ADB Sheet 1 Sol.

#: Setup

$$T_{\substack{access \\ 1\ block}} = S + rd + btt$$

- Seek time (to Position head on track)
- Main delay Culprit
- Often given as avg.

- Rotational delay (to reach block on track)
- $rd = 0.5\ T_{Full}$ on average

- block transfer time (to read the block)
- $btt = \frac{B}{tr}$ ← block size
- tr is transfer rate $= \frac{Tr}{T_{Full}}$ ← track size

For K blocks:

- different Places → $K(S + rd + btt)$
- Same Cylinder → $S + K(rd + btt)$
- Consecutive blocks → $S + rd + K(btt)$

//logic

- Interblock gap
  - → gap after each block (has Control info written during initialization that helps identify blocks)
  - → To more accurately Calculate T when reading consecutive blocks use bulk transfer rate (btr) rather than tr
  - $btr = tr.\left(\frac{B}{B+G}\right)$ ← gap size
  - $btt = \frac{B}{btr}$

- double buffering
  - → usually need to assume it for reading consecutive blocks
    - → Means that we can transfer next block while CPU is processing previous (have two buffers)
      - if that wasn't the case then need to wait for CPU before transfering next block

13.23)

- Given disk where

  B = 512 bytes
  G = 128 byte
  track = 20 blocks
  Surface = 400 tracks
  disk Pack = 15 double-sided disks (30 surfaces)

a) total and useful capacity of track

$\text{Track}|_{\text{tot.}} = (512 + 128) \times 20 = 12.8 \text{ KB}$

$\text{Track}|_{\text{useful}} = 512 \times 20 = 10.24 \text{ KB}$

b) No. of Cylinders = No. of tracks
= 400

c) total and useful capacity of Cylinder

$\text{Cylind}|_{\text{tot.}} = \text{track}|_{\text{tot}} \times \# \text{Surfaces}$
$= 12.8 \times 30 \text{ Kb} = 384 \text{ KB}$

$\text{Cylind}|_{\text{useful}} = 10.24 \times 30 \text{ Kb} = 307.2 \text{ KB}$

d) total & useful Capacity of disk Pack

= # Cylinders

$$\text{disk}|_{tot} = \text{Cylind}|_{tot.} \times \#\,\text{tracks}$$
$$= 384\,\text{Kb} \times 400 = 153.6\ \text{MB}$$

$$\text{disk}|_{useful} = \underset{(307.2\ \text{Kb})}{\text{Cylind}|_{useful}} \times \#\,\text{tracks} = 122.88\,\text{MB}$$

•Note: $\frac{\text{useful}}{\text{Tot}} = \frac{B}{B+G}$

e) . disk rotates the disk Pack @ 2400 rpm

$$T_{full} = \frac{1}{2400} \times (60 \times 1000) = 25\ \text{ms}$$

. find tr, btt, rd, btr

$$tr = \frac{\text{Track Size}}{T_{full}} = \frac{\overset{\text{total}}{12.8\ \text{Kb}}}{25\ \text{ms}} = 512\ \text{byte/ms}$$

$$btt = \frac{B}{tr} = \frac{512}{512} = 1\ \text{ms}$$

$$rd = T_{full}/2 = 12.5\ \text{ms}$$

$$btr = tr * \left(\frac{B}{B+G}\right) = 512 * 0.8 = 409.6\ \text{byte/ms}$$

f) given S = 30 ms

. T_{access} to locate and transfer a single block

$$= S + rd + btt = 30 + 12.5 + 1 = 43.5\,\text{ms}$$

9) Find T_{access} for transfer of

20 random blocks	20 consecutive blocks (with double buff.)
• $20 \times (s + rd + btt)$ $= 870$ ms	• $s + rd + 20 \times btt$ $= 30 + 12.5 + 20 \cdot \frac{512}{409.6}$ $= 67.5$ ms • a less accurate answer would use $btt = 1$

Setup

- r is the no. of records
- R is the record size
- blocking factor (bfr) is the no. of records per block $= \lfloor \frac{B}{R} \rfloor$ assuming unspanned org
  i.e., no record can span two blocks.
- b is the no. of blocks for the file $= \lceil \frac{r}{bfr} \rceil$
  (spanned or unspanned)
- linear search accesses $\frac{b}{2}$ blocks on average (stored contiguously or not depends on if file is sorted)
  → this assumes record will be found & its key attrib. 6
- binary search accesses $\lceil \log_2 b \rceil$ blocks and requires a sorted file.

13.24)

- File has $r = 20000$ records of fixed length
- Fields & their sizes are given (112 bytes for a record)
- an additional delete marker is used
- Stored on disk from previous problem

a) record Size:

$$R = 112 + 1 = 113 \text{ byte}$$

b) bfr and b assuming an unspanned org.

$$bfr = \left\lfloor \frac{B}{R} \right\rfloor = \lfloor 512/113 \rfloor = 4 \text{ records/block}$$

$$b = \left\lceil \frac{r}{bfr} \right\rceil = \lceil 20000/4 \rceil = 5000 \text{ block}$$

c) Access time for linear search if

- blocks are sorted consec. (+ double buffer)
- blocks scattered over disk

. In either case, need $\frac{5000}{2}$ block accesses

- $\underset{30}{s} + \underset{12.5}{rd} + 2500 \times \frac{B_{\rightarrow 512}}{btr_{\rightarrow 409.6}}$ $=$ 13.67 Sec
- $2500(\underbrace{s + rd + btt}_{43.5 \text{ ms}})$ $=$ 108.75 Sec

d) . Assume file is ordered by SSN
. Calculate search time for binary search

→ will need to access $\lceil \log_2 5000 \rceil = 13$ blocks

. hence, $13(s + rd + btt) = 0.5655$ Sec

. althought it requires seek time for each, its much less blocks

#:Setup

- Variable-length
  - → Some fields can have variable length
  - assuming they all exist (not optional) then need a seperator after each variable field to know it ended (e.g. $) e.g. int
  - Otherwise, can assign a type Code for each field and store record as a sequence of <type-Code, value> instead of just <value>
  - In this case, we may need a marker to terminate the record
- Spanned Organitation
  - → unspanned wastes $B-(bfr * B)$ bytes each block
  - → Can make use of such space by storing part of the record there and the other part in the next block (generally need a block pointer in this case)

  - If we use unspanned + variable records then $bfr = \frac{B^*}{R_{avg.}}$ represents no. of records stored on avg.
    - $B^* = B - Ptr.\_bytes$
    - → $b = \lceil \frac{r}{bfr} \rceil$ still holds

13.25)

30 Byte	9B	40B	9B	8B	1B	4B	4B	4B	3B	1B
None	SSN	Addr.	Phone	Bdate	Sex	Major dept Code	Minor deptC.	Class Code	Degree	Del
			↓ 80% of the time exists			↓ 85%	↓ 15%		↓ 90%	

(Variable len)

→ each record has a 1 byte type Code for each field occuring and a 1 byte end of record marker.

→ Spanned Org. with 5-byte block Pointer

a) Calculate average record length R:

$$R_{avg} = ((30+1)+(9+1)+(40+1)+0.8\times(9+1)$$
$$+(8+1)+(1+1)+0.85\times(4+1)+0.15(4+1)$$
$$+(4+1)+0.9\times(3+1)\underset{\text{del.}}{+1}\underset{\text{end of record}}{+1})$$

$= 116.6$ bytes

- total file size avg $= \overset{20\times10^3}{Rr} = 2332$ Kb

b) no. of blocks needed for file

$$bfr = \frac{B-5}{R} = \frac{512-5}{116.6} = 4.348 \text{ records/block}$$

$$b = \left\lceil \frac{r}{bfr} \right\rceil = \left\lceil \frac{20\times10^3}{4.348} \right\rceil = 4600 \text{ block}$$

→ Savings of 400 blocks compared to last problem

13.26)

- Disk unit where
- $S = 20$ ms, $rd = 10$ ms, $btt = 1$ ms
- $B = 2400$ byte, $G = 600$ byte

-

SSN	LName	FName	Middle	Bdate	Addr	Phone	SSSN	Dept	Jobc.	d
9Byte	20Byte	20B	1B	10B	35B	12B	9B	4B	4B	1B

- $r = 30\times10^3$ records
- Fixed length, unspanned blocking

a) R, bfr, b

$R = (9+20+20+1+10+35+12+9+4+4+1)$
$= 125$ byte

$bfr = \lfloor \frac{B}{R} \rfloor = \lfloor \frac{2400}{125} \rfloor = 19$ record/block

$b = \lceil \frac{r}{bfr} \rceil = 1579$ blocks

b) wasted space in each disk block (due to unspanned)

waste $= B - (bfr) \times R = 2400 - 19 \times 125$
$= 25$ bytes

c) tr, btr

$tr = \frac{\text{track size}}{T_{full}}$ • We have btt and $btt = \frac{B}{tr}$

$tr = \frac{B}{btt} = \frac{2400}{1} = 2400$ byte/ms

$btr = tr * (\frac{B}{B+G}) = 2400 \times \frac{2400}{2400+600} = 1920$ byte/ms

d) avg. no. of blocks accessed for linear search

$\frac{b}{2} = 789.5$ blocks

• Search on key attr. that exists

$b = 1579$ blocks

• Search on non-key attr. or not found

e) time to search for an arbitrary file if the blocks are sorted consecutively (+double buffering)

$$T = s + rd + \frac{b}{2}\left(\frac{B}{btr}\right)$$ (assuming case 1)

$= 1.017$ sec

f) Scattered all over the disk

$$T = \frac{b}{2}(s + rd + btt)$$ (assuming case 1)

$= 24.475$ sec

g) ordered via a key field find avg. no of block access and avg. time

→ $\lceil \log_2 15797 \rceil$ block accesses

$$T = \underbrace{\lceil \log_2 15797 \rceil}_{11} (s + rd + btt) = 341 \text{ msec}$$

13.27)

- File with Part # as hash key has records with the following part numbers [2369, 3760, 4692, 4871, 5659, 1821, 1074, 7115, 1620, 2428, 3943, 4750, 6975, 4981, 9208]
- file uses 8 buckets (0 → 7)
- each bucket is a disk block that holds 2 records
- use $h(K) = K \bmod 8$ to load the records into the file at the given order

Bucket	Part #	
0	3760	9208
1	2369	
2	1074	
3	5659	7115
4	4692	1620
5	1821	4981
6	4750	
7	4871	3943

Overflow Page

2428
6975

- if another record collided, would need another Page (3 accesses)

$$\#\text{accesses} = \frac{13}{15} \times 1 + \frac{2}{15} \times 2$$

Probability of requesting a record in overflow

2 accesses

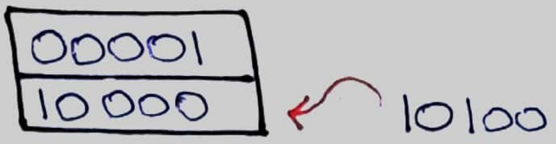
<3 Liad

13.28) Use extendible hashing to load the following

- explained in lec.

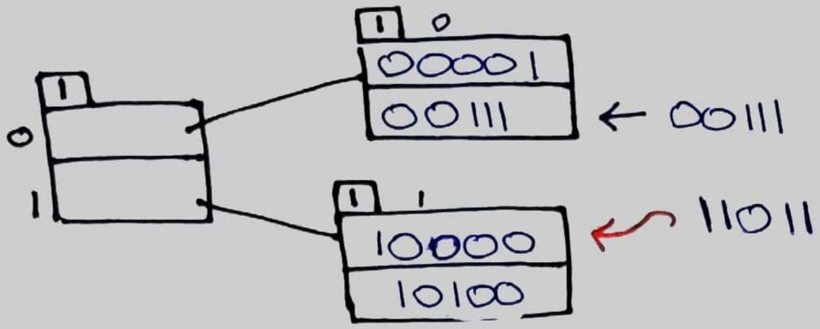
Part #	binary (K mod 32)
2369	00001
3760	10000
4692	10100
4871	00111
5659	11011
1821	11101
1074	10010
7115	01011
1620	10100
2428	11100
3943	00111
4750	01110
6975	11111
4981	10101
9208	11000

1.

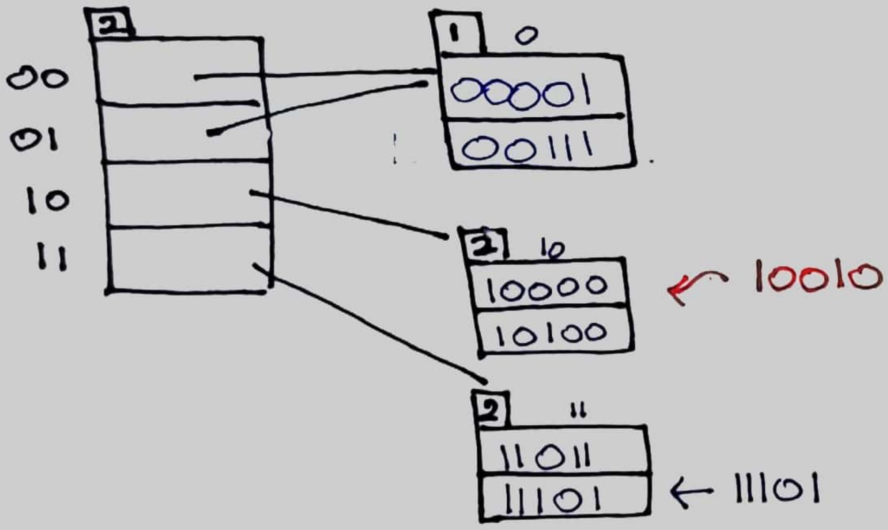


} initialy, Just one block of data.

2.

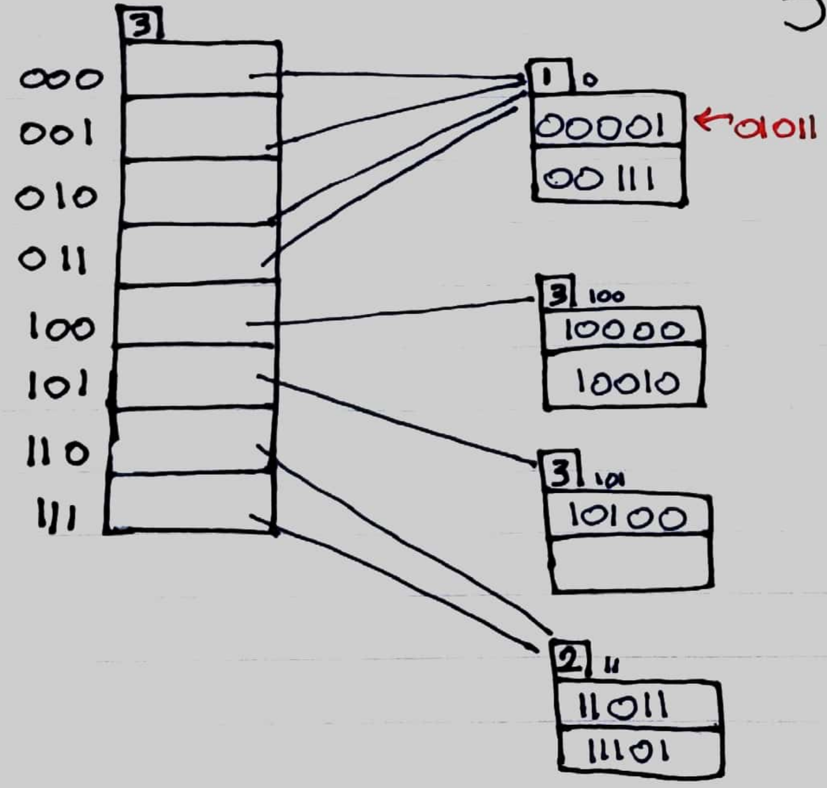


3.

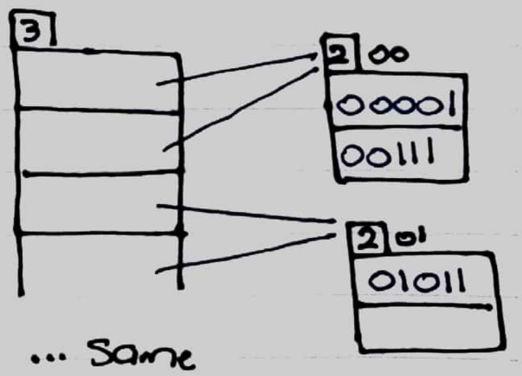


*Notice that any bucket gets $2^{\text{global-local}}$ Pointers

4.



5.



... Same

there are Still 5 more records to go but Still doesn't go beyond the 3 cases

- insert Safely
- double bucket
- double bucket & direc.