Pattern Matching:

INPUT: \( T = t_1 t_2 t_3 \ldots \ldots \ldots t_n \in \Sigma^* \)
\[ P = p_1 p_2 p_3 \ldots \ldots \ldots p_m \in \Sigma^* \]

OUTPUT: All the indices \( i \) such that \( T_i = t_i t_{i+1} t_{i+2} \ldots \ldots \ldots t_{i+m-1} = P \)

Algorithm: for \( i = 1 \) to \( (n-m+1) \) do
check if \( T_i = P \)
using the previous algorithm (for checking the equality of two integers)
if yes, output \( i \);

Analysis: Let the prime be picked from the interval \([1,k]\) \( \Rightarrow \)
# of such primes \( = \Theta (k/\log k) \)

probability of an incorrect answer for a specific \( i = m/(k / \log k) \)
\( \Rightarrow \) probability of an incorrect answer for at least one such \( i \) is \( \leq \ n.m/(k / \log k) \)

we want this to be \( \leq n^{-\alpha} \)

\( \Rightarrow \ n.m/(k / \log k) = n^{-\alpha} \)

\( \Rightarrow \ m.n^{\alpha+1} = k/(\log k) \)

pick \( k \) to be \( (m.n^{\alpha+1}) \log (m.n^{\alpha+1}) = \Omega (m.n^{\alpha+1} \log n) \).

Note: \( T_i = 2^{m-1} t_i + 2^{m-2} t_{i+1} + \ldots \ldots \ldots \ldots + 2 t_{i+m-2} + t_{i+m-1} \)
\( T_{i+1} = 2^{m-1} t_{i+1} + 2^{m-2} t_{i+2} + \ldots \ldots \ldots \ldots + 2 t_{i+m-1} + t_{i+m} \)
\( 2T_i = 2^m t_i + 2^{m-1} t_{i+1} + 2^{m-2} t_{i+2} + \ldots \ldots \ldots \ldots + 2 t_{i+m-1} \)
\( T_{i+1} = 2T_i - 2^m t_i + t_{i+m} \).
The above equality implies that for each \( i \ (1 \leq i \leq n-m+1) \), checking if \( T_i = P \) takes only \( O(1) \) time.
As a result, the total runtime is \( = O(n) \).

We can convert this into a Las Vegas algorithm by brute force checking for every
“HIT”. A “HIT” occurs for position i if $T_i \mod p = P \mod p$, where p is the prime number used. The worst case runtime of this algorithm is $\Omega(mn)$.

**RANDOMIZED SKIP LIST:**

A randomized skip list is a data structure that can be used to realize a dictionary, i.e., a data structure that supports these three operations: SEARCH, INSERT, and DELETE.

Let S be a given ordered set.
A leveling of S with r levels is a sequence:
$\mathcal{L}_r \subseteq \mathcal{L}_{r-1} \subseteq \cdots \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_1$ where $\mathcal{L}_1 = S$ & $\mathcal{L}_r = \emptyset$

**Definition:**
The level of any element $x$ is $\ell(x) = \text{Max } i$ such that $x \in \mathcal{L}_i$.

**Definition:**
An interval at any level is nothing but an interval of two successive elements. The following is an example where $S = \{2, 3, 5, 15, 17, 28, 31, 45, 62, 75\}$. Assume that the two elements $-\infty$ and $+\infty$ are members of each level. Using the intervals of the different levels we can construct a tree as shown below.

```
-\infty 17 45 +\infty
-\infty 15 17 45 +\infty
-\infty 3 15 17 45 62 +\infty
-\infty 2 3 5 15 17 28 31 45 62 75 +\infty
```

**TREE:**
Definition:
For any element x, let $I_j(x)$ stand for the interval that x belongs in level j.

SEARCH(x):
Go through: $I_r(x), I_{r-1}(x), I_{r-2}(x), \ldots \ldots \ldots \text{until the answer is found.}$

TIME NEEDED: $\sum_{j=r}^{1} c(I_j(x))$ where $c(I_j(x))$ is the # of children of $I_j(x)$.

$\text{Prob}[\text{level(x)} = h] = (\frac{1}{2})^{h-1}(1/2)=(1/2)^h$
$\text{Prob}[\text{level(x)} > h] = (\frac{1}{2})^{h+1}[1 + \frac{1}{2} + 1/4 + \ldots \ldots ] \leq (\frac{1}{2})^h$
$\text{Prob}[\exists x \text{ whose height is } > h] \leq n \cdot (\frac{1}{2})^h$
we want this to be $\leq n^\alpha$
$\Rightarrow n^\alpha = n \cdot (\frac{1}{2})^h$
$\Rightarrow 2^h = n^{\alpha+1}$
$\Rightarrow h = (\alpha+1) \log (n)$
$\Rightarrow$ The height of the tree is $\Omega(\log n)$

What is $E\left[\sum_{j=r}^{1} c(I_j(x))\right]$?
If some node Q at level j has q children, this could only be because the elements $x_2, \ldots, x_{q-1}$ were not picked to be in $L_j$ & they were in $L_{j-1}$. The # of such elements (that are not in $L_j$) is upper bounded by a Geometric Distribution with parameter $\frac{1}{2}$.

$\Rightarrow$ the expected value = 2

$\Rightarrow E[c_j(I)] = O(1)$ for any interval I

$\Rightarrow E[\sum_{j=r}^1 c(I_j(x))]$

$\Rightarrow (1 - n^{-\alpha})O(\log n)O(1) + n^{-\alpha}O(n)$

$\Rightarrow O(\log n)$


**INSERT(x):**

Pick a random level for x. If $\ell(x) > r$ increment r by 1. Use the search algorithm to find a relevant place for x. Some of the intervals may have to be split.

Expected time = $O(\log n)$.
Delete also is processed likewise.

**Theorem:** In a random skiplist we can perform the following operations in an expected $O(\log n)$ time: SEARCH, INSERT, and DELETE.