# CS 600 Advanced Algorithms

Sri Dhanush Reddy Kondapalli

CWID: 10476150

Homework 8

## 1 R-20.6.3 For what values of d is the tree T of the previous exercise an order-d B-tree?

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
Taking (4, 8), the value of d will be 8.
Taking (5, 9), the value of d will be 9.

## 2 A-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 $O(n/B)$.

Considering some as outside memory inefficient word reference usage based on groupings Efficient sequence-based dictionary implementations. If the sequence representing a dictionary is implemented as an unsorted, doubly linked list, then insert and remove may be done with O(1) transfers each, with each block containing an item to be deleted. Furthermore, searching necessitates $\Theta(n)$ movements in the worst-case scenario, because each connection link hop we make may lead to a different block. This search time may be reduced to O(n/B) exchanges, where n represents the number of Enqueue and Dequeue operations and B is the number of hubs of the list's nodes that can fit into a block.We could, on the other hand, use a sorted array to actualize the sequence. In this case, a hunt uses binary search to perform O(log n) exchanges.

However, in the worst-case scenario, we will need $\Theta(n/B)$ transfers to accomplish an insert or removal operation since we may need to access all blocks to shift pieces up or down. As a result, implementations of sequence dictionaries are inefficient for external memory.

Only $O(log_B n) = O(logn/logB)$ transfers are required to accomplish dictionary searches and updates.

The key concept for enhancing the dictionary implementations' external-memory efficiency is to do up to O(B) internal-memory accesses to avoid a single disk transfer, where B represents the size of a block. This many internal-memory visits are performed by the disk only to move a block into internal memory, and this is only a minor portion of the cost of a disk transfer. Thus, O(B) high-speed internal-memory accesses are a minor cost to avoid a time-consuming disk transfer.

We can represent our dictionary using a multi-way search tree, which is a generalization of the (2, 4) tree data structure to a structure known as an (a, b) tree, to reduce the importance of the performance difference between internal and external memory accesses for searching.

Hence, a Buffered Repository Tree is a structure with a lower insertion cost than a higher lookup cost.

# 3 A-20.6.22 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.

Using a B-tree to implement a union-find data structure. Assume that at first, all of the nodes are in singleton trees (having height 1) When a node is joined to a bigger group and the number of nodes in the tree is at least doubled, the height of the tree grows by one. Because the maximum number of nodes in any tree is n, the height of the resultant tree can only be $\log n$.

As a result, the search operation will take O(log n) since it will visit $O(\log n)$ nodes, each union operation will take O(1), and executing O(log n)union will take O(log n).

Implementing a union-find data structure with n elements requires $O(\log_B n)$ disk transfers during the union and find operations.

## 4 R-23.7.11 What is the longest prefix of the string "cgtacgttcgtacg" that is also a suffix of this string?

The longest prefix of the string that is also a suffix of the string is **"cgtacg"**.

## 5 C-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 $\Omega(nm)$.

Consider text T as "XXX.....XY" of length n.
Consider text P as "X....XXY" of length m.
When comparing each letter of pattern P in test string T, the match will be discovered at the end of the string.
If the worst-case running time for brute force is $mn$, the running time will be $\Omega(nm)$.

## 6 A-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. Describe an O(n)-time method for detecting if a string, M, is a subsequence of a string, C, of length n.

We can follow the below method:

1. Start with the first letter as index in $M$.
2. Compare the letters in $C$, until we discover the first occurrence.
3. Then, with index as the second letter, look for the same letter in $C$ in the string following the previous letter.
4. Loop until the last letter of string $M$ appears.
5. If it succeeds, $M$ is a sub-sequence of $C$.

**Time Complexity**: The time complexity will be $O(n)$ as we traverse the string C which is of length $n$.