1. Consider a disk with the following parameters; block size B = 512 bytes, interblock gap size G = 128bytes (some space between blocks used to record some control info. No data is stored there), number of blocks per track = 20; number of tracks per surface = 400. A disk pack consists of 15 double sided platters.
2. What is the total capacity of a track and what is its useful capacity (that is excluding interblock gaps)?

Total: (512 + 128) \* 20 = 640 \* 20 = 12800 bytes

Useful: 512 \* 20 = 10240 bytes

1. How many cylinders are there?

400

1. What are the total capacity and the useful capacity of a cylinder? a disk pack?

Cylinder:

Total: 12800 \* 2 \* 15 = 384000 = 384K

Useful: 10240 \* 2 \* 15 = 307,200 = 307 K

Disk space:

Total: 400\* Cylinder = 400 \* 38400 = 153.6M

Useful: 400\* Cylinder = 122,880,000 = 122.88 M

1. Suppose that the disk drive rotates the disk pack at a speed of 2400rpm (rev per min); what are the transfer rate *tr* in bytes/msec and the block transfer time *btt* in msec? What is the average rotational delay *rd* in msec? What is the bulk transfer rate?

tr = ((512+128)\*20)\* 2400 / (60\*1000) = 512 bytes/msec

btt = 512 / 512 = 1 msec

rd = 60\*1000 / (2400\*2) = 12.5 msec

bulk transfer rate = (B/(B+G))\*tr = (512/640)\*512 = 409.6 bytes/msec

1. Suppose that an average seek time is 30msec. How much time does it take on the average in msec to locate and transfer a single block, given its block address?

(s + rd+btt) = 30 + 12.5 + 1 = 43.5 msec

1. Calculate the average time it would take to transfer 20 random blocks and compare this with the time it would take to transfer 20 consecutive blocks.

Consecutive = s + rd + 20\*btt = 30 + 12.5 + 20 \* 1 = 62.5 msec

Random = 20\*(s+rd+btt) = 20 \* (30+12.5+1) = 20 \* 43.5 = 870 msec

1. Consider a disk with the same characteristics as in Qu1. Further, consider a STUDENTS file with r = 20,000 records, fixed length format with an unspanned organization (blocks cannot span two blocks). Each record has the following fields: : Ssn, 9 bytes; Name, 30 bytes; First\_name, 20 bytes; Address, 40 bytes; Phone 9 bytes; birthdate, 8 bytes; Sex, 1byte; Major‐dept\_code, 4 bytes ; Minor\_dept\_code , 4 byte; class\_code, 4bytes and Degree\_prog, 3bytes. An additional 1 byte was used as a deletion marker (these are used to marked records as deleted for organizations that do not compact at the same time as deletion occurs).
2. The record size *R* (including the deletion marker), the blocking factor *bfr* (number of records in a block), and the number of disks blocks *b* for the file.

R = 9+30+20+40+9+8+1+4+4+4+3+1 = 133 bytes.

bfr = 512 / 133 = 3.85 3 records/block

b = 20000 / 3 = 6666.7 6667 blocks/file

1. Calculate the average time it takes to find a record by doing a linear search on file
2. if the file blocks are stored contiguously;

Cylinder: s + 2\*15\*rd + 20\*2\*15\*btt = 30 + 30\*12.5 + 600 = 1005 msec

Search half of the records: 3333 / 600 = 5 cylinders

Total: 5 \* 1005 + (s+(333 / 20)\* rd + 333\*btt) = 5025 + (30 + 17\*12.5 + 333)

= 5025 + 575.5 = 5600.5 msec

1. if the file blocks are not stored contiguously

3333 \* (s+rd+btt) = 144985.5 msec = 145 sec

3.

(a) 6 pages. There are 6 empty slots available.

(b)

|  |  |  |  |
| --- | --- | --- | --- |
| 64 |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | 25 | 5 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 10 |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 31 | 15 | 7 | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| 11 |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 44 |  |  |  |

(c) After we insert 11, first bucket splits. Then we are in (b). Then we insert 1 and 41, so second bucket needs an overflow page and it splits. After splitting, the second bucket will contain 9,25,1,41, so we can insert 57 to make the third bucket split. Then all the available buckets contain only one item, so we need 4 more pages to make the fourth one split. The total is 1+2+1+4 = 8

(d) Next becomes 0 again. Assume that the last 4 insertions happen in the first bucket, the fourth bucket will split and only 3 and 11 will remain in the fourth bucket.

4.

1. 6 pages. There are 6 empty slots available.

|  |  |  |  |
| --- | --- | --- | --- |
| 64 | 44 |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | 25 | 5 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 10 |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 11 |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 31 | 15 | 7 |  |

1. After inserting a page 11 in fourth bucket, it splits. To split the other 3 buckets, we need 6+3 = 9 pages. The total is 9 + 1 = 10.
2. There is no Next in Extendible hashing index. After insertions, the number of pages in fourth bucket is 2.

5. Suppose that a page can contain at most four data values and that all data values are integers. Using only B+ trees of order 2, give examples of each of the following:

( a ) A B+ tree whose height changes from 2 to 3 when the value 25 is inserted. Show your structure before and after the insertion.

Before inserting 25:

After inserting 25:

( b ) A B+ tree in which the deletion of the value 25 leads to a redistribution. Show your structure before and after the deletion.

Before deleting 25:

After deleting 25:

( c ) A B+ tree in which the deletion of the value 25 causes a merge of two nodes but without altering the height of the tree.

Before deleting 25:

After deleting 25:

6. Consider a relation R(*a,b,c,d,e*) containing 5,000,000 records, where each data page of the relation holds 10 records. R is organized as a sorted file with secondary indexes. Assume that *R.a* is a candidate key for R, with values lying in the range 0 to 4,999,999, and that R is stored in *R.a* order. For each of the following relational algebra queries, state which of the following three approaches is most likely to be the cheapest:

• Access the sorted file for R directly.

• Use a (clustered) B+ tree index on attribute *R.a*.

• Use a linear hashed index on attribute *R.a*.

1. *σa<*50*,*000(*R*) Use a (clustered) B+ tree index on attribute *R.a*.

2. *σa*=50*,*000(*R*) Use a linear hashed index on attribute *R.a*.

3. *σa>*50*,*000􀗨 *a<*50*,*010(*R*) Use a (clustered) B+ tree index on attribute *R.a*.

4. *σa*≠50*,*000(*R*) Access the sorted file for R directly.