

Homework 4

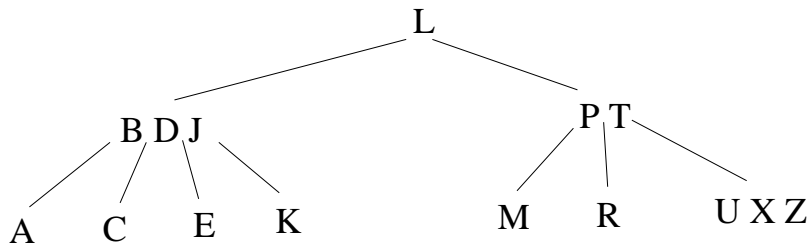
March 30, 2004

Due Date: April 6

1. (10 points) Suppose that you have an application in which you want to use B-trees. Suppose that the computer you will be using has disk blocks holding 4096 bytes, each key is 4 bytes long, each child pointer (which is a disk block id) is 4 bytes, each data record reference (which is a disk block id along with an offset within the block) is 8 bytes. Finally, each node holds an integer (which takes 4 bytes) which holds the number of keys.

You have an application in which you want to store 10,000,000,000 items in your B-tree. What value would you select for t ? (Show how you derived it.) What is the maximum number of disk pages that will be brought into main memory during a search? Remember that the root is kept in main memory at all times.

2. (30 points) Consider the following B-tree. Each of the three problems below will use this B-tree as its starting point. If you have any uncertainties here, I strongly recommend that you look at the practice problems provided on the course web page.



- (a) (8 pts) Show the B-tree that results from inserting F, G, and then H, (in that order) into the above B-tree where the minimum branching factor $t = 2$. Circle (or in some way clearly mark) the B-tree obtained after each split AND each insertion is completed.
- (b) (12 pts) Starting with the ORIGINAL B-tree shown above, show the result from deleting J, M, R and then P. Show enough work that we can follow what you did and circle the B-tree that results after each deletion is completed.
- (c) (10 pts) Show one of the two legal ways to represent the ORIGINAL B-tree as a red-black tree. You can indicate the color of each node by circling it with red or black or just by putting a “r” or “b” next to it. Now using the red-black tree you gave, show the result that will occur when W is inserted. Show enough work that we can follow what you did.
3. (40 points) For the following problem you are to pick a data structure that is best suited for the problem (i.e. the required operation will run as efficiently as possible). The following components should be clearly given in each of your solutions. Your grade will also depend on you selecting the most efficient data structure possible.
- You should very clearly describe your data structure choice, including all decisions about how the data structure is to be applied (e.g. what is used as the key, what is the associated data, ...)

- You should clearly describe how each of the provided operations will be implemented AND analyze the efficiency for each of the operations Do NOT give an code or even pseudo-code. Assume the person reading your solution is familiar with all of the material we have covered. You only need to describe any new methods or variations of standard methods that you need.
- Briefly discuss your choice of data structures. In particular why did you choice the data structure that you choose? Convince us that it was a good choice.

Suppose you have a multimedia document with a set of video segments that are scheduled to play. Each segment s consists of a beginning time b , an ending time e , and a reference to a video object v . You should assume the the video segment is large and must be stored on disk. (For simplicity you can assume that each video object is placed on a single disk page). You must support the below operations.

- Given a new segment $s = (b, e, v)$ determine if s overlaps any segment currently scheduled to play.
- Insert a new segment $s = (b, e, v)$ into the schedule if it does not overlap any currently scheduled segments. (If it does overlap any segment then just report that it cannot be inserted).
- Remove the segment with the earliest beginning time, returning a reference to the associated video segment.

Finally, you should assume that no more than 1000 video segments will be held by your data structure at any given time.

Challenge Problem: (5 points)

A common implementation of sequential files on disk has each block point to its successor, which may be any block on the disk. This method requires a constant amount of time to write a block, to read the first block in the file, and to read the i th block , once you have read the $i - 1$ st block. Reading the i th block from scratch, requires time proportional to i . Show how by adding just one additional pointer per node you can keep all the other properties, but allow the i th block to be read in time $O(\log i)$.