# EC 504
# Spring, 2022
# HW 8

1. (16 pts) Answer True or False to each of the questions below, **AND YOU MUST PROVIDE SOME FORM OF EXPLANATION (brief is OK)**. Each question is worth 2 points. Answer true only if it is true as stated, with no additional assumptions.

   (a) There are instances of the integer knapsack problem which can be solved in polynomial time by a greedy algorithm.

   **Solution:** Yes. Simplest one is when all of the tasks have unit capacity requirement.

   (b) Every problem in class $P$ is also in class $NP$.

   **Solution:** True; it is the reverse that is in question.

   (c) Every problem in class $P$ is can be reduced in polynomial time to a problem that is $NP$-complete.

   **Solution:** True. Indeed, this can be done for every problem in class $NP$.

   (d) The problem of finding a shortest path in a graph can be polynomially reduced to an instance of the integer knapsack problem.

   **Solution:** Let's interpret this as the decision problem associated with the shortest path problem in a graph. Then, this is true, since the integer knapsack problem is NP-complete and the shortest path decision problem is in class NP.

   (e) Consider the class of integer knapsack problems where the maximum knapsack capacity is bounded by 1000. The decision problems associated with this class belong to class $P$.

   **Solution:** True. The algorithm provided in class solves this in time $O(NT)$ for $N$ tasks, capacity $T$. If you fix $T$, the algorithm is polynomial in $N$.

   (f) Assume that there is a decision problem which can be described in an input with $n$ bits. Assume that the worst-case instance of this decision problem of size $n$ bits requires $O(2^n)$ operations to solve. Then, the decision problem belongs to class NP.

   **Solution:** False. It might be in NP, but there is no certification that a solution can be evaluated in polynomial time.

   (g) The class NP consists of all decision problems which cannot be solved in time $O(n^k)$ for some positive integer $k$, where $n$ is the size of the problem instance description in bits.

   **Solution:** False. Plenty of decision problems in NP can be solved in linear time, since it includes P.

   (h) Finding the minimum-distance tour in a traveling salesperson problem is an NP-Hard problem.

   **Solution:** True. Indeed, it is an optimization problem, which is equivalent to an NP-complete decision problem. Thus, it is NP-Hard.

2. (20 pts) A wide range of questions can be asked about social networks: graphs whose nodes represent people, with edges joining pairs who know one another.

   Consider a small unnamed college (X), where the Office of Student Life decides to make use of some social network algorithms. X, they reason, is one of the best places for making friendships that last a lifetime; they want to make sure the students are taking as much advantage of this opportunity as possible. So to facilitate new interactions, they decide to invite selected groups of students to a series of "ice-breaking events."

   To avoid the situation where people talk only to their friends, they would like to invite a set of people to the event with the property that no two know each other. Naturally these restrictions impose some limits on how large an ice-breaking event one can have. Thus, the Ice-breaking Event problem is the following. Given a set of students, and a list of all pairs of students who know one another, is there a set $S$ of $k$ students such that no two students in $S$ know each other?

(a) Show that this problem is in **NP**.

**Solution:** We need to show that, if given a candidate solution, we can evaluate whether the answer is true or not in polynomial time. A candidate solution is a set $S$ of $k$ students, that supposedly do not know each other. Verifying that they indeed do not know each other can be done in time proportional to $O(|V|^2)$, where $V$ is the set of students, as we simply need to verify that, for any $i, j \in S$, the edge $\{i, j\}$ is not in the edges $E$ of the graph. Hence, the problem is in NP.

(b) Suppose the social network graph is a line (lots of strangers here, and no student knows more than two other students!) Can you describe a polynomial time algorithm for solving the ice-breaking event problem for a line of $n$ students?

**Solution:** The simplest algorithm is to start at the beginning of the line, pick that person, and then pick every other person after that in the line. It is guaranteed to give a set of size $\lceil \frac{n}{2} \rceil$, which is also an upper bound on the size of the maximum number of people that do not know each other. If $\lceil \frac{n}{2} \rceil \geq k$, the answer is yes. Otherwise, the answer is no. This algorithm is $O(1)$.

(c) Suppose the social network graph is a tree. Note following observation: if person $k$ is a leaf in the tree, then there exists a maximum cardinality ice-breaking event that includes person $k$. The reason is that, if it were not, then a maximum cardinality set must include the parent vertex of that leaf (or else adding that leaf increases cardinality). However, one can exchange that leaf for its parent and still have a valid ice-breaking event of the same cardinality. Use this fact to develop a polynomial algorithm for finding the maximum size of an ice-breaking event when the social network is a tree.

**Solution:**

Following the hint, initialize $S$ by adding all the vertices as the start of the ice-breaking set that are leaves in the tree. That is, all vertices that have only one neighbor. Now, remove all these leaves from the tree, and their neighbors.

You are now left with a forest of trees (could be a single tree). Note that every vertex in those trees is not a neighbor of any of the vertices in the ice-breaking set so far.

We can now iterate on this idea, and add vertices in the forest that continue to preserve the ice-breaking property. For each tree in the forest, if it has 3 or more vertices, the tree will have some vertices with degree greater than one. In this case, add all the leaves (degree 1 vertices) in the tree to the ice-breaking set, and remove all the leaves and their neighbors, which result in smaller trees to add to the forest. If the tree has either one or two vertices, add one of the remaining vertices to the ice-breaking set and delete the remaining tree from the forest.

Note that this algorithm terminates, and will generate the largest possible ice-breaking set.

(d) Show that, for general undirected graphs, the ice-breaking problem above is NP-complete.

**Solution:** With a little bit of thought, you can show that this problem is equivalent to the Independent Set problem. That is, Independent Set $=_P$ Ice-breaker Problem. Since Independent Set is NP-complete, so is Ice-breaker.

3. (20 pts) You are consulting for a trucking company that does a large amount of business shipping packages between New York and Boston. The volume is high enough that they have to send a number of trucks each day between the two locations. Trucks have a fixed limit $W$ on the maximum amount of weight they are allowed to carry.

Boxes arrive to the New York station one-by-one, and each package $i$ has a weight $w_i$. At the moment the company is using a simple greedy algorithm for packing: they pack boxes in the order they arrive, and whenever the next box does not fit, they send the truck on its way.

Assume that we have a set of $n$ packages such that the weight of package $i$ is $w_i$, and each truck has a limit $W$ on the maximum amount of weight. You can assume each $w_i$ and $W$ are integers. The goal is to minimize the number of trucks that are needed in order to carry all the packages.

(a) Let us consider the decision version of the problem as follows: given a set of packages with weights $w_1$, ..., $w_n$, and a weight limit for each truck $W$, determine whether or not we can fit all the packages into $k$ trucks. Show that this problem is in NP.

**Solution:** Given a candidate solution, which maps packages into $k$ trucks, we can verify that the packages mapped into each of the trucks have total weight less than or equal to $W$, easily done in polynomial time. Hence, the problem is in NP.

(b) Assume that $W_1 = \sum_{i=1}^{n} w_i \leq 2W$. Define an extra package with weight $w_{n+1} = 2W - W_1$. Show that the subset sum problem (which is NP-complete) can be reduced in polynomial time to an instance of the 2 truck packing problem, where we must decide whether the $n+1$ packages fit into 2 trucks. This shows truck packing is NP-complete.

**Solution:**

We will show in general that SubsetSum $\leq_P$ TruckPacking. Thus, we are given an arbitrary instance of subset sum—namely a set of weights $w_1, ..., w_n$ and a target weight $W$, and we wish to find a subset $S$ whose sum of weights exactly equals $W$, which can then be used to solve the truck packing problem. To do this, let $W_1 = \sum_{i=1}^{n} w_i$ be the sum of all weights in the subset sum instance. Assume that $W_1 \leq 2W$. Let's create $n+1$ packages for the truck packing problem, with the first $n$ having weights $w_i$, and the last one with weight $2W - W_1$. Note that this reduction is of polynomial complexity (indeed, it is done in linear time).

Note that the only way to pack the $n+1$ items into 2 trucks of size $W$ is to have a subset of items with total weight equal to $W$. Hence, obtaining a solution to TruckPacking with 2 trucks will obtain a solution to the SubsetSum problem.

What if $W_1 > 2W$? Define $W_2 = W_1 - W$. The problem of finding a subset sum equal to $W$ is equivalent to finding a subset sum equal to $W_2$ (it just finds the complement of the original subset.) Now, note that $W_1 \leq 2W_2$, so this problem is the same as the above case.

Hence, SubsetSum $\leq_P$ TruckPacking, which shows that TruckPacking is NP-complete.

(c) Show that the number of trucks used by the greedy algorithm is within a factor of 2 of the minimum possible number, for any set of weights and any value of $W$.

**Solution:** Suppose that the number of trucks $k$ used by the greedy algorithm is even. We can imagine grouping trucks into pairs, i.e., the first two trucks and then the next two trucks, etc. For each pair of trucks, the total sum of the weights of the packages in that pair must be greater than $W$; this is because the greedy algorithm starts packing a new truck whenever adding a package would exceed the weight limit W for the current truck, so the sum of the weights of the new package (which will go into the second truck) plus whatever is in the first truck.

Consequently, on average each truck must be at least half full. In the best case, an optimal solution would fully pack every truck, which implies that the number of trucks required by the greedy method is no worse than twice that required by the optimal method. In the case that the greedy algorithm uses an odd number of trucks, then a similar argument can be used to establish that if the number of trucks is $2k+1$, where $k$ is an integer, then the optimum solution uses at least $k$ trucks for the first $2k$ greedy trucks, plus one more for the last truck, so it uses $k+1$ trucks, also establishing the bound.

4. (20 pts) Consider the weighted, undirected graph in Figure 1 below. Assume the real graph is dense, and all the missing edges have length 2, so Figure 1 is only showing the length 1 edges. Thus, the edges between nodes 2 and 3, and between nodes 1 and 4, have length 2.

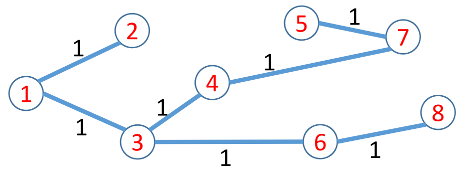

Figure 1: Figure for Problem 1

(a) What is the weight of the minimum spanning tree for this graph?

**Solution:** The figure is the minimum spanning tree, weight 7. Any other spanning tree would use an edge of weight 2, not shown in the graph, and have higher weight.

(b) Show that the distances in the arcs in this graph satisfy the metric property: $d(i, j) \leq d(i, k) + d(k, j)$ for any $i, j, k$ nodes.

**Solution:** We know $d(i, j) \leq 2$, and $d(i, k), d(k, j) \geq 1$. Hence, the arcs in the network satisfy the metric property.

(c) Given the unique minimum spanning tree in this graph, double the edges in the spanning tree and construct an Euler tour that visits all the vertices, albeit more than once, starting in vertex 1 and selecting the edges at branches that goes to the lowest numbered node.

**Solution:** The nodes will be visited in order 1, 2, 1, 3, 4, 7, 5, 7, 4, 3, 6, 8, 6, 3, 1.

(d) From this Euler tour, form a Traveling Salesperson Tour by skipping vertices that have already been visited. What is the length of this tour?

**Solution:** The tour is 1, 2, 3, 4, 7, 5, 6, 8, 1. The length is 11.

(e) Is the above tour optimal? Explain why or why not.

**Solution:** Not optimal, as there are tours of length 10: 1-2-5-7-4-3-8-6-1. You can find this tour using the Christofides heuristic, as the odd-degree vertices in the spanning tree are 2, 5, 3, and 8. The smallest weight matching is any matching, so let's randomly pick 2-5 and 3-8 as edges to add.

Now, let's do an Euler tour starting from 1 in this augmented graph with every vertex of even degree. The tour is 1, 2, 5, 7, 4, 3, 8, 6, 3, 1. Now, skip vertices that have already been visited to get the tour 1, 2, 5, 7, 4, 3, 8, 6, 1.