Ali Abdullah Ahmad
20031246
CS600

Q1. No. 20.6.3
For what values of d is the tree T of the previous exercise an order-d B-tree?
Ans.

The value order d B-tree in (a, b) with a= d/2 and b=d.

The tree T is an order-d B-tree for:

- d=8 when (a,b)=(4,8)
- d=9 when (a,b)=(5,9)

Q2. No. 20.6.21

Suppose you are processing a large number of operations in a consumer-producer process, such as a buffer for a large media stream. Describe an external-memory data structure to implement a queue so that the total number of disk transfers needed to process a sequence of N enqueue and dequeue operations are O(n/B).

Ans.

Using a Linked List for an External-Memory Queue:

A linked list in external memory can be implemented such that:

- Each node (block) stores B elements.
- Each node has a pointer to the next node on disk.

Enqueue (Insertion at the End)

- We maintain a pointer to the last node (tail).
- Insert a new element into the last block in memory.
- If the block is full:
    - Allocate a new block, update the tail pointer, and write the old block to disk.
- Disk Transfer Cost: Writing a full block requires O(1) disk transfers every B operations.
- Total cost for N enqueue operations: O(N/B) disk transfers.

Dequeue (Deletion from the Front)

- Read elements from the first block in memory.
- If the block becomes empty:
    - Free the block and move to the next one.
    - Load the next block from disk into memory (one disk transfer per block).
- Disk Transfer Cost: Reading a full block requires O(1) disk transfers every B operations.
- Total cost for N dequeue operations: O(N/B) disk transfers.

Total Disk Transfers for N Operations

- Since both enqueue and dequeue take O(N/B) disk transfers,
- The total number of disk transfers for N operations is: O(N/B)

Q3. No. 20.6.22

Imagine that you are trying to construct a minimum spanning tree for a large network, such as defined by a popular social networking website. Based on using Kruskal's algorithm, the bottleneck is the maintenance of a union-find data structure. Describe how to use a B-tree to implement a union-find data structure (from Section 7.2) so that union and find operations each use at most O(log n/log B)disk transfers each.

Ans.

Using B-Trees for Union-Find in External Memory

A B-tree is an optimized search tree for external memory, reducing disk transfers by storing multiple keys per node. Implementing union-find with a B-tree ensures efficient Find and Union operations, both executing in O(log n /log B) disk transfers.

1. Find Operation
   - Searches for an element in the B-tree and follows parent pointers to the root.
   - Path compression minimizes future lookups.
   - Complexity: O(log n /log B) disk transfers.

2. Union Operation
   - Uses Find to locate set representatives.
   - Updates the parent pointer of one root in the B-tree.
   - Union by rank/size ensures balanced merging.
   - Complexity: O(log n /log B) disk transfers.

Reason fro choosing B-Trees

- Efficient disk access: Batches multiple elements in each block, minimizing I/O.
- Reduced tree height: Unlike binary trees (O(log n)), B-trees have height O(log n/log B), lowering disk accesses.
- Supports large-scale datasets: Ideal for Kruskal's algorithm on large networks.

Conclusion:

By using B-trees, we ensure that both Find and Union operations run in O(log n/logB) disk transfers

Q4. No. 23.7.11
What is the longest prefix of the string "cgtacgttcgtacg" that is also a suffix of this string?
Ans.
Suffix: “cgtacg”

Q5. No. 23.7.15

Give an example of a text T of length n and a pattern P of length m that force the brute-force pattern matching algorithm to have a running time that is Ω(nm).

Ans.

To force the brute-force pattern matching algorithm to run in Ω(nm) time, use:

- Text T = "AAAAAAA" (length n)
- Pattern P = "AAAAB" (length m)

Worst Case

1. At each alignment, the first m−1 characters ('A's) match.
2. The last character ('B') mismatches, causing a shift by one position.
3. This repeats for O(n) shifts, each taking O(m) comparisons, leading to Ω(nm) complexity.

Q6. No. 23.7.32

EXERCISE | 23.7.32

(a) One way to mask a message, M, using a version of ***steganography***, is to insert random characters into M at pseudo-random locations so as to expand M into a larger string, C. For instance, the message,

ILOVEMOM,

could be expanded into

AMIJLONDPVGEMRPIOM.

It is an example of hiding the string, M, in plain sight, since the characters in M and C are not encrypted. As long as someone knows where the random characters where inserted, he or she can recover M from C. The challenge for law enforcement, therefore, is to prove when someone is using this technique, that is, to determine whether a string C contains a message M in this way. Thus, describe an $O(n)$-time method for detecting if a string, M, is a subsequence of a string, C, of length n.

Ans.

To check if M is a subsequence of C in O(n) time, we can use a two-pointer approach:

1. Traverse C (length n) from left to right.
2. Maintain a pointer for M (length m).
3. If a character in C matches the current character in M, move the pointer in M forward.
4. Always move the pointer in C forward.
5. If we reach the end of M, it means M is a subsequence of C.

Time Complexity Analysis:

- We traverse C once O(n).
- Each character in C is checked at most once O(n).
- The worst case is when M is absent or found at the very end.

Thus, the overall runtime remains O(n)