**Project Report: Skip List Implementation in C++**

**1. Introduction**

A Skip List is a probabilistic data structure that allows for fast search, insertion, and deletion operations. It combines elements of ordered linked lists and binary search trees, enabling average-case logarithmic performance for key operations.

Skip Lists were proposed by William Pugh in 1990 as a simpler alternative to balanced trees, with the ability to be implemented efficiently with randomization.

**2. Objectives**

* Implement the Skip List in C++.
* Analyze its performance in terms of time and space complexity.
* Compare it with other data structures like Linked Lists and Binary Search Trees.
* Provide a user interface for dynamic operations.

**3. Methodology**

* **Language:** C++
* **Features Implemented:**
  + Insertion, Deletion, and Search
  + Random level generation
  + Level-based node structure
  + User-friendly command-line interface

**Skip List Node Structure:**

class Node {

int key;

Node \*\*forward;

};

**Operations Implemented:**

* insert(int key)
* deleteKey(int key)
* search(int key)
* display()

**Randomization:** The level of each node is decided by a random function with a probability factor P = 0.5.

**4. Analysis**

**Time Complexity:**

| **Operation** | **Average Case** | **Worst Case** |
| --- | --- | --- |
| Search | O(log n) | O(n) |
| Insert | O(log n) | O(n) |
| Delete | O(log n) | O(n) |

**Space Complexity:** O(n log n) due to forward pointers.

**Comparison with Other Structures:**

* Better performance than Linked Lists for search operations.
* Easier to implement and maintain than AVL/Red-Black Trees.

**5. Results**

The program was tested with different sizes and sequences of input values. It successfully performed insertions, deletions, and searches, while maintaining the probabilistic balancing properties of a Skip List.

Sample operations:

Insert: 10, 20, 30

Search: 20 -> Found

Delete: 10

Display: Shows level-wise structure

**6. Conclusion**

The Skip List is a powerful and elegant data structure that balances efficiency and simplicity. Its probabilistic nature makes it ideal for applications requiring fast dynamic set operations.

**7. Future Work**

* Implement concurrency for multithreaded environments.
* Add visualization using graphics libraries.
* Analyze performance with real-world datasets.

**8. References**

* William Pugh, "Skip Lists: A Probabilistic Alternative to Balanced Trees", Communications of the ACM, 1990.
* <https://en.wikipedia.org/wiki/Skip_list>
* GeeksforGeeks, "Skip List"

**End of Report**