ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute for Operations Research
ETH Zurich HG G21-22

Prof. Dr. Rico Zenklusen and Assistants
Contact: math.opt@ifor.math.ethz.ch

IFOR
Institute for Operations Research
Department of Mathematics

Fall 2020

Mathematical Optimization – Problem set 6

https://moodle-app2.let.ethz.ch/course/view.php?id=12839

Problem 1: Complementary slackness
Consider the following linear program in canonical form.

$$\begin{array}{lrrrrrrrrrrrrl}
\max & & & 4x_2 & + & 2x_3 & - & x_4 & + & 3x_5 & - & 7x_6 & & \\
 & x_1 & & & & & - & x_4 & & & + & 3x_6 & \leq & -1 \\
 & 7x_1 & + & x_2 & + & x_3 & & & + & x_5 & + & x_6 & \leq & 5 \\
 & & & x_2 & - & x_3 & & & & & - & x_6 & \leq & 1 \\
 & & & x_2 & - & x_3 & + & x_4 & + & x_5 & + & x_6 & \leq & 5 \\
 & x_1 & & & + & x_3 & - & 4x_4 & + & 5x_5 & & & \leq & 0 \\
 & & & & & & & & & & & x_i & \geq & 0 \quad \forall i \in \{1, \ldots, 6\}
\end{array}$$

(a) Write the dual of the above linear program.

(b) Consider the primal feasible solutions

$$x^* = \begin{pmatrix} 0 \\ 2 \\ 2 \\ 4 \\ 0 \\ 1 \end{pmatrix} \quad \text{and} \quad x^* = \begin{pmatrix} 0 \\ 3 \\ 2 \\ 1 \\ 0 \\ 0 \end{pmatrix} .$$

Use the complementary slackness theorem (Theorem 1.90 in the script) to decide whether these two candidates are optimal solutions of the given linear program.

Problem 2: Asymptotic growth and Landau notation
Indicate whether the following statements are true or false by ticking the right box. Additionally, provide a short justification of your answer (the given space should be enough).

(a) Let $f(n) = \log(n^2)$ and $g(n) = \log(n)$. Then $f = \Theta(g)$.

True ☐ False ☐

Short justification:

(b) Let $f(n) = n + \sqrt{n}$ and $g(n) = n \cdot \log(n)$. Then $f = O(g)$.

True ☐ False ☐

Short justification:

(c) Let $f(n) = 2^n$ and $g(n) = n!$. Then $f = \Theta(g)$.

True ☐ False ☐

Short justification:

(d) Let $f(n) = \sqrt{n}^n$ and $g(n) = n^{\sqrt{n}}$. Then $f = \Theta(g)$.

True ☐ False ☐

Short justification:

(e) Let $f(n) = 3^n$ and $g(n) = 2^{n+\log n}$. Then $f = \Omega(g)$.

True ☐ False ☐

Short justification:

(f) Let $f, g \colon \mathbb{Z}_{\geq 1} \to \mathbb{Z}_{\geq 1}$. Then, at least one of the relations $f = O(g)$, $g = O(f)$, or $f = \Theta(g)$ holds.

True ☐ False ☐

Short justification:

Problem 3: Running time of a sorting algorithm

We consider the problem of sorting n given integers. More precisely, the input is an array A of length n, where $A[i] \in \mathbb{Z}$ for $i \in \{0, \ldots, n-1\}$, and the goal is to sort the array in non-decreasing order and return the sorted array.

A natural algorithm achieving this is called `InsertionSort`, and it works as follows. If, for some $i \in \{1, \ldots, n-1\}$, the first i elements of the array are in non-decreasing order, then take the $(i+1)^{\text{th}}$ element $A[i]$ and exchange it with its predecessor until it is at the right position within the first $i+1$ elements. After these exchange steps, the first $i+1$ elements are in non-decreasing order, and we can iterate with a larger i.

The above intuitive idea is formally described in the following algorithm.

Algorithm 1. `InsertionSort`

Input: Array A of length n with $A[i] \in \mathbb{Z}$ for all $i \in \{0, \ldots, n-1\}$.
Output: Array A sorted in non-decreasing order.

```
for i = 1, ..., n − 1:
    j = i
    while j ≥ 1 and A[j − 1] > A[j]:
        x = A[j]
        A[j] = A[j − 1]
        A[j − 1] = x
        j = j − 1
return A
```

(a) Track how a call of Algorithm 1 on the array $A = [4, 3, 1, 2]$ changes A.

(b) Prove that Algorithm 1 has a worst-case running time of $O(n^2)$.

(c) Show that Algorithm 1 has a running time of $\Omega(n^2)$ on some inputs.

(d) Show that there exist inputs on which Algorithm 1 has a running time of only $O(n)$.

Remark: Observe that together, (b) and (c) prove that the worst-case running time of Algorithm 1 is $\Theta(n^2)$, i.e., that the analysis in (b) is sharp. Omitting the argument in (c), it would not be clear whether $O(n^2)$ is a strong or a weak bound on the running time.

Problem 4: Quering edge existence in incidence lists

Let $G = (V, E)$ be a graph given by an incidence list and let $u, v \in V$ two vertices. Show that in time $O(\min\{\deg(u), \deg(v)\})$, it can be decided whether G contains an edge $\{u, v\}$.

Problem 5: Meeting at a central point

Imagine three caterpillars sitting on vertices v_1, v_2, and v_3 of a connected graph $G = (V, E)$. The three caterpillars would like to meet at one of the vertices of the graph. All they can do is moving from one vertex to another via the edges of the graph, and they can only stop at vertices of the graph, but not on an edge. When traversing an edge, the three caterpillars travel at the same speed of one edge per hour.

For an example, consider Figure 1. To meet at vertex v_5, the caterpillar from vertex v_1 could travel to v_5 via the vertices v_4 and v_2 in three hours (and there is no faster option), and the caterpillars from v_2 and v_3 could be there in one and two hours, respectively. Thus, an earliest meeting at v_5 could take place three hours after the caterpillars start moving.

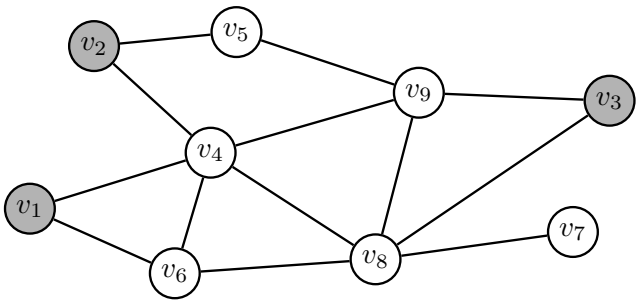

Figure 1: A graph $G = (V, E)$ with starting points of the caterpillars marked in gray.

Of course, the caterpillars want to meet at a vertex where an earliest possible meeting can be achieved. Can you provide an algorithm that determines such a vertex? Analyze the running time of your algorithm. Is it possible to construct an algorithm that solves the given problem and has running time $O(|V| + |E|)$?

Programming exercise

Work through the notebook `measuringRunningTime.ipynb`, where you learn how to measure running time in Python, and how to efficiently implement an algorithm that determines the number of connected components of a graph.