PROCESS CONTROL MANAGEMENT and CPU SCHEDULING

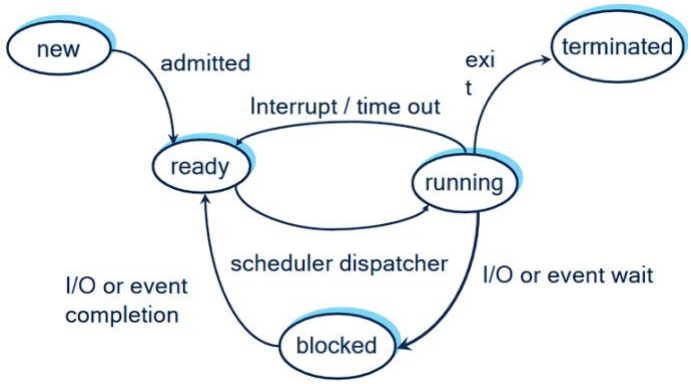
What is Process?

- Program instruction in execution.

Process State Diagram

New, Ready , Running, Terminated, Blocked.

- New to Ready: A process has just been created and is waiting for CPU allocation.
- Ready to Running: The process is assigned to a processor.
- Running to Blocked: The process is waiting for an I/O operation to complete.
- Blocked to Ready: The event the process was waiting for has occurred.
- Running to Ready: The process has reached its maximum allowable time for uninterrupted execution or needs a resource not immediately available.
- Running to Terminated: The process has completed or been aborted.



Multitasking and Multiprocessing

- Multitasking include single CPU rapidly switching between multiple tasks
- Multiprocessing involves multiple CPUs working in parallel. (Cores)

PCB(Process Control Block)

- a data structure used by the operating system to keep track of each process.

- Process state, program counter, CPU registers, memory limits, list of open files, process priority, and accounting information.

What is process scheduling, and why is it important?

- Deciding which process gets to use the CPU.
- For ensuring efficient CPU utilization and responsiveness.

Why CPU scheduling is important?

- To maximize the utilities. (one short sentence)
- Fairness (fair share of CPU)
- Efficiency (maximise CPU utilization)
- Response time (minimize)

What is the role of the scheduler dispatcher?

- allocates CPU time to processes in the Ready state, transitioning them to the Running state.

Concurrent Processes

- Independent (like Time count in Mario game)
- Cooperation (depending and work together with other processes)

What is process forking or spawning?

- the creation of a new process (child) from an existing process (parent).

What is the difference between a process and a thread?

- Process is a unit of work in execution

  Eg: Each browser tab is a separate process, isolated from others

- Thread is a mini lightweight process that can execute independently of other parts of the process.
  Eg: Threads can handle other processes like rendering , networking and JS script execution within a browser.

Turnaround time and throughput

- Turnaround time is time between submission and job creation
- Throughput is the number of job completed.
- Response time is when the job submitted and getting result.

Preemptive and Non-Preemptive scheduling

- Preemptive (can be interrupted)
- Non-Preeemptive (cannot be interrupted and only after completion)

Starvation

- After processing the first ones, there will be less chance to run the last processes.

Time sharing = Multitasking

Preemptive algorithms

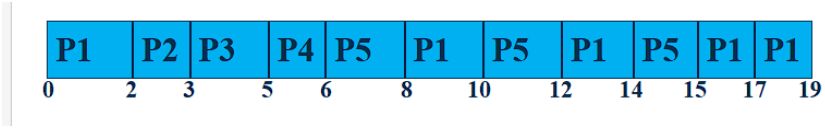
1. Round robing (long waiting time)
2. Multilevel Queue (has own scheduling) (Starvation can happen)
3. Multilevel feedback queue (when there is too much utilities , will move to lower Priority queue)

Non-Preemptive algorithms

1. FIFO

- First Process will be executed. For example, P1, P2,P3, P4....
- Be sure to look at Time slice amount.
- In this one, All the processes will be sliced by 2 mms.
- FIFO is easy because you just need to add by sequencing.
- After the Burst time is finished in one process, add only other unfinished ones.
- For eg: in this calculation, After P2 has finished which is 1 mms burst time, you will only see other undone processes.

Turnaround time = waiting time + burst time



Step 2:- Calculate waiting times and average waiting time

$T_W(P1) = (0+6+2+1)=9$

$T_W(P2) = 2$

$T_W(P3) = 3$

$T_W(P4) = 5$

$T_W(P5) = (6+2+2)=10$

$T_W(average)=(9+2+3+5+10)/5$

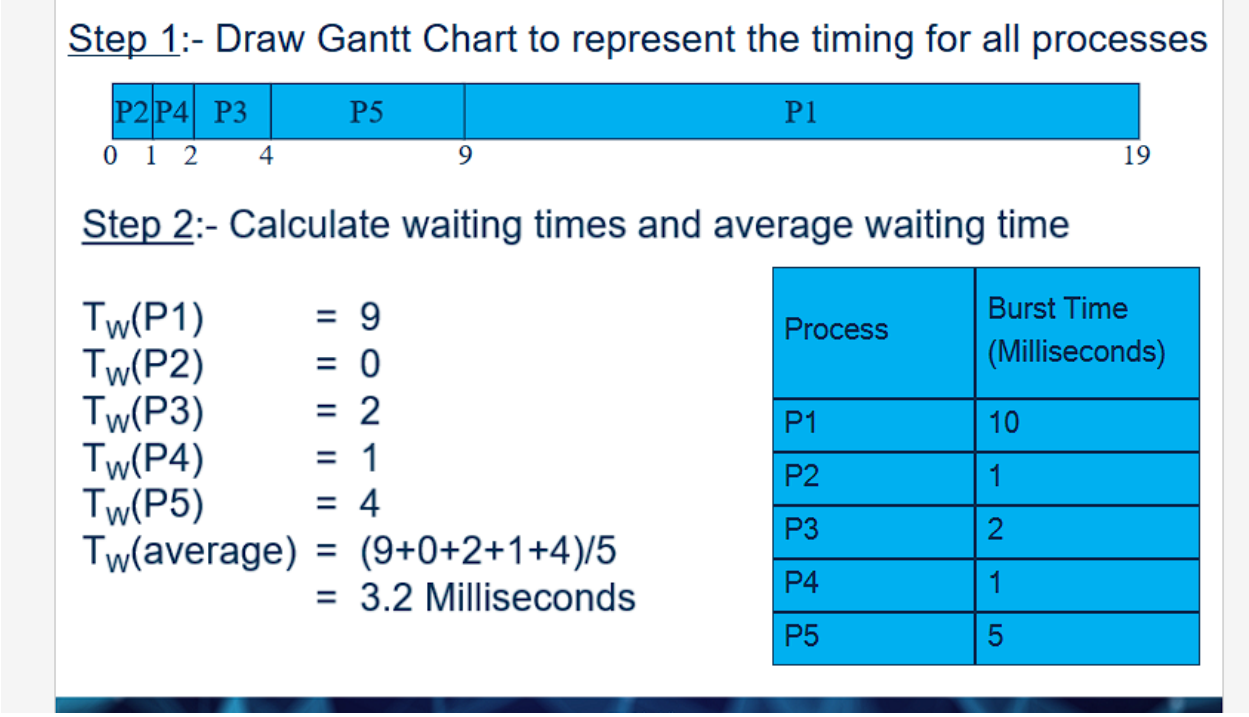
=5.8 Milliseconds

Time Slice = 2 Milliseconds	
Process	Burst Time (Milliseconds)
P1	10
P2	1
P3	2
P4	1
P5	5

Waiting time is just adding the numbers that has reached to the process.

Adv: (easy to implement)
Disadv : (avg waiting time is long)

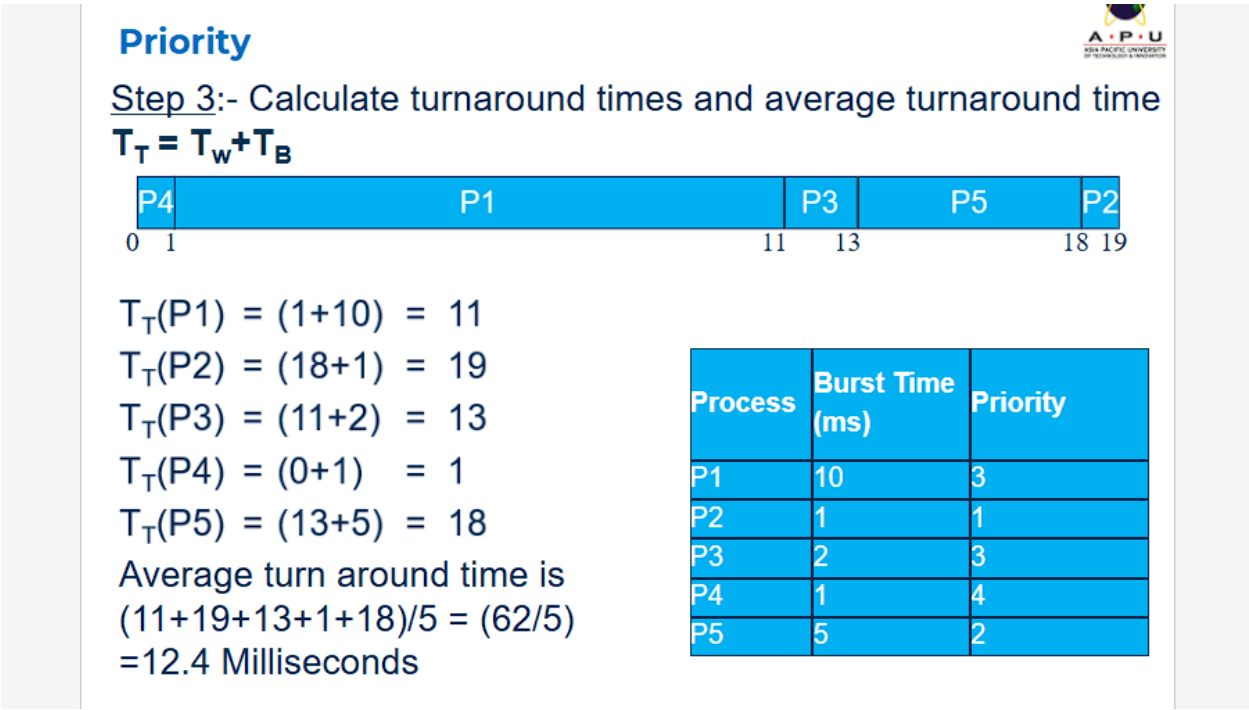
2. SJF
   - Just pick up the lowest Burst time, and add in the first place.
   - then lowest to highest numbers.



Adv: (minimum avg waiting time)
Disadv: (it does not give fairness)

3. Priority

- In priority , you must to choose the greatest priority numbers, and add in the first place.
- Then, P1 will be placed as second because of the First row.



Process	Burst Time (ms)	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

Adv: can be high or low

Disadv : Starvation

Note: In SJF and Priority has no Time slices.

What does the term "aging" refer to in priority scheduling?

- Increasing the priority of older processes

Memory Management and Operation System

1. Partitioning
2. Addressing

Partitaioning

1. Fixed-Partitioning
- No one can use the free space in fixed size
- Using Internal Fragmentation

2. Variable Partitioning
- Using External Fragmentation
- If the Process 1 is finished, the other P can use the space.

Paging – divided into blocks of same size.

Segmentation - Memory management scheme

Logical address is collection of segments.

Demand paging

Adv – Decreases Paging time

- Avoid reading unused pages
- Decreases physical memory needed
- Most efficient

Disadv - May cause overhead due to frequent page faults.

- Can lead to increased latency as pages are loaded on demand
- Requires efficient page replacement algorithms

Demand Segmentation

- Allocate segments instead of pages in memory
- Allows for efficient use of memory by loading variable-sized segments.
- Disadvantages are same as Demand Paging.

Differences between demand Paging and demand Segmentation

Demand Paging loads individual pages of a process into memory only when needed, whereas **Demand Segmentation** loads entire segments of a process into memory only when required and it supports a user view of memory.

Page Fault

When there is no pages that are not in memory. The OS will interrupt and try to use missing page.

(Swap in and Swap out)

What is the result of a page fault?

- The operating system interrupts the process

What is the ideal page-replacement scheme? What is it mainly used for and what is the main disadvantage?

-Optimal Page Replacement

- Disadv : Requires future knowledge of page references

Thrashing

- Spending more paging time than execution.

What is a victim page in page replacement?

- The page to be replaced

What is Belady's anomaly?

Increasing numbers of page frames >>>>>> Increasing numbers of page faults.

First in First out

- Simplest and easiest
- Oldest page is victim.
- Suffers from belady's anomaly

Optimal

- Lowest page faults
- Does not suffer from beladys anaomaly.
- (farthest page)

Least recently used

- Looking backwards
- Last used page is victim

What is the significance of the bootstrap program in an OS?

- for loading the OS into memory during the booting process

Microkernel

- Mostly used in Mac books, in which faults do not directly affects to the kernel and not crashing everything

Monolithic Kernel

- Less reliable and secure because a bug in any part of the kernel can crash the entire system

Hierarchical Model of an OS

- Independent layers and providing a modular and structured approach to OS design

8 Types of OS

1. Single user
2. Single user systems and workstations
3. Mainframe system
4. Network servers
5. Mobile OS
6. Real-Time systems
7. Embedded systems
8. Distributed systems

How do operating systems balance the trade-offs between user interface responsiveness and background process execution?

- priority scheduling, time-sharing, and advanced resource allocation

Disk Management

1. **First Come First Serve(FCFS)**

- **Simple Algorithms**
- **Does not provide Fastest service.**
- **Satisfied in order of arrival**

For example, the queue = 98,132,156,167,56,90,78

Start at 53.

- Draw Line . and calculate by the sequence.
- Always subtract from the greater number.

Then calculate the difference: 98-53 = 45

132-98 = 34

156-132 = 24

167-156 = 11

167-56 = 111

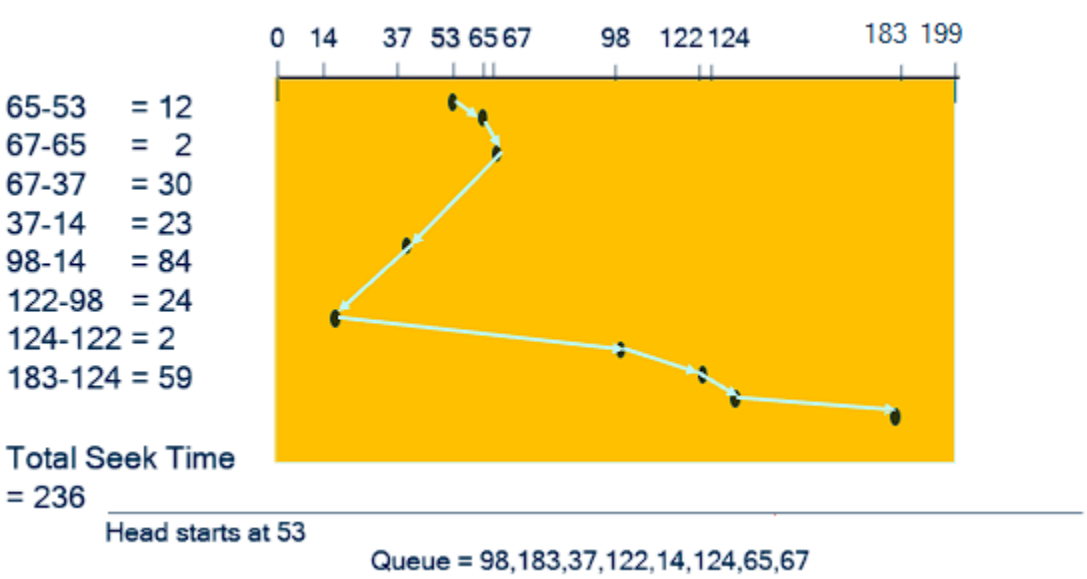
90 – 56 = 34

90 – 78 = 12

Total seek time = 271

Shortest Seek Time First

- **Find the nearest number.**
- **Minimize the seek time.**



The start point is 53.

The Queue = 98,183,37,122,14,124,65,67

Nearest numbers >>> 53>>65>>67>>37>>14>>

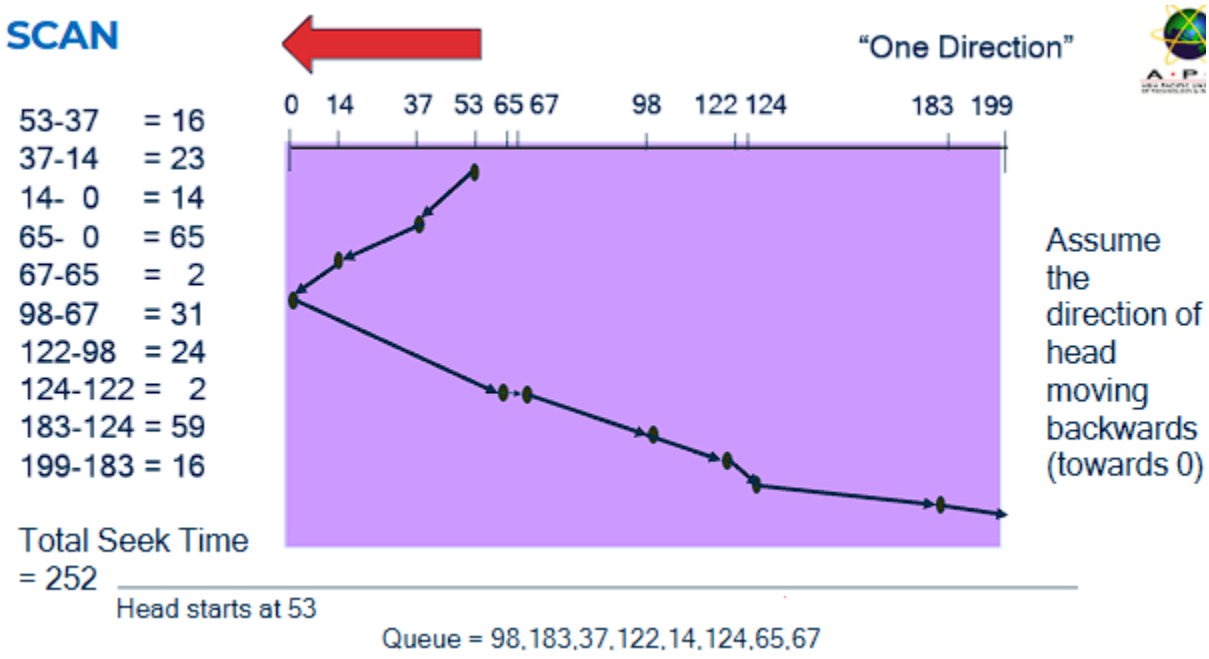
Then , after 14, 98 is the nearest number in highest.

So, 14>>98>>122>>124>>183

Scan

Move the head in one direction, servicing all requests until the end is reached, then reverse direction.

- Go until the other end,
- Based on the Head moving (if the question is saying to move forward >> go to higher numbers)
- (if the question is saying to move backwards >> go to lower numbers.

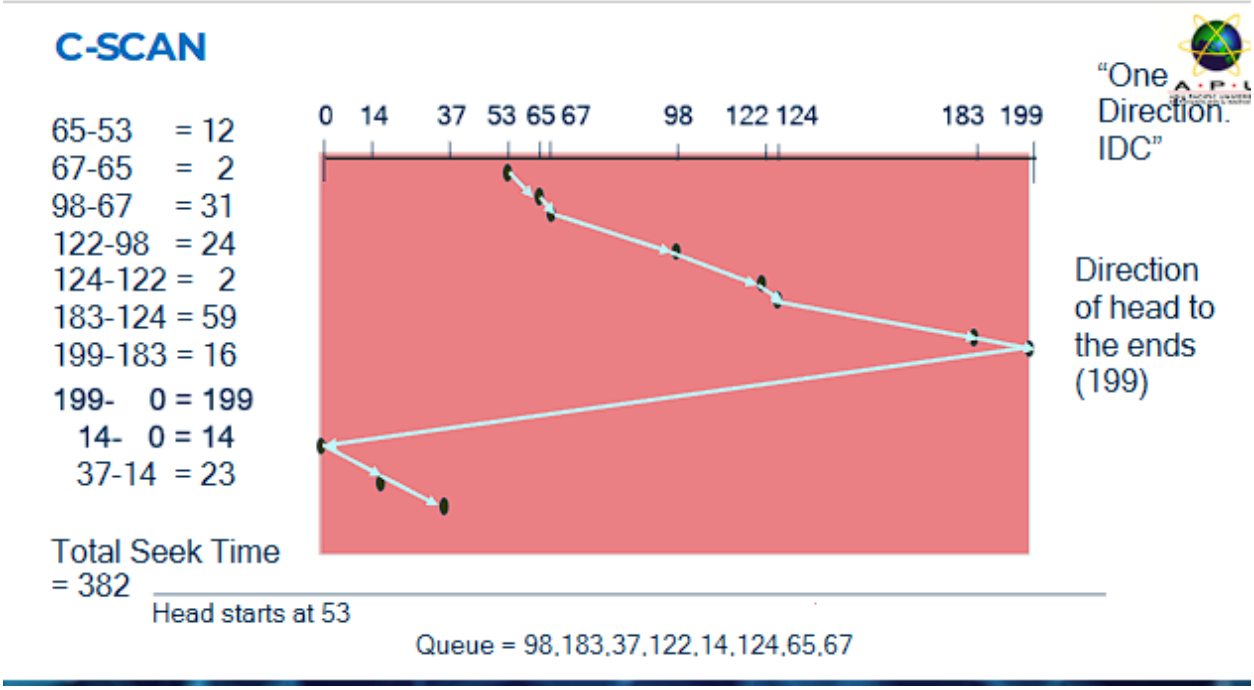


The main difference between C-Scan and Scan is that C-Scan will go to the other side directly whereas , Scan only go to the nearest number (reversing direction) after hitting the end.

In this picture, After hitting "0", it went to "65" . Not directly went to 199.

C-Scan

Once it reaches the end, it immediately returns to the beginning without servicing any requests on the way back.



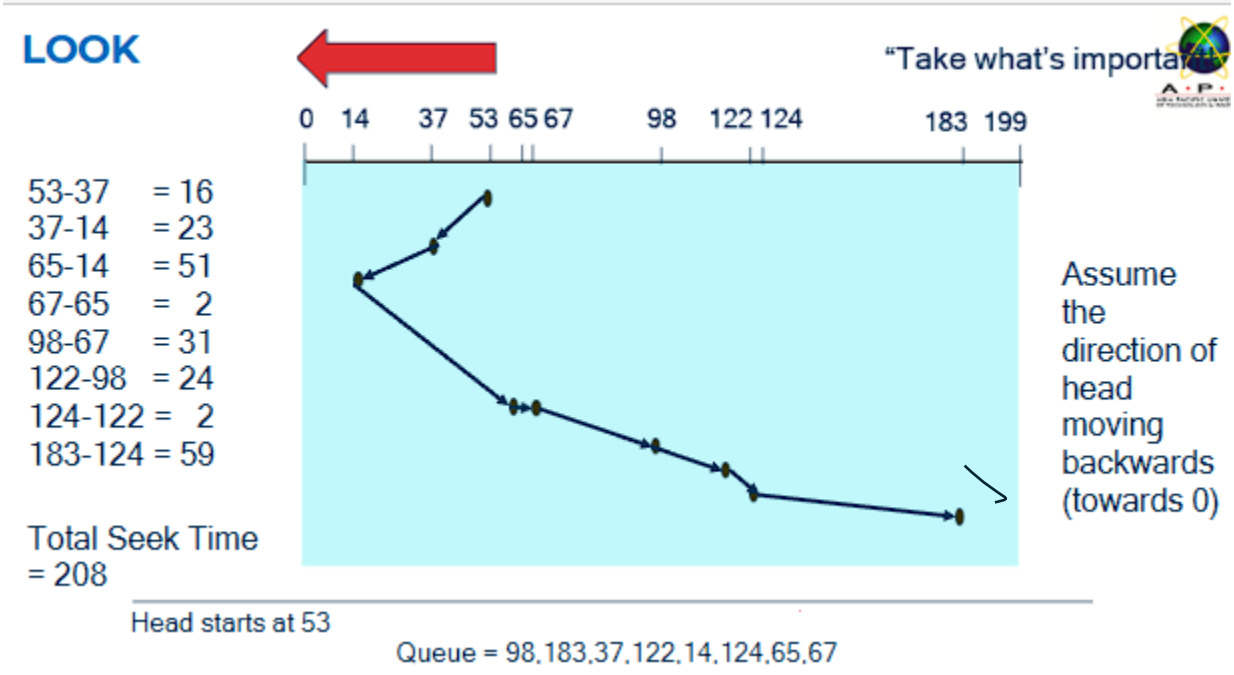
In this C-Scan , the number will hit the end , then didn't go directly to 37. And It went to "0" directly.
This is the main difference.

Look

Similar to SCAN but does not go to the end if there are no requests.

If there is no number request in the queue , it will not hit the end.

Calculate the seek time for all requests in the moving direction, then add the seek time for returning to the start without servicing requests.



C-Look

Similar to SCAN but does not go to the end if there are no requests.

Calculate the seek time for all requests in the moving direction, but stop at the last request instead of the end.

Just only go to the numbers which are included in the queue.

- **Just Substract from largest number "183"**
- **183-53 = 130**
- **183-14 = 169**
- **37-14 = 23**
- **Total seek time = 130+169+23 = 322**