Skip Lists

→ provide a clever compromise to efficiently support search/update operations

→ a skip list for map M consists of a series of lists $\{S_0, S_1, \ldots, S_n\}$
  → each list S_i for map M stores a subset of the items from the map sorted by inc. keys
  → each list also has 2 sentinel keys denoted $-\infty$ and $+\infty$, where $-\infty$ smaller than every possible key and $+\infty$ larger than every possible key

→ skip lists also satisfy:
  1.) List S_0 contains every item of map M
  2.) For $i = 1, \ldots, h-1$, list S_i contains randomly generated subset of items of list S_{i-1}
  3.) List S_n contains only $-\infty$ and $+\infty$

→ it is customary to view lthink of skip lists with list S_0 at bottom and lists $S_1 \ldots S_n$ above it

→ we refer to h as the height of skip list S

→ intuitively, the lists are set up so that S_{i+1} contains more or less alt. items of S_i

Search/Update in Skip Lists

→ Searching: suppose we are given search key k;
  → we begin skip search by setting position at top left position of S
  → then perform following steps:
    1.) if S.below(p) is None, search terminates -- we are at bottom and have located the item in S with the largest key <= search key k. Otherwise, drop down to next lower level in current tower by setting p = S.below(p)
    2.) Starting at position p, move p forward until it is at rightmost position on present level such that key(p) ≤ k. Called scan forward step
    3.) Return to step 1

```
Algo SkipSearch(k)
  p = start
  while below(p) != None
    p = below(p)
    while k ≥ key(next(p))
      p = next(p)
  return p
```