# Midterm 1

- Do not open this quiz booklet until you are directed to do so. Read all the instructions first.
- This quiz contains 4 problems. You have 120 minutes to complete all problems. The points for each problem in the midterm roughly indicate how many minutes you should spend on the problem.
- This quiz booklet contains 12 pages, including this one and 3 sheets of scratch paper. **Please do not remove any pages from the booklet, including the scratch papers!**
- Write your solutions in the space provided. If you continue your solution on the back of any page, please make a note of it. If you run out of space, continue on a scratch page and make a note that you have done so.
- Do not waste time deriving facts that we have studied. Just cite results from class.
- When we ask you to "give an algorithm" in this quiz, describe your algorithm in plain English or if necessary, pseudocode, and provide a short argument for correctness and running time. You do not need to provide a diagram or example unless it helps make your explanation clearer.
- Do not spend too much time on any one problem.
- **Please write your name on every single page of this exam.**
- Good luck!

| Problem | Title | Points | Parts | Grade | Initials |
|---|---|---|---|---|---|
| 0 | Name | 1 | 1 | | |
| 1 | True/False Questions | 30 | 7 | | |
| 2 | Majority Element | 30 | 1 | | |
| 3 | Parenthesizing | 30 | 1 | | |
| 4 | Planting Flowers | 30 | 1 | | |
| Total | | 121 | | | |

**Quiz 1-1: [30 points] True/False Questions**

Mark each statement as either true or false and **provide a short explanation for each answer**.

**(a) [5 points]** The median of medians algorithm, when run on groups of size 7 instead of 5, has the recurrence $T(n) = T\left(\frac{n}{7}\right) + T\left(\frac{5n}{7}\right) + O(n)$.

**(b) [4 points]** If the primal Linear Program is feasible and has a bounded optimum, then the dual Linear Program is feasible.

**(c) [4 points]** The majority of the work in the mergesort algorithm is done at the root of the recurrence tree.

**(d) [4 points]** Let $G = (V, E)$ be a connected graph with unique edge weights where $|E| \geq 2$ with at most one edge between each pair of vertices. Then the edge with the second smallest weight must be in the MST for $G$.

**(e) [4 points]** If an edge $e$ is in the MST for a graph $G$, then it must be a minimum weight edge across some cut in $G$.

**(f) [4 points]** Given two sets of integers $X$ and $Y$, such that $|X| = |Y| = n$, we can compute their Minkowski Sum $\{x + y | x \in X, y \in Y\}$ using FFT in $O(n \log n)$ time.

**(g) [5 points]** *In Johnson's algorithm for APSP in a graph G, recall that we compute the re-weighting function $h : V \rightarrow \mathbb{R}$ by introducing a new vertex $s$ and adding an edge of weight 0 from $s$ to all the vertices in G. We then run Bellman-Ford from $s$ and set $h(u)$ to be the shortest path from $s$ to $u$.*

Instead of 0, we can set the weight of the new edges from $s$ to be an arbitrary constant $c \neq 0$ and still obtain a function $h$ that works.

**Quiz 1-2: [30 points] Majority Element**

Suppose we have a list of $n$ objects, where $n = 2^k$ for some positive integer $k$. The only operation that we can use on these objects is the following: $isEqual(i, j)$, which runs in constant time and returns True if and only if two objects $i$ and $j$ are equal. The objects are neither hashable nor comparable.

Our goal is to find if there exists a majority (more than $n/2$ occurrences) of the same element and if so, return it.

Design an $O(n)$ time algorithm that finds whether a majority element exists, and if so, returns it. Provide arguments for your algorithm's correctness and runtime.

**Note:** A fully correct $O(n \log n)$ time algorithm with correctness and runtime arguments will receive 25 / 30 points.

**Hint for $O(n)$ solution:** Consider implementing and using a subroutine $Reduce(L)$ which converts the list $L$ into an $n/2$ sized list of each adjacent pairs.

For an example list $L$, where

$$L = [\ddagger, \oslash, \star, \ddagger, \sqcap, \sqcap, \ddagger, \ddagger, \ddagger, \oslash, \ddagger, \bigcirc, \ddagger, \uplus, \ddagger, \ddagger]$$

A single iteration of $Reduce(L)$ would create a list $L'$ as follows:

$$L' = [(\ddagger, \oslash), (\star, \ddagger), (\sqcap, \sqcap), (\ddagger, \ddagger), (\ddagger, \oslash), (\ddagger, \bigcirc), (\ddagger, \uplus), (\ddagger, \ddagger)]$$

**Quiz 1-3: [30 points] Parenthesizing**

Ben is preparing a homework for his students to practice arithmetic operations. He has written down a sequence of $n$ numbers $a_1, a_2, \ldots, a_n$ with $n-1$ arithmetic operations $O_1, O_2, .. O_{n-1}$ between each pair of consecutive numbers, where each $O_i$ is an addition $(+)$ or multiplication $(\times)$. The expression looks like:

$$a_1\, O_1\, a_2\, O_2\ ..\ O_{n-1}\, a_n$$

He notices that the final answer depends on how one may parenthesize the expression. Unfortunately, Ben suffers from sevenophilia. Thus, he won't be happy with the assignment unless the final answer is zero modulo seven. Alice wants to help him by adding exactly $n-1$ pairs of parentheses (one for each operation) to the expression, such that the final answer becomes zero modulo seven. It turns out there may be many possible ways to do this.

Give an $O(n^3)$ time algorithm that finds how many ways there are to add parentheses to the expression so that the final expression evaluates to zero modulo seven.

**Example:** Given the expression $5 + 2 \times 3$, the solution is $1$. This is because there is only one way to parenthesize the expression to get $0 \mod 7$, namely $((5+2) \times 3) \equiv 0 \mod 7$.

**Note:** You may use without proof that $a \times b \mod p = (a \mod p) \times (b \mod p) \mod p$.

**Quiz 1-4: [30 points] Planting Flowers**

Ben Bitdiddle wants to plant flowers in his garden, which is a grid with $n$ rows and $m$ columns. He learns that city regulations require that each row $i$ can have at most $r_i$ flowers and each column $j$ can have at most $c_j$ flowers. Subject to these rules, Ben wants to plant as many flowers in his garden as possible.

Unfortunately some grid cells in his garden have large rocks in them, so there is no space for flowers. All other grid cells can each contain up to $d$ flowers. You may assume you are given $b_{i,j}$ where $b_{i,j} = 1$ if cell $(i, j)$ has space for flowers, and 0 otherwise (if there is a large rock there).

Given all $d$, $r_i$, $c_j$ and $b_{i,j}$, find an $O(n^2m^2d)$ algorithm that outputs how many flowers to plant in each cell such that the number of flowers is maximized. Argue the correctness of your algorithm and its runtime.

**Hint:** Consider constructing a flow network.

**Scratch page**

**Scratch page**

*Name*____________________________________

**Scratch page**