Ally Smith

Homework 5

1. A certain string processing language allows the programmer to break a string into two pieces. It costs n units of time to break a string of n characters into two pieces, since this involves copying the old string. A programmer wants to break a string into many pieces, and the order in which the breaks are made can affect the total amount of time used. For example, suppose we wish to break a 20-character string after characters 3, 8, and 10. If the breaks are made in left-right order, then the first break costs 20 units of time, the second break costs 17 units of time, and the third break costs 12 units of time, for a total of 49 steps. If the breaks are made in right-left order, the first break costs 20 units of time, the second break costs 10 units of time, and the third break costs 8 units of time, for a total of only 38 steps.

   Give a dynamic programming algorithm that takes a list of character positions after which to break and determines the cheapest break cost in $O(n^3)$ time.

   (a) **Recursive Idea:** Using the example from above, we can imagine a scenario in which we break up our string at the 10th index first for a cost of 20, leaving us with the remaining 10 characters to be split at indexes 8 and 3. We can continue doing this until we have no more splits left to make.

   (b) **Recursive Relation:** The relation can be defined as follows, where $cost(i, j)$ is the cost to split a string starting at index i and going to j.
   $cost(i, j) = \min\{\text{length of substring}+cost(i, k) + cost(k, j), i < k < j\}$
   For all $j \leq i+1$, $cost(i, j) = 0$ since we cannot split the string any further. This is our base case.

   (c) **Table:**

$\frac{i}{j}$	0	1	2	3	...	$n-1$	n
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2		0	0	0	0	0	0
3			0	0	0	0	0
...				0	0	0	0
$n-1$					0	0	0
n						0	0

2. The traditional world chess championship is a match of 24 games. The current champion retains the title in case the match is a tie. Each game ends in a win, loss, or draw (tie) where wins count as 1, losses as 0, and draws as 1/2. The players take turns playing white and black. White has an advantage, because they move first. The champion plays white in the first game. They have probabilities w_w, w_d, and w_l of winning, drawing, and losing playing white, and has probabilities b_w, b_d, and b_l of winning, drawing, and losing playing black.

   (a) Write a recurrence for the probability that the champion retains the title. Assume that there are g games left to play in the match and that the champion needs to win i games (which may end in a $\frac{1}{2}$)

   (b) Based on your recurrence, give a dynamic programming algorithm to calculate the champion's probability of retaining the title.

   (c) Analyze its running time for an n game match.