Describe an optimal dynamic programming algorithm to find the maximum product of a contiguous sequence of positive numbers $A[1..n]$. For example, if $A = (.1, 17, 1, 5, .05, 2, 4, 1, .7, .02, 12, .3)$, then the answer would be 85 because of the subsequence $(17, 1, 5)$.

**SOLUTION:** For every $i$, we compute $MaximumProduct[i]$, the maximum product of a contiguous sequence of positive numbers in $A[1..i]$ whose rightmost member is $A[i]$. The dynamic programming formulation is

$$MaximumProduct[i] = \max \{ A[i], MaximumProduct[i-1]*A[i] \}$$

This translates to the program

```plaintext

for i ← 2 to n do

    MaximumProduct[i] = \max \{ A[i], MaximumProduct[i-1]*A[i] \}

return \max \{ MaximumProduct[i] \}
```

Show that your algorithm is optimal.

The algorithm above works in linear time. To show that it is optimal, assume there is an algorithm to solve the problem which works in sublinear time, that is, time in $o(n)$. If an algorithm finds the correct answer for some array $A[1..n]$ but doesn’t examine every element, then some element, say $A[j]$, is not examined. Consider a new array $A^*$ which is identical to $A$ except that $A^*[j] = \infty$. Because $A^*[j]$ was not examined before, it is not examined with the change. The algorithm must give the same answer, which is incorrect since the new answer must contain $A^*[j]$. This contradiction shows that a sublinear time algorithm can not exist.