# CSE 6140 - Homework 2

Alexander Winkles

For assignment 2, I worked with Willian Kong and Haodong Sun. I utilized the class textbooks (Kleinberg, CLRS, and Benoit) as well as the website "GeeksForGeeks" for different perspectives on designing the algorithms.

**Problem 1**

1. Consider a greedy solution that sorts by smallest $t_i$. This is not optimal, consider three emails of weights 1, 2, and 200 with times 1, 2, and 5 respectively. The greedy solution gives $\sum_{i=1}^{3} w_i C_i = 1607$, where if we order by weight this value becomes 1022. Thus this is not optimal by counter example.

2. Consider a greedy solution that sorts by largest $w_i$. This is not optimal - consider two emails of weights 200 and 100 with times 200 and 1 respectively. The greedy solution gives 60100, where if we swap the two emails we the value becomes 40300. Thus this is not optimal by counter example.

3. This greedy algorithm is optimal. Consider an optimal solution $O$ and a greedy solution $A$. Our goal is to use an exchange argument to gradually transform $O$ into $A$ without increasing the overall cost. We define an *inversion* to be a pair of jobs $i$ and $j$ such that $\frac{w_i}{t_i} > \frac{w_j}{t_j}$ but $j$ is scheduled before $i$. By defintion of the greedy algorithm, $A$ has no inversions. Because $O$ and $A$ are solutions to the same set of emails, they take the same total amount of time. Swapping any inverted jobs will not increase the cost of the solution. Now if $O$ has any inversion, then it has a consecutive pair of emails that are inverted and thus can be swapped without increasing the overall cost. By swapping all inversions found in $O$, we arrive at a solution identical to $A$.

**Problem 2**

1. The indices are 4 and 7, with a sum of 32.

2. **Algorithm:**

   (a) Divide the list into two sublists.

   (b) Return the maximum of the following values:

      i. Maximum sum of left subarray, using a recursive method

      ii. Maximum sum of right subarray, using a recursive method

      iii. Maximum sum of subarray that includes middle point

   Because the maximum sums of the left and right array use recursive methods, they have a time complexity of $T(n) = 2T(n/2) + \Theta(n)$. Using the Master Theorem, this algorithm is $\Theta(n \log n)$.

3. A linear time algorithm would be such:

   **Algorithm:**

   (a) $val1 = val2 = T[0]$

   (b) for $x \in T - T[0]$

      i. $val1 = max x, val1 + x$

      ii. $val2 = max val1, val2$

   (c) return $val2$

**Problem 3**

1. By the Master Theorem, $a = 49$, $b = 7$, and $d = 2$, so $T(n) \in \Theta(n^2 \log n)$.

2. In this problem $a = \frac{1}{4}$, $b = 9$, and $d = 1$, so the Theorem does not apply as $a \geq 1$ is required.

3. In this problem $a = 4$, $b = 2$, and $d = 2$. Since this problem is not monotonically increasing, we may not apply the theorem.

4. In this problem $a = 2$, $b = 4$, and $d = 0.6$. Thus by the Master Theorem, $T(n) \in \Theta(n^{0.6})$.

5. In this problem $a = 3$, $b = 2$ and $d = 0$. Thus by the Master Theorem, $T(n) \in \Theta(n^{\log_2 3})$.

**Problem 4**

1. To start, we will prove the problem has optimal substructure.