Tutorial 1, Design and Analysis of Algorithms, 2021

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 logarithm function (base 2) i times on n we get a number less than or equal to 1. E.g. $\log^*(2^2)$ 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 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 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 $O(\log_2 g(n))$.

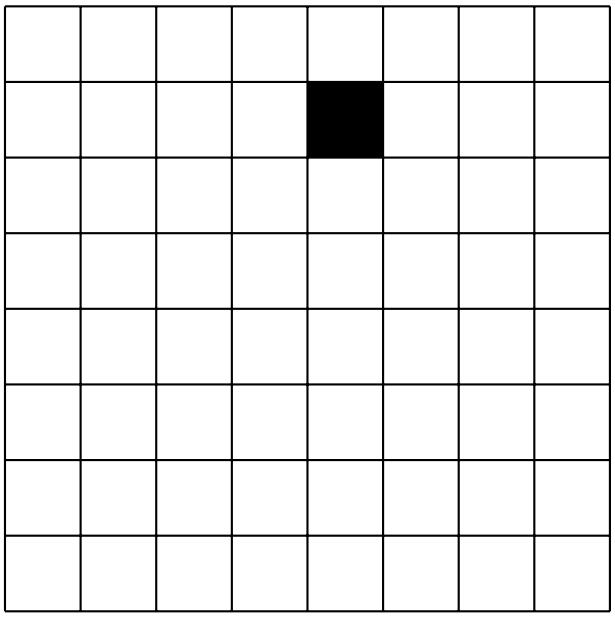
   (b) $2^{f(n)}$ is $O(2^{g(n)})$.

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

4. Solve the following recurrence relation (without using the Master Theorem):

$$T(n) = \begin{cases} 1, & \text{for } n \leq 4; \\ 2T(\sqrt{n}) + \frac{\log n}{\log\log n}, & \text{for } n > 4. \end{cases}$$

5. Tile the follwoing 8×8 defective chessboard using triominoes using the divide and conquer algorithm, and draw the divide and conquer graph for the solution.



6. 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.

7. Device a "binary" search algorithm that splits the set not into two sets of (almost) equal sizes but into two sets, one of which is twice the size of the other. How does this algorithm compare with binary search?

8. 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 binary search.

9. The sets A and B have n elements each given in the form of sorted arrays. Design $O(n)$ algorithms to compute $A \cup B$ and $A \cap B$.

10. Consider an n-node complete binary tree T, where $n = 2^d - 1$ for some d. Each node v of T is labeled with a real number x_v. You may assume that the real numbers labeling the nodes are all distinct. A node v of T is a *local minimum* if the label x_v is less than the label x_w for all nodes w that are joined to v by an edge. You are given such a complete binary tree T, but the labeling is only specified in the following *implicit* way: for each node v, you can determine the value x_v by probing the node v. Show how to find a local minimum of T using only $O(\log n)$ *probes* to the nodes of T. Give a proof of correctness of your algorithm and also prove its time complexity.