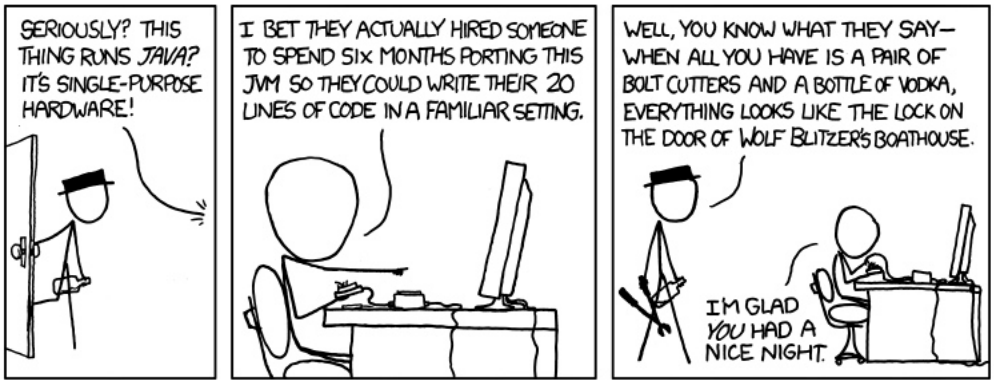
CS302 - Assignment 13

Due: Thursday, April 11 at the beginning of class
Hand-in method: paper



http://xkcd.com/801/

Solve each of the problems below using dynamic programming. You **do not** need to provide pseudocode, but make sure your algorithm descriptions are clear and understandable. You will be graded on efficiency.

1. **[13 points]** Given a sequence of characters $s_1, s_2, ..., s_n$ we'd like to determine an algorithm that finds the longest subsequence that is the same whether read left to right or right to left. For example, given the sequence

   $$A, C, G, T, G, T, C, A, A, A, A, T, C, G$$

   one of the longest such subsequence is G, C, A, A, A, A, C, G and two non-optimal such subsequences are G, T, T, G and A, T, A. Recall that a subsequence is not necessarily contiguous.

   (a) **[3 points]** Write the recursive relationship for the problem (e.g. for fibonacci it was $Fib(n) = Fib(n-1) + Fib(n-2)$).

   (b) **[8 points]** Describe a DP algorithm for this problem.

   (c) **[2 points]** State the running time of your approach.

2. **[18 Points]** You've been asked to design an algorithm for deciding who to invite to a company party. The structure of the company can be described by a tree as follows: the CEO is at the

root, below the root are supervisors, below them are managers, below them are team leaders, etc., etc., until you get down to the leaves (summer interns). The tree is not necessarily binary; some non-leaf nodes may have one "child", others two, and others even more.

To make the party fun, we won't invite an employee along with their immediate boss (their parent in the tree). In addition, each person has been assigned a positive real number called their *coefficient of fun*. The goal is to invite employees so as to maximize the total sum of the coefficients of fun of all invited guests, while not inviting an employee with his or her immediate boss.

(a) **[4 points]** Describe a recursive algorithm for this problem (i.e. non-dynamic programming). Assume that the tree is represented as a collection of nodes with links from parents to children and also from children to parents. The tree is passed to you by giving you a reference to the root.

(b) **[3 points]** Describe the data structure that you plan on using to store the intermediary solutions, specifically what it is, how it is indexed and what each entry means.

(c) **[8 points]** Describe a DP algorithm for this problem. You may assume that each of the n nodes in the tree has a unique number between 1 and n associated with it. You may also assume that you have a function that will give you a list (array or linked list) of all of the leaves in the tree in time $O(n)$.

(d) **[3 points]** State the running time of your approach.