# CS 561, HW4

Prof. Jared Saia, University of New Mexico

*Due: October 15th*

1. Consider the following alternative greedy algorithms for the activity selection problem discussed in class. For each algorithm, either prove or disprove that it constructs an optimal schedule.

   - Choose an activity with shortest duration, discard all conflicting activities and recurse
   - Choose an activity that starts first, discard all conflicting activities and recurse
   - Choose an activity that ends latest, discard all conflicting activities and recurse
   - Choose an activity that conflicts with the fewest other activities, discard all conflicting activities and recurse

2. Now consider a weighted version of the activity selection problem. Imagine that each activity, $a_i$ has a *weight*, $w(a_i)$ (weights are totally unrelated to activity duration). Your goal is now to choose a set of non-conicting activities that give you the largest possible sum of weights, given an array of start times, end times, and values as input.

   (a) Prove that the greedy algorithm described in class - Choose the activity that ends first and recurse - does not always return an optimal schedule.

   (b) Describe an algorithm to compute the optimal schedule in $O(n^2)$ time. Hint: 1) Sort the activities by finish times. 2) Let $m(j)$ be the maximum weight achievable from activities $a_1, a_2, \ldots, a_j$. 3) Come up with a recursive formulation for $m(j)$ and use dynamic programming. Hint 2: In the recursion in step 3, it'll help if you precompute for each job $j$, the value $x_j$ which is the largest index $i$ less than $j$ such that job $i$ is compatible with job $j$. Then when computing $m(j)$, consider that the optimal schedule could either include job $j$ or not include job $j$.

3. Consider a data structure over an initially empty list that supports the following two operations. APPEND-NUMBER(x): Adds the number $x$ to the beginning of the list; and MIN-MAX: Computes the min and max in the list in linear time and then deletes all elements from the list except for the min and max.

   (a) Assume an arbitrary sequence of $n$ operations are performed on this data structure. What is the worst case run time of any particular operation?

   (b) Show that the amortized cost of an operation is $O(1)$ using the potential method. Make sure to prove your potential function is valid.

4. Suppose we are maintaining a data structure under a series of operations. Let $f(n)$ denote the actual running time of the nth operation. For each of the following functions $f$, determine the resulting amortized cost of a single operation.

   - $f(n) = n$ if n is a power of 2, and $f(n) = 1$ otherwise.
   - $f(n) = n^2$ if n is a power of 2, and $f(n) = 1$ otherwise.

5. Describe and analyze a data structure to support the following operations on an array $A[1 \ldots n]$ as quickly as possible. Initially, $A[i] = 0$ for all $i$.

   - Given an index i such that $A[i] = 0$, set $A[i]$ to 1.
   - Given an index $i$, return $A[i]$
   - Given an index $i$, return the smallest index $j \geq i$ such that $A[j] = 0$, or report that no such index exists.