# CS 473: Algorithms, Spring 2021
# HW 2 (due Wednesday, February 17th at 8pm)

This homework contains three problems. **Read the instructions for submitting homework on the course webpage**.

**Collaboration Policy:** For this home work, each student can work in a group with up to three members. Only one solution for each group needs to be submitted. Follow the submission instructions carefully.

1. **[10pts]** Given an undirected graph $G = (V, E)$ we defined its *square*, denoted by square$(G)$ as the graph $G' = (V, E')$ where $uv \in E$ iff there is a path of length at most 2 between $u$ and $v$ in $G$. That is, $uv \in E'$ if $uv \in E$ or if there is a node $w$ such that $uw, wv \in E$. In class we saw an algorithm for the maximum weight independent set problem in a tree $T = (V, E)$. Design and analyze an algorithm for the maximum weight independent set problem for the square of a given tree $T = (V, E)$.

2. You are given a sequence of numbers $a_1, a_2, \ldots, a_n$. You are playing a game against an opponent, where you both take turns to pick a number from either end of the sequence. The game ends when no number remains to be picked, and the one with the largest sum of numbers picked wins. For example, if the sequence is $3, 6, 1, 9, 5$ and it is your turn to play, then you can either choose 3 or 5, but no other intermediate number. Suppose you pick 5, then it is your opponent's turn, the sequence become $3, 6, 1, 9$, and they can choose either 3 or 9.

   Suppose you always start first, and your opponent always greedily picks the highest of the two available numbers.

   (a) **[3pts]** Prove that you should not also use the greedy strategy. That is, show that there is a game that you can win, but only if you do not follow the same greedy strategy.

   (b) **[7pts]** Describe and analyze an algorithm to determine the maximum total number that you can collect playing against the greedy opponent.

3. Consider the following bin packing problem. The input consists of $n$ items with sizes $w_1, w_2, \ldots, w_n$ and that needs to be packed into bins $B_1, B_2, \ldots, B_m$ of large enough sizes. The amount of a bin used is the sum of the sizes of the items packed in that bin. Given a packing of all the items into bins the *maximum load* is the maximum amount of any bin used. To find a packing that minimizes the maximum load is a well-known NP-hard problem.

   (a) **[10pts]** Consider the setting where there are only 3 distinct item sizes $\{s_1, s_2, s_3\}$. That is, $w_i \in \{s_1, s_2, s_3\}$ for $1 \le i \le n$. Describe a polynomial-time algorithm for this problem.

   (b) **[Extra Credit, 3pts]** Consider now the setting where there are $k$ distinct item sizes, where $k$ is some fixed constant. Again, describe a polynomial-time algorithm for this problem.