Homework 2, CS561, Fall 2014

Aaron Gonzales

September 18, 2014

1 Skip lists

In this problem you will use Chernoff bounds to show that for most of the levels of a skip list, the size of the level is very tightly bounded around its xpectation.
Chernoff Bounds: Assume you have n independent, indicator random variables X_1, X_2, ..., X_n and let $X = \sum_{i=0}^{n} X_i$ and $\mu = E(X)$. Then Chernoff bounds tell us that for any $0 \leq \delta \leq 1$:

$$Pr(X \leq (1-\delta)\mu) \text{ or } X \geq (1+\delta)\mu) \leq 2e^{\mu \frac{\delta^2}{4}}$$

Use Chernoff and Union bounds to show that with probability at least $1\frac{1}{n}$, for all $0 \leq j \leq logn - loglog\, n - 5$, List j in a skip list contains between $\frac{n}{2^{j+1}}$ and $\frac{3n}{2^{j+1}}$ nodes. Let List 0 be the bottom list, List 1 be the next higher up, etc. You may assume that n is sufficiently large, e.g. n is larger than some constantn_0.
Note: Chernoff bounds are more powerful than bounds on Binomial distributions since X need not be binomially distributed. The only requirement is that the X_i be independent. Hint: Remember that $e^x \leq 2^x$.

Our first steps are to find both δ and μ.
We know that the probability of a node landing on a layer j in the skip list is $Pr(\frac{1}{2^j})$.

$$\mu = \sum_{i=n}^{n} \frac{1}{2^j} = \frac{n}{2^j}$$

δ we need to find an upper and lower bound:
Upper:

$$\begin{aligned}(1-\delta)\mu &= \frac{n}{2^{j+1}} \\ (1-\delta)\frac{n}{2^j} &= \frac{n}{2^j+1} \\ 1-\delta &= \frac{1}{2} \\ \delta &= \frac{1}{2}\end{aligned}$$

lower :

$$\begin{aligned}(1+\delta)\mu &= \frac{3n}{2^{j+1}} \\ (1+\delta)\frac{n}{2^j} &= \frac{3n}{2^j+1} \\ 1+\delta &= \frac{3}{2} \\ \delta &= \frac{3}{2}\end{aligned}$$

plugging these into our Chernoff bounds:

$$Pr(X \leq (1-\delta)\mu) \text{ or } X \geq (1+\delta)\mu) \leq 2e^{\mu \frac{\delta^2}{4}}$$

$$Pr\left(X \leq \frac{n}{2^{j+1}} \text{ or } X \geq \frac{3n}{2^j+1}\right) \leq 2e^{\mu \frac{\delta^2}{4}}$$

$$Pr\left(X \leq \frac{n}{2^{j+1}} \text{ or } X \geq \frac{3n}{2^j+1}\right) \leq 2e^{1-\frac{n}{16*2^j}}$$

2 pebbles and graphs

You toss m pebbles onto the n nodes of a k-regular undirected graph (recall that a graph is k-regular if every node has degree k). Each pebble lands on a node selected uniformly at random. A pair of pebbles is said to "collide" if they fall on the same node or on two nodes that are neighbors. What is the expected number of pairs of pebbles that collide? About how large must m be before you would expect at least 1 pair of pebbles to collide?

We can use the Birthday Paradox to help us out here. Knowing that the BP says:

$$E(X) = \frac{m(m-1)}{2n}$$

where $m =$ people or pebbles and $n =$ number of days or nodes and $E(X)$ is the expected number of pairs of people or pebbles that have the same birthday or node. In our problem let us state:

Let Xi_j be an indicator variable saying

$$X_{i,j} = I[X] \begin{cases} 1 & \text{if pebbles collide (same or adjacent node)} \\ 0 & \text{if not} \end{cases}$$

and for all pairs of pebbles i, j the probability of a collision is $\frac{K+1}{n}$.

$$\sum_{i=1}^{m} \sum_{j+1}^{m} X_{i,j}$$

$$\binom{m}{2} \frac{k+1}{n} = \frac{m(m-1)(k+1)}{2(n)}$$

3 Recurrence

Consider the Recurrence $f(n) = 3f(\frac{n}{2}) + \sqrt{n}$

(a) Use the Master method to solve this recurrence

The generic form of the Master equation is:

$$T(n) = aT\left(\frac{n}{b} + f(n)\right) \text{ where } a \geq 1, b > 1$$

where: $T(n)$ has the following asymptotic bounds:

1. if $f(n) = O\left(n^{log_b a - \epsilon}\right)$ for some constast $\epsilon > 0$, then $T(n) = \Theta(n^{log_b a})$
2. if $f(n) = \Theta(n^{log_b a}), \text{then} T(n) = \Theta\left(n^{log_b a} lg\, n\right)$.
3. if $f(n) = \Omega\left(n^{log_b a + \epsilon}\right)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.

in our example, we have $a = 3$, $b = 2$, and $f(n) = \sqrt{n}$.

$$n^{log_b a} = n^{log_2 3} = O(n^{1.58})$$

and $\sqrt{n} = n^{1/2} = O(n^{1.58})$,so $T(n) = \Theta(n^{1.58})$.

(b) Now use annihilators (and a transformation) to solve the recurrence. Show your work. (This is perhaps stating the obvious, but please note that your two bounds should match)

Our recurrence is: $f(n) = 3f(\frac{n}{2}) + \sqrt{n}$.

and needs a transformation. Let's let $n = 2^i$ and rewrite $T(n)$ as:

$$T(2^i) = 3T(\frac{2^i}{2}) + 2^{i/2}$$

$$T(2^i) = 3T(2^{i-1}) + 2^{i/2}$$

let $t(2^i) = t(i)$

$$t(i) = 3T(i-1) + \sqrt{2^i}$$

Using annhilators, we can eliminate the sequence $t = \langle t_i \rangle$, eliminating the homogenous parts seperately from the recursive parts.

$$(L-3)(L-\sqrt{2})$$

From the lookup table $(L-3)(L-\sqrt{2})$ will annihilate relations with the form of $c_0 * 3^i + c_1^i$

back transformation gives us:

$$lgn = i$$

$$3^{lgn} + \sqrt{(2)^{lgn}}$$

$$n^{lg3} + \sqrt{n}$$

$$= O(n^{lg3})$$

where $lg3 = lob_2 3$.

4 More recurrence

consider the following function:

```
1  int f (int n){
     if (n == 0) return 2;
3    else if (n&=& 1) return 5;
     else{
5      int val = 2*f (n-1);
       val = val - f (n-2);
7      return val;
     }
9  }
```

(a) Write a recurrence relation for the value returned by f. Solve the recurrence exactly. (Don't forget to check it.)

We can state this algorithm as follows:

$$f(n) = 2f(n-1) - f(n-2)$$

with a base case of $f(0) = 2$. For the following values of n, we get:

$$f(0) = 2 \tag{1}$$
$$f(1) = 5 \tag{2}$$
$$f(2) = 8 \tag{3}$$
$$f(3) = 11 \tag{4}$$
$$f(4) = 14 \tag{5}$$

Formally stating our recurrence as for all values of $n > 0$, i assume it returns $3n + 2$.

Proof. By induction: for

$$f(n) = 2f(n-1) - f(n-2), f(0) = 2,\ f(n) = 3n + 2$$

Base case: $f(0) = 2$. Inductive Hypothesis: $\forall j < n, f(j) = 3j + 2$
Inductive step:

$$\begin{aligned} f(n) &= 2f(n-1) - f(n-2) \\ &= 2(3n + 2 - 1) - 3n + 2 - 2 \text{ by I.H.} \\ &= 6n + 2 - 3n \\ &= 3n + 2 \end{aligned}$$

□

(b) (b) Write a recurrence relation for the running time of f. Get a tight upperbound (i.e. big-O) on the solution to this recurrence.

$$f(n) = 2f(n-1) - f(n-2) + \Theta(1)$$

We can see that this is similar to Fibonacci and use a few annihilators to solve this.For the homogenous parts:

$$T = \langle T_0, T_1, T_2, T_3, \ldots \rangle$$
$$LT = \langle T_1, T_2, T_3, T_4, \ldots \rangle$$
$$L^2T = \langle T_2, T_3, T_4, T_5, \ldots \rangle$$

$$(L^2 - L - 1)$$

In class, we learned that this factors to

$$(L - \phi)(L - \hat{\phi})$$

And our sequence must have the form

$$O(\phi^n + \hat{\phi}^n)$$

and reduces to

$$O(\phi^n)$$

5 Silly-Sort

```
1  Silly-Sort(A,i,j):
     if A[i] > A[j]:
3      then exchange A[i] and A[j]
     if i+1 >= j:
5      then return
     k = floor(j-i+1)/3)
7    Silly-Sort(A,i,j-k)
     Silly-Sort(A,i+k,j)
9    Silly-Sort(A,i,j-k)
```

(a) Argue (by induction) that if n is the length of A, then Silly- Sort(A,1,n) correctly sorts the input array $A[1 \dots n]$.

We can see that Silly-Sort is sorting 2/3 of the list on each recursive call. If we define a base case of a list where $n = 3$, and the list is $\{2, 3, 1\}$ the array will be sorted as such:

$$\{2, 3, 1\}$$
$$\{1, 3, 2\}$$
$$\{1, 2, 3\}$$

Inductive Hypothesis: Silly sort sorts any array length $j < n$, and knowing that Silly-Sort will always eventually divide the list into length 3, by the base case we can see that if the sublists of length three are sorted, the full list would be sorted as well.

(b) Give a recurrence relation for the worst-case run time of Silly-Sort and a tight bound on the worst-case run time

We can define a recurrence relation for the algorithm as

$$T(n) = 3T(\frac{2n}{3}) + \Theta(1)$$

and solve it by the master method:
$a = 3$, $b = 2/3$, $c = \Theta$ which gives us case
if $f(n) = O\left(n^{log_b a - \epsilon}\right)$ for some constast $\epsilon > 0$, then $T(n) = \Theta(n^{log_b a})$
in our example, we have $a = 3$, $b = 3/2$, and $f(n) = \sqrt{n}$.

$$n^{log_b a} = n^{log_{\frac{3}{2}} 3} = O(n^{2.7})$$

(c) Compare this worst-case runtime with that of insertion sort,merge sort, heapsort and quick-sort.

- Insertion sort: $O(n^2)$
- Merge sort: $O(nlgn)$
- heapsort: $O(nlgn)$
- quicksort: $O(n^2)$ (Worst case, average is $O(nlgn)$
- Silly Sort:

6 Primes and Probability

In this problem, you will use the following facts. 1) any integer can be uniquely factored into primes; 2) the number of primes less than any number m is $\Theta(m/logm)$ (this is the prime number theorem).
We will also make use of the following notation for integers x and y:
1) $x|y$ **means that x "divides" y, which means that there is no remainder when you divide y by x.**
and 2) $x = y(modp)$ **means that x and y have the same remainder when divided by p, or in other words,** $p|xy$

(a) Show that for any integer x, x factors into at most log x primes. Hint: 2 is the smallest prime.

If we factor a number x using a binary tree, we can see that the greatest number of prime factors will be less than or equal to the height of a binary tree, which is lgn in height. An example:

```
    8              15              16
  4   2          3  5          8      2
 2 2                          4      2
                            2  2
```

(b) Let x be some positive integer and let p be a prime chosen uniformly at random from all primes less than or equal to m. Use the prime number theorem to show that the probability that $p|x$ is $O\,(logx)(logm)/m)$.

Let X_i be an indicator variable saying

$$X_i = I[X] \begin{cases} 1 & \text{if prime factor } i \text{ of } x = p \\ 0 & \text{if not} \end{cases}$$

Our random variable is

$$X = Ex\left(\sum_{i=1}^{logx}(X_i)\right)$$

$$X = \sum_{i=1}^{logx} Ex(X_i)$$

The probability of $X_{i,p} \leq \frac{\frac{1}{m}}{\Theta logm} = \Theta\frac{logm}{m}$
and as such we get:

$$X = \sum_{i=1}^{logx} \frac{logm}{m}$$

$$= \frac{logx \times logm}{m}$$

(c) Now let x and y both be positive integers less than n and let p be a prime chosen uniformly at random from all primes less than or equal to m. Using the previous result, show that the probability that $xy(modp)$ is $O\left(logn\right)(logm)/m))$.

(d) If $m = log^2 n$ in the previous problem, then what is the probability that $xy(modp)$. Hint: If you're on the right track, you should be able to show that this probability is "small", i.e. it goes to 0 as n gets large.

(e) Finally, show how to apply this result to the following problem. Alice and Bob both have databases x and y where x and y have value no more than n, for n a very large number (think terabytes). They want to check to see if their databases are consistent (i.e. they want to check if they are the same) but Alice does not want to have to send her entire database to Bob. What is an algorithm Alice and Bob can use to check consistency with reasonably good probability by sending a lot fewer bits? How many bits does Alice need to send to Bob as a function of n, and what is the probability of failure, where failure means that this algorithm says the databases are the same but in fact they are different?

7 Bad Santa

A child is presented with n boxes, one after another. Upon receiving a box, the child must decide whether or not to open it. If the child does not open a box, he is never allowed to revisit it. Half the boxes have presents in them, but the decision about which boxes have presents is made by an omniscient and malicious Santa who wants the child to open as many empty boxes as possible before finding a present.
Devise and analyze a randomized algorithm for the child which minimizes the expected number of boxes that need to be opened before the child finds the first present. Assume Santa knows your algorithm, but can not predict the random choices made by your algorithm.
Hint: Birthday paradox.
Note: This problem has applications to wireless networks: basically boxes are time-steps, Santa is a jamming adversary, and opening a box means spending energy to listen in a time-step.

Our problem has the following setup:

- n boxes
- $\frac{n}{2}$ boxes have presents
- Birthday paradox tells us that we need $\sqrt{n}$ pebbles to fit expect a "collision" in n bins.

If we naively pick boxes in order, santa will obviously put the boxes in the last half of our 'box array'. if this is true, we will have to pick $\frac{n}{2} + 1 = O(n)$ boxes to get a gift.

Randomization helps us greatly here as Santa cannot account for our random choices, though his placements will probably not be random. If we divide the array in half we can find a lower bound than $O(n)$ for our number of opened boxes. As the birthday paradox was discussed in class, we can expect that given n bins and p pebbles tossed randomly into bins, $\sqrt{n}$ tosses gives us an expectation of 1 "collision" or two pebbles going into the same bin. If we think of our pick of a box as a pebble and santa's placement of a box as a pebble, a "collision" is us picking a box with a gift.

If we split the array in half and choose $\sqrt{n} + \epsilon$ (where ϵ is some arbitrary constant) boxes from the first subarray, we would exect a collision if there were at a handful of gift s in the first array. If we do not get a gift, we know that there are likely to be more gifts in the second have of the array, and as such, we can start choosing linearly from that array. Even if santa loads the back half of the array with gifts, we are still going to be picking $O(\sqrt{n+\epsilon} + C) = O(\sqrt{n+\epsilon})$.