<table>
<thead>
<tr>
<th>#</th>
<th>Score</th>
<th>Grader</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
To acknowledge the corn - This purely American expression means to admit the losing of an argument, especially in regard to a detail; to retract; to admit defeat. It is over a hundred years old. Andrew Stewart, a member of Congress, is said to have mentioned it in a speech in 1828. He said that haystacks and cornfields were sent by Indiana, Ohio and Kentucky to Philadelphia and New York. Charles A. Wickliffe, a member from Kentucky questioned the statement by commenting that haystacks and cornfields could not walk. Stewart then pointed out that he did not mean literal haystacks and cornfields, but the horses, mules, and hogs for which the hay and corn were raised. Wickliffe then rose to his feet, and said, "Mr. Speaker, I acknowledge the corn"

- Funk, Earle. "A Hog on Ice and Other Curious Expressions"

1 Required Problems

1. Upside Down
   [10 Points]

   Suppose that in addition to the standard kind of comparator, we introduce an "upside-down" comparator that produces its minimum output on the bottom wire and its maximum output on the top wire.

   (a) [6 Points] An \( n \)-input sorting network with \( m \) comparators, is represented by a list of \( m \) pairs of integers in the range from 1 to \( n \). Thus, a comparator between the wire \( i \) and \( j \) is represented as \((i, j)\). If \( i < j \) then this is a regular comparator, and if \( i > j \) it is an upside-down comparator.

   Describe an algorithm that converts a sorting network with \( n \) inputs, \( c \) upside-down gates and \( m \) overall gates into an equivalent (i.e., with the same number of gates) sorting network that uses only regular gates. How fast is your algorithm?

   (b) [4 Points] Prove that your algorithm is correct (i.e., it indeed outputs a network that uses only regular comparators, it always terminate, and the output network is equivalent to the input network).

2. Merge those sequences
   [20 Points]
2. Figure 1: Permutation Switch

(a) [10 Points] Consider a merging network with inputs $a_1, a_2, \ldots, a_n$, for $n$ an exact power of 2, in which the two monotonic sequences to be merged are $\langle a_1, a_3, \ldots, a_{n-1} \rangle$ and $\langle a_2, a_4, \ldots, a_n \rangle$ (namely, the input is a sequence of $n$ numbers, where the odd numbers or sorted, and the even numbers are sorted). Prove that the number of comparators in this kind of merging network is $\Omega(n \log n)$. Why is this an interesting lower bound? (Hint: Partition the comparators into three sets.)

(b) [10 Points] Prove that any merging network, regardless of the order of inputs, requires $\Omega(n \log n)$ comparators. (Hint: Use question 1.)

3. Permutations.

[20 Points]

A permutation network on $n$ inputs and $n$ outputs has switches that allow it to connect its inputs to its outputs according to any $n!$ possible permutations. Figure 1 shows the 2-input, 2-output permutation network $P_2$, which consists of a single switch that can be set either to feed its inputs straight through to its outputs or to cross them.

(a) [5 Points] Argue that if we replace each comparator in a sorting network with the switch of Figure 1, the resulting network is a permutation network. That is, for any permutation $\pi$, there is a way to set the switches in the network so that input $i$ is connected to output $\pi(i)$.

(b) [2 Points] Figure 2 shows the recursive construction of an 8-input, 8-output permutation network $P_8$ that uses two copies of $P_4$ and 8 switches. The Switches have been set to realize the permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(8)) = (4, 7, 3, 5, 1, 6, 8, 2)$, which requires (recursively) that the top $P_4$ realize $(4, 2, 3, 1)$ and the bottom $P_4$ realize $(2, 3, 1, 4)$. Show how to realize the permutation $\langle 5, 3, 4, 6, 1, 8, 2, 7 \rangle$ on $P_8$ by drawing the switch settings and the permutations performed by the two $P_4$’s.

(c) [5 Points] Let $n$ be an exact power of 2. Define $P_n$ recursively in terms of two $P_{n/2}$’s in a manner similar to the way we defined $P_8$.

Describe an algorithm (ordinary random-access machine) that runs in $O(n)$-time that sets the $n$ switches connected to the inputs and outputs of $P_n$ and specifies the permutations that must be realized by each $P_{n/2}$ in order to accomplish any given $n$-element permutation. Prove that your algorithm is correct.
(d) [3 Points] What are the depth size of $P_n$? How long does it take on an ordinary random-access machine to compute all switch settings, including those within the $P_{n/2}$'s?

(e) [5 Points] Argue that for $n > 2$, any permutation network (not just $P_n$) must realize some permutation by two distinct combinations of switch settings.

4. THE HARD LIFE OF A JOURNALIST
[20 Points]

(a) [2 Points] A journalist, named Jane Austen, travels to Afghanistan, and unfortunately falls into the hands of Bin Laden. Bin Laden offer Jane a game for her life – if she wins she can leave.

The game board is made out of $2 \times 2$ coins:

At each round, Jane can decide to flip one or two coins, by specifying which coins she is flipping (for example, flip the left bottom coin, and the right top coin), next Bin Laden goes and rotates the board by either 90, 180, 270, or 0 degrees (of course, rotation by 0 degrees is just keeping the coins in their current configuration).

The game is over when all the four coins are either all heads or all tails. To make things interesting, Jane does not see the board, and does not know the starting configuration. Describe an algorithm that Jane can deploy, so that she always win. How many rounds are required by your algorithm?
(b) [5 Points] After escaping from Bin Laden, and on her way to Kabul, Jane meets a peace loving, nuclear reactor selling, French diplomat. The French diplomat is outraged to hear that Jane prefers Hummus to French Fries, and instruct his bodyguards to arrest Jane immediately, accusing her of being a quisling of the French cuisine (Jane has French citizenship). Again, the diplomat offers her a game for her life, similar to the Bin Laden game, with the following twist: after Jane flips her coins, the diplomat will reorder the coins in an arbitrary order (without flipping any coin). Describe an algorithm that Jane can use to win the game. What is the expected number of rounds Jane has to play before winning (the lower your bound, the better).

(c) [5 Points] After escaping from the French diplomat, Jane travels to Hanoi to investigate rumors that the priests in charge of the Towers of Hanoi games, are spending all the money they get on buying computer games and playing them, instead of playing the holy game of Towers of Hanoi, as they are suppose to do.

However, the head priest is willing to do an interview with Jane, only if she plays the coin game (using the French diplomat version), with $n$ coins. Describe an algorithm that guarantees that Jane wins. Provide an upper bound (as tight as possible) on the number of rounds Jane has to play before winning. (Providing an exact bound here is probably hard. As such, a rough upper bound would be acceptable.)

(d) [5 Points] Jane, tired of all those coin games, goes to Nashville for a vacation. Unfortunately for her, she is kidnapped by an Elvis lookalike. Not surprisingly, he offers her to play the coin game for her life, with the following variants: There are $n$ coins, and at each round Jane can choose which of the $n$ coins she wants to flip. Before flipping the coin, the Elvis lookalike tells her whether the coin is currently head or tail, and Jane can decide whether she wants to flip this coin or not. After each round, the Elvis lookalike takes the coins and reorder them in any order he likes. Describe an algorithm that guarantees that Jane wins. Provide an exact bound on the expected number of rounds that Jane has to play before she wins. (The smaller your bound, the better.)

5. The Hard Hard Life of the IRS.
[10 Points]

The IRS receives, every year, $n$ forms with personal tax returns. The IRS, of course, can not verify all $n$ forms, but they can check some of them. Describe an algorithm, as fast as possible, that decides whether the number of incorrect tax forms is larger than $\varepsilon n$, where $\varepsilon$ is a prespecified constant between 0 and 1.

The decision of the algorithm is considered to be incorrect if it declares that the number of incorrect forms is smaller than $\varepsilon n$, but it is in fact larger than $2\varepsilon n$. Similarly, the algorithm is considered to be incorrect if it claims that the number of incorrect forms is larger than $2\varepsilon n$, where it is in fact smaller than $\varepsilon n$. (Namely, if the number of incorrect forms is between $\varepsilon n$ and $2\varepsilon n$, any of the two answers are acceptable.)

Your algorithm should output a correct result with probability $\geq 1 - 1/n^{10}$. What is the running time of your algorithm, assuming that verifying the correctness of a single tax form takes $O(1)$ time? (Hint: Use the Chernoff inequalities.)
6. Closest Numbers

Let $P$ be a set of $n$ real numbers. The purpose of this exercise is to develop a linear time algorithm for deciding whether there are two equal numbers in $P$. Let $x_1, \ldots, x_n$ be a random permutation of the numbers in $P$.

(a) \textbf{[5 Points]} Let $\pi_i = \min_{1 \leq k < j \leq i} |x_k - x_j|$ be the distance between the closest pair of numbers in $x_1, \ldots, x_i$. Prove that $Pr[\pi_i \neq \pi_{i-1}] \leq 2/i$.

(b) \textbf{[5 Points]} Given a parameter $r$, describe an algorithm, that decides in $O(i)$ time, whether $\pi_i < r$. Furthermore, if $\pi_{i-1} = r$ but $\pi_i < r$, then it computes $\pi_i$. (Hint: use hashing and the floor function.)

(c) \textbf{[5 Points]} Show how to modify the previous algorithm into a data-structure, so that after computing $\pi_i$, one can insert $x_{i+1}, \ldots, x_j$ into the data-structure in $O(1)$ time per element, where $\pi_{i+1} = \pi_{i+2} = \cdots = \pi_{j-1} > \pi_j$. And furthermore, the data-structure returns $\pi_j$.

(d) \textbf{[5 Points]} Describe an algorithm, with $O(n)$ expected running time, that computes $\pi_n$. Clearly, if $\pi_n = 0$ then there are two identical numbers in $P$. (Hint: Use (a) and (c).)

(Note, that the algorithm of (d) is faster than one can achieve in the comparison model (i.e., we only only to compare numbers). One can prove that the fastest algorithm for this problem in the comparison model requires $\Omega(n \log n)$ time. Namely, the only way to solve it is using sorting.)
2 Practice Problems

2.1 Sorting Networks

1. [10 Points] (Based on CLRS 27.1-2 and 27.1-4, or CLR 28.1-2 and 28.1-4)
   
   (a) [5 Points] Let \( n \) be an exact power of 2. Show how to construct an \( n \)-input, \( n \)-output comparison network of depth \( \log n \) in which the top output wire always carries the minimum input value and the bottom output wire always carries the maximum input value.

   (b) [5 Points] Prove that any sorting network on \( n \) inputs has depth at least \( \log n \).

2. [10 Points] (Based on CLRS 27.2-5 or CLR 28.2-5)

   Prove that an \( n \)-input sorting network must contain at least one comparator between the \( i \)th and \((i + 1)\)st lines for all \( i = 1, 2, ..., n - 1 \).

3. [10 Points] (Based on CLRS 27.5-1 and 27.5-2, or CLR 28.5-1 and 28.5-2)

   The sorting network \( \text{SORTER}[n] \) was present in class (it is also shown in CLRS Figure 27.12 or CLR Figure 28.12), where \( n \) is an exact power of 2. Answer the following questions about \( \text{SORTER}[n] \).

   (a) [5 Points] Give a tight bound for the number of comparators in \( \text{SORTER}[n] \).

   (b) [5 Points] Show that the depth of \( \text{SORTER}[n] \) is exactly \((\log n)(\log n + 1)/2\).

4. [10 Points] (Based on CLRS 27.5-3 or CLR 28.5-3)

   Suppose that we have \( 2n \) elements \( < a_1, a_2, ..., a_{2n} > \) and wish to partition them into the the \( n \) smallest and the \( n \) largest. Prove that we can do this in constant additional depth after separately sorting \( < a_1, a_2, ..., a_n > \) and \( < a_{n+1}, a_{n+2}, ..., a_{2n} > \).

5. [10 Points] (Based on CLRS 27.5-4 or CLR 28.5-4)

   Let \( S(k) \) be the depth of a sorting network with \( k \) inputs, and let \( M(k) \) be the depth of a merging network with \( 2k \) inputs. Suppose that we have a sequence of \( n \) numbers to be sorted and we know that every number is within \( k \) positions of its correct position in the sorted order, which means that we need to move each number at most \((k - 1)\) positions to sort the inputs. For example, in the sequence \( 3 2 1 4 5 8 7 6 9 \), every number is within 3 positions of its correct position. But in sequence \( 3 2 1 4 5 9 8 7 6 \), the number 9 and 6 are outside 3 positions of its correct position.

   Show that we can sort the \( n \) numbers in depth \( S(k) + 2M(k) \). (You need to prove your answer is correct.)

6. [20 Points] (Based on CLRS 27.5-5 or CLR 28.5-5)

   We can sort the entries of an \( m \times m \) matrix by repeating the following procedure \( k \) times:

   (1) Sort each odd-numbered row into monotonically increasing order.

   (b) Sort each even-numbered row into monotonically decreasing order.

   (c) Sort each column into monotonically increasing order.
(a) [8 Points] Suppose the matrix contains only 0’s and 1’s. We repeat the above procedure again and again until no changes occur. In what order should we read the matrix to obtain the sorted output \((m \times m)\) numbers in increasing order? Prove that any \(m \times m\) matrix of 0’s and 1’s will be finally sorted.

(b) [8 Points] Prove that by repeating the above procedure, any matrix of real numbers can be sorted. [Hint: Refer to the proof of the zero-one principle.]

(c) [4 Points] Suppose \(k\) iterations are required for this procedure to sort the \(m \times m\) numbers. Give an upper bound for \(k\). The tighter your upper bound the better (prove you bound).

2.2 Randomized Algorithms

1. Tail Inequalities
   [10 Points]
   
   (a) Prove the following theorem:

   **Theorem 2.1** Let \(X_1, X_2, \ldots, X_n\) be independent coin flips such that for \(1 \leq i \leq n\), we have \(Pr[X_i = 1] = p_i\), where \(0 < p_i < 1\). Then, for \(X = \sum_{i=1}^{n} X_i\), \(\mu = E[X] = \sum_i p_i\) and for any \(\delta > 0\),
   \[Pr \left[ X > (1 + \delta)\mu \right] < \left[ \frac{e^{\delta}}{(1 + \delta)^{(1+\delta)}} \right]^\mu,
   \]
   
   and \[Pr \left[ X < (1 - \delta)\mu \right] < \exp \left( \frac{-\mu\delta^2}{2} \right).\]

   (b) Consider a collection of \(n\) random variables \(X_i\) drawn independently from the geometric distribution with mean 2 – that is, \(X_i\) is the number of flips of an unbiased coin up to and including the first head. Let \(X = \sum X_i\). Derive an upper bound as small as possible on the probability that \(X > (1 + \delta)(2n)\) for any fixed \(\delta\).

2. Tournament without a winner
   [10 Points]
   
   Consider a tournament on \(n\) teams, in which each pair of teams play against each other once and a winner is always declared. Suppose we try to rank the teams in some total order based on the outcome of the tournament. Say that a game agrees with the ranking we have chosen if the team we ranked better won. Prove that for sufficiently large \(n\), there is a possible set of outcomes such that no ranking agrees with more than 51% of the games. (Hint: Pick the winner in a game randomly and use the results of exercise 1 above.)

3. Fuzzy Sorting of Intervals (Based on CLRS 7-6)
   [10 Points]
   
   Consider a sorting problem in which the numbers are not known exactly. Instead, for each number, we know an interval on the real line to which it belongs. That is, we are given \(n\) closed intervals of the form \([a_i, b_i]\), where \(a_i \leq b_i\). The goal is to fuzzy-sort these intervals, i.e., produce a permutation \(< i_1, i_2, \ldots, i_n >\) of the intervals such that there exist \(c_j \in [a_{i_j}, b_{i_j}]\) satisfying \(c_1 \leq c_2 \leq \cdots \leq c_n\).
(a) [5 Points] Design an algorithm for fuzzy-sorting \( n \) intervals. Your algorithm should have the general structure of an algorithm that quicksort the left endpoints (the \( a_i \)'s), but it should take advantage of overlapping intervals to improve the running time. (As the intervals overlap more and more, the problem of fuzzy-sorting the intervals gets easier and easier. Your algorithm should take advantage of such overlapping, to the extent that it exists.)

(b) [5 Points] Argue that your algorithm runs in expected time \( \Theta(n \lg n) \) in general, but runs in expected time \( \Theta(n) \) when all of the intervals overlap (i.e., when there exists a value \( x \) such that \( x \in [a_i, b_i] \) for all \( i \)). Your algorithm should not be checking for this case explicitly; rather, its performance should naturally improve as the amount of overlap increases.

4. APPROX MAX CUT

[5 Points]

Given a graph \( G = (V, E) \) with \( n \) vertices and \( m \) edges, describe an algorithm that runs in \( O(n) \) times, and output a cut \( S \subseteq V \), such that the expected number of edges in the cut is \( \geq M/2 \), where \( M \) is the number of edges in the maximum cut, where the number of edges in the cut is \( |(S \times (V \setminus S)) \cap E| \).

5. MODIFIED PARTITION (Based on CLRS 7.4-6)

[5 Points]

Consider modifying the PARTITION procedure by randomly picking three elements from array \( A \) and partitioning about their median. Approximate the probability of getting at worst an \( \alpha \)-to-(1-\( \alpha \)) split, as a function of \( \alpha \) in the range 0 < \( \alpha \) < 1.

6. SORTING RANDOM INPUT

[10 Points]

Let \( a_1, \ldots, a_n \) be \( n \) real numbers chosen independently and uniformly from the range [0, 1].

- [5 Points] Describe an algorithm with an expected linear running time that sorts the numbers.
- [5 Points] Show that the linear running time is with high probability.

7. ESTIMATING QUANTITIES

[10 Points]

(a) [5 Points] Assume that you are given a function \( \text{RandBit} \) that returns a truly random bit. However, you do not know what is the probability \( p \) that \( \text{RandBit} \) returns 1. Describe an algorithm (and prove its correctness), as fast as possible, that receives parameters \( \varepsilon, \delta \), and outputs a number \( x \), such that with probability \( \geq 1 - \delta \) we have \( p \leq x \leq p + \varepsilon \). Namely, the program estimates the value of \( p \) “reliably”.

(b) [5 Points] Let \( G = (V, E) \) be a graph with \( n \) vertices and \( m \) edges. Assume that the only way you can know whether there is an edge between vertices \( u \) and \( v \) is to probe the graph \( G \) and ask whether there is an edge \( uv \in E \) in constant time. You are given parameters \( \varepsilon > 0 \) and \( \delta > 0 \). Describe an algorithm, as fast as possible, that output a number \( k \) which is a good estimate of the number of edges of \( G \). Namely, such that \( m \leq k \leq m + \varepsilon n^2 \) with probability larger than 1 - \( \delta \).
8. **Yeh, whatever.**
   Provide a sub-quadratic \( o(n^2) \) time deterministic algorithm for the nuts and bolts matching problem. Your solution should be self contained.

9. **Random Bits in a Treap**

   Let’s analyze the number of random bits needed to implement the operations of a treap. Suppose we pick a priority \( p_i \) at random from the unit interval. Then the binary representation of each \( p_i \) can be generated as a potentially infinite series of bits that are the outcome of unbiased coin flips. The idea is to generate only as many bits in this sequence as is necessary for resolving comparisons between different priorities. Suppose we have only generated some prefixes of the binary representations of the priorities of the elements in the treap \( T \). Now, while inserting an item \( y \), we compare its priority \( p_y \) to other’s priorities to determine how \( y \) should be rotated. While comparing \( p_y \) to some \( p_i \), if their current partial binary representation can resolve the comparison, then we are done. Otherwise, the have the same partial binary representations (upto the length of the shorter of the two) and we keep generating more bits for each until they first differ.

   (a) Compute a tight upper bound on the expected number of coin flips or random bits needed for a single priority comparison. (Note that during insertion every time we decide whether or not to perform a rotation, we perform a priority comparison. We are interested in the number of bits generated in such a single comparison.)

   (b) Generating bits one at a time like this is probably a bad idea in practice. Give a more practical scheme that generates the priorities in advance, using a small number of random bits, given an upper bound \( n \) on the treap size. Describe a scheme that works correctly with probability \( \geq 1 - n^{-c} \), where \( c \) is a prespecified constant.

10. **Majority Tree**

    Consider a uniform rooted tree of height \( h \) (every leaf is at distance \( h \) from the root). The root, as well as any internal node, has 3 children. Each leaf has a boolean value associated with it. Each internal node returns the value returned by the majority of its children. The evaluation problem consists of determining the value of the root; at each step, an algorithm can choose one leaf whose value it wishes to

    (a) [5 points] Describe a deterministic algorithm that runs in \( O(n) \) time, that computes the value of the tree, where \( n = 3^h \).

    (b) [10 points] Consider the recursive randomized algorithm that evaluates two subtrees of the root chosen at random. If the values returned disagree, it proceeds to evaluate the third sub-tree. Show the expected number of leaves read by the algorithm on any instance is at most \( n^{0.9} \).

    (c) [5 points] Show that for any deterministic algorithm, there is an instance (a set of boolean values for the leaves) that forces it to read all \( n = 3^h \) leaves. (hint: Consider an adversary argument, where you provide the algorithm with the minimal amount of information as it request bits from you. In particular, one can devise such an adversary algorithm.)

11. **A Game of Death**

    Death knocks on your door one cold blustery morning and challenges you to a game. Death knows that you are an algorithms student, so instead of the traditional game of chess, Death
presents you with a complete binary tree with $4^n$ leaves, each colored either black or white. There is a token at the root of the tree. To play the game, you and Death will take turns moving the token from its current node to one of its children. The game will end after $2n$ moves, when the token lands on a leaf. If the final leaf is black, you die; if it’s white, you will live forever. You move first, so Death gets the last turn.

You can decide whether it’s worth playing or not as follows. Imagine that the nodes at even levels (where it’s your turn) are OR gates, the nodes at odd levels (where it’s Death’s turn) are AND gates. Each gate gets its input from its children and passes its output to its parent. White and black stand for TRUE and FALSE. If the output at the top of the tree is TRUE, then you can win and live forever! If the output at the top of the tree is FALSE, you should challenge Death to a game of Twister instead.

(a) Describe and analyze a deterministic algorithm to determine whether or not you can win. [Hint: This is easy!]

(b) Unfortunately, Death won’t let you even look at every node in the tree. Describe a randomized algorithm that determines whether you can win in $\Theta(3^n)$ expected time. [Hint: Consider the case $n = 1$.]