CS:3330 Approximation Algorithms Practice Problems
Spring 2017

1. Consider the "Shortest Interval first" greedy algorithm for the *Interval Scheduling* problem. In this algorithm, we repeatedly pick a shortest interval to include in our solution and as usual when an interval I is picked, then I and any overlapping intervals still present are deleted. The algorithm breaks ties arbitrarily; in other words, if there are multiple shortest intervals present, the algorithms picks one arbitrarily.

   (a) Show that this algorithm does *not* solve the Interval Scheduling problem. In other words, even though the algorithm returns a non-overlapping set of intervals, the set it returns need not be the largest possible set. (We have discussed a counterexample for this algorithm in class.)

   (b) Let A be the set of intervals returned by the algorithm for some input and let O be an optimal solution for this input. Prove that every interval in A overlaps at most two intervals in O.

   (c) Consider an arbitrary input and let A be the set of intervals returned by the algorithm for this input and let O be an optimal solution for this input. Now for each interval x in O, charge \$1 to an interval y in A that overlaps x. Note that y could be identical to x. Also, note that y has to exist; otherwise the greedy algorithm would have added x to the set A. Thus the number of dollars charged is exactly equal to $|O|$. Now answer the following questions: (i) what is the maximum number of dollars that an interval in A is charged? (ii) what does this tell us about the relative sizes of A and O? (Express your answer as an inequality connecting $|A|$ and $|O|$.), and (iii) what does this tell us about the "shortest interval first" algorithm being an approximation algorithm for Interval Scheduling?

2. The *Bin Packing* problem takes as input an infinite supply of *bins* $B_1, B_2, B_3, \ldots$, each bin of size 1 unit. We are also given n *items* $a_1, a_2, \ldots, a_n$ and each item a_j has a size s_j that is a real number in the interval $[0, 1]$. The Bin Packing problem seeks to find the smallest number of bins such that all n items can be packed into these bins.

   For example, suppose that we are given 4 items a_1, a_2, a_3 and a_4 of sizes 0.5, 0.4, 0.6, and 0.5 respectively. We could pack a_1 and a_2 in bin B_1 because $s_1 + s_2 = 0.9 \leq 1$. We could then pack a_3 into bin B_2, but we could not also add a_4 to bin B_2, because $s_3 + s_4 = 1.1 > 1$. So a_4 would have to be packed in bin B_3. This gives us a bin packing of the 4 items into three bins. An alternate way of packing items that would lead to the use of just two bins is to pack a_1 and a_4 into bin B_1 and a_2 and a_3 into bin B_2. This packing that uses only two bins is an optimal solution to the Bin Packing problem.

   The *First Fit* greedy algorithm processes items in the given order $a_1, a_2, \ldots, a_n$ and it considers the bins in the order $B_1, B_2, \ldots$. For each item a_j being processed, the algorithm packs a_j into the first bin that has space for it. It turns out that this very simple algorithm is a 2-approximation algorithm for Bin Packing. The following problems will help you prove this.

   (a) Suppose that the First Fit algorithm packs the given items into t bins. Prove that at most one of these bins is more than half-empty. Use this to deduce that the total size of the n input items is at least $(t-1)/2$.

   (b) Use what you showed in (a) to then show that if an optimal bin packing uses b^* bins, then the First Fit algorithm uses at most $2b^* + 1$ bins.