
We want to place wind turbines along a road to produce energy. Due to geographical reasons \( n \) different positions are possible, but laws prescribe that the distance between two wind turbines has to be at least \( D \). The possible positions \( d_1, \ldots, d_n \) are given as coordinates on a line, where the leftmost position has the coordinate value 0. The distance between the \( i \)-th possible position and the first possible position is \( d_i \). Hence, we have \( d_1 = 0 \) and \( d_i < d_{i+1} \) for each \( i \in \{1, \ldots, n - 1\} \). When a wind turbine is installed at position \( i \), it produces energy \( e_i > 0 \). The task is to find a positioning of wind turbines that maximizes the total energy yield.

\[
\begin{array}{cccccc}
  d_1=0 & d_2=4 & d_3=6 & d_4=10 & d_5=13 \\
  e_1=6 & e_2=9 & e_3=5 & e_4=15 & e_5=11 \\
\end{array}
\]

\[
\begin{array}{cccccc}
  \times & \times & \times & \times & \times \\
  1 & 2 & 3 & 4 & 5 \\
\end{array}
\]

\[D=5\]

\[D=5\]

Example: The image above shows a situation for \( n = 5 \) possible positions. For example, if a wind turbine is installed at position 3, then no wind turbines can be installed at the positions 2 or 4. When the wind turbines are placed on the positions 1, 3 and 5, then they produce \( 6 + 5 + 11 = 22 \) units of energy. This solution is not optimal: An installation of wind turbines on the positions 2 and 4 produces \( 9 + 15 = 24 \) units of energy.

a) Provide an example (as simple as possible) that shows that the following greedy strategy does not necessarily lead to an optimal solution: “Select a possible position with maximal energy yield until no other wind turbines can be placed.”

b) Describe a dynamic programming algorithm that computes the maximal producible energy as efficiently as possible.

c) Specify the running time of your solution.

d) Describe in detail how the algorithm from (b) has to be modified to additionally compute an optimal placement of wind turbines that produces a maximum amount of energy.

Please turn over.
Exercise 8.2  Branch and Bound.

In this exercise, we wish to apply the branch and bound method to a classical optimization problem. Given an (undirected) graph \( G = (V, E) \) with \( n = |V| \) vertices and \( m = |E| \) edges, the Minimum Dominating Set Problem asks to compute a set of vertices \( D \subseteq V \) of smallest possible size, such that every vertex in \( V \setminus D \) has a neighbor in \( D \) (such a set is called dominating set).

We consider the following example:

```
  a   b   c
  |
  d--
  |
  e   f   g
  |
  h   i
```

In this graph, \( D = \{a, b, c, h, i\} \) is a dominating set. No proper subset of \( D \) is a dominating set, but \( D \) is not as small as possible, so it is not a minimum dominating set. The exercise is to find a minimum dominating set for the above graph by applying the branch and bound method by hand. Proceed as follows:

a) First, develop a lower bound on the number of required vertices for a given partial solution. A partial solution is described by two sets \( In \) and \( Out \). The set \( In \) contains the vertices that are chosen to be part of the dominating set, while \( Out \) contains the vertices that are chosen not to be part of it.

b) Provide a branching rule to specify which vertex is the next one to be considered.

c) Perform branch and bound using the answers you gave in a) and b). You do not have to use the learning module. Draw a complete decision tree and indicate the order of the decisions and the corresponding lower bounds. Identify the final solution.

**Hand-in:** until Wednesday, 22nd April 2015.