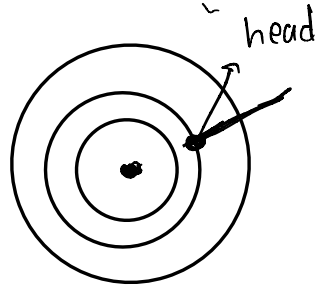
Disk scheduling

disk contains tracks, head moves between tracks to acess infosmation. This movement is controlled by OS.



FCFS (First come first serve)

at $t=0$, head is at location 0, movement over single location takes unit time

Ready time	Location	FCFS
6	45	51
14	53	59
20	10	72

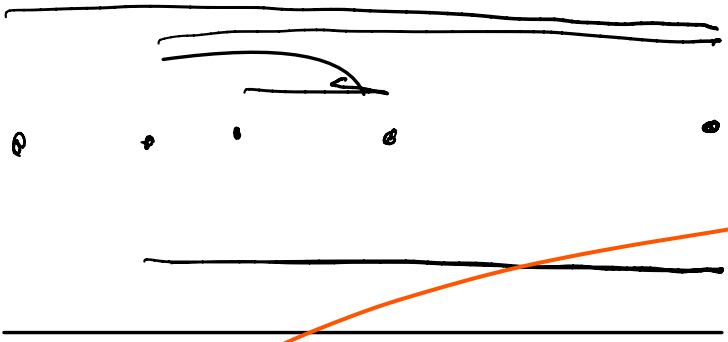
Ready time	Location	FCFS	SSTF	SCAN	LOOK
6	45	51	51	45	51
14	53	59	69 (89)	53	59
20	40	72	56	40	46
35	38	74	58	38	44
38	30	82	66	170	82

SSTF - Shortest seek time first.

upon multiple requests nearest request would be done first.

scan - 0 to 100 to 0 irrespective of the tasks starting from t=0
look - version of scan
- starts only when it receives a task
- stops where it is if there is no pending tasks
- maintains inertia if it has a task in the inertial direction
- changes direction if it doesn't have a task in inertial direction but only in opp. dir

Ready time	Location	FCFS	SSTF	SCAN	LOOK
6	45	51	51	45	51
14	53	59	89	53	59
20	40	72	56	40	46
35	38	74	58	38	44
38	30	82	66	170	82
60	44	96	80	156	68



construct an example in which FCFS is better than SSTF

Starvation: If a job is never served then this problem is called starvation.

9th Jan.

Memory

Tape, disk } non-volatile and slow, sequential in nature

random access memory

cache

register

demand paging: transfer of data from secondary storage ⟷ RAM on demand.

---

a : 65	b : 37
c : 84	d : 49
e : 36	f : 74
g : 36	h : 42
i : 49	j : 63

← DISK

RAM

	a	b	c	d	e	f	g	h	i	j	
Print(d)	0 94	0 63	0 38	1 49	0 42	0 18	0 21	0 31	0 89	0 79	DR
Print(a)	1 65										DR
Print(h)								1 42			DR
Print(a)											NT
Print(c)		1 84									DR
h = 36								2 36			NT
c = 29		2 29									NT
f = 36					2 36						NT

OVER

R → D

(h) 36, 29, 36(f)
(c)

1 65 . 2 29 1 49 2 36 2 36

memory locations with dirty bit 2 are transffered.

Page size: Ram is divided into equal memory blocks of page size.

a : 65 b : 37

c : 84 d : 49

e : 86 f : 74

g : 36 h : 42

i : 49 j : 63

Pages

P

Q

R

S

T

	P	Q	R	S	T
84 49 → Ram		1 84 49			
65 37 → Ram	1 65 37				
36 42 → Ram				1 36 42	
no transfer				4 36 36	
no transfer		4 29 49			
no transfer		garbage ← 3 36			
	(36,36), (29,49) R → DISK				
	(86,74) Disk → Ram			(86,36) Ram → Disk	

dirtybit is assosciated with every memory in RAM

non paging case

Dirty :

1	disk = ram	0: disk ≠ ram disk is correct
2	disk ≠ ram	

Dirty : paging

dirty	
0	disk ≠ ram (disk is correct)
1	disk = ram
2	1st correct
3	2nd correct
4	disk ≠ ram (ram is correct)

0 → 1, 2, 3

1 → 4

2 → 4

U	V	W	X	Y
d_{49}				
d_{49}	a_{65}			
d_{49}	a_{65}	h_{42}		
"				
d_{49}	a_{65}	h_{42}	c_{84}	
d_{49}	a_{65}	$\dot{h}_{36}$	C_{84}	
d_{49}	a_{65}	$\dot{h}_{36}$	$\dot{c}_{29}$	
d_{49}	a_{65}	$\dot{h}_{36}$	$\dot{c}_{29}$	$\dot{f}_{36}$

in case of memory constraints, replacement algorithms are used.

K	L	M	N
Q 84 49			
Q 84 49	P 65 37		
Q 84 49	P 65 37	S 36 42	
	"		
	"		
Q 84 49	P 65 37	S 36 36	
Q 29 49	P 65 37	S 36 36	
Q 29 49	P 65 37	S 36 36	R 06 36

↳ garbage

---

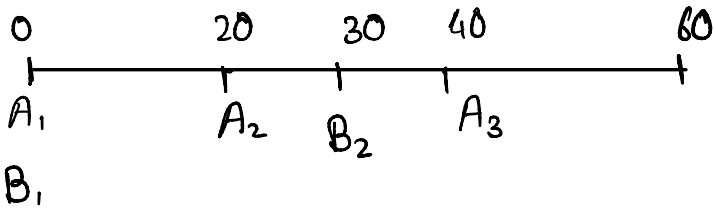
replacement algorithms ⟶ FIFO - First in first out

LRU - least recently used

Optimal -

10th Jan,

Rate monotonic scheduling: (for periodic jobs)

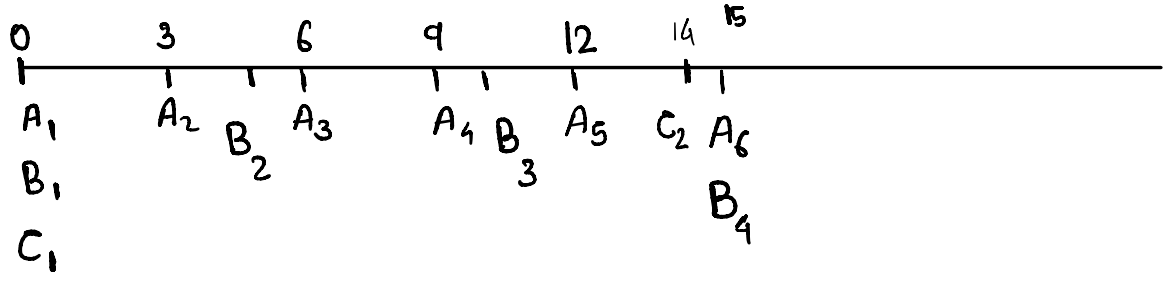


Job	period	service
A	20	5
B	30	12

A_1 0-5
B_1 5-17
A_2 20-25
B_2 30-42
A_3 42-47

deadline: a job should be completed before next job of same type comes.

Job	period	service
B	5	1
A	3	1
C	14	3



$A_1 \rightarrow$ 0-1 $A_2 \rightarrow$ 5-6 A_3 7-8 A_4 9-10

$B_1 \rightarrow$ 1-2 $B_2 \rightarrow$ 6-7 B_3 10-11 A_5 12-13

$C_1 \rightarrow$ 2-5 $C_2 \rightarrow$ 14-17

* a job with lesser timeperiod will be given priority. (unlike other algorithms where job time is used as criteria).

(service time)

Job	period	service
B	50	9
A	30	11
C	140	30

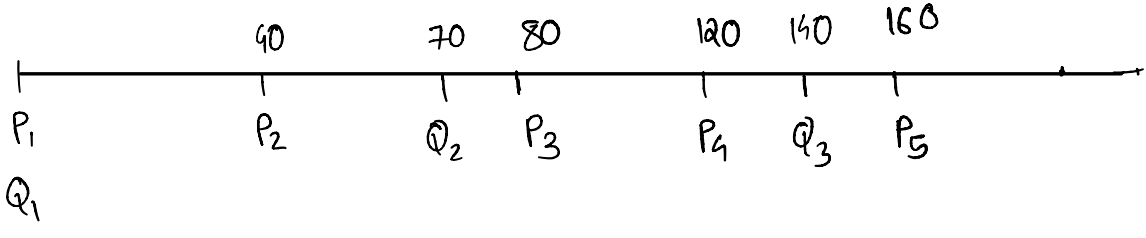
at $t = 140$ C_2 is served 140 - 170

at $t = 170$ A_6, B_4 are in tube

if B_4 is served based on service time A_6 would miss the deadline.

IMPPPPPPPP

Job	period	service
P	40	5
Q	70	50



(0) P_1 0-5

(40) P_2 55-60

(80) P_3 - 80-85

(120) P_4 : 135-140

(160) P_5 : 190-195

P_6 : 195-200

(0) Q_1 5-55

(70) ~~Q_2 70-120~~

85-135

↳ (waited for low time period job)

(140) Q_3 : 140-190 (no waiting).

non-feasability : Jobs can never be done.
no matter, any method applied.

Job	period	service	
U	20	12	60%
V	30	15	50% (impossible)

Sum of percentage of service time > 100 then impossible is a sufficient condition not a neccessary one

16th Jan

Main memory management

(A) continuous allocation

Best fit only

- Static partition — Best fit available
- Dynamic partition — First fit, Best fit, worst fit.

(B) Non continuous

- Paging
- Segmentation

H	G	F	E	D	C	B	A	Job
10	20	10	20	10	20	10	20	Service time
35	5	10	10	14	8	3	5	Memory needed
0-10	0-20	0-10	0-20	0-10	0-20	0-10	0-20	Service interval
55-90	50-54	40-49	30-39	16-29	8-15	5-7	0-4	Memory allocated

		N	M	L	K	J	I	Job
					7	5	20	Service time
		10	14	27	8	40	8	Memory needed
					10-17	20-25	0-20	Service interval
						0-39	90-97	Memory allocated

	For Job K	
First fit	16-23	
Best fit	40-47	
worst fit	55-62	

↳ takes memory from biggest past.

after $t = 10$sec, memory holes of size,

according to First fit : 3, 6, 10, 35

Best fit : 3, 14, 2, 35

worst fit : 3, 14, 10, 27

---

after addition of Jobs L, M, N :

$L = 27$

$M = 14$

$N = 10$

worst fit performs good.

First fit, Best fit dont work optimally.

If jobs added are L, M

35, 14 in this case only

best fit works optimally

If jobs added are L M N

35, 10, 6

First fit works optimally

remaining methods dont.

Memory compaction:

a time t = 10,

G is shifted from

50-54 to 16-20.

Now J can be allocated from

50-89.

at $t = 13$, Job k needs 5 more memory.

if memory is available, it will be allocated.

if continuous memory is not available

1. Terminate
2. Allot another past/else wait.
3. Deny.

Static partitioning :

Partitions are made before jobs arrive

P	5
Q	40
R	20
S	8
T	12
U	5
V	10

		F	E	D	C	B	A	Job
								Service time
		9	35	15	23	10	18	Memory need
								Service interval
only wait	available		wait.	Q	V	R	memory allocated	
0	3	0	0	17	0	2	internal fragmentation	
18	0	0	30				External frag.	

total alloted : 51

total exist : 100

available : 49

combined memory of available partitions is external fragmentation.

at job D

external fragmentation :

5 + 8 + 12 + 5 = 30

internal fragmentation = 19

external fragmentation is considered only when sum of partitions ≥ memory req.
(non alloted - available)
^
(non-alloted - available)

and individually every partition
^ < memory req

Fragmentation — internal
external
Table.

From example of dynamic allocation :

	First fit	Best fit	Worst fit
Internal	0	0	0
External	0	0	0

in dynamic partitioning,

internal fragmentation concept doesnot make any sense.

for job K

* if memory required = 70

holes: 3, 14, 10, 35 - no internal, no external.

* if memory required = 40

external fragmentation = 3+14+10+35

= 62 //

17th Jan

Processor scheduling:

Job	ready	service time	first come first serve	wait	shortest job first	wait	Preemptive scheduling	wait time.
A	5	15	5 - 20	0	5 - 20	0	5 - 12, 16 - 24	4
B	10	12	20 - 32	10	29 - 41 (24 - 36)	19 (14)	24 - 36	14
C	12	~~9~~ 4	32 - 41 (32 - 36)	20 (30) (30)	20 - 29 (20 - 24)	8 ~~(27)~~ (22)	12 - 16	0 (18)

assuming

~~2seconds time is required for swap-in,~~

2 sec Swap-out. 1 sec swap in

with swap time	wait
5-12, 19-27	7
27-39	17
14-18	2 (26)

Job	Ready	Service	with swap	wait time	FCFS	wait time.
P	0	20	0-10 22-32	12	0-20	0
Q	10	9	12-21	2 (11)	20-29	10 (10)

Job	Ready	Service	with swap	wait time	FCFS	wait tim
R	0	20	0-10 17-27	7	0-20	0
S	10	4	12-16	2 (9)	20-24	10 (10)

Job #	Ready	Service	Interval waiting (DETN of ready and not assigned)		FCFS	
R	0	20	0-10, 22-32	12	0-20	0
Q	10	9	12-21	2	20-29	10
				14		10

P	0	20	0-10, 17-27	7	0-20	0
S	10	4	12-16	2	20-24	10
				9		10

23/01/23 watch lecture

resource allocation

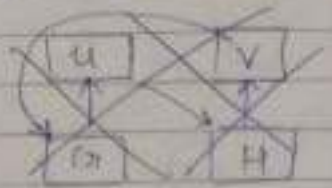
* Job A Run 10, needs P, run 15, needs Q, run 10
Job B Run 12, needs Q, run 20, need R, run 5

Resource Alloc^{n} graph Job Y waits for resource Z: Y → Z
Job Y holds resource X: Z → Y

A [0-10] [10-25 P] [25-37 waits] [37-47 PQ]
B [0-12] [12-32 Q] [32-37 QR]

* Static method Holding higher & demanding lower is not permitted (here, alphabetical order)
dynamic
Deadlock prevent
Deadlock avoid

* Job U Run 10, needs G, run 15, need H, run 10
Job V Run 12, need H, run 20, need G, run



Job U "
Job V Run 12, need G, need H, run 20+5

deadlock

U: [0-10] [10-25 G] [25- wait for H]
V: [0-12] [12-32 H] [32- wait for G]

U: [0-10] [10-25 G] [25-35 GH]
V: [0-17] [17- wait for G] [35-60 GH]
Job V: Run 12, need G, need H, run 20+5

Dynamic Method Job given to C but not to B

Job A Run 10, needs P, run 15, needs Q, run 10
Job B Run 12, needs Q, run 20, needs P, run 5
Job C Run 14, need Q, run 30, need K, run 12

A: [0-10] [10-25 P] [25-35 PQ]
B: [0-12] [wait for Q though available]
C [0-44] [14-44 Q] [44-56 QK] ; Q released @ 56
A [0-10] [10-25 P] [25-56 wait for Q] [56-66 PQ]
B [0-12] [wait for Q] [66-86 Q] [86-91 PQ]
D: [0-16] [16-26 K]

Job D: Run 16, need K, run 10

→ Static method on Job A-D : Q → Q' after sorting
(see next pg)

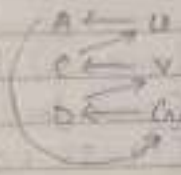
31

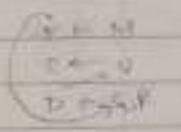
→

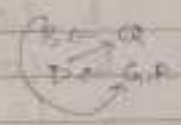
Job	Got	Will need
A	U	R, G
B	Q	G, H
C	V	M, T, U
D	R, G	Q, V
E	M	G, T

D asks for G

↳ Again deadlock !







→ R asks for G

Job	Got	Will need
A	U	R, G
B	R, G	M
C	V	M, T, U
D	R,	Q, G
E	M	G, T

B Q G H → can be released

DE can be done

A

C

* Banker's Algorithm

- Bank has 256

Person	Given	Will need
Vasu	90	60
Gyan	40	50
Sami	5	30

Vasu asks 10

Bank has 46

R	90+10 = 100	60-10 = 50
G	40	30
S	7	30

Gyan asks 10

Bank has 46

R	90	60
G	40+10 = 50	50-10 = 40
S	5	10

30

3' Job A: Run 10, need P, run 11, need Q, run 10
Job B: Run 12, need P, need Q, run 20+5
Job C: Run 14, need K, need R, run 30+12
Job D: Run 16, need K, run 10

A: [0-10] [10-25 P] [56-66 PQ]
B: [0-12] [wait for P] [66-91 PQ]
C: [0-14] [14-56 KQ]
D: [0-16] [wait for K] [56-66 K]

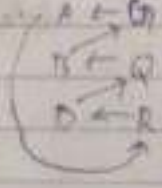
Static Method

Job	Got	Will need
A	U	R G1
B	Q	G1 U
C	V	M T U
D	R	Q V G1
E	M	G1 T

A asks R, wait for D
D asks Q, wait for B
B asks U, wait for A
} Deadlock

→ A asks for G1

Job	Got	Will need
A	U, G1	R
B	Q	G1
D	R	Q



Deadlock
∴ A asks for G1 Not given

max

How much money can be given to Ravi, Gyan, Sam?

	Bank	40	40
	Ravi	30	60
Max is ←	Gyan	40+16	50-16=34
	Sam	5	30

Q) Bank has 40

	Person	Given	Will need
Max ←	Ravi	30	60 ✓
45 of Sam	Gyan	40	50 ✓ } → 90
	Sam	5	30 ✓ } → +5 = 50

29/01/23

Disk		
t	txt	d
h	c	k

3 data, 1 pointer to next

logical u m c d k y h c t k name t.txt

Free a b c d e f g h i j k l m n o p q r s t u v w x y z

find 8th letter u m c d k y t h c

insert 5th letter as V

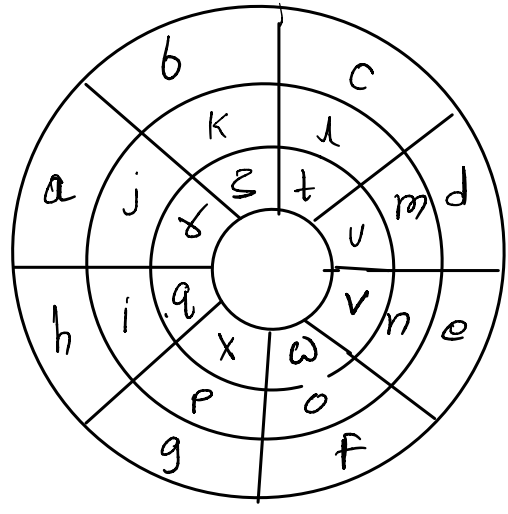
logical a b i k c d m p u d k h c y name h.c

k a h
i k c
m p m
t h a
d d
y k y

delete 6th letter (next ng)

1. least
h dky
u hc
m th

24th jan



t.txt :

u m c d k y h c t k

d : u m c

h : d k y

u : h c

m : t k

in disk

t.txt → d

every block : 3 data + 1 pointer to next.

in d : u m c h (next)

in h : d k y u

in u : h c m

in m : t k –

File: h.c

ah ikcd mpu dk hcy

k : ah	n
n : ikc	b
b : dm	q
q : Pu	V
V : dKh	c
c : cy	-

Find 8^{th} letter of t.txt u m c (b) d k y (U) h c

insert 5^{th} letter as V :

block_h : d k y U (before)

block h : d V _ f

new block_f : k y _ U

now,

t.txt : U m c d V k y h c t k

d: umc

h: dv

f: ky

v: hc

m: tk

- insert 10^{th} letter as h

d: umc (h) dv (f) ky (v) hch

↓ new letter

- delete 2^{nd} letter

d: uc (h) dv (f) ky (v) hc (m) tk

- delete 9^{th} letter:

d: uc (h) dv (f) ky (v) hck (m)

---

delete 6^{th} letter in h.c

k: ah (n) ikc (b) ~~d~~ m

↳ to be deleted

method-1 :

n : ik [b]

b : cm

method-2 :

n : ikc q

q : mpu

delete 6th letter

insert 6th letter as i

insert 4th letter as 'u'

delete 4th letter.

Table fragmentation:

30th Jan

H	G	F	E	D	C	B	A	Job
9	7	6	5	4	3	1	0	Ready
33	10	15	8	10	10	5	8	Memory need
86-98	56-65	41-55	33-40	23-32	13-22	8-12	0-7	Allocate
2	15	30	9	20	13	20	15	Service
9-11	7-22	6-36	5-14	4-24	3-16	1-21	0-15	Run

K	J	H	J	I
20	19	18	17	10
8	9	20	32	5
	13-21	66-85		66-70
		5	3	1
		18-23		11-12

at t=9

66-99 free

at t=17, 33-40, 0-7, 13-22, 66-99 is free.

at $t=9$

66 - 99 free

at $t=17$, 33 - 40, 0 - 7, 13 - 22, 66 - 99 is free.

⇓ (information regarding memory is also stored in memory).

66 - 99 (m - 99)

33 - 40 m99

0 - 7 m-98

13 - 22 m-97

(66 - 99) → 66 - 96 m-96

Storage of information regarding free space of memory in memory is known as table fragmentation.

- Because of table fragmentation, Job J is kept under waiting.

---

at t = 8,

m99	66, –

at t = 10,

m99	99, –

at t = 13,

m99	66, –

at t = 14,

m99	66, 40	66 - 99
m40	33, –	33 - 40

at t = 15,

m99	66, 40	66 - 99
m40	33, 7	33 - 40
m07	0, –	0 - 7

m99	66, 07
m07	0, 40
m40	33, –

at $t = 16$

m99	66, 40	66 - 99
m40	33, 7	33 - 40
m07	0, 22	0 - 7
m22	13, -	13 - 22

m99	66, 22
m22	13, 07

---

at $t = 17$ (32 units of memory needed)

Job J is allocated 66 - 97

m99	98, 40	98 - 99
m40	33, 7	33 - 40
m07	0, 22	0 - 7
m22	13, -	13 - 22

at $t = 19$,

m99	66, 40
m40	33, 7
m07	0, 22
m22	22, -

Y will be alloted 13 - 21 memory

at $t = 20$

m99	66, 07
m07	0, 22
m22	22, -

Memory :

95 - 98	m99 95, 31
19 - 31	m31 19, 45
42 - 45	m45 42, 68
60 - 68	m68 60, 51
49 - 51	m51 49, 12
8 - 12	m12 8, -

---

now, 55 - 58

changes:

m58 55, 31

m99 95, 58

---

if 55 - 59 m68 55, 51

---

if 69 - 72

m45 42, 72

m72 60, 51

---

if 48 - 48

m31 19, 68

m68 60, 51

m51 42, 12

	8		19		42		49	55	60
	12, -		31, 42		45, 60		51, 8		68, 49
	8 -12		19- 31		42- 45		49- 51		68- 68
								58, 95	

if 55 - 58 free

m55 58, 95

m99 55

	95	99
	98, 19	95
	95- 98	

if 55- 59 free:

m55 68, 49

m42 45, 55

if 69- 72 free

m60 72, 49

if 46- 48 free

m42 51 60

m60 68 8

m19 31, 60

m60 68, 42

m42 51, 8

if 7 memory needed :

M60 : 61, 49

if 4 memory needed :

M19 : 31, 60

31/01

Data Compression

Most zero(0)

(2) Block size 4

Complete 0 → 0
Single 1 (one) → 10 <addr of 1 in 2 bits>
more 1 → 11 < block >

0101, 0000, 0100, 0011

code 110101, 0, 1001, 110011

code 101011011000111011

0010 0110 0000 1011

11 0100 → if this is code

(1) Block size 4

data 0101000000111

0101, 0000, 0011

When complete
0 → 0
else 1 < block >
10101, 0, 10011

code 010110011101

0000 0110 0000 1101

(3) Block size (4)

Complete 0 : 00
Single 1 : 01 (address 2 bits)
Two 1s : 10 < addr in 3 bits >
More 1 : 11 < block >

1000, 0110,

code 0100 10011

0 - - 3

01 → 0
02 → 01
03 → 2
12 → 3
13 → 4 } 4,5
23 → 5 } 6,7

00001

(i) Block size 5

(101111) → single 1 encoding can be used

Complete 0 : 00
Single 1 : 01 <addr 1 in 2 bits>
Two 1s : 10 <addr in 4 bits>
More 1 : 11 <block>

01000 , 00101 , 00100 , 00001
↓ ↓ ↓ ↓
code→0101 101000 0110 101111

(ii) Block size 5

Complete 0 : 00
Single 1 : 01 <addr 1 in 3 bits>
Two 1s : 10 <addr in 3 bits>
01101 → 00101 , 01110 → 00011
More 1 : 11 <block>

01000 , 00101 , 00001 , 00011
↓ ↓ ↓
code→01001 01101 01100 01110
or
01111
0111

01→0
02→1
03→2
04→3
12→4
13→5
14→6
23→7
24 <015>
34 <016/17>

code→ 01101 01110 01111 10001
↓ ↓ ↓ ↓
00101 01001 00011 10100

(3) continued

1001 , 0101
↓ ↓
1000 1010X

0101, 0001, 0101, 1101

Now,
13 → 4, 5
23 → 6, 7

Code → 10100, 0111, 10100
10101 10101 | 1101

Use this ← lower one should be used

10100111 | 10101 | 1101
0101 0001 | 0101 1101

If only 4 is given to 23,
then 10101 is not decodable
After assigning both, further
data compression can be don

In (ii)
Since 1 → 01 (address in 3 or 2 bits)
Two 1 → 10 (address in 3 bits)

01 5 →
01 6 → 8
01 7 → 9

00010
↓
0110 freedom given

Code → 01001 | 01100

Further compression
can be done

6th Feb

Disk

P: 421739 Q: 814263

R: 381436 S: 940629

P: abc Q: def R: ghi S: jkl

RAM:

v x y z p q r s t u

v: 0 - invalid

v: 1 → disk to ram

x: 1 → 1st variable updated

y: 1 → 2nd variable updated

z: 1 → 3rd variable "

Instruction	P 0000123456	Q 0000862913	R 0000814213	S 0000318613	Disk operation
Print(d)		1000814263			Q: Disk → ram 814263 D → R
h = 83			0010818313		
b = 12	0010121256				
Print(g)			1010388336		R: Disk → ram 381436 D → R
f = 69		1001814269			
OVER	421739 D → R 421239 R → D	814269 Ram → disk	388336 ram → disk		

Instruction	P 0000 59 12 16	Q 0000 61 23 84	R 0000 81 82 13	S 0000 38 14 16
f=29		00016123 29		
i=61			00181826 1	
Print (a)	1000 42 17 39			
Print (e)		10018142 29		
b=19	10104219 39			
d=61		11016142 29		
g=69			01016982 61	
	421939 R→D	614229 R→D	381436 D→R 69 14 61 R→D	

V=0/1 xyz ==0 : nothing

V=1 x/y/z !=0 : R→D

V=0 x/y/z !=0 D→R, updates, R→D

x=y=z=1

0 0000 : 0

4 0001 : 1 {3}

3 0010 : 2 {2}

7 0011 : 3 {3,2}

2 0100 : 4 {1}

6 0101 : 5 {1, 3}

5 0110 : 6 {1,2}

8 0111 : 7 {1,2,3}

1000 : 8

8 1001 : 7

8 1010 : 7

8 1011 : 7

8 1100 : 7

8 1101 : 7

8 1110 : 7

8 1111 : 7

Instruction	P 0123456	Q 0421318	R 0312316	S 0818234	Disk operation
a = d+g	2193456	1814263	1381436		Q: D→R R: D→R
e = a+5		8812463			
k = a+5				3812434	
j = (h++)+k			8381536	5382434	
K = b-3	8191739			5381434	P: D→R
OVER	191739 R→D	81 2463 R→D	3815 36 R→D	941029 D→R 381429 R→D	

Instruction	P 0123456	Q 0421318	R 0312316	S 0818234	Disk operation
a = d+g	2193456	1814263	1381436		Q: D→R R: D→R
e = a+5		8812463			
k = a+5				3812434	
j = (h++)+k			8381536	8382429	S: D→R
k = b-3	8191739			8381429	
OVGR	191739 R→D	812463 R→D	381536 R→D	381429 R→D	

during updation of 2nd variable in a page make it consistent with disk

000 : garbage

001 } Partial update of only one variable without D-> R

110 : valid D->R and Ram==disk

111 : dirty at the end we need to do ram to disk

U: a b c d e
4 1 3 6 1

V: f g h i j
8 1 4 1 9

W: k l m n o
3 4 5 7 9

Instruction	U 000.81429	V 000 31461	W 000 84423		Disk operation
Print (9)		110 81419			V: D → R
m=6			011 84623		
a=4	001 41429				
k=7			111 74679		W: D → R

09/02/2023

Round Robin Scheduling

A job is given Δ time. If not over then another job is given Δ time. It is done in rotation.

> Wait = Finish - Ready - Service

RR Quantum 10

Job	Ready	Service	Interval
A	5	22	5-15, 25-35, 39-41
B	7	14	15-25, 35-39

Wait(A) 10+4 = 14

Wait(B) 8+10 = 18

Feedback

Ex2: Multi level Queue

Should be

Interval	Job	Ready	Service	RR Interval
5-15, 30-40, 44-46	A	5	22	5-15, 25-35, 44-46
15-25, 40-44	B	7	14	15-25, 40-44
25-30	C	19	5	35-40

at t =	Queue
14	B_1
15	A_2
19	$A_2 C_1$

MLFQ (Multi Level Feedback Queue)

Similar to RR but previous job is put at the end of low priority queue. [New job is given more priority]
↳ pushed into (HPQ)

① Low & High Priority Queue (HPQ)
② Less Quantum in High Priority
③ High Priority Job can preempt a low priority job.

Quantum 5/20

wt	Job	Arrival	Service	Interval
40	A	0	60	0-5, 10-30, 40-60, 85-100
2	B	3	5	5-10
61	C	14	30	30-35, 60-80, 100-105
58	D	17	10	35-40, 80-85,

* Preemption

Interval	Job	wt
0-5, 10-14, 24-44, 69-85, 90-105	A	105-0-60 = 45
5-10	B	10-3-5 = 2
14-19, 44-64, 85-90	C	90-14-30 = 50
19-24, 64-69	D	69-17-10 = 42

Ex:

Job	Service	Ready	FCFS Interval	SRTN ~~MLFQ~~ Interval	Context Switch
A					
B					

* No High priority job can preempt a low priority job.

Job	Arrival	Service	Interval
A	0	80	0-5, 10-14, 24-44, 49-89, 94-105
B	3	5	5-10
C	14	30	14-19, 44-64, 89-94
D	17	10	19-24, 64-69

*

Job	Service	Ready	Interval FCFS	Interval SRTN	Context switch: FCFS	Context switch: SRTN
A	50	0	0-50	0-10, 20-60	0	10
B	5	10	50-55	10-15	40	0
C	100	55			0	5
D	100	155			0	5
E	100	205			0	5