PS2

June 18, 2025

Problem 2-1

(a)

The height of the tree associted with LoD(n) is n. Since in each call n gets reduced by one until the recrsion terminates at $n = 0$.

(b)

The first call of Snowflake(n) has 3 calles of Snowflake-edge(n), which then has 4 calls of Snowflake-edge($n-1$). If the (as defined in the question) Snowflake is considered the 0 level of the recursion tree, we have $3 \cdot 4^i$ nodes in a recursion tree at level $0 \leq i < n$.

(c)

Each node of the recursion tree draws a new triangle so the asymptotic rendering time (triangle count) for a node in the recursion tree at level i, for $0 \leq i < n$ is $O(1)$.

(d)

Each level of the recursion tree has $3 \cdot 4^i$ nodes and each node draws a triangle therefore the asymptotic rendering time (triangle count) at each level i of the recursion tree $0 \leq i < n$ is $O\left(4^i\right)$.

(e)

The total asymptotic cost for the CPU, when rendering a snowflake with LoD n using 3D hardware-accelerated rendering is

$$T(n) = 3\sum_{i=0}^{n-1} 4^i = 3\frac{4^n - 1}{4 - 1} = 4^n - 1 = O\left(4^n\right)$$

(f)

The recursion tree is the same so the height of the recursion tree for rendering a snowflake of LoD n using 2D hardware-accelerated rendering

Problem 2-2

(a)

The methods which have the larges total time are in decreasing order "_find_min", "__lt__" which a run time of $\approx 66,53$, "step" has a run time of ≈ 0.2 and the rest < 0.1

(b)

The find_min method is called 259964 times.

(c)

The class containing the "_find_min" is PriorityQueue. The priority queue is implemented with a list and the minimum is found just by going through the non sorted list and finding the smallest element. This scales as $O(n)$.

(d)

If the data structure was mainted the index of the minimum key is just 0. But the pop method will change to $O(\log n)$ since the heap invariant property should be maintained.

(e)

The solution appears in as an imported class PriorityQueue in priority_queue.py. The code contains an implementation of the priority queue ADT employ a min-heap.