Homework Exercise I

1-1 Consider the following the weighted version of MAX-SAT problem. Each clause has a positive weight and the goal is to maximize the weights of the satisfied clauses. Show that there is a truth assignment that satisfies clauses the sum of whose of weights is at least half of the total clause weight.

1-2 Suppose that we have \( n \) jobs to distribute among \( m \) processors. For simplicity, we assume that \( m \) divides \( n \). A job takes 1 step with probability \( p \) and \( k > 1 \) steps with probability \( 1 - p \). Use Chernoff bounds to determine upper and lower bounds (that hold with high probability) on when all jobs will be completed if we randomly assign exactly \( n/m \) jobs to each processor.

1-3 In this exercise we study the deterministic bit-fixing routing strategy, that is, the bit-fixing strategy where we do not first route each message to a random intermediate destination, but where we immediately route to the final destination.

(i) Suppose that \( \text{dest}(i) \) is obtained by taking the “complement” of the bitstring for \( i \): every 1 is replaced by a 0, and vice versa. Analyze the number of rounds it takes until all messages have reached their destination (in the deterministic bit-fixing strategy).

(ii) Now consider the case where \( \log n \) is even and the destinations are defined as follows. Let \( 2k = \log n \). Consider a processor \( i \) and let \( b_1 \ldots b_k b_{k+1} \ldots b_{2k} \) denote the binary representation of processor \( i \). Then \( \text{dest}(i) \) has binary representation \( b_{k+1} \ldots b_k b_1 \ldots b_k \) (that is, the first half and the second half of the binary representation are exchanged). Show that the deterministic bit-fixing strategy needs \( \Omega(\sqrt{n}) \) rounds before all messages have reached their destination.

1-4 Let \( N := \{1, \ldots, n\} \) and let \( f : N \times N \to \mathbb{R} \) be a function. Assume for simplicity that \( f(i,j) \neq f(i',j') \) if \( (i,j) \neq (i',j') \). We want to compute the value \( M := \min_{1 \leq i,j \leq n} f(i,j) \). This can of course easily be done in \( O(n^2) \) time, by simply evaluating all \( f(i,j) \)'s—we assume that evaluating \( f(i,j) \) takes \( O(1) \) time—and then finding the minimum of all these values. Now suppose we have a magic subroutine \( \text{Test}(i, x) \) available that returns \( \text{true} \) if \( \min_{1 \leq j \leq n} f(i,j) \leq x \) and \( \text{false} \) otherwise. The function \( \text{Test} \) runs in \( T^*(n) \) time.

(i) Describe a randomized incremental algorithm to compute \( M \) that runs in \( O(n \log n + nT^*(n)) \) expected time. Argue that your algorithm is correct and show that your algorithm runs in the required time. 

\( \text{Hint: Define } F(i) = \min_{1 \leq j \leq i} f(i,j) \text{ and observe that } M = \min_{1 \leq i \leq n} F(i). \)

(ii) Describe a randomized divide-and-conquer algorithm to compute \( M \) that runs in \( O(n \log n + nT^*(n)) \) expected time. Argue that your algorithm is correct and show that your algorithm runs in the required time.

(iii) Bonus question: you do not have to solve this question, but if you can solve it you get bonus points. Give a deterministic algorithm for this problem that runs in \( O(n \log n + nT^*(n)) \) time in the worst case.

1-5 In a skip list, each element in a list \( L_i \) is promoted to \( L_{i+1} \) with probability \( 1/2 \). Hence, if \( n_i \) denotes the number of elements in \( L_i \) then the expected number of elements in
\( \mathcal{L}_{i+1} \) is \( n_i/2 \). The actual number of elements in \( \mathcal{L}_{i+1} \) may deviate from this, of course, depending on the random choices made by the algorithm.

Suppose that, instead of promoting each element independently with probability 1/2, we let \( \mathcal{L}_{i+1} \) be a random subset of \( \mathcal{L}_i \) of size \( \lfloor n/2 \rfloor \). Notice that this would imply that the height of the skip list is guaranteed to be at most \( \lfloor \log n \rfloor \) in the worst case. Is this a good idea? Explain your answer briefly.

1-6 We want to store a set \( S \) of \( n \) elements, where each element \( x_i \in S \) has a key \( key[x_i] \), in a dictionary. Suppose that some elements are searched for more often than others. Let \( p_i \) denote the probability that we search for \( x_i \), and assume \( p_i \) is known for each \( x_i \). We want to use this knowledge to make the search time for elements with a high probability to be searched smaller than the search time for elements with a low probability to be searched.

Sketch a method for achieving this when the dictionary is implemented as a treap, and sketch a method for achieving this when the dictionary is implemented as skip list. (Keep your answer short: a few lines for the treap and a few lines for the skip list is sufficient: I do not expect an analysis of the search times that would result if your idea is used.)

1-7 You are hosting a web service. Whenever someone visits your website an algorithm called \( LV-\text{Alg} \) is executed. \( LV-\text{Alg} \) is a Las Vegas algorithm whose expected running time is 2 seconds.

(i) Give a bound on the probability that the actual running time of \( LV-\text{Alg} \) exceeds 1 hour.

(ii) What is the expected number of visitors before one of them has to wait 1 hour for \( LV-\text{Alg} \) to finish?

(iii) Now consider the following algorithm, which we call \( LV-\text{Alg-With-Restart} \):

Start running \( LV-\text{Alg} \). If the algorithm terminates within 4 seconds, then we are done and so we stop. But if the algorithm runs for 4 seconds without terminating then we abort the execution, and start all over again.

(Thus \( LV-\text{Alg} \) is repeated until we get a run terminating within 4 seconds.)

Give a bound on the probability that the running time of \( LV-\text{Alg-With-Restart} \) exceeds 2 minutes. (Assume that testing whether the algorithm runs for 4 seconds, aborting the execution, and restarting does not take any time.)

(iv) What is the expected number of visitors before one of them has to wait for more than 2 minutes if we use the strategy from (iii)?

1-8 Consider the following randomized variant of \( \text{InsertionSort} \), where we first compute a random permutation and then apply the normal InsertionSort algorithm:

**Algorithm** \( \text{Rand-InsertionSort}(A) \)

1. Sorts an array \( A[1..n] \) in non-decreasing order
2. \( \text{RandomPermutation}(A) \)
3. for \( j \leftarrow 2 \) to \( n \)
4. do \( key \leftarrow A[j]; \ i \leftarrow j - 1 \)
5. While $(i > 0)$ and $(A[i] > key)$
6. do $A[i + 1] \leftarrow A[i]$; $i \leftarrow i - 1$
7. $A[i + 1] \leftarrow key$

Analyze the expected number of comparisons made by the algorithm. (An asymptotic analysis is not sufficient: you should analyze the exact number of expected comparisons.)

1-9 Let $S$ be a set of $n$ elements, each with a key, and let $T$ be a treap for $S$ where the priorities of the elements have been chosen uniformly at random in the interval $[0, 1]$. (Assume all keys are distinct and all priorities are distinct.) Consider an element $x \in S$. Define the backbone of $x$ to be the path that starts at the left child of $x$, and then always goes to the right, until a leaf is encountered. In other words, the backbone of $x$ is the path from the left child of $x$ to the element with the largest key in the left subtree of $x$. (If $x$ does not have a left child, then its backbone is empty.) Prove that the expected length of the backbone of $x_k$ (element with rank $k$) is $1 - (1/k)$.

1-10 Given an $n \times n$ matrix $A$ all of whose entries are 0 or 1. Find a column vector $b \in \{-1, +1\}^n$ that minimizes $\|Ab\|_\infty$.

1-11 A graph $G = (V, E)$ is called bipartite if $V$ can be partitioned into two subsets $V_1$ and $V_2$ such that for any edge $(v, w) \in E$ we either have $v \in V_1, w \in V_2$ or $v \in V_2, w \in V_1$. (Thus, there should be no edges internal to $V_1$ and no edges internal to $V_2$.) Give a simple randomized algorithm with expected running time $O(|V| + |E|)$ that finds a subset $E' \subset E$ such that $|E'| > |E|/4$ and $G' = (V, E')$ is bipartite.