Time Spent: 2 hours

Collaborators and Resources: Compared algorithms with Nathaniel MacArthur-Warner

---

Problem 2

For this problem, we're supposed to select players in a left-to-right order (that is, in strictly increasing order of indices) as well as in left-to-right order based on the actual values at those indices (strictly increasing order of experience). Specifically, we're given an array of A[1, ...,n] players where A[i] $\in \mathbb{R}_{\geq 0}$ represents the experience of player i. We just want to find the length of the longest subsequence of players such that the players are in left-to-right order (increasing order of indices) and the values (experience) at those indices are also in strictly increasing order.

Given an array of A[1, ...,n] players such that A[i] $\in \mathbb{R}_{\geq 0}$

1. Initialize an array S = numeric(n) and populate it with 1s
2. Initialize an empty array M
3. Set m = 0
4. For i in 1:n
    (a) For j in 0:i-1
        i. If A[i] > A[j] and S[j] + 1 > S[i] # That is, if the next element in A we're looking at is greater than the previous ones (we iterate over all previous ones) and we haven't already found A[i] to be a part of the sequence
            A. S[i] = S[j] + 1
5. For i in 1:n
    (a) If S[i] $\neq$ S[i+1]
        i. Add i to M

6. Set m = max(A) and return m (the length of the longest sequence)
7. Return M (the indices in A for the longest sequence)

Claim 1. The algorithm correctly finds the length of the longest sequence of players for which both the indices and values are strictly increasing in $O(n^2)$ time where n is the number of players. It also returns the indices corresponding to the values in the sequence.

Proof. We can demonstrate that this algorithm works because of the fact that it's going to return the largest value in S which is equal to the longest sequence of players such that players are in left-to-right order and are in strictly increasing order of experience. This is because the algorithm starts by assuming that the longest sequence has length 1. It goes through each element of A and finds whether that element is greater than any of the candidates for the most recent element of M, the longest sequence. Continuing in this way it will successfully find the length and indices for the longest left-to-right sequence of players in strictly increasing order of experience.

To illustrate this, take the result of applying the algorithm to the array A = [65,70,66,68,64]. The result will be that S = [1,2,2,3,1] and M = [65,66,68]. When the algorithm looks for the third player to add to M, it only considers players i such that S[i] = 2. Otherwise, it wouldn't find players in strictly increasing order of experience. By induction we can demonstrate that this algorithm works and that it identifies the players corresponding to an increase in total sequence length.

Furthermore, it is clear that it runs in $O(n^2)$ time because of the fact that all operations (checking if the players under consideration are valid and augmenting the count if so) are done in linear time except for the two nested for-loops, which run a maximum of n times each. So the upper bound for runtime is $O(n^2)$. □