Homework Assignment 2

2-AIN-205, Spring 2026

Deadline: April 13, 2026, 22:00

Before you start working on this homework assignment, read general instructions at the end of this document. You can write you solutions in Slovak or English. The solutions must be your work. Do not copy from others and do not attempt to find the solutions in literature or on the internet! Grade of 40 points will be considered as "full marks" for this assignment. The rest are bonus points.

1. [*20 points*] **Cool game.** Given is a directed acyclic graph, with the vertices numbered from 1 to n. Each edge leads from a vertex with a smaller number to a larger one. We are also given k pairs vertices $\{u_1, v_1\}$, $\{u_2, v_2\}$, $\ldots$, $\{u_k, v_k\}$.

   In the cool computer game, we start from vertex 1 and choose a path which brings us to vertex n. If the path passes through both vertices u_i and v_i, we get a 100 point bonus i which then disappears from the game. Our goal is to find the path with the most points possible.

   a) This problem is NP-hard. Show how to solve the problem using integer linear programming, i.e. show how to construct the appropriate integer linear program for an arbitrary input.

   b) Imagine that instead of k two-vertex bonuses we are given k single-vertex bonuses $w_1, w_2, \ldots, w_k$. Whenever we pass a bonus vertex, we get 100 points and the task is again to collect the most points possible. Find an efficient algorithm solving this simplified problem. (It is possible to do that in polynomial time.)

   c) **Bonus 10 points:** Imagine that instead of one pass through the graph we can have m passes through the graph. In each pass we can use a different path, but each bonus can only be collected once in all passes (i.e., if we pass through the same bonus pair twice, we only receive one bonus). Propose an algorithm to solve this modified problem.

2. [*20 points*] **Vertex cover, independent set, and clique.** Consider undirected graph $G = (V, E)$. Recall that *vertex cover* of the graph is a set of vertices that covers at least one end of each edge; *independent set* is a set of vertices such that no two vertices are connected by an edge; *clique* is a set of vertices where each pair of vertices is connected by an edge.

   Set of vertices $U \subseteq V$ is an independent set if and only if $V - U$ is a vertex cover. Set U is also an indepentent set in graph G if and only if complement of U is a clique in a complement of graph G.

   Assume that you have invented a polynomial-time algorithm A for finding the largest independent set. In the following tasks, either **prove** the claims of prof. Knowitall, or **find a counterexample.**

   a) We can find the largest clique in graph G by running the algorithm A on the complement of graph G. The resulting set will also be the largest clique in graph G.

   Prof. Knowitall claims that, in the same way, we could transform a polynomial-time algorithm with a constant approximation factor for finding the largest independent set into a polynomial-time algorithm with constant approximation factor for finding the largest clique.

   b) We can find the smallest vertex cover of graph G by running the algorithm A on the same graph. Algorithm A returns set of vertices U as the largest independent set, and the smallest vertex cover will be simple set $V - U$.

   Prof. Knowitall claims that in the same way we could transform a polynomial-time algorithm with a constant approximation factor for finding the largest independent set into a polynomial-time algorithm with constant approximation factor for finding the smallest for finding the smallest vertex cover.

3. [*20 points*] **Programmming task.**

You are given a weighted oriented complete graph with n vertices. The goal is to find the shortest Hamiltonian path, i.e. the path that visits every vertex exactly once (and does not have to end where it starts) using integer linear programming. Your task is to write a program that receives a graph as an input and outputs a corresponding integer linear program.

The linear program has to contain variables $x1\_1, x1\_2, \ldots, x1\_n, x2\_1, \ldots, xn\_n$, where $xi\_j = 1$ if and only if the chosen path uses the edge from vertex i to vertex j. You can also use other variables and these can be named as you wish. The program must contain at most 1000 variables and 1000 conditions and must be solvable within a 5 second time limit.

Input format: The first line of the input file contains number of vertices n. Following n lines represent the graph by the adjacency matrix, i.e. the j-th number in i-th line corresponds to the weight c_{ij} of the directed edge from vertex i to vertex j; thus, each of these lines has n numbers.

Output format: See `http://lpsolve.sourceforge.net/5.0/CPLEX-format.htm`.

Restrictions: To obtain full marks, your program has to be able to solve inputs of size $n \leq 20$.

Input example:

```
3
0 1 4
1 0 1
4 1 0
```

Output example:

```
Minimize
obj: 1 x1_2 +4 x1_3 +1 x2_1 +1 x2_3 +4 x3_1 +1 x3_2
Subject To
ci1: x1_2 + x1_3 <= 1
ci2: x2_1 + x2_3 <= 1
ci3: x3_1 + x3_2 <= 1
co1: x2_1 + x3_1 <= 1
co2: x1_2 + x3_2 <= 1
co3: x1_3 + x2_3 <= 1
cx1: x1_2 + x1_3 + x2_1 + x3_1 >= 1
cx2: x2_1 + x2_3 + x1_2 + x3_2 >= 1
cx3: x3_2 + x2_3 + x3_1 + x1_3 >= 1
Binary
x1_2 x1_3 x2_1 x2_3 x3_1 x3_2
End
```

General Instructions

Theoretical tasks. Submit answers to theoretical tasks in **one pdf file per task** in vektor.fmph.uniba.sk (your answer can be typeset or scanned, but please make sure that everything is legible and easy to read). No late submissions are allowed.

Write your solutions so that they contain all information necessary to easily understand them, but at the same time try to aim for brevity. Prove all claims, including in the cases when it is not explicitly written in the problem statement.

If the task is about solving an algorithmic problem, submit the best algorithm you can design. The first criterion is that the algorithm *is correct*, the second criterion is the *running time and memory complexity*. Correct and slow algorithm is worth more points than fast incorrect algorithm. Inefficient but correct algorithms will always receive at least 50% of points. In your solution, you should:

- Describe the main idea of the algorithm.
- Write a pseudocode for your algorithm.
- Prove the correctness of your algorithm.
- Analyse its running time and memory complexity.

Programming tasks. You need to submit a functional program, no written part is required. Your solutions will be evaluated immediately on several inputs and you will learn number of points (there are several sets of inputs and you only gain points if you solve correctly all inputs in a particular set). You can submit a solution several times, we will only take into account the last submitted solution before the deadline.

You can submit your program in any supported programming language. However, the time limit are set of C++ and Python only, so we cannot guarantee that solutions in other programming languages can get a full grade. The details about the requirements as well as the list of supported programming languages can be found at `https://judge.ksp.sk/docs/co-odovzdavat`.