CS 413 Homework 3

Sterling Jeppson

February 26, 2021

Problem 3

Proposition. *The number of boxes delivered by n trucks using the greedy algorithm is greater than or equal to the number of boxes delivered by n trucks using any other algorithm.*

Proof. Let the property $P(n)$ be the proposition to be proved.

Show that $P(1)$ is true: The number of boxes sent by the greedy truck is the maximum number of boxes that can be sent under the weight and ordering restraints of the problem. Hence the non-greedy truck can send only as many boxes as the greedy truck and so $P(1)$ is true.

Show that for all integers $k \geq 2$, $P(k-1) \implies P(k)$: Let k be any integer with $k \geq 2$ and suppose that the proposition is true for $k-1$ trucks. We must show that this implies that the proposition is true for k trucks. Let box b_i and b_j be the last box sent by the $k-1$th greedy and non-greedy truck respectively. By the inductive hypothesis, $i \geq j$. Now load the kth greedy truck and suppose that the range of boxes loaded by this truck is $[b_{j+1}, b_x]$. In order for the greedy truck to keep up it would need to load at least all boxes in the range $[b_{i+1}, b_x]$. But it can clearly do this as the set of boxes sent by the greedy truck is a subset of the boxes sent by the other truck and therefore will not exceed the weight limit. □

It follows immediately that the greedy algorithm uses the least number of trucks.

Problem 4

Let S be a sequence of length n and let S' be a possible sub-sequence of length m so that $S = s_0, ..., s_{n-1}$ and $S' = s'_0, ..., s'_{m-1}$. The algorithm linearly searches through the elements of S until a match is found between an element in S and the current element in S'. Then the algorithm increments the current index of S'. This repeats until all elements in S' have been matched with an element in S or until S has no more elements to match with S'. The algorithm is given below:

```
template <typename T>
bool isSubsequence(vector<T> S, vector<T> SPrime) {
    unsigned i = 0, j = 0, n = S.size(), m = SPrime.size();
    while(j < m && i < n)
        if(S[i++] == SPrime[j]) j++;
    return (j == m);
}
```

Since i is incremented in each iteration of the while loop, and since the body of the while loop runs in constant time, we can conclude that the algorithm runs in $O(n)$ time where n is the size of the sequence.

We must show that the algorithm returns true $\implies$ S' is a sub-sequence of S and the algorithm returns false $\implies$ S' is not a sub-sequence of S. But the definition of a sub-sequence of $\{a\}$, is a sequence $\{b\}$ such that $b_k = a_{l_k}$, where $l_1 < l_2 < ... < l_k$. It follows from this definition that if the algorithm returns true then S' is a sub-sequence of S. This is because i is incremented in every iteration and so the successive indices of S which match with an element in S' must be increasing. Also the algorithm only returns true if j is one larger than the max index of S' and j is only incremented if an element in S' matches with an element in S. Hence every element in S' must have been matched with an element in S.

In order to prove part two, we must first clarify the following detail. Suppose that S' is a sub-sequence of S. This means that we can reconstruct S' by extracting elements from S at certain indices while preserving their order. For example, if $s_{l_0}, ..., s_{l_{m-1}}$ is the same as the sequence S', $s_{l_0}, ..., s_{l_{m-1}}$ are all in S, and $l_0 < ... < l_{m-1}$, then $s_{l_0}, ..., s_{l_{m-1}}$ is one such ordered list of extracted elements from S that can construct S'. However, they are not necessarily the only ordered elements from S that can do this. Finally define k_j to be the index of S at which the algorithm determines that $s'_j = s_{k_j}$ for all $j = 0, ..., m-1$.

Proposition. *For all executions of the algorithm in which S contains some sub-sequence S', the algorithm will determine, for all $j = 0, ..., m-1$, that $s'_j = s_{k_j}$ where $k_j \leq l_j$.*

Proof. Let the property $P(j)$ be the proposition to be proved.

Show that $P(0)$ is true: Since S' is a sub-sequence of S, every element in S' is in S. Hence there must exist an index k_0 of S such that $s'_0 = s_{k_0}$. The algorithm starts from the smallest index of S and checks for equality of s'_0 until it finds it. Hence $k_0 \leq l_0$.

Show that for $j \geq 1$, $P(j-1) \implies P(j)$: Let j be any integer such that $j \geq 1$ and j is a valid index of S'. By inductive hypothesis the algorithm will find an element k_{j-1} so that $s'_{j-1} = s_{k_{j-1}}$ where $k_{j-1} \leq l_{j-1}$. Now the algorithm will find the first element in S after k_{j-1} which is equal to s'_j. If the algorithm can find such an element then it will be at index k_j. But we know that the algorithm can find such an index because $s_{l_j} = s'_j$ and $k_{j-1} \leq l_{j-1} < l_j$. It follows that $k_j \leq l_j$. □

It immediately follows that if S contains the sub-sequence S', the algorithm will find this sub-sequence in optimal time. Furthermore since we now have that S' is a sub-sequence of S $\implies$ the algorithm returns true, we also have that the algorithm returns false $\implies$ S' is not a sub-sequence of S. It is established that the algorithm is correct and optimal.