Exercises 20 : randomized select

Questions

1. Suppose you choose a pivot randomly for a list of size n. What is the probability that the resulting two lists l1 and l2 will both have size at most 9 n /10 ?

2. In the proof of the linear time expected performance of select, the algorithm is in phase j at most how many times?

3. In the randomized select method for a given input, what is the sample space ?

4. What is the probability that a worst case pivot is chosen in every call for select ? Assume the initial list has n elements.

The next two questions were added April 6.

5. Using the randomized quicksort algorithm given in the lecture, we concluded on page 5 that the height of the quicksort call tree is log_(4/3) n. Why?

6. What is the length of the shortest path in the quicksort call tree ?

Answers

1. If the position of the pivot is i >= n/10, then l1 will have at least 1/10-th of the elements and l2 will have at most 9n/10 of the elements. If the position of the pivot is i <= 9n/10, then l1 will have at most 9/10-th of the elements and l2 will have at least 1/10 of the elements. Thus, if i >= n/10 and i <= 9n/10, then l1 and l2 will each have at most 9n/10 of the elements. This event is precisely the condition we are looking for and it has probability 8/10. [Updated April 5.]

2. The interval is [n(3/4)^j+1, n(3/4)^j) which has width n(3/4)^j - n (3/4)^j+1 = n(3/4)^j (1 – ¾) = n(3/4)^j /4. In the worst case, the randomly chosen pivot is the minimal or maximal element of the list and the size of the problem is only reduced by 1. So the answer is: n(3/4)^j /4.

3. The sample space is the sequence of positions of the pivots in the recursive calls. (This is not easy to express mathematically. Fortunately, the proof doesn't require us to do so.)

4. For a list of n elements, choosing the pivot to be either the first or the last element would be the worst case since it would reduce the problem size by 1 only (and the element we are selecting happens to not be the pivot each time). The probability of this event is 2/n for the

first call, 2/(n-1) for the second call, 2/(n-k+1) for the kth call. Multiplying all these probabilities together gives the answer. When the list is suffiently small (say n<=3) you would define a base case. [Updated April 6.]

5. The longest path in the quicksort call tree occurs when the list shrinks as slowly as possible from node to node along that path. Because we only recurse when a good partition is found, the size of the child problem can be at most ¾ as large as the size of the parent. The number of times you can multiply n by ¾ until you get a value equal to 1 is log_(4/3) n, since if (3/4)^x * n = 1 then n = (4/3)^x, and so log_(4/3) n = x. Here I am ignoring the case that x is not an integer.

6. The shortest path is log_4 n. It's a similar argument as the previous question. Because we only recurse when a good partition is found, the size of each of the two child problems of a node must be at least ¼ as large as the size of the parent. The number of times you can multiply n by ¼ until you get a value equal to 1 is log_4 n, since if (1/4)^x * n = 1 then n = 4^x, and so log_4 n = x.