# Indian Institute of Technology, Madras
# Department of Applied Mechanics & Biomedical Engineering
# Computational Tools: Algorithms, Data Structures and Programs - ID6105

Assignment set - I

To be submitted by: Monday 25th August, 2025

## 1 Problem 1

In class we discussed a number of problems with varying time complexities. Revisit these problems and by modifying the codes that are already shared, find the space complexity of each of them (including all the input variables). Submit your response by writing down the final expression of the space complexity and also the associated plot from the numerical computations with the correct scaling exponent.

---

## 2 Problem 2

Prove each by applying directly the definitions of $O, \Omega$ and $\Theta$. It is not pertinent that you give the exact proofs, use arguments similar to the ones shown in the class,

(a) $53n$ is $\Theta(n)$.
(b) $n^2\sqrt{n}$ is not $\Omega(n^3)$
(c) $n^2$ is $\Omega(20n)$
(d) $2n^4 - 3n^2 + 32n\sqrt{n} - 5n + 60$ is $\Theta(n^4)$
(e) $100n^2$ is $O(n^3)$
(f) $2n^3$ is $O(2^n)$
(g) $10n^{10}$ is $O(2^n)$
(h) $n^{10}$ is $\Omega(1000n)$
(i) $n^3 + \log_2(n)^{10}$ is $O(n^3)$
(j) $n\sqrt{n}\log_2(n)$ is $O(n^2)$

---

## 3 Problem 3

In class we discussed the binary search algorithm which divided the problem into halfs and proceeded recursively, consider the follow-ups given below.

(a) Write down a python-code for ternary search where the initial search problem is divided

into three parts, with each sub problem of size $n/3$ and then solves it recursively.

(b) Generalise to a $k-ary$ search algorithm that divides the problem into $k$ parts. That is, it makes at most $k-1$ comparisons and works on problems of size $n/k$.

(c) For any fixed value of $k$, explain why the complexity is $O(\log n)$ time. Plot the results from the python code and show the scaling.

(d) Explain why the number of inequality tests made by the algorithm of Part (b) is at most $\lceil (k-1)\log_k(n) \rceil$ in the worst case.

---

# 4 Problem 4

A perfect integer square partition is a square where the interior area is partitioned into distinct square regions, in which the dimensions of all of these regions are integers. For example, as shown in the figure 1, a square of size 112 by 112 can be partitioned into distinct square regions where the smallest two have size 2 by 2 and 4 by 4 and the largest has size 50 by 50:

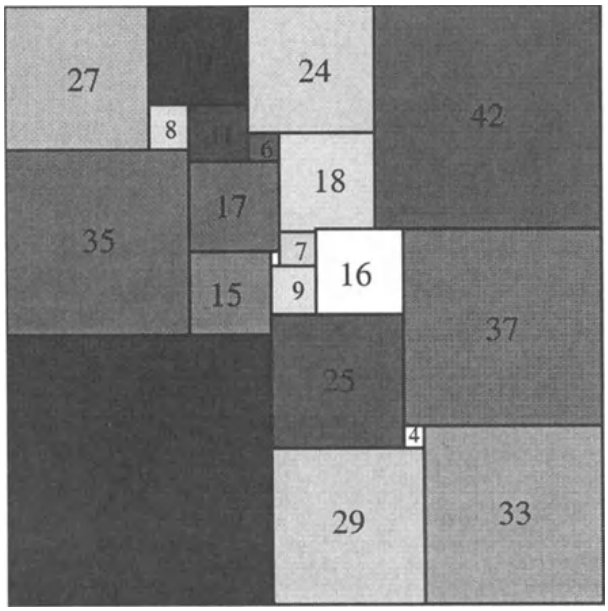


Figure 1: Partitioning of a square of size $112 \times 112$

Describe an algorithm and give python-code which, for an integer n, computes a perfect integer square partition of the n by n square or determines that none is possible. Analyze the time and space used by your algorithm, and plot the scaling with you python code.

---