# Comp 160: Algorithms, **Recitation 2**

1. You are given 2 sorted arrays $A$ and $B$ of size $n$ each. We want to design an algorithm that can find the median of the array obtained after merging the above 2 arrays (i.e. array of length $2n$). You can assume that all numbers are distinct.

   (a) Can you give a trivial solution? How fast does it run?

   (b) Now we will look for a faster algorithm. Let's first answer a simpler question: what is the solution when $n = 1$?

   (c) Say that $A[n/2] < B[n/2]$. What can you say about the rank of A[n/2] and B[n/2] in the merged array? How big and small can they be? Justify your answer with 1-2 sentences.

   (d) Describe the algorithm that you would use for solving the original question. You may assume that $n$ is a power of two. **Hint**: use the answer to the previous questions to design an algorithm

   (e) Briefly justify correctness of your algorithm (1-2 sentences).

   (f) Find a recurrence to denote the runtime of your algorithm, justify it with a couple of sentences, and solve it (using whatever method you like)

   **Note**: if this were an exam and you could not solve part c) just describe any recursive algorithm and analyze the runtime. You can get full credit for this part regardless of your previous answers.

   **Fun fact**: this question has been *borrowed* from Leetcode (`https://leetcode.com/problems/median-of-two-sorted-arrays/`). Their question just asks for a fast algorithm for computing the median (without giving intermediate questions). My guess is that interviewers expect candidates to do $\Theta(n)$ or $\Theta(n \log n)$ algorithms fast, and then with guidance questions end up deducing the $\Theta(\log n)$ algorithm. If this comes up in an interview, go straight to the $\Theta(\log n)$ solution to surely impress everyone!

2. Illustrate the operation of Radix Sort on the following list of words. What are the values of $d$ and $l$ in describing the complexity of this sort procedure?

   `COW, DOG, SEA, RUG, ROW, MOB, BOX`

3. Varying the Radix of RadixSort

   (a) (Warmup to the Warmup) Represent 1000 in hexadecimal, working it out by hand. Check your answer.

   (b) (Warmup) Suppose you are sorting $n$ numbers in the range $1 \to 10^{10}$ using Radix Sort. For your radix you have a choice of binary ($d = 2$), decimal ($d = 10$) or hexadecimal ($d = 16$). In each case give the maximum possible length $\ell$ of your numbers.

   (c) Now suppose you are sorting $n$ numbers in the range $1 \to n^5$. For your radix you have a choice of binary ($d = 2$), decimal ($d = 10$) or hexadecimal ($d = 16$). In each case give the maximum possible length $\ell$ of your numbers, and then the (asymptotic) runtime of radix sort.

4. In lecture we explained how to make CountingSort stable. Now, it is your time to make ANY comparison based algorithm (InsertionSort, MergeSort, HeapSort, and so on) stable. How can you do this with a single modification that could apply to any comparison-based algorithm?

   **Hint**: modify the $<$ and $=$ operators. If you are comparing two numbers that are equal, what do you want to make sure?

---

**Additional practice questions:**

5. In class, we learned how to compute the median of $n$ elements (or more generally the $i^{\text{th}}$ element) in $\Theta(n)$ time using groups of 5. In homework we explore groups of $k \geq 6$.

   (a) (Warmup/Review) What is the exact recurrence we would get if we make groups of 7 instead (assuming that $n$ is divisible by 7)?

   (b) Can we still achieve linear time if we use smaller groups? Say groups of 3? Write the recurrence and analyze the runtime. You can assume that $n$ is always divisible by 3.

   (c) (Hard! Extra challenge.) What about $\sqrt{n}$ groups of $\sqrt{n}$?

6. Explain to your partner why does CountingSort work. What Lemma would you state to show correctness? How would you prove it?

   **Note**: we are interested in correctness (ignore runtime for this question). Also, you do not need to give the full proof. Just say what technique you would use to prove it.

7. Heap structure review. You are given an array $A$ that contains the following elements in this order: 45, 2, 18, 7, 23, 47, 3, 15, 32.

   (a) Build a min-heap using the bottom-up (reverse) method presented in class, showing your work. How many comparisons do you make?

   (b) Build a min-heap using the top-down (forward) method presented in class, showing your work. How many comparisons do you make?

   (c) Are the two heaps that you get identical? Is this a coincidence of this particular instance? Or would the same happen for other instances? Explain in your own words whether the two methods will always produce identical heaps or not.

8. Let's review heaps. We have a heap and after several insertions the underlying array contains the following values: $[70, 30, 50, 10, 25]$

   (a) Is it a max or a min heap? Why?

   (b) Note that the array shows final situation of the heap, but we not know in which order the elements could have been inserted. Let's try to argue which number was inserted last. Could it be that 70 was inserted last? why?

   If the answer is yes you can simply give an order of insertion. If the answer is no, you could either ($i$) try all possibilities or ($ii$) be smart and try to argue why something is not possible.

   **Hint**: we love answers that go for ($ii$). Remember what are the invariants of a heap and see if you can maintain them all

(c) Using a similar argument, answer the same question for each of the numbers (that is, could 30 have been inserted last? what about 50? 10? 25?).

9. Go to Leetcode and answer questions:

   - Medium difficulty: (1093) https://leetcode.com/problems/statistics-from-a-large-sample/
   - Hard difficulty: (480): https://leetcode.com/problems/sliding-window-median/
   - Hard difficulty: (295): https://leetcode.com/problems/find-median-from-data-stream/

   It's OK to give the code but also try to solve them with lemmas and proofs the way were are focusing on in Comp 160.