Sorting and Searching Assignment

> In a word document briefly describe each sort/search method in your own words from what you heard. Include the advantages and disadvantages of each method. What is the code complexity in terms of N?

Selection Sort

Code complexity: $O(n^2)$

This algorithm divides the array into two parts: the sorted array and the unsorted array. The program looks for the smallest (or biggest) number in the unsorted array and swapping it with the leftmost element of the sorted array. The sorted array grows one element bigger, and it repeats.

Bubble Sort

Code complexity: $O(n^2)$

This algorithm goes through the array and swaps an element with its adjacent element if they are not sorted correctly. It does this over and over again to the array until it is completely sorted.

Binary Search

Code complexity: $O(\text{log}(n))$

This search algorithm divides the sorted array into half, into two sub-arrays. The target value is compared to the middle value of the entire array, and the half of the array which for certain does not have the value is eliminated. This process is repeated until the middle value equals the target value.

```
public class binarySearch {
      public static String[] selectionSort( String[] array )
  {

    // Find the string reference that should go in each cell of
    // the array, from cell 0 to the end
    for ( int j=0; j < array.length-1; j++ ) // for loop to iterate through the outer array
    {
      // Find min: the index of the string reference that should go into cell j.
      // search for the element that is the first in alphabetical order
      int min = j;
      for ( int k=j+1; k < array.length; k++ )
        if ( array[k].compareTo( array[min] ) < 0 ) min = k;

      // Swap the reference at j with the reference at min
      String temp = array[j];
      array[j] = array[min];
      array[min] = temp;

    }
  return array; // return the final array
  }
    public static int binarySearch(String[] arr, String element) {
        int l = 0, r = arr.length - 1;
        while (l ≤ r) {
            int m = l + (r - l) / 2;
```

```
            int res = element.compareTo(arr[m]);

            // Check if x is present at mid
            if (res == 0)
                return m;

            // If x greater, ignore left half
            if (res > 0)
                l = m + 1;

            // If x is smaller, ignore right half
            else
                r = m - 1;
        }

        return -1;
    }
    public static void main(String[] args) {
        String[] arr = new String[] {"ree", "bob", "mark", "svadrut"};
        String[] newArr = selectionSort(arr);
        System.out.println(binarySearch(newArr, "ono"));
    }
}
```