7 October

Searching

- **binary search**: locate target value in sorted array by recursively eliminating half the array
  - expected runtime?
    - Notice: element at index N/2 can be found in 1 comparision
    - element at N/4, 3N/4 can be found in 2 comparisons
    - need x comparisons to find all of 2^(x-1) elements
    - if we have k operations to find N=2^(k-1) elements, k=O(log(N))
  - another way:
    - after k operations, we will have N/2^k elements remaining in array
    - task is solved when we have one element left: N/2^k <= 1
    - number of operations is proportional to log(N)
- built-in method to Java `Array` class, but array must be sorted first
  - returns index, or negative (index where the element *would* have been + 1)
- pseudocode of looping algorithm

```
public static int binarySearch(int[] a, int target) {
  int min = 0;
  int max = a.length - 1;

  while (min <= max) {
    int mid = (min + max) / 2;
    if (a[mid] < target) {
      min = mid + 1;
    } else if {a[mid] > target) {
      max = mid -1;
    } else {
      return mid;
    }
  }

  return -(min+1);
}
```

- psuedocode of recursive algorithm

```
public static int binarySearch(int[] a, int target) {
  return binarySearch(a, target, 0, a.length - 1);
}

private static int binarySearch(int[] a, int target, int min, int max) {
  if (min > max) {
    return -(min+1);
  } else {
    int mid = (min + max) / 2;
    if (a[mid] < target) { // too small, go right
      return binarySearch(a, target, mid+1, max);
    } else if (a[mid] > target) { // too large, go left
      return binarySearch(a, target, min, mid-1);
    } else {
      return mid;
    }
  }
```

```
}
```

Sorting

- fundamental in computer science / software engineering
- many solutions
- to sort, need comparison functions: $<$, $>$, `compareTo`

Example: Selection Sort

- look through list to find smallest value. Swap to index 0.
- look through to find second-smallest. swap to index 1.
- ...
- Repeat until all values are in proper places.

Runtime?

$$(n-1) + (n-2) + ... + 1 = \sum_{i=1}^{n-1} i = (n-1)\frac{(n-1)+1}{2} \tag{1}$$