University of New Mexico
Department of Computer Science

Midterm Examination

CS 561 Data Structures and Algorithms
Fall, 2010

Name:
Email:

- *"Nothing is true. All is permitted"* - Friedrich Nietzsche. Well, not exactly. **You are not permitted to discuss this exam with any other person.** If you do so, you will surely be smitten: collusion on any problem will result in a 0 on the entire exam. However, you may consult any non-human sources including books, papers, web pages, computational devices, animal entrails, seraphim, cherubim, etc. in your quest for truth and solutions. Please acknowledge your sources.
- *Show your work!* You will not get full credit if we cannot figure out how you arrived at your answer. A numerical solution obtained via a computer program is unlikely to get much credit, if any, without a correct mathematical derivation.
- Write your solution in the space provided for the corresponding problem.
- If any question is unclear, ask for clarification.

Question	Points	Score	Grader
1	25		
2	25		
3	25		
4	25		
Total	100		

1. **Recurrences**

   Remember that when the base case for a recurrence is not explicitly given, assume that it is constant for inputs of constant size.

   (a) (4 points) Solve the following recurrence using annihilators: $f(n) = 3f(n-1) - 2f(n-2) + n$. Do not solve for the constant coefficients

   (b) (4 points) Solve the following recurrence using a transformation and the Master method: $f(n) = 4f(\sqrt{n}) + \log n$. Do not solve for the constant coefficients. If an algorithms runtime is given by this recurrence, how would it compare with algorithms with runtimes of $\theta(2^n)$, $\theta(n)$, $\theta(\sqrt{n})$, $\theta(\log n)$?

How many ways can you tile a n by 1 rectangle if you have an infinite supply of dominoes of size x by 1 for each x, $1 \leq x \leq n$? Assume dominoes of the same size are indistinguishable.

(c) (4 points) Let $f(n)$ be the number of unique tilings of a n by 1 rectangle. Write a recurrence relation for $f(n)$. Include the base case(s).

(d) (5 points) Now guess an exact solution for this recurrence relation and prove your solution is correct using proof by induction.

(e) (8 points) Now what if the dominoes can be red or black? Write down the recurrence, and an inductive proof of the solution.

2. **Probability**

The following two problems are similar to a problem in homework 1, although note that the cards now have an additional attribute.

Imagine a card game where each card has 4 attributes: number, shape, color and shading; and each attribute has three possible values: number is 1, 2 or 3; shape is circle, square or triangle; color is red, green or blue; and shading is none, dashed or solid. Each card in the deck is unique, so there is a total of $3 * 3 * 3 * 3 = 81$ cards.

A *match* is a set of 3 cards where for each of the 4 attributes, the 3 cards either all have the same value for that attribute, or they all have different values for that attribute. For example the following set of cards is a match: (1, circle, red, none), (2, square, red, dashed), (3, triangle, red, solid)

(a) (4 points) If I lay out n cards on a table where $n \leq 81$, what is the expected number of sets of 3 cards that will form matches? Show that your answer makes sense for the boundary conditions ($n = 3$ and $n = 81$)?

(b) (4 points) Now, use Markov's inequality to bound the probability that there are at least k matches for any k, when n cards are laid on the table. In particular, what does this say about the probability that there is at least one match?

Consider a wireless network consisting of n^2 nodes laid out on a n by n grid. A pair of nodes are said to be *neighbors* if they are immediately adjacent either horizontally or vertically on the grid (thus a node has at most 4 neighbors). For some number ℓ, each node chooses a channel uniformly at random from 1 to ℓ. Two nodes are said to *collide* if they are neighbors and they have both chosen the same channel.

Note: The events that collisions occur are *not* independent. In particular, consider 4 nodes on a square: a and b on the left and c and d on the right. If a and b, b and c, and c and d collide, then a and d **must** collide.

(c) (6 points) Use a union bound to get an upperbound on the probability that there are *any* collisions. How large must ℓ be to ensure that this probability is less than 1/2?

(d) (5 points) Now use Markov's inequality to bound the probability that n pairs of nodes collide.

(e) (6 points) Imagine that the purpose of the wireless network is to convey messages from the top to bottom where the same message is sent redundantly along each column; and to convey messages from left to right, where the same message is sent redundantly along each row. Thus, we want to ensure that there is some column where each pair of neighboring nodes in that column does not collide, and some row with the same property. Using your work from above, determine how large ℓ must be to ensure that this property holds with probability at least 9/10. (Remember: the collisions are *not* independent!)

3. **Data Structures**

Your colleague wants to change the rules of red-black trees to the following:

- The root node and leaf nodes (NIL) can be either red or black
- If a node is red and not a leaf node, both of its children are black
- If a node is black and not a leaf node, both of its children are red
- For each node, all paths from the node to descendant leaves contain the same number of black nodes

(a) (6 points) Is it possible to use these rules to create a balanced BST data structure? If so, sketch your solution. If not, show how things can go wrong with a minimum size counter-example.

(b) (5 points) Your boss wants to create the following data structure in the comparison model and to name it after himself, the *Merkle*. A Merkle has the following operations and properties on it. BuildMerkle takes an arbitrary array and builds a Merkle from it in $O(n)$ time. The resulting Merkle will provide the following operations. FindMin (resp. FindMax) will return the minimum (resp. maximum) element and run in $O(\log n)$ time. Successor(x) (resp. Predecessor(x)) return the next largest (resp. smallest) element in the Merkle after the element x, and both of these operations run in $O(1)$ time. Intuitively, your boss wants you to combine the nice properties of the heap with the nice properties of a data structure like skip lists. Can you immortalize your boss's name in CS textbooks by creating this data structure?

In this problem, you will modify count-min sketches so that they handle negative counts. As in class, assume you are presented with a stream of tuples of the form (i_t, c_t), except now c_t may be either a negative or positive integer. The data structure you will use will consist of two count-min sketches, a positive count-min sketch for positive counts and a negative count-min sketch for negative counts. In particular, each of the two sketches will use m counters and k hash functions, where all hash functions can be assumed to be independent. If c_t is positive, in the positive count-min sketch (positive sketch for short), for each $1 \leq a \leq k$, $C_{a,h_a(i)}$ will be incremented by c_t. If c_t is negative, in the negative sketch, for each $1 \leq a \leq k$, $C_{a,h_a(i)}$ will be incremented by $-c_t$. The estimate of the count of an item, i at time T is $m^+(i,T) - m^-(i,T$, where $m^+(i,T)$ is the value of the smallest counter associated with i in the positive sketch and $m^-(i,T)$ is the value of the smallest counter associated with i in the negative sketch. As in class, let $Count(i,T)$ be the true count of item i in the stream up to time T. Also assume that $k = m\epsilon/e$ for the positive sketch and for the negative sketch.

(c) (7 points) Give a good bound on the probability that the following holds:

$$Count(i,T) - \epsilon \sum_{i=1}^{T} |c_i| \leq m(i,T) \leq Count(i,T) + \epsilon \sum_{i=1}^{T} |c_i|$$

Please prove your bound.

(d) (7 points) Now imagine you are given a constant number of data streams $D_1, D_2, \ldots, D_c$ and weights associated with them $w_1, w_2, \ldots w_c$ that may be positive or negative real numbers. For each item i, at time T, define $Count(i, T)$ to be the weighted sum of the count values seen in all data streams up to time T, where a count value seen in stream i is weighted by w_i. Assume now that all count values seen are positive. Describe a data structure based on count-min sketches that can approximate $Count(i, T)$. How much memory does your data structure use? How closely can you approximate $Count(i, T)$ and with what probability? Please justify your answers. For consistency in notation, please let $S(i, j, T)$ be the sum of the counts of item i in stream j up to time T.

4. **Dynamic Programming**

Consider a collection of n nodes aligned on a line, numbered 1 to n. Two nodes are connected by an edge if they are adjacent on the line, e.g. nodes i and $i+1$ are neighbors. For each pair i, $i+1$ of neighboring nodes, there is a weight $w_{i,i+1}$ associated with the pair, which may be either positive or negative.

In this problem, each node will be colored with one of two colors, red or blue. If a pair $(i, i+1)$ of neighboring nodes are colored the same, the cost associated with that pair is $w_{i,i+1}$; if the pair are colored differently, the cost associated with that pair is $-w_{i,i+1}$. The total cost of a coloring is the sum of the costs of all neighboring pairs.

(a) (10 points) Describe a dynamic program to output the minimum cost of any coloring, when given all edge weights. Hint: Let $c(i, r)$ be the minimum cost of coloring nodes 1 through i when node i is colored red. Let $c(i, b)$ be the minimum cost of coloring nodes 1 through i when node i is colored blue.

Now imagine that the nodes are connected in a n by n grid, and that each node can be colored with m possible colors. There is an edge between a pair of nodes on the grid if they are immediately adjacent either horizontally or vertically; again, each edge has a weight associated with it that may be either positive or negative. (For example neighboring nodes (i, j) and $(i+1, j)$ would have an edge with weight $w((i, j), (i+1, j))$)

(b) (15 points) Describe a dynamic program to output the minimum cost of any coloring, when given all edge weights for a grid. What is the runtime of your algorithm?