Homework 3 (15 pts)[[1]](#footnote-1)

1. [9 pts]: Suppose that you have *n* dice (normal, six-sided dice with sides labeled 1, 2, 3, 4, 5, 6). Calculate the number of ways to get the dice to sum to *d* – that is, to show a total of *s* dots - when you roll them all at once.

Part A: [3 pts] Suppose numDots(n,d) is the number of ways to show *d* dots when you show *n* dice. Write a recurrence that expresses the number of ways to show *d* dots on *n* dice as a function of the number of dots shown on *n­*-1 dice. Note that as in the Edit Distance problem, this will have multiple cases.

***Comment/hint:***

One of the cases for the recurrence will say that numDots(n,d) = 0 if d < n. Another case will say that numDots(n,d) = 1 if d = n.

***Solution:***

Part B: : [3 pts] Develop your recurrence into pseudocode. Note that this is pseudocode, it doesn’t need to be syntactically correct Java or any other language.

***Solution:***

Part C: : [3 pts] Modify the pseudocode to use bottom-up dynamic programming so that you don’t make duplicate calls to handle overlapping subproblems.

***Solution:***

1. [4 pts]: Suppose that you have a set of jobs you can do for pay. Assume that you work *h* hours per day, and each job *j* takes time *tj* and gives you revenue *rj*. Your goal is to maximize the amount of revenue that you can earn in one day.

Write the recurrence that describes the maximum possible revenue recursively. Call the method maxrev(J), where J is the set of jobs J = {j1, j2, …, jn}.

***Solution:***



1. [4 pts] Given a text consisting entirely of letters with no punctuation or spacing, divide that text into words such that there are no extraneous letters left over. For instance, this is Lincoln’s Gettysburg address with all spaces and punctuation and capitalization removed.

fourscoreandsevenyearsagoourfathersbroughtforthonthiscontinentanewnationconceivedinlibertyanddedicatedtothepropositionthatallmenarecreatedequalnowweareengagedinagreatcivilwartestingwhetherthatnationoranynationsoconceivedandsodedicatedcanlongendurewearemetonagreatbattlefieldofthatwarwehavecometodedicateaportionofthatfieldasafinalrestingplaceforthosewhoheregavetheirlivesthatthatnationmightliveitisaltogetherfittingandproperthatweshoulddothisbutinalargersensewecannotdedicatewecannotconsecratewecannothallowthisgroundthebravemenlivinganddeadwhostruggledherehaveconsecrateditfaraboveourpoorpowertoaddordetracttheworldwilllittlenotenorlongrememberwhatwesayherebutitcanneverforgetwhattheydidhereitisforusthelivingrathertobededicatedheretotheunfinishedworkwhichtheywhofoughtherehavethusfarsonoblyadvanceditisratherforustobeherededicatedtothegreattaskremainingbeforeusthatfromthesehonoreddeadwetakeincreaseddevotiontothatcauseforwhichtheygavethelastfullmeasureofdevotionthatweherehighlyresolvethatthesedeadshallnothavediedinvainthatthisnationundergodshallhaveanewbirthoffreedomandthatgovernmentofthepeoplebythepeopleforthepeopleshallnotperishfromtheearth

Converting this into real English text is a good job for a computer. But it’s not quite as simple as saying “find the first word, set it off, then continue”. If you tried that with the Gettysburg address, you’d have a problem because you would start out:

fourscoreandseven 🡪 four scoreandseven 🡪 four score andseven 🡪 four score an dseven

But although “an” is a word, it’s not the *right* word for this text (which is “and”). The problem is that some words are prefixes of other words. There is no word that will start with “dseven…”. One way to handle this is to use dynamic programming.

Suppose that L[1…n] is an array of letters. We want to divide these letters up into an array of words W (the size of the array isn’t known in advance) so that each element of the array of words is an English language word, and there are no extraneous letters.

Define variables and write the recursive formula that uses dynamic programming to figure out an array W such that each element of W is an English language word, with no letters left over. Assume that you have a Boolean function isWord() available to you that will tell if you any given string of letters is or is not a word (but you won’t need this as part of your formulation). You also know that the longest word in the dictionary has length *k* (it’s dictionary-dependent).

You can express this as an equation, with pseudocode, or in English Language prose, whichever you prefer.

***Solution:***

1. Note that 17 points are possible, so basically this homework includes a little extra credit. [↑](#footnote-ref-1)