Midterm

MPCS 55005: Advanced Algorithms
Due: Noon, Monday, August 3, 2020

Instructions: Write up your answers and submit them in your Github repository. You can either type them in LaTeX or write them on paper and take pictures.

When a problem asks you to give an algorithm, you should provide pseudocode, as well as an English explanation of how your algorithm works and why it is correct. You should also analyze the runtime.

You may use your notes or the recorded lectures. You should not discuss the problems with other students until Monday afternoon.

1. (10 points) You are given a dictionary $D[1..N]$ with N words. You also have an array $Q[1..M]$ of "query words". Each word is a string of letters A-Z. Note that Q may have repeats, but D may not. Give an algorithm that outputs the array $A[1..M]$ with $A[i] =$ whether or not $Q[i]$ is in the dictionary. The runtime of your algorithm should be O(total number of letters in D and Q). Your algorithm should be **deterministic**.

2. In week 2's programming assignment, you were asked to implement a "Multi Bloom Filter" B which consisted of m normal "child" Bloom filters $b_1, \ldots, b_k$ each of capacity n. Recall that an element was said to be contained in a Multi Bloom Filter if and only if it was contained in each of the k children. Also, when an element is added, it is added to each of the k children.

   Clearly, a Multi Bloom Filter meets the requirement of a standard Bloom filter; that is, a Multi Bloom Filter never gives false negatives (neither does a Bloom filter that returns "in set" for every query!). In this problem, you will determine how to optimally size the child filters so that false positive rates are minimized.

   You are given an overall capacity $N = k \cdot n$ and a number of distinct elements $C \cdot N$ that have been added to B. The numbers N and C are fixed; we are trying to choose the best k. You may assume that each child Bloom filter uses an *ideal hash family* $H := H(n) = \{h : \mathbb{Z} \to [n]\}$ such that

   $$\text{For all tuples } (x_1, \ldots, x_\ell) \text{ of distinct integers and } (y_1, \ldots, y_\ell) \in [n]^\ell$$

   $$\Pr_{h \in H}[h(x_1) = y_1 \wedge h(x_2) = y_2 \wedge \cdots \wedge h(x_\ell) = y_\ell] = \frac{1}{n^\ell}$$

   Note that the hash functions for each child are selected independently from this family.

   (a) (3 points) Show that for an ideal hash family $h \in H$ and any $t \in [n]$,

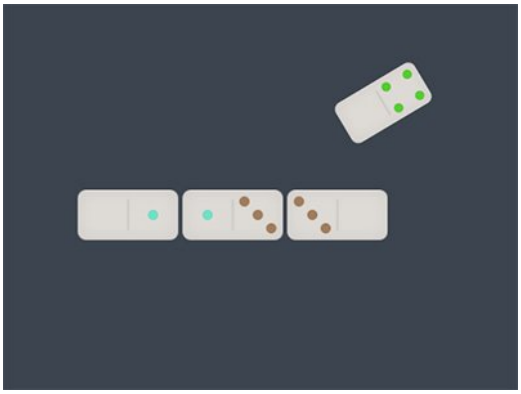
   $$\Pr_{h \in H}[\vee_{i=1}^{\ell} h(x_i) = t] = 1 - (1 - \frac{1}{n})^\ell$$

   for any tuple of distinct integers $(x_1, \ldots, x_\ell)$.

   (b) (3 points) Assume we have inserted $C \cdot N$ elements using k child Bloom filters. Show that the false positive probability of the next query is at least $(1 - e^{-Ck})^k$.

(c) (4 points) Use calculus to compute which k (not necessarily an integer) minimizes the function from the previous part. Use software to plot the function for $C = 1, 2, 3$.

3. In the classic and not-very-fun game of dominos you have a collection of domino tiles, which are rectangular tiles with two numbers on them, one on each end. The goal is to arrange the dominos in a line so that adjacent numbers are equal; see the picture (note that the blank regions on some dominos denote the number 0).[1]



(a) (5 points) Give a polynomial time algorithm for the following problem. Given a set D of dominos, determine whether or not it is possible to arrange all the dominos in a valid line.

Hint: Leonhard Euler

(b) (5 points) Prove that the following problem is NP-complete. Given a set D of dominos and a number n, determine whether or not it is possible to arrange a subset of the dominos in a line so that every number from 1 to n appears exactly twice.

4. You just got a new job at Comcast, and your first task is to help them route packets.

There are n computers connected in a cycle labeled 1,2,...,n. You are given a list of packets $A[1..m]$ with $A[i] = (s_i, t_i)$ which you must route from computer s_i to computer t_i. For each packet you can choose to route it either clockwise or counterclockwise around the cycle. Your goal is to minimize the max over edges of the number of times the edge is used for routing. We will develop an approximation algorithm for this problem.

(a) (3 points) Let OPT be an optimum solution. Let l_i be the number of edges traveled by packet i in OPT. Prove:

$$\mathsf{OPT} \geq \left\lceil \frac{\sum_{i=1}^{m} l_i}{n} \right\rceil$$

(b) (7 points) Give a 3-approximation algorithm for the problem that routes each packet independently of the others. Prove.

(c) (5 points extra credit) Give a 2-approximation for the problem. Prove.

[1] This is actually simplified a bit; read Wikipedia if you want to become the dominos world champion.

5. (10 points, 5 points extra credit) Uh-oh! You forgot to pay the Troll Toll, and as a result, you are forced to play a game for your life. Since the Troll is flush with coins, the game board is played on a circular formation of n quarters (the board). Your objective is to flip all the coins so that they have the same side showing – without ever seeing the board!

   Each round consists of two moves. You begin the round by deciding to flip one or two coins by specifying position(s) from $\{1, 2, 3, 4\}$ (example $n = 4$)

$$\text{BOARD} = \begin{matrix} H & T \\ T & T \end{matrix}, \qquad \text{POSITIONS} = \begin{matrix} 1 & 2 \\ 4 & 3 \end{matrix}$$

   Next, the Troll rotates the board $(360 \cdot \frac{i}{n})$ degrees for some $i = 1, \ldots, n$ (that you **don't know**) and the result is the starting configuration for the next round. You win if, at any time, all the coins are simultaneously all heads or all tails.

   The example below shows the sequence of moves made by you and the Troll in two turns. In the example, $n = 4$, and the initial grid is BOARD_0, and you choose to flip positions 1 and 2, then the Troll rotates the board 90 degrees, then you choose to flip positions 2 and 4 and the Troll rotates the board 180 degrees.

$$\text{BOARD}_0 = \begin{matrix} H & T \\ T & T \end{matrix}, \qquad \text{FLIP}(1,2) = \begin{matrix} T & H \\ T & T \end{matrix}, \qquad \text{ROTATE}(90^\circ) = \begin{matrix} T & T \\ T & H \end{matrix} = \text{BOARD}_1,$$

$$\text{BOARD}_1 = \begin{matrix} T & T \\ T & H \end{matrix}, \qquad \text{FLIP}(2,4) = \begin{matrix} T & H \\ H & H \end{matrix}, \qquad \text{ROTATE}(180^\circ) = \begin{matrix} H & H \\ H & T \end{matrix} = \text{BOARD}_2$$

   Again, you never observe the board or the initial configuration.

   (a) i. (1 point) Show that for $n = 3$, there is no deterministic strategy for beating the Troll.

   ii. (4 points) Show that for $n = 4$, there is a deterministic winning strategy. State the worst-case number of rounds for your strategy.

   (b) (2 points) The Troll chooses the four-quarter ($n = 4$) board and loses to your winning strategy. The Troll is very upset after you beat him. Rather than make good on his promise, he insists you play another game against him. As a compensation **he allows you to flip arbitrary subsets of coins in each move**.

   Show that for all $n \geq 1$, there is a randomized winning strategy for the n-quarter board, i.e., a randomized strategy such that the probability you win in at most r moves tends to 1 as $r \to \infty$. What is the expected number of rounds for your strategy?

   (c) (3 points) Again, the Troll loses and insists you play one more game. He thinks flipping arbitrary subsets of coins is too powerful, so he changes the rule and allows you to flip **at most one coin**. As a compensation he decides that you can ask (before specifying a position) if a single coin is heads before deciding whether or not to make a move.

   Describe a randomized strategy for defeating the Troll, and compute the expected number of rounds.

(d) (5 points extra credit) If you are just allowed to flip at most one coin at a time and not allowed to query any coin before deciding to make a move or not, is there still a randomized winning strategy? Give such a strategy and estimate the expected number of rounds it takes, or show that no such strategy exists.