1. In our linear-time selection algorithm, the inputs are divided into groups of 5. What if you used groups of 3 instead? What if used groups of 7 or larger (odd integers)? (15 points)

2. Not every greedy strategy works. Show that if in the activity-selection problem we always select the activity of least duration (another greedy strategy) from among those that are compatible with the previously selected activities we might not get an optimal solution. (15 points)

3. Give an $O(n)$-time dynamic programming algorithm for the maximum subarray problem. That is, given an array of real numbers $a_1, a_2, \ldots, a_n$, your algorithm must compute the maximum sum $\sum_{i=j}^{k} a_i$, where $1 \leq j \leq k \leq n$. Draw the subproblem graph. How many vertices and edges are in the graph? (15 points)

4. Suppose we perform a sequence of $n$ operations on a data structure in which the $i$-th operation costs $i$ if $i$ is an exact power of 2, and 1 otherwise. Use the accounting method to determine the amortized cost per operation and use this to get an upper bound on the actual cost of the sequence of $n$ operations. (20 points)

5. Consider the same set-up as in the previous problem but now use the potential method to determine the amortized cost per operation and use this to get an upper bound on the actual cost of the sequence of $n$ operations. (20 points)

6. Describe an algorithm for finding a spanning tree with minimal weight containing a specified acyclic (cycle-free) set of edges $A$ in a connected weighted undirected graph. (15 points)