10. tutorial

1. Sum. Say we have a set of natural numbers and a number x. We want to find out as quickly as possible whether our set contains a pair of elements which sum up to x.
What if I had a fixed x, but wanted my set to be dynamic, that is, I can `INSERT` and `DELETE` elements, and I want to be able to quickly check whether there is or isn't a pair of elements summing up to x. In what time is this possible?

2. Collision. You were given a hash function $h : [U] \to [m]$. Unless you know anything else about the function, how many function evaluations do you need to find a set of k elements of $[U]$ which all collide, that is, $\{a_1, \ldots, a_k\} \subseteq [U]$ and $h(a_1) = \cdots = h(a_k)$?

---

3. Mergesort with more parts? Describe a sorting algorithm which decomposes the input into more than 2 parts, and then sorts these recursively. Can it be faster than Mergesort?

4. Transitive closure. A *transitive closure* of a directed graph with vertices $\{1, \ldots, n\}$ is a 0/1 matrix T of shape $n \times n$, where $T_{uv} = 1$ if and only if the graph contains a path from u to v. Show that if we can multiply $n \times n$ matrices in time $\mathcal{O}(n^\omega)$, then we can compute the transitive closure in time $\mathcal{O}(n^\omega \log n)$. Even if you figure out this task without explicitly thinking "divide & conquer", try to recognize this pattern in it.
(ω is commonplace notation for the best possible exponent of matrix multiplication.)

5. Change of base. Say we have an n-digit number in base z and we want to change it to a different base z' (consider both z, z' a constant). Show how to do it in a Divide & Conquer way in time $\mathcal{O}(\max(M(n), n \log n))$, where $M(n)$ is the time needed to multiply n-digit numbers in the new base z'.´

6. Space complexity of Karatsuba. Prove that Karatsuba's multiplication algorithm works with linear space complexity (it requires space $\mathcal{O}(n)$). Recall that space complexity can be analyzed by consider the (worst-case of) root-leaf paths in the recursion tree – at any point in the run of the algorithm, only data related to some such path is in memory.

7. Recurrence 1. Solve the recurrence $T(n) = 2T(n/2) + \Theta(n \log n)$, $T(1) = 1$.

8. Recurrence 2. Solve the recurrence $T(n) = n^{1/2} \cdot T(n^{1/2}) + \Theta(n)$, $T(1) = 1$.

9. Twisted cable. We have a long cable with n wires on both ends. Each wire on the left end is connected to exactly one wire on the right end, and we want to find out which one. We can use the following operations to do that: (1) connect electricity to a given wire on the left end, (2) disconnect electricity from a given wire on the left end, and (3) measure electricity on a given wire on the right end. Design an algorithm which finds out which wire is connected to which, minimizing the number of these operations.

10. Twisted cable – nonadaptive version. Can you slightly change the algorithm above to work *non-adaptively* – that is, it's next step doesn't depend on it's previous step, i.e., it gathers some data in a way which doesn't depend on the input, and then reports its answer?