Answer to the Question No. 2

Code:

```
import bisect


def longest_increasing_subsequence_optimized(nums):
    if not nums:
        return 0, []



    tails = []
    prev = [-1] * len(nums)
    indices = []

    for i, num in enumerate(nums):
        idx = bisect.bisect_left(tails, num)
        if idx == len(tails):
            tails.append(num)
            indices.append(i)
        else:
            tails[idx] = num
            indices[idx] = i



        if idx > 0:
            prev[i] = indices[idx - 1]


    max_length = len(tails)
    last_index = indices[-1]
    lis = []
    while last_index != -1:
        lis.append(nums[last_index])
        last_index = prev[last_index]
    lis.reverse()

    return max_length, lis

nums = [10, 9, 2, 5, 3, 7, 101, 18]
length, subsequence = longest_increasing_subsequence_optimized(nums)
print(f"Optimized Length of LIS: {length}")
print(f"Optimized LIS: {subsequence}")
```

Output:

```
Optimized Length of LIS: 4
Optimized LIS: [2, 3, 7, 18]
```