University of Washington
Department of Computer Science and Engineering
CSE 417, Winter 2020

February 25, 2020

Homework 7 problem 3

Problem 3 (10 points) Word segmentation:

(This problem is based on problem 5 on Page 316 of the text without the excessive verbiage.) The word segmentation problem is: given a string of characters $Y = y_1 y_2 \dots y_n$, optimally divide the string into consecutive characters that form words. (The motivation is that you are given at text string without spaces and have to figure out what the words are. For example, *"meetateight"* could be *"meet ate ight"*, *"me et at eight"* or *"meet at eight"*.) The problem is to find best possible segmentation. We assume we have a function *Quality* which returns an integer value of the goodness of a word, with strings that correspond to words getting a high score and strings that do not correspond to words getting a low score. The overall quality of a segmentation is the sum of the qualities of the individual.

Give an dynamic programming algorithm to compute the optimal segmentation of a string. You can assume that calls to the function *Quality* take constant time and return an integer value.

Answer:

algorithm:

```
#define an two array with size n
opt = [0 for i in range(n+1)]
seg = [0 for i in range(n+1)]

for j in range(n):
    tempOpt = 0
    for k in range(1, j):
        if opt(k-1) + Quality(k-1,j) > tempOpt:
            tempOpt = opt(k-1) + Quality(k-1,j)
            seg[j] = k - 1
    opt[j] = tempOpt

p = seg[n]
while p != 0:
    # add a partition point at position
    split(word, p)
    p = seg[p]
```

Correctness proof by induction:

base case:

For $j = 0$:
When there are split with zero length separation, the optimal split and quality score is 0.

induction hypothesis:
For $j > 0$:
opt[j] = max(opt[k-1] + quality(k,j)) where $1 < k < j$
Since at each character at position j, there are multiple choices (a various number depend on j's position) to compare and get the optimal quality split. Therefore the optimal value at j (opt[j]) is equal to the separation with best quality on the non-fixed part (1 to j).
Since the function opt[j] always correctly choose the optimal result for position j, and opt[k] is built opon value of left partition position opt value, hence by always choosing the optimal quality, the function garantees to correctly find the local optimal quality. Adding to final accumulated optimal total quality of all possible partition position.

complexity:
Since on each subproblem, the algorithm only traces and compares only the left value with constant time per opearation, and each inner iterator increase the subproblem size by 1, there is a arithmetic sequence of 1, 2, 3, ..., n-1, n, therefore the time complexity is $(n^2 + n)/2 = O(n^2)$.