Pablo Acuna
CSCI 4020
Anshelevich

Computer Algorithms Homework 3

6.

a. Give an efficient algorithm to find a partition of a set of words W into valid lines, so that the sum of the squares of the slacks of all lines (including the last line) is minimized.

Subproblems: OPT[j] = the optimal solution on the set of words from w_1, ... w_j such that the sum of the squares of the slacks are minimized.

Define: $S_{i,j}$ to be the slack between w_i, ..., w_j (slack summation defined in the question). If $S_{i,j} >$ L we will make that $S_{i,j} = \infty$

Recursion Relation: OPT[j] = $min_{1 \leq i \leq j}$ ($S_{i,j}^2$ + OPT[i-1])

This recursion relation works very similarily to segmented least squares in that we compute all subsequences and find the line from i to j that has the lowest cost and previous cost, that will be our optimal solution at word j.

Psuedocode

Precompute all $S_{i,j}$
OPT[0] = 0
for j = 1 to n
    OPT[j] = $min_{1 \leq i \leq j}$ ($S_{i,j}^2$ + OPT[i-1])
return OPT[n]

Analysis : If like in class we treat the precomputated values to take O(n^2) then that will be our runtime. That is because we have n subproblems that take n time to compute. Therefore, the loop also takes O(n^2) time and overall runtime is proved to be O(n^2).

Pablo Acuna
CSCI 4020
Anshelevich

Computer Algorithms Homework 3

9.

a. Give an example of an instance with the following properties.
– There is a "surplus" of data in the sense that $x_i > s_i$ for every i.
– The optimal solution reboots the system at least twice.

Day	1	2	3	4	5
s_i	100	1	1	1	1
x_i	101	101	101	101	101

In this example, we will reboot twice in order to get the optimal solution. No reboot will yield 104 TB, one reboot will yield 202 TB, and two reboots on day 2 and 4 will yields 300 TB.

b. Give an efficient algorithm that takes values for $x_1, x_2, \ldots, x_n$ and $s_1, s_2, \ldots, s_n$ and returns the total number of terabytes processed by an optimal solution.

Subproblems: OPT[i,j] = the max amount of data processed that can be done starting from day i through day n and the last reboot happened j days prior.

Recursion Relation: OPT[i,j] = max(OPT[i+1,1], min(x_i, s_j) + OPT[i+1,j+1])

This relation is correct because we have two choices, either reboot on a day or process data that day. Rebooting corresponds to OPT[i+1,1] because we go back to j = 1. And processing corresponds to min(x_i, s_j) + OPT[i+1,j+1] because we add the data processing on that day to the previous day's max. For this to work though we will need to fill up the table from i=n-1 to 1 and j=1 to n.

Psuedocode

```
for j = 1 to n
    OPT[n,j] = min(x_n, s_j)
for i = n-1 to 1
    for j = 1 to i
        OPT[i,j] = max(OPT[i+1,1], min(x_i, s_j) + OPT[i+1,j+1])
return OPT[1,1]
```

Analysis : There are n^2 subproblems that take constant time to compute therefore the time complexity of the algorithm is O(n^2).