CS340 Analysis of Algorithms
Fall 2023

Homework:	**10**	**Professor:**	**Dianna Xu**
Due Date:	**12/14/23**	**E-mail:**	`dxu@cs.brynmawr.edu`
Office:	**Park 203**	**URL:**	`http://cs.brynmawr.edu/cs340`

For full approximation algorithm design questions, you must provide a proof for the approximation factor. In other words, your correctness proof should contain the typical parts of termination, validity and now, approximation factor

1. You are given an undirected graph $G = (V, E)$ with integer weights on its edges (which may be positive, negative or zero). The *zero cycle problem* (ZC) decides whether there exists a simple cycle consisting of at least three edges whose total weight is zero. Show that ZC is NP-Complete.

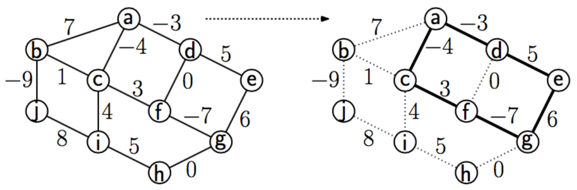


Figure 1: The zero cycle is shown in bold

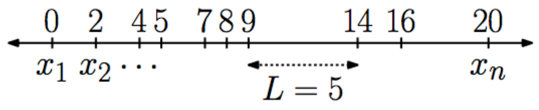
2. Recall the interval scheduling problem where given a set of $R = x_1, ..., x_n$ of n activities with associated start and finish times, we want to find the maximum number of nonoverlapping activities. Let's consider the approximation performance of some non-optimal heuristics:

1. Earliest Activity First (EAF). Give an example to show that not only is EAF not optimal, but its approximation ratio can be *arbitrarily* high.
2. Shortest Activity First (SAF): Prove that SAF has an approximation ratio of 2.

3. Write the IP for the Knapsack problem. This is just the formulation, you do not need to prove that it works via reduction.

4 Full write-up. The *bottleneck TSP* problem is the following: Find a cycle that visits each point exactly once, such that the *maximum* distance traveled between any two consecutive points is as small as possible.

Suppose you have a sequence of points $X = \langle x_1, ..., x_n \rangle$ from left to right along a line. The distance L between two points x_i and x_j is just their absolute difference $|x_i - x_j|$. Design a 2-approximation algorithm for the bottleneck TSP problem for points on a line.



5. Full write-up. You are an evil registrar. You are given a set of classes to schedule $C = c_1, ..., c_n$ and maximal subsets of classes $S_1, ..., S_m$ that could be scheduled at one of the given m time slots (already taking into account professors, room conflicts, etc.). You are given the popularity value p_i (the number of students who want to take the class) for each class c_i . Your goal is to make the students as miserable as possible, while still seeming to do your job. So you would like to choose at least one class to be scheduled at each of the time slots (i.e., from each of the sets S_j) such that the total popularity of the chosen classes is minimized. (This might mean that you schedule the same unpopular class for each timeslot, if that's allowed based on your given S_j subsets. In that case, this class's popularity will only count once to your total.) The maximum number of classes in any subset S_j is k (which is upper bounded by n). Give a polynomial-time k-approximation algorithm that finds a set of classes to schedule that is at most k times more popular than the least popular set of classes.

Please hand in your assignment electronically on Moodle.