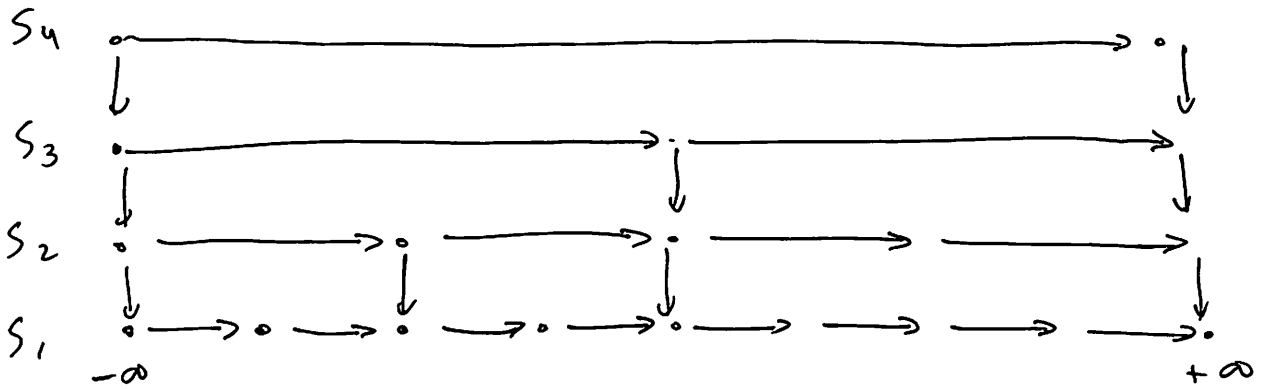
Skip List



Idea: keep levels of "express" lists s.t. gaps grow exp.

```
def search(k)
    curr ← top left node
    while curr.down ≠ ∅
        curr ← curr.down
        while curr.right.key ≤ k
            curr ← curr.right
    // either curr.key = k, or
    // curr.key < k and curr.right.key > k
    return curr
```

In a perfect skip list

- ★ height is $\log_2 n$
- ★ $|S_i| = \frac{|S|}{2^{i-1}}$
- ★ search down moves = height of tree
- ★ search right moves ≤ down moves

⇒ $O(\log n)$ searches

Randomization

For each $e \in S_1$

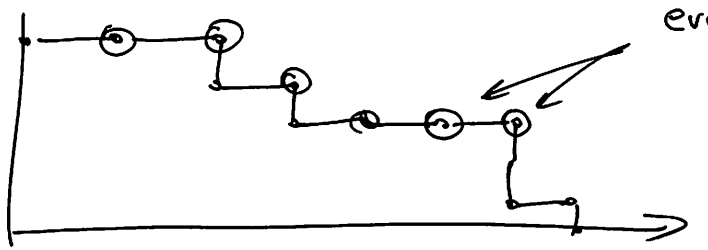
$k \leftarrow$ # flips until heads

add e in $S_1, S_2, \ldots S_k$

* $E[|S_i|] = \frac{|S_1|}{2^{i-1}}$
* space complexity is $O(\sum_i |S_i|) = O(|S_1|) = O(n)$
* $Pr[e \in S_k] = \frac{1}{2^{k-1}}$

$\Rightarrow Pr[e \in S_{2\log n+1}] = \frac{1}{n^2} \Rightarrow Pr[|S_{2\log n+1}| \neq 0] \leq \frac{1}{n}$

* # down moves $\leq O(\log n)$ with high prob
* to bound # right moves

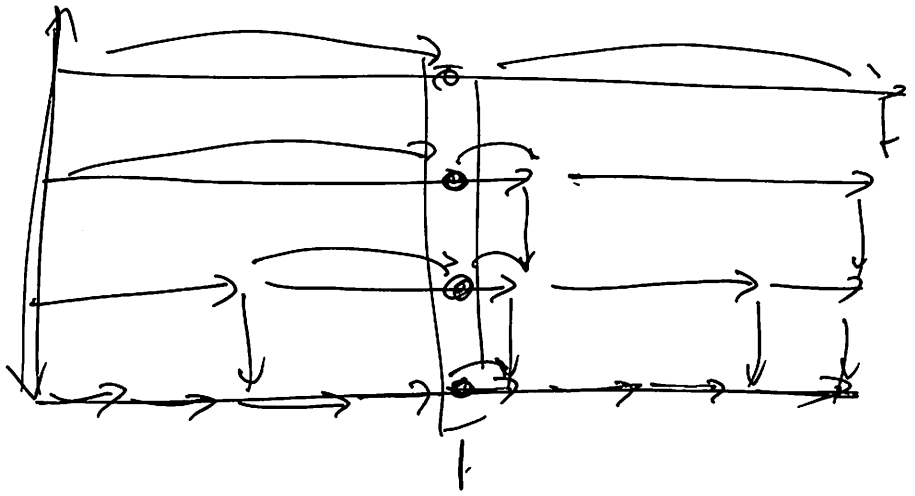


every time we went "left" is because we got heads

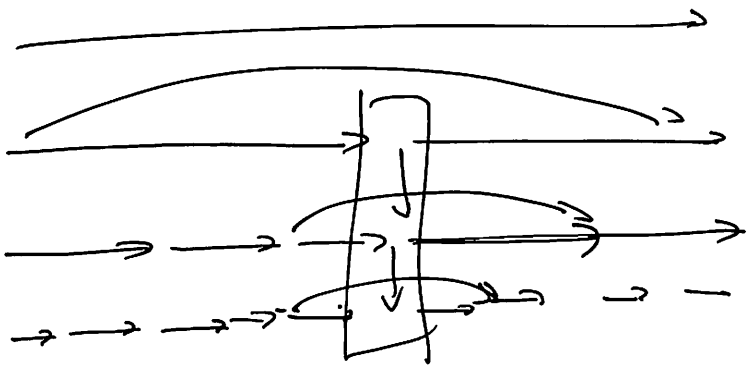
$\Rightarrow$ we expect to see (roughly) half of them being heads $\Rightarrow$ # right moves $= O($# left moves$)$

Inset / Delete

To insert, find the positn where the new key should go, flip a coin to get the height of that node and insert in correspondly lists



To delete a positn remove it from all the lists it belongs to



Each operatn can be done in O(height) time