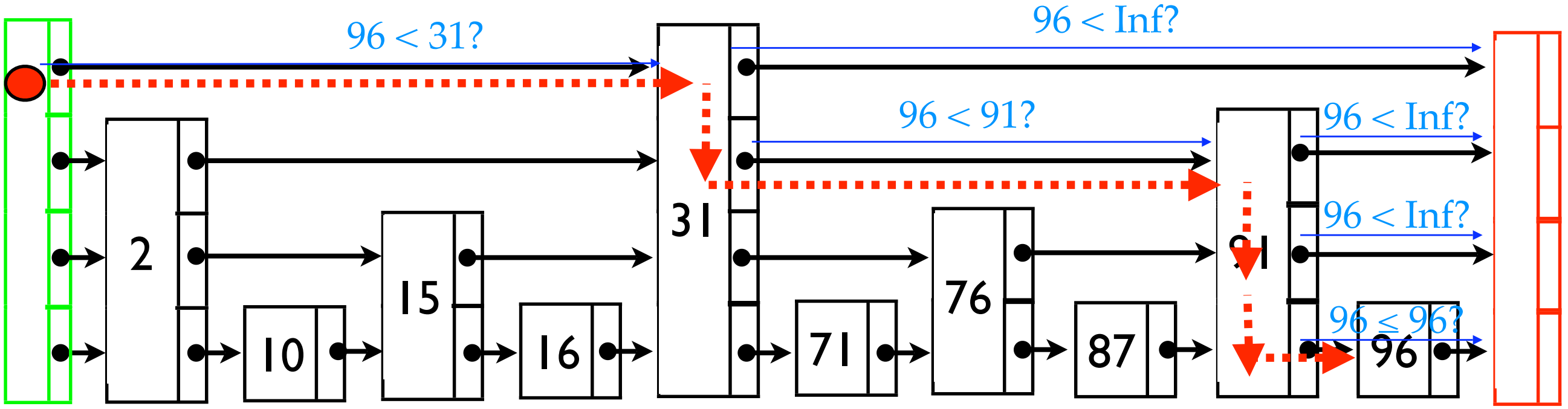
Perfect Skip Lists, continued

Comparison

Change current location

Find 96



When search for k:

If k = key, done!

If k < next key, go down a level

If k ≥ next key, go right

Search Time:

- O(log n) levels --- because you cut the # of items in half at each level

- Will visit at most 2 nodes per level:
  If you visit more, then you could have done it on one level higher up.

- Therefore, search time is O(log n).

Insert & Delete

- Insert & delete might need to rearrange the entire list
- Like Perfect Binary Search Trees, Perfect Skip Lists are *too* structured to support efficient updates.
- Idea:
  - Relax the requirement that each level have exactly half the items of the previous level
  - Instead: design structure so that we *expect* 1/2 the items to be carried up to the next level
  - Skip Lists are a *randomized* data structure: the same sequence of inserts / deletes may produce different structures depending on the outcome of random coin flips.

Randomization

- Allows for some imbalance (like the +1 -1 in AVL trees)
- Expected behavior (over the random choices) remains the same as with perfect skip lists.
- Idea: Each node is promoted to the next higher level with probability 1/2
  - Expect 1/2 the nodes at level 1
  - Expect 1/4 the nodes at level 2
  - ...
- Therefore, expect # of nodes at each level is the same as with perfect skip lists.
- Also: expect the promoted nodes will be well distributed across the list

Randomized Skip List: