Probability and Algorithms **Caltech CS150, Fall 2014**
Leonard J. Schulman, schulman@caltech.edu **TA: Nicholas Schiefer, nschiefer@caltech.edu**
Problem set 2 **Out W 15/Oct Due W 29/Oct in class**

Errors were corrected by email; fixed in this version

As announced in class, due to my travel, there will be no class on Oct 20,22 or 27. In addition to the make-up class we had in advance on Oct 10, there will be another on Friday Oct 31. I will hold my office hours as usual from 3:00-4:00 on Oct 27 provided my flight is on time.

1. *In this problem you'll need to recall basic material from Group Theory, but little more than the meaning of subgroups, and the theorem that the order of a subgroup divides the order of the group.*

   We are interested in testing whether a group is abelian, that is to say, whether $ab = ba$ for all $a, b \in G$. Although slightly reminiscent of the "testing associativity" problem we saw in class, the details here are entirely different, starting with how we are given the group: we are given *generators* $g_1, \ldots, g_k \in G$, and a black box that performs each of the following tasks in unit time: (a) Given two group elements, returns their product. (b) Given a group element, returns its inverse. (c) Given a group element, answers whether it is the identity. (You may suppose that group elements are simply identified by binary strings. Two different computations might give rise to different binary strings that actually refer to the same group element, which is why you need operation (c).)

   We do not have to check that the group axioms hold for the black box: the algorithm is permitted to fail if they do not.

   We can exhaustively test in $O(k^2)$ time that the generators commute, and this will resolve the question. But there is a way to test whether the group is abelian in time $O(k)$ using randomization:

   (a) Pick two elements of the group, a and b, as follows: first pick u.a.r. bits $a_1, \ldots, a_k, b_1, \ldots, b_k \in \{0, 1\}$, and then compute $a = g_1^{a_1} \cdots g_k^{a_k}$ and $b = g_1^{b_1} \cdots g_k^{b_k}$.

   (b) Compute ab and ba. If $ab = ba$ declare "G is abelian".

   If G is actually abelian, the algorithm will certainly succeed. In this exercise you are to show that if G is nonabelian, there is probability at most $3/4$ of the algorithm erring. (Naturally, this can be improved by repetition.)

   Hint:

   (a) Show that if H is a proper subgroup of G (that is, $H \neq G$), $h_1, h_2 \in H$ and $g \notin H$, then $h_1 g h_2 \notin H$.

   (b) Show that for a sampled as above and H a proper subgroup of G, $\Pr(a \in H) \leq 1/2$.

   (c) Apply these ideas to two subgroups: first, the *center* C of G, which is the set of elements that commute with all of G; second, the *centralizer* $C(a)$ of an element a you have sampled, which is the set of elements that commute with a.

2. Consider the following modification of the associativity-verification algorithm. Recall that in class we extended the operation on S to an operation on "the algebra over S with coefficients in $\mathbb{Z}/2$" (let's call that $\mathbb{S}_2$). Now let p be any prime, and extend the operation in a similar way to "the algebra over S with coefficients in $\mathbb{Z}/p$" (let's call that $\mathbb{S}_p$).

   Argue that for any prime p, the operation on $\mathbb{S}_p$ is associative if and only if the operation on S is.

   Now define $g : S^3 \to \mathbb{S}_p$, just as in class, by

$$g(a, b, c) = (a \circ b) \circ c - a \circ (b \circ c).$$

and extend it, just as in class, to $\mathbb{S}_p^3$.

Use the Schwartz-Zippel lemma to show that the same algorithm, performed over $\mathbb{S}_p$ for any prime $p > 3$, detects nonassociativity with a positive probability.

3. A *ranking* of a tournament is a permutation σ on the n "players" (vertices of the graph), and the "fitness" of the ranking is $\mathrm{fit}(T, \sigma) = \sum_{i<j} T(i,j)\mathrm{sign}(\sigma_i - \sigma_j)$.

   Show that there exists a tournament T such that $\max_\sigma \mathrm{fit}(T, \sigma) < n^{3/2} \log^{1/2} n$. That is to say, there exist tournaments in which even knowing the optimal ranking of the players does not give you a more than $o(1)$ advantage over a fair coin flip, in guessing the outcome of matches. Specifically the advantage over $1/2$ in predicting the outcome of a random match is $O(\sqrt{\frac{\log n}{n}})$.