Exercise 1. Consider sorting an array `a[]` containing the following strings:

E A S Y Q U T I O N

a. (*Selection Sort*) What is `a[]` when `i = 3` and after the exchange on line 10?

```
1   public static void sort(Comparable[] a) {
2       int n = a.length;
3       for (int i = 0; i < n; i++) {
4           int min = i;
5           for (int j = i + 1; j < n; j++) {
6               if (less(a[j], a[min])) {
7                   min = j;
8               }
9           }
10          exchange(a, i, min);
11      }
12  }
```

b. (*Insertion Sort*) What is `a[]` when `i = 4` and the inner loop (lines 4 – 6) is complete?

```
1   public static void sort(Comparable[] a) {
2       int n = a.length;
3       for (int i = 1; i < n; i++) {
4           for (int j = i; j > 0 && less(a[j], a[j - 1]); j--) {
5               exchange(a, j, j - 1);
6           }
7       }
8   }
```

c. (*Shell Sort*) What is `a[]` after a 4-sort, ie, after the first iteration of the second while loop (lines 7 – 14)?

```
1   public static void sort(Comparable[] a) {
2       int n = a.length;
3       int k = 1;
4       while (k < n / 3) {
5           k = 3 * k + 1;
6       }
7       while (k >= 1) {
8           for (int i = k; i < n; i++) {
9               for (int j = i; j >= k && less(a[j], a[j - k]); j -= k) {
10                  exchange(a, j, j - k);
11              }
12          }
13          k /= 3;
14      }
15  }
```

Exercise 2. Consider sorting an array `a[]` containing the following strings:

E A S Y Q U T I O N

What is `a[]` when the function call `merge(a, aux, 0, 2, 4)` returns?

```
public static void sort(Comparable[] a) {
    Comparable[] aux = new Comparable[a.length];
    sort(a, aux, 0, a.length - 1);
}

private static void sort(Comparable[] a, Comparable[] aux, int lo, int hi) {
    if (hi <= lo) {
        return;
    }
    int mid = lo + (hi - lo) / 2;
    sort(a, aux, lo, mid);
    sort(a, aux, mid + 1, hi);
    merge(a, aux, lo, mid, hi);
}

private static void merge(Comparable[] a, Comparable[] aux, int lo, int mid, int hi) {
    for (int k = lo; k <= hi; k++) {
        aux[k] = a[k];
    }
    int i = lo, j = mid + 1;
    for (int k = lo; k <= hi; k++) {
        if (i > mid) {
            a[k] = aux[j++];
        } else if (j > hi) {
            a[k] = aux[i++];
        } else if (less(aux[j], aux[i])) {
            a[k] = aux[j++];
        } else {
            a[k] = aux[i++];
        }
    }
}
```

Exercise 3. Consider partitioning an array `a[]` containing the following keys, by calling the `partition()` function (shown below) from quick sort, as `partition(a, 0, a.length - 1)`:

P A R T I O N E X M L

```
private static int partition(Comparable[] a, int lo, int hi) {
    int i = lo;
    int j = hi + 1;
    Comparable v = a[lo];
    while (true) {
        while (less(a[++i], v)) {
            if (i == hi) {
                break;
            }
        }
        while (less(v, a[--j])) {
            if (j == lo) {
                break;
            }
        }
        if (i >= j) {
            break;
        }
        exchange(a, i, j);
    }
    exchange(a, lo, j);
    return j;
}
```

a. What is the value of the pivot element `v`?

b. What is the value returned by the function call, ie, what is the destination index of the pivot?

c. What is the state of the array after the call?

Exercise 4. Insert the following keys in that order into a max-heap:

E A S Y Q U T I O N

a. What is the state of the array `pq` representing the resulting tree?

b. What is the height of the tree (the root is at height zero)?

Exercise 5. Suppose that a letter in the input means *insert the letter* into an initially empty min-PQ and an asterisk ($*$) means *remove the minimum* from the priority queue. What is left in the priority queue after the following input is processed?

```
P R I O * R * * I * T * Y * * * Q U E * * * U E *
```

Exercise 6. Consider an array `a` of n (> 1) integers.

a. Provide an algorithm for computing the mode (value that occurs most frequently) of the array. For example, the mode of the array `a = {1, 3, 1, 1, 5, 5, 3}` is 1.

b. What is the running time $T(n)$ of your algorithm?

Solutions

Solution 1.

a. A E I N Q U T S O Y

b. A E Q S Y U T I O N

c. E A S I O N T Y Q U

Solution 2. A E Q S Y U T I O N

Solution 3.

a. P

b. 7

c. E A L M I O N P X T R

Solution 4.

a. - Y S U O Q E T A I N

b. 3

Solution 5. U

Solution 6.

a.

```
public static int mode(int[] a) {
    int mode = a[0];
    int freq = 0;
    int maxFreq = 1;
    Arrays.sort(a);
    for (int i = 1; i < a.length; i++) {
        if (a[i] == a[i - 1]) {
            freq += 1;
        } else {
            if (freq > maxFreq) {
                maxFreq = freq;
                mode = a[i];
            }
            freq = 0;
        }
    }
    return mode;
}
```

b. $T(n) = n \log n$