Advanced Algorithms, Fall 2009	**Problem Set 2**
Assigned: October 6	Due: October 20 before class

Guidelines If you write "I do not know how to solve this problem" then you get 1/4 of the score for the problem. If you write nonsense then you get 0.

As we are going to learn in this class, *time* and *space* are very valuable resources. Strive to give effective, compact solutions. Your solutions should touch on all the main points, but long calculations or discussions are not required nor sought.

Do not worry if you sometimes "do not get it." The problems are meant to stimulate you, not to overwork you. Unless specified otherwise, you can collaborate, but you must acknowledge all your collaborators in your solutions. I need your solutions on paper (not file). To hand them in: give it to me or slide them under my door West Village H (246).

Problem 1. Expected height of a skip list. Prove that the expected number of levels $E[h]$ in a random skip list of n elements is $E[h] = \theta(\log n)$. Note this requires both an upper and a lower bound.

Hint: For the upper bound first prove and then use the fact that any random variable X taking values in $\{0, 1, 2, \ldots\}$ satisfies $E[X] = \sum_{i \geq 1}^{\infty} \Pr[X \geq i]$. For the lower bound use Markov's inequality. You may also want to recall that $1 + x \leq e^x$ for every $x \in \mathbb{R}$.

Problem 2. Succinct data structure for partial sums. Show how to store an array $A[1..n]$ of n bits into $n + r$ bits of memory so that r is as small as possible and each partial sum $Sum(i) := \sum_{k=1}^{i} A[k] \in \{0, 1, \ldots, n\}$ can be computed by reading only $O(1)$ cells of $w := \log n$ bits. Memory is organized in cells of w bits each; you can assume that $\log n$ is an integer.

It is trivial to achieve $r = n \cdot \log n$.

For partial credit, show $r = O(n)$.

For full credit, show $r = o(n)$.

For extra credit, show $r = n/\log^2 n$.

For context, this data structure features as a component of many central succinct data structures, including those to store subsets. It can be shown that, if partial sums are computed by reading c cells, then the optimal space is $r = n/\log^{\Theta(c)} n$.

Problem 3. Existence of expanders. In this problem you will prove the existence of expanders that are good enough for the 1-bit-probe data structure for storing sets.

Prove that for every constant $\varepsilon > 0$ there are constants a, b such that for every N and K there is an expander graph $G : [N] \times [D] \to [M]$ with N vertices on the left, degree $D := a \cdot \log N$, and $M := b \cdot K \cdot \log N$ vertices on the right such that every set $S \subset [N]$ of size at most K has at least $|S|D(1 - \varepsilon)$ neighbors. (Hint: Choose the graph at random, and show that it is not an expander with probability < 1.)

For context, we point out that for the data structure application the degree of the expander is irrelevant, but it is not clear how to simplify the proof in this case.

Problem 4. CLRS 17-3: Amortized weight-balanced trees. Notes: For part (a) you can use the simple fact that there is a linear time and linear space algorithm that given a binary search tree outputs all its keys in a sorted array. You may want to interpret part (d) as follows: Show that there exists a constant c, depending on α, such that whenever we rebuild a tree with m nodes, the potential is at least the cost to rebuild the tree (which is Am by part (a) for an absolute constant A).