

CMPT-454 Spring 2009
Instructor: Martin Ester
TA: Bahareh Bina

Solution Midterm

Total marks: 150 (15 % of the class)
Date: February 23, 2009

Problem 1 (60 marks)

Consider a disk with a sector size of 1KB, block size of 4KB, 100 sectors per track, 10,000 tracks per surface, and 5 double-sided platters. Suppose that the average seek time is 10 msec, the average rotational delay is 5 msec, and the transfer rate is 4MB per second. Suppose that a file containing 1,000,000 records of 200 bytes each is to be stored on such a disk and that no record is allowed to span two blocks.

a) How many blocks are required to store the entire file?

20 records per block, i.e. we need $1'000'000 / 20 = 50'000$ blocks

b) What is the transfer time per block? Round this time to the nearest msec. What is the transfer time per cylinder?

Transfer time per block is $4KB / (4MB / sec) = 1/1024$ sec \sim 1 msec

Transfer time per cylinder is 250 msec

c) Assume that the data is arranged optimally, i.e. we first fill cylinder 1, then cylinder 2, etc. Suppose also that every seek operation takes the average seek time (independent from the actual number of tracks between the current track and the destination track) and that the head assembly is initially on a random track and a random sector. What is the time required to read the whole file sequentially?

Cylinder has $25 * 10$ blocks, i.e. we need $50'000 / 250 = 200$ cylinders to store the whole file.

The time to read the whole file is therefore

average rotational delay + 200 (average seek time + transfer time per cylinder) =
 5 msec + $200 (10$ msec + 250 msec) = $52'005$ msec \sim 52 sec

Note: the solution omitting the term for the rotational delay (similar to the sample solution of one of our assignments) is also acceptable.

d) What is the time required to read the file in some random order? Assume that each block request incurs the average seek time and average rotational delay.

The time to read the whole file is

$$50'000 \text{ (average seek time + average rotational delay + transfer time per block)} = \\ 50'000 (10 \text{ msec} + 5 \text{ msec} + 1 \text{ msec}) = 800'000 \text{ msec} = 800 \text{ sec}$$

Problem 2 (30 marks)

Consider the following two storage schemes: (1) mirrored disks, (2) RAID level 4.

Compare these two storage schemes in terms of the following criteria:

a) cost for storage device (number of disks), assuming that we want to store data that fills B disks.

With mirrored disks, we need $2B$ disks compared to only $B+1$ disks with RAID level 4.

b) time for equality search (if there is no disk failure).

The time is the same, since in both cases we read only from one disk.

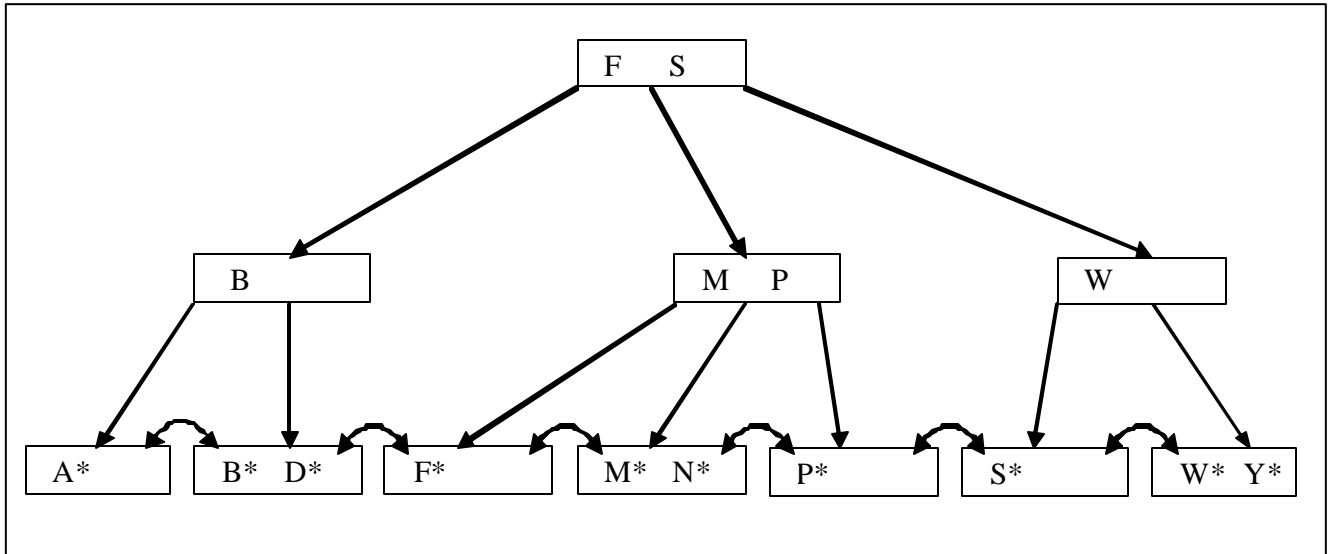
c) mean time to data loss.

Mean time to data loss is shorter for RAID level 4, roughly by a factor of $\frac{1}{2}N(N-1)$

where N is the number of all disks.

Problem 3 (30 marks)

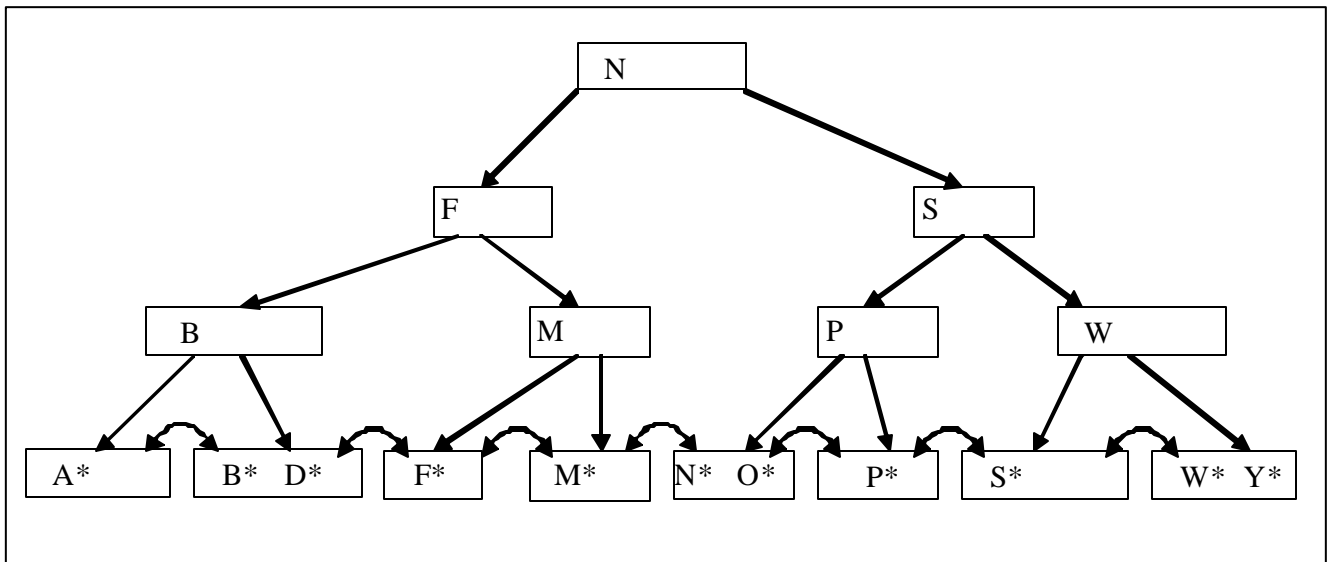
Consider the following B-tree of order $n = 2$, i.e. each node of the B-tree contains one or two search key values. We denote index entries by A, B , etc. and data entries by A^*, B^* , etc.



a) Suppose that there are no duplicate key values. Give a search key whose insertion would lead to an increase of the height of the B-tree.

O

b) Draw the resulting B-tree after inserting the search key given under a).



Note: this is the solution according to the lecture slides. The slightly different solution according to the textbook (where O instead of N is copied and pushed up) is also acceptable.

Problem 4 (30 marks)

Suppose an Extensible Hash table with a capacity of c entries per block.

- a) Describe the situation in which an insertion leads to an overflow that cannot be handled by splitting the corresponding bucket.

Suppose that the local depth of the overflowing bucket is j . The specified situation occurs if all the $c + 1$ entries that fall into the bucket (including the newly inserted entry) have the same value in the $j+1$ -th bit of their binary hash key.

Note: the $c + 1$ hash keys do not need to be duplicates, e.g.

00000, 00001, 00010

 00011

with $c = 3, j = 2$, leads to an overflow chain.

- b) Assume that (1) for any given position of the binary hash keys both 0 and 1 have the same probability and (2) the values of a hash key at any two different positions are independent from each other. Given that an insertion leads to an overflow of the corresponding bucket, what is the probability that this overflow cannot be handled by splitting the corresponding bucket?

We consider the event of $c + 1$ hash keys that either all have a 0 in their $j+1$ -th bit or all have a 1 in their $j+1$ -th bit. The probability of this event is $0.5^{c+1} + 0.5^{c+1} = 0.5^c$.