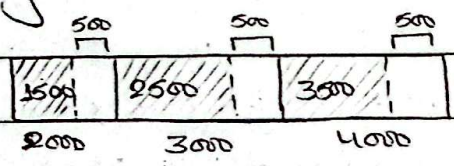
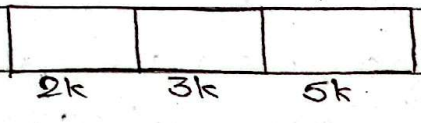
Assignment 3 [OS]

Page:
Date: / /

1. Explain the difference between internal & external fragmentation.

⇒

S.N	Internal Fragmentation	External Fragmentation
1.	It is the unused space within allocated memory blocks due to fixed block size allocation.	It is the unused memory space between allocated memory blocks that arise due to variable block sizes.
2.	It is caused when memory is allocated in blocks larger than required, leading to waste space within a block.	It is caused when memory is allocated in a way that leaves small gaps between blocks, which can't be used.
3.	It occurs inside an allocated memory block.	It occurs outside the allocated memory blocks, in free memory regions.
4.	It causes the wastage of memory inside the allocated areas.	It causes the wastage of memory in the free space between allocated areas.
5.	It is common in fixed-partition memory management or paging systems.	It is common in variable-partation memory management or segmentation systems.
6.	eg:- 	eg:- 
	Here, the 500 memory space left is internal fragmentation.	Here, the 2k, 3k & 5k ie: the allocated memory is the external fragmentation.

Page :
Date : / /

5. What is the purpose of paging the page tables?

⇒ The purpose of paging the page table is to efficiently manage memory in a virtual memory system, especially when the page table itself becomes too large to fit in a single, continuous block of physical memory.

Purpose of paging the page table are:-

* Efficient Use of Memory
The page table can be very large for processes with large address spaces. So, paging allows page table to be divided into smaller, more manageable chunks.

* Handling large Page Tables
Paging the page table helps in ~~mana~~ handling the large number of pages by breaking it into smaller parts.

* On-Demand loading of Page Table Entries
Only parts of the page table need to be loaded into memory, reducing the overhead of storing the entire table in physical memory at all times.

* Hierarchial Page Tables
Paging the page table leads to the creation of multi-level page tables, where each level is a page ~~its~~ table itself. This hierarchial structure reduces the memory overhead associated with storing large, flat page tables.

Page:
Date: / /

Scalability

It allows the virtual memory system to scale efficiently for processes with very large address spaces, supporting modern applications without requiring excessive physical memory.

What is the cause of thrashing? How does the system detect thrashing? Once it detects thrashing, what can the system do to eliminate this problem?

Thrashing occurs when a system spends more time swapping pages in & out of memory than executing actual process. This caused by:-

Insufficient Physical Memory

Too many processes are running with insufficient memory, leading to frequent page replacements.

High Multiprogramming level.

The system lacks enough RAM to hold the working sets of all active processes.

Poor Page Replacement Policies

Inefficient algorithms may lead to replacing pages that are still needed, increasing page faults.

Detecting Thrashing:-

A system can detect thrashing by monitoring:-

High Page Fault Rate

A sharp increase in page faults indicates thrashing.

Page :
Date : / /

ii) CPU Utilization
When the CPU utilization decreases while the page fault rate is high, it suggests the system is thrashing.

iii) Working Set Model
The working set of processes (the set of pages a process actively uses) exceeds the available physical memory.

Eliminating Thrashing :-
To address thrashing, the OS can take the following actions :-

i) Adjust the Degree of Multiprogramming
Reduce the number of active processes by suspending some & swapping them out to disk. This allows remaining processes to access sufficient memory.

ii) Implement Working Set Policy
Use a working set model to ensure that only processes with their entire working set in memory are allowed to execute.

iii) Improve Page Replacement Algorithms
Use better page replacement strategies to reduce unnecessary page replacements.

Page :
Date : / /

iv) Increase Physical Memory
If hardware permits, add more RAM to the system.

v) Dynamic load Control
Dynamically adjust the load on the system by monitoring memory usage & page fault rates to maintain optimal performance.

vi) Priority Adjustment.
Lower the priority of processes causing excessive page faults to give other processes more memory & CPU time.

10. What are the advantages & disadvantages of supporting memory mapped I/O to device control registers?

⇒ Advantages:-

- Unified Addressing
Both memory & device registers share the same address space, simplifying the CPU design since only one set of instructions is needed for memory & I/O operations.

- Ease of Programming
Standard memory-access instructions can be used to interact with device registers, simplifying the development of device drivers.

Page:
Date: / /

- **Efficient Context Switching**
With memory-mapped I/O, there's no need for special instructions or modes to access device registers, reducing overhead during context switches.

- **Hardware Simplification**
Fewer specialized I/O instructions or control mechanisms are required, as the CPU accesses device registers just like regular memory.

- **Compatibility with Caching**
Device registers can leverage caching mechanisms, improving performance in specific scenarios, like repeated accesses to the same register.

- **Flexibility in System Design**
The system designer has greater flexibility in assigning memory addresses to device registers, especially in systems with sparse address spaces.

Disadvantages:

- **Conflict with Main Memory**
Memory-mapped I/O reduces the addressable memory available for the system's RAM, especially in systems with limited address space (eg: 32-bit systems).

Page:
Date: / /

- **Caching Issues**
  Cache mechanisms might inadvertently cache device registers, leading to stable stable or incorrect data. Special handling is required to avoid this.

- **Performance Overhead**
  Marking regions as non-cacheable or disabling certain optimizations for I/O registers can impact performance.

- **Complexity in Address Space Management**
  Allocating unique memory addresses for devices & ensuring no overlaps can become challenging in complex systems with many devices.

- **Device Access Timing Constraints**
  Some devices have strict timing requirements that may not align well with the memory access patterns of modern CPUs.

- **Probability Issues**
  Code that relies on memory-mapped I/O may need modifications when ported to systems with different address spaces or I/O configurations.

- **Security Risks**
  If memory-mapped regions are not properly protected, malicious programs could directly access & manipulate device registers.

Page :
Date : /. /

11. Explain the difference between deadlock, livelock & starvation.

⇒

Aspect	Deadlock	Livelock	Starvation
Definition	Processes are blocked indefinitely, waiting for resources held by each other in a circular dependency.	Processes continuously change state in response to eachother without making progress.	A process is perpetually denied access to resources due to unfair scheduling or resource allocation.
State	Processes are blocked & don't perform any action.	Processes are active but do not make progress.	Other processes make progress, but one process is indefinetly delayed.
Cause	Circular dependency on resources.	Continuous retries or conflict resolution with no resolution.	Unfair resource allocation or scheduling.
Activity	No activity, processes are stuck.	Constant activity without progress.	Other processes progress, but the affected process doesn't.
Resolution	Deadlock prevention (ie: avoid circular wait), detection, or recovery.	Introduce randomness or thresholds to break repetitive behavior.	Use fair scheduling (eg: round-robin) or aging to increase priority over time.

Page:
Date: / /

12. Differentiate between segmentation & paging.

⇒

S.N	Segmentation	Paging
1.	It divides the memory into variable-sized segments based on the logical divisions of a program, such as functions, arrays or data blocks.	It divides the memory into fixed-sized blocks called pages, which correspond to fixed sized frames in physical memory.
2.	Memory division is done in variable-sized segments.	Memory division is done in fixed-sized pages.
3.	logical address consists of a segment number & offset.	logical address consists of a page number & page offset.
4.	Segment table maps segment numbers to physical memory locations.	Page table maps page numbers to physical memory frames.
5.	Suffers from external fragmentation due to variable segment sizes.	Suffers from internal fragmentation if processes don't fully utilize a page.
6.	It allows the logical division of a program, making it more intuitive & easier to handle.	Fixed-size blocks make memory management simpler but less intuitive.
7.	More complex due to variable sizes & the need to handle external fragmentation.	Easier to implement because of uniform page sizes & no external fragmentation.
8.	eg:- A segment might represent a function array or stack.	eg:- A page is a fixed-sized block of memory ie: 4KB.

Page :
Date : / /

13. Explain Producer Consumer Problem using Semaphore.

OS Assignment 3

PAGE NO.

2. Given five memory partitions of 100 KB, 500 KB, 200 KB, 300 KB & 600 KB (in order), how would each of the first-fit, best-fit & worst-fit algorithms place processes of 212 KB, 417 KB, 112 KB & 426 KB (in order)? Which algorithm makes the most efficient use of memory?

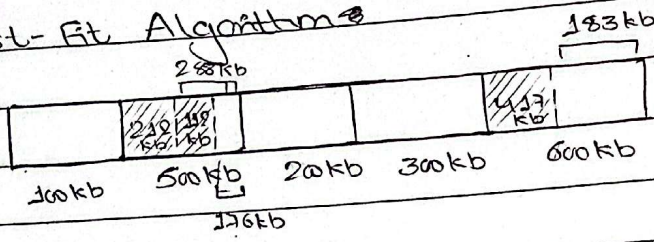
Soln-

Given that;

Memory partitions :- 100 kb, 500 kb, 200 kb, 300 kb, 600 kb

Process size :- 212 kb, 417 kb, 112 kb, 426 kb

1) First-Fit Algorithm

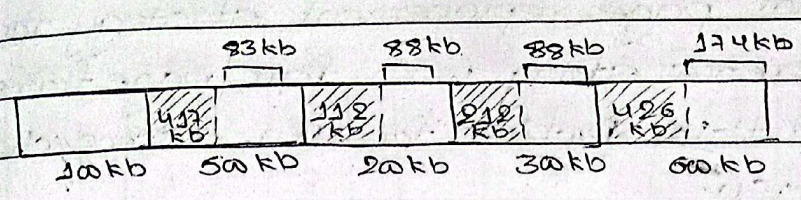


- 212 kb is placed in 500kb.
  ∴ Remaining size = 288 kb
- 417 kb is placed in 600 kb.
  ∴ RS = 183 kb
- 112 kb is placed in 288 kb.
  ∴ RS = 176 kb
- 426 kb must wait.

∴ Total fragmentation = 176 + 183 = 359 kb

Raghukul
DATE / /
PAGE NO.

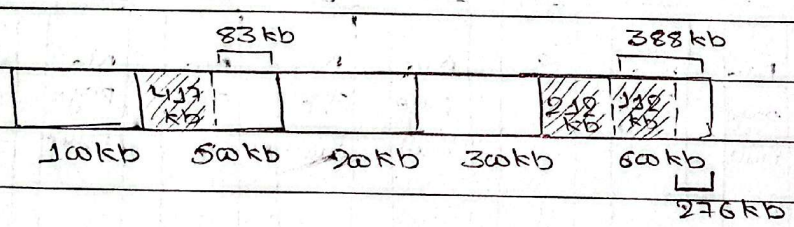
(i) Best fit Algorithm



- 212 kb is placed in 300 kb
  ∴ RS = 88 kb
- 417 kb is placed in 500 kb
  ∴ RS = 83 kb
- 112 kb is placed in 200 kb
  ∴ RS = 88 kb
- 426 kb is placed in 600 kb
  ∴ RS = 174 kb

∴ Total fragmentation = 83 kb + 88 kb + 88 kb + 174 kb
= 433 kb

(ii) Worst fit Algorithm



- 212 kb is placed in 600 kb
  ∴ RS = 388 kb
- 417 kb is placed in 500 kb
  ∴ RS = 83 kb
- 112 kb is placed in 388 kb
  ∴ RS = 276 kb
- 426 kb must wait.

∴ Total fragmentation = 83 + 276 kb = 356 kb

∴ Best fit algorithm makes the most efficient use of memory.

Raghukul
DATE / /
PAGE NO.

6. What is the minimum number of page faults for an optimal page replacement strategy for the following reference string with four page frames? Also repeat this problem for LRU, second chance, FIFO, ~~ARU~~, MFU.

1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2

Soln:-

Given that:-

Page frames = 4

Optimal Page Replacement:-

1	2	3	4	5	3	4	1	6
1	1	1	1	1	No page fault	No page fault	No page fault	6
	2	2	2	5				5
		3	3	3				3
			4	4				4

7	8	7	8	9	7	8	9	5	4
6	8	No page fault	No page fault	8	No page fault	No page fault	No page fault	No page fault	8
5	5			5					5
7	7			7					4
4	4			9					9

5	4	2
No page fault	No page fault	2
		5
		4
		9

∴ Total page fault = 11

∴ Total hit = 11

LRU (Least recently use):-

1	2	3	4	5	3	4	1	6	7
1	1	1	1	5	No page fault	No page fault	5	6	6
	2	2	2	2			1	1	1
		3	3	3			3	3	7
			4	4			4	4	4

8	7	8	9	7	8	9	5	4	5
6	No page fault	No page fault	6	No page fault	No page fault	No page fault	5	5	No page fault
1			9				9	9	
7			7				7	4	
8			8				8	8	

4	2
No page fault	5
	9
	4
	2

∴ Total page fault = 13

∴ Total hit = 9

FIFO [First-In-First-Out]:-

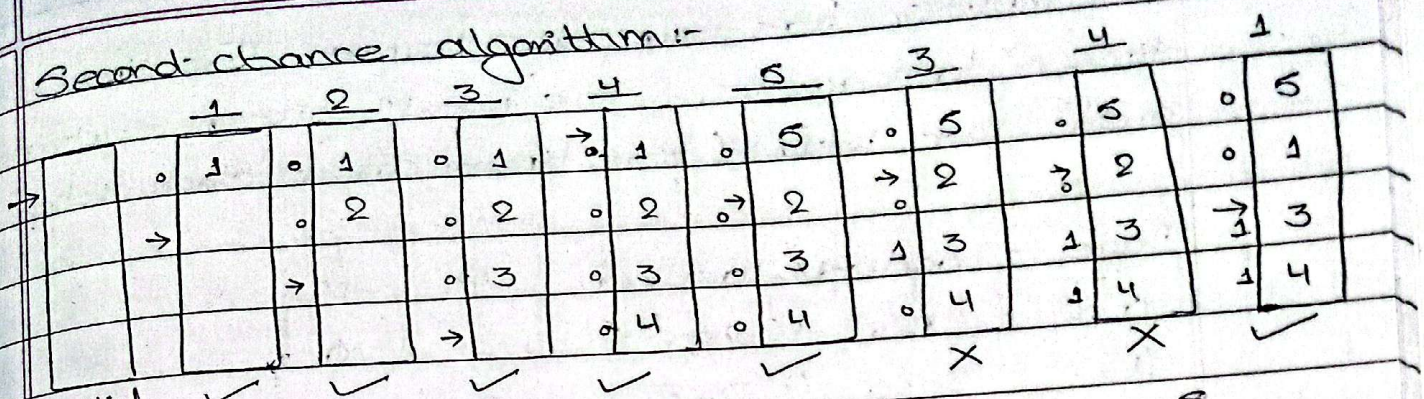
1	2	3	4	5	3	4	1	6	7	8	7
1	1	1	1	5	No page fault	No page fault	5	5	5	8	No page fault
	2	2	2	2			1	1	1	1	
		3	3	3			3	6	6	6	
			4	4			4	4	7	7	

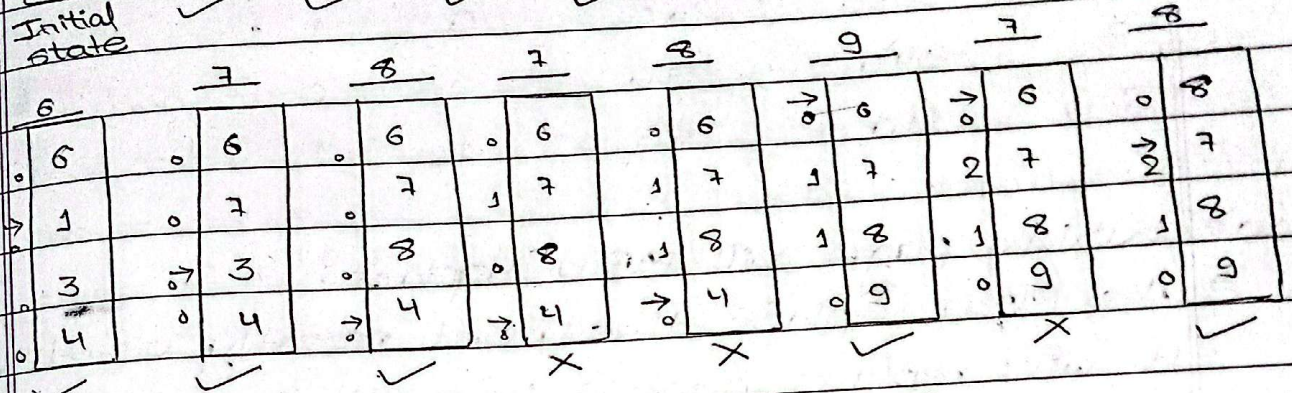
8	9	7	8	9	5	4	5	4	2
No page fault	8	No page fault	No page fault	8	8	8	No page fault	No page fault	2
	9			9	9	9			9
	6			No page fault	5	5			5
	7				7	4			4

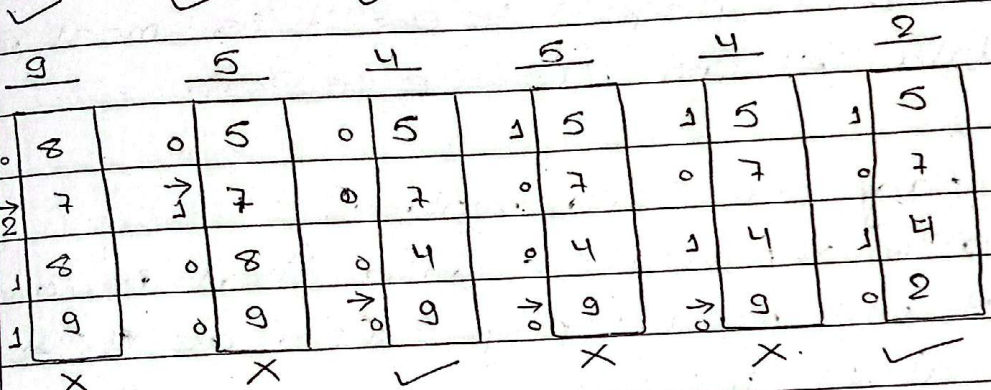
∴ Total page fault = 13

∴ Total hit = 9

Second chance algorithm:-







Total page fault = 13

Total hit = 9

MFU [Most frequently used]:-

1	2	3	4	5	3	4	1	6	7
1	1	1	1	5	No page fault	No page fault	5	5	5
	2	2	2	2			2	2	2
		3	3	3			1	1	7
			4	4			4	6	6

8	7	8	9	7	8	9	5	4	5	4	2
5	No page fault	No page fault	5	5	5	No page fault	No page fault	5	No page fault	No page fault	2
8			8	7	8			4			4
7			9	9	9			9			9
6			6	6	6			6			6

Total page fault = 14

Total hit = 8

RS	Counter
1	12
2	12
3	12
4	1234
5	123
6	1
7	123
8	123
9	12

Raghukul
DATE / /
PAGE No.

8. A computer has four page frames. The time of loading, time of last access, & the R & M bits of each page are as shown below (the times are in clock ticks):

Page	Loaded	last ref.	R	M
0	126	280	1	0
1	230	265	0	1
2	140	270	0	0
3	110	285	1	1

a) Which page will MRU replace?

⇒ As MRU replace the most recently used data, it will replace page 3 as it is most recently used data ie: last reference is 285.

b) Which page will FIFO replace?

⇒ As FIFO replace the data that is loaded first, it will replace page 3 (110) as it is loaded at 1st ie: 110.

c) Which page will LRU replace?

⇒ As LRU replace the data that is least recently used, it will replace page 1 as it is least recently used ie: last reference is 265.

d) Which page will second chance replace?

⇒ FIFO order :- page 3 → page 0 → page 2 → page 1

Checking R-bit :-

page 3 :- R = 1 → skip

page 0 :- R = 1 → skip

page 2 :- R = 0 → replaced

∴ page 2 will be replaced.

DATE / /
PAGE NO.

9. Suppose that a disk drive has 5000 cylinder numbered 0 to 4999. The drive is currently serving a request at cylinder 143, & the previous request was at cylinder 125. The queue of pending request, in FIFO order is:

86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests, for each of the following disk scheduling algorithms?

a) FCFS b) SSTF c) SCAN d) C-SCAN e) LOOK f) C-LOOK

Soln

Given that,

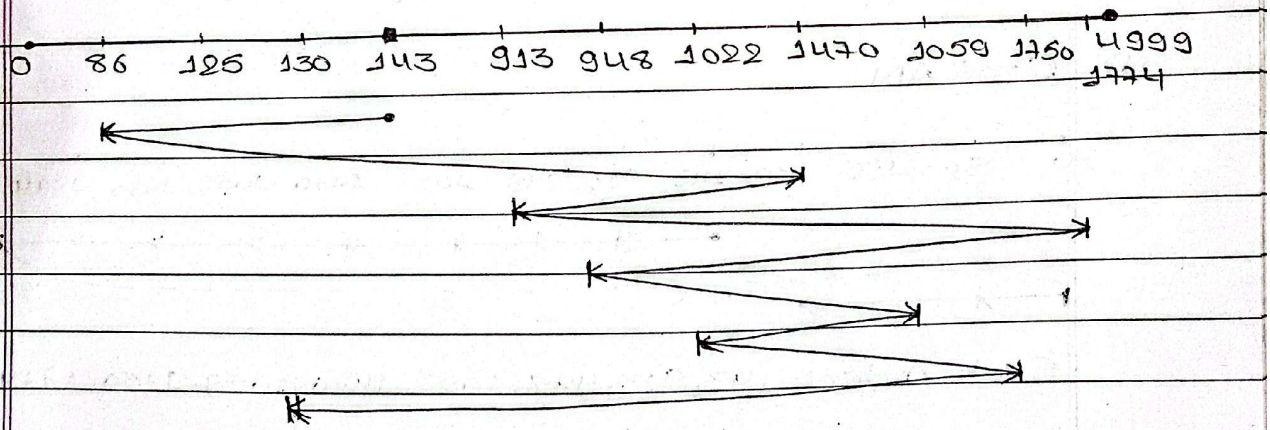
Disk size = 5000 cylinder

Request Queue = 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

Current pointer = 143

Previous pointer = 125

a) FCFS



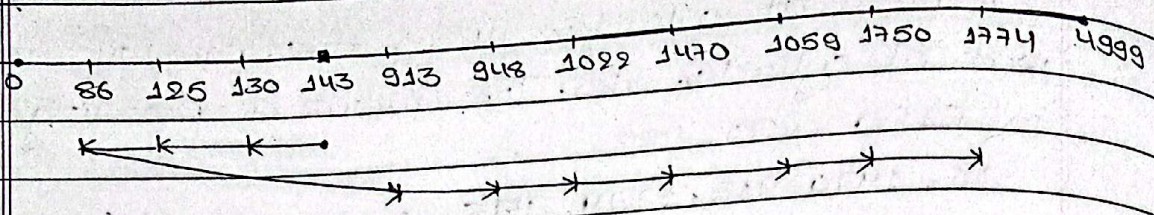
Seek order:- 143, 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

Total seek distance:- |143-86| + |86-1470| + |1470-913| + |913-1774| + |1774-948| + |948-1509| + |1509-1022| + |1022-1750| + |1750-130|

= 7086 cylinders

Raghukul
DATE / /
PAGE No.

b) SSTF

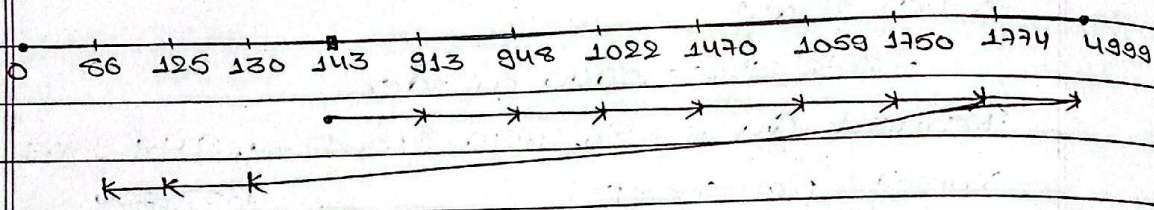


Seek order :- 143, 130, 125, 86, 913, 948, 1022, 1470, 1059, 1750, 1774,

Total seek distance :- |143-86|+|86-1774|

= 1745 cylinders

c) SCAN

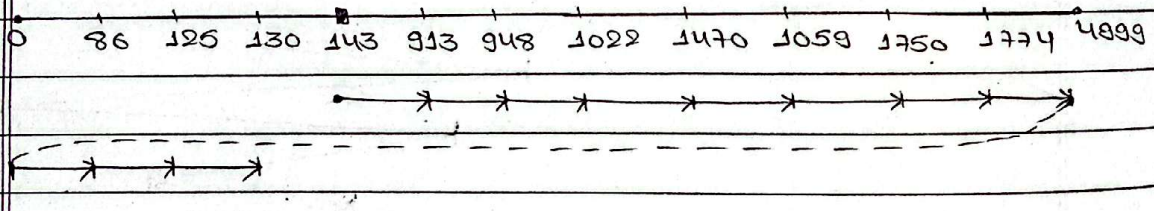


Seek order :- 143, 913, 948, 1022, 1470, 1059, 1750, 1774, 4999, 130, 125, 86

Total seek distance :- |143-4999|+|4999-86|

= 9769 cylinders

d) C-SCAN



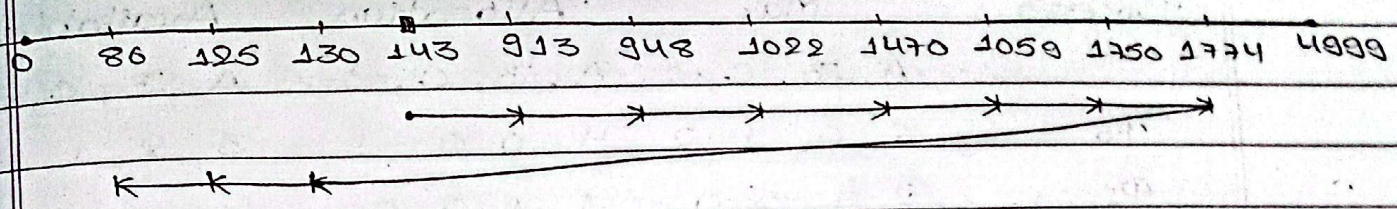
Seek order :- 143, 913, 948, 1022, 1470, 1059, 1750, 1774, 4999, 0, 86, 125, 130

Total seek distance :- |143-4999|+|4999-0|+|0-130|

= 9985 cylinders

Raghukul
DATE / /
PAGE NO.

e) Look

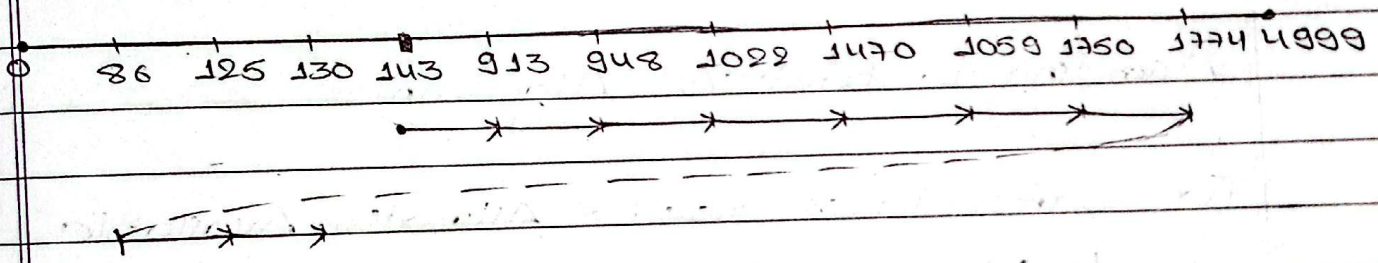


Seek Order :- 143, 913, 948, 1022, 1470, 1059, 1750, 1774, 130, 125, 86

Total Seek distance = |143 - 1774| + |1774 - 86|

= 3319 cylinders.

f) C-Look



Seek Order :- 143, 913, 948, 1022, 1470, 1059, 1750, 1774, 86, 125, 130.

Total Seek distance :- |143 - 1774| + |1774 - 86| + |86 - 130|

= 3363 cylinders.

Raghukul
DATE / /
PAGE NO.

14. Consider following snapshot:

Process	Max	Allocation	Available
	A, B, C, D	A, B, C, D	A, B, C, D
P_0	6 0 1 2	4 0 0 1	3 2 1 1
P_1	2 7 5 0	1 1 0 0	
P_2	2 3 5 6	1 2 5 4	
P_3	1 6 5 3	0 6 3 3	
P_4	1 6 5 6	0 2 1 2	

Is this a safe state? If yes, what is safe sequence?

Soln

According to Banker's Algorithm:-

For safe state:- need ≤ ~~Allocation~~ Available

Need = Max - Allocation

ie: Need Matrix = Max - Allocation

Need Matrix:

	A	B	C	D
P_0	2	0	1	1
P_1	1	6	5	0
P_2	1	1	0	2
P_3	1	0	2	0
P_4	1	4	4	4

Here, the need of P_0 is ≤ available. So, we can run P_0 process.

Run P_0:- After completion, it releases its allocated resources.

∴ New available = Allocation of P_0 + available.

= (4, 0, 0, 1) + (3, 2, 1, 1)

= (7, 2, 1, 2)

Raghukul
DATE / /
PAGE NO.

The need of P_2 is $\leq$ available. So, we can run P_2 process.

Run P_2 :- After completion, it releases its allocated resources.

∴ New available = Allocation of P_2 + available
= (1, 2, 5, 4) + (7, 2, 1, 2)
= (8, 4, 6, 6)

The need of P_3 & P_4 ~~are~~ is $\leq$ available. So, we can run both P_3 & P_4 process.

Run P_4 :- After completion, it releases its allocated resources.

∴ New available = (0, 2, 1, 2) + (8, 4, 6, 6)
= (8, 6, 7, 8)

The need of both P_1 & P_3 is $\leq$ available. So, we can run both ~~be~~ process.

Run P_3 :- After completion.

∴ New available = (0, 6, 3, 3) + (8, 6, 7, 8)
= (8, 12, 10, 11)

Run P_1 :- After completion.

∴ New available = (1, 1, 0, 0) + (8, 12, 10, 11)
= (9, 13, 10, 11)

∴ Yes the system is in safe state because all system ca are safe. One of the safe sequence is :- P_0, P_2, P_4, P_3, P_1.