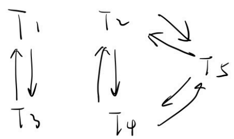
Question1

Time	T1	T2	T3	T4	T5	Time	T1	T2	T3	T4	T5
1		x2 = Read(X)				1		S-lock(X) x2 = Read(X)			
2					x5 = Read(X)	2					S-lock(X) x5 = Read(X)
3				y4 = Read(Y)		3				S-lock(Y) y4 = Read(Y)	
4				y4= y4* 2		4				y4= y4* 2	
5	z1 = Read(Z)					5	S-lock(Z) z1 = Read(Z)				
6		x2 = x2 + 1				6		x2 = x2 + 1			
7			z2 = Read(Z)			7			S-lock(Z) z2 = Read(Z)		
8					x5 = x5 * 2	8					x5 = x5 * 2
9	z1 = z1 * 3					9	z1 = z1 * 3				
10		y2 = Read(Y)				10		S-lock(Y) y2 = Read(Y)			
11			z3 = z3 -1			11			z3 = z3 -1		
12				Write(Y, y4)		12				X-lock(Y) Abort(Y)	
13		Write(X, x2)				13		X-lock(X) Wait Write(X, x2)			
14	Write(Z, z1)					14	X-lock(Z) Wait Write(Z, z1)				
15		y2 = y2 + x2				15					X-lock(X) Abort
16				x4 = Read(x)		16		Write(X,x2)			
17					Write(X, x5)	17		y2 = y2 + x2			
18				x4 = y4 + x4		18			X-lock(Z) Abort		
19			Write(Z, z3)			19	Write(Z, z1) Commit				
20		Write(Y, y2)				20		X-lock(Y) Write(Y, y2) Commit			
21				Write(X, x4)		21					

It is has circles therefore is not schedule seriaizable.

X = 2, Y = 4, Z = 9

Time	T1	T2	T3	T4	T5
1		x2 = Read(X)			
2					x5 = Read(X)
3				y4 = Read(Y)	
4				y4 = y4 * 2	
5	z1 = Read(Z)				
6		x2 = x2 + 1			
7			z2 = Read(Z)		
8					x5 = x5 * 2
9	z1 = z1 * 3				
10		y2 = Read(Y)			
11			z3 = z3 - 1		
12				Write(Y, y4)	
13		Write(X, x2)			
14	Write(Z, z1)				
15		y2 = y2 + x2			
16				x4 = Read(x)	
17					Write(X, x5)
18				x4 = y4 + x4	
19			Write(Z, z3)		
20		Write(Y, y2)			
21				Write(X, x4)	



Question 2

solution:

1. insert 16, 22

   Bucket 0 : (16, 22)

   Next bucket to be split: bucket 0

2. insert 3, 7

   Bucket 0: (16, 22)

   Bucket 1: (3,7)

   Next bucket to be split: bucket 0

3. insert 1

   Bucket 00: (16)

   Bucket 10: (22)

   Bucket 1: (3,7)-(1)

Next bucket to be split: bucket 1

4. insert 15, 2

   Bucket 00: (16)

   Bucket 01: (1)

   Bucket 10: (22, 2)

   Bucket 11: (3, 7) - (15)

   Next bucket to be split: bucekt 11

5. insert 19, 9, 4

   Buckett 00: (16, 4)

   Bucket 01: (1, 9)

   Bucket 10: (22, 2)

   Bucket 011: (3, 19)

   Bucekt 111: (7, 15)

   Next bucket to be split: bucket 00

6. insert 21, 0

   Bucket 000: (16, 0)

   Bucket 01: (1,9) - (21)

   Bucket 10: (22, 2)

   Bucekt 100: (4)

   Bucekt 011 (3, 19)

   Bucket 111 (7, 15)

   Next bucket to be split: bucket 01

7. insert 8

   Bucket 000: (16, 0) - (8)

   Bucket 101: (21)

   Bucket 10: (22, 2)

   Bucket 100: (4)

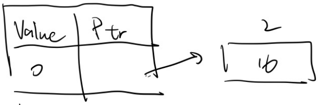
   Buckekt 011: (3, 19)

   Bucket 111: (7, 15)

   Bucket 001: (1, 9)

   Next bucket to be split: bucket 10

① 16



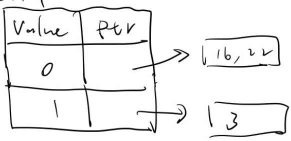
level: 0

② 22

level: 0

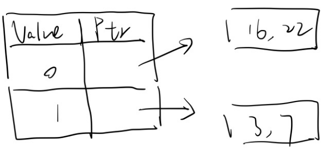
③ 3

level: 1



④ 7

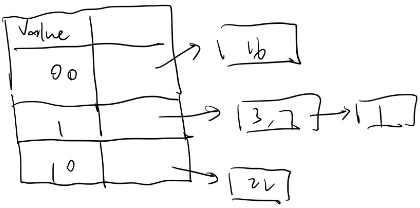
level: 1



000000	0
000001	1
000010	2
000011	3
000100	4
000101	5
000110	6
000111	7
001000	8
001001	9
001010	10
001011	11
001100	12
001101	13
001110	14
001111	15
010000	16
010001	17
010010	18
010011	19
010100	20
010101	21
010110	22
010111	23
011000	24
011001	25

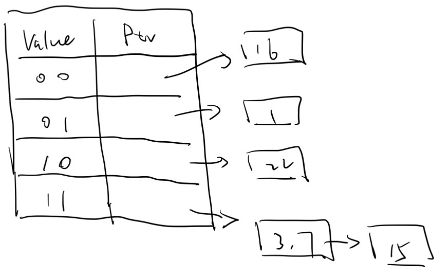
⑤ 1

level : 1



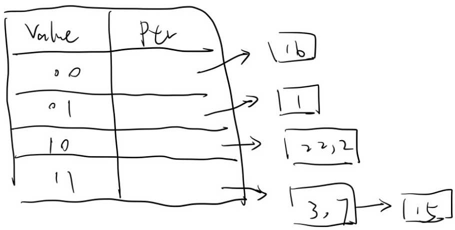
⑥ 15

level : 2



⑦ 2

level : 2



⑧ 19

level: 2

Value	Ptr
00	→ 16
01	→ 1
10	→ 22, 2
011	→ 3, 19
111	→ 7, 15

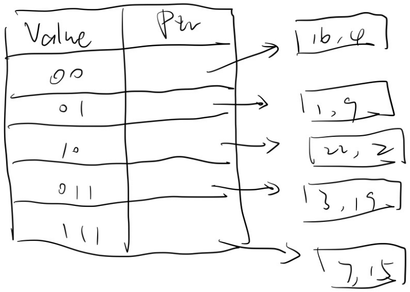
⑨ 9

level: 2

Value	Ptr
00	→ 16
01	→ 1, 9
10	→ 22, 2
011	→ 3, 19
111	→ 7, 15

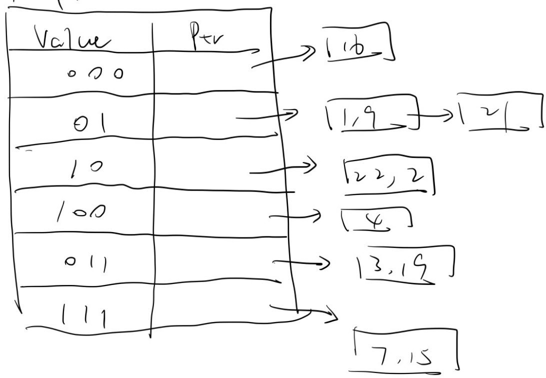
10 4

level: 2

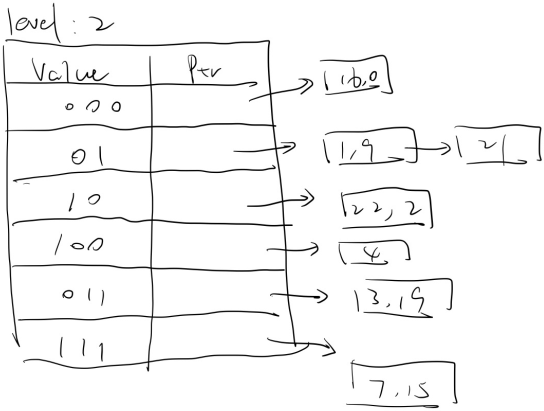


11 21

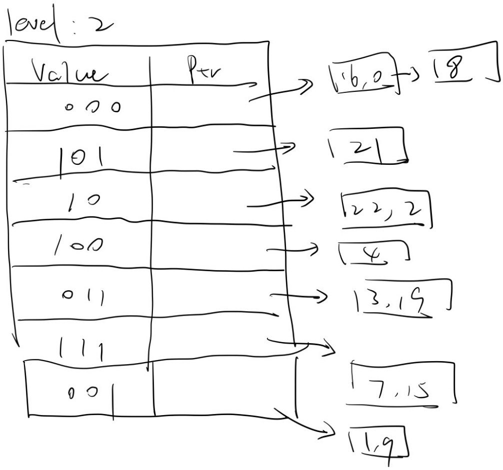
level : 2



12 0



13 8



Question 3

a.

A: $\frac{100 \times 20{,}000}{2{,}000} = 1,000\, pages$

B: $\frac{200 \times 30{,}000}{2{,}000} = 3,000\, pages$

C: $\frac{300\times60{,}000}{2{,}000} = 9,000\ pages$

b.

i. sequential scan, read all. Because C is evenly split between two tracks, which means we have to seek+rotation twice.

$Time = 2\ (seek + rotation) + 9,000 \times (read) = 2 \times 50ms + 9,000 \times 0.5ms = 4,600ms$

Therefore, the time for this query is 4600 ms.

ii. $k \times (50 + 0.5) = 50.5k\ ms$

Therefore, the time for this query in worst case is $50.5k$ ms.

iii. $50.5k \geq 4,600 \rightarrow k \geq \left\lceil \frac{4{,}600}{50.5} \right\rceil \rightarrow k \geq 90$

Therefore, when $k \geq 90$ such that (i) is a faster way to execute the query than (ii)

Question 4

a. $A_{A.x=C.x} \bowtie C$ is small, and $A_{A.x=B.x} \bowtie B$ or $B_{B.x=C.x} \bowtie C$ is quite large

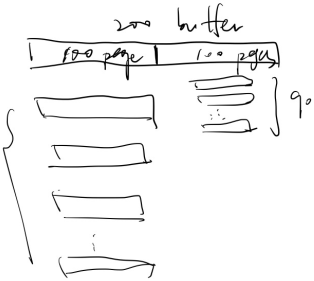
b. A should be in the outer loop.

Because A has 1000 pages smaller than C has 9,000 pages. Nested loop, the each block in the outer loop need to be read once. For inner loop, each block has to be read once for each block of the outer loop. Therefore, the number of block access = $b_r + b_r \times b_s$. Therefore, $b_r \times b_s$ is fixed, depend on the blocks of A and the block C. But when b_r smaller, the block access will more smaller, which means, the number of block access is up to b_r . Therefore, the outer loop should be smaller table. Therefore, A should be in the outer loop.

c.

c. $B = 1000$

$C = 9000$



10

$$\frac{9000}{100} = 90 \text{ segments}$$

10×90

A need to be read once

$$\frac{1000}{100} = 10 \text{ segments}$$

Time = read outer loop + read inner loop

$$= 1000 \text{ read} + \frac{1000}{100} \times \frac{9000}{100} \times 100 \text{ read}$$

$$= 1000 \text{ read} + 10 \times 9000 \text{ read}$$

$$= 91000 \text{ read}$$

$$= 91{,}000 \times 0.5 \text{ ms}$$

$$= 45500 \text{ ms}$$

d.

d. ∵ A.x has values between [1, 1000], evenly distributed

C.x has values between [201, 2200], evenly distributed.

and ∵ A has 20,000 pages, C has 60,000 pages.

$$\therefore 20,000 \times \frac{1000-201+1}{1000} \times \frac{60,000}{2200-201+1} \times \frac{1000-201+1}{2200-201+1}$$

$$= 20,000 \times \frac{800}{1000} \times \frac{60,000}{2000} \times \frac{800}{2000}$$

= 192000 tuples

e. The bytes per tuple of join = 100 + 300 = 400 bytes.

$$\text{pages} = \frac{\text{byte per tuple} \times \text{the number of tuples}}{\text{byte per page}}$$

$$= \frac{400 \text{ byte} \times 192000 \text{ tuples}}{2000 \text{ byte}}$$

= 38400 pages.

f.

f.

we have 38400 pages from e.

we have to use the output buffer to write it in the disk.

time = 38400 × write = 38400 × 0.5ms = 19200 ms.

g.

g.

buffer

200 pages.

∵ Every value there appears in B.x appears in A.x

∴ can Join whole B tuples 3000 pages.

B as outer loop. particles as inner loop.
38400 pages

| 100 page | 100 pages |

outer loop: read once.

3000 read

$$\text{Time} = 3000 \text{ read} + \frac{3000}{100} \times \frac{38400}{100} \times 100 \text{ read}$$

$$= 3000 \times 0.5\text{ms} + 30 \times 38400 \times 0.5\text{ms}$$

$$= 577500 \text{ ms}$$

h.

h. overall time for the query. Assume not write the final result to the disk.

$A \bowtie C \bowtie B$

Over all cost = Sum of cost of individual operations + cost of writing intermediate result to disk.

= Time ($A \bowtie C$) + write in disk + Read (result $A \bowtie C$) + $A \bowtie C \bowtie B$

$= c + f + g$

= 45500 ms + 19200 ms + 577500 ms

= 64220 ms

Question 5

a.

$$range = \frac{30}{10\ columns} = 3$$

Range	Frequency
1-3	10
4-6	1
7-9	5
10-12	4
13-15	3
16-18	11
19-21	1
22-24	1
25-27	8
28-30	6

b.

$frequency = \frac{50}{10} = 5$

Range	Frequency
1-2	5
2	5
4-8	5
8-12	5
13-16	5
16-17	5
17-20	5
24-25	5
26-28	5
28-30	5

c.

i.

c.

where $d.x = 2$

i ① equal-width:

range	frequency
1-3	10

$10 \times \frac{1}{3} = \frac{10}{3} \approx 3.33 \approx 3$

② equi-depth:

range	frequency
1-2	5
2	5

$5 \times \frac{1}{2} + 5 = 2.5 + 5 = 7.5 \approx 8$

ii.

iii. where d.x >= 16 and d.x <= 30

① equal-width:

range	frequency
16-18	11
19-21	1
22-24	1
25-27	8
28-30	6

$11+1+1+8+6 = 11+2+8+6 = 13+14 = 27$ tuples.

② equi-depth

range	frequency
13-16	5
16-17	5
17-20	5
24-25	5
26-28	5
28-30	5

13-16 : 13, 14, 15, 16

$\frac{1}{4} \times 5 + 5 \times 5 = 1.25 + 25 = 26.25 = 26$ tuples

iii.

iii. $p=11$, $q=16$

where d.x >= 11 and d.x <= 16

① equal-width

range	frequency
10 - 12	4
13 - 15	3
16 - 18	11

10 - 12 : 10, 11, 12

16 - 18 : 16, 17, 18

$\frac{2}{3} \times 4 + 3 + \frac{1}{3} \times 11 = \frac{8}{3} + 3 + \frac{11}{3} = \frac{19}{3} + 3 = 9.\dot{3} \approx 9$ tuples.

② equi-depth :

range	frequency
8 - 12	5
13 - 16	5
16 - 17	5

8 - 12 : 8, 9, 10, 11, 12

16 - 17 : 16, 17

$\frac{2}{5} \times 5 + 5 + \frac{1}{2} \times 5 = 2 + 5 + 2.5 = 7 + 2.5 = 9.5 \approx 10$ tuples.

Question 6

solution:

Write-ahead logs(WAL): All operations must be logged first. Log records must be force to stable storage before acutally operations can be executed.

- Atomicity

  For example, if log after excuation and there is a crash between excuation and log.

  A = $1000 and B = $ 2000, A transfer $50 to B.

  ```
  write A = $ 950 (T has not commit yet)
  ------------------------------ crash
  log(A excuation)
  ```

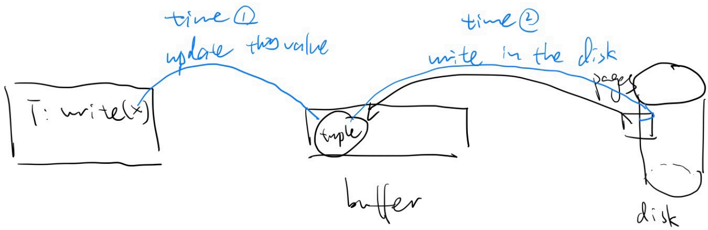
  When the system restarts, it will not any record about A has already losed $50. It cannot find the write operation on the log. We cannot do undo the transaction, because we do not know happen the thing. Therefore, it is not satified the atomicity.

- Durability

  Suppose the crash occurs right after a transaction T committed.

  ```
  write A = $ 950 (T commit)
  ------------------------------------ crash
  log (A excuation)
  ```

  When the system restart, T may have written something onto the disk. However, the writes may not have propagated to the disk. And log is after the excuation, therefore, we can not find the transaction that has committed. We cannot do the redo. Therefore, it is not satified the durability.



> By the way, T commited also need time. And in some situation, for example, if five other transactions want to read x. It's probably better to store the x tuple in the buffer for a while longer before write it to the dis and release the buffer for other transcations. And there is crash at that moment. We didn't log the exucation. when the system recovers, it cannot find the write operation on the log. We cannot redo.

Therefore, it log after the execute, it will conflict atomicity and durability.