Tutorial 1, Design and Analysis of Algorithms, 202 3

1. For a real number n the function $\log^*(n)$ is defined as follows: $\log^*(n)$ is the smallest natural number i so that after applying the logarithmic function (base 2) i times on n we get a number less than or equal to 1. For example, $\log^*(2^2)$ is 2 because $\log(\log(2^2)) = 1 \leq 1$. $\log^*(2^{2^2})$ is 3 because $\log\log(\log(2^{2^2})) = 1 \leq 1$.
   Either prove or disprove:

   (a) $\log(\log^*(n)) = O(\log^*(\log(n)))$.

   (b) $\log^*(\log(n)) = O(\log(\log^*(n)))$.

2. Take the following list of functions and arrange them in ascending order of growth rate (with proof). That is, if the function $g(n)$ immediately follows function $f(n)$ in your list, then it should be the case that $f(n)$ is $O(g(n))$.

   (a) $g_1(n) = 2^{\sqrt{\log n}}$.

   (b) $g_2(n) = 2^n$.

   (c) $g_3(n) = n^{\frac{4}{3}}$.

   (d) $g_4(n) = n(\log n)^3$.

   (e) $g_5(n) = n^{\log n}$.

   (f) $g_6(n) = 2^{2^n}$.

   (g) $g_7(n) = 2^{n^2}$.

3. Assume that you have functions f and g such that $f(n)$ is $O(g(n))$. For each of the following statements, decide whether you think it is true or false and give a proof or counterexample:

   (a) $\log_2 f(n)$ is $O(\log_2 g(n))$.

   (b) $2^{f(n)}$ is $O(2^{g(n)})$.

   (c) $f(n)^2$ is $O(g(n)^2)$.

4. Prove that
$$\Theta(n-1) + \Theta(n) = \Theta(n).$$
   Does it follow that
$$\Theta(n) = \Theta(n) - \Theta(n-1)?$$
   Justify your answer.

5. Use mathematical induction to show that when $n \geq 2$ is an exact power of 2, the solution of the recurrence is
$$T(n) = 2T(n/2) + n, n > 2$$
$$T(n) = 2, n = 2$$
   is $T(n) = n \log n$

6. Give asymptotic upper and lower bounds for $T(n)$ for $T(n) = \sqrt{n}T(\sqrt{n}) + n$.

7. **Prove:** Every comparison-based algorithm for finding both the minimum and maximum of n elements requires at least (3n/2)-2 comparisons.

8. **Prove:** Every comparison-based algorithm for finding both the maximum and the second maximum of n elements requires at least $n + \log n - 2$ comparisons.

9. In an infinite array, the first n cells contain integers in sorted order, and the rest of the cells are filled with ∞. Present an algorithm that takes x as input and finds the position of x in the array in $O(\log n)$ time. You are not given the value of n.

10. Device a ternary search algorithm that first tests the element at position $\frac{n}{3}$ for equality with some value x, and then checks the element at $\frac{2n}{3}$ and either discovers x or reduces the set size to one-third the size of the original. Compare this with the binary search.

11. Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n points in $\mathbb{R}^2$, where $p_i = (x_i, y_i)$. Let s_{ij} be the slope of the line segment that joins p_i and p_j and $S = \{s_{ij} \mid 1 \leq i < j \leq n\}$. Basically, S contains all the slopes. Thus, $|S| = O(n^2)$. The objective is to compute the maximum slope in S in $O(n \log n)$ time, rather than the obvious $O(n^2)$ time. Devise an algorithm for the objective mentioned above. *(Hint: 1. Use geometrical property, 2. the desired time complexity is another clue.)*