Dr. Alexander Munteanu
Jan Höckendorff, Dennis Köhn,
Jonathan Langer, Dr. Vibha Salot

winter term 2024/2025

1. Home Exercises

for the lecture

Efficient Algorithms

Please hand in via ILIAS

Ex. 1 *(2 + 3 + 3 points)*

Consider the **Load Balancing** problem and the **GreedyLoadBalancing** algorithm introduced in the lecture. In this exercise, we modify the algorithm with an additional step that first sorts the given jobs by their length in descending order and then applies the algorithm, i.e., the algorithm assigns the job with largest length first.

a) Show that this modified algorithm is still not optimal, i.e., find an instance for which the makespan of the solution obtained by the algorithm is larger than the makespan of the optimal solution.

b) Assume that there are at least $m + 1$ jobs, where m is the number of machines. Prove that after sorting the jobs we have

$$\text{makespan}(f) \geq 2 \cdot t(m+1)$$

for any schedule f, where $t(j)$ denotes the length of job j.

c) Prove that the modified algorithm is a $\frac{3}{2}$-approximation algorithm.

Ex. 2 *(4 points)*

Consider the **Set Cover** problem and the **GreedySetCover** algorithm introduced in the lecture. Apply the algorithm to the following input instance. Use the following tie-breaking rule: if multiple sets have the same size, the set with the smallest index is chosen.

The set we want to cover is given by

$$S = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}.$$

The collection $\mathcal{T} = \{T_1, T_2, \ldots, T_8\}$ of sets that we can use to cover S is given by

$$\begin{aligned} T_1 &= \{1, 2, 3\}, \\ T_2 &= \{1, 3, 4\}, \\ T_3 &= \{2, 3, 6, 9\}, \\ T_4 &= \{6, 7, 8\}, \\ T_5 &= \{9\}, \\ T_6 &= \{9, 10\}, \\ T_7 &= \{5, 9, 10\}, \\ T_8 &= \{1, 5, 7\}. \end{aligned}$$

For each iteration of the **while**-loop, explain which set T_i is chosen and why and show how all sets T_ℓ are updated.

Ex. 3 *(4 + 2 + 2 Points)*

Let $G = (V, E)$ be an unweighted graph. A matching in G is a set $M \subseteq E$ such that no pair of edges in M are adjacent (share a vertex). A maximum matching is a matching of maximum size.

a) Design a $\frac{1}{2}$-approximation algorithm for computing a maximum matching of a given graph G. The running time of the algorithm should be in $O(|E| + |V|)$.

b) Analyze the running time of your algorithm.

c) Prove the correctness of your algorithm, i.e., prove that the algorithm computes a valid matching and that this matching satisfies the required approximation guarantee.