimum depth of a leaf we can assume $\alpha^m n = 1$ and $\alpha^m = 1/n$. If we take the log to solve for m, we get $m \lg \alpha = -\lg n \rightarrow m = -\lg n / \lg \alpha$.

In comparison, for max depth, the larger part of the partition is always taken resulting in the fraction $1 - \alpha$ elements at all times. When there is one element left, the max depth M, has been reached. This is shown by $(1 - \alpha)^M n = 1 \rightarrow M = -\lg n / \lg(1 - \alpha)$.

7-2

a) Because all elements are equal, randomized quicksort would have the same running time as the original quicksort. Also because every element is the same, the worst case is achieved and therefore $\Theta(n^2)$.

b) I would add the following code before the loop in the pseudo code to produce the desired results.

```
1: i = p − 1
2: for j = p to r − 1 do
3:     if A[j] = x then
4:         exchange A[i] with A[r − 1]
5:     end if
6: end for
```

c)
```
1: RANDOMIZED-QUICKSORT(A,p,r)
2: if p < r then
3:     q, t = RANDOMIZED-PARTITION(A,p,r)
4:     RANDOMIZED-QUICKSORT(A,p,q-1)
5:     RANDOMIZED-QUICKSORT(A,t,r)
6: end if
```

d) Adjusting the algorithm gives us expected $\Theta(n - s \cdot \lg n)$ where s is the number of elements that are the same. This takes into account the fact that the adjusted randomized-quicksort does not touch items that are deemed the same.