School of Computer Science and Technology
University of Science and Technology of China

# Exercise Sheet 1 for
# Design and Analysis of Algorithms
# Autumn 2023

Due 25 Oct 2023 at 18:00

---

**Exercise 1** *40*

Suppose that someone gives you a black-box algorithm $\mathcal{A}$ that takes an undirected graph $G = (V, E)$, and a number $k$, and behaves as follows.

- If $G$ is not connected, it simply returns "$G$ is not connected."
- If $G$ is connected and has an independent set of size at least $k$, it returns "yes."
- If $G$ is connected and dose not have an independent set of size at least $k$, it returns "no."

Suppose that the algorithm $\mathcal{A}$ runs in time polynomial in the size of $G$ and $k$.
Show how, using calls to $\mathcal{A}$, you could then solve the Independent Set Problem in polynomial time: Given an arbitrary undirected graph $G$, and a number $k$, does $G$ contain an independent set of size at least $k$?

---

**Exercise 2** *30*

Consider the following greedy algorithm for the knapsack problem. We initially sort all the items in order of nonincreasing ratio of value to size so that $v_1/s_1 \geq v_2/s_2 \geq \cdots \geq v_n/s_n$. Let $i^*$ be the index of an item of maximum value so that $v_{i^*} = \max_{i \in I} v_i$. The greedy algorithm puts items in the knapsack in index order until the next item no longer fits; that is, it finds $k$ such that $\sum_{i=1}^{k} s_i \leq B$ but $\sum_{i=1}^{k+1} s_i > B$. The algorithm returns either $\{1, \ldots, k\}$ or $\{i^*\}$, whichever has greater value. Prove that this algorithm is a 1/2-approximation algorithm for the knapsack problem.

---

**Exercise 3** *30*

One key element in the construction of the fully polynomial-time approximation scheme for the knapsack problem was the ability to compute lower and upper bounds for the optimal value that are within a factor of $n$ of each other (using the maximum value piece that fits in the knapsack to get the lower bound). Use the result of the previous exercise to derive a refined approximation scheme that eliminates one factor of $n$ in the running time of the algorithm.

---