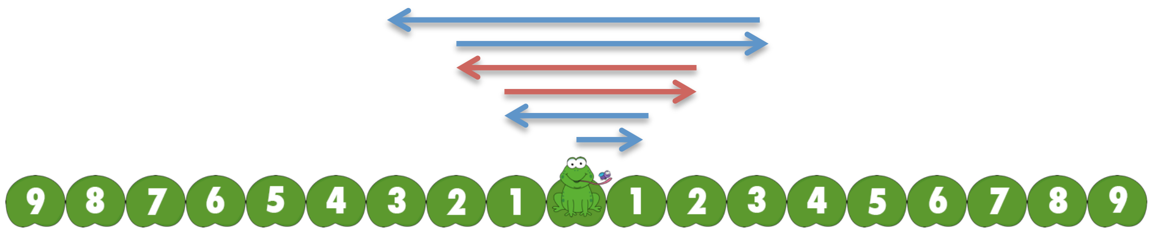
Data Structures II - Assignment 1

Micah Stairs and William Fiset
COMP 2631 – Winter 2015
Dr. Keliher

A long time ago, in a pond far, far away, there was a row of lily pads arranged in a perfectly straight line, stretching forever in both directions. On one of these lily pads sat a frog. This frog knew that one of the lily pads was a magic lily pad, and that hopping onto this magic lily pad would transform the frog into a powerful rock star named Prince. The problem was that all the lily pads looked exactly the same, so there was no way to tell which one was the magic lily pad except to hop onto it. The only action the frog can (repeatedly) perform is to hop from its current lily pad to one of the two adjacent lily pads. Suppose the magic lily pad is exactly n hops away from the frog's current position, but the frog does not know the value of n nor the direction in which the magic lily pad lies. How can the frog perform O(n) hops and always be guaranteed to find the magic lily pad?

This is a clever question. Since the lily pads are arranged in a line stretching forever in both directions (and the frog does not know how far away this magic lily pad is), he cannot simply hop in one direction then turn around once he has reached the end and check the lily pads in the other direction (which would take 3n hops, or O(n) hops). After all, you will never reach the end, since it is infinitely far away, so this technique will clearly not work.

The naïve approach to solving this problem, then, is to have the frog go back and forth, increasing the depth by 1 each time:

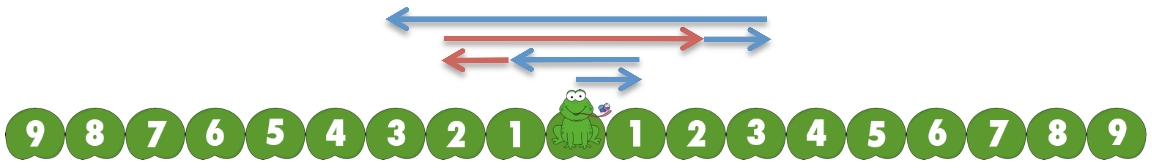


This method does not satisfy our criteria for an O(n) algorithm. This naïve approach takes ***2n² + n*** *steps, which gives us a complexity of* ***O(n²)****. Lets see if we can make our frog smarter than O(n²)…*

# Lily pads deep (n)	# Total Hops
1	$2(1)^2 + 1 =$ **3**
2	$2(2)^2 + 2 =$ **10**
3	$2(3)^2 + 3 =$ **21**
4	$2(4)^2 + 4 =$ **36**
5	$2(5)^2 + 5 =$ **55**

Data Structures II - Assignment 1

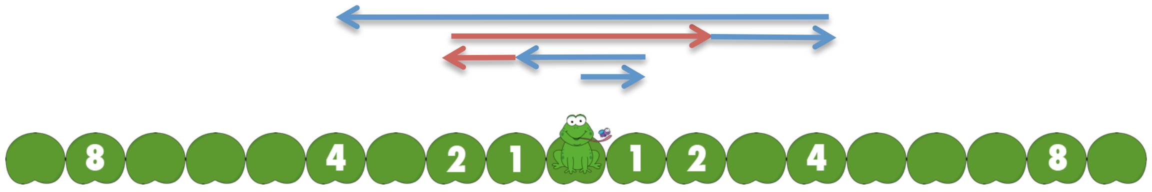
To greatly reduce the number of steps taken, he could take one more step deeper before turning around and heading the other direction, thus eliminating the need to go back and forth so much. This technique is clearly depicted below:



This technique yields the results on the right. It turns out that the relationship between the total hops and n is ***n^2+2n****, which still gives us a complexity of* ***O(n^2)****. Let's continue looking…*

# Lily pads deep (n)	# Total Hops
1	$(1)^2 + 2(1) =$ **3**
2	$(2)^2 + 2(2) =$ **8**
3	$(3)^2 + 2(3) =$ **15**
4	$(4)^2 + 2(4) =$ **24**
5	$(5)^2 + 2(5) =$ **35**

To reduce this algorithm down to a complexity of O(n), we need to make one small modification. Instead of simply incrementing the depth by 1 each time, we should double it.



With the exception of the first time, each iteration of this procedure takes 2^i+2^{i-2} steps. It turns out that the relationship between the total number of hops and n is ***5n-2****, therefore this algorithm has a complexity of* ***O(n)****!*

Iteration (i)	# Lily pads deep (n)	# Total Hops
1	1	**3**
2	2	$3 + (2^2 + 2^0) =$ **8**
3	4	$8 + (2^3 + 2^1) =$ **18**
4	8	$18 + (2^4 + 2^2) =$ **38**
5	16	$38 + (2^5 + 2^3) =$ **78**