

AP COMPUTER SCIENCE A – SEARCHING SPEEDS

Linear Search

The following method performs a linear search on an `int []` list of length n .

```
/*
 * Given:  list = a sorted list
 *         num = an integer
 *
 * Return: the index of num in list,
 *         or -1 if num is not in list.
 */
public static indexOf ( int [] list, int num ) {
    /**/
    for ( int i=0; i<list.length; ++i ) {
        if ( list[i] == num ) return i;
    }
    return -1;
}
```

Note the loop is executed, on average, $n/2$ times, so that the time t to sort the list is of order n , i.e., $t \sim O(n)$.

Binary Search

The binary search algorithm searches for a number in a *sorted* list. To see how the algorithm works, consider the `int []` list in the table below, and say that we are looking for the number 44. On the first pass, we are looking through the whole list from $b = 0$ to $e = 8$. The variable m is at the middle of the list, i.e., $m = (b+e) / 2$. Since $\text{list}[m] = \text{list}[4] = 41 < 44$, for the second pass b is redefined to be 4. Since $\text{list}[m] = \text{list}[6] = 47 > 44$, for the third pass e is redefined to be 6. Finally, $\text{list}[m] = \text{list}[5] = 44$, and the number 44 has been found at index 5.

| i | list[i] | pass through list | | | | b | m | e |
|-------|---------|-------------------|-------|-------|-------|-------|-------|-------|
| ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- |
| 0 | 13 | | | | 1 | 0 | 4 | 8 |
| 1 | 19 | | | | 2 | 4 | 6 | 8 |
| 2 | 27 | | | | 3 | 4 | 5 | 6 |
| 3 | 38 | | | | | | | |
| 4 | 41 | | | | | | | |
| 5 | 44 | | | | | | | |
| 6 | 47 | | | | | | | |
| 7 | 53 | | | | | | | |
| 8 | 57 | | | | | | | |

This is a “divide-and-conquer” algorithm, so that the number of passes is at most $\log_2 n$, where n is the length of the list. In other words, the time t required to search the list is $t \sim O(\log n)$.