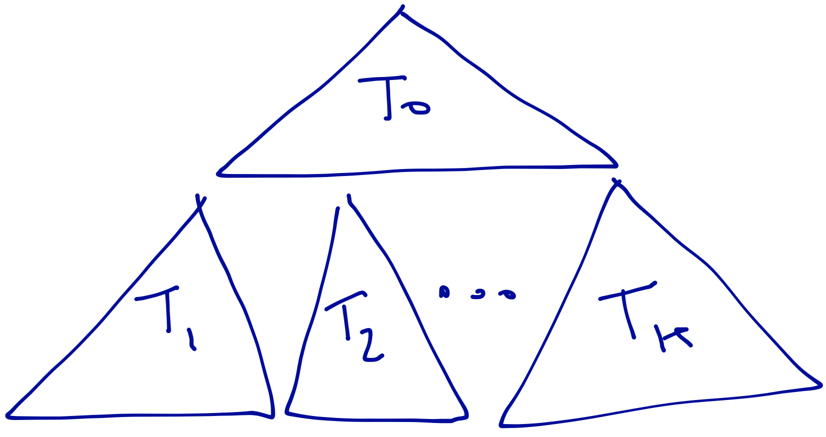
Analysis of the vEB layout

- Given a complete binary (search) tree of height $h > 1$, let h' be the largest power of 2 strictly less than h
- The lowest h' levels form the bottom trees $T_1, T_2, \ldots, T_k$
- The highest $h-h'$ levels form the top tree T_0



- When h is a power of 2, both $h' = h - h' = \frac{h}{2}$
  Also, letting $N-1$ ($N = 2^h$) be the number of nodes, we have that $|T_0| = |T_1| = \cdots = |T_k| = 2^{h/2} - 1 = \sqrt{N} - 1$ and $k = 2^{h/2} = \sqrt{N}$

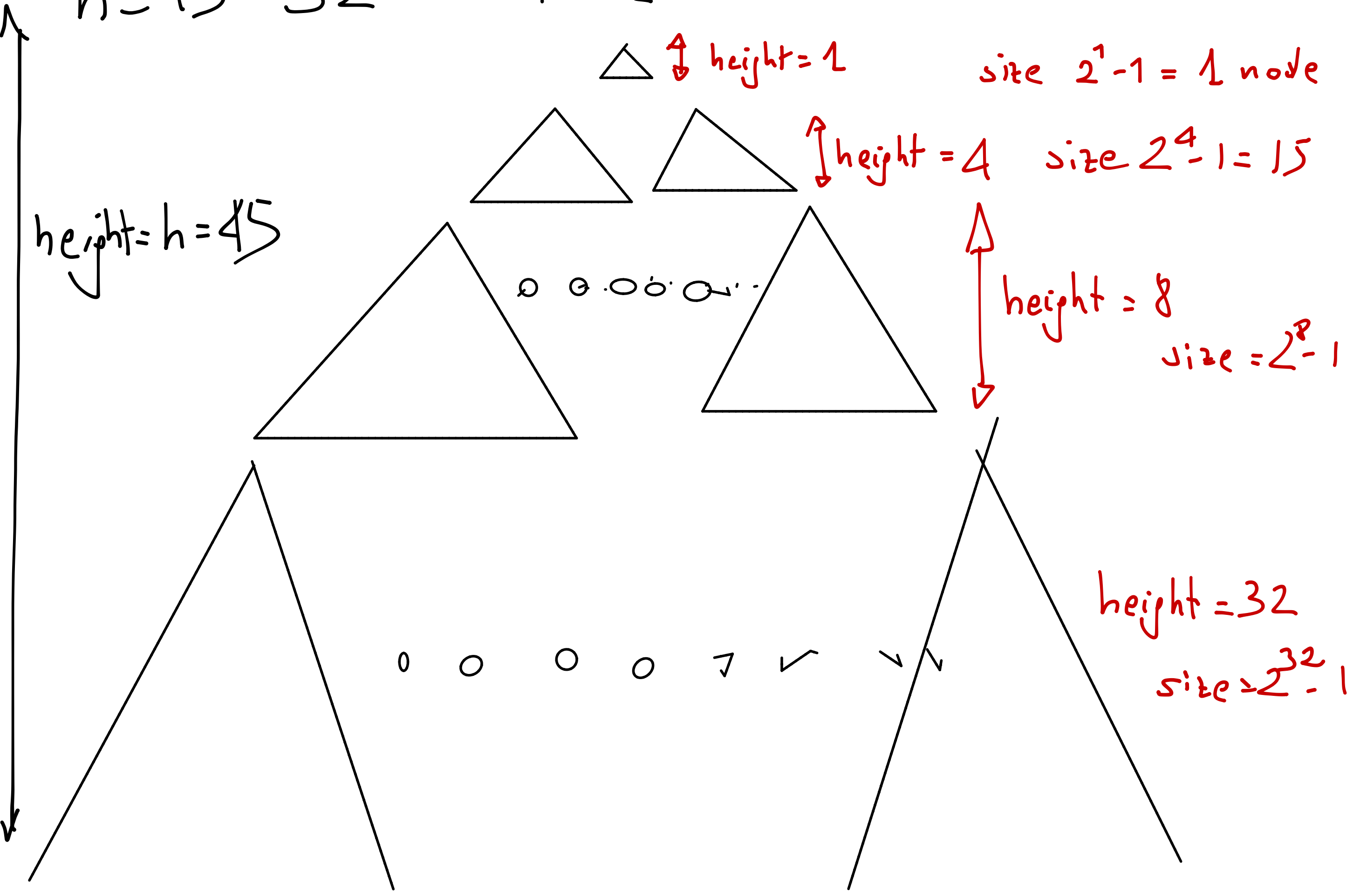
- When h is not a power of 2, it is $h' \geq \frac{h}{2}$ to be a power of 2

▷ this is called hyperceiling decomposition in the vEB layout

▷ Note that a simple way to look at this decomposition is that of iteratively finding the bottom trees of the largest power of 2, say h', and subtract h' from h, and repeat.

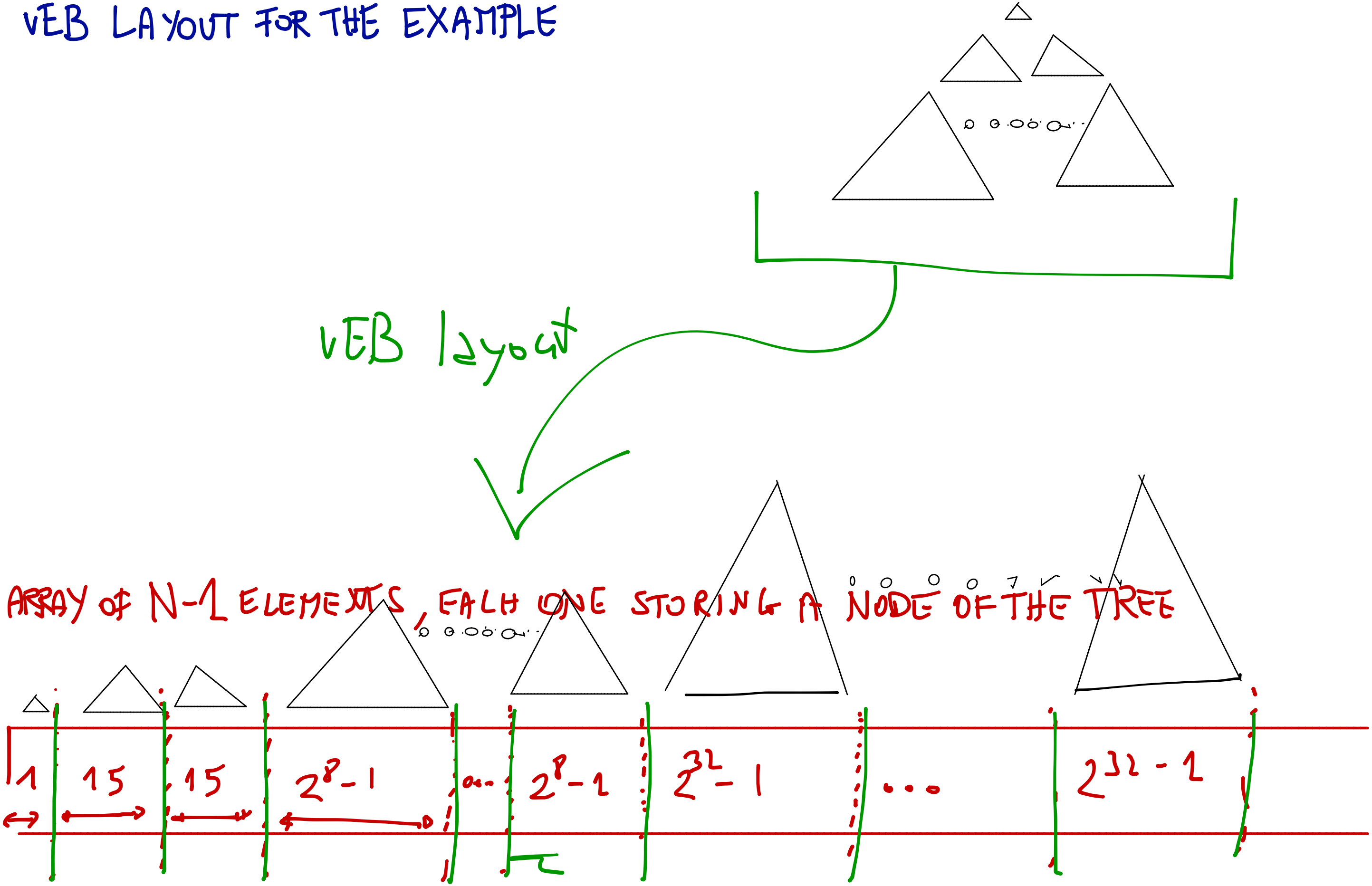
Example ⟶

$h = 45 = 32 + 8 + 4 + 1$



height = h = 45
height = 1
size $2^1 - 1 = 1$ node
height = 4
size $2^4 - 1 = 15$
height = 8
size = $2^8 - 1$
height = 32
size = $2^{32} - 1$

VEB LAYOUT FOR THE EXAMPLE



• SUBPROBLEM: how to layout each subtree, now of height a power of 2 (see previous slide).

EASY: split always in half the height

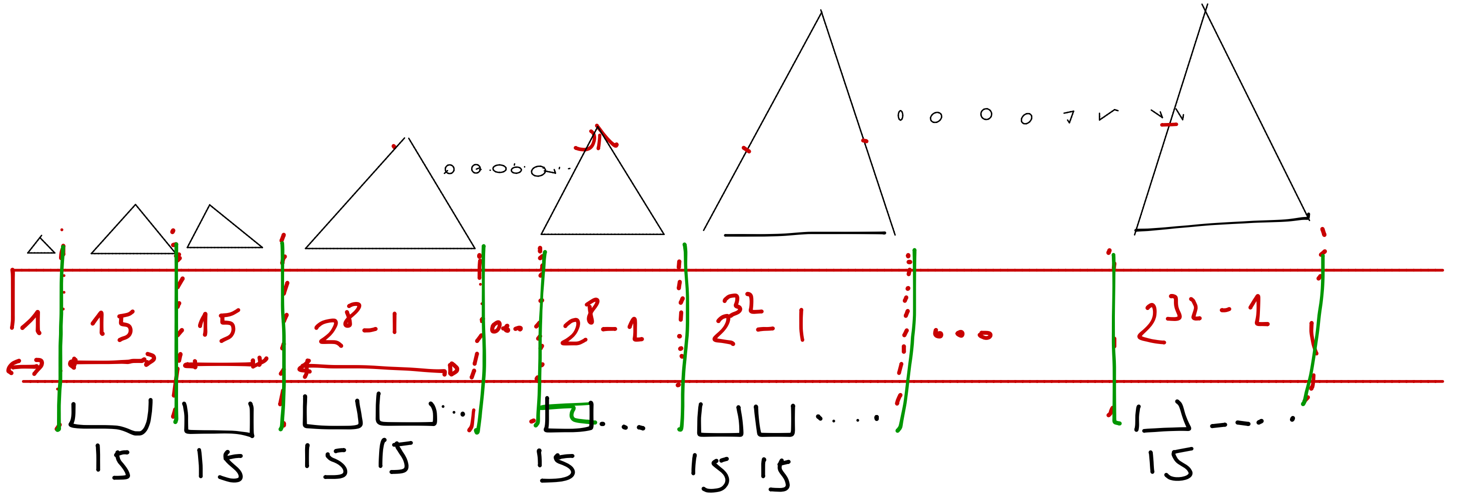
Property Split always produces top/bottom trees of height a power of 2.

$\Longrightarrow$ Possible sizes of these subtrees are

$$1, 3, 15, \underbrace{2^8-1}_{\text{size } \Theta(\sqrt{S})}, \underbrace{2^{16}-1}_{\text{size } S}, \underbrace{2^{32}-1}_{\text{size } \Theta(S^2)}, 2^{64}-1, \dots, \underbrace{2^{2^k}-1}_{= S}, \dots$$

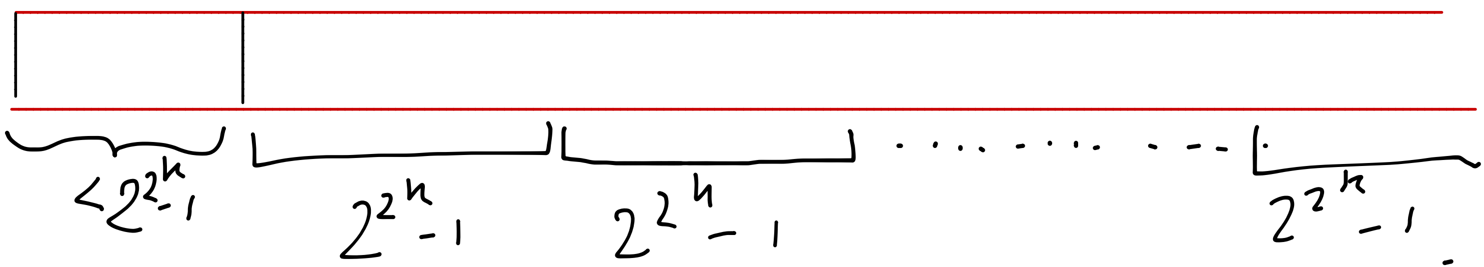
IMPORTANT For $k = 0, 1, 2, \dots$, the k-view of the array in the previous slide is looking at recursion when size is $2^{2^k}-1$:

SEE $\Longrightarrow$

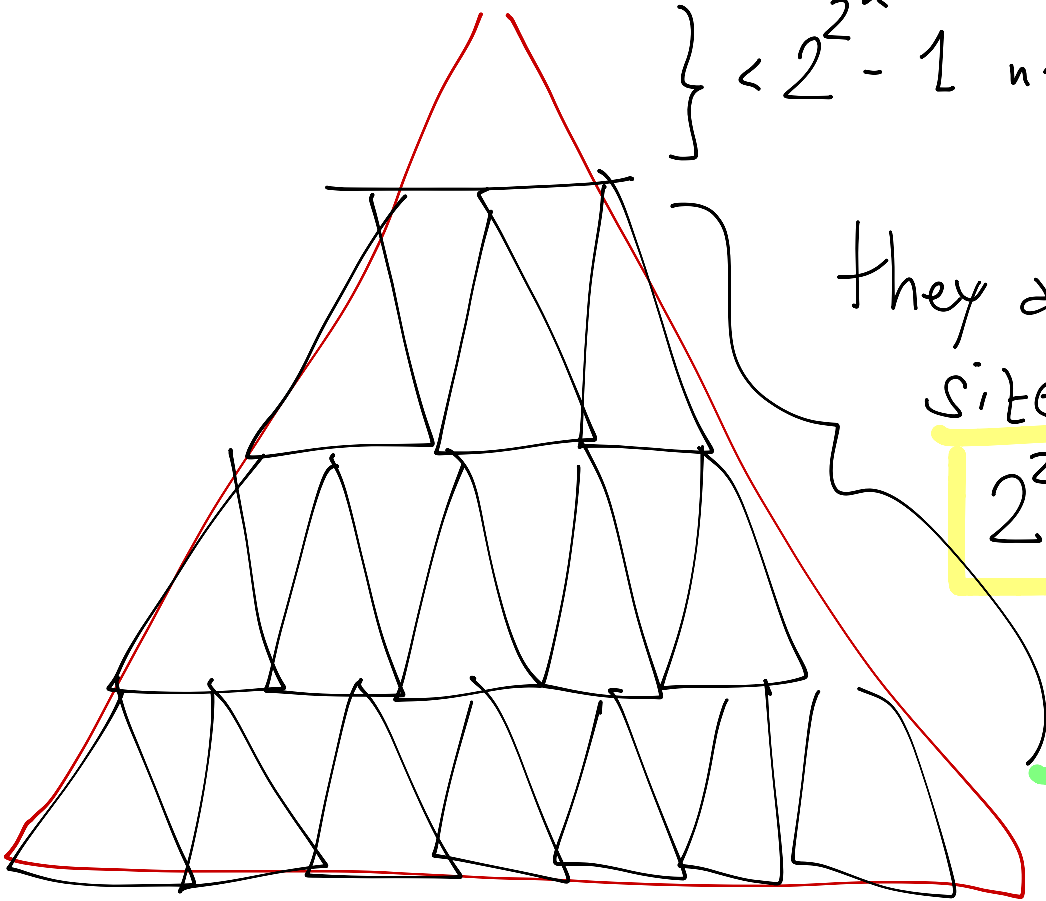


$k=2 \Rightarrow 2^{2^k}-1 = 15$

In general ...



This corresponds to this picture

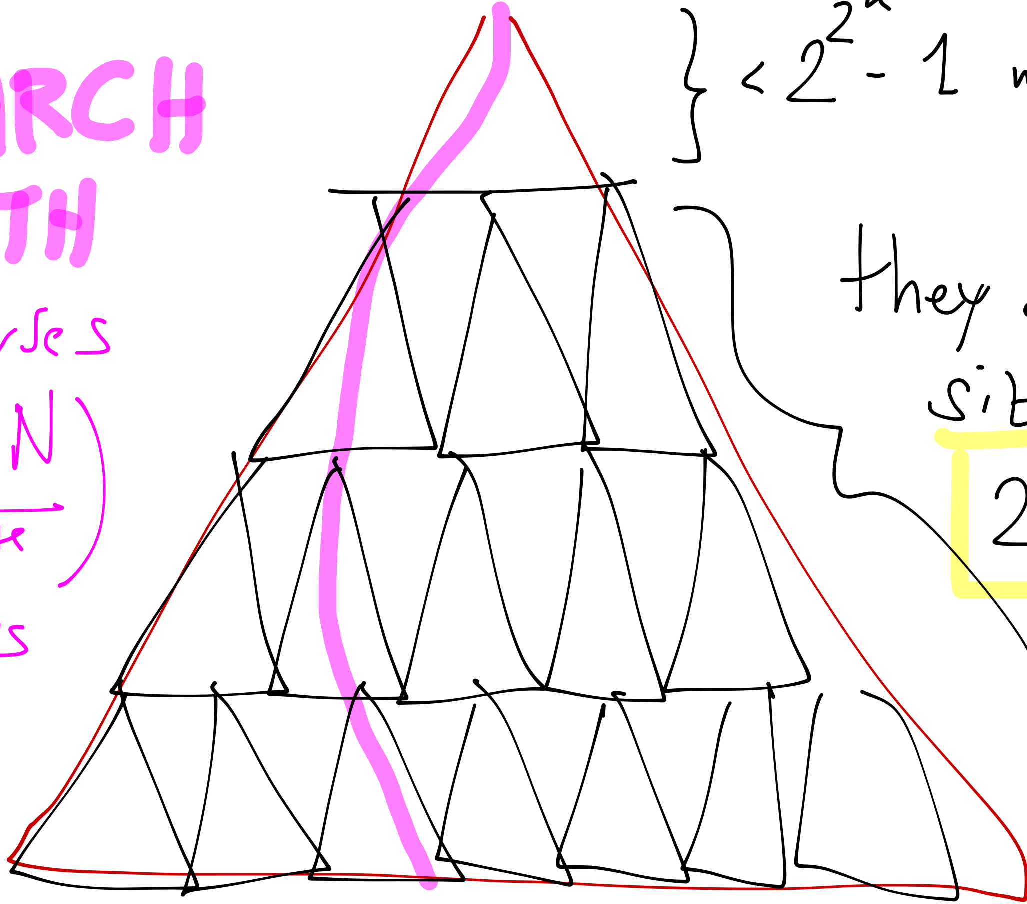


$< 2^{2^k} - 1$ nodes

they all have size $\boxed{2^{2^k} - 1}$ and height, $\boxed{2^k}$

This corresponds to this picture

SEARCH PATH traverses $O\left(\frac{\lg N}{2^k}\right)$ subtrees



$\} < 2^{2^k} - 1$ nodes

they all have size $\boxed{2^{2^k} - 1}$ and height $\boxed{2^k}$

Each subtree is stored in a contiguous segment of memory

- Suppose $B = 2^{2^h} - 1 \Rightarrow$ search path accesses

$$O\left(\frac{\lg N}{2^h}\right) \text{ blocks}$$

$$\parallel$$

$$O(\lg_B N) \text{ as } 2^h = \Theta(\lg B)$$

- suppose $2^{2^k} - 1 < B < 2^{2^{k+1}} - 1$

  (there must exist k with this property since $B \leq N$)

$\Rightarrow$ $2^k = \Theta(\lg B)$ and each subtree of size $2^{2^k} - 1$ is now stored in at most <u>two</u> blocks of size B

$\Rightarrow$ still $O\left(\frac{\lg N}{2^k}\right) = O(\log_B N)$ is the I/O cost.

$\blacksquare$