# Divide & Conquer
# Problem Set 1 – CS 6515/4540 (Fall 2025)

Answers to problem set 1, question 3.

## 1 Median of Medians

1. The median of medians algorithm involves computing a good pivot and then partitioning our input array around that pivot to produce 2 sub-arrays containing numbers smaller and larger than the pivot.

   - The good pivot finding algorithm relies on dividing the input array into small finite-sized groups which can be sorted individually in O(1) time. Then we collect the middle points of all these arrays and recursively divide them again into small finite-sized groups and repeat the above algorithm.
   - Once we have a good pivot, then we can partition our input array around it. Our $k^{th}$ largest element in the worst case will always end up in the larger partitioned sub-array.

   Given an $M \geq 3$, we will divide the array (containing N elements) into $\frac{N}{M}$ groups, each group containing M elements. Considering M to be a small enough finite constant, we can sort each of the $\frac{N}{M}$ groups in $O(1)$ time. Recursively repeating this algorithm we will get a pivot element such that it's larger than at least $\lceil\frac{M}{2}\rceil * \frac{N}{2M}$ and also smaller than at least $\lceil\frac{M}{2}\rceil * \frac{N}{2M}$ elements in the input array.
   Therefore, in the worst case after identifying the pivot, we can remove (ignore) $\lceil\frac{M}{2}\rceil * \frac{N}{2M}$ elements. And then perform the whole algorithm for the remaining set of elements. This gives us the recurrence:

   $$\boxed{\text{T(n)} \leq T(n - \lceil\tfrac{M}{2}\rceil * \tfrac{n}{2M}) + T(\tfrac{n}{M}) + O(n)}$$

   Furthermore, given M is odd $\lceil\frac{M}{2}\rceil = \frac{M+1}{2}$, so for an upper bound, we get

   $$T(n) \leq T(n - \frac{M+1}{2} * \frac{n}{2M}) + T(\frac{n}{M}) + O(n)$$

   $$\boxed{\text{T(n)} \leq T(\tfrac{n(3M-1)}{4M}) + T(\tfrac{n}{M}) + O(n)}$$

   It's important to note that the above recurrence is $\Theta(n)$ for all $M \geq 5$, but for $M = 3$ it's a super-linear recurrence, it ends up being $O(nlogn)$ in worst case. This can be easily proved by induction. For $M = 3$, the recurrence for the worst-case running time of the algorithm is

   $$T(n) \;\leq\; T(\lceil\frac{5n}{6}\rceil) + T(\lceil\frac{n}{3}\rceil) + d * n,$$

   for some constant $d > 0$.

   We'll assume that there exists a constant $c > 0$ such that

   $$T(n) \leq cn \quad \forall n \leq n_0,$$

   and try to prove a contradiction for $T(n_0 + 1) \leq c(n_0 + 1)$.

   By the recurrence and the induction hypothesis,

   $$\begin{aligned} T(n_0+1) &\leq T(\lceil\frac{5(n_0+1)}{6}\rceil) + T(\lceil\frac{n_0+1}{3}\rceil) + d(n_0+1) \\ &\leq c * (\frac{5(n_0+1)}{6} + 1) \;+\; c * (\frac{n_0+1}{3} + 1) \;+\; d(n_0+1) \\ &\leq \frac{7}{6}c(n_0+1) + 2c + d(n_0+1). \end{aligned}$$

To conclude $T(n) \leq cn$, we would require

$$c(n_0 + 1) \;\geq\; \frac{7}{6}c(n_0 + 1) + d(n_0 + 1),$$

i.e.,

$$0 \;\geq\; \left(\frac{c}{6} + d\right)(n_0 + 1).$$

This leads to a contradiction, since $n > 0$, $d > 0$, and $c > 0$, the right-hand side is strictly positive. Hence, no finite $c$ can satisfy the induction, and $T(n)$ cannot be bounded by $O(n)$.
Hence, the algorithm with $M = 3$ does not run in linear time.

2. Given $M = 7$, the recurrence relation becomes:

$$T(n) = T(\lceil\frac{5n}{7}\rceil) + T(\lceil\frac{n}{7}\rceil) + d * n$$

I'll prove that $\boxed{\text{T(n)} = \Theta(n)}$ by induction.

- My induction hypothesis is: $T(k) \leq ck \ \forall c \geq 7d$.
- My base case is: $T(1) = 1 \leq c$ (we can choose such a c).
- My induction step is: I assume the hypothesis holds for all $k \leq n_0$ then I'll prove that the hypothesis will also hold for $k = n_0 + 1$.

$$T(n_0+1) = T(\lceil\frac{5n_0+5}{7}\rceil) + T(\lceil\frac{n_0+1}{7}\rceil) + d*(n_0+1)$$

$$T(n_0+1) \leq c*(\lceil\frac{5n_0+5}{7}\rceil) + c*(\lceil\frac{n_0+1}{7}\rceil) + d*(n_0+1)$$

$$T(n_0+1) \leq c*(\frac{6*(n_0+1)}{7} + 2) + d*(n_0+1)$$

We choose our $c \geq 7d$, this is arbitrarily easy to choose for large n. Therefore, we get

$$T(n_0+1) \leq c*(n_0+1)$$

Therefore, choosing $c \geq 7d$ and large enough to include the $2c$ constant, we have proved that $T(n) \leq c*n \ \forall n \geq n_0$. Combining this with our initial hypothesis we get that $T(n) = O(n)$.

Via a similar induction hypothesis we can also prove that $T(n) = \Omega(n)$. To do so,

- My induction hypothesis is: $T(k) \geq ck \ \forall c \leq 7d$
- My base case is: $T(1) = 1 \geq c$ (we can choose such a c).
- My induction step is: I assume the hypothesis holds for all $k \leq n_0$ then I'll prove that the hypothesis will also hold for $k = n_0 + 1$.

$$T(n_0+1) = T(\lceil\frac{5n_0+5}{7}\rceil) + T(\lceil\frac{n_0+1}{7}\rceil) + d*(n_0+1)$$

$$T(n_0+1) \geq c*(\lceil\frac{5n_0+5}{7}\rceil) + c*(\lceil\frac{n_0+1}{7}\rceil) + d*(n_0+1)$$

$$T(n_0+1) \geq c*(\frac{6*(n_0+1)}{7} + 2) + d*(n_0+1)$$

We choose our $c \leq 7d$, this is arbitrarily easy to choose for large n. Therefore, we get

$$T(n_0+1) \geq c*(n_0+1)$$

Therefore, choosing $c \leq 7d$ and large enough to include the $2c$ constant, we have proved that $T(n) \geq c*n \ \forall n \geq n_0$. Combining this with our initial hypothesis we get that $T(n) = \Omega(n)$.

Therefore, $T(n) = \Theta(n)$.

3. The asymptotic running time of the algorithm has not changed by choosing a larger M. $\boxed{\text{It is still } \Theta(n)}$. Meaningfully, I think we get no benefit from taking $M > 5$, that's because for larger M we end up taking more time for sorting the individual groups when trying to find a good pivot, and we haven't achieved any great benefit in the number of elements we're able to ignore after partitioning around the pivot. We do get some minor gain in the time taken to compute the pivot, 2nd term of the recursion, $T(\frac{N}{M})$. All to say, unless we do a more rigorous analysis it is difficult to analyze M=7 vs M=5 in absolute time.