6.006 Homework	**Problem set 6**	# 1 – Placing Parentheses
Xola Ntumy		April 29, 2011
Collaborators: None		

(a) The set of subproblems that one would use to solve this problem would be a set of substrings that give an optimal ordering to perform the operations.

b The recurrence would be the maximum of the total number of subproblems.

C(i,j) = max((x1 oper(i) (x2 oper(j) x3)), ((x1 oper(i) x2) oper(j) x3)) such that i = index of ith operation, j = index of jth operation and oper(i) is the ith operation and op(j) is the jth operation.

c The number of subproblems is n*n. The time for each problem however is O(1) therefore the total running time would be O(n*n)

6.006 Homework	**Problem set 6**	# 2 – Text Formatting
Xola Ntumy		April 29, 2011
Collaborators: None		

a The subproblem would be to minimize the badness of each line as well the remaining space on the last line.

b The recurrence would then the minimum of the sum of each line's badness and the remaining space on the last time, ie. the sum of all the subproblems.

c The running time would be the the time it takes to do a subproblem by the number of subproblems. We have n subproblems which can be solved in O(1) so the running time is O(n).

d We could fill up each line with words up until just before or right after the target line length has been reached and check whether or not stopping at the word before we reach the target line length or stopping after that word will produce minimum badness for that line until we run out of words. We then minimize the max badness for the current line by taking words from the previous line until badness is minimized. We can do this for every two consecutive lines starting from the bottom and going up by recursion.

6.006 Homework	**Problem set 6**	# 3 – Dungeons and Dragons and Dynamic Programming
Xola Ntumy		April 29, 2011
Collaborators: Wesam Manassra		

Solving this problem will involve the use of the 2 tables through the bottom-up approach. The first table would keep track of the net HP(i,j) and the other keeps track of the which previous square gave the current square its current total. The subproblem is finding the optimal path that will make the sum of XPs along the path maximum without HP(i,j) ever being less than or equal to 0. We can guess that if we start from bottom of the table and work towards the position which we are at, we will obtain max XP and squares that have not been traversed yet are set to negative infinity HP points.

The recurrence relation:

XP(i,j,h) = max(XP(i, j-1, h+H(i,j-1)), XP(i-1, j-1, h+H(i,j-1)), XP(i+1, j-1, h+H(i,j-1))) where H(i,j) is the sum of hitpoints

I propose we start by way of the bottom-up approach for all n-starting positions. The bottom row XP points only based on their own hp and xp, and not the previous rows since they are the starting position.

On each subsequent row, look at the XP points of the previous row and add the xp points of the current box to the max of the total XP points of the previous rows (only for the grids that can come to the current grid). Update the two tables accordingly for the entire row and then move up to the next row. If hp(i,j) ever goes reaches 0 or below, discontinue that path. At the end of the algorithm, simple follow the path which produced highest xp.

Since we are dealing with an n*n grid the number subproblems = 0(n*n) Time per subproblem = 0(1)

The running time is therefore subproblems * time per subproblem = 0(n*n)

6.006 Homework	**Problem set 6**	# 4 – Railroads and Gauges
Xola Ntumy		April 29, 2011
Collaborators: Kwadwo Nyarko		

Subproblem: C(s, g1,g2,g3) representing the minimum cost (based on number of different gauges) for the subtree rooted at the node v where v is a switching stations and not a town.

D(g1, g2, g3) = 0 if all three are the same, 1 if is different, 2 if all three are different

The recurrence now becomes:

C(s, g1, g2, g3) = min (D(g1,g2,g3)+C(u1,g1,g4,g5)+ C(u2,g2,g6,g7)+ C(u3,g3,g8,g9))

for every g1,g2,g3 within the constraints of any switching stations connected to a town or a previously minimized switching station.

We first attempt all combinations of labeling Ss: u1,u2,u3. We utilize a spanning tree for this problem as it seems to be the best data structure applicable. We select the combination that gives us the minimum cost and run the algorithm via the bottom-up approach . At each node, we iterate through every possible labeling and store whatever we obtain as the best score for each of them. We then use this result later on in the problem. We will eventually propagate all the solutions up to the root and know all the switching station gauges.

Using the sequence of arrays suggested, by binary search, Ai can be turned to Ai+1 and this will be done in O(logn) time. Since we have n arrays the total running time for this algorithm would be O(nlogn). The LIS(z) would simply the length of the An array.