1 Suppose we are looking for the 5th order statistic in an input array of 10 elements. Suppose that after partitioning the array, the pivot element ends up in the third position. On which side of the pivot element should we recurse, and what order statistic should we look for?

1. The 3rd order statistic on the left side of the pivot
2. The 2nd order statistic on the right side of the pivot
3. The 5th order statistic on the right side of the pivot
4. We might need to recurse on both the left and right sides of the pivot

(2) is correct. In the original array we have computed the $3rd$ order statistic. The $5th$ order statistic must be on the right side of the pivot but we are working with a smaller array where the $4th$ order statistic is the index after the pivot.

2 What is the running time of the *RSelect* algorithm if pivot elements are always chosen in the worst possible way?

1. $\Theta(n)$
2. $\Theta(n \cdot logn)$
3. $\Theta(n^2)$
4. $\Theta(2^n)$

(3) is the correct answer. In the worst case we recurse on a subproblem of length $n-1$. Hence we have a recurrence of the form $T(n) = T(n-1) + O(n) = O(n^2)$.

3 How many recursive calls does a single call to *DSelect* typically make?

1. 0
2. 1
3. 2
4. 3

(2) is the correct answer. If we don't get lucky and only do one call we typically have to do two to first reduce the problem then recurse again.

4 Let α be some constant independent of the input array length n, strictly between $\frac{1}{2}$ and 1. Suppose you are using the *RSelect* algorithm to compute the median element of a length-n array. What is the probability that the first recursive call is passed a subarray of length at most $\alpha \cdot n$?

1. $1 - \alpha$
2. $\alpha - \frac{1}{2}$
3. $1 - \frac{\alpha}{2}$
4. $2 \cdot \alpha - 1$

(4) is the correct answer.

$Pr((1-\alpha) \cdot N < x < \alpha \cdot N) = \frac{\alpha \cdot N - N + \alpha N}{N} = \frac{2\alpha \cdot N - N}{N} = \frac{2 \cdot \alpha - 1}{1} = 2 \cdot \alpha - 1$

5 Let α be some constant, independent of the input array length n, strictly between $\frac{1}{2}$ and 1. Assume that every recursive call to *RSelect* makes progress as in the preceding problem-so whenever a recursive call is given an array of length k, its recursive call is passed a subarry with length at most $\alpha \cdot k$. What is the maximum number of successive recursive calls that can occur before triggering the base case?

1. $-\frac{lnn}{ln\alpha}$
2. $-\frac{lnn}{\alpha}$
3. $-\frac{lnn}{ln(1-\alpha)}$
4. $-\frac{lnn}{ln(\frac{1}{2}+\alpha)}$

(1) is the correct answer

$$\begin{aligned} \alpha^d n &\leq 1 \\ d \cdot ln\alpha + lnn &= 0 \\ d &= -\frac{lnn}{ln\alpha} \end{aligned}$$

6 Suppose we modify the *DSelect* algorithm by breaking the elements into groups of 7, rather than group of 5. Does this modified algorithm asl run in $O(n)$ time? What if we use groups of 3?

It will still work for a group of 5 but not a group of 3.