Intro to Algorithms, COMP-160, Homework #6
Benjamin Tanen, 03/09/2016

2. In randomized selection, we pick a pivot, partition and find its rank, and then recurse on one side. Given n elements, define a pivot as *balanced* if each side of the partition ends up with at least a constant fraction $\frac{1}{c} \cdot n$ of the input. Otherwise, a pivot is called *unbalanced*. In both cases, the partition takes $d \cdot n$ time.

   Suppose that when we run the algorithm, that every balanced pivot is followed by u unbalanced pivots, where u is a constant.

   Show that under the above conditions, the algorithm will take $O(n)$ time in the worst case. As always for upper bounds, exaggerate and simplify.

   In the case of a balanced partition, each recursion would simply be a constant fraction of the previous work, or $T_B(n) = T(\frac{n}{c}) + \Theta(n)$. For an unbalanced partition however, we are not sure how many elements our recursion will be working on. The worst case would be where we pick a partition that is either greater than or less than all of the other elements in our list. In this case, our partitioning can only reduce our list size / work by one element, so for an unbalanced partition we see $T_U(n) = T(n-1) + \Theta(n)$.

   Consider the scenario where we get one balanced partition followed by u unbalanced partition. Since our first partition is balanced, we would recurse on a fraction of our list ($\frac{n}{c}$). After this, each unbalanced partition would recurse on one less elements. If this process unbalanced partitioning is repeated u times, we can get:

$$T(n) = T_B(n) + T_U(\frac{n}{c}) + T_U(\frac{n}{c} - 1) + \ldots + T_U(\frac{n}{c} - (u-1))$$

$$T(n) = \Theta(n) + T(\frac{n}{c}) + \Theta(\frac{n}{c}) + T(\frac{n}{c} - 1) + \Theta(\frac{n}{c} - 1) + \ldots + \Theta(\frac{n}{c} - (u-1)) + T(\frac{n}{c} - u)$$

$$T(n) = u\Theta(n) + \sum_{i=0}^{u} T(\frac{n}{c} - i)$$

   We know that $T(\frac{n}{c} - i) \leq T(\frac{n}{c})$ for all i $\in \mathbb{Z}^+$. We can also express $u\Theta(n)$ as udn (d is a constant). Therefore, we can exaggerate our recurrence to say:

$$T(n) \leq udn + T(\frac{n}{c}) + u \cdot T(\frac{n}{c}) = udn + u \cdot T(\frac{n}{c})$$

$$T(n) \leq udn + u \cdot T(\frac{n}{c}) = u(dn + T(\frac{n}{c}))$$

   From here, since u is a constant, we can use the master method by temporarily disregarding u. We therefore get the following:

$$T(n) \leq u(dn + T(\frac{n}{c})) \rightarrow T_1(n) = dn + T(\frac{n}{c})$$

$$a = 1, b = c, f(n) = dn$$

$$n^{\log_c 1} = n^0 = 1, \text{ so } f(n) \text{ dominates}$$

$$\text{Thus } T_1(n) = O(f(n)) = O(dn) = O(n)$$
$$\text{So } T(n) = uT_1(n) = \Theta(1)O(n) = O(n)$$

Therefore, we see that our partitioning is linear.