Reading
Chapter 16.1, 16.2
Chapter 23
ADw8
1. p.422 16.1-2
2. p.447 16-2(a)
3. p.637 23.2-8

1. p.422 16.1-2

If we view time in reverse, then this approach is the same as choosing activities that are first to finish.

Suppose $S_{optimal} = \{a_i\}\ for\ some\ a_i \subseteq S$ where $a_i = [s_i, f_i)$ is the greedy solution to this approach. We now convert $a_i = [s_i, f_i)$ to $a'_i = [s'_i, f'_i)$ where $s'_i = f_i$ and $f'_i = s_i$. This has become the original problem of choosing the first activity to finish, with time scale inverted, and $S'_{optimal} = \{a'_i\}$ is the optimal greedy solution to the original problem.

2. p.447 16-2(a)

Greedy strategy that picks the task with shortest runtime clearly works.
We can define recursive algorithm. Z_S denotes an optimal sequence of tasks from the set S.

$$\{Z_S\} = \begin{cases} NUL & if\ S = \emptyset \\ \{a_i,\ Z_{S-\{a_i\}}\} & if\ S \neq \emptyset \end{cases} \text{ where } p_i \leq p_j\ \forall\ 1 \leq j \leq n$$

Let Z denote an optimal sequence of tasks given by the above recursive algorithm. We prove it's optimal by contradiction. Suppose is it not optimal – this means that we schedule some a_j instead of a_i first. Suppose this a_j is the k[th] tasks in our original Z sequence. Then we have increased the average completion time by $\frac{1}{n}(k-1)(p_j - p_i) > 0$. Therefore, our original Z is the optimal sequence.

The running time would be the time complexity incurred from sorting the tasks by p_i. This could be done in $\Theta(nlogn)$ using e.g. merge sort.

3. p.637 23.2-8

We prove the correctness/incorrectness of the recursive scheme through induction.

We assume that the algorithm finds the MST T_1 for $G_1 = (V_1, E_1)$, and analogously for G_2. In the merge step, we are however not guaranteed to find a MST for the entire graph G.
Suppose there exists $(v_i, v_k) \in T_1$, and $(v'_i, v'_k) \in T_2$ such that

$$w(v_i, v_k) + w(v'_i, v'_k) > w(v_i, v'_i) + w(v_k, v'_k)$$
$$\text{and } \min\{w(v_i, v_k), w(v'_i, v'_k)\} > \min\{w(v_i, v'_i), w(v_k, v'_k)\}$$

In the recursive algorithm, we would connect vertices (v_i, v_k), (v'_i, v'_k) and the lightest vertex say (v_i, v'_i). This incurs total weight of $W = w(v_i, v_k) + w(v'_i, v'_k) + \min\{w(v_i, v'_i), w(v_k, v'_k)\}$.

Were we to connect vertices (v_i, v'_i), (v_k, v'_k) and say (v_i, v_k), we incur a lower total weight $W' = w(v_i, v'_i) + w(v_k, v'_k) + \min\{w(v_i, v_k), w(v'_i, v'_k)\}$. $W > W'$.