

1. We are given an undirected graph $G = (E, V)$, as well as an assignment of each edge to a type, where there are t different types. Furthermore, we are given a sequence of t type targets c_1, \dots, c_t . We say that a subset of edges $S \subseteq E$ *meets the type targets* if for $1 \leq i \leq t$, S has exactly c_i edges of type i .
 - (a) Say we are also given a weight for each edge. Describe a greedy algorithm that finds the maximum weight subset S that meets the type targets, if such a subset exists, and outputs “none” if no such subset exists. Use matroid theory to prove that your algorithm works for any input.
 - (b) Say that we are no longer given weights for the edges. Our goal now is to find a subset S of the edges such that S meets the type targets and (S, V) is a connected graph. Prove that we can determine whether or not such a subset S exists in polynomial time. Note: you don’t have to describe the algorithm that does this - you only need to prove that such an algorithm exists.

2. For many optimization problems of the generic form based on subset systems, there can be multiple solutions that achieve the optimal value.
 - (a) Prove that for any subset system (E, I) that is a matroid, if the elements of E are assigned distinct weights (i.e., no two weights are equal), then the maximum weight subset in I is unique.
 - (b) Now consider an assignment of weights such that there is more than one subset in I that achieves the maximum weight. From part (a), we know that such an assignment must have weights that are not distinct. Thus, when we sort the elements of E by non-increasing weight at the start of the greedy algorithm, there are multiple sorted orders that can result, depending on how we break ties. In fact, how we break ties can effect which of the optimal solutions the greedy algorithm returns.
 Prove that for every subset $i \in I$ that achieves the maximum weight, there is some way to break ties between elements of E with the same weight, such that the greedy algorithm returns the solution i .

3. In this problem, your task is to design polynomial time dynamic programming algorithms for special cases of two problems that, if we do not restrict them to special cases, are NP-Complete. (We shall cover NP-Completeness later in the course; basically, if a problem is NP-Complete, then we do not expect there to be an efficient algorithm that solves the problem in general.)
 - (a) An independent set in a graph $G = (V, E)$ is a subset $S \subseteq V$ such that no two vertices in S are connected by an edge of E . Say we are given an undirected graph G that consists of a single path. In other words, we can number the vertices v_1, \dots, v_n in such a way that for $1 < i < n$, v_i is connected to v_{i-1} and v_{i+1} , but no other vertices. We are also given an assignment of weights to the vertices, and wish to find the maximum weight independent set in G .
 Demonstrate that the greedy algorithm does not solve this problem. Provide a dynamic programming algorithm for this problem, and prove that your algorithm returns the correct answer in time that is polynomial in the number of vertices in G .
 - (b) In the heaviest path problem, we are given a directed and weighted graph and two vertices s and t . Our task is to find the simple path of edges from s to t of largest total weight. (A simple path

does not repeat any vertices.) Say we are given a directed, weighted graph and a sequence of the n vertices of the graph v_1, \dots, v_n , such that the graph has the property that if there is an edge from v_i to v_j , then $j > i$. Describe an algorithm that finds the heaviest path from v_1 to v_n .

Your algorithm should run in time polynomial in n (and thus independent of the cost of the edges). If there is no path from v_1 to v_n , your algorithm should return “no path”.

4. Suppose that presidential candidate Jerry Burry of the Demoblican political party has a series of public events he would like to attend. Unfortunately, due to travel time between events, he will not be able to make it to all of them. However, he would like to maximize the number of events that he is able to attend.

- (a) To start with, let’s say that the events all occur along a single highway. We can measure positions along the highway in terms of their distance from the beginning of the highway, and let’s assume that all of the events occur exactly at the mile markers of the highway (i.e., at exactly integral mile distances from the start of the highway). There is exactly one event per hour, and Jerry gets credit for the event if he is present at the start of that hour. Unfortunately, due to the many friends that travel with him, he is only able to move from one event to another at a rate of one mile per hour.

Consider the following problem. As input, we are given a sequence of n event locations e_1, e_2, \dots, e_n , where each e_i is a valid event location along the highway, and event e_i occurs at the i th hour of the campaign. Design an algorithm that computes the maximum number of events that Jerry can attend, assuming that he starts at the beginning of the highway.

For example, if the input sequence is $\{1, 3, 2, 4, 5, 4\}$, the value to be computed would be 4, since Jerry can schedule his travel so that he can attend the first event and the last three. Note that he could travel to the third event, but by doing so he would miss the fourth and fifth events.

Your algorithm should run in time that is polynomial in the size of the input. Describe how to use your algorithm to also compute the actual events that Jerry will travel to.

- (b) Let’s consider some generalizations to the problem. Assume now that the event locations are specified by two integer coordinates, instead of one. For example, this would represent the scenario when Jerry is campaigning in Manhattan: the events take place on street corners, the first coordinate represents the street number, and the second coordinate represents the avenue number. Jerry can still only move at a rate of one mile an hour, and he can only change his location with respect to one coordinate during any hour. In the Manhattan example, this would correspond to him only being able to travel along a street or along an avenue during any hour.

In addition to the more complicated coordinate system, assume that each event has a weight associated with it. Design an algorithm that computes the maximum weight set of events that Jerry can attend.