BUCKNELL UNIVERSITY
Computer Science

CSCI 315 Operating Systems Design

Disk Scheduling

Disk Structure

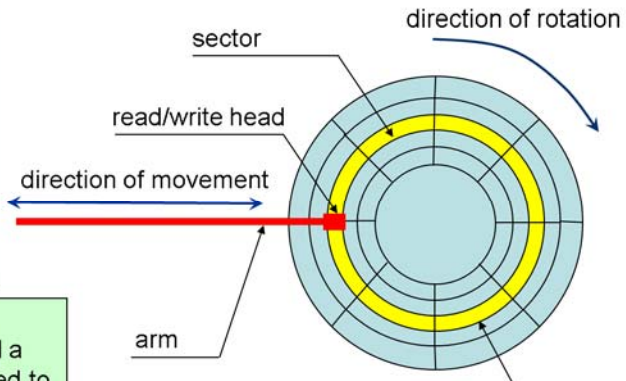
Points to consider:

Sector sizes (number of bits per sector) should be fixed.

The density of the magnetic material is constant on the surface of the disk.

Size of the sector gets smaller as the radius of the track gets smaller.

The disk rotates at a constant speed. To find a block, the head is moved to the appropriate track, and then the correct sector is found as the disk rotates.



Organization of a disk surface

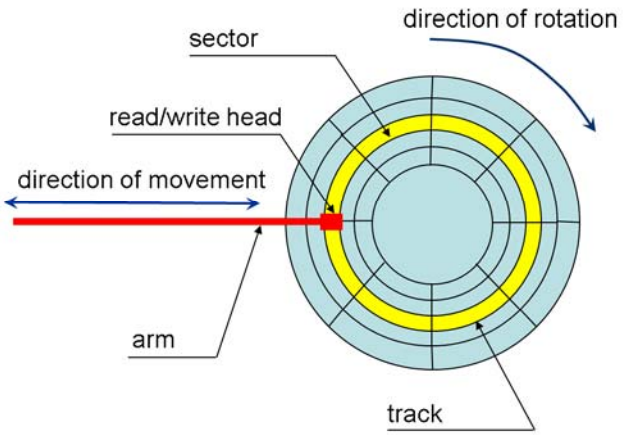
Disk Structure

The disk rotation is given in rotations per minute (RPM).

The time to find a track is proportional to the distance the head must travel.

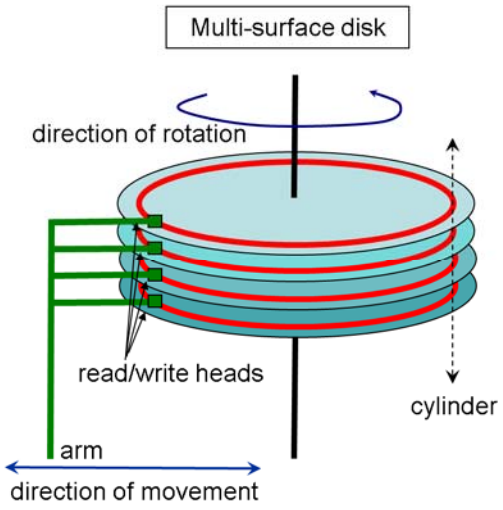
The average time to find a sector within a track is roughly **half the time for a full rotation**.

Question: If the time to move from track i to track $(i+1)$ is given by δ, assuming that the disk head is at track 0 (all the way out), could you calculate the time to get to sector 4 in track 5?



Organization of a disk surface

Disk Structure



A cylinder is the collection of all the same tracks across all the multiple disk surfaces.

There is a time associated with turning heads on and off so that a different surface can be accessed. We call this overhead the **head-switching** time.

The time to move the arm to read another cylinder is due to the mechanics of the arm. It is certainly much large than the head-switching time, which is due to electronics only.

Question: How should one organize data across multiple surfaces to minimize access overhead?

Disk Request

A request must specify:

- Whether the operation is input or output,
- The disk address for the transfer,
- The memory address for the transfer,
- The size of the data block to be transferred (number of bytes).

Disk requests are generated by processes. When the disk is busy serving a request, other incoming requests must be stored for later processing. For this purpose, the OS organizes these requests in a queue of pending requests for that disk unit. When the disk finishes serving a request, the OS much check the queue and if it is not empty, serve another request.

What policy should be used for dealing with the queue?

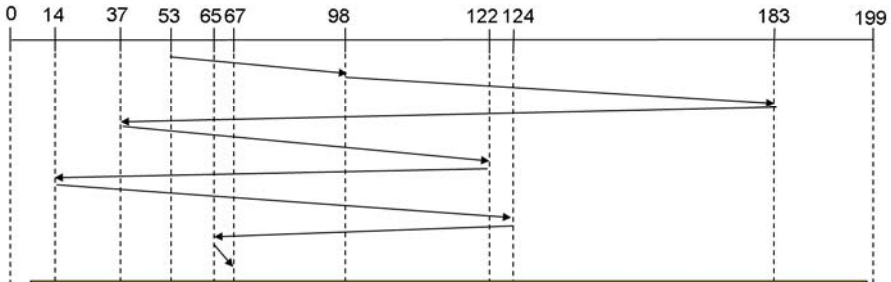
Disk Scheduling

FCFS Scheduling

Consider the following sequence of requests where each number corresponds to a disk cylinder:

98, 183, 37, 122, 14, 124, 65, 67

queue = 98, 183, 37, 122, 14, 124, 65, 67



Question: How much does the disk head travel to serve these requests?

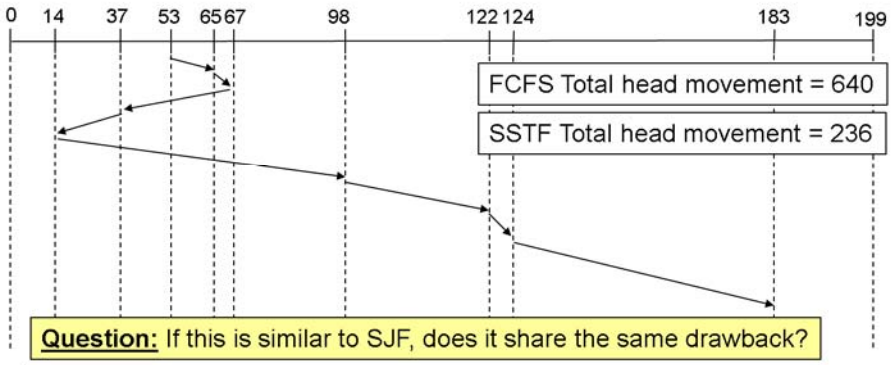
Question: What metric could one define to characterize performance here?

Question: How does this scheduling of requests affect the disk performance?

SSTF Scheduling

98, 183, 37, 122, 14, 124, 65, 67

queue = 65, 67, 37, 14, 98, 122, 124, 183



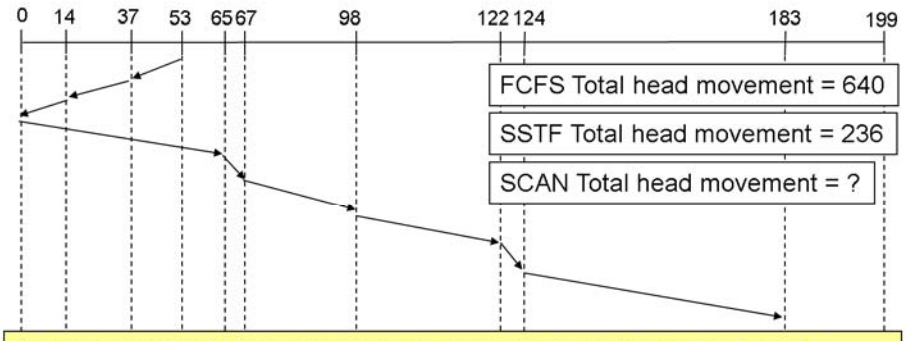
Question: If this is similar to SJF, does it share the same drawback?

Question: Is the performance of SSTF any better than that of FCFS?

Question: Is the performance of SSTF optimal?

SCAN Scheduling

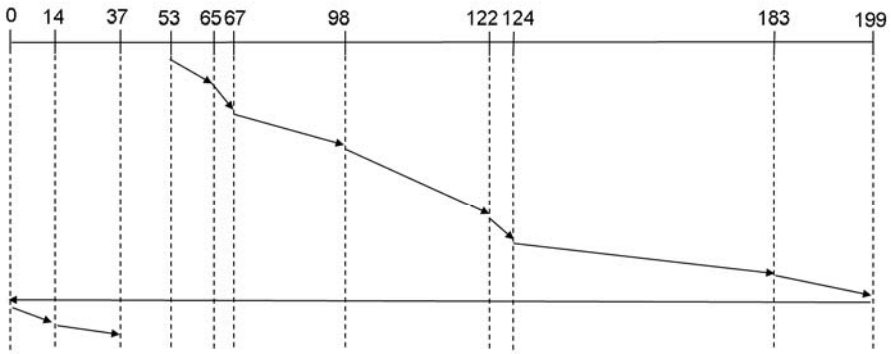
98, 183, 37, 122, 14, 124, 65, 67



Assume the distribution of requests for cylinders is uniform. Consider the density of requests when the head reaches one end and reverses direction. Hmmm… few requests will be right in front of the head… The heaviest density will be at the other end of the disk and those requests will have waited the longest. Why not just go there first?

C-SCAN Scheduling

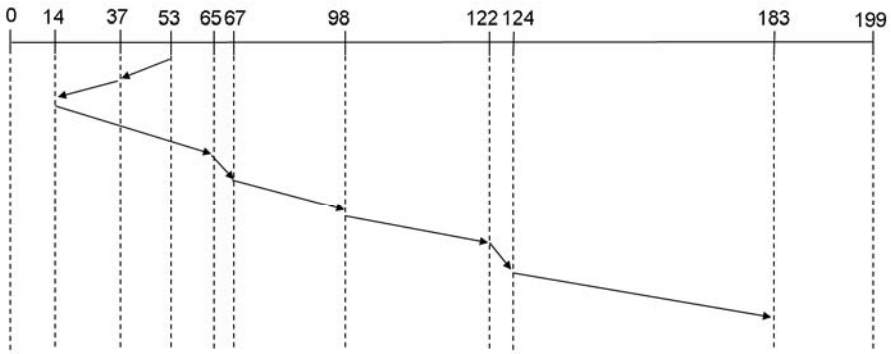
98, 183, 37, 122, 14, 124, 65, 67



When the head reaches the end of the disk, it immediately returns to cylinder 0. The algorithm essentially treats the cylinders as a **circular list** that wraps around from the final cylinder to the first one.

LOOK Scheduling

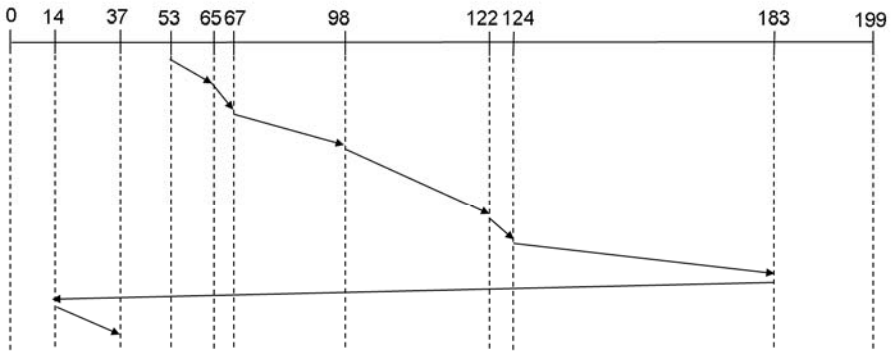
98, 183, 37, 122, 14, 124, 65, 67



Well, by now you have realized that scanning all the way to the extreme ends of the disk is perhaps wasteful. In practice, one could move the head only as far as the request that is farthest out.

C-LOOK Scheduling

98, 183, 37, 122, 14, 124, 65, 67



Choosing a Disk Scheduling Algorithm

- The criteria should involve fairness and performance.
- Performance depends on the number and type of requests.
- Given a string of cylinder references, one could find the **optimal schedule** to serve them all using, for instance, *dynamic programming*. Remember, however, that **computing the optimal schedule takes time!**
- Another point to remember is that disk performance depends heavily on the file allocation method, on the location of directories and indices.
- If disk scheduling is a separate OS module, it can easily be replaced.
- Modern disk units don't disclose the physical location of disk blocks, so it can be challenging to do scheduling in the OS. Disk controllers, however, can handle disk scheduling themselves, lessening the burden on the OS.