Algorithm HW#2

Divide-and-conquer

1 Binary Search

In this problem, you will implement the binary search algorithm that allows searching very efficiently (even huge) lists, provided that the list is sorted.

Sample 1.

Input:

```
1 5 8 12 13
8 1 23 1 11
```

Output:

```
2 0 -1 0 -1
```

2 Merge Sort

Sample 1.

Input:

```
38 27 43 3 9 82 10
```

Output:

```
3 9 10 27 38 43 82
```

3 Quick Sort

Sample 1.

Input:

```
9 -3 5 2 6 8 -6 1 3
```

Output:

```
-6 -3 1 2 5 6 8 9
```

4 Fast Exponentiation

1. Suppose that we need to *exactly* compute the value of $a^e \bmod n$ *for* some reasonably large *exponent e and n>1*.
2. The simplest and naïve algorithm performs $n - 1$ multiplications, by computing $a \times a \times \ldots \times a$.
3. Remark: we can do better by observing that
    a. If e is even: $e = e/2 + e/2$, then $ae \bmod n = (a^{e/2})^2 \bmod n$.
    b. If e is odd: $e = 1 + (e-1)/2 + (e-1)/2$, then ….
4. Find and implement an algorithm using the previous remark.
5. What is the worst time complexity of your algorithm?

5

Task. You are given a set of points on a line and a set of segments on a line. The goal is to compute, for each point, the number of segments that contain this point.

Input Format. The first line contains two non-negative integers s and p defining the number of segments and the number of points on a line, respectively.

Output Format. Non-negative integers is the number of segments.

Sample 1.

Input:

```
2 3
0 5
7 10
1 6 11
```

Output:

```
1 0 0
```

Here, we have two segments and three points. The first point lies only in the first segment while the remaining two points are outside of all the given segments.

Sample 2.

Input:

```
1 3
-10 10
-100 100 0
```

Output:

```
0 0 1
```

Sample 3.

Input:

```
3 2
0 5
-3 2
7 10
1 6
```

Output:

```
2 0
```

6 Majority Element

Task. The goal is to check whether an input sequence contains a majority element.

Output Format. Output 1 if the sequence contains an element that appears strictly more than n/2 times, and 0 otherwise.

Sample 1.

Input:

```
5
2 3 9 2 2
```

Output:

```
1
```

2 is the majority element.

Sample 2.

Input:

```
4
1 2 3 4
```

Output:

```
0
```

There is no majority element in this sequence.

Sample 3.

Input:

```
4
1 2 3 1
```

Output:

```
0
```

This sequence also does not have a majority element (note that the element 1 appears twice and hence is not a majority element).

Good Luck