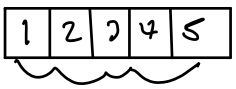
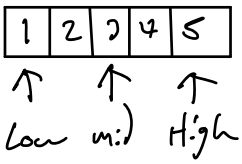
Michael Cooper 17C Midterm - work for 1-3

Linear Search → 

n = 5 O(n)
val = 5

Binary Search → 

Each iteration the problem reduced by half or $\log_2 n$ which can be written as $O(\log n)$

Bubble Sort

1	4	3	2	0	1	6	2	3	0	1	4	11	0	1	2	3	4
2	3	4	2	0	1	7	2	0	3	1	4	12	0	1	2	3	4
3	3	2	4	0	1	8	2	0	1	3	4						
4	3	2	0	4	1	9	2	0	1	3	4						
5	3	2	0	1	4	10	0	1	2	3	4						

n-1 swaps the first loop
n-2 swaps the second loop
n-3 swaps the third loop or n
$O(n^2)$

Selection Sort

$O(n^2)$ Same logic as bubble sort, check for smallest number and swap then continue

Simple Vector push

$O(n)$ - either arraySize ≠ maxSize and ptr[arraySize++] = val or copy into bigger new array which is $O(n)$ and push new value.

Simple vector efficient new

worst case is O(n) similar to previous version but less likely to occur since memory is doubled.