Advanced Algorithms, Spring 2021

Homework 3. Due date: Thursday 20/5/21

Please follow the HW guidelines published in HW1

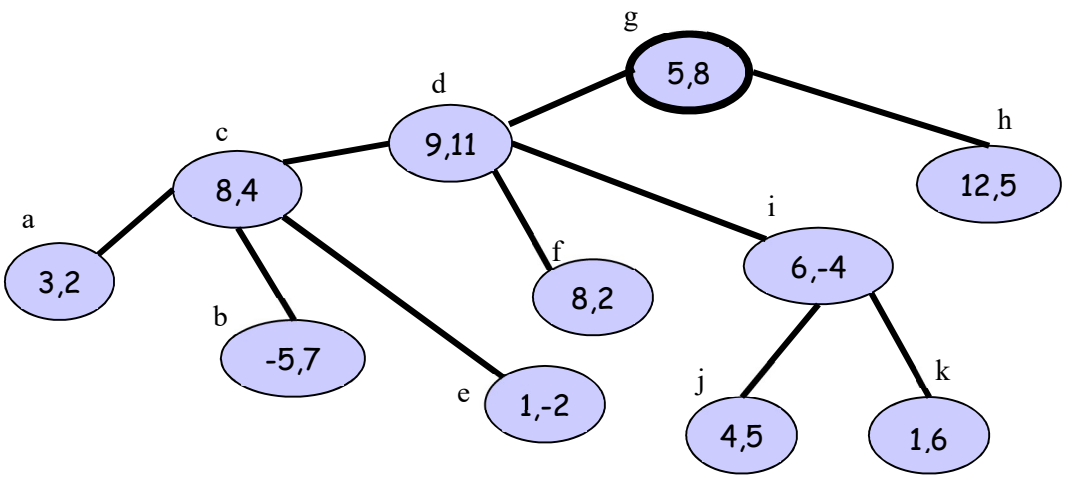
Question 1 (24 pts.)

Given a graph G, each vertex is associated with two values a(v), b(v). The goal is to find a vertex cover S for which the value $\sum_{v\in S} a(v) + \sum_{v\notin S} b(v)$ is minimal.

(a) Prove that the problem is NP-hard.

(b) Suggest an optimal algorithm, based on dynamic programming, for solving the problem in a **tree**. Explain shortly the correctness, no need to prove formally.

(c) Run your algorithm on the following tree (each vertex is labeled by a(v),b(v)), present the result in a table similar to the one we filled in class for MWIS.



Question 2 (20 pts.)

Definition: A set S of intervals is a **laminar family of intervals** if for any two intervals I_1 and I_2 in S, either they don't intersect at all (that is, $I_1 \cap I_2 = \varnothing$) or one of them is contained in the other (i.e., either $I_1 \subseteq I_2$ or $I_2 \subseteq I_1$).

Definition: A **clique cover** of an undirected graph G=(V,E) is a partition of V into subsets, each forming a clique in G (every two vertices in the same subset are adjacent).

A **minimum clique cover** is a clique cover that uses as few cliques as possible.

Given a **laminar family** of intervals, suggest a linear time algorithm for calculating a minimum clique cover in its corresponding interval graph. Prove the correctness of your algorithm, and justify its time complexity (in high-level, no need to write pseudo-code or to give implementation details).

Directions: Assume that the laminar family is given as a forest F of rooted trees, where each vertex in a tree in F corresponds to an interval, and a vertex v_1 is an ancestor of a vertex v_2 if and only if $I_2 \subseteq I_1$.

Question 3 (20 pts.)

Oren works in the HR department of a high-tech company. There are n workers in the company. Following the success of the event organized in homework 2, Oren organizes another big event. He sends the workers a long list of suggestions for morning activities and a long list of suggestions for afternoon activities (the lists are disjoint). Every worker is asked to mark her favorite morning activity and her favorite afternoon activity from the lists. Oren's budget and space constraints allow selecting k_1 activities for the morning, and k_2 activities for the afternoon.

Suggest an algorithm that determines whether Oren can select the activities such that for every worker, at least one of her two chosen activities is selected.

The time complexity of your algorithm should be $O(f(k_1+k_2)n^c)$ for some constant c.

1. List the kernelization rules and justify their correctness.
2. Describe the algorithm based on these rules, and analyze its time complexity. No need to specify the exact $f(k_1+k_2)$, but do explain why the kernel size depends on the parameters.

You can assume that no two workers mark the same pair of activities (this can be achieved by a simple preprocessing).

Question 4 (16 pts.)

4.1 Consider the problem $1|\ r_j\ |\ \Sigma_j C_j$, that is, non-preemptive scheduling on a single machine, with release times, where the goal is to minimize the total completion time. Show by an example that SPT algorithm (that schedule the shortest available jobs) may produce a schedule for which $\Sigma_j C_j$ is more than twice the minimal possible $\Sigma_j C_j$. Describe the instance, its SPT schedule and an optimal schedule.

4.2. The problem $P_2||\Sigma_j C_j$ is solved optimally on an instance of n=2k jobs and two identical parallel machines. The solutions' value is z. The processing time of each of the n jobs is increased by 1. The problem $P_2||\Sigma_j C_j$ is then solved on the resulting instance. What is the solution's value, as a function of n and z. Explain shortly.

Question 5 (20 pts.)

In class (lecture of May 10th), we introduced a PTAS for the minimum Makespan problem $P||C_{max}$ on m identical machines. The PTAS and its proof are given in Section 9.2.1.1 in the file PTAS_chatper available in the course web-page (under supplementary material).

1. Read and understand the PTAS and its analysis. For full credit, declare that you read and understood it. PLEASE SAVE PAPER – don't print the whole chapter, all you need is page 3.

2. Consider the following algorithm A'_k:

(a) Schedule optimally with no intended idle the longest k jobs,

(b) Schedule the longest unscheduled job (#k+1) on the least loaded machine.

(c) Schedule the remaining jobs in arbitrary order (not necessarily LPT) using List-Scheduling (each job is scheduled on the least loaded machine).

Can algorithm A'_k replace the algorithm A_k in the PTAS? If yes, explain in one paragraph why the proof 'survives'. If no, point to the part of proof that does not survive.