Analysis of algorithms

In all cases explain clearly and as succinctly as possible.

Problem 1

Answer: n T (n) = 2T (n) + log n 2 2 n = 4T (n) + logn n + log n 4 2 2 2 n = 4T (n) + log nn−1 + log n 4 2 2 = ... ∑log2 n 1 = nT (1) + n i= 1 i ∑n Since i= 1 1 → ln n, T (n) ∈ Θ(n log log n) i

Problem 2

Answer: The general idea is to use the technique similar to quicksort, by doing partition on both lids and cups.

First, we pick a cup randomly and use it to partition the lids into two subsets: those lids smaller than the size of that cup, and those larger than the size of the cup. We can also find the corresponding lid for that chosen cup. Second, we use that lid to partition the cups and divide them into two sets. We keep on repeating this procedure on each subset of cups/lids until all the cups/lids are paired. The overall time complexity is O(n log n) (Worst case: O(n2)).

Problem 3

Answer: In this problem, we are more interested in finding the median instead of the minimum/maximum element. The ⌊ n ⌋th element in a min/max heap is not the median. 2 In this case, we should develop a new type of heap to adapt to this problem. Problem 3 2 The solution is to use two heaps: a min-heap and a max heap. Suppose the total number of elements is n, we set the restriction that the max heap should contain ⌊ n ⌋ 2 elements. Correspondingly, the min-heap contains n - ⌊ n ⌋ elements. 2 When we insert an element, we always insert it into the max heap.

If the number of elements in the max heap exceeds ⌊ n ⌋, we remove the maximum element in the 2 max heap (the root) and insert it into the minimum heap. During this procedure, we need to do heapify to maintain the heap structure for both heaps. Under this setting, it is easy to see that all the elements in the max heap are less than those in the min-heap, and the two elements at the root of both heaps represent the ⌊ n ⌋th 2 and (⌊ n ⌋ + 1)th element.

2 Suppose the median is de? ned to be the ⌊ n ⌋th element overall n elements.

When 2 we delete the median, we just delete the root of the max heap, and the following two cases might occur: (1) If the max heap contains ⌊ n ⌋ 1 - elements, then we do delete-max to the max 2 heaps. (2) If the max heap contains ⌊ n - 1 ⌋ - 1 elements, we take out the root of the min 2 heap and set it to be the root in the max heap (because it is larger than all the elements in the max heap), then we do delete-min to the min-heap. It is straightforward to see that the time complexity for both insert and delete-median is O(log n).