CSE373 Assignment 3

Aditya Balwani
SBUID: 109353920

April 19, 2016

1 Optimal Network Transmission Combination

Lets assume that $Request[i]$ contains 3 variables which are startTime, endTime and size.
Lets assume that the list is sorted by the start time of each element.
At each position in the array, the maximum non overlapping sum is the maximum value out of the max overapping sum till previous index or the max overlapping sum of current index where the max sum uptil index j such that stop time doesn't overlap with current element.

$$T[i] = \begin{cases} Request[i].size & \text{if } (i == 0) \\ max(T[i-1], T[j] + Request[i].time) & \text{s.t. } j = max_{j \in [0,i-1]} Request[j] \text{ does not overlap } Request[i] \end{cases}$$

The asymptotic running time is $O(nlogn)$
The asymptotice space requirement is $O(n)$

2 Mining For Gold

At each position we can either go up or right. So the maximum gold value that can be can be found is the maximum value out of the max gold value obtained by going up and the max gold value obtained by going right plus the gold value of the current spot. If we can't go up then the max value is obtained by going right and if we can't go right then max value is from going up. At the top right corner the max value is the value of the position itself. At the end we return the gold vale of T[0,0] where

$$T[i,j] = \begin{cases} Gold[i,j] & \text{if } (i == M-1 \land j == N-1) \\ Gold[i,j] + T[i,j+1] & \text{if } (i == M-1) \\ Gold[i,j] + T[i+1,j] & \text{if } (j == N-1) \\ Gold[i,j] + max(T[i,j+1], T[i+1,j]) & \text{otherwise} \end{cases}$$

The asymptotic running time is $O(mn)$
The asymptotice space requirement is $O(mn)$

3 Maximum Product

For an array A, to find the subseries with the maximum product, we fill the dynamic programming table using the rules :

$$T[i,j] = \begin{cases} A[i] & \text{if } i == j \\ A[i] * T[i-1,j] & \text{if } i > j \\ A[i] * T[i+1,j] & \text{if } i < j \end{cases}$$

where i and j are indecis in the array

The asymptotic running time is $O(n^2)$
The asymptotice space requirement is $O(n)$

4 String Decomposition

Lets say a and b are the 2 strings and m and n are the length of Our recursive function takes in all 3 strings: a, b and c. If a and b are both of length 0 then we return false. If the first lett of c is equal to the first letter of a then we we return the value returned by the recursive call containing a[1:n],b and c[1:m+n] . If not then we compare the first letter of b and c and if equal then the make the recursive call with a, b[1:n], c[1:m+n]. If that is also unequal then we return false.

$$T[i,j] \begin{cases} \text{false} & \text{if } m \text{ and } n \text{ are both } 0 \\ T[i-1,j] & \text{if } a[0] == c[0] \\ T[i,j-1] & \text{if } b[0] == c[0] \\ \text{false} & \text{otherwise} \end{cases}$$

The asymptotic running time is $O(mn)$
The asymptotice space requirement is $O(mn)$

Psuedocode:

```
def isStringDecomposition(a, b, c, i, j):
    if(len(a) == 0 && len(b) == 0):
        T[0,0] = False
        return false
    if(a[0] == c[0]):
        if(T[i-1, j] != None):
            return T[i-1, j]
        else:
            success = isStringDecomposition(a[1:i], b, c[1:i+j], i-1, j)
            T[i-1, j] = success
            return success
    elif(b[0] == c[0]):
        if(T[i, j-1] != None):
            return T[i, j-1]
        else:
            success = isStringDecomposition(a, b[1:j], c[1:i+j], i, j-1)
            T[i, j-1] = success
            return success
    T[i, j] = False
    return False
```

5 Two salesmen

The problem we're trying to solve is to split A into B and C such that

$$\min\{|distance(B) - distance(C)|\}$$

To solve this question we first need to fill in the dynamic programming array. We define the array T such that T[i,j] contains 1 if there is a subset of $A_0..A_i$ such that its sum is equal to j.
The rule for filling in the table is as follows for this is as follows :

$$T[i,j] = \begin{cases} 1 & \text{if some subset of } A_0..A_i \text{ has the sum } j \\ 0 & \text{otherwise} \end{cases}$$

This gives us the following recursive structure

$$T[i,j] = max(T[i-1,j], T[i-1, j - A[i]], 0)$$

Now we try to find the ideal possible sum which would be :

$$S = distance(A)/2$$

We now try find minimum value $i \leq S$ s.t.

$$\min_{i \leq S}\{S - i : T[n,i] = 1\}$$

Once we find such an i, we have the best sub we can obtain for B as i and the sum of C is 2S-i ie.

$$distance(B) = i \qquad\qquad distance(C) = 2S - i$$

This is the most optimal split sum we can get and we can get the actual sets by retaining backpointers while filling in T

The time complexity of this problem is $O(n^2k)$
The space complexity is $O(n^2k)$