The “Classroom Problems” will be discussed and solved ex tempore at the tutorials in the same week as this problem set is handed out. These are intended to help you in getting started with the material, and will not yield any course credit.

The “Homework Problems” you should solve at home, and attend one of the tutorial sessions in the following week, prepared to present your solutions at class if requested. You earn one tutorial credit point (= 0.33 course points, for a maximum total of 6) for each of these problems that you have solved, and marked as solved at the tutorial session.

The “Demonstration Problems” illustrate further interesting aspects of the material. These will be discussed at the tutorials in the same week as the Homework Problems (time permitting) and printed solutions will be provided. You are encouraged to work on these, but no course credit is given for them.

Note: The bonus point deadline for batches P4 and P5 of computerised programming assignments is Sun 12 Nov (cf. MyCourses/Programming assignments). Hence there are no lectures or tutorials on the course in the preceding week (Week 45 = 6–10 Nov). The Classroom Problems from this problem set are discussed at the tutorials in Week 44 and the Homework Problems in Week 46. Instead of classroom teaching, there are again “walk-in clinic” sessions to assist you in the programming assignments, in case you get stuck in working on them. These take place on Mon 6 Nov, 4–5 p.m., and Fri 10 Nov, 11–12 p.m., in Meeting Room B226 of the CS building.

Classroom problems:

1. Find an optimal (= maximum-value) selection of items for a knapsack with weight limit \( W = 10 \), when the item types are:

<table>
<thead>
<tr>
<th>Type</th>
<th>Weight</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>7</td>
</tr>
</tbody>
</table>

You may select several items of the same type. Justify your selection.

2. Prove the following results (cf. Lemma 2 in Lecture 10):

   (i) \( IS \leq_p CLIQUE \),
   (ii) \( CLIQUE \leq_p VC \).

Homework problems:

3. [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:
Input: A list of numbers, \( a_1, a_2, \ldots, a_n \).
Output: A contiguous subsequence of \( a_1, a_2, \ldots, a_n \) 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.

Hint: For each \( 1 \leq j \leq n \), consider contiguous subsequences that end at position \( j \).

4. [Dasgupta et al., Ex. 6.22] Give an algorithm with running time \( O(nt) \) for the following task.

Input: A list of \( n \) positive integers \( a_1, a_2, \ldots, a_n \) and a positive integer \( t \).
Question: Decide (output “yes” or “no”) whether there is a subset of the \( a_i \)'s whose sum is equal to \( t \). Each \( a_i \) may be used at most once.

5. 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.

Demonstration problems:

6. [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.)