CS 473 ✧ Spring 2016

Homework 11

Solutions will be released on Tuesday, May 3, 2016.

This homework will not be graded.
However, material covered by this homework *may* appear on the final exam.

---

1. The ***linear arrangement*** problem asks, given an n-vertex directed graph as input, for an ordering $v_1, v_2, \ldots, v_n$ of the vertices that maximizes the number of forward edges: directed edges $v_i \to v_j$ such that $i < j$. Describe and analyze an efficient 2-approximation algorithm for this problem. (Solving this problem exactly is NP-hard.)

2. Let $G = (V, E)$ be an undirected graph, each of whose vertices is colored either red, green, or blue. An edge in G is *boring* if its endpoints have the same color, and *interesting* if its endpoints have different colors. The *most interesting 3-coloring* is the 3-coloring with the maximum number of interesting edges, or equivalently, with the fewest boring edges. Computing the most interesting 3-coloring is NP-hard, because the standard 3-coloring problem is a special case.

   (a) Let $wow(G)$ denote the number of interesting edges in the most interesting 3-coloring of G. Suppose we independently assign each vertex in G a *random* color from the set $\{\text{red}, \text{green}, \text{blue}\}$. Prove that the expected number of interesting edges is at least $\frac{2}{3} wow(G)$.

   (b) Prove that with high probability, the expected number of interesting edges is at least $\frac{1}{2} wow(G)$. *[Hint: Use Chebyshev's inequality. But wait... How do we know that we* ***can*** *use Chebyshev's inequality?]*

   (c) Let $zzz(G)$ denote the number of boring edges in the most interesting 3-coloring of a graph G. Prove that it is NP-hard to approximate $zzz(G)$ within a factor of $10^{10^{100}}$.

3. Suppose we want to schedule a give set of n jobs on on a machine containing a row of p identical processors. Our input consists of two arrays $duration[1..n]$ and $width[1..n]$. A valid schedule consists of two arrays $start[1..n]$ and $first[1..n]$ that satisfy the following constraints:

   - $start[j] \geq 0$ for all j.
   - The jth job runs on processors $first[j]$ through $first[j] + width[j] - 1$, starting at time $start[j]$ and ending at time $start[j] + duration[j]$.
   - No processor can run more than one job simultaneously.

   The *makespan* of a schedule is the largest finishing time: $\max_j(start[j] + duration[j])$. Our goal is to compute a valid schedule with the smallest possible makespan.

   (a) Prove that this scheduling problem is NP-hard.

(b) Describe a polynomial-time algorithm that computes a 3-approximation of the minimum makespan of the given set of jobs. That is, if the minimum makespan is M, your algorithm should compute a schedule with makespan at most $3M$. You may assume that p is a power of 2. *[Hint: Assume that p is a power of 2.]*

(c) Describe an algorithm that computes a 3-approximation of the minimum makespan of the given set of jobs ***in $O(n \log n)$ time***. Again, you may assume that p is a power of 2.