Scott Fenton

CS310: Homework 4

3/1/17

1. Josephus figured out where to stand in the vicious circle and survived. He also saved his friend, the penultimate winner.

1. For some small values of n, write down the position of the penultimate winner, I(n).You may extend the following table as much as you like.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| I(n) | 1 | 3 | 5 | 1 | 3 | 5 | 7 | 9 | 11 | 1 | 3 | 5 | 7 | 9 | 11 | 13 |

1. The reccurence relation is J(2m + 1)

2. Give an O(n log k)- time algorithm that merges k sorted lists with a total of n elements into one sorted list.

We solve this by using a min-heap to merge k sorted lists. We pick the smallest element in each list, and insert them into a minimum heap. When we insert it into the min-heap we keep track of what index of the list it came from. Then we deleteMin from the heap and insert that into the new array merges all three lists. Each time we delete a value from the min-heap, we insert an element from the corresponding list that the element came from. In this algorithm it takes O(k) to build the heap for every element in the k lists. It takes O(logk) to deleteMin and O(logk) to insert the next element from the corresponding array. The total time is O(k + nlogk) which equals O(n log k).

3. There are many duplications in the input sequence of n numbers, such that only O(log n) numbers are distinct. Give an O(n log log n) worst-case time algorithm to sort such a sequence.

By maintaining a balanced binary tree of all the distinct elements that have occurred. We use the binary tree to calculate the number of occurrence of each number. Since there are only log N distinct numbers, all of the operations are performed in log(log n) time, and to sort the sequence we only have to traverse the tree and print each integer the required number of times in pre-order traversal. O(n log log n).

4. Design an O(n log n) worst-case time algorithm that counts the number of inversions in an array of n numbers. Hint: Modify mergesort.

We can design this algorithm to determine the number of inversions in an array of n numbers by modifying mergesort to return the number of inversions, instead of sorting the elements.

5. The k-th quantiles of a set of n numbers are the k−1 numbers that divide the sorted set into k equal-sized sets (to within 1). For example, if the set is {1, 2, 3, . . . , 99}, the 10-th quantiles are 10, 20, . . . , 90. Design an O(n log k) time algorithm to find the k-th quantiles of a set. Hint: Use median of medians.

Quantiles(A, k, z)

1. If (k == 1) return