Romil V. Shah
20008692

CS / CPE 600
Prof. Reza Peyrovian

Homework Assignment 8
Submission Date: 11/13/2022

Q1. No. 20.6.3

For what values of d is the tree T of the previous exercise an order-d B-tree?

Sol.

From previous exercise T is a valid (a, b) tree for (4, 8) or (5, 9) tree.

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

For the values (4, 8), the value of d will be 8.
For the values (5, 9), the value of d will be 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 is $O(n/B)$.

Sol.

Linked list can be used to implement a queue so that total number of disk transfer needed to process a sequence of n enqueue and dequeue operation.
Insertion (enqueue) of an element at the end of the linked list will use O(1) disk transfers.
To dequeue elements from the front, linked list will return the block to the free memory heap when it becomes empty.

So, linked list will need O(n / B) disk transfers to process a sequence of n enqueue and dequeue operations.

Q3. No. 20.6.22

Imagine that you are trying to construct a minimum spanning tree for a large network, such as is 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.

Sol.

We can implement union-find data structure using B-tree. B-tree is a special type of self-balancing search tree in which each node can contain more than one key and can have more than two children. It is a generalized form of the binary search tree. It is also known as a height-balanced m-way tree.

Suppose initially all the nodes are in singleton trees (having height 1), the height of the tree increases by 1, when a node attached with the larger group and the number of nodes in the tree is doubled at least. Maximum number of nodes in any tree is n, so the height of the resulting tree can be at most log n.
So, find operation will take O(log n) time because visiting O(log n) nodes and each union operation will take O(1) and performing O(log n) union will take O(log n) time.

Implementing union-find data structure using B-tree with n items executes O(log n / log B) disk transfers in union and find operation.

Q4. No. 23.7.11

What is the longest prefix of the string `"cgtacgttcgtacg"` that is also a suffix of this string?

Sol.

String: “cgtacgttcgtacg”
Longest Prefix: “cgtacg”
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 $\Omega(nm)$.

Sol.

Consider a text T of length n as KKKKKKK ……. KR
Now, consider a pattern P of length m as KKKR

While comparing each letter of pattern P in test string T, the match will be found at the end of the string. The worst cast running time for brute force is O(mn), then the running time will be Ω(nm).

Q6. No. 23.7.32

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.

Sol.

We traverse both the strings M and C from left to right.
If we find a matching character, we will increment the pointer in both the strings.
Otherwise, we will increment the pointer of string C only.
If we traverse all the characters of M that means, M is a subsequence of C.

We traverse the string C which is of length n. So, the run time for this will be O(n) time.