Jonathan Nguy
603 799 761
DISC 1A

CS 143 Homework 4

1. (a) Capacity of this disk

$$10{,}000\ cylinders * \frac{6\ tracks}{1\ cylinder} * \frac{500\ sectors}{1\ track} * \frac{1024\ B}{sector} = 30.720\ GB$$

(b) Access time = seek time + rotational delay + transfer time

$$seek\ time: \frac{2}{100}s = 20ms\ (\sim 10ms)$$

$$rot\ delay: \frac{60sec/min}{6000\ rot/min} = \frac{\frac{1}{100}sec}{rot} = .01\frac{sec}{rot} = \frac{10ms}{rot} \sim 5ms\ rotational\ delay$$

$$transfer\ time: \frac{10ms}{rot} * \frac{1}{500} = \frac{10}{500} = \frac{1}{50}ms = .02ms\ transfer\ time$$

$$average\ time = 10ms + 5ms + 0.02ms = 15.02ms$$

(c) Char: 1 bytes, Integer: 4 bytes, so for 1 tuple:

$$2*1 + 5*4 + 30*1 + 20*1 = 72\ bytes \Longrightarrow\ 72*1000 = 72{,}000\ bytes => \frac{72000}{1024}$$

$$= 70.31\ sectors, so\ about\ 71\ sectors.$$

(d) Since we're reading 71 sectors (+ seek + rotation time), we can use the following:

$$10ms + 5ms +\ .02*71ms = 16.42ms$$

(e) Because we have to read 3 sectors each time, we can do the following:

$$(10ms + 5ms) * 24 + 0.02 * 71ms = 361.42ms$$

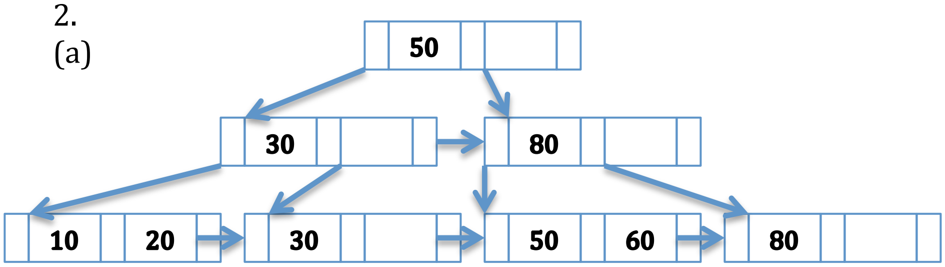
(f) Assuming that memory access is negligible, we don't have to look through all the blocks to find the ones greater than 2005.

$(10ms + 5ms) * n$, where n is the number of tuples that have year >2005.

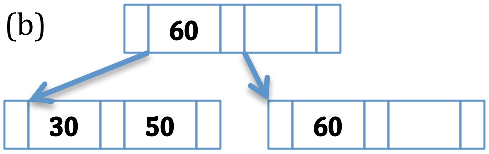
It's slightly helpful to have the B+ tree because you do not have to search the whole disk to compare the years. You'll already have the locations of the tuples you need on the disk, so this will make it slightly faster.

2.

(a)



(b)

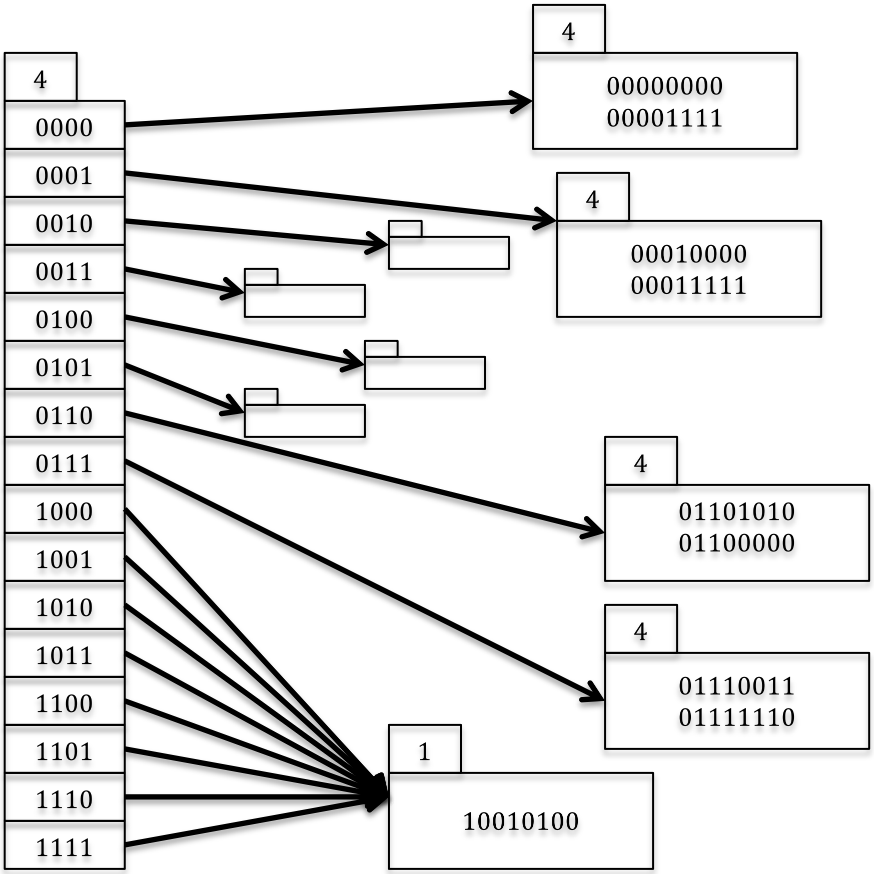


3. For the max height, we want the least number of keys inside a node. Because the minimum keys of a non-leaf with 5 pointers is ceiling(n/2) – 1, that means each node has to have at least 2 records. Also, because there's at most 5 pointers, we know the min pointers of a non leaf per node is ceil(n/2), which is 3. For leaf nodes, it's ceiling((n+1)/2) for min ptrs (3) and ceiling((n-1)/2) for min keys. With 300 records, the very bottom layer would have 150 nodes (300/2). From this, we /3 each time, getting => 150/3 = 50 => 17 => 6 => 2 => 1. This means the height is 6.

For the minimum, we want the most inside each node. This means 4 keys in a node, with 5 pointers. Starting at 350 => 350/4 = 88 => 88/5 = 18 => 4 => 1. So this is a height of 4.

4.

Key:	Key % 256	Binary
106	106	01101010
115	115	01110011
916	148	10010100
0	0	00000000
96	96	01100000
126	126	01111110
16	16	00010000
15	15	00001111
31	31	00011111


4
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
4
00000000
00001111
4
00010000
00011111
4
01101010
01100000
4
01110011
01111110
1
10010100