

For answers that involve filling-in a , fill-in the shape completely: .

1. Mark all of the following true expressions about the height of a binary search tree of size N .

$O(\log N)$

$\Theta(\log N)$

$\Omega(\log N)$

$O(N)$

$\Theta(N)$

$\Omega(N)$

Comprehending. In class, we discussed the problem with binary search trees: depending on the order we insert items into the tree, they are sometimes bushy but other times spindly. This should clue us into performing case analysis. Because the question does not specify the shape of the tree nor specify worst case or best case analysis, we need to do separate case analysis and then summarize the results for the overall order of growth for the height of the tree.

Modeling: Best Case. The best case is the smallest order of growth: in this case, the smallest height. This occurs when the tree is bushy. This is logarithmic because, given a bushy tree of size N , there are about $N/2$ items on the last level, $N/4$ items on the level above the last level, $N/8$ items on the level above that... This continues until the root where we have 1 item.

How many levels are in this tree? We know there are N total items, divided into levels with the following expression.

$$\frac{N}{2} + \frac{N}{4} + \frac{N}{8} + \dots + 4 + 2 + 1$$

How many levels (different terms) are in this equation? There are almost exactly $\log_2 N$ of these terms.

Modeling: Worst Case. The worst case is the greatest order of growth: in this case, the greatest height. This occurs when the tree is spindly. This height grows exactly linear with the number of items in the tree since it's just like an `OrderedLinkedList`, hence the order of growth of the height in the worst case is $\Theta(N)$.

Formalizing: Overall. Since this question did not specify best or worst case, we should give our answer with respect to all possible scenarios, best and worst case inclusive. The height of any binary search tree of size N can be anywhere between $\log_2 N$ and N . The Change of Base formula tells us that $\log_2 N \approx \log N$ since all logarithm bases are a constant factor from each other.

Therefore, the overall order of growth is bounded below by $\Omega(\log N)$ and above by $O(N)$.

2. Mark all of the following true expressions about the height of a binary heap of size N .

$O(\log N)$

$\Theta(\log N)$

$\Omega(\log N)$

$O(N)$

$\Theta(N)$

$\Omega(N)$

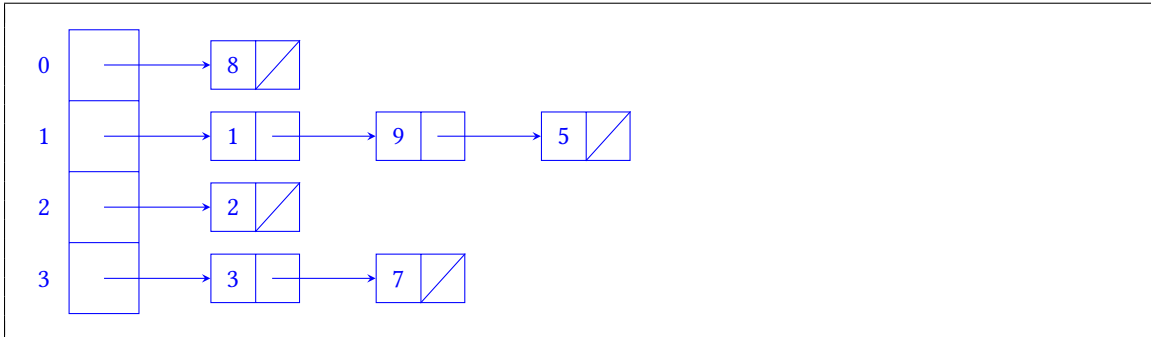
This problem is somewhat simpler than the first question since heaps are always balanced!

Comprehending. In class, we discussed that heaps are always balanced.

Modeling. More specifically, their height is always about $\log_2 N$.

Formalizing: Overall. The Change of Base formula tells us that $\log_2 N \approx \log N$ since all logarithm bases are a constant factor from each other. The order of growth of the height of a heap of size N is in $\Theta(\log N)$. This also implies that the order of growth of the height is also in $O(\log N)$ and $\Omega(\log N)$. But the height is also in $O(N)$ since all logarithms are bounded above by linear functions.

- Draw the separate-chaining hash table with $M = 4$ buckets that results from inserting the following items in this order: 1, 2, 3, 7, 8, 9, 5. Assume that the hash function for integers returns the value of the integer and that items are added to the end of the linked list.



To insert an item into a separate-chaining hash table, we first compute the item's hash code. For this question, we assume that the hash function for integers returns the value of the integer. Then, we compute the bucket index based on the $M = 4$ buckets. If the item is not already in the bucket, we add it to the end of the linked list. Item 1 goes to bucket index 1. Item 2 goes to bucket index 2. Item 3 goes to bucket index 3. Item 7 also goes to bucket index 3 because $7 \bmod 4$ is 3. (The remainder after dividing 7 by 4 is 3.)

- Draw the separate-chaining hash table after resizing to $M = 8$ buckets.

