6.006 Homework | **Problem Set 1, Part A** | # 1 – Asymptotic Growth
6.006 TA | February 18, 2010
Collaborators: none

(a) Group 1: f_2, f_4, f_1, f_3

(b) Group 2: f_2, f_1, f_3, f_4

(c) Group 3: f_3, f_1, f_2, f_4

6.006 Homework	**Problem Set 1, Part A**	# 2 – Binary Search
6.006 TA		February 18, 2010
Collaborators: none		

(a) The running time is $\Theta(\log n)$. Each iteration of the loop takes $\Theta(1)$ time. Each iteration either finds an occurence of `item` and is the last iteration of the loop, or it halves the size of the problem. Halving can only be done $O(\log n)$ times.

(b) The algorithm has bounded running time, so it always terminates.

It remains to prove that it always gives a correct solution.

- The algorithm returns `True` only if it finds an occurence of `item`. So if `item` is not present, it will return `False`.
- Suppose now that `item` is present in the array. In this case, we prove that the algorithm maintains the invariant that one of `alist[first]`, `alist[first+1]`, ..., `alist[last]` equals `item`. The invariant clearly holds before the first iteration. Then, in each iteration, if `item` is not found, then using the fact that the array is sorted, the algorithm compares `item` to `alist[midpoint]`, and correctly decides which of `alist[first]`, ..., `alist[midpoint -1]` and `alist[midpoint+1]`, ..., `alist[last]` contains `item`.

  Since the above invariant is maintained, the algorithm eventually finds `item`.

(a)

0	3	0	6	0	0	0
0	4	5	6	0	0	0
0	3	0	6	0	0	0
1	2	1	6	0	0	0
0	0	0	6	0	0	0
0	0	8	7	0	0	0
0	0	0	6	0	0	0

The algorithm makes a bad decision and focuses on the top left 3×3 quadrant, which does not have a peak. The largest number in it is 5, but the boundary 6, which was removed earlier, is its neighbor.

(b) The modified algorithm always focuses on a subarray that has a number that is greater than any of the boundary numbers that have been removed, and finds a number greater than those removed boundary elements. This way, it makes sure that the number it finds is a peak for the entire array, not just for the subarray.