**VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY**

**UNIVERSITY OF SCIENCE**

**FACULTY OF INFORMATION TECHNOLOGY**

****

**PROJECT REPORT**

**Chủ đề:** Điều khiển PC từ xa qua E-Mail

**Môn học:** Mạng máy tính

*Thành viên:*   *Giảng viên:*

Vũ Trần Minh Hiếu (24127003) Thầy Đỗ Hoàng Cường

Hoàng Đức Thịnh (24127240) Cô Huỳnh Thụy Bảo Trân

Trần Viết Bảo (24127270)

Table of Contents

[1. Introduction about Skip List 1](#_Toc194947139)

[1.1. History of Skip List 1](#_Toc194947140)

[1.2. Applications of Skip List 1](#_Toc194947141)

[2. Variants and Improvements 2](#_Toc194947142)

[2.1. Deterministic Skip List 2](#_Toc194947143)

[2.1.1. Overview 2](#_Toc194947144)

[2.1.2. Characteristics 2](#_Toc194947145)

[2.1.3. Implementation Methods 2](#_Toc194947146)

[2.1.4. Applications 2](#_Toc194947147)

[2.2. Concurrent Skip List 3](#_Toc194947148)

[2.2.1. Overview 3](#_Toc194947149)

[2.2.2. Characteristics 3](#_Toc194947150)

[2.2.3. Implementation Methods 3](#_Toc194947151)

[2.2.4. Applications 3](#_Toc194947152)

[2.3. Skip Tree 4](#_Toc194947153)

[2.3.1. Overview 4](#_Toc194947154)

[2.3.2. Characteristics 4](#_Toc194947155)

[2.3.3. Implementation Methods 4](#_Toc194947156)

[2.3.4. Applications 4](#_Toc194947157)

[3. Skip List Overview 6](#_Toc194947158)

[3.1. Skip List structure 6](#_Toc194947159)

[3.2. Skip List operations 6](#_Toc194947160)

[3.2.1. Initialization 6](#_Toc194947161)

[3.2.2. Insertion 6](#_Toc194947162)

[3.2.3. Deletion 7](#_Toc194947163)

[4. Skip List Operations Example: Insertion and Deletion 8](#_Toc194947164)

[4.1. Initial Skip List 8](#_Toc194947165)

[4.2. Insertion Process 8](#_Toc194947166)

[4.3. Deletion process: 10](#_Toc194947167)

[5. Time Complexity Analysis 12](#_Toc194947168)

[5.1. Search Operation 12](#_Toc194947169)

[5.2. Insertion Operation 12](#_Toc194947170)

[5.3. Deletion Operation 13](#_Toc194947171)

[5.4. Test whether a Skip List is empty 13](#_Toc194947172)

[5.5. Get the number of elements in a Skip List 13](#_Toc194947173)

[5.6. Build a Skip List from given items 14](#_Toc194947174)

[5.7. Remove all elements from the Skip List 14](#_Toc194947175)

[6. Comparison between Skip List and Regular Linked List 15](#_Toc194947176)

[6.1. Similarities 15](#_Toc194947177)

[6.2. Dissimilarities 15](#_Toc194947178)

[7. Test Cases and Experiment Results 17](#_Toc194947179)

[7.1. Statements 17](#_Toc194947180)

[7.2. Test case and Experiment results 17](#_Toc194947181)

[7.3. Analysis of test case and experiment results 19](#_Toc194947182)

[7.3.1. Algorithm accuracy 19](#_Toc194947183)

[7.3.2. Execution time and complexity analysis 19](#_Toc194947184)

[7.3.3. Conclusion 19](#_Toc194947185)

[8. Appendix 20](#_Toc194947186)

[8.1. Group Information 20](#_Toc194947187)

[8.2. Work Assignment Table and Contribution Evaluation 20](#_Toc194947188)

[8.3. Project requirement completion rate 20](#_Toc194947189)

[9. References 22](#_Toc194947190)

**Table of Tables**

[**Table 1:** Execution time analysis of Skip List operations 18](#_Toc194614253)

[**Table 2:** Group information 20](#_Toc194614254)

[**Table 3:** Work assignment table and contribution evaluation 20](#_Toc194614255)

**Table of Figures**

[**Figure 1:** Skip List 8](#_Toc194614265)

[**Figure 2:** Skip List after Inserting element 40 9](#_Toc194614266)

[**Figure 3:** Skip List after Inserting element 20 10](#_Toc194614267)

[**Figure 4:** Skip List after Deleting element 30 11](#_Toc194614268)

# Console Menu

## History of Skip List

* Skip Lists were invented by William Pugh in 1989 and first described in his paper "Skip Lists: A Probabilistic Alternative to Balanced Trees," published in the June 1990 issue of Communications of the ACM.
* Pugh developed Skip Lists as a simpler alternative to balanced trees, which required complex rebalancing operations. By using randomization instead of strict balancing rules, Skip Lists achieve good average-case performance while being much easier to implement and maintain.

## Applications of Skip List

* Skip Lists find applications in various domains:
  + **Database Systems**: Used in Redis for sorted sets, MemSQL (now SingleStore), and LevelDB.
  + **Programming Languages and Libraries**: Several standard libraries include skip list implementations, such as Java's ConcurrentSkipListMap and ConcurrentSkipListSet. .NET Framework, and various C++ libraries.
  + **Concurrent Data Structures**: Well-suited for concurrent access due to localized modifications and support for lock-free implementations.
  + **Network Applications**: Used in peer-to-peer systems, distributed hash tables, and content delivery networks.
  + **Memory-Efficient Indexing**: Provides good balance between performance and memory usage in constrained environments.

# Email Menu

## Deterministic Skip List

### Overview

* The Deterministic Skip List is a variant of the classic Skip List, introduced by William Pugh in 1990. Unlike the original Skip List which uses randomization, the Deterministic Skip List employs deterministic rules to build its data structure, ensuring more stable and predictable performance.

### Characteristics

* **Deterministic Structure:** Does not use randomization; nodes are distributed according to fixed rules.
* **Balanced Performance**: Guarantees search time in the worst case.
* **Memory Efficient**: Optimizes memory usage due to the predetermined structure.
* **Guaranteed Balance**: Not affected by uneven distributions as in randomized Skip Lists.

### Implementation Methods

* **Perfect Skip List**: The node appears at level if is a multiple of , creating a perfect structure.
* **Deterministic SkipNet**: Uses partitioning techniques to ensure balance and even distribution.
* **Biased Skip List**: Adjusts the probability of nodes appearing at certain levels based on their positions.

### Applications

* **Databases**: Used in database systems requiring stable performance.
  + **Embedded Systems**: Suitable for systems with limited resources and high determinism requirements.
* **Real-Time Applications**: Ensures consistent response times for real-time applications.
* **Routing Algorithms**: Used in network routing algorithms demanding high reliability.

## Concurrent Skip List

### Overview

* The Concurrent Skip List is a version of the Skip List specifically designed for multi-threaded environments, allowing multiple threads to access and modify the data structure simultaneously without causing consistency issues. This structure combines the advantages of Skip Lists with efficient synchronization mechanisms.

### Characteristics

* **Thread-Safe**: Designed to handle concurrent operations from multiple threads.
* **High Scalability**: Performance increases with multiple participating threads, suitable for high-load systems.
* **No Global Locks**: Uses sophisticated locking or lock-free techniques to avoid bottlenecks.
  + **High Consistency**: Ensures data consistency even with multiple concurrent operations.

### Implementation Methods

* **Lock-free Skip List**: Uses atomic operations such as Compare-And-Swap (CAS) to ensure consistency without locks.
* **Java concurrentSkipListMap**: Implementation in Java's standard library, using advanced concurrency techniques.
* **Optimized Locking Methods**: Uses fine-grained locking to minimize conflicts between threads.
* **Transactional Memory**: Employs transactional memory to simplify concurrent programming.

### Applications

* **Distributed Systems**: Used in distributed systems needing to process multiple concurrent requests.
* **Multi-threaded Caching**: Implements high-performance caching in multi-threaded environments.
* **Online Transaction Systems**: Processes multiple financial transactions simultaneously with high reliability.
* **Java Concurrent Collections**: Implementations such as ConcurrentSkipListMap and ConcurrentSkipListSet in Java.
* **Memory Management Systems**: Used in efficient memory management algorithms for multi-threaded systems.

## Skip Tree

### Overview

* Skip Tree is an extension of the Skip List concept that incorporates tree-like properties to enhance performance and flexibility. While Skip Lists are essentially linked lists with additional forward pointers to allow for fast traversal, Skip Trees combine the simplicity of Skip Lists with the structural advantages of search trees.

### Characteristics

* **Hierarchical Structure**: Organizes data in a multi-level tree-like structure while maintaining Skip List properties.
* **Improved Search Complexity**: Offers search time with better practical performance than standard Skip Lists in certain scenarios.
* **Dynamic Rebalancing**: Many implementations include mechanisms for automatic rebalancing, improving worst-case performance.
* **Space Efficiency**: Some implementations provide better space utilization compared to traditional Skip Lists.

### Implementation Methods

* **Deterministic Skip Trees**: Uses deterministic rules to build the tree structure, similar to Deterministic Skip Lists but with tree-based organization.
* **Probabilistic Skip Trees**: Employs randomization to determine the structure, similar to the original Skip List concept.
* **Self-Adjusting Skip Trees**: Dynamically adjusts the structure based on access patterns to optimize frequently accessed paths.
* **Multi-dimensional Skip Trees**: Extensions that support efficient queries on multiple dimensions or attributes simultaneously.

### Applications

* **Spatial Data Structures**: Used in geographic information systems for efficient spatial queries.
* **Database Indexing**: Provides alternative indexing mechanisms in database management systems.
* **Network Routing**: Applied in advanced routing algorithms that require hierarchical data representation.
* **Memory-Efficient Search Systems**: Utilized in systems where memory overhead must be minimized while maintaining fast search capabilities.
* **Range Query Optimization**: Particularly effective for range-based queries compared to standard Skip Lists.

# Appendix

## Group Information

**Table 2:** Group information

|  |  |  |
| --- | --- | --- |
| **No.** | **Full Name** | **Student ID** |
| 1 | Vũ Trần Minh Hiếu | 24127003 |
| 2 | Hoàng Đức Thịnh | 24127240 |
| 3 | Trần Viết Bảo | 24127270 |

## Work Assignment Table and Contribution Evaluation

**Table 3:** Work assignment table and contribution evaluation

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Member** | **Assigned Task** | **Completion Rate** |
| 1 | Vũ Trần Minh Hiếu | Code, Report | 100% |
| 2 | Hoàng Đức Thịnh | Slide, Code | 100% |
| 3 | Trần Viết Bảo | Slide, Report, Video | 100% |

## Project requirement completion rate

* **Source Code:**
  + The basic operations of Skip List have been fully implemented, including:
    - Checking if the Skip List is empty
    - Counting the number of elements in the Skip List
    - Inserting a new element
    - Deleting an element
    - Constructing a Skip List from given elements
    - Deleting the entire Skip List
  + Additionally, we have implemented some improved operations, such as:
    - Search operation
    - Finding the smallest element that is greater than or equal to
    - Finding the smallest element that is greater than
  + In the function, we also provide several illustrative examples for basic operations such as insertion, deletion, and searching in Skip List to meet the given requirements.
  + **Evaluation:** 100% completed
* **Report:**
  + The report has been completed fully according to the requirements:
    - Introduction to the history and applications of Skip List
    - Overview of some variations and improvements of Skip List and their applications
    - Step by step illustration of insert and delete operations
    - Time complexity analysis of basic operations in Skip List
    - Point out similarities and differences between Skip List and Regular Linked List
  + Additionally, we have implemented some extra features, such as:
    - Explanation of how the Skip List is structured in the attached source code file
    - Explanation of the concept behind implementing basic operations like initialization, insertion, and deletion in Skip List
    - Validation of code correctness through a test case with 10 randomized tests, each varying in the number of operations performed. Furthermore, we provide an analysis and evaluation of the correctness and efficiency of the code when handling large-scale operations
  + **Evaluation:** 100% completed
* **Video:**
  + Presentation video URL: [Presentation video](https://drive.google.com/file/d/1BFSAs5WytBK4Lmn85YHpd4U1uHihJKlD/view?usp=drive_link)
  + **Evaluation:** 80% completed
* **Conclusion:** Approximately 93% of the requirements have been fulfilled.

# References

1. Papadakis, T. (1993). *Deterministic Skip Lists* (Technical Report CS-93-28). University of Waterloo.
2. Wikipedia contributors. (n.d.). *Skip list*. Wikipedia, The Free Encyclopedia.
3. Singh, P. (2024, June 30). *Deterministic Skip List – Advanced Data Structures*.
4. Munro, J. I., Papadakis, T., & Sedgewick, R. (1993). Deterministic Skip Lists. In *Proceedings of the Workshop on Algorithms and Data Structures*.
5. Sasidharan, A. (2023). *Concurrent Deterministic Skiplist and Other Data Structures*.
6. Pugh, W. (1989). *Concurrent maintenance of skip lists*. University of Maryland.
7. Gonzalez-Beltran, A. (2008). *Range queries over skip tree graphs*.