CSE 203 Homework 2
Randomized algorithms
Due Wednesday, March 10

For each of the algorithm problems, design as asymptotically efficient an algorithm as possible. Give a correctness argument (explanation, if it is relatively simple, or proof if not) and time analysis. You may use any well-known algorithm or data structure, or algorithm from the text or from class, as a subroutine without needing to provide details. The first four problems are 10 points each, the last 20 points for five parts.

Batch RSA: The signature verification algorithm takes a public key e, N, a message m, a signature s and verifies that: $s^e \equiv m \ mod \ N$. If so, we say s is a valid signature of m. The batch verification problem is: given e, N and n supposed signed messages $s_1, m_1, s_2, m_2, ...s_n, m_n$, are all of the s_i's valid signatures of the corresponding m_i ? Give a probabilistic algorithm that decides this problem with error probability at most ϵ and uses $O((n + \log e) \log 1/\epsilon)$ modular multiplications. (Hint: you just have to return "Yes, they are all valid", or "No, at least one is invalid". You don't have to find the set of invalid signatures.)

Max Cut for degree 3 graphs: Consider the Max Cut problem restricted to graphs where each node has at most 3 neighbors. Give the best poly-time probabilistic approximation algorithm you can (in terms of approximation ratio). If you can, go on to make your algorithm deterministic. (Hint: One approach is to consider a maximal matching in the graph. Your ratio should be at least 5/9.)

Popular elements: Let $a_1, ..a_n$ be a stream of not necessarily distinct elements from $\{0,1\}^k$. Say that x is *ϵ-popular* if $a_i = x$ for at least ϵn values of i. Give a randomized algorithm that uses memory $O(poly(k + \log n + \log(1/\epsilon))/\epsilon)$ that returns a set of strings so that with probability .99, the following two properties hold: every element returned is $\epsilon/2$-popular, and every ϵ-popular element is returned. (Hint: the memory requirement above allows you to store approximately $\log(1/\epsilon)/\epsilon$ elements from the stream, and associated counters. The

On-line makespan : Consider the makespan problem with m machines when restricted to items of either size 1 or 2. Call this problem 1, 2-makespan. In the on-line version, jobs arrive in an arbitrary order, and need to be immediately assigned to one of the machines.

- **a** Prove that no deterministic on-line algorithm for 1, 2 makespan has approximation ratio less than 3/2 .
- **b** Give and analyze a deterministic on-line 1, 2 makespan algorithm with approximation ratio 3/2.

c Remember that an algorithm has *competitiveratio* R if there is a constant B with $Cost(Algorithm(x)) \leq R * Cost(Opt(x)) + B$ for every x. Give a deterministic on-line $1, 2$ makespan algorithm with competitive ratio 1 (Hint: to make later parts easier, try proving $B = 1$; $B = 2$ should be immediate).

d A probabilistic algorithm has approximation ratio R if $E(Cost(Algorithm(x))) \leq RCost(Opt(x))$ for every instance x. Prove that no probabilistic on-line 1-2 makespan algorithm has approximation ratio less than $4/3$.

e Give and analyze a probabilistic on-line 1-2 makespan algorithm with approximation ratio $4/3$, i.e., better than any deterministic algorithm. (Hint: look at the algorithm in part c. What inputs are bad for this algorithm? Devise a second algorithm that is good for this type of input. The probabilistic algorithm picks one of the two deterministic algorithms, with some probability p, and the other with probability $1 - p$, to get the best average.)