Homework 4

Date: Oct 5, 2021 — Algebra & Computation-F21
Instructor: Mrinal Kumar — Due: Oct 29, 2021, 5 pm

Instructions

- It is slightly preferred that you type your homeworks up in LaTeX. In case you turn in scans of handwritten notes, please make sure that they are legible.
- Discussion on the problems with other members of the class is permitted and to an extent, even encouraged. But, you *must* write the solutions on your own. You *must* also acknowledge any discussions you might have had with others separately for every problem.
- Please do not look up solutions on the internet or in other references. In case you use any sources outside the notes for this course, again properly acknowledge them.
- To get the most out of the problem sets, you are encouraged to think about the problems on your own before discussing them with others, consulting the references or looking at the hints (which some of the problems might have).

Problems

1. **(10 points)** We saw in one of the lectures that if a polynomial $f \in \mathbb{C}[x_1, x_2, \ldots, x_n]$ of degree d is computable by an arithmetic circuit of size s over $\mathbb{C}$, then there is a multioutput homogeneous circuit of size at most $O(sd^2)$ which computes all the homogeneous components of f. Show that for every $k \leq d$, there is a (not necessarily homogeneous) circuit of size at most $O(s \cdot k \cdot \mathsf{poly}(\log k))$ which computes the homogeneous component of degree k of f.

   The interesting regime of parameters for this is when d is much much larger than k. For instance, when $d = 2^{\Omega(s)}$, but $k = \mathsf{poly}(s)$.

2. **(10 points)** An arithmetic circuit is said to be a formula if the underlying graph of the circuit is a tree. If a polynomial f of degree at d can be computed by a formula of size s, show that for every homogeneous component f_i of f, there is a formula of size at most $O(sd)$ which computes f_i. You might need to assume that the underlying field is large enough for this.

3. **(10 points)** In the class, we saw a fast parallel algorithm to detect perfect matching in bipartite graphs via the determinant. In this question, your goal is to build upon those ideas to design an efficient parallel algorithm for the following problem: the input is a bipartite graph $G = (L, R, E)$, a set $S \subseteq E$ and a natural number k. Here, L, R are the two sets of vertices in the bipartition, $E \subseteq L \times R$ is the set of all edges in G and the set $S \subseteq E$ is a subset of edges of G. The goal is to detect whether G has a perfect matching consisting of exactly k edges from the set S.

4. **(20 points)** A polynomial $P(x_1, \ldots, x_n) \in \mathbb{C}[x_1, \ldots, x_n]$ is said to vanish with *multiplicity* at least k on input $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{C}^n$ if for all monomials $\mathbf{x^e}$ of degree at most $k-1$,

$$\frac{\partial P}{\partial \mathbf{x^e}}(\mathbf{a}) = 0\,.$$

The multiplicity of P at $\mathbf{a}$, denoted by $\mathsf{Mult}(P, \mathbf{a})$ is the largest non negative integer m, such that P vanishes on $\mathbf{a}$ with multiplicity at least m.

The goal of this problem is to prove the following higher order version of the Schwartz-Zippel Lemma that we saw in the class. Show that for an n variate *non-zero* polynomial $P \in \mathbb{C}[x_1, \ldots, x_n]$ of total degree at most d, and a set $S \subseteq \mathbb{C}$,

$$\sum_{\mathbf{a} \in S^n} \mathsf{Mult}(P, \mathbf{a}) \leq d \cdot |S|^{n-1} .$$