CS 577 Homework 3

Alex Pizzuto

Question One

We must schedule n tasks such that we can maximize the number of tasks to be scheduled within a given period of time.

a We define our greedy algorithm as follows: We will complete the tasks in order of increasing running time, so we can express our solution as $A = \{a_1, a_2, \ldots, a_n\}$ where $t(a_i) > t(a_j)$ if $i < j$. Thus, we would need to sort our tasks in order of decreasing running time by using Merge Sort and then we would take successive elements from the list (if it is in ascending order, as is the default of Merge Sort, then take elements starting from the end of the list, and if it is sorted in reverse order, take elements from the beginning), until we have met our time threshold. Now, we demonstrate that it is the optimal.

Proof: Greedy Stays Ahead (by Induction)

First, some notation. We have already defined our greedy solution, A. Now, let us define a purposed optimal solution, $O = \{o_1, o_2, \ldots, o_n\}$ such that there exists at least set of indices, i, j such that $t(o_i) < t(o_j)$ where $i < j$.

Basis of Induction: We first look at the case of the first element in our greedy solution and our purported optimal solution. By definition, $t(a_1) \leq t(a_i) \quad \forall i \in 2, \ldots, n$. We also know that all elements in A are also elements of O, thus $\exists j$ such that $o_1 = a_j$. Therefore, we have $t(a_1) \leq t(o_1)$, and our base case holds as we have fit the greatest number into the narrowest time window that allows for the completion of at least one task given our set.

Inductive step: Suppose that $\exists \tau$, a period of time, such that $|A_\tau| \geq |O_\tau|$. We will show that if we increase the time window, $\tau \to \tau' = \tau + \epsilon$, that $|A_{\tau'}| \geq |O_{\tau'}|$. Now, there are a few subcases, all of which we will prove result in our claim still being true. First, we assume that the change in time window, ϵ, is not large enough such that there are remaining tasks in A or in O that can be scheduled. In that case, $|A_{\tau'}| = |A_\tau| \geq |O_\tau| = |O_{\tau'}|$, and our solution still performs better. Next, assume that there exists unscheduled elements in A that are smaller than ϵ, but not in O, in this case $|A_{\tau'}| > |A_\tau| \geq |O_\tau| = |O_{\tau'}|$, and our solution still performs better (it is worth noting that this case is not actually possible). Next, we assume that there are unscheduled elements in O that are smaller than ϵ, but none such remaining in A. In this case, $|O_{\tau'}| > |O_\tau|$, and $|A_{\tau'}| = |A_\tau|$. We will prove this subcase by contradiction. Assume that $|A_{\tau'}| < |O_{\tau'}|$, and for convenience define $N = |A_{\tau'}|$ and $M = |O_{\tau'}|$. That would mean that $\sum_{i=1}^{N} t(a_i) \leq \tau'$ and $\sum_{i=i}^{M} t(o_i) \leq \tau'$. However, in order to have one sum with more elements than another that are bounded by the same constant as tightly as possible, the elements in the set with larger cardinality must be on average smaller than the elements in the other set. This goes against the construction of A, which is selected as having the smallest elements first, so it is not possible to have $|A_{\tau'}| < |O_{\tau'}|$. The last case is that the ϵ is large enough such that there is room to add tasks from both lists. This claim is stronger than subcase 3, as we have shown that $|A_\tau| \geq |O_{\tau'}|$, from case 3, and now we have $|A_{\tau'}| > |A_\tau|$.

As we have shown that our solution has a larger cardinality than the purported optimal solution in all subcases, our greedy solution is the best performing algorithm.

b Our algorithm uses a Merge sort ($\mathcal{O}(n \log n)$) and then an iteration through the list adding up tasks until we have reached our time threshold ($\mathcal{O}(n)$), thus our overall complexity is $T(n) = \mathcal{O}(n \log n)$.

Question Two

Our greedy algorithm is as follows: Suppose you have stopped at location d_i to refuel (Assuming that we have filled up at our origin as well, in order to be applicable in all edge cases). In order to pick the next location to refuel, look for the location, d_j that is the largest value such that $d_i + m > d_j$. We call our solution set $A = \{a_i, \ldots, a_l\}$ where a_i corresponds to a d_j in our list of gas stations. Suppose there is a purported optimal solution, $O = \{o_i, \ldots, o_k\}$, and we seek to prove that $l < k$.

Proof: Greedy Stays Ahead (by Induction) Claim: $\forall r \geq 1$, we have $a_r \geq o_r$. This claim amounts to the same proof. Basis of Induction: Let $r = 1$. Then if $o_1 > a_1$, this would mean that either $a_1 < o_1 < m$, which is not possible because then our greedy algorithm would have picked a different value, or it could be that $o_1 > m$, but this is also not possible because then we would have run out of fuel before making it to o_1. Thus, our base case holds.

Inductive Step: Suppose $\exists r > 1$ such that $a_r \geq o_r$, we will prove that this implies that $a_{r+1} \geq o_{r+1}$. Suppose, for contradiction, that $a_{r+1} < o_{r+1}$. This would mean that we have $o_r \leq a_r < a_{r+1} < o_{r+1}$. However, as a_{r+1} is selected by looking for the largest d_j such that $d_j < a_r + m$, thus, $o_{r+1} - ar > m$, and we have guaranteed that the distance between o_r and o_{r+1} is greater than m as a result of the displayed chain of inequalities. Thus, it is impossible to have $a_{r+1} < o_{r+1}$, and our claim holds. QED.

Question Three: 4.6

We are asked to schedule triathlon participants in such a way that minimizes the completion time, assuming that for each participant, i, we are given their swimming, biking, and running times, s_i, b_i, and r_i, and with the restriction that no two participants can be swimming at the same time. First, some quick dimensionality reduction. It is clear from the statement of the problem that the biking and running time independently do not play a role, instead, we are only interested in the swimming time, s_i, and the combined biking plus running time, $t_i = b_i + r_i$.

With this formalism, our greedy algorithm is as follows: Perform a sort (Merge Sort implementation) of the combined running biking times, t_i in descending order. Then start participants based off of this order.

Proof: Exchange Argument

Consider a purported optimal solution, O that does not have the same order as our greedy solution. Then there must exist at least one inversion in O such that $i < j$ but $t_i < t_j$. Now, we consider the solution that we would get if we were to swap i with j in the optimal solution. In this case, it is clear that j finishes earlier than j did before, but we must now show that now i finishes before j would have in the previous scenario. In the original scenario, if contestent i started at τ_0, then contestant j could start at $\tau_0 + s_i$ and would therefore finish at $\tau_0 + s_i + s_j + t_j$. In the swapped scenario, j would start at τ_0, and i would therefore start at $\tau_0 + s_j$ and finish at $\tau_0 + s_j + s_i + t_i$. Comparing the two finish times, it is clear that the swapped scenario ends sooner,

as we specified that $t_i < t_j$. Thus our swapped solution does not have a greater completion time, so it is optimal. We continue iterating through all of the possible $\frac{n(n-1)}{2}$ inversions present in O. Once that is complete, we will have a schedule that is the same as the one from our greedy solution, thus our greedy solution is optimal.

Finally, we analyze the computing complexity. Here, we must do a sort of decreasing combined running and biking times. The combination of running and biking times is $\mathcal{O}(n)$, and with a merge sort implementation, the sorting step is $\mathcal{O}(n \log n)$. Thus, our algorithm is $T(n) = \mathcal{O}(n \log n)$.