Randomized Algorithms
Homework Set 1
Due Date: Dec 19th

Moni Naor

1. A tournament is a directed graph (V, E) on set of nodes V where for every $u, v \in V$ where $u \neq v$ we have that exactly one of the directed edges (u, v) or (v, u) is in E. For a subset $V' \subset V$ we will call the tournament (V', E') where for every $u, v \in V'$ we put the edge $(u, v) \in E'$ iff $(u, v) \in E$ the induced tournament on V'.

A **king** in a tournament is a node $v \in V$ where for every other node $u \in V$ there is a directed path from v to u of length at most two. That is, with at most two hops it is possible to get to any other node.

1. For $v \in V$ Let $D(v)$ be the set of nodes u where the edges (v, u) exist and $C(v)$ be those where (u, v) exist (i.e. v dominates all those in $D(v)$ and is dominated by those in $C(v)$).

   Show that if $w \in C(v)$ is a king in the tournament induced by $C(v)$, then w is also a king in the full tournament (V, E).

2. Argue that any tournament (V, E) contains at least one king.

3. Suggest a linear in $|V|$ expected time randomized algorithm for finding a king in a tournament. Assume that the tournament is in matrix form for and for $u, v \in V$ checking whether there is an edge (u, v) or (v, u) takes one operation.

2. The purpose of this question is to study randomized communication complexity in the private coins model. Read the paper "Lower bound for communication complexity with no public randomness" by Ben-Sasson and Maor[1]. You should use the ideas of the paper, but write the answers in your own language. Let the function $f\colon \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}^n$ be one without redundant rows or columns.

1. Prove that for any function $f(x, y)$ without redundant rows or columns the deterministic communication complexity is at least $\Omega(\log n)$.

2. Suppose that we have a randomized protocol π with private random strings r_A, r_B and communication $\lambda \ll \log n$ that *attempts* to compute the function $f(x, y)$ (π is not necessarily a good protocol for f). The question is about how efficient is it to find bad input to π,

[1] see https://eccc.weizmann.ac.il/report/2015/139/

We have adversary M who wishes to find bad inputs to the protocol π, i.e. a pair (x, y) where the two parties running π fail to compute correctly with probability at least ε. The adversary M has limited time - some function (perhaps very large) of λ and polynomial in n and assume that simulating the two parties on an input takes one unit of time.

Show that if the protocol π always uses at most λ bits of communication then there is an adversary M working in time polynomial in n and some function of λ that can find a bad input to the protocol.