# Fall 2021 CS: 4310 Homework 5

1. For each $k = 2, 3, \ldots$, we define a graph $G_k$ as follows. Let $n = k!$. Start with a subset $U$ containing $n$ vertices labeled $u_1, u_2, \ldots, u_n$. We will add to the graph other vertices and edges so that the degree of each vertex in $U$ ends up being $k$. Now add to the graph, $k$ subsets of vertices $V_1, V_2, \ldots, V_k$, where $|V_j| = n/j$ for each $j$, $1 \leq j \leq n$. Now we will describe the edges incident on each vertex in $V_j$. Consider the $n/j$ vertices in $V_j$ in some order. Connect the first vertex in $V_j$ to vertices $u_1, u_2, \ldots, u_j$, then the second vertex in $V_j$ to $u_{j+1}, u_{j+2}, \ldots, u_{2j}$, then the third vertex in $V_j$ to $u_{2j+1}, u_{2j+2}, \ldots, u_{3j}$, and so on. Thus every vertex in $V_j$ has degree $j$ and every vertex in $U$ is connected to exactly one vertex in $V_j$, for any $j$.

   (a) Carefully draw and label the graph $G_3$.

   (b) Now consider the *Minimum Vertex Cover (MVC)* problem and a simple greedy algorithm for the problem. You've encountered MVC in Reading Response 5. The greedy algorithm I want you to consider, which I will call GreedyDegreeBased is this: repeatedly pick a vertex that covers the most, as yet uncovered edges. This algorithm continues until all edges are covered. Execute this algorithm on $G_3$ with the following tie-breaking rule: whenever there is a tie between two or more vertices, your algorithm should choose a vertex from $\cup_{j=1}^{k} V_j$, rather than a vertex from $U$. It does not matter how ties are broken between pairs of vertices in $\cup_{j=1}^{k} V_j$ or between pairs of vertices in $U$. What is the vertex cover produced by the algorithm in $G_3$? What is an optimal vertex cover for $G_3$?

   (c) Your friend claims that the GreedyDegreeBased algorithm is a 3-approximation algorithm. What is the smallest member of the family of graphs $G_k$ defined above that you could use as a counterexample to disprove your friend's claim? Justify your answer in 1-2 sentences.
   **Note:** Here it may help you to know that $\sum_{i=1}^{10} 1/i$ is approximately 2.93 and $\sum_{i=1}^{11} 1/i$ is approximately 3.02.

2. Consider the "Shortest Interval first" greedy algorithm for the *Interval Scheduling* problem. (The problem discussed in Section 4.2 "Scheduling Classes" is usually called 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. By solving this problem, you will be showing that the "Shortest Interval first" greedy algorithm is a 2-approximation algorithm.

   (a) 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?

3. 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 has half or more of its space empty. Use this to deduce that the total size of the $n$ input items is (strictly) more than $(t-1)/2$.

   (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 $2b^*$ bins.

---