CS218 Design and analysis of algorithms **Jan-Apr 2025**

Exercise sheet 1

Lecture 1-3 (Jan 6, 7, 9) Binary search and variants

1. Consider a building with infinitely many floors. You need to find the highest floor h from which an egg can be dropped without breaking. Can you do it with $O(\log h)$ egg droppings?

2. Given an array with n positive integers $[a_1, a_2, \ldots, a_n]$ and a target value S, find the minimum length subarray whose sum is at least S. Can you do it in $O(n \log n)$ time?

   A subarray means a contiguous subset. That is for some $i \leq j$, $[a_i, a_{i+1}, \ldots, a_{j-1}, a_j]$.

3. Given two sorted integer arrays (with all distinct numbers in them) of size n, we want to find the median of the union of the two arrays. Can you find it by accessing only $O(\log n)$ entries in the two arrays.

4. Given an array of n integers and an integer S, find a pair of integers in the array whose sum is S. Can you do it in $O(n \log n)$ time?

5. Let $f : \mathbb{R} \to \mathbb{R}$ be a convex function that is promised to have a minimizing point. The function $f(x)$ and its derivative $f'(x)$ are not given explicitly, but via oracle access. That is, you can give any point $x \in \mathbb{R}$ and the oracle will give the values of $f(x)$ and $f'(x)$. How many queries do you need you find the point minimizing $f(x)$?

6. Given an integer a, check if it is of the form b^k for some (unknown) integers b and $k > 1$. Can you do this in time $O(\log^c a)$ for some constant c?

7. You are given a list of landholdings, say $a_1, a_2, \ldots, a_n$ (in hectares, can be zero). We want to give land to everyone who has less than f hectares and bring them up to f hectares. For this, we need to take away land from everyone who has more than c hectares, bring them down to c hectares, and redistribute the obtained land. (1) What is the highest value of f that can be feasible? (2) For a chosen value of f, find the right value of c?

8. You are standing on the number line at $x = 0$. There is a hidden treasure at $x = N$ for some unknown integer N (it could be positive or negative). You have a detector which will beep if you pass through the location of the treasure. What should be your strategy to minimize the total distance traveled and find the treasure? Can you ensure that the total distance travelled is $O(N)$.

   One strategy is to just keep traveling in the positive direction. If the treasure is on the positive side, then you can find it with distance travelled N. But, if it is on negative side, you will never find it. So this strategy does not work.

9. Prove that $\log(n!) \geq (n/2)\log(n/2) = (1/2)n \log n - n/2$.

10. Prove that $\log(n!) = \sum_{i=1}^{n} \log i \geq \int_1^n \log x \, dx$. Solve the integral to get a lower bound.

11. True or false?

    - $2n + 3$ is $O(n^2)$.
    - $\sum_{i=1}^{n} i^2$ is $O(n^2)$.
    - $\sum_{i=1}^{n} 1/i$ is $O(\log n)$.
    - n^n is $O(2^n)$.
    - 2^{3n} is $O(2^n)$.

- $(n+1)^3$ is $O(n^3)$.
- $(n+\sqrt{n})^2$ is $O(n^2)$.
- $\log(n^3)$ is $O(\log n)$.

Below exercises are just for interest, not in the syllabus.

12. We have seen the binary search method for computing square root of a number. For each query, we compute the square of current value. Is there a faster implementation like the division algorithm, where we don't need to compute the square in each iteration from scratch?

13. Consider two algorithms for finding square root of an integer, Babylonian method (Newton-Raphson method, check wiki) and Binary search. Which one do you think is faster?

14. Compare two algorithms to compute a root of a polynomial. Binary search and Newton-Raphson.