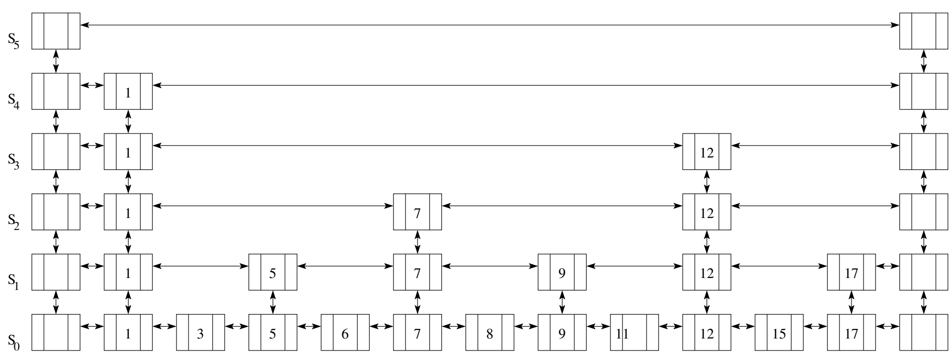
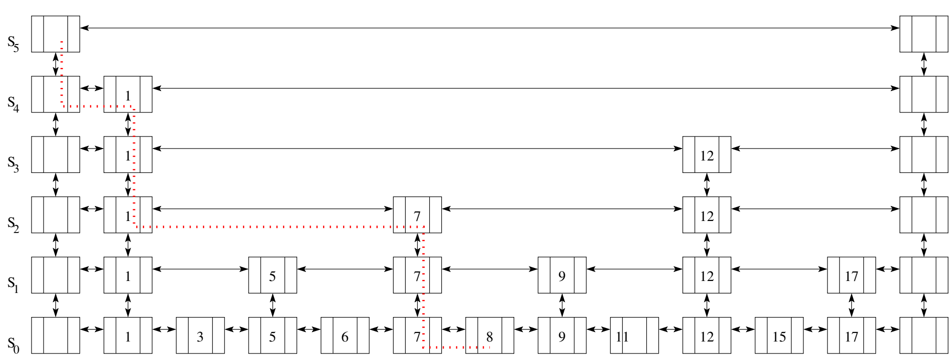
Skip Lists

- There are several schemes for keeping search trees reasonably balanced with $O(\log n)$ height. Somewhat complicated.
- When we discussed Quick-sort we saw how randomization can lead to good expected running times.
- Randomization can be used to obtain a very simple search structure with expected performance $O(\log n)$ for all (independent of data!)
- Idea in a skip list is best illustrated if we try to build a "search tree" on top of double linked list:
  - Insert elements $-\infty$ and ∞
  - Repeatedly construct double linked list (level S_i) on top of current list (level S_{i-1}) by choosing every second element (and link equal elements together)
- Since every level is half teh size of the one below, it follows that there are $O(\log n)$ levels.



- *Search*(e): Start at topmost left element. Repeatedly drop down one level and search forward until max element $\leq e$ is found.
- Example: Search for 8



- How to *Insert*/*Delete* ? seems hard to do in better than $O(n)$ time since we might need to rebuild the entire structure after one of the operations.
- Idea: level S_i consist of a randomly generated subset of elements at level S_{i-1}.
  - To decide if an element on level S_{i-1} should be on level S_i, we flip a coin and include the element if it is head.
    $\Downarrow$
    Expected size of S_1 is $\frac{n}{2}$
    Expected size of S_2 is $\frac{n}{4}$
    $\vdots$
    Expected size of S_i is $\frac{n}{2^i}$
    $\Downarrow$
    Expected height is $O(\log n)$
- Operations:
  - $Search(e)$ as before.
  - $Delete(e)$: Search to find e and delete all occurrences of e.
  - $Insert(e)$:
    * search to find position of e in S_0
    * Insert e in S_0.
    * Repeatedly flip a coin; insert e and continue to next level if it comes up head.
- Running time of all the operations is bounded by search running time
  - Down search takes $O(height) = O(\log n)$ expected.
  - Right search/scan:
    * If we scan an element on level i it cannot be on level $i+1$ (because then we would have scanned it there)
      $\Downarrow$
    * Expected number of elements we scan on level i is the expected number of times we have to flip a coin to get head
      $\Downarrow$
    * We expect to scan 2 elements on level i
      $\Downarrow$
    * Running time is $O(height) = O(\log n)$ expected.
- Note:
  - We only really need forward and down pointers.
  - Expected space use is $\sum_{i=0}^{\log n} \frac{n}{2^i} \leq n \cdot \sum_{i=0}^{\infty} \frac{1}{2^i} = O(n)$.