Basic problems:

1. Give a dynamic programming algorithm for the shortest realiable path problem:
   \[\text{Input:} \text{ A graph } G \text{ with lengths on the edges, two vertices } s \text{ and } t \text{ of } G, \text{ and a nonnegative integer } k.\]
   \[\text{Output:} \text{ A shortest path from } s \text{ to } t \text{ that uses at most } k \text{ edges, or assert that no such path exists.}\]

2. [Dasgupta et al., Ex. 6.1] A contiguous subsequence of a list \( S \) is a subsequence made up of consecutive elements of \( S \). For example, if \( S \) is
   \[5, 15, -30, 10, -5, 40, 10\]
   then \( 15, -30, 10 \) is a contiguous subsequence but \( 5, 15, 40 \) is not. Give a linear-time algorithm for the following task:
   \[\text{Input:} \text{ A list of numbers, } a_1, a_2, \ldots, a_n.\]
   \[\text{Output:} \text{ A contiguous subsequence of } a_1, a_2, \ldots, a_n \text{ that has the maximum possible sum (the sum of an empty subsequence is zero).}\]
   
   For the preceding example, the output is \( 10, -5, 40, 10 \), with a sum of 55.
   \[\text{Hint:} \text{ For each } 1 \leq j \leq n, \text{ consider contiguous subsequences that end at position } j.\]

3. [Dasgupta et al., Ex. 6.2] You are going on a long trip. You start on the road at mile post 0. Along the way there are \( n \) hotels, at mile posts \( a_1 < a_2 < \cdots < a_n \), where each \( a_i \) is measured from the starting point. The only places you are allowed to stop are at these hotels, but you can choose which of the hotels you stop at. You must stop at the final hotel (at distance \( a_n \)), which is your destination.
   
   You’d ideally like to travel 200 miles a day, but this may not be possible (depending on the spacing of the hotels). If you travel \( x \) miles during a day, the penalty for that day is \( (200 - x)^2 \). You want to plan your trip so as to minimize the total penalty—that is, the sum, over all travel days, of the daily penalties.

   Give an efficient algorithm that determines the optimal sequence of hotels at which to stop.

4. Prove the following results (cf. Lemma 2 in Lecture 10):
   (i) \( IS \leq^p CLIQUE \),
   (ii) \( CLIQUE \leq^p VC \).

Advanced problems:

5. [Dasgupta et al., Ex. 6.21] A vertex cover of a graph \( G = (V, E) \) is a subset of vertices \( S \subseteq V \) that includes at least one endpoint of every edge in \( E \). Give a linear-time algorithm for the following task:
Input: A tree \( T = (V,E) \).
Output: The size of a smallest vertex cover of \( T \).

For instance, in the following tree, the possible vertex include \{A, B, C, D, E, F, G\} and \{A, C, D, F\} but not \{C, E, F\}. The size of a smallest vertex cover is 3. Indeed, \{B, E, G\} is such a vertex cover, and no set of two vertices is a vertex cover.

(Hint: Have a look at the dynamic programming algorithm for independent sets on trees presented in Lecture 11.)

6. Prove the following basic facts about polynomial-time reducibility and complexity classes:

   (i) If \( S \leq_P T \) and \( T \leq_P U \), then \( S \leq_P U \).
   (ii) If \( S \leq_P T \) and \( T \in \text{P} \), then \( S \in \text{P} \).
   (iii) Let \( T \) be an \text{NP}-complete problem. If \( T \in \text{P} \), then \( \text{P} = \text{NP} \).
   (iv) Let \( S \) be some \text{NP}-complete problem, \( T \in \text{NP} \) and \( S \leq_P T \). Then also \( T \) is \text{NP}-complete.