CS4445/9544 Analysis of Algorithms II

Second concept assignment
Due date: November 8 during class.

1. A group L of ℓ lecturers in some university need to teach a set $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$ of courses. Each course is taught once per day. For each course c_i we are given its staring time t_i and its duration d_i. Two different courses can be taught by the same lecturer if they do not overlap. We want to determine whether all the courses can be taught by the ℓ lecturers.

   - (20 marks) Give in pseudocode an algorithm for solving this problem. If all the courses can be taught by the lecturers, the algorithm must return the value *true*; otherwise it must return the value *false*.
   - (15 marks) Prove that your algorithm correctly solves the problem. You need to show that the algorithm terminates and that it computes the correct output.
   - (5 marks) Compute the time complexity of your algorithm.

2. A set $F = \{f_1, f_2, \ldots, f_n\}$ of files with integer sizes $s_1, s_2, \ldots, s_n$ needs to be stored in a hard disk of capacity K. We wish to find a subset of files of maximum total size **but not larger** than K to be stored in the disk. For example, if we have 4 files with sizes 3, 5, 8, and 6 and $K = 15$, an optimum solution is to store in the hard drive the files of size 6 and 8. The above problem is NP-hard.

   Consider the following algorithm for the problem. The following two questions refer to this algorithm.

   Algorithm files(F, S, n, K)
   In: Set F of siles, set $S = \{s_1, s_2, \ldots, s_n\}$ of n file sizes, and hard disk capacity K
   Out: Set of files of total size at most K

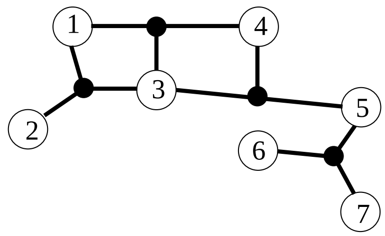
   ```
   A ← ∅
   total ← 0
   for i ← 1 to n do
           if total + s_i ≤ K then {
                   Add file f_i to A
                   total ← total + s_i
           }
   return A
   ```

   (*i*) (*10 marks*) For the above algorith the value SOL of the solution computed by the algorithm is equal to the total size of the files in set A, i.e. $SOL = total$. Show that the approximation ratio, OPT/SOL, of this algorithm is arbitrarily large by giving an instance in which the total size of the files in the set A returned by this algorithm is very small compared to the value of an optimum solution. Note that the files are not sorted in any particular manner.

   (*ii*) (*20 marks*) Assume now that the files are sorted in non-increasing order of size. Compute the approximation ratio OPT/SOL of the algorithm.

3. In a graph an edge (u, v) is specified by its two endpoints. A *3-hypergraph* $H = (V_H, E_H)$ consists of a set of vertices and a set of hyperedges, where a hyperedge is specified by

three endpoints (u, w, v), not by two as in graphs. The minimum vertex cover problem on 3-hypergraphs is to find the smallest set S if vertices, such that every hyperedge has at least one endpoint in S. For example, for the following 3-hypergraph with hyperedges $(1, 2, 3)$, $(1, 3, 4)$, $(3, 4, 5)$, and $(5, 6, 7)$ a minimum vertex cover is $S = \{3, 5\}$.



The minimum vertex cover problem on 3-hypergraphs is NP-hard. Consider the following generalization of the algorithm that we studied in class for the minimum vertex cover problem.

Algorithm VertexCover($H = (V_H, E_H)$)
Input: 3-hypergraph H
Output: Vertex cover of H of small size.

$S \leftarrow \emptyset$
while $E_H \neq \emptyset$ **do** {
    Choose a hyperedge $(u, v, w) \in E_H$
    $S \leftarrow S \cup \{u, v, w\}$
    Remove from E_H every hyperedge incident on u, v, or w
}
Output S

- (*20 marks*) Compute the approximation ratio of this algorithm.
- (*10 marks*) Suppose that in every iteration of the **while** loop instead of adding to S the three vertices u, v, and w we only add u to S. The idea, is that then we might be able to produce a smaller solution as we add only one vertex to S, not three of them. Compute the approximation ratio of this modified algorithm.