Chris Piech
CS 109

Section #7

Section 7 Solution

1. **Binary Tree**: Consider the following function for constructing binary trees:

```
def random_binary_tree(p):
    """
    Returns a dictionary representing a random binary tree structure.
    The dictionary can have two keys, "left" and "right".
    """
    if random_bernoulli(p):   # returns true with probability p
        new_node = {}
        new_node["left"] = random_binary_tree(p)
        new_node["right"] = random_binary_tree(p)
        return random_binary_tree
     else:
        return None
```

   The `if` branch is taken with probability p (and the `else` branch with probability $1 - p$). A tree with no nodes is represented by `nullptr`; so a tree node with no left child has `nullptr` for the `left` field (and the same for the right child).

   Let X be the number of nodes in a tree returned by `randomTree`. You can assume $0 < p < 0.5$. What is $E[X]$, in terms of p?

2. **Timing Attack**:

   In this problem we are going to show you how to crack a password in linear time, by measuring how long the password check takes to execute (see code below). Assume that our server takes T ms to execute any line in the code where $T \sim N(\mu = 5, \sigma^2 = 0.5)$ seconds. The amount of time taken to execute a line is always independent of other values of T.

```
# An insecure string comparison
def string_equals(guess, password):
    n_guess = len(guess)
    n_password = len(password)
    if n_guess != n_password:
        return False                  # 4 lines executed to get here
    for i in range(n_guess):
        if guess[i] != password[i]:
            return False              # 6 + 2i lines executed to get here
    return True                       # 5 + 2n lines executed to get here
```

   On our site all passwords are length 5 through 10 (inclusive) and are composed of lower case letters only. A hacker is trying to crack the root password which is “gobayes" by carefully measuring how long we take to tell them that her guesses are incorrect.

   a. What is the distribution of time that it takes our server to execute k lines of code? Recall that each line independently takes $T \sim N(\mu = 5, \sigma^2 = 0.5)$ ms.

b. First the hacker needs to find out the length of the password. What is the probability that the time taken to test a guess of correct length (server executes 6 lines) is longer than the time taken to test a guess of an incorrect length (server executes 4 lines)? Assume that the first letter of the guess does not match the first letter of the password. Hint: $P(A > B)$ is the same as $P(A - B > 0)$.

c. Now that our hacker knows the length of the password, to get the actual string she is going to try and figure out each letter one at a time, starting with the first letter. The hacker tries the string “aaaaaaa" and it takes 27s. Based on this timing, how much more probable is it that first character did not match (server executes 6 lines) than the first character did match (server executes 8 lines)? Assume that all letters in the alphabet are equally likely to be the first letter.

d. If it takes the hacker 6 guesses to find the length of the password, and 26 guesses per letter to crack the password string, how many attempts does she need to crack our password, “gobayes"? Yikes!