∽ New CS 473: Algorithms, Spring 2015 ∾
Homework 0

Due Tuesday, January 27, 2015 at 5pm

- **This homework tests your familiarity with prerequisite material:** designing, describing, and analyzing elementary algorithms (at the level of CS 225); fundamental graph problems and algorithms (again, at the level of CS 225); and especially facility with recursion and induction. Notes on most of this prerequisite material are available on the course web page.
- **Each student must submit individual solutions for this homework.** For all future homeworks, groups of up to three students will be allowed to submit joint solutions.

☞ Some important course policies ☜

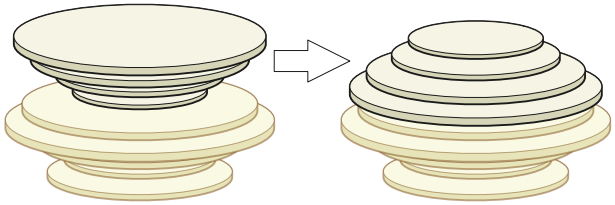
- **You may use any source at your disposal**—paper, electronic, or human—but you ***must*** cite ***every*** source that you use, and you ***must*** write everything yourself in your own words. See the academic integrity policies on the course web site for more details.
- **Submit your solutions on standard printer/copier paper.** Please use both sides of the paper. Please clearly print your name and NetID at the top of each page. If you plan to write your solutions by hand, please print the last four pages of this homework as templates. If you plan to typeset your homework, you can find a LaTeX template on the course web site; well-typeset homework will get a small amount of extra credit.
- **Start your solution to each numbered problem on a new sheet of paper.** Do not staple your entire homework together.
- **Submit your solutions in the drop boxes outside 1404 Siebel labeled "New CS 473".** There is a separate drop box for each numbered problem; if you put your solution in the wrong drop box, we won't grade it. Don't give your homework to Jeff in class; he is fond of losing important pieces of paper.
- **Avoid the Three Deadly Sins!** There are a few dangerous writing (and thinking) habits that will trigger an automatic zero on any homework or exam problem, unless your solution is nearly perfect otherwise. Yes, we are completely serious.
  - Always give complete solutions, not just examples.
  - Always declare all your variables.
  - Never use weak induction.
- Answering any homework or exam problem (or subproblem) in this course with "I don't know" *and nothing else* is worth 25% partial credit. We will accept synonyms like "No idea" or "WTF", but you must write *something*.

See the course web site for more information.

If you have any questions about these policies,
please don't hesitate to ask in class, in office hours, or on Piazza.

---

1. Suppose you are given a stack of n pancakes of different sizes. You want to sort the pancakes so that smaller pancakes are on top of larger pancakes. The only operation you can perform is a ***flip***—insert a spatula under the top k pancakes, for some integer k between 1 and n, and flip them all over.



Flipping the top four pancakes.

   (a) Describe an algorithm to sort an arbitrary stack of n pancakes, which uses as few flips as possible in the worst case. *Exactly* how many flips does your algorithm perform in the worst case?

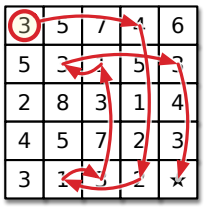
   (b) Now suppose one side of each pancake is burned. Describe an algorithm to sort an arbitrary stack of n pancakes, so that the burned side of every pancake is facing down, using as few flips as possible in the worst case. *Exactly* how many flips does your algorithm perform in the worst case?

2. *[From last semester's CS 374 final exam]* A ***number maze*** is an $n \times n$ grid of positive integers. A token starts in the upper left corner; your goal is to move the token to the lower-right corner.

   - On each turn, you are allowed to move the token up, down, left, or right.
   - The distance you may move the token is determined by the number on its current square. For example, if the token is on a square labeled 3, then you may move the token three steps up, three steps down, three steps left, or three steps right.
   - However, you are never allowed to move the token off the edge of the board. In particular, if the current number is too large, you may not be able to move at all.

   Describe and analyze an efficient algorithm that either returns the minimum number of moves required to solve a given number maze, or correctly reports that the maze has no solution. For example, given the maze shown below, your algorithm would return the number 8.

3	5	7	4	6
5	3	1	5	3
2	8	3	1	4
4	5	7	2	3
3	1	3	2	★



A 5 × 5 number maze that can be solved in eight moves.

[Hint: Build a graph. What are the vertices? What are the edges? Is the graph directed or undirected? Do the vertices or edges have weights? If so, what are they? What textbook problem do you need to solve on this graph? What textbook algorithm should you use to solve that problem? What is the running time of that algorithm as a function of n?]

3. (a) The ***Fibonacci numbers F_n*** are defined by the recurrence $F_n = F_{n-1} + F_{n-2}$, with base cases $F_0 = 0$ and $F_1 = 1$. Here are the first several Fibonacci numbers:

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
0	1	1	2	3	5	8	13	21	34	55

Prove that every non-negative integer can be written as the sum of distinct, non-consecutive Fibonacci numbers. That is, if the Fibonacci number F_i appears in the sum, it appears exactly once, and its neighbors F_{i-1} and F_{i+1} do not appear at all. For example:

$$17 = F_7 + F_4 + F_2, \qquad 42 = F_9 + F_6, \qquad 54 = F_9 + F_7 + F_5 + F_3 + F_1.$$

(b) The Fibonacci sequence can be extended backward to negative indices by rearranging the defining recurrence: $F_n = F_{n+2} - F_{n+1}$. Here are the first several negative-index Fibonacci numbers:

F_{-10}	F_{-9}	F_{-8}	F_{-7}	F_{-6}	F_{-5}	F_{-4}	F_{-3}	F_{-2}	F_{-1}
−55	34	−21	13	−8	5	−3	2	−1	1

Prove that $F_{-n} = -F_n$ if and only if n is even.

(c) Prove that *every* integer—positive, negative, or zero—can be written as the sum of distinct, non-consecutive Fibonacci numbers *with negative indices*. For example:

$$17 = F_{-7} + F_{-5} + F_{-2}, \quad -42 = F_{-10} + F_{-7}, \quad 54 = F_{-9} + F_{-7} + F_{-5} + F_{-3} + F_{-1}.$$

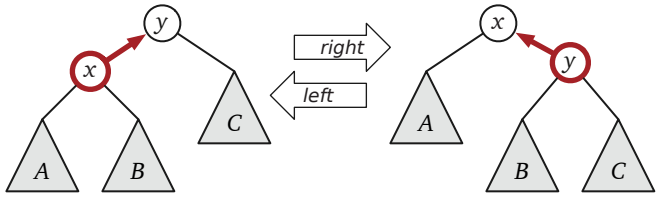
[Hint: Zero is both non-negative and even. Don't even think about using weak induction!]

4. ***[Extra credit]*** Let T be a binary tree whose nodes store distinct numerical values. Recall that T is a ***binary search tree*** if and only if either (1) T is empty, or (2) T satisfies the following recursive conditions:

- The left subtree of T is a binary search tree.
- All values in the left subtree of T are smaller than the value at the root of T.
- The right subtree of T is a binary search tree.
- All values in the right subtree of T are larger than the value at the root of T.

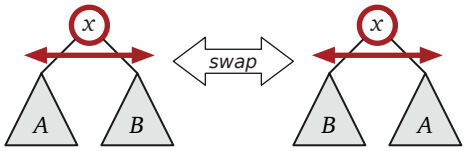
Describe and analyze an algorithm to transform an *arbitrary* binary tree T with distinct node values into a binary search tree, using ***only*** the following operations:

- Rotate an arbitrary node. Rotation is a local operation that decreases the depth of a node by one and increases the depth of its parent by one.



Left to right: right rotation at x. Right to left: left rotation at y.

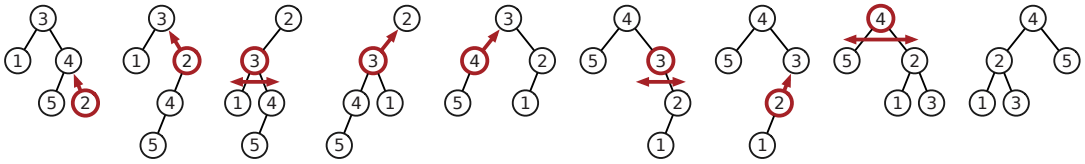
- Swap the left and right subtrees of an arbitrary node.



Swapping the subtrees of x

For both of these operations, some, all, or none of the subtrees A, B, and C may be empty.

The following example shows a five-node binary tree transforming into a binary search tree in eight operations:



"Sorting" a binary tree in eight steps: rotate 2, rotate 2, swap 3, rotate 3, rotate 4, swap 3, rotate 2, swap 4.

Your algorithm cannot directly modify parent or child pointers, and it cannot allocate new nodes or delete old nodes; the ***only*** way it can modify T is using rotations and swaps. On the other hand, you may *compute* anything you like for free, as long as that computation does not modify T. In other words, the running time of your algorithm is *defined* to be the number of rotations and swaps that it performs.

For full credit, your algorithm should use as few rotations and swaps as possible in the worst case. *[Hint: $O(n^2)$ operations is not too difficult, but we can do better.]*

New CS 473 Spring 2015 — Homework 0 Problem 1

Name:

NetID:

(a) Describe an algorithm to sort an arbitrary stack of n pancakes, which uses as few flips as possible in the worst case. *Exactly* how many flips does your algorithm perform in the worst case?

(b) Now suppose one side of each pancake is burned. Describe an algorithm to sort an arbitrary stack of n pancakes, so that the burned side of every pancake is facing down, using as few flips as possible in the worst case. *Exactly* how many flips does your algorithm perform in the worst case?

New CS 473 Spring 2015 — Homework 0 Problem 2

Name:
NetID:

Describe and analyze an efficient algorithm that either returns the minimum number of moves required to solve a given number maze, or correctly reports that the maze has no solution.

New CS 473 Spring 2015 — Homework 0 Problem 3

Name:

NetID:

(a) Prove that every non-negative integer can be written as the sum of distinct, non-consecutive Fibonacci numbers.

(b) Prove that $F_{-n} = -F_n$ if and only if n is even.

(c) Prove that *every* integer—positive, negative, or zero—can be written as the sum of distinct, non-consecutive Fibonacci numbers *with negative indices*.

Do not use weak induction.

New CS 473 Spring 2015 — Homework 0 Problem 4

Name:

NetID:

Submit your solution in the drop box for problem 1 (but ***don't*** staple problems 1 and 4 together).

[Extra credit] Describe and analyze an algorithm to transform an *arbitrary* binary tree with distinct node values into a binary search tree, using ***only*** rotations and swaps.
