I'll write explanation & calculation only for block size = 512 bytes

a) Both R and S are unordered files, and no indexing has been mentioned. So, we go through the entire main file.

(rl) Record length = 296 bytes

(nr) Num rows = 2^{20}

(bf) Blocking factor = $\left\lfloor \frac{\text{block size}}{rl} \right\rfloor = \left\lfloor \frac{512}{296} \right\rfloor$ = 1 / 13 / 110

(nb) Number of blocks = $\left\lceil \frac{2^{20}}{bfr} \right\rceil$ = 2^{20} / 80660 / 9533

Hence number of block accesses will be nb (worst case of course)

(i) Hence ans: 2^{19} / 40330 / 4766.5 (4767)

Average case can be found by dividing this by 2

(ii) Again for KR > val we'll need to access all records.

So worst case block accesses will remain same

i.e. 2^{20} / 80660 / 9533

However, here average case will be slightly different.

(iii) S.A = val

All calculations remain same, but rl = 20 bytes, nr = 2^8

So bf = $\left\lfloor \frac{\text{block size}}{rl} \right\rfloor$ = 1 / 15 / 126

nb = $\left\lceil \frac{2^8}{bf} \right\rceil$ = 256 / 18 / 3

which is equal to number of block accesses

Avg case is also 256 / 18 / 3 since A is not key

iv) S.A > val

Again, following the same logic as in (ii) we get

ans : 256 / 18 / 3

---

b) Assuming files are ordered on key.

(i) S.KS = val

As there's no indexing again, worst case we'll have to go through all records again (but we can use binary search)

~~ans: 256 / 18 / 3~~ ~~1280 / 90 / 4.5~~ (KS is key)

ans : 8 / 5 / 2

(ii) S.KS > val (we can start from last block and stop when, KS <= val)

ans : $\frac{256}{2} / \frac{18}{2} / \frac{3}{2}$ = 128 / 9 / 1.5 (2)

(iii) R.B = val

ans : 2^{20} / 80660 / 9533

(iv) R.B > val

ans : 2^{20} / 80660 / 9533

(c.) Primary index

(i) R. KR = val

re (Record length, main) = 296 bytes

nr (number of records, main) = 2^{20}

rli (Record length, index) = 40 bytes (32 + 8)

As seen in ~~ex~~ (a), ~~number of blocks in main (nb)~~ =

bf (main) = 1 / 13 / 110

So, (nb) number of blocks in main file = 2^{20} / 80660 / 9533

$\left(\left\lceil \frac{nr}{bf} \right\rceil \right)$

Number of blocks in index file = $\left(\left\lceil \frac{nr_i}{bf_i} \right\rceil \right) = \left(\left\lceil \frac{nb}{bf_i} \right\rceil \right)$

bfi (blocking factor, index) = $\left\lfloor \frac{\text{block size}}{rl_i} \right\rfloor$

$= \left\lfloor \frac{512}{40} \right\rfloor = 12$ / 102 / 819

$\longrightarrow nb_i = \left\lceil \frac{2^{20}}{12} \right\rceil = 87382$ / 791 / 12

So, number of block accesses (worst case) = log (nbi) + 1

= 17+1 = 18 / 11 / 5

(ii) Now, we know from (a), nb(main file) = 256 / 18 / 3

rli (record length, index) = 16 + 8 = 24 bytes

bfi (blocking factor, index) = $\left\lfloor \frac{\text{block size}}{rli} \right\rfloor = \left\lfloor \frac{512}{24} \right\rfloor$ = 21 / 170 / 1365

→ nbi (number of blocks in index) = $\left\lceil \frac{nb}{bfi} \right\rceil = \left\lceil \frac{256}{21} \right\rceil$ = 13 / 1 / 1

So, number of block accesses = $\lceil \log_2 nbi \rceil + 1$

= 5 / 2 / 2

Note that $\log_2 1 = 0$ but we still need to access the block in the index file and then go to the only block that's stored in the index.

---

d)(i) We are given that we need to follow clustered indexing.
So, number of entries in index file will be equal to number of distinct entries of R.B = 2^{10} (nei)

rli (record length, index) = 28 + 8 = 36 bytes.

bfi (blocking factor, index) = $\left\lfloor \frac{\text{block size}}{rli} \right\rfloor = \left\lfloor \frac{512}{36} \right\rfloor$ = 14 / 113 / 910

nbi (number of blocks, index) = $\left\lceil \frac{nei}{bfi} \right\rceil = \left\lceil \frac{2^{10}}{14} \right\rceil$ = 74 / 10 / 2

Now, to find R.B < val, we can simply start accessing from first block and stop when R.B >= val.

It's easy to ~~easy~~ see, on average this'll take $nb/2$ accesses

$$\text{avg (number of block accesses)} = 2^{20}/2$$
$$= 2^{19} \,/\, 40330 \,/\, 4766.5 (4767)$$

(ii) Clustered on S.A

(ne) Number of distinct entries of S.A = number of entries in index file = 10

$$rl_i = 24 + 8 = 32 \text{ bytes}$$

$$bf_i = \left\lfloor \frac{\text{block size}}{rl_i} \right\rfloor = \left\lfloor \frac{512}{32} \right\rfloor = 16 \,/\, 128 \,/\, 1024$$

$$nb_i = \left\lceil \frac{ne}{bf_i} \right\rceil = 1/1/1$$

Now, we need to find all blocks with S.A = val.

So, we access index file and then iterate on blocks in main file till we find S.A = val

There are nearly 256/10 = 26 records for each value of S.A

So, number of acess = 1 + nb

$$bf(\text{main}) = \left\lfloor \frac{512}{260} \right\rfloor = 1 \,/\, 15 \,/\, 126$$

$$\boxed{\text{number of accesses} = 1 + \left\lceil \frac{26}{bf} \right\rceil = 27 \,/\, 3 \,/\, 2}$$

(e) Secondary index on key (KS), each entry in index will point to each record in the file. index entry will have key and pointer (block / record)

> Assuming data pointer

$rl_i = 16 + 8 = 24$ bytes

$bf_i = \left\lfloor \frac{\text{block size}}{rl_i} \right\rfloor = 21 / 170 / 1365$

$nb_i = \left\lceil \frac{ne}{bf_i} \right\rceil = \left\lceil \frac{2^8}{21} \right\rceil = 13 / 2 / 1$

KS is key, so we can use binary search

number of block accesses $= 1 + \lceil \log_2(nb_i) \rceil$

$= 5 / 2 / 2$ → same logic as before

> Assuming record pointer

$rl_i = 16 + 16 = 32$ bytes

$bf_i = \left\lfloor \frac{\text{block size}}{rl_i} \right\rfloor = 16 / 128 / 1024$

$nb_i = \left\lceil \frac{ne}{bf_i} \right\rceil = \left\lceil \frac{2^8}{16} \right\rceil = 16 / 2 / 1$

Again binary search on key

number of block accesses $= 1 + \lceil \log_2(nb_i) \rceil$

$= 5 / 2 / 2$

(f) Assuming spanned blocks in this part

(i) Secondary index on non key (R.C)

ne (number of entries in index) = no. of distinct values of R.C = 2

~~Sr~~ Rli = 4 + 8 = 12 bytes

Each entry in index ⟶ points to indirection blocks which contain pointer to 2^{19} records (since uniform)

~~Size~~ Total size of entries in indirection blocks = $2 \cdot 2^{19} \cdot$ (record pointer length)

$= 2^{20} \times 16$ bytes

$= 2^{24}$ bytes

⇒ Number of blocks to store indirection blocks $= 1 + \dfrac{2^{24}}{\text{block size}}$

for index file

indirection blocks

Each indirection block will need a block pointer to next block, a linked list like structure.

so, number of blocks $= \dfrac{1 + 2^{24}}{504} = 33290 / 4106 / 514$

8 bytes for block pointer $512 - 8 = 504$

Number of block accesses $= 1(\text{index file}) + \underbrace{2^{19} \cdot 2^{4}}_{\text{record pointer length}} / \overset{B-8}{504} + \left(\dfrac{2^{19}}{2}\right)$ → (num blocks)

need to access each block

$= \cancel{540934 / 4215268 / 524 \ldots}$

$= \boxed{540934 / 42384 / 5024}$

(ii) S.B has 2^8 unique values. So, S.B is candidate key.
But we've been given that we should assume it as non-key.

So, ~~ind~~ ne_i (no. of entries in index) $= 2^8$ each having S.B value & pointer to indirection block

Size of index record $= \overset{S.B}{28} + \overset{\text{block pointer}}{8} = 36$ bytes

Size of index files $= 2^8 \cdot 36 = 9216$ bytes

no of indirection blocks $= 2^8$

$\therefore$ Total blocks to store $= 2^8 + 9216/B$

$= \boxed{274 ? 258 ? 257}$

Now, we use same approach as (d) to find all S.B < val.
i.e. we iterate till S.B >= val. So, on average we search number of block/2 blocks in index file and $2^8/2$ indirection blocks

So, number of block accesses $= 9216/B/2 + 2^8/2$

$= \boxed{137 ? 129 ? .128}$

(iii) This is similar to the previous part.

n_{ei} (no. of entries in sec index) $= 2^8$

r_{li} (size of record of secondary index file) $= 16+8 = 24$ bytes

Total size of sec index file $= 2^8 \times 24$ bytes
$= 6144$ bytes

Since KS is uniform having 2^8 values, each value will have 2^{12} records

So, indirection block for each index entry will have 2^{12} record pointers.

Total size of indirection block $= 2^8 \cdot 2^{12} \cdot 16$ → record pointer length

$= 2^{24}$ bytes

Number of block required to store $= \underbrace{\left\lceil \frac{2^{24}}{\text{block size} - 8} \right\rceil}_{\text{redirection blocks}} + \underbrace{\left\lceil \frac{2^8}{\text{block size}} \right\rceil}_{\text{index file}}$

redirection blocks
(block size -8, same logic as before)

$\left\lceil \frac{2^8 \cdot 24}{512} \right\rceil + \left\lceil \frac{2^{24}}{504} \right\rceil = \boxed{33301 / 4106 / 513}$

Now again KS is key, so we can use binary search on index file and iterate on indirection blocks.

$$\boxed{\therefore \text{No. of block accesses} = \left\lceil \log_2 (2^8 \cdot 24 / 512) \right\rceil + \left\lceil \frac{2^{12} \cdot 2^4}{504} \right\rceil + 2^{12} = 4231 / 4114 / 4100}$$

(i) Here, we can keep the entire relation S in buffer since buffer size is $(2^{10}+2)$ and can store entire S.

So, number of block accesses = $N_R + N_S$

where N_R = Number of blocks in R

— N_S = ———————— S

Ans : $2^{20} + 256$ / $80660 + 18$ / $9533 + 3$

$= 1048832$ / 80678 / 9536

(l) Worst retrieval technique → sequential
indexing → Linked List

In the worst case, it'll take N_B block accesses

where N_B is the number of blocks.

(m) Size of each row of index file = 16 + 16 bytes = 32 bytes

Size of primary index on S.KS = $2^8 \cdot 32 = 8192$ bytes

Now, we made a linked list or a 1 way search tree

size of each node = 32+16 = 48 bytes

$bf = \lfloor \frac{8192}{48} \rfloor = 170$ node

$b = \lceil \frac{256}{170} \rceil = 2$ blocks

Average access blocks = 2+1 = 3 ~~block~~

Scanned with CamScanner

9) B-tree on R. KR — 2/3 full blocks.

Let B = block size (Variable).

size of block ptr = 8 bytes, record ptr = 16 bytes

len of key = 32 bytes.

Fanout in tree = q (let)

$q_m \times 8 + (16+32) \times (q_m - 1) \leq B$

$\Rightarrow 8q_m + 48q_m - 48 \leq B$

$\Rightarrow 56 q_m \leq B + 48 \Rightarrow q_m \leq \frac{B+48}{56}$

q_m : being max q for tree

$q \leq \frac{2}{3} q_m \leq \frac{2}{3} \frac{B+48}{\cancel{56}\ 28} \Rightarrow q = \left\lfloor \frac{B+48}{84} \right\rfloor$

Records $= 2^{20}$. Need atleast these many record pointers.

1) $B = 512$. $q = 6$. $\rightarrow$ 5 record pointers per level.

if there are n levels:

$5 + 6 \times 5 + 6^2 \times 5 + \ldots + 6^{n-1} \times 5 \geq 2^{20}$

$\cancel{5} \cdot \frac{6^n - 1}{\cancel{6-1}} \geq 2^{20} \Rightarrow 6^n \geq 2^{20} + 1 \geq n \geq \log_6 2^{20} + 1$

$\Rightarrow n \geq 7.73$

$\approx n = 8$

block accesses

$= 8 + 1 = 9$.

2) $B = 4096$. $q = 49$, 48 record ptrs.

$48 + 49 \times 48 + 49^2 \times 48 + \ldots + 49^{n-1} \times 48 \geq 2^{20}$

$\Rightarrow \cancel{48} \cdot \frac{49^n - 1}{\cancel{49-1}} \geq 2^{20} \Rightarrow n \geq \log_{49} 2^{20} + 1$

$\Rightarrow n \approx 4$

block accesses $= 4 + 1 = 5$

3) $B = 32768$, $q = 390$. $n \approx \lceil \log_{390} 2^{20} + 1 \rceil = 3$

block accesses $= 3 + 1 = 4$

• Note : These are MAX number of block accesses. — Can vary between 2 & the max number.

h) B^+ tree on R.KS. 2^8 Records
Leaf Nodes point to the indirection blocks which contain
$\rightarrow 2^{20}/2^8 = 2^{12}$ records.
Notation: internal nodes have outflow/fanout of P
block-ptr size = 8 bytes, record-ptr size = 16 bytes.
size of key = 16 bytes.

$P \times 8 + (P-1) \times 16 \leq B \Rightarrow 24P \leq B + 16$

$\Rightarrow P \leq \frac{B+16}{24}$

Incorporating the 2/3 full constraint:

$P \leq \frac{\cancel{2}}{3} \frac{B+16}{\cancel{24}\,12} = \frac{B+16}{36}$

Notation: Leaf Nodes have fanout = q

$\therefore q \times (16+16) + 8 \leq B \Rightarrow 32q \leq B - 8$

$\Rightarrow q \leq \frac{B-8}{32}$

Adding the 2/3 full constraint:

$q \leq \frac{2}{3} \cdot \frac{B-8}{32}$

	B	q	P
1)	512	10	14
2)	4096	85	114
3)	32768	682	910

To Find Number of levels: $\lceil \log_P 2^8/q \rceil + 1$.

B	levels
512	3
4096	2
32768	2

Now, as KS is a foreign key for relation S we cannot consider the B+ tree for all 2^{20} rows of R — with KS as key.
[Need of a multilevel index]

For S.KS: number of block accesses:

(1) 512 = B	4
(2) 4096 = B	3
(3) 32768 = B	3

(j.) indexed on S.KS, we iterate over all R and for each row in R, find rows in S

Buffer → $2^{10}+2$ blocks

Now, size of block pointer = 8 bytes

Size of field (S.KS) = 16 bytes

We can have upto p tree pointer and $8p + 16(p-1) \leq 512$

$\Rightarrow$ max val of p = 22 / 171 / 1366

Now, Pleaf·(record pointer length + field size) + block pointer length ≤ block size

This gives max value of pleaf = 15 / 127 / 1023

Now we know number of records in leaf node of B+ tree.

So, we compute num (level) of the tree.

num_levels = $\log_p(2^8/p_{leaf}) + 1$

= 2 / 2 / 2

B+ tree will have $2^8 = 256$ record pointers

∴ We need $\left\lceil \frac{256}{p_{leaf}} \right\rceil$ leaf blocks & 1 root block

i.e. we need $\left\lceil \frac{256}{15} \right\rceil + 1$ = 19 / 4 / 2 blocks

Nres calculation

Number of rows = 2^{20}

size of each row = 556 bytes (296 + 260)

$bf_1 = \left\lfloor \frac{\text{Block size}}{556} \right\rfloor$ = 0 / 149797 / 18079

use spanned

(1138688 block)

i.e. we need $\left\lceil \frac{256}{15} \right\rceil + 1 = 19 / 4 / 2$ blocks

To do a join, we keep all N_r blocks of S and B+ Tree index in the memory buffer just like in (i).

$\therefore$ Num (disk block accesses) $= N_r + N_s + N_{s\text{-index}} + N_{res}$

$= 2187539 / 230479 / 27617$ accesses