Max Subsequence Sum Correctness Proof

Here’s the code from the Subsequence Sum Correctness video:

```python
def max_sum(A):
    ",",
    "Pre: A is a list of integers (positive and negative)"
    "Post: Returns a tuple (a,b) where"
    "a is the max subsequence sum in A, and"
    "b is the max sum of the PREFIXES of A (includes [])"
    ",",
    "if len(A) == 0:
        return (0,0)
    else:
        max_so_far, max_prefix = max_sum(A[1:])
        if A[0] + max_prefix > 0:
            new_max_prefix = max_prefix + A[0]
        else:
            new_max_prefix = 0
        
        new_max_so_far = max(max_so_far, new_max_prefix)
        return (new_max_so_far, new_max_prefix)
```

Let’s say that there are two program paths, with the latter divided into two subpaths. (Alternately, you could say that there are three program paths.)

On the first path, \(A\) is empty, so \(\text{len}(A) == 0\). In this case, the only subsequence and only prefix of \(A\) are the empty subsequence, which has a sum of 0. So returning (0,0) on line 10 satisfies the postcondition.

Now consider the second program path, when the else branch is executed (so \(\text{len}(A) > 0\)). First we deal with the recursive call, which is made on \(A[1:]\), which is a list of integers of length \(\text{len}(A) - 1 < \text{len}(A)\). So we can conclude that the recursive call terminates, and using its postcondition, we know that after line 12, \(\text{max}_\text{so}_\text{far}\) has the value of the maximum subsequence sum in \(A[1:]\), and \(\text{max}_\text{prefix}\) is the maximum prefix sum of \(A[1:]\).

The next part of the code (lines 13-16) calculates the maximum prefix sum of \(A\). There are only two possibilities for the max prefix: either it doesn’t include \(A[0]\), in which case it must be empty, or it does, in which case it contains \(A[0]\) and then the max prefix of \(A[1:]\). Since the empty list has a sum of 0, and the latter case has a sum of \(A[0] + \text{max}_\text{prefix}\), after this block occurs, \(\text{new}_\text{max}_\text{prefix}\) contains the value of the max prefix of \(A\).

Next, line 18 computes the max subsequence sum in \(A\). There are two possibilities for the maximum subsequence: either it doesn’t contain \(A[0]\), in which case it is contained in \(A[1:]\) (and so its value is \(\text{max}_\text{so}_\text{far}\)) or it’s a prefix of \(A\), in which case its value is \(\text{new}_\text{max}_\text{prefix}\) (by the discussion from the previous paragraph). Setting \(\text{new}_\text{max}_\text{so}_\text{far}\) to the max of these two values results in this variable containing the maximum subsequence sum of \(A\), and so the algorithm returns the correct values specified by the postcondition at line 20.

By the way, lines 13-16 could be replaced by the single line \(\text{new}_\text{max}_\text{prefix} = \text{max}(A[0] + \text{max}_\text{prefix}, 0)\).