Assignment: Implementing a B+ Tree with External Merge Sort

Due Date : 15 June 23.55

In this assignment, you will implement a B+ Tree data structure using real-world university placement data. The goal is to understand tree-based indexing, secondary indexing, and external sorting through practical application.

The task requires the use of the **C programming language** exclusively, with compilation performed using **GCC on a Linux-based system**.

Assignment will be done by 2 person groups.

Objective

1)Implement a B+ Tree where leaf nodes represent department names.

2)Link each leaf node to a linked list of universities, sorted by descending placement score.

3)Compare sequential insertion and bulk loading (via external merge sort).

Dataset

You are provided with a CSV file where each row contains:

- A unique ID (UUID)
- University Name
- Department Name
- Base placement Score

Part 1 – B+ Tree Construction

Keys: Department names sorted alphabetically

Each leaf node contains:

- A department name
- A pointer to a linked list of universities sorted by base placement score using replacement selection sort
- All leaf nodes are linked for range queries

Part 2 – Insertion Methods

Sequential Insertion: Insert each record into the B+ tree one-by-one while maintaining balance.

Bulk Loading: Sort all records first by department, then by score using **replacement selection for external sorting.** Load leaf nodes in sequence, then build internal nodes.

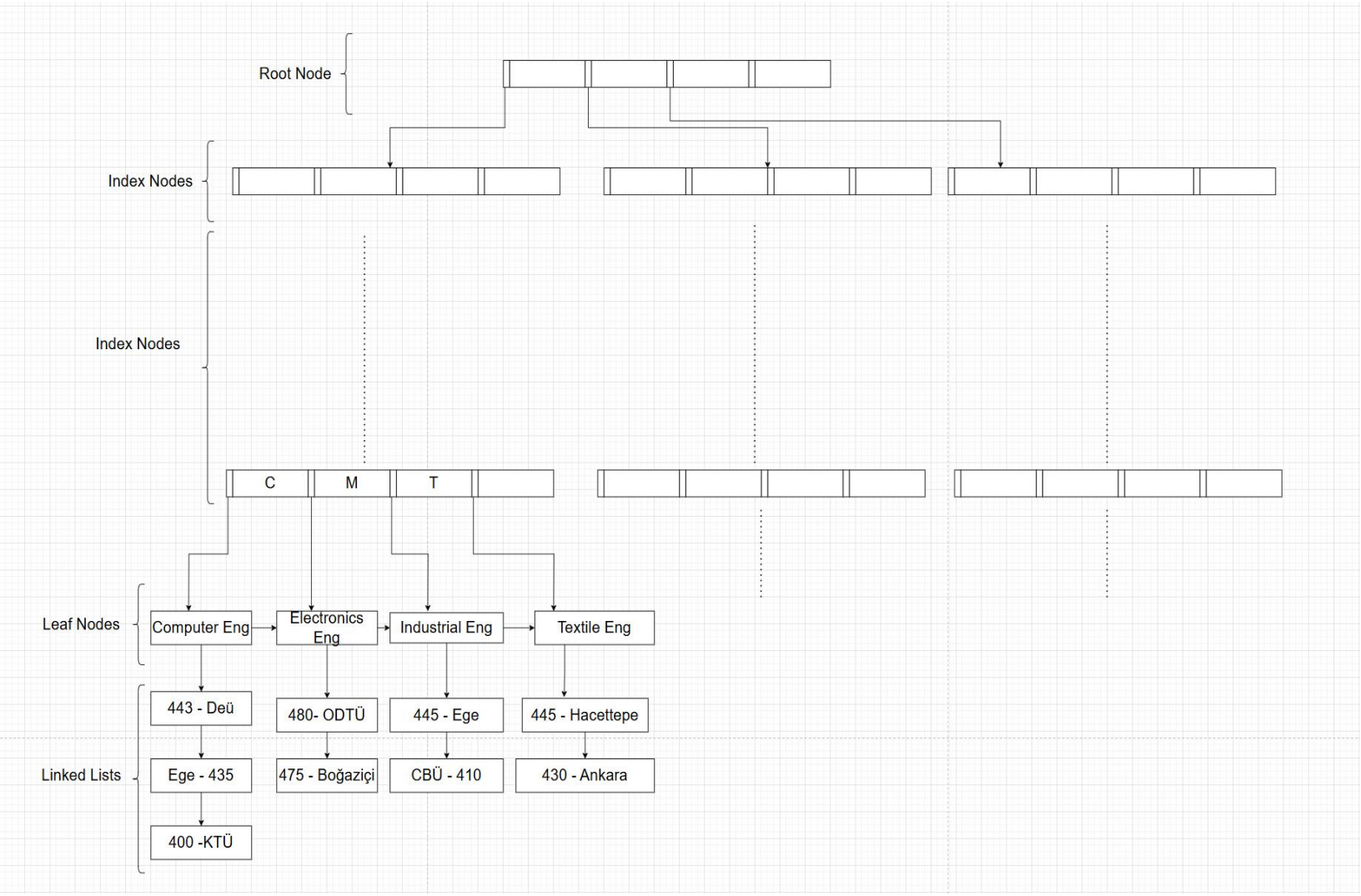
Part 3 – Comparative Analysis

Compare sequential and bulk loading by measuring:

- Memory usage
- Seek time of one record
- Tree depth
- Number of splits

Technical Notes

- Choose a B+ tree order (e.g., 4 or 8)
- Use dynamic memory structures (pointers, linked lists)
- Ensure all leaf nodes are doubly linked and point to their respective university lists



B+ Tree must be constructed according to the department names and they must be in order alphabetically. After construction of B+ Tree, then each node points to the new linked list that is also **sorted by base placement score using replacement selection sort.** The order of the B+ Tree passed as parameter and according to the degree value height must be changed.

Deliverables

The program must provide a menu-based interface allowing the user to choose between two loading options and those options must be passed as parameter to main:

1. **Sequential Insertion (Option 1)** – Insert records one by one into the B+ Tree.
2. **Bulk Loading (Option 2)** – Build the B+ Tree using external merge sort.

Depending on the selected option, the program must report and compare the following metrics:

- Number of splits during tree construction
- Estimated memory usage
- Height of the final B+ Tree
- Seek time for locating a single record

Additionally, the system must support a **search method** that accepts **two parameters**:

- **Department name (string)** — used for indexing and traversing the B+ Tree
- **University position (integer)** — indicating the rank of the university in the linked list for that department

The method should return the university name and its score based on the provided index position in the department's linked list.

Step	System Prompt / Input Example
1. Select Loading Method	Please choose a loading option: 1 - Sequential Insertion 2 - Bulk Loading (with external merge sort)
2. User Input (Option)	>> 2
3. Metrics Output (Option 1 Print Metrics)	Bulk loading completed. Number of splits: 24 Memory usage: 8.2 MB Tree height: 4 Average seek time: 0.0035 sec
4. Search Prompt (Option 2 Search)	Please enter the department name to search: >> Computer Engineering Please enter the university rank in that department: >> 3
5. Search Output	Dokuz Eylul University with the base placement score 490.

Component	Description	Weight (%)
B+ Tree Construction	Correct structure with internal and leaf nodes, linkage between nodes	30%
Linked List Integration	Proper university–score linked list for each department, sorted correctly	20%
Sequential Insertion	Step-by-step insertion while maintaining tree balance	15%
Bulk Loading (Ext. Sort)	External merge sort using replacement selection, correctly built structure	30%
Comparative Report	Performance comparison with insights, optionally supported by charts/tables	10%

Submission Guidelines

- Name your C files as <your_id>.c
- Ensure readable, documented and well structured code.
- Late submission will not be accepted.
- If you fail to follow the naming conventions you will be deduced 10 pts.
- Your submissions will be scanned among each other as well as the internet repository. Any assignments that are over the similarity threshold will be get zero.
- We strongly encourage you not to submit your assignment rather than a dishonest submission.