Tutorial 5

Prof. Nick Harvey *University of British Columbia*

1. **(Finding Recurrence from Code)** Give a recurrence relation for the running time of the following algorithm as a function of $n = last - first + 1$. To simplify your answer, do not specify the floors and ceilings in your recurrence.

```
procedure TurtleSort(A, first, last)
    if last − first ≤ 2 then
        if last − first = 1 then
            if A[first] > A[first + 1] then
                swap(A[first], A[first + 1])
            end if
        end if
        if last − first = 2 then
            if A[last − 1] > A[last] then
                swap(A[last − 1], A[last])
            end if
            if A[first] > A[first + 1] then
                swap(A[first], A[first + 1])
            end if
        end if
    else
        q1 ← ⌊(3 · first + last)/4⌋
        q2 ← ⌊(first + last)/2⌋
        q3 ← ⌊(first + 3 · last)/4⌋
        TurtleSort(A, first, q2)
        TurtleSort(A, q1, q3)
        TurtleSort(A, q2, last)
        TurtleSort(A, q1, q3)
        TurtleSort(A, first, q2)
        TurtleSort(A, q1, q3)
    end if
end procedure
```

2. **(Exponential-Type Recurrence Relations)** Using the *guess-and-verify* method, prove that the function $T(n)$ defined by the recurrence relation

$$T(n) = \begin{cases} 2T(n-1) + 8T(n-2) + n & \text{if } n \geq 3 \\ 7 & \text{if } n = 2 \\ 2 & \text{if } n = 1 \end{cases}$$

is in $O(4^n)$.

3. **(Divide-and-conquer with several sizes of subproblems)** Use the Akra-Bazzi method to prove a Big-Θ bound for the function $T(n)$ defined by

$$T(n) \;=\; \begin{cases} 6T(\lfloor n/2 \rfloor) + 16T(\lfloor n/4 \rfloor) + 2n^3 & (n \geq 4) \\ 1 & (n \leq 3) \end{cases}.$$