Dictionary data structure: fully dynamic
Search, insert, delete

<table>
<thead>
<tr>
<th>Arrays</th>
<th>Linked lists</th>
<th>BST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. for sorted arrays: log n time on search (because of random access)</td>
<td>Sequential search: ( O(n) ) even for sorted list</td>
<td>( O(\log n) )</td>
</tr>
<tr>
<td>2. Insert/Delete are not efficient</td>
<td>Insert/Delete: ( O(1) ) given that we know location</td>
<td>( O(\log n) ) for balanced AVL, red-black, B-trees, AB-trees, BB-trees, ST-tree</td>
</tr>
</tbody>
</table>
find an interval: consecutive elements $x_i \leq 49 \leq x_{i+1}$

extra pointers are sometimes called fingers
Total size of all links:
\[ n + \frac{n}{2} + \frac{n}{4} + \cdots = O(n) \]

Search time:
Time to search for \( L_t \) +
\[ \sum \left( \text{Time to refine the search from all levels } L_i \text{ to } L_{i-1} \right) \]
\[ = O(1) + \# \text{levels} \times O(1) \]
\[ = O(\# \text{ levels}) = O(\log n) \]
The efficiency of the data structure is based on the invariant:
It takes $O(1)$ time to descend for $L_i \rightarrow L_{i-1}$ (expected $O(1)$).

This is easy to maintain in the static case (every 2<sup>nd</sup> element)
but very difficult with insert/delete.

Construct $L_i$ for $L_{i-1}$ using coin tossing:
If it is heads, then promote else not.

**Skip List**
# Elements in a level, # levels depend on outcomes of coin tosses and may be thought of as random variables

**Space of the data**

**Expected size of the Skip list**

\[
\text{Expected size} \leq \text{expected # copies for each element}
\]

\[
\downarrow
\]

how many times (expected) do we toss a coin to get a "tail"

\[
= 2 \quad (\text{follows from geometric distribution})
\]

\[
= 2n
\]

**Expected length of the path within a level** = 2

\[
\begin{array}{c}
1 \quad 2 \quad 3 \quad 4 \quad 5 \\
\hline
H \quad \text{red line}
\end{array}
\]
How many levels?

Two ways to stop the process:

1. We fix the height to be
   \[ k = 2 \log n \]
   \[ \text{where } n \text{ elements in level } 2 \log n \]

2. What is the expected number of levels when the size of the last level is less than 10?