Homework 1, CS561, Fall 2014

Aaron Gonzales

September 4, 2014

1 exercise 3.1-5 CLRS - Prove theorem 3.1

For any two functions $f(n)$ and $g(n)$, we have $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

By definition, $\Omega(g(n)) = \{f(n) :$ there exist positive constants c, n_0 such that $0 \leq cg(n) \leq f(n) \forall n \geq n_0\}$.

By definition, $O(g(n)) = \{f(n) :$ there exist positive constants c, n_0 such that $0 \leq f(n) \leq cg(n) \forall n \geq n_0\}$.

Let f and g be functions, such that $f(n) = \Theta(g(n))$.

By definition of Θ, there exist positive constants $k1$ and $k2$ such that, for sufficiently large n:

$$k_1 * g(n) \leq f(n) \leq k_2 * g(n)$$

Thus, for sufficiently large n:

$$f(n) \leq k_2 * g(n)$$

Therefore $f(n) = O(g(n))$ by definition.

And, for sufficiently large n:

$$k_1 * g(n) \leq f(n)$$

Therefore $f(n) = \Omega(g(n))$ by definition.

2 Let $f(n) and g(n)$ be two functions that take on nonnegative values and assume $f(n) = O(g(n))$. Prove that $g(n) = \Omega(f(n))$.

By definition, $\Omega(g(n)) = \{f(n) :$ there exist positive constants c, n_0 such that $0 \leq cg(n) \leq f(n) \forall n \geq n_0\}$.

By definition, $O(g(n)) = \{f(n) :$ there exist positive constants c, n_0 such that $0 \leq f(n) \leq cg(n) \forall n \geq n_0\}$.

if $f(n)$ is bounded above by $g(n)$, $g(n)$ must be at least equal to or lower than $f(n)$ for all $n \geq n_0$.

3 Problem 3-2 on Relative asymptotic growths

A	B	O	o	Ω	ω	Θ
$lg^k n$	n^ϵ	F	F	T	T	F
n^k	c^n	T	T	F	F	F
$\sqrt{n}$	n^{sinn}	F	F	F	F	F
2^n	$2^{n/2}$	T	F	T	F	T
n^{lgc}	c^{lgn}	T	F	T	F	T
$lg(n!)$	$lg(n^n)$	T	F	T	F	T

4 True or false

Assume you have functions f and g, such that $f(n)$ is $O(g(n))$. for each of the following statements, decide wether you think it is true or false and give either a proof or a counterexample.

(a) $log_2 f(n) = O(log_2(g(n))$

False only in the following case: $f(2)$, $g(1)$.

$$f(2) = log_2(2) = 1$$

$$g(1) = log_2(1) = 0$$

For large enough n, this is true, as this would become $O(log_2(n)) = O(log_2(2))$.

(b) $2^{f(n)} = O(2^{g(n)})$

True. No counterexample can be found that would show $f(n) > g(n)$.

(c) $f(n)^2 = O(g(n)^2)$

For all n we can have $f(n)^2 = g(n)^2$.

True.

O is a loose upper bound. $0 \leq f(n) \leq g(n)$.

5 Problem 7-3: Alternative Quicksort analysis

(a) Argue that, given an array of size n, the probability that any particular element is chosen as the pivot is $1/n$. Use this to define indicator random variables $X_i = I\{i$ th smallest element is chosen as the pivot$\}$. What is $E[X_i]$?

$Pr(X = X_i) = 1/n$

Defining indicator variable $X_i \begin{bmatrix} 1{:}if\,pivot \\ 0{:}not\,pivot \end{bmatrix}$

$$\sum_{1}^{n} E(X_1) = X_1 Pr(x = X_1) + X_2 Pr(x = X_2) + \cdots + X_n Pr(x = X_n)$$

(b)

Let $T(n)$ be a random variable denoting the running time of quicksort on an array of size n. Argue that

$$E[T(n)] = E\left[\sum_{q=1}^{n} X_q(T(q-1)) + T(n-q) + \Theta(n))\right]$$

We would say that $T(q-1)$ and $T(n-q)$ are the work being done on the subarrays in quicksort. Probabilistically, we expect them to be roughly equal over time. So we would expect that the overall running time of quicksort could be the sum of the expected running times of the work done on the subarrays.

(c)

Show that we can rewrite equation 7.5 as:

$$E[T(n)] = \frac{2}{n}\sum_{q=2}^{n-1} E[T(q)] + \Theta(n)$$

.

Here is the start of the equation:

$$E[T(n)] = E\left[\sum_{q=1}^{n} X_q(T(q-1)) + T(n-q) + \Theta(n))\right]$$

by LOE, we get

$$E[T(n)] = X_q \sum_{q=1}^{n} E\left[(T(q-1)) + T(n-q) + \Theta(n))\right] \tag{1}$$

$$\tag{2}$$

$$X_q \sum_{q=1}^{n} E\left[(T(q-1)) + T(n-q)\right] + \left[\sum_{q=1}^{n} \Theta(n)\right]$$

We know X_i, so :

$$\frac{1}{n} \sum_{q=1}^{n} E\left[(T(q-1)) + T(n-q)\right] + \left[\sum_{q=1}^{n} \Theta(n)\right]$$

If we expand just the summation a bit, (and drop out Θ for a moment) we can see:
For $q = 1, q = 2, q = n-1, q = n$:

$$(T(0) + T(n-1)) + (T(1) + T(n-2)) + \ldots + (T(n-2) + T(1)) + (T(0) + (n-1))$$

each term repeats twice, so we can see a factor of 2 can be pulled out and when $q = 0$ or $q = 1$, these are 'base cases' and can be ignored in the summation, changing our q -> n. (and let's bring back Θ.)

$$\frac{2}{n} \sum_{q=2}^{n-1} E\left[(T(q)\right] + \Theta(n)$$

(d)

show that

$$\sum_{k=2}^{n-1} k\,lg\,k \leq \frac{1}{2} n^2\,lg\,n - \frac{1}{8} n^2$$

.

$$\sum_{k=2}^{n/2} k\,lg\,k + \sum_{k=n/2}^{n-1} k\,lg\,k$$

(e)

Using the bound from equation 7.7, show that the recurrance in equation 7.6 has the solution $E[T(n)] = \Theta(nlgn)$. Comparison-based sorting is $\Omega(nlgn)$, and for a function to be $\Theta(nlgn)$ it must be both $\Omega(nlgn)$ and $\Omega(nlgn)$, which this is.

6 Ladders

Imagine you are doing a stress test on a particular model of smart phones. you have a ladder with n rungs. You want ot determine the highest rung from which you cand rop a phone wihtout it breaking and you want to do it with the smallest number of phone drops.

(a) Imagine that you have exactly 2 phones. Devise an algorithm that can determine the highest safe rung using $o(n)$ drops. (little o).

Let it be stated that a ladder must have at least one one rung. If it has one rung, we only need one phone to test and see if that rung is safe (if it breaks, it's unsafe, if not, it's safe).

If a ladder has $n > 1$ rungs, we start by dropping the phone from the middle ladder rung ($\frac{n}{2}$). if the phone breaks, we search the bottom half of the ladder iteratively (first rung, second rung, etc.). If it doesn't break, we iteratively search the upper portion of the ladder.

We only test at most $\frac{n}{2}$ rungs, which $\frac{n}{2} < n \forall n \geq 2$. By definition, $o(g(n)) = \{f(n)$: for any positive constant $c > 0$, there exisists a constant $n_0 > 0$ such that $0 \leq f(n) \leq cg(n) \,\forall\, n \,\geq n_0\}$. as such, the algorithm is $o(n)$.

(b) Now suppose you have k phones. Devise an algorithm that can determine the highest safe rung with the smallest number of drops. If $f_k(n)$ is the number of drops that your algorithm needs, what is $f_k(n)$ asymptotically? Hint: you should ensure that $f_{k+1}(n) = o(f_k(n))$ for any k.

This is an example of binary search. If we assume that the rungs in the ladder are in a sorted order (and how could they not be? did my adversary misnumber my rungs?) the algorithm could be stated as follows:

As before, start with the middle rung of the ladder ($\frac{n}{2}$). (If the ladder has an even number of rungs, choose the $\lfloor \frac{n}{2} \rfloor$ rung). If the phone dropped breaks, do the same procedure on the bottom half of the ladder. If it doesn't break, do the same procedure on the upper half of the ladder. each recursive step reduces the size of our search space by $\frac{1}{2}$. If for some reason we get to $k = 1$, then iteratively search the current subarray and return the rung prior to the rung on which the phone broke.

we can state this recurrence relation as:

$$T(n) = T(n/2) + C$$

which is $\Theta(log\, n)$. Since our binary search is $\Theta(log\, n)$, we get our work being done by the algorithm as $o(n)$.

7 The game of Match.

The game of Match is played with a special deck of 27 cards. Each card has three attributes: color, shape and number. The possible color values are {red, blue, green}, the possible shape values are {square, circle, heart}, and the possible number values are {1, 2, 3}. Each of the 333 = 27 possible combinations is represented by a card in the deck. A match is a set of 3 cards with the property that for every one of the three attributes, either all the cards have the same value for that attribute or they all have different values for that attribute. For example, the following three cards are a match: (3, red, square), (2, blue, square), (1, green, square).

(a) If we shuffle the deck and turn over three cards, what is the probability that they form a match? Hint: given the first two cards, what is the probability that the third forms a match?

We know that given any two cards, only one more card can make a match. If we draw two cards, only one more card in the deck can make a match for the prior two cards. After drawing two cards, we only have 25 cards left from which we can draw, so we get $Pr(match) = \frac{1}{25}$.

(b) If we shuffle the deck and turn over n cards where $n \leq 27$, what is the expected number of matches, where we count each match separately even if they overlap? Note: The cards in a match do not need to be adjacent! Is your expression correct for $n = 27$?

I believe we can have $\binom{n}{3} * \frac{1}{25}$ possible matches, which comes to

$$\begin{aligned} TotalMatches &= \binom{n}{3} * \frac{1}{25} \\ &= \binom{27}{3} * \frac{1}{25} \\ &= \frac{27!}{3!(27-3)!} * \frac{1}{25} \\ &= \frac{27 * 26 * 25}{6} * \frac{1}{25} \\ &= 117 \end{aligned}$$

8 Drunken Debutants

Drunken Debutants: Imagine that there are n debutants, each with her own porsche. After a late and wild party, each debutante stumbles into a porsche selected independently and uniformly at random (thus, more than one debutant may wind up in a porsche). Let X be a random variable giving the number of debutants that wind up in their own porsche. Use linearity of expectation to compute the expected value of X. Now use Markovs inequality, to bound the probability that X is larger than k for any positive k.

We have n debutants and n porsches. $\frac{1}{n}$ is our probability of a debutant getting into her own porche. Let X_i be an indicator variable saying

$$X_i = I[X] \begin{cases} 1 & \text{if debutante gets in her own car} \\ 0 & \text{if not} \end{cases}$$

The number of debutants who return to their own can be expressed as:

$$X_i = \sum_{i=1}^{n} X_i$$

and the expected number of debutants can be expressed as:

$$E[X_i] = E\left[\sum_{i=1}^{n} X_i\right]$$

By linearity of expectation we can state it as such:

$$\begin{aligned} E[X_i] &= E\left[\sum_{i=1}^{n} X_i\right] \\ E[X_i] &= \sum_{i=1}^{n} E[X_i] \\ &= \sum_{i=1}^{n} \frac{1}{n} \\ &= \frac{1}{n}\sum_{i=1}^{n} 1 \\ &= \frac{n}{n} \\ &= 1 \end{aligned}$$

We only expect one drunken debutant to get in her own porche. (Note - This presumes that all of the debutants made it out of the party and no one passed out inside.)

Markov's theorem states:

$$\mathbf{Pr}\left[X \geq k\right] \leq \frac{\mathbf{E}[X]}{k}$$

$$\mathbf{Pr}\left[1 \geq k\right] \leq \frac{1/n}{k}$$

$$\mathbf{Pr}\left[1 \geq k\right] \leq \frac{1/n}{k}$$

9 Pairs on a circle

Imagine n points are distributed uniformly at random on the perimeter of a circle that has circumference 1. Show that the expected number of pairs of points that are within distance $\Theta(1/n^2)$ of each other is greater than 1. FYI: this problem has applications in efficient routing in peer-to-peer networks. Hint: Partition the circle into n^2/k regions of size k/n^2 for some constant k; then use the Birthday paradox to solve for the necessary k.

Let $X_{i,j}$ be an indicator variable where:

$$X_{i,j} = I[X] \begin{cases} 1 & \text{if a pair of points (i,j) is within} \Theta(\frac{1}{n^2}) \\ 0 & \text{if not} \end{cases}$$

If the generic birthday paradox can have an expected collision of (p = people, b = birthdays):

$$\frac{p(p-1)}{2b}$$

if we have n^2/k regions

$$\frac{p(p-1)}{2n^2/k}$$