Midterm 1

Write in the following boxes clearly and then double check.

Name :

SID :

- **The exam will last 110 minutes.**
- The exam has 4 questions with a total of 100 points. You may be eligible to receive partial credit for your proof even if your algorithm is only partially correct or inefficient.
- Only your writings inside the answer boxes will be graded. **Anything outside the boxes will not be graded.** The last page is provided to you as a blank scratch page.
- Answer all questions. Read them carefully first. Not all parts of a problem are weighted equally.
- Be precise and concise.
- The problems may **not** necessarily follow the order of increasing difficulty.
- The points assigned to each problem are by no means an indication of the problem's difficulty.
- Unless the problem states otherwise, you should assume constant time arithmetic on real numbers.
- If you use any algorithm from lecture and textbook as a blackbox, you can rely on the correctness of the quoted algorithm. If you modify an algorithm from textbook or lecture, you must explain the modifications precisely and clearly, and if asked for a proof of correctness, give one from scratch or give a modified version of the textbook proof of correctness.
- Good luck!

1 Antipasti Short Answers

(a) (5 pts) Let $f(n) = \log(n!)$. Prove $f(n) = \Theta(n \log n)$.

(b) (3 pts * 4) Fill out the following four blanks in terms of n, then give a construction of such a graph.

i.e. if your answer for a blank is $f(n)$, then describe describe how, given n, you can construct a graph on n vertices with $f(n)$ edges that satisfy the conditions. You are allowed to fill the blank in form of sums. **All graphs are simple** (no self-loops, at most one edge between two vertices).

In the first two parts below, we say a directed graph G is *weakly connected* if the undirected graph obtained by removing all edge directions from G is connected.

(i) There are at least _____ edges in a weakly connected directed acyclic graph on n vertices.

(ii) There are at most _____ edges in a weakly connected directed acyclic graph on n vertices.

(iii) There are at least _____ edges in a strongly connected directed graph on n vertices.

(iv) There are at most _____ edges in a strongly connected directed graph on n vertices.

(c) (7 pts) To implement `Dijkstra`'s algorithm, is it faster to use an adjacency list to represent the graph or is it faster to use an adjacency matrix? Explain why. Assume $|E| = o(|V|^2)$ (informally, assume the graph is not dense).
Hint: Recall from lecture that an adjacency matrix is stored as a 2D array (hence the word "matrix") and an adjacency list is stored as a 1D array of linked lists.

(d) (4 pts) How would you speed up computing the `FFT` of $(a_0, a_0, a_1, a_1, a_2, a_2, \dots)$ by a factor of roughly 2 (*i.e.* halve the runtime)?

(e) (8 pts) Is the following statement true or false? Justify your answer.

"Let an M-path between two vertices s and t (in an undirected, connected graph G) be a path such that the weight of every edge is at most M. If G has an M-path between s and t, then there is a minimum spanning tree of G with an M-path between s and t."

2 Whole Lotta Love for Chicken Wings

(20 pts) Kevin is placing L-shaped chicken wings on a n by n oven tray, where n **is a power of 2**. Each chicken wing can be viewed as a 2 by 2 square with one of its 1 by 1 squares missing. One of the oven tray's squares was severely burnt and must be left empty.

Given n, x, y (n is a power of 2, $n \geq 2$, $x, y \in \{1 \dots n\}$), where (x, y) is the position of the burnt square that he wants to leave empty, how do we fill the entire n by n tray excluding square (x, y) with chicken wings? The chicken wings cannot overlap each other, nor can they extend out of the tray (or they'll get burnt!).

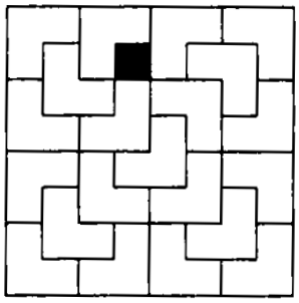
Describe an efficient algorithm that outputs a possible placement of the chicken wings, given n and (x, y). Then prove the algorithm's correctness and analyze its runtime.

Feel free to draw figures in your answer if it helps you clarify your description. Marking and deleting the placement of one chicken wing takes constant time.

Example:

$$n = 8; (x, y) = (4, 2)$$

One possible output. We denote the burnt square by shading it black.



Hint: Divide and conquer. Draw a 8x8 or 16x16 grid to try something out first.

3 When the Levee Breaks

(18 pts) A levee broke and the water is flooding a group of villages[1]. There are n villages and m roads between the villages. The water floods one road at a time. For each road e_i, you're given the unique integer timestep t_i of when this road is flooded. $(t_i)_{i=1}^m$ is a permutation of $\{1 \dots m\}$. Give an efficient algorithm that computes the array A, where A_i denotes the number of connected components of villages[2] at timestep $i \in \{1 \dots m\}$ and then analyze its runtime. **Proof of correctness is not needed for this question.**
Example:

$$n = 4, m = 4.$$
$$\text{The roads } (e_i)_{i=1}^4 = (1,2);\ (2,3);\ (3,4);\ (4,1)$$
$$\text{Timesteps } (t_i)_{i=1}^4 = 3;\ 2;\ 4;\ 1$$

At timestep 1, $e_4 = (1,4)$ is flooded, there's 1 connected component: $\{1,2,3,4\}$.
At timestep 2, $e_4 = (1,4)$ and $e_2 = (2,3)$ are flooded, there are 2 connected components: $\{1,2\}, \{3,4\}$.
At timestep 3, $e_4, e_2,$ and e_1 are flooded, there are 3 connected components: $\{1\}, \{2\}, \{3,4\}$.
At timestep 4, all roads are flooded, there are 4 connected components: $\{1\}, \{2\}, \{3\}, \{4\}$.
Therefore $A = (1,2,3,4)$

[1] All villagers have evacuated so there are no casualties.
[2] A group of villages in which any two villages are connected to each other by unflooded roads, and which is connected to no additional villages in the rest of the n total villages.

4 String matching with wildcards

We are given two strings P, T over the alphabet $\Sigma = \{0, 1, \ldots, s-1\}$. P has length m and T has length n, where $m \leq n$. We would like to find all occurrences of the pattern P in T. For example, the pattern $P = 101$ appears twice in the string $T = 10101$: namely at positions 0 and 2 in T (using 0-based indexing). In what follows, you only need to provide a short algorithm without proof.

Note: The last two parts of this problem are difficult. We assigned fewer points to the last two parts to encourage you to check your answers for other problems instead if you finished everything else on the exam but stuck on (d) and (e).

(a) (6 pts) When $s = 2$, give an $O(n \log n)$ time algorithm for this problem. **Hint:** what if you replace each 1 in an input string with 01 and replace 0 with 10?

(b) (5 pts) What if we only want to find positions in T that patch P up to at most 10% error?

(c) (7 pts) Show how to improve the runtime of (a) to $O(n \log m)$. **Hint:** solve $O(n/m)$ instances of a problem solved in class.

(d) (4 pts) Give an $O(n \log n)$ time (no dependence on s) algorithm for exact matching over general alphabets, i.e. where s is not necessarily 2. **Hint:** $a = b$ iff $(a-b)^2 = a^2 + b^2 - 2ab = 0$.
Note: If your answer for this part is entirely correct you automatically get full points for part (a). If you wrote something like "use my algorithm described in (d)" in (a), you'll get 0 points for (a) unless you obtain full score in (d).

(e) (4 pts) Give an $O(n \log n)$ time algorithm for exact matching (i.e. not 10% error) even if the strings are allowed to have wildcard characters (a wildcard matches any single character).