School of Computing and Information Systems
COMP90038 Algorithms and Complexity Tutorial Week 7

1. Let T be defined recursively as follows:

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(n-1) + n/2 \quad n > 1 \end{aligned}$$

The division is exact division, so $T(n)$ is a rational, but not necessarily natural, number. For example, $T(3) = 7/2$. Use telescoping to find a closed form definition of T.

2. Use the Master Theorem to find the order of growth for the solutions to the following recurrences. In each case, assume $T(1) = 1$, and that the recurrence holds for all $n > 1$.

   (a) $T(n) = 4T(n/2) + n$

   (b) $T(n) = 4T(n/2) + n^2$

   (c) $T(n) = 4T(n/2) + n^3$

3. When analysing quicksort in the lecture, we noticed that an already sorted array is a worst-case input. Is that still true if we use median-of three pivot selection?

4. Let $A[0..n-1]$ be an array of n integers. A pair $(A[i], A[j])$ is an *inversion* if $i < j$ but $A[i] > A[j]$, that is, $A[i]$ and $A[j]$ are out of order. Design an efficient algorithm to count the number of inversions in A.

5. A *tromino* is an L-shaped tile made up of three 1×1 squares (green/hatched in the diagram below). You are given a $2^n \times 2^n$ chessboard with one missing square (red/grey in the diagram below). The task is to cover the remaining squares with trominos, without any overlap. Design a divide-and-conquer method for this. Express the cost of solving the problem as a recurrence relation and use the Master Theorem to find the order of growth of the cost.

   Hint: This is a nice example where it is useful to split the original problem into *four* instances to solve recursively.