# Advanced Analysis of Algorithms - Homework IV

K. Subramani
LCSEE,
West Virginia University,
Morgantown, WV
{ksmani@csee.wvu.edu}

## 1 Instructions

1. The homework is due on December 10, in class.
2. Each question is worth 4 points.
3. Attempt as many problems as you can. You will be given partial credit, as per the policy discussed in class.
4. The work must be entirely your own. You are expressly **prohibited** from consulting with colleagues or the internet (with the exception of the material on the course website, your class notes and [NN09]).

## 2 Problems

1. (a) Let $e$ be a maximum-weight edge on some cycle of a connected, undirected graph $\mathbf{G} = \langle V, E, \mathbf{c}\rangle$. Note that $V$ is the vertex set, $E$ is the edge set and $\mathbf{c}$ is the weight function, which associates a positive integer with each edge in $E$. Prove that there is a minimum spanning tree of $\mathbf{G}$ that does not include the edge $e$.

   (b) Professor Borden proposes a new Divide-and-Conquer algorithm for computing minimum spanning trees, which goes as follows. Given a graph $\mathbf{G} = \langle V, E, \mathbf{c}\rangle$, partition the set $V$ of vertices into two sets $V_1$ and $V_2$ such that $|V_1|$ and $|V_2|$ differ by at most 1 . Let $E_1$ be the set of edges that are incident only on vertices in $V_1$, and let $E_2$ be the set of edges that are incident only on vertices in $V_2$ . Recursively solve a minimum-spanning-tree problem on each of the two subgraphs $\mathbf{G_1} = \langle V_1, E_1, \mathbf{c_1}\rangle$ and $\mathbf{G_2} = \langle V_2, E_2, \mathbf{c_2}\rangle$. Finally, select the minimum-weight edge in $E$ that crosses the cut $(V_1, V_2)$, and use this edge to unite the resulting two minimum spanning trees into a single spanning tree. Either argue that the algorithm correctly computes a minimum spanning tree of $\mathbf{G}$, or provide an example for which the algorithm fails.

2. Assume you are given $n$ tapes $T = \{T_1, T_2, \ldots T_n\}$. Tape $T_i$ contains $l_i$ records, $i = 1, 2, \ldots, n$. We wish to merge these tapes into a single tape containing all the records. When two tapes are merged, the number of record moves is equal to the sum of the number of records contained in each. For instance, if tape $T_5$ is merged with tape $T_{10}$, the number of record moves is $(l_5 + l_{10})$. The goal is to merge the tapes, two at a time, such that the number of record moves is minimized. Design an algorithm for this problem and analyze its running time.

3. In class, we discussed the Branch-And-Bound enumeration technique for solving integer programs. Essentially, we use the linear programming relaxations to find good bounds, which permits the pruning of the search tree. Solve the

following integer program using Branch-And-Bound enumeration:

$$\begin{aligned} \max z \quad & = -x_1 + 4x_2 \\ \text{subject to} & \\ -10 \cdot x_1 + 20 \cdot x_2 & \leq 22 \\ 5 \cdot x_1 + 10 \cdot x_2 & \leq 49 \\ 8 \cdot x_1 - x_2 & \leq 36 \\ x_1, x_2, x_3, x_4 & \geq 0 \\ x_1, x_2, x_3, x_4 & \text{ integer} \end{aligned}$$

To solve this problem, you do need to know how to solve 2-dimensional linear programs. This can be done either graphically or through *lp-solve*.

4. The Maximum Independent Subset (MIS) problem is defined as follows: Given an undirected graph $\mathbf{G} = \langle V, E \rangle$ and a target $K$, is there a set $V'$ of vertices, such that $V' \subseteq V$, $|V'| \geq K$ and two vertices in $V'$ are connected by an edge in $G$. In class, we mentioned the fact that the MIS problem is is **NP-complete**. What can you say about the complexity of the problem, if the graph has no cycles? If you claim that the problem is in **P**, then you should provide a polynomial time algorithm. On the other hand, if claim that the problem is **NP-complete**, then you must provide a reduction from an **NP-hard** problem to the MIS problem on trees.

5. Let $S = \{a_1, a_2, \ldots, a_n\}$ denote a finite set. Let $C$ denote a collection of $m$ subsets of $S$, i.e., $C = \{S_1, S_2, \ldots S_m\}$, where each $S_i \subseteq S$. We are interested in the following problem: Is there a partition of $S$ into two subsets $R$ and $T$, such that no set in $C$ is completely contained in either $R$ or $T$? Prove that this problem is **NP-complete**.

# References

[NN09] Richard Neapolitan and Kumarss Naimipour. *Foundations of Algorithms Using C++ Pseudocode*. Jones and Bartlett, 2009.