Recursion - More Magic (or not)

Due date: by the end of the recitation session

Introduction

The objective of this lab is to get more practice on solving problems using recursion. Some of the problems that you will look at can be solved using iterations (and some of them should in practice be solved using iterations). But in order to get more practice on recursion, we will look at their recursive solutions.

Worksheet: https://goo.gl/ElsOsv

Part 1

The binary search has a much better performance than a linear search (i.e., it finds things faster) when searching in sorted lists of elements. The general idea if a binary search is that it eliminates half of the remaining elements on each comparison, thus reducing the remaining search space much faster than a linear search.
Here is a general outline of the binary search algorithm. Assume that `key` is the element that we are searching for.

1. If the array is not empty, pick an element in the middle of the current array. Otherwise, go to step 5.
2. If that element is smaller than the `key`, discard all elements in the left half of the array (including the element that was just used) and go back to step 1.
3. If that element is greater than the `key`, discard all elements in the right half of the array (including the element that was just used) and go back to step 1.
4. If that element is equal to the `key`, return its index.
5. If the array is empty, the `key` is not there.

There are both, iterative and recursive implementations of this algorithm.

Consider the following array of integers. For each "search" indicate which of the array locations will be visited (their values compared to the `key`) before the binary search finds the desired element or is able to declare that it is not there. Specify how many comparisons would have been performed in a linear search. Specify how you determine the middle element.

0	1	2	3	4	5	6	7	8	9	10
4	12	15	17	21	22	30	35	50	70	73

Search for 22, 73, 35, 12, 21.
Enters your answers in the worksheet.

Part 2

In this part we will revisit the CodingBat website to attempt some of the more advanced recursion problems. .

The website tests your code using many inputs. Try to write code solutions to the following problems. Once you have the correct version (that passes all the tests) or an almost correct version (that might be failing some of the tests), take a screenshot of the screen and paste the image in the worksheet.

Here are some problems that you should look at. You do not need to solve all of them! Concentrate on two problems (other than the first warm-up problem) and spend time as a group to try to come up with the best solution that you can.

1. Group Sum, http://codingbat.com/prob/p145416 - this is just a warm up exercise, you can (and probably should) see its solutions on the website.
2. Group Sum 5, http://codingbat.com/prob/p138907
3. Group Sum 6, http://codingbat.com/prob/p199368
4. Group Sum No Adjacent, http://codingbat.com/prob/p169605
5. Split Array, http://codingbat.com/prob/p185204
6. Split Odd 10, http://codingbat.com/prob/p171660
7. Split 5 3, http://codingbat.com/prob/p168295