Worth: 10% **Due:** Thursday June 6, 11:59pm.

Remember to write your *full name*, and *student number*.
*Please make sure that the work you submit does not contain someone else's words or ideas. Then, to protect yourself, list on the front of your submission **every** source of information you used to complete this homework (other than your own lecture and tutorial notes, and materials available directly on the course webpage). For example, indicate clearly the **name** of every student with whom you had discussions, the **title** of every additional textbook you consulted, the **source** of every additional web document you used,* etc.

For each question, please write up detailed answers carefully. Make sure that you use notation and terminology correctly, and that you explain and justify what you are doing. Marks **will** be deducted for incorrect or ambiguous use of notation and terminology, and for making incorrect, unjustified, ambiguous, or vague claims in your solutions. In any of your answers you can use any algorithm we discussed in class without proving it solves the problem we discussed in class optimally. If we discussed the runtime of the algorithm you can also use that without reproving it. The same goes for any Lemma, Theorem or Fact we discussed in class.
Note that any algorithm you propose should have a polynomial run time (algorithms with exponential run time such as brute force do not earn any point).
You will receive 20% in each question if you leave it completely blank or say "I don't know".

---

[10] 1. Problem 5.2 (a) from the DPV textbook.

[10] 2. Consider a road (a long line segment with an eastern endpoint and a western endpoint) with houses scattered very sparsely along it. We want to place cell phone base stations at certain points along the road, so that every house is within four miles of one of the base stations. Give an efficient algorithm to solve this problem using as few base stations as possible. Justify the optimality of your algorithm (you don't need to provide a full proof here; explain the key idea).

[10] 3. Consider a connected graph G with n vertices and $n+6$ edges. Assume that all the edge costs are distinct. Design an algorithm with running time $O(n)$ that returns a minimum spanning tree of G.

[10] 4. Consider n symbols with frequencies $f_1, \cdots, f_n$. We want to identify the Huffman encoding of these symbols. What is the longest a codeword could possibly be? Support your claim by producing an example set of frequencies that results in this case.

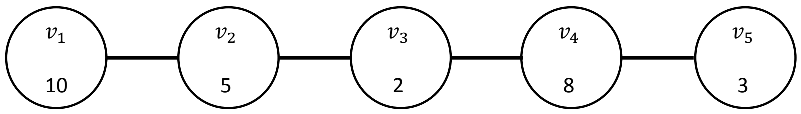
[10] 5. Show that if there is a character with frequency more than 40% (40% of the sum of the frequencies), then in the Huffman encoding scheme the existence of a codeword of length 1 is guaranteed.

[10] 6. Consider the Job Interval Scheduling Problem (JISP) that is an extension of ISP. Namely, there are n jobs $J = \{J_1, \cdots, J_n\}$. Each job $J_i = (s_i, f_i, \kappa_i)$ has a start time s_i, finish time f_i, and a class κ_i. In this problem, we say jobs are compatible if (1) they do not intersect, and (2) no two jobs belong to the same class (at most we choose one job from each class). The goal is to identify the maximum *number* of compatible jobs; i.e. identify a subset $S \subseteq J$ such that (1) S is compatible, and (2) $|S|$ is maximized. This problem is NP-hard that means there is no polynomial time algorithm that can identify the optimal solution.

Show that the greedy EFT algorithm (namely sort jobs by the earliest finish time and add them to S if they are compatible with the previously added jobs in S) provides a 2-factor approximation for JISP. In other words, show that if the result of EFT is S and the optimal solution is OPT, then $|OPT| \leqslant 2 \times |S|$.

[**Hint:** Use the charging proof technique. Show that at most two jobs in OPT can be mapped to a single job in S.]

[10] 7. We call a graph $G = (V, E)$ a *path* if its nodes can be written as $v_1, \cdots, v_n$ with an edge between v_i and v_j if and only if the numbers i and j differ by exactly 1. Each node v_i is associated with a positive integer weight w_i. For example, the following graph is a path with 5 nodes $v_1, \cdots, v_5$ with weights $10, 5, 2, 8, 3$ (respectively).



A subset of nodes is called an *independent set* if no two nodes in that subset are joined by an edge. Here, the subset $\{v_1, v_3\}$ is an independent set. Similarly the set $\{v_2, v_4\}$ and the set $\{v_1, v_3, v_5\}$ are independent sets. However, the set $\{v_2, v_3\}$ is not an independent set.

Give a dynamic programming algorithm that takes an n-node path G with weights and returns an independent set of maximum total weight (total weight is the sum of the weight of all nodes in that independent set). The running time should be polynomial in n, independent of the values of the weights.

Note that in the aforementioned graph, the independent set with maximum total weight is $\{v_1, v_4\}$. The total weight for this set is 18.