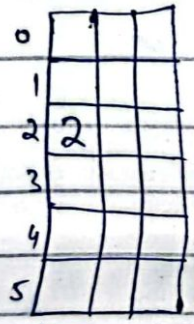
Question 2

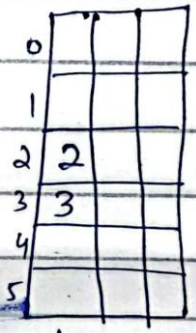
Customes ID values:
(2, 3, 5, 7, 11, 17, 19, 23, 29, 31)

(a) hash function = h(k) = k mod 6

insert 2 2 mod 6 = 2.

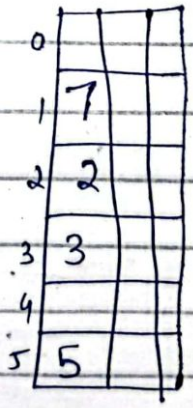


insert 3, 3 mod 6 = 3

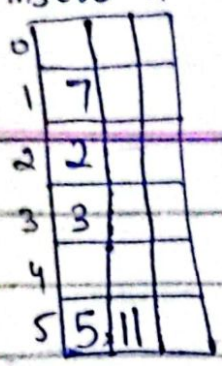


insert 5, 5 mod 6 = 5

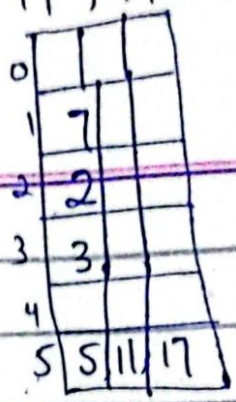
insert 7, 7 mod 6 = 1.



insert 11 11 mod 6 = 5

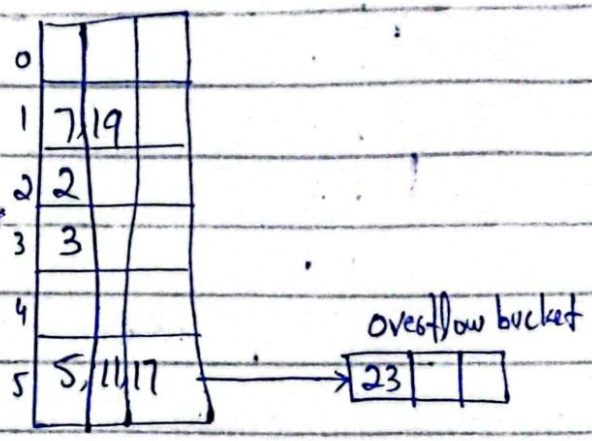


insert 17, 17 mod 6 = 5



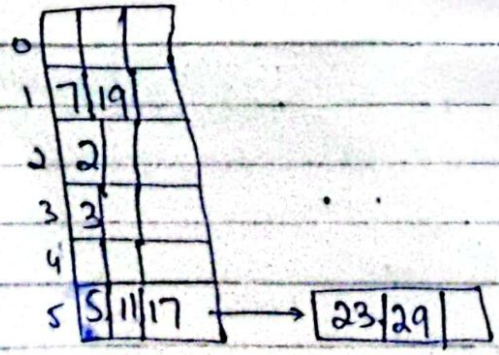
insert 19, 19 mod 6 = 1

insert 23, 23 mod 6 = 5



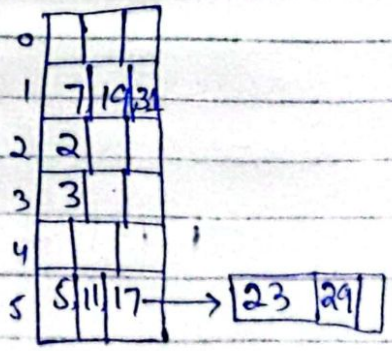
insert 29

29 mod 6 = 5

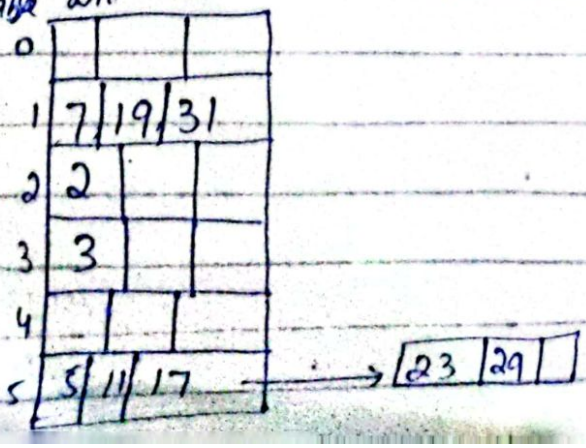


insert 31

31 mod 6 = 1



at the end table will be



(b)

as to access key from normal bucket we need 1 I/O and to access key(custome ID) from overflow bucket we need 2 I/O so

$$\text{average no. of block access} = \left(\text{no. of I/O} \times \frac{\text{ID's in normal bucket}}{\text{total ID's}}\right) + \left(\text{no. of I/O} \times \frac{\text{ID's in overflow bucket}}{\text{total ID's}}\right)$$

$$= \left(1 \times \frac{8}{10}\right) + \left(2 \times \frac{2}{10}\right)$$

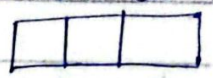
$$= \frac{8}{10} + \frac{4}{10}$$

$$= \frac{12}{10}$$

$$= \boxed{1.2}$$

(C) Expandable Hashing $h(k) = K \bmod 7$

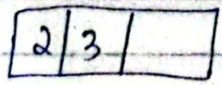
Initially, the first 3 records will be inserted as it because our bucket can hold max 3 records and we also do not need directory initially.



$d = d' = 0$

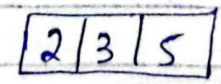
insert 2, $2 \bmod 7 = 2 = 010$

insert 3, $3 \bmod 7 = 3 = 011$



insert 5

$5 \bmod 7 = 5 = 101$



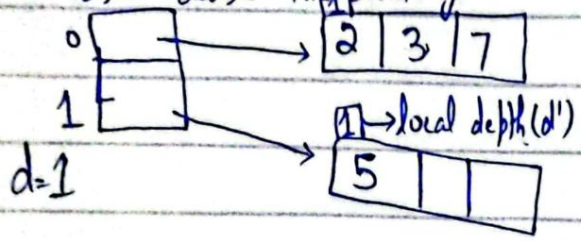
insert 7

7 mod 7 = 0 = 000

overflow, as d'=d

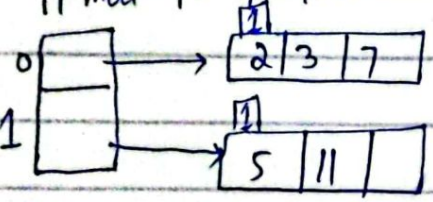
so split the bucket

also double the directory.



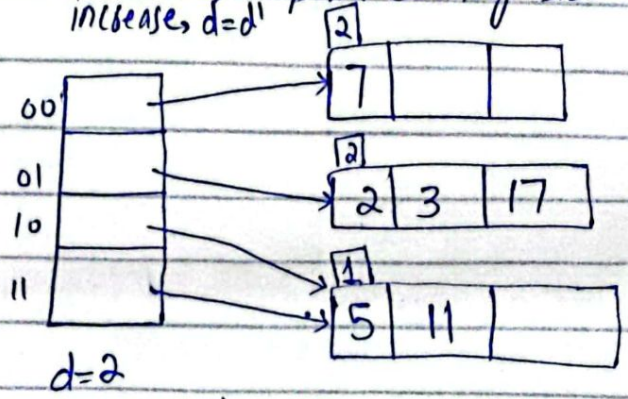
insert 11

11 mod 7 = 4 = 100

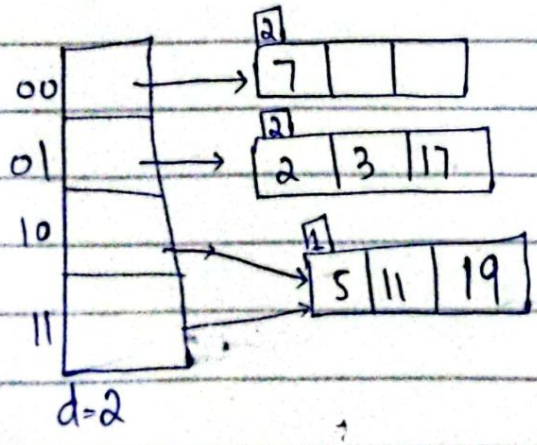


insert 17, 17 mod 7 = 3 = 011

overflow, split + directory size increase, d=d'

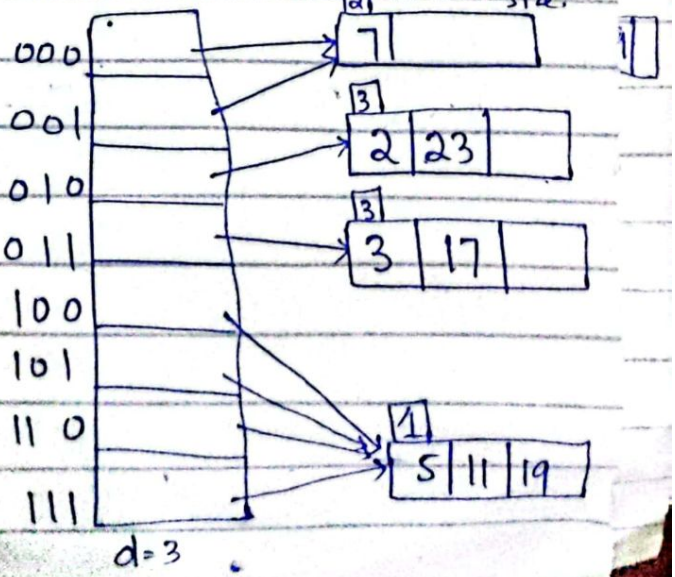


insert 19, 19 mod 7 = 5 = 101



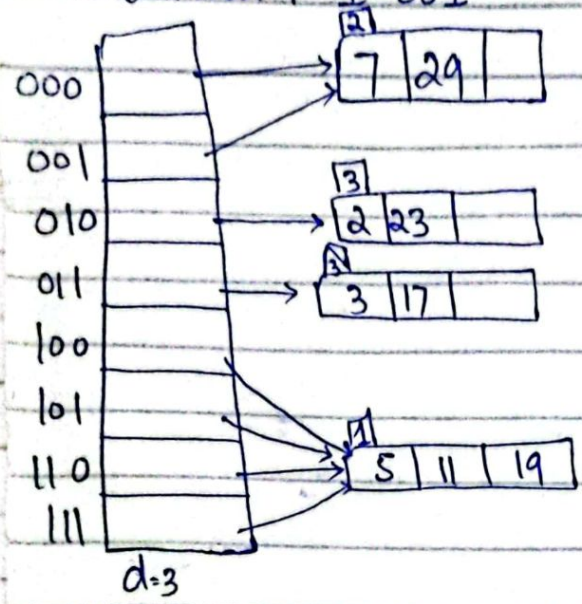
insert 23, 23 mod 7 = 2 = 010

overflow, as d'=d so we will split the bucket and also increase directory size.

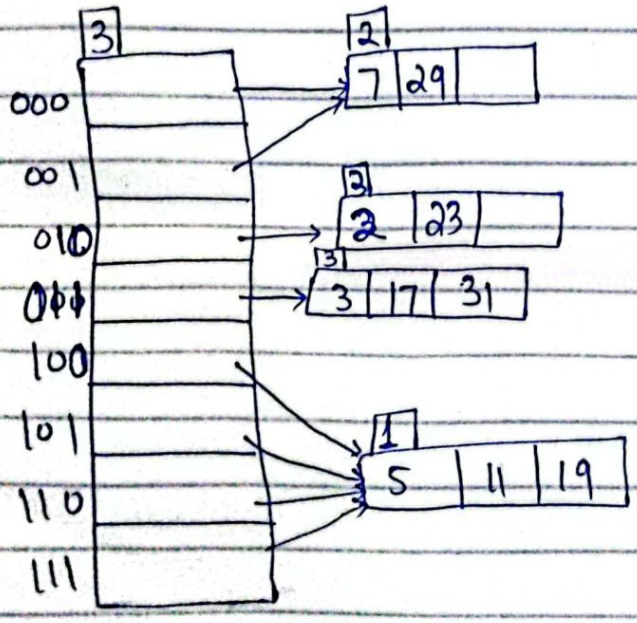


insert 29

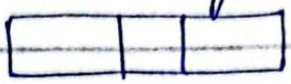
29 mod 7 = 1 = 001



insert 31 , 31 mod 7 = 3 = 011

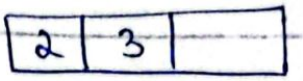


(d) As we have to show each step of dynamic hashing, so for that i will take help from part (c) as I have already done working there.

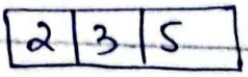
initially 

insert 2

insert 3

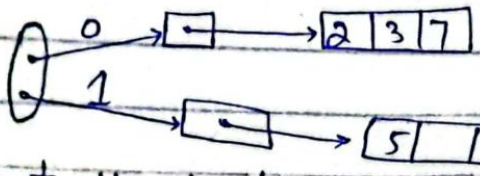


insert 5

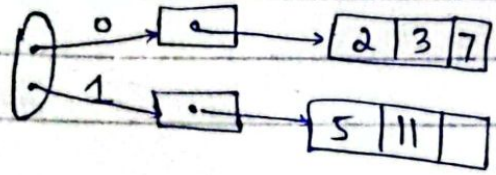


insert 7, 7 mod 7 = 0 = 000

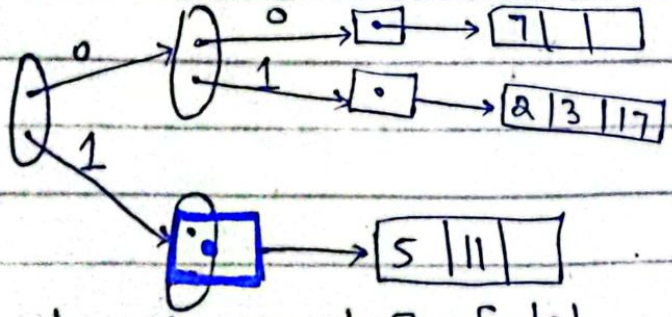
overflow



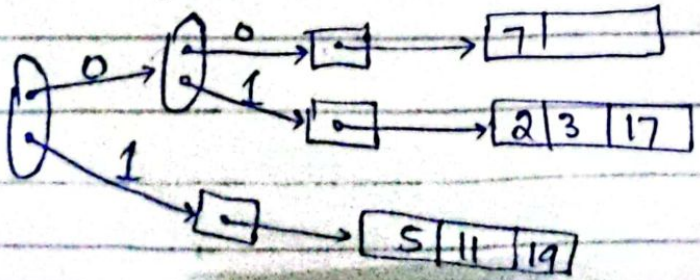
insert 11, 11 mod 7 = 4 = 100



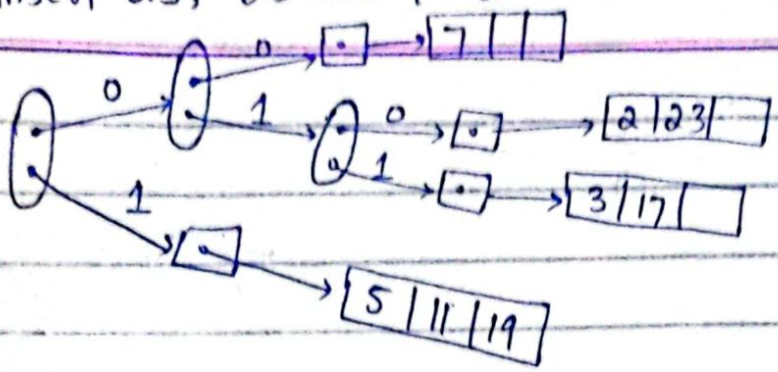
insert 17, 17 mod 7 = 3 = 011



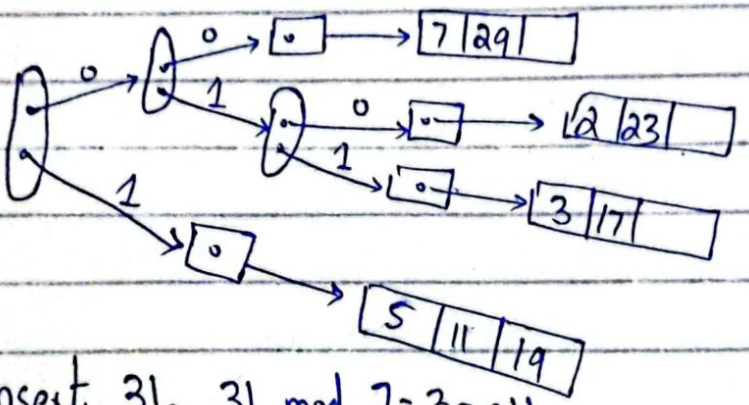
insert 19, 19 mod 7 = 5 = 101



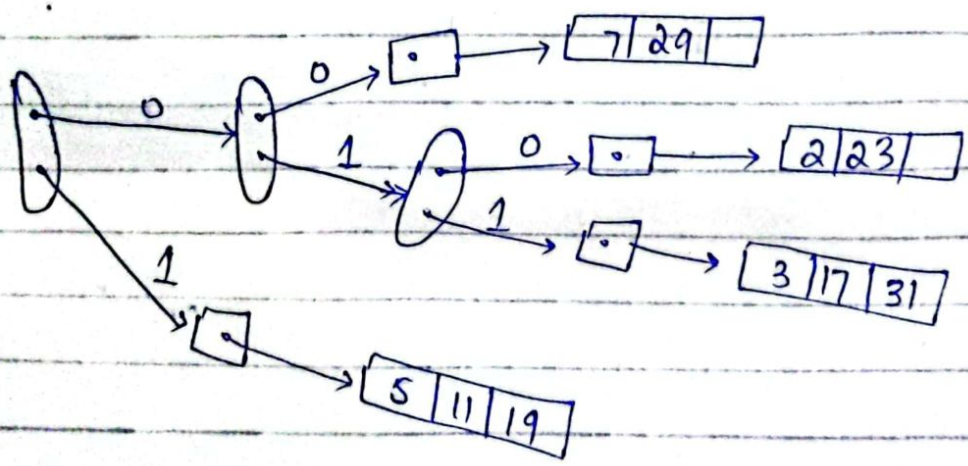
insert 23, 23 mod 7 = 2 = 010



insert 29, 29 mod 7 = 1 = 001



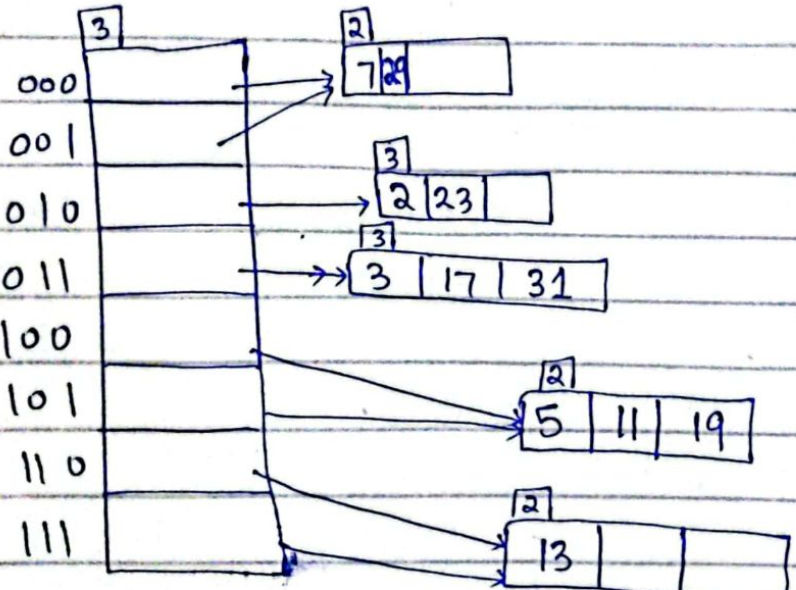
insert 31, 31 mod 7 = 3 = 011



(7) insert 13 into final structure of part c.

13 mod 7 = 6 = 110

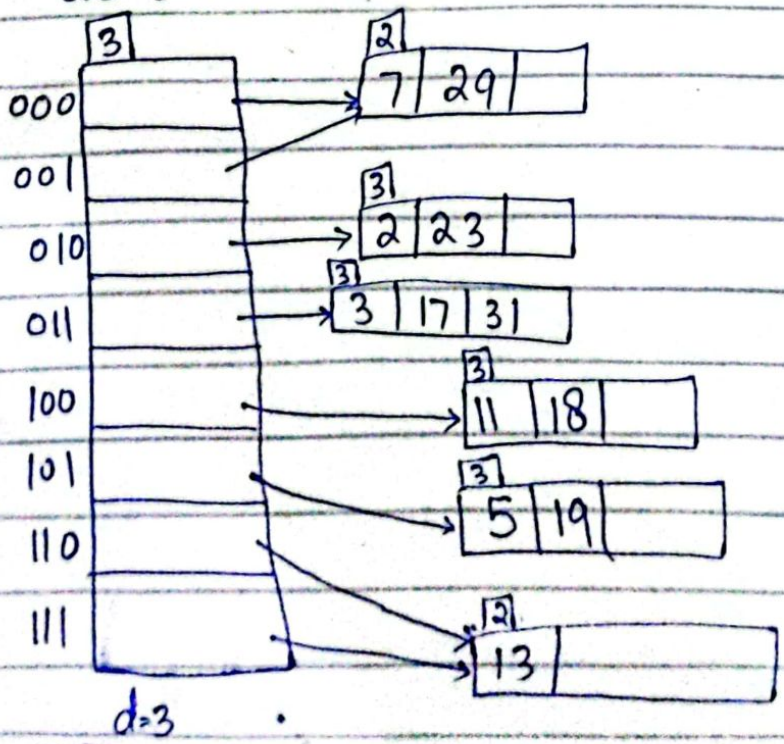
(overflow) as d'<d so we will only split the bucket.



(8) insert 18 into final structure of part (7)

18 mod 7 = 4 = 100

(overflow) again d'<d we will only split the bucket.



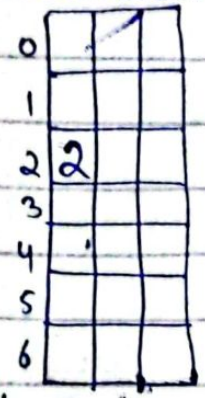
(e) Linear Hashing $h(k) = k \bmod 7$

new hash function = $k \bmod 2M$ M=7 (initially)

initially, our table will be empty and we will have 7 buckets from 0 to 6.

n=0

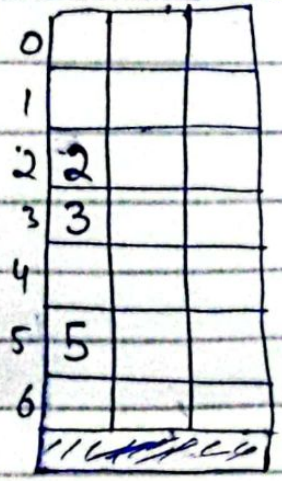
insert 2, $2 \bmod 7 = 2$



insert 3, $3 \bmod 7 = 3$

insert 5, $5 \bmod 7 = 5$

n=0



insert 7, $7 \bmod 7 = 0$

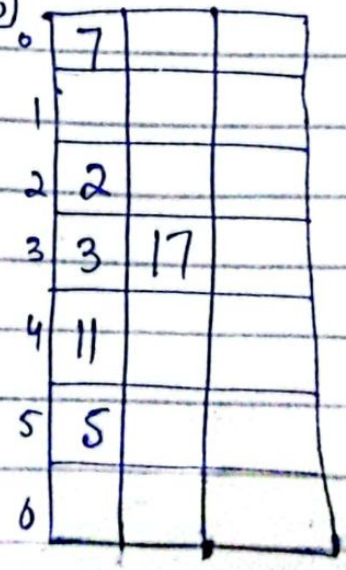
n=0

insert 11, $11 \bmod 7 = 4$

n=0

insert 17, $17 \bmod 7 = 3$

n=0



insert 19

19 mod 7 = 5

n=0

0	7		
1			
2	2		
3	3	17	
4	11		
5	5	19	
6			

insert 23

23 mod 7 = 2

n=0

0	7		
1			
2	2	23	
3	3	17	
4	11		
5	5	19	
6			

insert 29, 29 mod 7 = 1

n=0

0	7		
1	29		
2	2	23	
3	3	17	
4	11		
5	5	19	
6			

insert 31, 31 mod 7 = 3

n=0

0	7		
1	29		
2	2	23	
3	3	17	31
4	11		
5	5	19	
6			

so at the end our table will be

n=0

0	7		
1	29		
2	2	23	
3	3	17	31
4	11		
5	5	19	
6			

Question 3

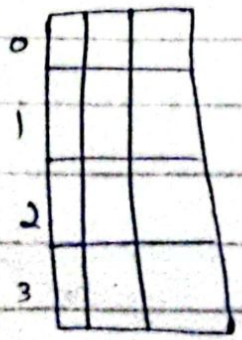
first i will write the binary into decimal

a[010000] = 16
b[011010] = 26
c[111100] = 60
d[001110] = 14
e[010111] = 23
f[011010] = 26
g[101001] = 41
h[010111] = 23
i[000110] = 6
j[101001] = 41

hash function = k mod M

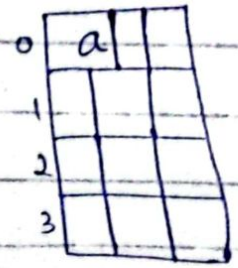
initially, we will take 4 buckets, so M=4
our orignal hash function will be h(k) = k Mod M
and new hash function will be h(k) = k Mod 2M
and n=0 initially.

n=0

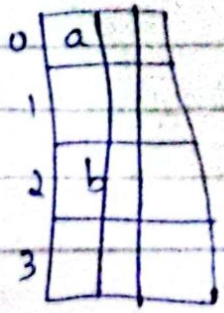


insert a[010000] = 16
16 mod 4 = 0

n=0

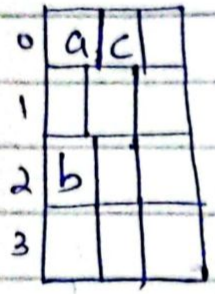


insert b[011010] = 26 mod 4 = 2



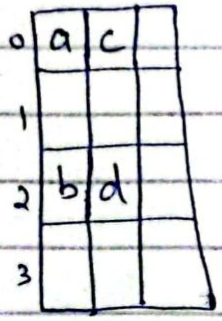
insert c [111100] = 60 mod 4 = 0

n=0



insert d [001110]
= 14 mod 4 = 2

n=0

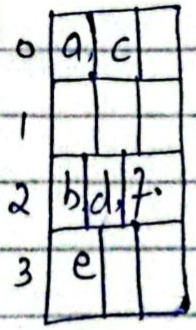


insert e [010111]
= 23 mod 4 = 3

n=0

insert 7 [011010]
= 26 mod 4 = 2

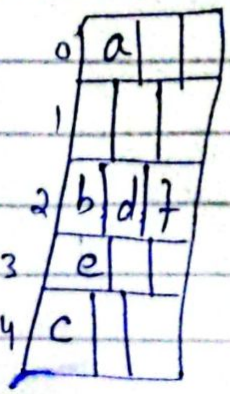
n=0



as threshold was of 2 records so as the third record comes into a bucket we will add a new bucket and redistribute the bucket with n value of index.

adding an extra bucket and redistributing to bucket using new hash function.

n=1



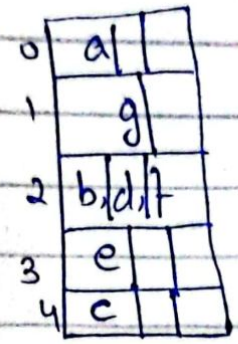
as 0th bucket contain a and c so

a = 16, 16 mod 8 = 0

c = 60, 60 mod 8 = 4

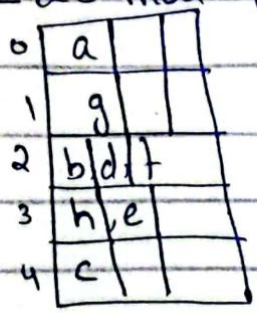
insert g[10|001] = 41 mod 4 = 1

n=1



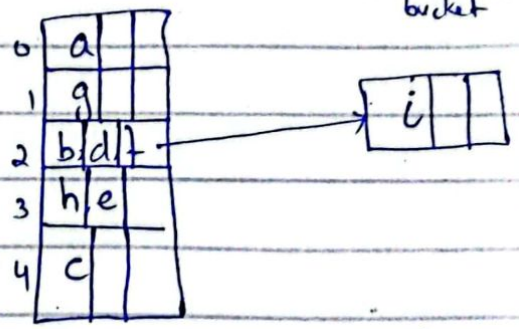
insert h[0|0111]
= 23 mod 4 = 3

n=1



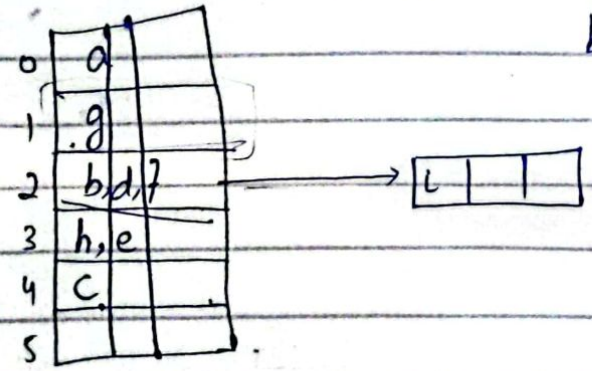
insert i[000|10]
= 6 mod 4 = 2 we need to add an overflow bucket

n=1



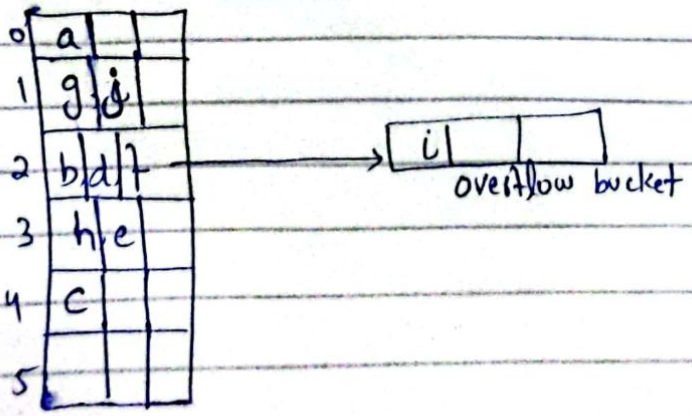
now we need to add an extra bucket and redistribute the bucket 1 using new hash function.
bucket 1 contain:
g[10|001] = 41 mod 8 = 1

n=2



insert j[10|001] = 41 mod 4 = 1

n=2



5

73

Name: Hamza Majeed

Roll No: 20L-2074

Assignment # 03 (ADB-6A)

Q#1

B = 4096 r = 1 omillion

r = 10,000000 = 10^7

block pointer $P_= 6$ bytes

record pointer $P_R = 7$ bytes

R = 30+9+9+40+9+8+1+4+4+1 = 115

$bfr = floor\left(\frac{B}{R}\right) = floor\left(\frac{4096}{115}\right) = 35$

$b = ceiling\left(\frac{r}{bfr}\right) = \left(\frac{10^7}{35}\right)ceiling = ceiling(285714.3)$

b = 285715

(a) Primary index on SSN

R_i = SSn Size + block pointer size

$R_i = 9+6 = 15$

$bfr_i = floor\left(\frac{B}{R_i}\right) = \left\lfloor\left(\frac{4096}{15}\right)\right\rfloor = floor(273.06) = 273$

first level records $= r_1 = 285715$

$b_1 = \text{ceiling}\left(\frac{r_1}{bfr_i}\right) = \left\lceil \frac{285715}{273} \right\rceil = 1047$

$r_2 = 1047$

$b_2 = \text{ceiling}\left(\frac{r_2}{bfr_i}\right) = \text{ceiling}\left(\frac{1047}{273}\right) = 4$

$r_3 = 4$

$b_3 = \text{ceiling}\left(\frac{r_3}{bfr_i}\right) = \text{ceiling}\left(\frac{4}{273}\right) = 1$

So we will need 3 levels in case of multilevel indexing.

total no. of blocks required $= b_1 + b_2 + b_3$

$= 1047 + 4 + 1 = 1052$ blocks required

no. of block access needed will be $= x + 1$

$= 3 + 1 = 4$

(b) Secondary Index on SSN

$R_i = 15$ already calculated in a part.

$bfr_i = 273$

first level records $= r_1 = 10^7$

$b_1 = \text{ceiling}\left(\frac{r_1}{bfr_i}\right) = \left\lceil \frac{10^7}{273} \right\rceil = 36631$

$r_2 = 36631$, $b_2 = \left\lceil \frac{36631}{273} \right\rceil = 135$

have already solved the part (a) in detail so I will not write general formulas again and again I will solve the problem by putting values.

$r_2 = 135$

$b_2 = \left\lceil \frac{135}{273} \right\rceil = 1$

So $x = 3$

we will need 3 levels.

No of blocks $b_i = b_1 + b_2 + b_3 = 36631 + 135 + 1 = 36767$

no. of blocks in secondary indexing is much more than primary indexing.

No of block access $= x + 1 = 3 + 1 = 4$

(C) Secondary Index on Department Code

R_i = Dept Code size + P_b = 9 + 6 = 15

$bfr_i = \text{floor}\left(\frac{B}{R_i}\right) = \text{floor}\left(\frac{4096}{15}\right) = 273$

total distinct values of dept. code = 20000

no. of employee records against each dept. code = $\frac{10^7}{20000} = 500$

# of bytes needed for level of indirection block = 7 * 500 = 3500 so all 500 record pointers can fit in 1 block. So there will be 20,000 blocks in level of indirection.

no. of records at first level = $r_1 = 20,000$

$b_1 = \text{ceiling}\left(\frac{r_1}{bfr_i}\right) = \left\lceil \frac{20,000}{273} \right\rceil = 74$

$b_2 = 74$

$b_2 = \lceil \frac{74}{273} \rceil = 1$

So in case of multilevel indexing there will be 2 levels.

$x = 2$.

total no of blocks = $b_1 + b_2 +$ no. of blocks in level of indirection

$= 74 + 1 + 20,000 = 20075$

block access = $x + 1 +$ average record for each value

$= x + 1 + 500 = 2 + 1 + 500 = 503$

(d) Clustering Index on Dept. Code

$R_i = 9 + 6 = 15$

$bfr_i = \text{floor}\left(\frac{B}{R_i}\right) = \lfloor \frac{4096}{15} \rfloor = 273$

no. of first level records = $r_1 =$ 20000

$b_1 = \lceil \frac{20000}{273} \rceil = 74$

$b_2 = 74$

$b_2 = \lceil \frac{74}{273} \rceil = 1$

So we need $x = 2$ levels in case of multilevel indexing.

total no. of blocks required = $b_1 + b_2 = 74 + 1 = 75$

total no. of block access = $x + \frac{S}{bfr}$

$= 2 + \lceil \frac{500}{35} \rceil = 2 + 15 = 17$ block access.

$bfr = \lfloor \frac{4096}{115} \rfloor$

$bfr = 35$

$p*P+(p-1)(V_{SSN/SR})<B$

(e) B^+ tree

for non-leaf nodes

$(p*P)+(p-1)(V_{SSN})<B$

$(p*6)+(p-1)(9)<4096$

$6p+9p-9<4096$

$15p<4096+9$

$p<\frac{4105}{15}$

$p<273.6$

$\boxed{p=273}$

$P_{leaf}*(V_{SSN}+P_R)+P<B$

for leaf nodes

$p_{leaf}*(V_{SSN}+P_R)+P<B$

$p_{leaf}*16+6<4096$

$p_{leaf}<\frac{4096-6}{16}$

$p_{leaf}<255.6$

$p_{leaf}=255$

as blocks are approximately 69% full so

$0.69\times255=175.9 \approx 176$ approximately

$b_1=\text{ceiling}\left(\frac{10^7}{176}\right)=56819$

for non leaf nodes.

$0.69\times273=188.3\approx189$

$b_2 = \text{ceiling}\left(\frac{b_1}{f_o}\right) = \left\lceil \frac{56819}{189} \right\rceil = 301$

$b_3 = \left\lceil \frac{301}{189} \right\rceil = 2$

$b_4 = \left\lceil \frac{2}{189} \right\rceil = 1$

total number of blocks = $b_1 + b_2 + b_3 + b_4 = 56819 + 301 + 2 + 1$
$= 57123$

block access = $x + 1 = 4 + 1 = 5$

(7) B-tree

$(p * P) + (p-1) * (V_{SSN} + P_R) < B$

$p * 6 \quad (p-1)(16) < 4096$

$p < \frac{4096+16}{22}$

$p < 186.9$

$p = 186$

as block is 69% full so $0.69 \times 186 = 128.3$
≈ 129

so no of blocks at $b_1 = \left\lceil \frac{10^7}{129} \right\rceil = 77520$

$b_2 = \left\lceil \frac{77520}{129} \right\rceil = 601$

$b_3 = \left\lceil \frac{601}{129} \right\rceil = 5$

$b_4 = \left\lceil \frac{5}{129} \right\rceil = 1$

total blocks = $b_1 + b_2 + b_3 + b_4 = 77520 + 601 + 5 + 1 = 78127$

total block access = $x + 1 = 5$

In case of B-tree no. of blocks (total) is much more than B+ tree total no. of blocks although block access cost remains same.

Q No: 2

$B = 8192 \quad r = 10 \text{ billion}$

$r = 10 \text{ billion} = 10 \times 1000 \text{ million}$

$r = 10000 \text{ million} \quad r = 10000,000000$

$r = 10^{10}$

$R = 115$

$bfr = floor\left(\frac{B}{R}\right) = \left\lfloor \frac{8192}{115} \right\rfloor = 71$

$b = ceiling\left(\frac{r}{bfr}\right) = \left\lceil \frac{10^{10}}{71} \right\rceil = 140845071$

(a) Primary Index on SSN

$R_i = SSn\,Size + P_b = 9 + 6 = 15$

$bfr_i = floor\left(\frac{B}{R_i}\right) = \left\lfloor \frac{8192}{15} \right\rfloor = 546$

$r_1 = 140845071$

$b_1 = ceiling\left(\frac{r_1}{bfr_i}\right) = \left\lceil \frac{140845071}{546} \right\rceil = 257959$

$r_2 = 257959$

$b_2 = \left\lceil \frac{257959}{546} \right\rceil = 473$

$r_3 = 473$

$b_3 = \left\lceil \frac{473}{546} \right\rceil = 1$

So 3 levels will be needed in case of multilevel indexing.

total no. of blocks required = $b_1+b_2+b_3 = 257959+473+1$
$= 258433$

no. of blocks access needed = $x+1 = 3+1 = 4$

(b) Secondary Index on SSN

$R_i = 15$
$bfr_i = 546$ already computed in part a.

First level records = $r_1 = 10^{10}$

$b_1 = \text{ceiling}\left(\frac{r_1}{bfr_i}\right) = \left\lceil \frac{10^{10}}{546} \right\rceil = 18315019$

$r_2 = 18315019$

$b_2 = \left\lceil \frac{18315019}{546} \right\rceil = 33544$

$r_3 = 33544$

$b_3 = \left\lceil \frac{33544}{546} \right\rceil = 62$

$r_4 = 62$

$b_4 = \left\lceil \frac{62}{546} \right\rceil = 1$

So we will need 4 levels in case of multilevel indexing.

In the total no. of blocks = $b_1+b_2+b_3+b_4 = 18315019+33544+62+1$
$= 18348626$

no. of block access = x+1 = 4+1 = 5

So in case of Secondary index on SSN total no. of blocks are much much more than primary indexing on ssn also block access cost is 5 in Secondary and 4 in primary.

(C) Secondary Index on Dept. Code

R_i = DeptCodesize + P_B = 9+6 = 15

$bfr_i = floor\left(\frac{B}{R_i}\right) = \left\lfloor\frac{8192}{15}\right\rfloor = 546$

total distinct values of dept. code = 20,000

average no. of employee records against each dept. code

$= \frac{10^{10}}{20,000} = 500,000$

# of bytes needed for level of indirection blocks

$= 7 \times 500000$

$= 3500000$ bytes need against 1 dept. code value in level of indirection.

as 1 block size = 8192

so $\left\lceil\frac{3500000}{8192}\right\rceil = \lceil 427.2 \rceil = 428$ blocks needed to hold record pointers against one distinct dept. code

so there will be total $428 \times 20,000 = 8560000$ blocks in level of indirection.

no of records at first level of indexing = r_1 = 20000

$b_1 = ceiling\left(\frac{r_1}{bfr_i}\right) = \left\lceil\frac{20,000}{546}\right\rceil = 37$

$r_2 = 37$

$b_2 = \left\lceil \frac{37}{546} \right\rceil = 1$

So in case of multilevel indexing there will be $x = 2$ levels.

total no of blocks = $b_1 + b_2$ + no. of blocks in level of indirection $= 37 + 1 + 856000$

$= 856038$

block access = $x + 428$ + average record for each value.

$= 2 + 428 + 500{,}000$

$= 500430$

(d) Cluster index on Dept. Code

$R_i = q + b = 15$

$b_{fi} = \text{floor}\left(\frac{B}{R_i}\right) = \left\lfloor \frac{8192}{15} \right\rfloor = 546$

no of first level records = $r_1 = 20{,}000$

$b_1 = \left\lceil \frac{20{,}000}{546} \right\rceil = 37$

$r_2 = 37,$

$b_2 = \left\lceil \frac{37}{546} \right\rceil = 1$

So we need $x = 2$ levels in case of multilevel indexing.

total no. of blocks = $b_1 + b_2 = 37 + 1 = 38$

total no of block access = $x + \frac{S}{b_{fr}}$

$= 2 + \left\lceil \frac{500{,}000}{71} \right\rceil$

$= 2 + 7043 = \boxed{7045}$

$b_{fr} = \left\lfloor \frac{8192}{115} \right\rfloor$

$b_{fr} = 71$

e) B+ tree

for non-leaf nodes

$(p * P) + (p-1)(V_{SSN}) < B$

$(p*6) + (p-1)(9) < 8192$

$15p < 8192 + 9$

$p < \frac{8192+9}{15}$

$p < 546.73$

$p = 546$

for leaf node

$p_{leaf} * (V_{SSN} + P_R) + P < B$

$16 p_{leaf} + 6 < 8192$

$p_{leaf} < \frac{8192-6}{16}$

$p_{leaf} < 511.625$

$p_{leaf} = 511$

as blocks are approximately 69% full so

$0.69 \times 511 = 352.5 \approx 353$

$b_1 = \text{ceiling}\left(\frac{10^{10}}{353}\right) = \lceil 28328611.9 \rceil = 28328612$ (no. of leaf level blocks)

for non-leaf nodes

$0.69 \times 546 = 376.7 \approx 377$

$b_2 = \text{ceiling}\left(\frac{28328612}{377}\right) = 75143$

$b_3 = \text{ceiling}\left(\frac{75143}{377}\right) = 200$

$b_4 = \text{ceiling}\left(\frac{200}{377}\right) = 1$

total no. of blocks $= b_1 + b_2 + b_3 + b_4 = 28328612 + 75143 + 200 + 1$

$= 28403956$

block access $= x + 1 = 4 + 1 = 5$

(f) B-tree

$(p * P) + (p-1)*(V_{SSN} + P_R) < B$

$6p + 16p - 16 < 8192$

$p < \frac{8192+16}{22}$

$p < 373.09$

$p = 373$

as block is 69% full so

$0.69 * 373 = 257.3 \approx 258$

$b_1 = \text{ceiling}\left(\frac{10^7}{258}\right) = 38759690$

$b_2 = \text{ceiling}\left(\frac{38759690}{258}\right) = 150232$

$b_3 = \text{ceiling}\left(\frac{150232}{258}\right) = 583$

$b_4 = \text{ceiling}\left(\frac{583}{258}\right) = 3$

$b_5 = \text{ceiling}\left(\frac{3}{258}\right) = 1$

total blocks $= b_1 + b_2 + b_3 + b_4 + b_5 = 38759690 + 150232 + 583 + 3 + 1$

$= 38910509$

total block access $= x + 1 = 5 + 1 = 6$

In case of B-tree total no. of blocks are way more than B+ tree. Also block access cost is increased by 1 and 1 level is also added.

QNo:3 (a) B^+-tree

For B^+ tree we need to find order for both non-leaf and leaf nodes.

Non-leaf

$$p * P + (p-1)(V_{SSN} + P_R) < B$$

$$56p + (p-1)(8+12) < 1024$$

$$56p + 20p - 20 < 1024$$

$$p < 16.125$$

$$p = 16$$

Leaf

$$p_{leaf} * (V_{SSN} + P_R) + P < B$$

$$p_{leaf} \times (8+12) + 56 < 1024$$

$$20p_{leaf} + 56 < 1024$$

$$p_{leaf} < 48.4$$

$$p_{leaf} = 48$$

We have 3 levels in B^+ tree including root.

Leaf-level having key-value pairs can have 48 index records (with data pointers)

Root can have 16 childs and the 2nd level nodes can also have 16 childs each.

So as in B^+ tree index records are only present in leaf level hence

maximum records that can be indexed $= 16 \times 16 \times 48$

$= 12288.$

(b) B-tree

First we calculate order p.

$$p*P + (p-1)*(V_{SSN} + P_R) < B$$

$$p*56 + (p-1)(8+12) < 1024$$

$$56p + 20p - 20 < 1024$$

$$76p < 1024+20$$

$$p < \frac{1024+20}{76}$$

$$p < 13.7$$

$$p = 13$$

First level can have 12 index records

Second level can have 13x12 index records because first level can have maximum of 13 child and each child can have 13 child further.

Hence each child at second level can have 12 index records

Third level can have 13x13x12 index records.

So

total will be = 12 + (13x12) + (13x13x12)

= 12 + 156 + 2028

= 2196

Before marking this Q check the last sheet too. (last page after Q. 4).

QNo: 4

$x = 4$

(a)

$x + 1$

$= 4 + 1$

$= 5$ as index is on key field

(b)

$x + 1$

$= 4 + 1 = 5$ by 5 block access we will reach the block with A=111 and then in that block we will search for B=3

(c) $2(x+1) = 2(4+1) = 2(5) = 10$

index is on key field and we need two A values.

(d) as there are total 5000 blocks and each block holds 1 value of A so

$(5000 - 100) + (x+1)$

$= 4900 + 4 + 1 = 4905$

(e)

$(5000 - 100) + x$ here we only need count not records so not

$= 4900 + 4 = 4904$

I have done Q3 with sir's method. There is also another method present at some sites (online) (sir told us in class) so I will also do Q3 with that method too. (last papers)

Q:3

(a) B+-tree

let each node of a B+tree contain at most n pointers and n-1 keys so $(n*12)+(n-1)*8+56<1024$

$12n+8n-8+56<1024$

$n<\frac{1024-56+8}{20}$

$n<48.8$

$n=48$

Therefore, the leaf level of a B+ tree can hold at most $48\times48\times47$ record pointers. Therefore the maximum no. of records that can be indexed is 108288

(b) B-tree ; let each inner node of a B-tree contain at most n index pointers, n-1 keys and n-1 record pointers.

$(n-1)8+((2n-1)\times12)+56<1024$

$8n-8+24n-12+56<1024$

$n<\frac{1024-56+12+8}{32}$

$n<30.8$

$n=30$

The first level can hold at most 29 record pointers.

The second level can hold at most 29×30 record pointers, the leaf level can hold at most $29\times30\times30$ record pointers.

therefore the maximum no. of records that can be indexed is $= 29+(29\times30)+(29\times30\times30)$

$= 29+870+26100$

$= 26999$

MIDTERM EXAM ANSWER SHEET

NATIONAL UNIVERSITY
OF COMPUTER & EMERGING SCIENCES
LAHORE

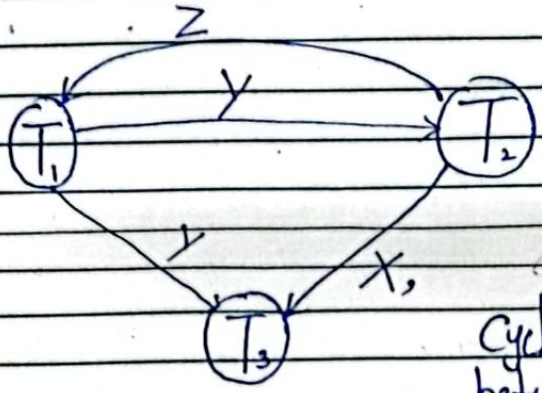
Course ADB Answer Sheet No. 11238

Student's Name Hamza Majeed Signature Hamza 28

Roll No. 202-2074 Section 6A Date 28-Feb-2022.

Q1

Precedence Graph:-



Cycle Detected between T_1 and T_2.

5

The schedule is not conflict serializable as there exist a cycle in the precedence graph.

Q:2

3

S1: $r_1(X)$; $w_3(X)$; $w_2(Y)$; $r_1(Y)$; $r_1(Z)$; $r_2(Y)$; c_2; c_3; c_1

Dirty read exist. (S1 is recoverable because:)
The transaction which write on item Y has committed first and the T who read item Y is committed later. So that's why S1 is "recoverable".

Dirty Read = T1 write ... item ... same item later.

Q / Part No. Rough Work Q / Part No.

S2: $r_1(X); W_2(X); r_1(Y); r_1(Z); r_3(Y); W_2(Y); C_2; C_3; C_1$

as I explained above what is dirty read. So we see S2 these is not dirty read in it so its neither recoverable nor non-recoverable. as of cascadeless and strict, these is not a single write on write operation on same data item by different Transactions, so its also not cascadeless or strict.)

(but these are Some conflict operations $r_1(x), w_2(x)$ on that base $r_3(y), w_2(y)$. we can say its)

Q NO: 3

(a) Master log record

LSN
101

10

(b) During the analysis, the recovery manager check when did the last check point occur, and then after that check point analyze all the transactions in which all the T with their last operation and whether these are committed or active mentioned.
During analysis it will also make a dirty page table in which all the pages with their first update operation are written. The analysis will start from 101 and end on LSN "112".

Dirty Page Table

Page-ID	LSN
X	103
Y	104
Z	106

Transact

≡

During redo

We will minimum page ta

Redo

d

D

active from and active

As

T4.

L

Q / Part No.

Rough Work

Transaction Table

Transaction	LSN	status
T_1	105	committed
T_2	108	committed
T_3	111	committed
T_4	112	in process

c Redo

During the redo process we will redo all the operations.

We will start redo from the minimum LSN of page from dirty page table and redo till end.

Redo

Start LSN	END LSN
103	112

d

During undo, we will undo only active Transactions. we will start from last operation of active Transaction and end at first operation of active Transaction.

As there is only 1 active transaction T_4. So we will only undo T_4.

Undo (T_4)

Start LSN	END LSN
112	109

Q / Part No.

Rough Work

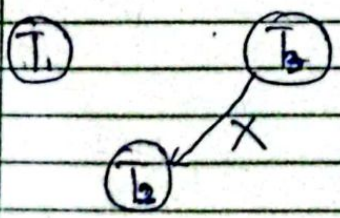
Q / Part No. (b)

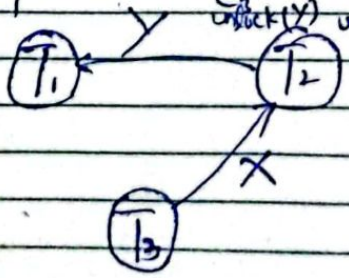
QNo: 4

(a)

Rigorous 2PL (wait for graph)

T_1	T_2	T_3
S1-lock(W)		
$r_1(W)$		
	S2-lock(X)	
	$r_2(X)$	
		X3-lock(X)
		conflict T_3 will wait for T_2 on X.
X1-lock(Y)		
$W_1(Y)$		
	S2-lock(Y)	
	conflict T_2 will wait for T_1 on Y.	
X1-lock(Z)		
$W_1(Z)$		
C_1		
unlock(W)		
unlock(Y)		
unlock(Z)		
	$r_2(Y)$ wake up call	
	S2-lock(Z)	
	$r_2(Z)$	
	C_2	
	unlock(Y)	
	unlock(X)	
	unlock(Z)	
		wake up $W_3(X)$
		S3-lock(Y)
		$r_3(Y)$
		C_3
		unlock(Y) unlock(X).





(b)

Ba

T1 can restart here.

Conflict abort T1

Q / Part No. (b)

Basic TO

T_1	T_2	T_3	W RTS	W WTS	X RTS	X WTS	Y RTS	Y WTS	Z RTS	Z WTS
$r_1(W)$			T_1							
	$r_2(X)$				T_2					
		$w_3(X)$				T_3				
$w_1(Y)$								T_1		
	$r_2(Y)$						T_2			
	$r_2(Z)$								T_2	
		$r_3(X)$			T_3					
	C_2	C_3								
$w_1(Z)$ conflict abort T_1.										

T_1 can restart here.

Name: Hamza Majeed 935

Roll No: 20L-2074

ADB Assignment #1 (6A)

Schedule S1:

$r_1(X), r_2(Y), r_1(Z), r_2(X), w_2(X), r_2(Z), r_3(Y), w_2(Z), w_1(Y), w_3(Y), C_3, r_1(Y), w_2(Y), C_2, C_1$

Q No: 1 Basic 2PL (wait-die Policy)

T_1	T_2	T_3
S_1-lock(X)		
$r_1(X)$		
	S_1-lock(Y)	
	$r_2(Y)$	
S_1-lock(Z)		
$r_1(Z)$		
	$r_2(X)$	
	x_2-lock(X) abort due to T_1	
		S_3-lock(Y)
		$r_3(Y)$
x_1-lock(Y)		
wait for T_3 on Y		
		$w_3(Y)$ lock upgraded to exclusive
		unlock(Y)
		C_3
$w_1(Y)$ wake up		
$r_1(Y)$		
unlock(X)		
unlock(Y)		
unlock(Z)		
C_1		
	T_2 can restart here.	

Q No:2 Strict 2PL (wound-wait Policy)

T_1	T_2	T_3
S_1-lock(X)		
$r_1(X)$		
	S_2-lock(Y)	
	$r_2(Y)$	
S_1-lock(Z)		
$r_1(Z)$		
	$r_2(X)$ read cnt++	
	x_2-lock(X) conflict with T_1 it will wait...	
		~~S_3-lock(Z)~~
		$r_3(Y)$ read cnt++
x_1-lock(Y) conflict as it is older so it will abort T_3, T_2.	aborted	aborted due to T_1
$r_1(Y)$		
unlock(X)		
unlock(Z)	~~$w_2(X)$ wake up~~	
C_1	~~S_2-lock(Z)~~	
unlock(Y)	~~$r_2(Z)$~~	
	~~$u_2(Z)$ lock unlocked~~	
	~~x_2-lock(Y)~~	
	~~$w_2(Y)$~~	
	T_2 can restart here	T_3 can restart here.

QNo:3 Rigorous 2PL (wait-die policy)

T_1	T_2	T_3
S_1-lock (X)		
$r_1(X)$		
	S_2-lock (Y)	
	$r_2(Y)$	
S_1-lock (Z)		
$r_1(Z)$		
	$r_2(X)$ read(cnt+)	
	X_2-lock (X) conflict aborted due to T_1.	
		S_3-lock (Y)
		$r_3(Y)$
X_1-lock (Y) conflit, it will wait for T_3 on "y".		
		$w_3(Y)$ lock upgraded
		C_3
		unlock (Y)
$w_1(Y)$ (wake up)		
$r_1(Y)$		
C_1		
unlock (X)		
unlock (Y)		
unlock (Z)		
	T_2 can restart here.	

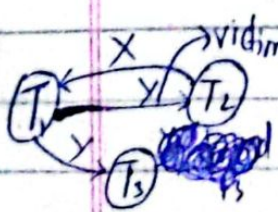
Q No: 4 Rigorous 2PL (wound-wait Policy)

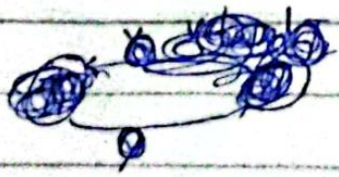
T_1	T_2	T_3
S1-lock(X)		
$\gamma_1(X)$		
	S2-lock(Y)	
	$\gamma_2(Y)$	
S1-lock(Z)		
$\gamma_1(Z)$		
	$\gamma_2(X)$	
	X2-lock(X) conflict - it will wait due to T_1	
		~~S3-lock(Y)~~
		$\gamma_3(Y)$ read cont++
X1-lock(Y) conflict as it is older so it will abort T_3 and T_2. and it will continue...	aborted due to T_1	aborted due to T_1.
$\gamma_1(Y)$		
C_1		
unlock(X)		
unlock(Y)		
unlock(Z)		
	T_2 can restart here	T_3 can restart there.

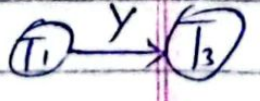
∵ Younger T will be victimized and aborted in case of deadlock.

Q No: 5 Rigorous 2PL (wait-for-graph)

T_1	T_2	T_3
S_1-lock(X)		
$r_1(X)$		
	S_2-lock(Y)	
	$r_2(Y)$	
S_1-lock(Z)		
$r_1(Z)$		
	$r_2(X)$	
	X_2-lock(X)	
	conflict -- T_2 will wait for T_1 on X	
		S_3-(Y)
X_1-lock(Y)		
T_1 will wait for T_2 and T_3 on Y.	aborted T_2. and shared lock of Y will be transferred to T_3.	X_3-lock(Y) lock upgraded. ~~(victimized and restart with same TS)~~
	T_1 is waiting for T_2 on Y and T_2 is waiting for T_1 on X. Victimize T_2 (abort) and restart with same TS	$w_3(Y)$
		C_3
$w_1(Y)$ wake up		unlock(Y).
$r_1(Y)$		
C_1		
unlock(X)		
unlock(Y)		
unlock(Z)		
	T_2 can restart here	







Q No: 6 Basic Timestamp Ordering

QNo:7 Strict TO

T_1	T_2	T_3	X RTS	X WTS	Y RTS	Y WTS	Z RTS	Z WTS
$r_1(X)$			T_1					
	$r_2(Y)$				T_2			
$r_1(Z)$							T_1	
	$r_2(X)$		T_2					
	$W_2(X)$			T_2				
	$r_2(Z)$						T_2	
		$r_3(Y)$			T_3			
	$W_2(Z)$							T_2
$W_1(Y)$ abort T_1 and restart with new TS								
		$W_3(Y)$				T_3		
		C_3						
	$W_2(Y)$ abort T_2 and restart with new TS							

Q No: 8 Thomas's Write Rule

			X		Y		Z	
T_1	T_2	T_3	RTS	WTS	RTS	WTS	RTS	WTS
$r_1(X)$			T_1					
	$r_2(Y)$				T_2			
$r_1(Z)$							T_1	
	$r_2(X)$		T_2					
	$W_2(X)$			T_2				
	$r_2(Z)$						T_2	
		$r_3(Y)$			T_3			
	$W_2(Z)$							T_2
$W_1(Y)$ ignore this operation						T_3		
		$W_3(Y)$						
		C_3						
$r_1(Y)$ ignore								
	$W_2(Y)$ ignore							
	C_2							
C_1								

Q No: 9 Multi-Version TO

T_1	T_2	T_3	X RTS	X WTS	Y RTS	Y WTS	Z RTS	Z WTS
$r_1(X)$			T_1					
	$r_2(Y)$				T_2			
$r_1(Z)$							T_1	
	$r_2(X)$		T_2					
	$w_2(X)$		T_2	T_2				
	$r_2(Z)$						T_2	
		$r_3(Y)$			T_3			
	$w_2(Z)$						T_2	T_2
$w_1(Y)$					T_1	T_1		
		$w_3(Y)$			T_3	$\{T_1, T_3\}$		
		C_3						
$r_1(Y)$					T_1			
	$w_2(Y)$ abort T_2.							
C_1			4					

Schedule S2

$r_2(X), w_1(Y), w_2(Y), w_1(X), w_3(Z), c_1, w_2(X), c_2$
$w_3(X), c_3$

Q No:1 Basic 2PL (wait-die policy)

T_1	T_2	T_3
	S2-lock(X) $r_2(X)$	
X1-lock(Y)		
	X2-lock(Y) about due to T_1 and all locks released.	
X1-lock(X) $w_1(X)$		
		X3-lock(Z) $w_3(Z)$
unlock(Y) Unlock(X) c_1		
		X3-lock(X) $w_3(X)$ unlock(X) Unlock(Z) c_3
	T_2 can restart here.	

Q No:2 Strict 2PL (wound-wait policy)

T_1	T_2	T_3
	S_2-lock(X)	
	$r_2(X)$	
x_1-lock(Y)		
$w_1(Y)$		
	x_2-lock(Y) conflict so T_2 will wait...	
x_1-lock(X) conflict, so it will abort T_2 and continue...		
	aborted due to T_1	x_3-lock(Z)
		$w_3(Z)$
c_1		x_3-lock(X)
unlock(X)		$w_3(X)$
unlock(Y)		c_3
	T_2 can restart here.	unlock(X)
		unlock(Z)

Q No: 3 Rigorous 2PL (Wait-Die Policy)

T_1	T_2	T_3
	S_2-lock(X)	
	$r_2(X)$	
x_1-lock(Y)		
$w_1(Y)$		
	x_2-lock(Y)	
	conflict - aborted due to T_1 all locks released.	
x_1-lock(X)		
$w_1(X)$		x_3-lock(Z)
		$w_3(Z)$
C_1		
unlock(Y)		
unlock(X)		x_3-lock(X)
		$w_3(X)$
		C_3
		unlock(X)
		unlock(Z)
	T_2 can restart here.	

Q No: 4 Rigorous 2PL (wound-wait Policy)

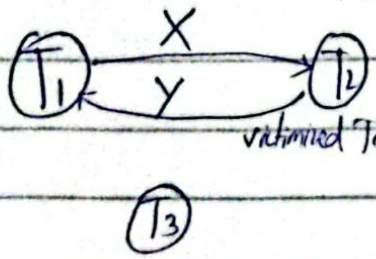
T_1	T_2	T_3
	S_2-lock(X)	
	$r_2(X)$	
X_1-lock(Y)		
$w_1(Y)$		
	X_2-lock(Y)	
	conflict, it will wait due to T_1.	
X_1-lock(X)		
it will abort T_2	aborted due to T_1	
		X_3-lock(Z)
		$w_3(Z)$
C_1		
unlock(X)		X_3-lock(X)
unlock(Y)		$w_3(X)$
		C_3
		unlock(X)
		unlock(Z)
	T_2 can restart here.	

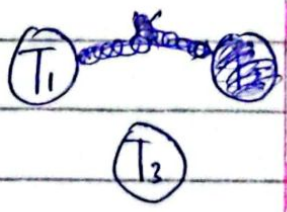
S2

PS:- Younger T2 will be victimized and aborted in case of deadlock.

QNo:5 Rigorous 2PL (wait for graph)

T_1	T_2	T_3
	S_2-lock(X)	
	$r_2(X)$	
x_1-lock(Y)		
$w_1(Y)$		
	x_2-lock(Y)	
	T_2 will wait for T_1 on Y.	
X_1-lock(X) conflict		
T_1 will wait due to T_2 on X		
	victimize T_2 and abort it all locks will be released.	
$W_1(X)$ wake up		
		x_3-lock(Z)
		$w_3(Z)$
C_1		
unlock(X)		
unlock(Y)		
		x_3-lock(X)
		w_3-(X)
		C3
		unlock(Z)
		unlock(X)
	T_2 can restart here.	





QNO:6 Basic TO

			X		Y		Z	
T_1	T_2	T_3	RTS	WTS	RTS	WTS	RTS	WTS
	$R_2(X)$		T_2					
$W_1(Y)$						T_1		
	$W_2(Y)$					T_2		
$W_1(X)$ abort T_1 and restart with new TS.								
		$R_3(Z)$						T_3
	$W_2(X)$			T_2				
	C_2							
		$W_3(X)$		T_3				
		C_3						

QNo: 7 Strict TO

			X		Y		Z	
T_1	T_2	T_3	RTS	WTS	RTS	WTS	RTS	WTS
	$r_2(X)$		T_2					
$W_1(Y)$						T_1		
	$W_2(Y)$ delay until C_1 or a_1							
$W_1(X)$ abort T_1 and restart with new TS.								
	$W_2(Y)$ wake up.					T_2		
		$W_3(Z)$						T_3
	$W_2(X)$			T_2				
	C_2							
		$W_3(X)$		T_3				
		C_3						

Q No: 8 Thomas's Write Rule

			X		Y		Z	
T_1	T_2	T_3	RTS	WTS	RTS	WTS	RTS	WTS
	$R_2(X)$		T_2					
$W_1(Y)$						T_1		
	$W_2(Y)$					T_2		
$W_1(X)$ ignore		$W_3(Z)$						T_3
C_1								
	$W_2(X)$			T_2				
	C_2							
		$W_3(X)$		T_3				
		C_3						

Q No: 9 Multi-Version

T_1	T_2	T_3	X		Y		Z	
			RTS	WTS	RTS	WTS	RTS	WTS
	$R_2(X)$		T_2					
$W_1(Y)$					T_1	T_1		
	$W_2(Y)$				T_2	$\{T_1, T_2\}$		
$W_1(X)$			T_1	T_1				
		$W_3(Z)$					T_3	T_3
C_1								
	$W_2(X)$		T_2	$\{T_1, T_2\}$				
	C_2							
		$W_3(X)$	T_3	$\{T_1, T_2, T_3\}$				
		C_3						

here.

Name : Hamza Majeed

Roll No: 20L-2074 45

Assignment # 02 (ADB - 6A)

Q No: 1

$B = 1024$ bytes
$r = 10,000,000$
$R = 100$ bytes
No. of students per department = 50,000

$$\text{Blocking factor} = bfr = \left\lfloor \frac{B}{R} \right\rfloor = \left\lfloor \frac{1024}{100} \right\rfloor = \lfloor 10.24 \rfloor$$

$bfr = 10$

$$\text{Total no. of data blocks} = b = \left\lceil \frac{r}{bfr} \right\rceil = \left\lceil \frac{10,000000}{10} \right\rceil$$

$b = 1000,000$

(a) As file is unordered so it will take linear time.

no. of block fetches needed = $O(b) =$ **1000,000**

(b) As file is ordered on roll no and we only need to read one record so it will take: $O(\log_2 b)$

no. of block fetches needed = $\log_2 b = \log_2(1000,000) = \lceil 19.93 \rceil = \boxed{20}$

(c) As file is not ordered so it will take linear time. O(b)
no. of block fetches needed = 1000,000

(d) As file is ordered on department no and we need to fetch 1 whole department as there are 50,000 students in one department so selection cardinality = 50,000

no. of block fetches needed = $O(\log_2 b) + \left\lceil \frac{s}{bfr} \right\rceil - 1$

$= \lceil \log_2 (1000000) \rceil + \left\lceil \frac{50,000}{10} \right\rceil - 1$

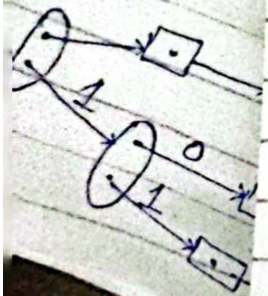
$= 20 + 50,00 - 1$

$= 20 + 4999$

$= 5019$

insert

insert 7, overflow



Name : Hamza Majeed

Roll No: 20L-2074

Quiz #2 (ADB-6A)

Q

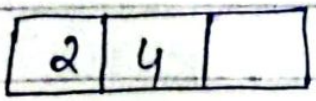
hash function = k mod 8

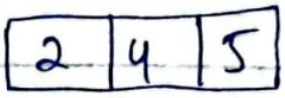
as a bucket can hold 3 records first 3 values will be inserted without any issue.

insert 2

2 mod 8 = 2 = 010

insert 4, 4 mod 8 = 4 = 100 , insert 5 5 mod 8 = 5 = 101



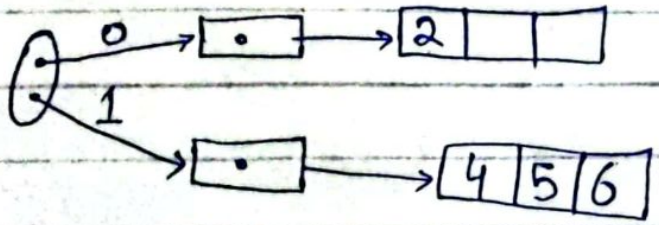


Binary Table. (for help)

0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

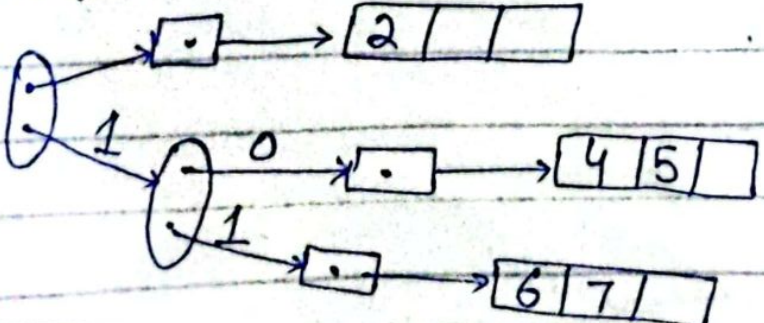
insert 6 6 mod 8 = 6 = 110

overflow (add extra bucket)



insert 7, 7 mod 8 = 7 = 111

overflow



Name :

No: 20L-2

ment # 02 (AD

B = 1024 by

r = 10,000,000

R = 100 bytes

No. of students pe

Blocking factor = bf r =

bf = 10

& no. of data bloc

b = 1000,000

file is unordere

time.

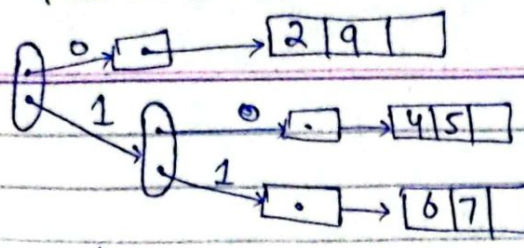
block fetches neede

file is ordered on x

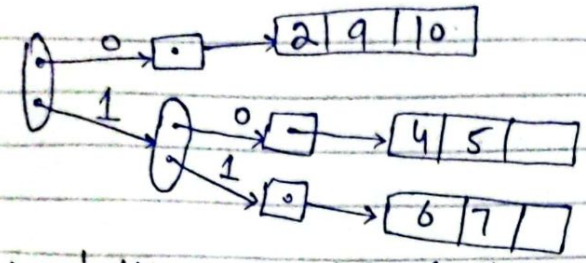
& one record

block fetches needed

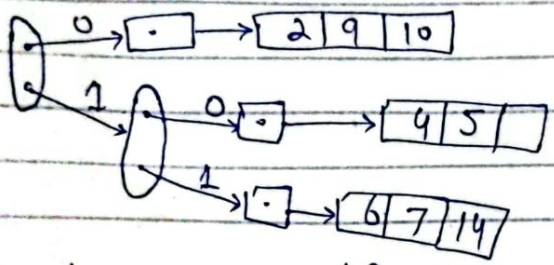
9 mod 8 = 1 = 001



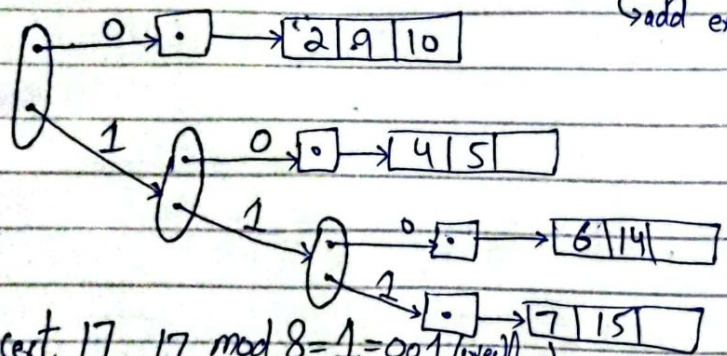
insert 10, 10 mod 8 = 2 = 010



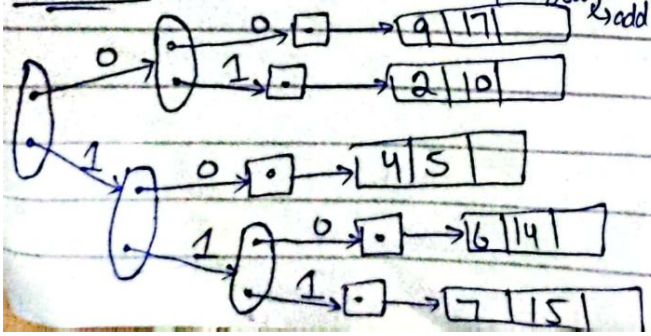
insert 14 14 mod 8 = 6 = 110



insert 15 . 15 mod 8 = 7 = 111 (overflow) → add extra bucket



insert 17 17 mod 8 = 1 = 001 (overflow) → add extra bucket



MIDTERM EXAM ANSWER SHEET

NATIONAL UNIVERSITY
OF COMPUTER & EMERGING SCIENCES
LAHORE

FAST

23+1

Course: Advance Database Concepts Answer Sheet No: 5608

Student's Name: Hamza Majeed Signature:

Roll No: 20L-2074 Section: 6A Date: 10-april-2023

Question No: 1

$r = 100,000$

$R = 10+10+10 = 30$ bytes

$B = 1024$

6

$bfr = floor\left(\frac{B}{R}\right) = floor\left(\frac{1024}{30}\right) = floor(34.13) = 34$

no. of blocks $= b = ceiling\left(\frac{r}{bfr}\right) = \left\lceil\frac{100,000}{34}\right\rceil = 2942$

(a) as file is ordered on C so this query will take linear time.

So we will apply linear search:

no. of block fetches needed $= O(b) = 2942$

(b) as file is ordered on C so we will apply binary search. also we will consider selection cardinality of C in this part.

Selection Cardinality $= s = 5000$

no. of block fetches needed $= \log_2(b) + \left\lceil\frac{s}{bfr}\right\rceil - 1$

$= \log_2(2942) + \left\lceil\frac{5000}{34}\right\rceil - 1$

$= 12 + 148 - 1 = 159$ block fetches.

(c) we need 2 records in this part.
As file is ordered on C so
no. of block fetches needed $= 2\left(\log_2(b) + \left\lceil \frac{s}{bfr} \right\rceil - 1\right)$

we have already computed $\left(\log_2(b) + \left\lceil \frac{s}{bfr} \right\rceil - 1\right)$
in previous part so
no. of block fetches needed $= 2(159)$
$= 318$

Question # 2

- key field size = 30 bytes
- Record pointer = 10 bytes
- Block pointer = P_B = 10 bytes
- B = 1024
- b = 1000,000

As it is a B^+ tree, so first we will calculate order p and order p leaf.

Order p:

$$p * P + (p-1) * (\text{key field size}) < B$$
$$p \times 10 + (p-1)(30) < 1024$$
$$10p + 30p - 30 < 1024$$
$$40p < 1024 + 30$$
$$p < \frac{1024+30}{40}$$
$$p < 26.35$$
$$p = 26$$

Rough Work

2 / Part No.

Order p_{leaf}

$$p_{leaf} \times (\text{keyfield size} + P_R) + P < B$$

$$p_{leaf}(30+10) + 10 < 1024$$

$$p_{leaf} \times 40 < 1024 - 10$$

$$p_{leaf} < \frac{1024-10}{40}$$

$$p_{leaf} < 25.35$$

$$p_{leaf} \leq 25$$

we will use p_{leaf} for leaf-level.

(a) no. of blocks/nodes at leaf level $= b_1 = \left\lceil \frac{1000,000}{25} \right\rceil$

$$b_1 = 40000$$

for internal levels we will use p_{order}.

$$b_2 = \text{ceiling}\left(\frac{b_1}{26}\right) = \text{ceiling}\left(\frac{40000}{26}\right) = 1539$$

$$b_3 = \text{ceiling}\left(\frac{1539}{26}\right) = 60$$

$$b_4 = \text{ceiling}\left(\frac{60}{26}\right) = 3$$

$$b_5 = \text{ceiling}\left(\frac{3}{26}\right) = 1$$

So the resulting tree will have 5 levels.

(b) no. of nodes at level b_1 (leaf level) will be = 40,000

at level b_2 no. of nodes will be = 1539

similarly at $b_3 = 60$

at $b_4 = 3$

at $b_5 = 1$

total nodes will be equal to $b_1 + b_2 + b_3 + b_4 + b_5 = 40,000 + 1539 + 60 + 3 + 1$

$= \boxed{41603}$

:cords fast
ny issue.

5 mod 8 = 5
= 101

5

Table. (for help)

000
001
010
011
100
01
10
11

Q1 Part iv.

Q

(c) As all the nodes (leaf + non-leaf) fill factor of 75%. so:

for b_{leaf}

$$0.75 \times b_{leaf} = 0.75 \times 25 = 18.75 \approx 19$$

So each leaf level node will have approximately 19 key records.

for b:

$$0.75 \times b = 0.75 \times 26 = 19.5 \approx 20$$

So each non-leaf level node will have approximately 20 key records.

So, now we will calculate no. of levels again.

no. of blocks/nodes at leaf level $= b_1 = \left\lceil \frac{1000,000}{19} \right\rceil$

$$b_1 = 52632$$

$$b_2 = \text{ceiling}\left(\frac{52632}{20}\right) = 2632$$

$$b_3 = \text{ceiling}\left(\frac{2632}{20}\right) = 132$$

$$b_4 = \text{ceiling}\left(\frac{132}{20}\right) = 7$$

$$b_5 = \text{ceiling}\left(\frac{7}{20}\right) = 1$$

So in this case we will again have 5 levels.

Part No.

Rough Work

Question # 3

as in both the queries we are joining the two tables on CID and OID.

So we will create primary index on CID for customer table and primary index on OID for order table (as we are accessing data frequently on the base of these two key values).

on genders we will create Bitmap as its selection is low. (for customer table)

~~for~~ date in order table we can create clustering index if dates are in ordered form or secondary index if date is in unordered.
For date (in order) we will use B+ tree as it is a range query so B+ tree works best in range queries.
Note: to make query more fast we can create hash index on CID and OID.

Q / Part No.

Q / P

Question # 4

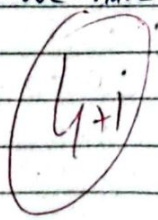
Key values: 19, 14, 21, 8, 23, 15, 31, 25

hash function is $h(k) = k \bmod 4$

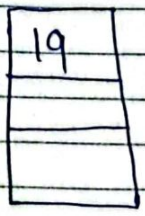
as each bucket can hold 3 records So first 3 records will be inserted as it is without any overflow.

Binary table	
0	00
1	01
2	10
3	11

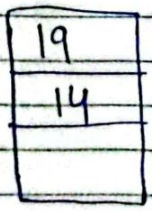
Step I: Initially global depth d=0 and local depth d'=0 and we have only 1 bucket



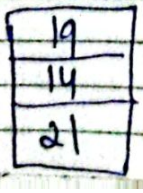
insert 19 $19 \bmod 4 = 3$ (11)



insert 14 $14 \bmod 4 = 2$ (10)



insert 21 $21 \bmod 4 = 1$ (01)

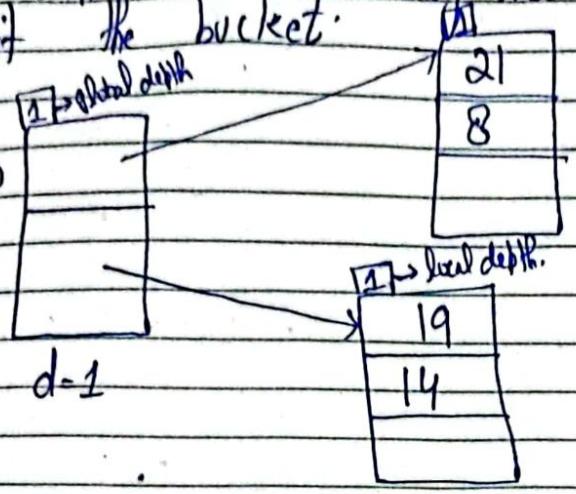


: No.

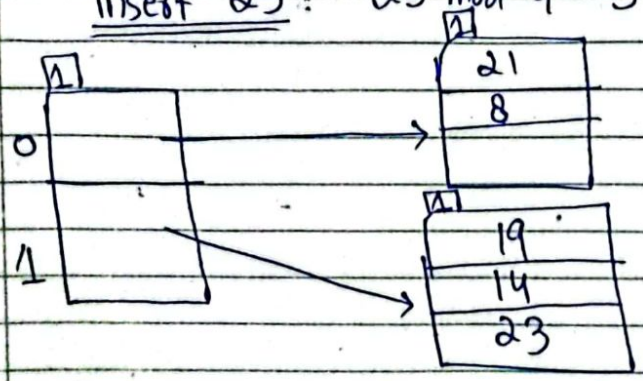
Rough Work

insert 8 8 mod 4 = 0 (00)

as overflow occured, since d'=d so we will increase directry size and also split the bucket.

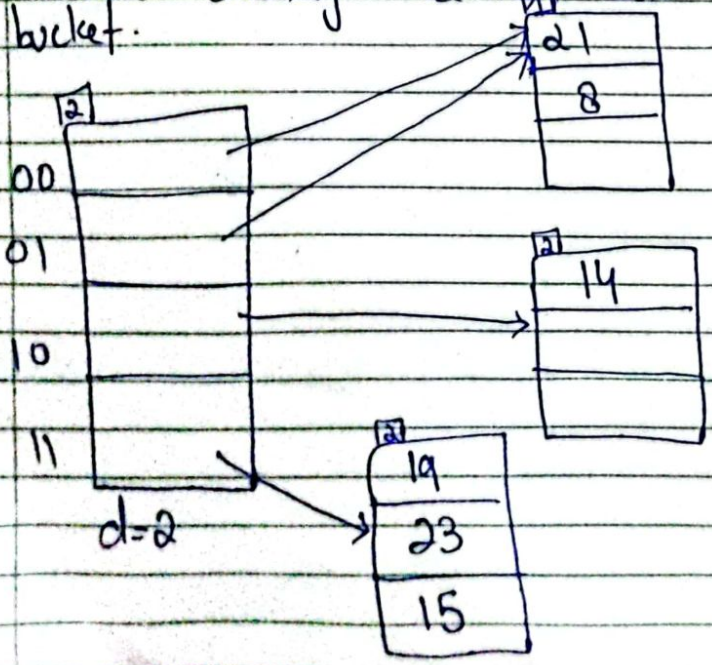


insert 23: 23 mod 4 = 3 (11)



insert 15 15 mod 4 = 3 (11)

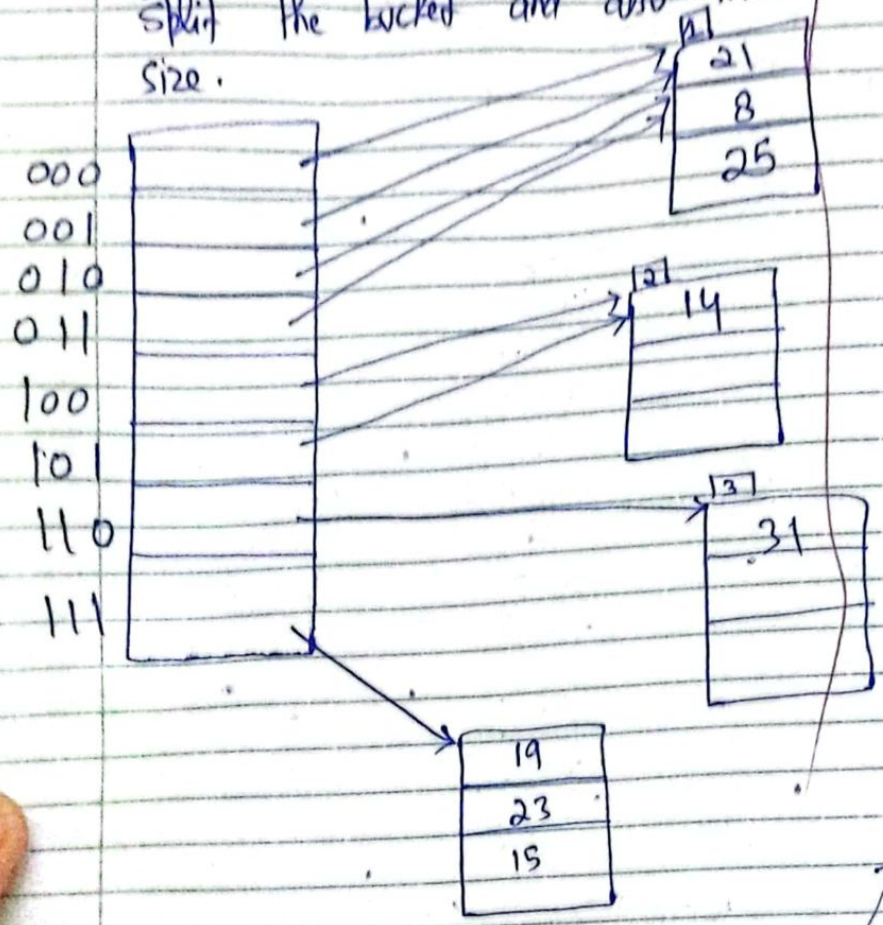
overflow as d'=d, so we will increase directory size and also split the bucket.



Q / Part No

insert 31 31 mod 4 = 3 (11)

again overflow as d' = d so we will split the bucket and also increase directory size.



insert 25 25 mod 4 = 1 (01)