Problem Set 6 Solutions

This problem set is due **at 11:59pm** on **Wednesday, April 5, 2017.**

EXERCISES (NOT TO BE TURNED IN)

Randomness

- Do Problem 6-2 on page 133: Analysis of max program
- Do problem 6-4 on page 133: Probailistic counting

Random Walks

- Make a directed graph with about 10 vertices. Figure out whether it has a unique stationary distribution.
- How can you efficiently calculate the random walk distribution after t steps? Can you do it faster than $O(t)$ for a constant sized graph?

Problem 6-1. Randomized Quicksort Analysis [50 points]

In this problem, we will try to understand the time complexity of RANDOMIZED QUICKSORT in more detail. In particular, instead of just looking at the expected running time of RANDOMIZED QUICKSORT, we also want to bound the probability of RANDOMIZED QUICKSORT not running fast, i.e. within some $O(n \log n)$ amount of time. Here, n is the length of the array that we run RANDOMIZED QUICKSORT on.

We will prove that RANDOMIZED QUICKSORT makes no more than $kn \log n$ comparisons with probability at least $1 - \frac{1}{n^\alpha}$, where k depends on α but not n. In other words, we will prove that RANDOMIZED QUICKSORT runs in $O(n \log n)$ with high probability.

Recall that RANDOMIZED QUICKSORT(A) works as follows:

- Choose an element $x \in A$ uniformly at random. x is the pivot.
- Partition A into two sets: L, all elements of A less than x, and G, all elements of G greater than x.
- Recursively sort L and G.
- Recombine in the order L, x, G.

Let's pick an element x_i, which is the element at position i in the input array A. We want to analyze the number of times this element is compared to another element when we run RANDOMIZED QUICKSORT on A.

Assume that all elements of A are distinct.

In solving this problem, it will be useful to use the following error probability bound:

Claim: Let $c > 1$ be a real constant, and let α be a positive integer. Then, with probability at least $1 - \frac{1}{n^\alpha}$, $3(\alpha + c) \lg n$ tosses of a fair coin produce at least $c \lg n$ heads. Any series of random events where some outcome occurs independently with probability $\frac{1}{2}$ each time obeys this bound.

A more general version of this bound is the Chernoff Bound, which will be covered in recitation just after this PSet is due.

(a) [10 points] Consider a particular element x_i. Consider a recursive call of RANDOMIZED QUICKSORT on a subarray A' of size $m \geq 2$ which includes element x_i. Prove that, with probability at least $\frac{1}{2}$, either this call to RANDOMIZED QUICKSORT chooses x_i as the pivot element, or the next recursive call to RANDOMIZED QUICKSORT containing x_i involves a subarray of size at most $\frac{3}{4}m$.

Pre-solution:

We're thinking about the size of the subarray that x_i is going to be placed in. Before we focus on one particular element, let's think about the size of the subarray that each of the of elements end up in. All the elements smaller than the pivot end up in one subarray, and all of the larger elements end up in the other subarray. So there's one

subarray of size k, and one of size $m - k - 1$, where k is the number of elements smaller than the pivot.

Now, we want to focus on the situation where a subarray is of size at most $3/4$. If $k \leq 3m/4$, the subarray of elements smaller than the pivot is at most that size. If $k \geq m/4$, the subarray of elements larger than the pivot is at most that size. So half the time, both subarrays are at most that size, and so x_i is either the pivot or in such an array.

Solution:

Suppose the pivot value is x, and has rank j in A'. Rank is the number of elements of the list which that element is greater than or equal to.

If $\lfloor \frac{m}{4} \rfloor + 1 \leq j \leq m - \lfloor \frac{m}{4} \rfloor$, then both subarrays produced by the partition have size at most $\frac{3m}{4}$. Moreover, the number of values of x in this range is at least $\frac{m}{2}$, so the probability of choosing such a value is at least $\frac{1}{2}$. Then both subarrays meet the specified bound on size, so either x_i is the pivot value or it is in one of the subarrays, and in both cases the requirement is satisfied.

(b) [18 points] Consider a particular element x_i. Prove that, with probability at least $1 - \frac{1}{n^2}$, the total number of times the algorithm compares x_i with pivots is at most $d \lg n$, for a particular constant d. Give a value for d explicitly.

Pre-solution:

This question is basically asking when the program is going to finish with a particular element. If we track a particular element through the execution of the program, it'll always be in some array, get compared with a pivot, moved into a smaller array, and so on and so on until it's done.

How many times can this happen before we're done? It can happen any number of times, if we get unlucky. So we need to look at probabilities. So let's look at the probability we're trying to prove. We want to prove something with a probability of at least $1 - 1/n^2$. Where have we seen something that looks like that recently? In the claim.

The claim says that we can guarantee that an event which happens half of the time will happen at least $c \lg n$, with a similar looking probability. Why would that be useful?

Well, one place where logs often come up is when something is changing in size by a multiplicative factor again and again. This applies to part (a), where the size of x_i's subarray is changing by a multiplicative factor, shrinking by $3/4$ at least half of the time. So our coin flips are going to be shrinkages of at least that size, and after a logarithmic number of successful shrinks, we'll be done with x_i.

Solution: We use part (a) and the Claim. By part (a), each time QUICKSORT is called for a subarray containing x_i, with probability at least $\frac{1}{2}$, either x_i is chosen as the pivot value or else the size of the subarray containing x_i reduces to at most $\frac{3}{4}$ of what it

was before the call. Let's say that a call is "successful" if either of these two cases happens. That is, with probability at least $\frac{1}{2}$, the call is successful.

Now, at most $\log_{4/3} n$ successful calls can occur for subarrays containing x_i during an execution, because after that many successful calls, the size of the subarray containing x_i would be reduced to 1. Using the change of base formula for logarithms, $\log_{4/3} n = c \lg n$, where $c = \log_{4/3} 2$.

Now we can model the sequence of calls to QUICKSORT for subarrays containing x_i as a sequence of tosses of a fair coin, where heads corresponds to successful calls. By the Claim, with $c = \log_{4/3} 2$ and $\alpha = 2$, we conclude that, with probability at least $1 - \frac{1}{n^2}$, we have at least $c \lg n$ successful calls within $d \lg n$ total calls, where $d = 3(2 + c)$. Each comparison of x_i with a pivot occurs as part of one of these calls, so with probability at least $1 - \frac{1}{n^2}$, the total number of times the algorithm compares x_i with pivots is at most $d \lg n = 3(2 + c) \lg n = 3(2 + \log_{4/3} 2) \lg n$. The required value of d is $3(2 + \log_{4/3} 2) \leq 14$.

(c) [12 points] Now consider all of the elements $x_1, x_2, \ldots, x_n$. Apply your result from part (b) to prove that, with probability at least $1 - \frac{1}{n}$, the total number of comparisons made by QUICKSORT on the given array input is at most $d' n \lg n$, for a particular constant d'. Give a value for d' explicitly.

Hint: The Union Bound may be useful for your analysis.

Pre-solution:

The union bound talks about situations where you want to bound the probability that any of a number of different things happen. We're probably going to be doing something related to every single element, since our previous results were only talking about one element.

What's something that we care about happening to any of the elements? Or to think of it the opposite way, what's something we care about happening to all of the elements?

Well, it'd be nice if every single element had no more than the number of comparisons from (b). Or to but it another way, we'd like to show that it's unlikely that any of the elements had more than that number of comparisons. We can apply the union bound to these probabilities.

Solution: Using a union bound for all the n elements of the original array A, we get that, with probability at least $1 - n(\frac{1}{n^2}) = 1 - \frac{1}{n}$, every value in the array is compared with pivots at most $d \lg n$ times, with d as in part (b). Therefore, with probability at least $1 - \frac{1}{n}$, the total number of such comparisons is at most $dn \lg n$. Using $d' = d$ works fine.

Since all the comparisons made during execution of QUICKSORT involve comparison of some element with a pivot, we get the same probabilistic bound for the total number of comparisons.

(d) [10 points] Generalize your results above to obtain a bound on the number of comparisons made by QUICKSORT that holds with probability $1-\frac{1}{n^\alpha}$, for any positive integer α, rather than just probability $1-\frac{1}{n}$ (i.e., $\alpha = 1$).

Your bound should be of the form $kn \log n$, where k depends on α but not n.

Pre-solution:

Our solution in (c) is essentially the same as this statement, except with $\alpha = 1$. We just need to increase that exponent. We had an event with probability $1 - 1/n^2$ in part (b), which became $1 - 1/n$ in (c). Therefore, to get $1 - 1/n^\alpha$ in part (d), we just need to start with $1 - 1/n^{\alpha+1}$ in part (b).

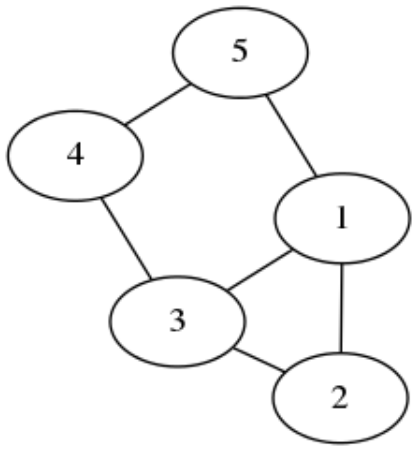
Solution: The modifications are easy. The Claim and part (a) are unchanged. For part (b), we now prove that with probability at least $1 - \frac{1}{n^{\alpha+1}}$, the total number of times the algorithm compares x_i with pivots is at most $d \lg n$, for $d = 3(\alpha + c)$. The argument is the same as before, but we use the Claim with the value of α instead of 2.

Then for part (c), we show that with probability at least $1 - \frac{1}{n^\alpha}$, the total number of times the algorithm compares any value with a pivot is at most $dn \lg n$, where $d = 3(\alpha + c)$.

Thus, $k = 3(\alpha + \log_{4/3} 2)$.

Problem 6-2. Random Walks and the Stationary Distribution [50 points]

In this problem, we will be working with the following 5 vertex graph, which we will call G:



(a) [6 points] Perform a 10-step random walk on G, starting at vertex 1. Write down the sequence of 11 vertices that you traverse.

Solution: An example solution: $[1, 5, 1, 2, 1, 2, 1, 2, 1, 2, 3]$

Any 10 step walk is fine.

All steps must be edges in the graph, and not from a vertex to itself.

(b) [6 points] What is the walk matrix for the graph G?

Recall that the walk matrix is the matrix W such that $Wp_t = p_{t+1}$, where p_t is the column vector of probabilities that a random walk on G is at a given vertex after t steps.

Solution:

$$\begin{bmatrix} 0 & 1/2 & 1/3 & 0 & 1/2 \\ 1/3 & 0 & 1/3 & 0 & 0 \\ 1/3 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 1/3 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$

(c) [6 points] What is the stationary distribution of the graph G?

Pre-solution:

$\pi_i = \frac{deg(v_i)}{\sum_j deg(v_j)}$

Solution:

$$\begin{bmatrix} 1/4 \\ 1/6 \\ 1/4 \\ 1/6 \\ 1/6 \end{bmatrix}$$

(d) [12 points] Let p_t be the vector of probabilities that a random walk is at each vertex after t steps. Let p_0, the initial distribution, be 1 for vertex 1 and 0 for the rest:

$$p_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

What is the minimal value of t such that each entry of p_t is within an additive error of 0.01 of the corresponding entry of the stationary distribution from part (c)?

Note: If you want to compute matrix powers, but you don't want to write a program, you can use this website: `http://comnuan.com/cmnn01012/`

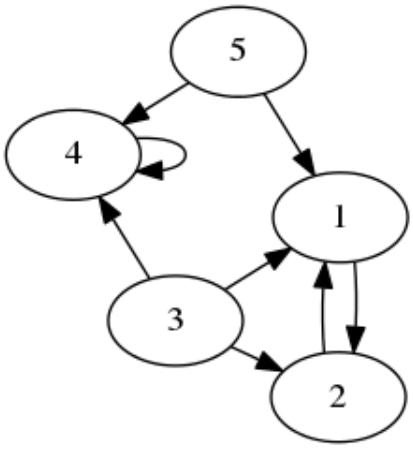
Pre-solution:

$p_t = W^t p_0$. It turns out that p_t will converge smoothly towards the stationary distribution, so we can use binary search to speed up the process of finding the correct value of t, if you're doing it one p_t at a time. Or you can write a program, or do it the slow way.

Solution: $t = 17$.

$$p_{17} = \begin{bmatrix} 0.2410 \\ 0.1667 \\ 0.2590 \\ 0.1577 \\ 0.1757 \end{bmatrix}$$

(e) [8 points] Now, we will introduce a new graph, G'. G' is very similar to G, except that now, every edge is directed.



Compute a stationary distribution of G'.

Pre-solution:

A really simple stationary distribution is starting at 4, because 4 simply points to itself.

Solution:

$$\begin{bmatrix} 1/2 \\ 1/2 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

are two of the possible stationary distributions. Any linear combination such that the entries are positive and sum to one is also stationary.

(f) [12 points] If we start a random walk with an arbitrary initial distribution over starting vertices, will the random walk distribution after t steps, p'_t, converge to a unique stationary distribution as $t \to \infty$? Why or why not?

Pre-solution:

It's not strongly connected, so we know the answer is no. We also know there exists as stationary distribution, from (e). So all we need to do is find something that acts differently.

For instance, if we said the distribution at 4 last time, we could look at the part of the graph that can't get to 4, such as starting at 2, where the walk oscilates between 1 and 2.

Solution: It won't.

There are two problems: There are multiple stationary distributions, and if the initial distribution is stationary, it won't converge to anything else.

In addition, the 2-cycle between 1 and 2 can lead to periodic behavior that will never converge to any distribution. For instance, the initial condition of starting at vertex 5 leads to the random walk distribution alternating between $[1/2, 0, 0, 1/2, 0]$ for t odd and $[0, 1/2, 0, 1/2, 0]$ for t even.