CS 561, HW2

Prof. Jared Saia, University of New Mexico

Due: Sept. 20

1. Consider the following set of dice:

   - Die A has sides 2, 2, 4, 4, 9, 9
   - Die B has sides 1, 1, 6, 6, 8, 8
   - Die C has sides 3, 3, 5, 5, 7, 7

   Let A,B,C be random variables giving the outcome of dies A, B, C. What is $E(A)$, $E(B)$, $E(C)$? What is $Pr(A > B)$? $Pr(B > C)$? $Pr(C > A)$?

2. In a k-coloring of a graph, you must assign one of k different colors to each node in the graph. An edge (u, v) in the graph is said to be *satisfied* if u and v have been assigned different colors. In the problems below, assume your graph has n nodes and m edges.

   (a) Consider the following randomized algorithm for 3 coloring a graph: assign each node a color selected independently and uniformly at random. What is the expected number of edges satisfied by this algorithm?

   (b) Give a lower bound on the probability that the above algorithm satisfies at least $m/2$ edges. Hint: Let Y be a random variable that is equal to the number of edges that are not satisfied.

   (c) Now assume that each node is assigned a color independently and uniformly at random from a set of cm colors for some constant c. What is an upper bound on the probability that some edge is not satisfied? Hint let ξ be the bad event that some edge is not satisfied.

3. Consider the recurrence $f(n) = 3f(n/2) + \sqrt{n}$

   (a) Use the Master method to solve this recurrence

(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)

4. Consider the following function:

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

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

(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.

5. *Silly-Sort* Consider the following sorting algorithm

```
Silly-Sort(A,i,j)
  if A[i] > A[j]
    then exchange A[i] and A[j];
  if i+1 >= j
    then return;
  k = floor(j-i+1)/3);
  Silly-Sort(A,i,j-k);
  Silly-Sort(A,i+k,j);
  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...n]

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

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

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 $\theta(m/\log m)$ (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) $x \equiv y \pmod{p}$ means that x and y have the same remainder when divided by p, or in other words, $p|(x-y)$.

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

   (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 $O((\log x)(\log m)/m)$.

   (c) Now let x and y both be positive integers less than n, such that $x \neq y$, 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 $x \equiv y \pmod{p}$ is $O((\log n)(\log m)/m))$.

   (d) If $m = \log^2 n$ in the previous problem, then what is the probability that $x \equiv y \pmod{p}$. 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 (i.e. Bad) 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 asymptotically optimizes 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. Your algorithm must always eventually find a present.

Hint: Birthday paradox.

Note: this problem has applications to wireless networks. Boxes are time-steps, "Santa" is a jamming adversary, and opening a box means spending energy to listen in a time-step.