Homework 3, Computational Complexity 2024

The deadline is 23:59 on Wednesday 4 December. Please submit your solutions on Moodle. Typing your solutions using LaTeX is strongly encouraged. The problems are meant to be worked on in groups of 2–3 students. Please submit only one writeup per team. You are strongly encouraged to solve these problems by yourself. If you must, you may use books or online resources to help solve homework problems, but you must credit all such sources in your writeup and you must never copy material verbatim.

1 For functions $f\colon \{0,1\}^n \to \{0,1\}$ and $g\colon \{0,1\}^m \to \{0,1\}$ define their composition $f \circ g$ as the function $\{0,1\}^{nm} \to \{0,1\}$ given by

$$(f \circ g)(x) \coloneqq f(g(x^1), \ldots, g(x^n)) \qquad \text{where} \qquad x = (x^1, \ldots, x^n) \in (\{0,1\}^m)^n.$$

(In the lecture, we discussed the special case of $f = \textsc{And}$ and $g = \textsc{Or}$.) Show that decision tree complexity D^{dt} behaves multiplicatively under composition:

$$\mathsf{D}^{\mathsf{dt}}(f \circ g) = \mathsf{D}^{\mathsf{dt}}(f) \cdot \mathsf{D}^{\mathsf{dt}}(g).$$

(*Hint: For the "$\geq$" direction, suppose we are given adversary strategies for f and g—how to design an adversary strategy for $f \circ g$?*)

Solution: We first show that $\mathsf{D}(f \circ g) \leq \mathsf{D}(f) \cdot \mathsf{D}(g)$. Fix the best decision tree A for solving f and B for solving g (i.e. A has depth $\mathsf{D}(f)$ and B has depth $\mathsf{D}(g)$). The following is a query algorithm that computes $f \circ g$ on input $(x^1, \ldots, x^n)$. Run the tree A and whenever it asks for the value of $g(x^i)$ for some $i \in [n]$, run B on x_i to get the value $g(x^i)$ and continue. Observe that the number of queries on any path is at most $\mathsf{D}(f) \cdot \mathsf{D}(g)$.

We now prove that $\mathsf{D}(f \circ g) \geq \mathsf{D}(f) \cdot \mathsf{D}(g)$. Let A be the best adversary for f and B be the best adversary for g (i.e. A can delay the resolution of f by $\mathsf{D}(f)$ queries and likewise for B and g). The following is an adversary for $f \circ g$:

1. Instantiate n copies $B_1, \ldots, B_n$ of adversary B.
2. Upon receiving query to bit x^i_j with $i \in [n]$ and $j \in [m]$,
3. If adversary B_i can still delay the query without fixing the value of $g(x^i)$, respond with the answer of B_i.
4. If adversary B_i cannot delay the fixing of the function anymore (i.e. $\mathsf{D}(g) - 1$ have already been made to that copy), then consult adversary A to decide how to fix the value of $g(x_i)$ as to delay the fixing of $f \circ g$ as much as possible.

Each of the B_i adversaries can answer $\mathsf{D}(g) - 1$ queries without fixing the value and the overall adversary A can delay fixing the value of the overall $f \circ g$ for $\mathsf{D}(f)$ copies, so that the overall adversary can handle $\geq \mathsf{D}(f) \cdot \mathsf{D}(g)$ queries.

2 Let us define a CNF formula F. We have a set of nodes $[n] = \{1, \ldots, n\}$ and each node $i \in [n]$ is associated with $\log n$ boolean variables $x^i \in \{0,1\}^{\log n}$. We identify the set $\{0,1\}^{\log n}$ with $[n]$ so that each x^i encodes a *pointer* to another node in $[n]$. Thus there are altogether $n \log n$ many variables $x = (x^1, \ldots, x^n)$. Each assignment x can be visualised as a directed graph $G_x = ([n], E)$ where $(i,j) \in E$ iff i points to j (that is, $x^i = j$). We add the following constraints to F:

1. First node points forward: $x^1 > 1$.
2. Every node points either to itself (selfloop) or forward: $x^i \geq i$ for all $i \in [n]$.
3. No node points forward to a selfloop: $(x^i > i) \to (x^{x^i} \neq x^i)$ for all $i \in [n]$.

Note that each constraint above involves only $O(\log n)$ variables and hence they can be encoded as a $O(\log n)$-width CNF of size polynomial in n. This defines the formula F.

First, show that F is unsatisfiable. Second, show that any tree-like resolution proof of F requires $\Omega(n)$ depth.

[***Update**: Turns out (thanks to Luca!) the formula F admits tree-like resolution refutations of polynomial size. (For additional fun, prove this.) Thus, F is an example showing that tree-like resolution proofs cannot be efficiently converted into* balanced trees*: there is a tree-like proof of size $S = \text{poly}(n)$ but no proof of depth $\log S = O(\log n)$.*]

Solution: To prove F is unsatisfiable, it is enough to show for every assignment x at least one constraint is unsatisfied. We find one such constraint as follows: First we check if $x^1 = 1$. If yes, then we are done. Otherwise, we start from $j = 1, i = x^1$ and repeat the following process, with the invariant that $x^j = i$: Each time, we compare x^i with i, there are three outcomes:

- If $x^i < i$, then the second constraint with respect to i is violated.
- If $x^i = i$, as $i = x^j, i > j$, the third constraint with respect to j is violated.
- We update $j \leftarrow i, i \leftarrow x^i$, and continue.

Since after each iteration, i is increasing, we can find an unsatisfied constraint after at most n rounds, as desired.

To show any tree-like resolution proof of F requires $\Omega(n)$ depth, it suffices to prove the search problem Search_F has decision tree depth at least $\Omega(n)$. To simplify our job, let us assume that each time one can get all the bits of x^i with unit cost. To prove the lower bound, we devise the following adversary strategy: We maintain a node i which records the end of the path which starts from 1 given by the partial query outcomes $x \in \{0,1,\star\}^n$ (where $\star$ means unqueried). Initially, $i = 1$. Each time, whenever x^j is queried, the adversary answers

$$x^j = \begin{cases} j & j \neq i \\ \min_{x^k = \star} k & j = i \end{cases}.$$

It remains to show that the opponent cannot determine the answer unless $n-1$ queries are made. In fact, first note that nodes point to itself can never be a solution. Furthermore, unqueried nodes may not be a solution since it can point to itself. Finally, for each node j on the path start from 1 given by the partial assignment, let $k = x^j$. Either $x^k \neq \star$ and $x^k > k$, so j is not a solution; or $x^k = \star$, note that $k < n$ by our choice of adversary, so k can point forwards, which implies that j may not be a solution.

3 Give a reduction from the *Set-Intersection* problem (to be discussed in Lecture 11) to show that the following two-party communication problem requires $\Omega(n)$ bits of (randomised) communication:

- Alice holds a graph $G_A = ([n], E_A)$;
- Bob holds a graph $G_B = ([n], E_B)$;
- Decide whether their union $G_A \cup G_B = ([n], E_A \cup E_B)$ contains a perfect matching.

Solution: The reduction works as follows: In the *Set-Intersection* problem, Alice holds an input $x \in \{0,1\}^n$, Bob holds an input $y \in \{0,1\}^n$. To decide if their inputs intersect, they first construct their respect graph G_A and G_B without communicating: They share the same vertex set $[2n] = \{(i,j) \mid i \in \{1,\dots,n\}, j \in \{0,1\}\}$. Then Alice constructs the edge set $E_A = \{((i,0),(i,1) \mid x_i = 0\}$, and similarly, Bob constructs $E_B = \{((i,0),(i,1)) \mid y_i = 0\}$. Observe that if x intersects with y, namely, there exists i s.t. $x_i = y_i = 1$, then $|E_A \cup E_B| \leq n-1$ so $G_A \cup G_B$ does not have a perfect matching. On the other hand, if for all $i \in [n]$, either $x_i = 0$ or $y_i = 0$, then the edge $((i,0),(i,1))$ is present in $E_A \cup E_B$, so $G_A \cup G_B$ has a perfect matching. To conclude, if Alice and Bob can solve this problem with $T(2n)$ bits of communication, then they can solve *Set-Intersection* with $T(2n)$ bits of communication. This implies that the problem in the statement has communication complexity $\Omega(n)$.