

Q1: Design an $O(\log n)$ -time alg that takes $a \in \mathbb{R}$ and $n \in \mathbb{N}^+$ as input and finds a^n . Assume addition, subtraction, division, multiplication, and rounding to the nearest integer are $O(1)$ operations. You are not allowed to use $\log$ or $\exp$.

Answer 1: Let $M = 2^m$ be the smallest power of 2 greater than n . (so $m = \lceil \log_2 n \rceil$)

Repeatedly square a to compute $a, a^2, a^4, a^8, a^{16}, \dots, a^M$, which takes $O(\log n)$ multiplications. Say

Write out the binary representation of $n = \sum_{i=0}^m b_i 2^i$, ($b_m = 0$, since $2^m = M > n$)

In other words, $b_i = 1$ if the i th bit is 1.

Then $a^n = \prod_{i=0}^m b_i a^{2^i}$, and we need only $O(m) = O(\log n)$ operations.

Answer 2: We can do the above answer in $O(1)$ space as well:

Set temp var $b = a$ and a cumulative variable $o = 1$.

Repeat until $n = 0$

- if n is odd, multiply o by b
- Update b to b^2
- Update n to $\lfloor \frac{n}{2} \rfloor$

Output o

Answer 3: Let $f(a, n) := a^n$. The base case is $f(a, 0) = 1$.

We use recurrence on the 2nd arg.

$$f(a, n) = \begin{cases} f(a, n/2)^2 & \text{if } n \text{ is even} \\ f(a, \frac{n-1}{2})^2 \cdot a & \text{if } n \text{ is odd} \end{cases}$$

Each step we perform $O(1)$ ops & reduce the size (n) to half.

So runtime is $O(\log n)$

Wrong answer: $f(a, n) = f(a, \lfloor n/2 \rfloor) \cdot f(a, \lceil n/2 \rceil)$ because runtime could be $O(n)$.
(since each recursion does two different recursive steps)

Wrong answer: $T(n) = T(n/2) + T(n/2)$ because runtime could be $O(n)$.
(since each recursion does two different recursive steps rather than reusing a single one)

Q2: Draw a 2D skyline given a list of building x-coords & their heights
 $(l_1, h_1, r_1), (l_2, h_2, r_2), \dots, (l_n, h_n, r_n)$
Sorted in increasing order of left coordinate.

Design an $O(n \log n)$ algorithm to produce the skyline line to
draw in format $x_1, y_1, x_2, y_2, x_3, y_3, \dots$

Answer 1: Divide & Conquer.

Base case is single building (l_1, h_1, r_1) , which is trivial setting $x_1 = l_1, y_1 = h_1, x_2 = r_1$.

To draw skyline of n buildings, divide buildings into two halves (arbitrarily)
and recursively call the skyline-drawing alg on the halves.

To merge two overlapping skylines A & B , keep track of current heights $h_A = h_B = h_Z = 0$.
Consider the skyline in (x, y) pairs, except for the last x-coordinate, which is $(x, 0)$.

Repeat until both lists empty:

Look at both lists A & B and choose the pair with smallest x-coordinate (x, y) .

If we chose from list A , set $h_A = y$.

If we chose from list B , set $h_B = y$.

Set $h_Z = \max(h_A, h_B)$

Append (x, h_Z) to Output list.

Walk through output list removing pairs with duplicate y-coordinate,
since those are extraneous.

This merge operation takes $O(n)$ time.

Thus, our recursion is $T(n) = 2T(n/2) + O(n)$

$\Rightarrow T(n) = O(n \log n)$.

The trick is realizing that we always keep the taller height.

Answer 2: We sweep from left to right, maintaining a max-heap.

In the heap, every element is a building and the building with
largest height is at the top.

In our left-right sweep:

- Every time we reach a building, insert into heap.

- 1.1. - reach the right-most pt of current top, delete the

- Every time we reach a building, insert into heap.
- When we reach the right-most pt of current top, delete the top from the heap repeatedly until we get to a building whose right-most point is to the right of current point.
- If the height of the building at the top changes, output current x-coordinate + the new height.

We insert/delete each building once, so runtime of each insert/delete is $O(\log n)$, so overall runtime is $O(n \log n)$.

Note: the building at top of heap is always valid, and all currently valid buildings are in heap (though some expired buildings might be there, but it doesn't matter). Thus, the heights are correct, and the x-coordinates are correct because height only changes at building boundaries.

(This problem is on LeetCode, ID 218).