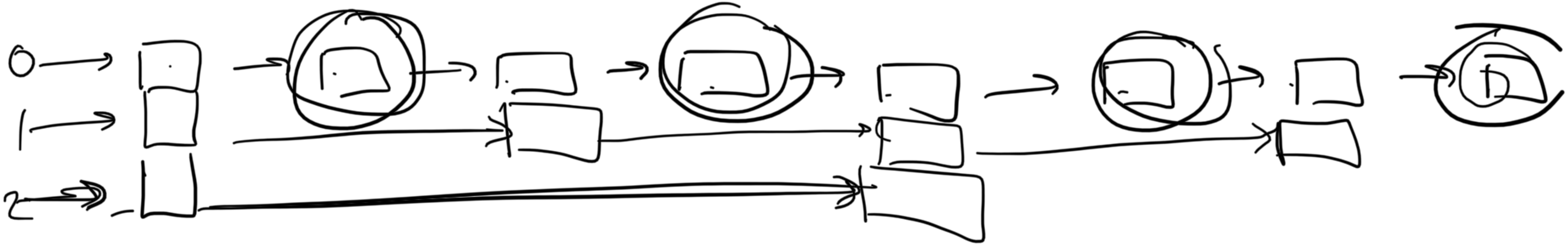
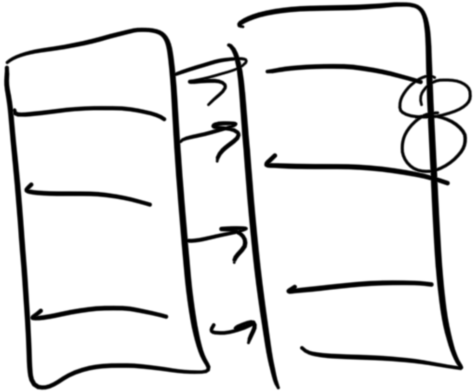
Skiplists

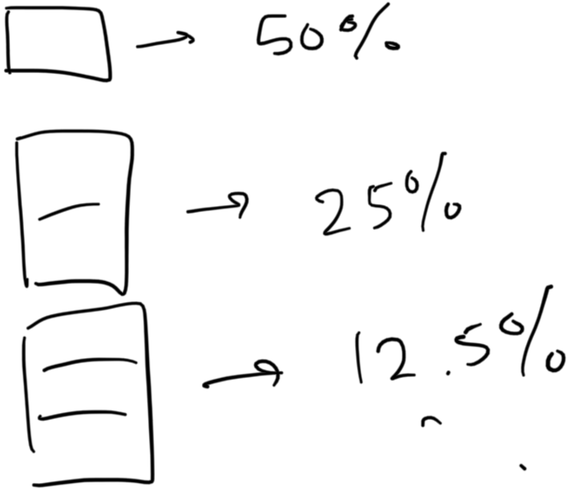
A probabilistic data structure...

The Skiplist...

- ...does not *guarantee* O(logN) runtime for operations,...
- ...does provide O(logN) performance *with a high probability*.
- ...can provide a balance between good performance and ease of implementation.







$$\frac{1}{2^{10}}$$

Ideally...

- You would have enough levels so that the ideal search would skip half the values, then a quarter of the remaining values, then an eighth of the remaining values, etc…
- Which would mean that a *search* (or an *insert*) would be O(logN) with high probability.
- Space: If the head node isn't counted,
  - ½ the nodes should have 1 pointer...
  - ¼ should have 2 pointers...
  - ⅛ should have 3 pointers…
  - and so on…
- However, we don't enforce the *ideal* situation.
- Instead, we rely on probability and randomization.

Inserting & removing nodes...

- Instead, a new node is assigned a level at random where there is
  - a 50% chance that it will have 1 pointer (the lowest level),
  - a 25% chance that it will have 2 pointers (the second lowest level)...
  - and so on...
- This is what makes skiplists *probabilistic.*
- https://opendsa-server.cs.vt.edu/ODSA/Books/CS3/html/SkipList.html#id1

$$N + \frac{N}{2} + \frac{N}{4} + \cdots + 1 \Rightarrow O(N)$$

Analysis

$\frac{1}{2^N}$

- Consider the extremes:
    - All the nodes are at a high level, so lots of pointers. In that case, the insert cost could be quite high (even O(N)). (Basically, this would be a linked list with extra pointers).

      $(\frac{1}{2})(\frac{1}{2})(\frac{1}{2}) \ldots (\frac{1}{2})$
    - All the nodes are at a low level (e.g. level 0), so they only have 1 pointer each, which is basically just a linked list. In that case, insert and search could be O(N).
- The good news: These extremes are unlikely to happen. For example, there is only 1/1024 chance that 10 nodes in a row will be in level 0.
- Similar to Quicksort, we accept that the randomization will make sure that the worst case is unlikely to happen.
- Expected memory use: O(N)

Skiplist vs. BST

- Recall that a BST made from random keys is expected to have a height of approximately *logN*.
- A skiplist's performance is expected to be about the same as a BST in which the keys are inserted in random order.
- The key difference: There is no randomization built into a BST.
- In fact: The larger the skiplist gets, the closer it will get to O(logN) performance because it becomes less and less likely that the worst case will happen.