

# Homework 2

Due: Thursday, Nov 8, 8:30pm

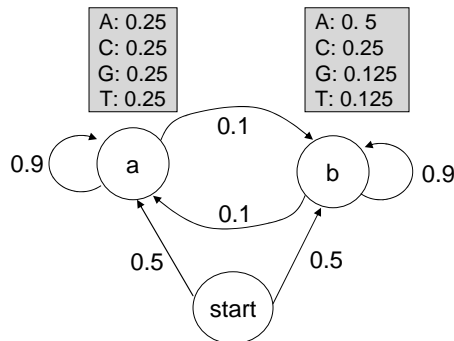
## Problem 1: Probability (15 points)

Consider that you have a box of two types of dice: 90% are type I and 10% are type II. Both types of dice are loaded. Type I dice have 10% of chance of rolling a six, and type II dice have 50% chance of rolling a six. You randomly pick a die from the box and roll once. Then you put the die back into the box and randomly pick one again.

- (1) What is the probability that you will see a six in your first roll?
- (2) If you observed a six, what is the probability that the die is of type I?
- (3) What is the probability that you will see two six in a row?
- (4) If you observed two six in a row, what is the probability that both dice are of type I?

## Problem 2: Hidden Markov Model (15 points)

Consider the hidden Markov model below with given transition and emission parameters.



- (1) What is the most probable state path for a sequence AACT? What is the probability of that path?
- (2) What is the probability of the sequence AACT (considering all possible paths)?

Hint: It's best to use Viterbi and Forward algorithms to answer the two questions, although not absolutely necessary (it would be necessary if I give you a sequence of length 10).

## Problem 3: Boyer-Moore (20 points)

- (1) Implement the extended bad character rule for the Boyer-Moore algorithm. Assume that your pattern and text strings contain only upper-case letters (so you can directly compute indices for each char from its ascii code subtracted by 64. In ascii code A is 65, B is 66, etc.).
  - The syntax of your program should look like this:  
./BM <patternString> <textFileName> .
  - If your input text file contains multiple lines, concatenate them into a single string before you do the matching.
  - Output the starting position of each occurrence of patternString in your text file.

- Count separately the number of character comparisons and the number of steps needed to find the next matching character using the bad character rule.
- (2) Implement the naive string matching algorithm. Input and output formats should be similar to the one above.
  - (3) Download two text files from the course website. One file contains a famous novel, “Moby Dick”, with spaces and punctuation marks removed. The other file contains 1M random DNA sequences. Create several patterns of your choice and find them in the files with your programs. Does BM actually show some advantage? For what kind of patterns BM worked the best and for what kind of patterns BM did not do so well? Why?
  - (4) Bonus (10 points) for implementing the good suffix rule. You can use a naive algorithm to do the preprocessing. (May not worth the points. But I give you an opportunity if you desire to implement it anyway. I just did and it took me about 3 hours.). Similar to last time, if you have done a perfect job for the other problems in this assignment, the bonus points will be wasted.

**Note: Turn in your source code, experimental results on a few (less than 10) patterns, and some discussion. Also email me your source code. Don't print out patterns that appear more than 10 times in the text. It would be a waste of paper.**

## Problem 4: Suffix Tree (20 points)

- (1) Draw a suffix tree for a string ACTACTACA. Label the edges and terminal nodes.
- (2) Draw a joint suffix tree for three strings ACTAC, ATCAT, TCACT. Label the edges and terminal nodes.
- (3) Design an efficient algorithm for finding the shortest nonrepeated string in a text, that is, a shortest string that appears in the text only once.
- (4) Design an efficient algorithm to find the minimum  $l$  for a set of strings  $T_1, T_2, \dots, T_k$ , such that there exist a unique “signature” substring of length  $l$  for each string. For example, if  $T_1 = \text{ACGACGTA}$ ,  $T_2 = \text{ACTATGAC}$ , and  $T_3 = \text{GATAGTA}$ , the smallest  $l = 2$ , since a signature of length 2 can be found for each string: CG only appears in  $T_1$ , CT only in  $T_2$  and AG only in  $T_3$ .

Some informal description of the algorithms is good enough, as long as I can understand your idea and why it is efficient. You can draw graphs to help your explanation.

## Bonus (5 points)

How much time did you spent? Who did you discuss with and what was the discussion about? Any comments about the course and homeworks? What topic would you like to learn more about? What topic you didn't enjoy so much?