Brandon Vo

HW5

a) Given a set S of n random elements in an array without duplicates, if we pick one of these elements, p, randomly, we know how to find the rank of p, by using p as a pivot to partition S, in *dn* time. Here, d is some positive constant. Now suppose that we pick three random elements, p_1, p_2, p_3. Show how to find all three ranks (and partition S into four groups), in significantly less than 3*dn* time. This means that you should have a leading constant that is smaller than 3.

1. Arrange p_1, p_2, and p_3 such that $p_1<p_2<p_3$. Select any element that is not p_1, p_2, or p_3 and compare them with p_2. For now, treat p_2 as though it were rank k.
2. If the element chosen is $< p_2$, then place it to the left of p_2. Otherwise, place it to the right of p_2. Repeat this process until all elements have been grouped up in terms their relative size to p_2.

This is splitting the array of n numbers into two groups. To group all numbers, it would take dn time when using p_2 as the pivot.

3. After dividing the set S by p_2, we have two grouped set of numbers. In one set, the numbers range from 0 to p_1 to p_2. Use p_1 as the new rank k and group the numbers based on their comparison with p_1. Repeat this process until there is a set of numbers $< p_1$ on the left of p_1 and all numbers that are greater than p_1 are placed to the right of p_1.

When using p_1 as rank k, there are k-1 elements to sort through out of n elements, excluding p_2. This will take $(k-1)d$ time to sort through the elements within this group when using p_1 as the pivot.

When using p_3 as rank k, there will be n-k elements to sort through. The time to make these comparisons using p_3 as the pivot will take $(n-k)d$ time.

4. The other set of grouped numbers will have a range from p_2 to p_3 to n. Hold rank k at p_3 and group the numbers based on their comparison to p_3. All numbers $< p_3$ are placed to the right of p_2 but to the left of p_3. The numbers $> p_3$ are placed to the right of p_3.

$$T(n) = dn\ (p_2\ pivot) + (n-k)d\ (p_3\ pivot) + (k-1)d\ (p_1\ pivot) = 2dn - d \leq 3dn$$

This process of creating four different groups takes less time than 3dn.

b) Say we want to find the median of S. Let's consider a variant of RandSelect: Suppose that we use three random elements from S. If all three have ranks that are outside of the $[\frac{n}{4}, \frac{3n}{4}]$ range, we just declare that we wasted our time, and pick three new random elements (this means repeats are possible). Otherwise, we use the element (among those three) with rank closest to the median, as a pivot to partition, and then recurse as usual.

Derive the probability that we waste our time, given our choice of three elements.

The range of $\left[\frac{n}{4}, \frac{3n}{4}\right]$ forms 3 quadrants. Quadrant 1 is $< \frac{n}{4}$, Quadrant 2 lies between $\frac{n}{4}$ and $\frac{3n}{4}$. Quadrant 3 is any number $> \frac{3n}{4}$. The probability that RandSelect chooses an element that wastes our time would be the probability that an element lands in quadrant 1 or quadrant 3.

Assuming that the elements of n are evenly distributed, we can assume that there is a fair probability of an element being placed anywhere within the range from 0 to n. The probability of an element being plaed

in quadrant 1 or quadrant 3 is $n - \frac{n}{2} = \frac{n}{2}$. The range of elements is from 0 to n, so the probability out of n elements is $\frac{\frac{n}{2}}{n} = \frac{1}{2}$ chances that one random element chosen will be outside the desired range.

For three elements with their own probabilities to be placed anywhere, each have an independent probability of choosing any number within the range of n elements. With repeats allowed, each random element will have ½ chances of choosing an element outside the desired range.

$$\frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{1}{8}\ chances\ for\ all\ 3\ elements\ to\ be\ chosen\ outside\ range\ \left[\frac{n}{4}, \frac{3n}{4}\right], wasting\ our\ time$$

c) Follow the reasoning on slide 7 of the new condensed course notes in section 8.2, to get an upper bound for the expected time complexity of this RandSelect variant. You may exaggerate as in the course notes. In fact, you should keep the calculation simple, on par with the course notes. Do you get a better bound than the standard algorithm? (factoring in the leading constant)

There are two scenarios with the algorithm: an unbalanced split or a balanced split. As mentioned in question b), the probability of 3 randomly chosen numbers not being within range is 1/8.

$$Balanced\ Split:\ T(n) \leq T\left(\frac{3n}{4}\right) + dn$$

The balanced split occurs if the pivot chosen is not within the range of $\left[\frac{n}{4}, \frac{3n}{4}\right]$. For our modified RandSelect variant, a Balanced Split occurs if all 3 numbers are not within our desired range which, as mentioned in question b), is 1/8 chances of occurring.

$$Unbalanced\ Split: T(n) \leq T(n-1) + dn$$

The unbalanced split occurs when the pivot is within the desired range. For the modified RandSelect algorithm, we have an unbalanced split when the three random elements chosen are within the desired range which is 7/8.

$$T(n) \leq \frac{1}{8} * (T(n) + 2dn) + \frac{7}{8} * \left(T\left(\frac{3n}{4}\right) + 2dn\right)$$

$$T(n) \leq \frac{1}{8} * T(n) + \frac{dn}{4} + \frac{7}{8} * T\left(\frac{3n}{4}\right) + \frac{7}{4}dn$$

$$\frac{7}{8} * T(n) \leq 2dn + \frac{7}{8} * T\left(\frac{3n}{4}\right)$$

$$T(n) \leq \frac{16}{7}dn + T\left(\frac{3n}{4}\right)$$

$$This\ is\ greater\ than\ the\ slideshow\ algorithm:\ T(n) \leq T\left(\frac{3n}{4}\right) + 2dn$$

This means that the modified version of RandSelect is more costly compared to the original version because there is an increased cost in constructing the ranks and partitions of 3 random numbers compared to the original RandSelect's 1 element partition.