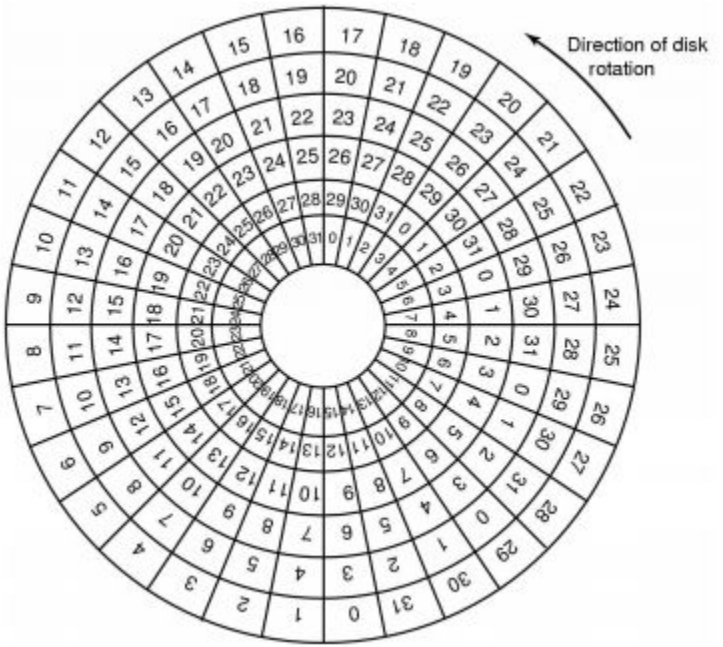
Discussion 4/19/19

Disk Arm Scheduling Algorithms

- Moving the disk arm takes time.
- Disk access' happen all over the disk.
- Servicing each request in the order they arrive is slow! (Why?)
- So, we must choose a scheduling algorithm.



First Come First Serve

- Given the following seeks:

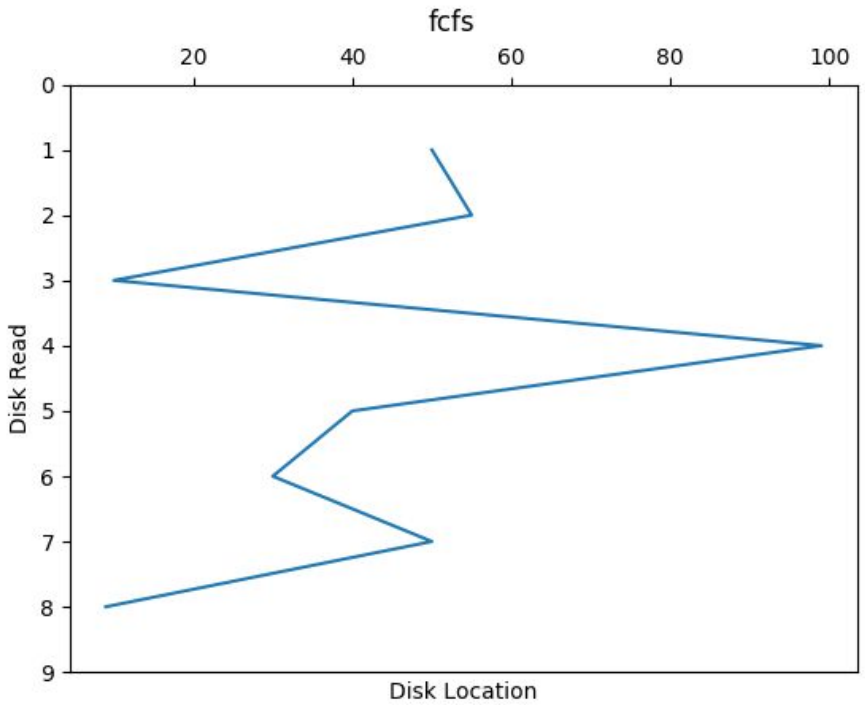
  50 55 10 99 40 30 50 9

- Order is:

  50, 55, 10, 99, 40, 30, 50, 9

- Total:

  5 + 45 + 89 + 59 + 10 + 20 + 41 = 269



Shortest Seek Time First

- Given the following seeks:

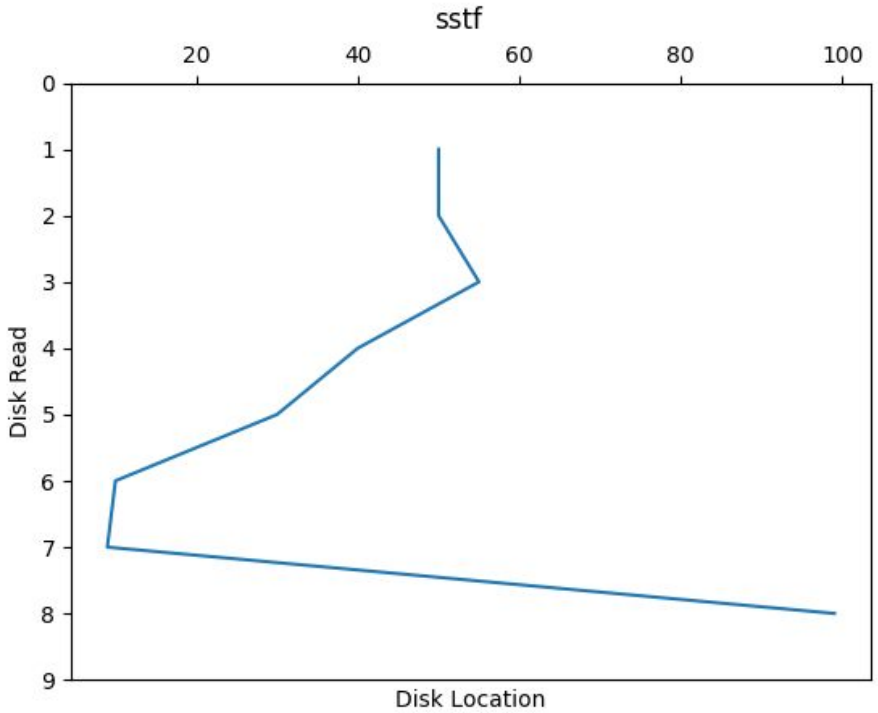
  50 55 10 99 40 30 50 9

- Order:

  50, 50, 55, 40, 30, 10, 9, 99

- Total:

  0 + 5 + 15 + 10 + 20 + 1 + 90 = 141



Elevator Scheduling (Scan)

- Given the following seeks:

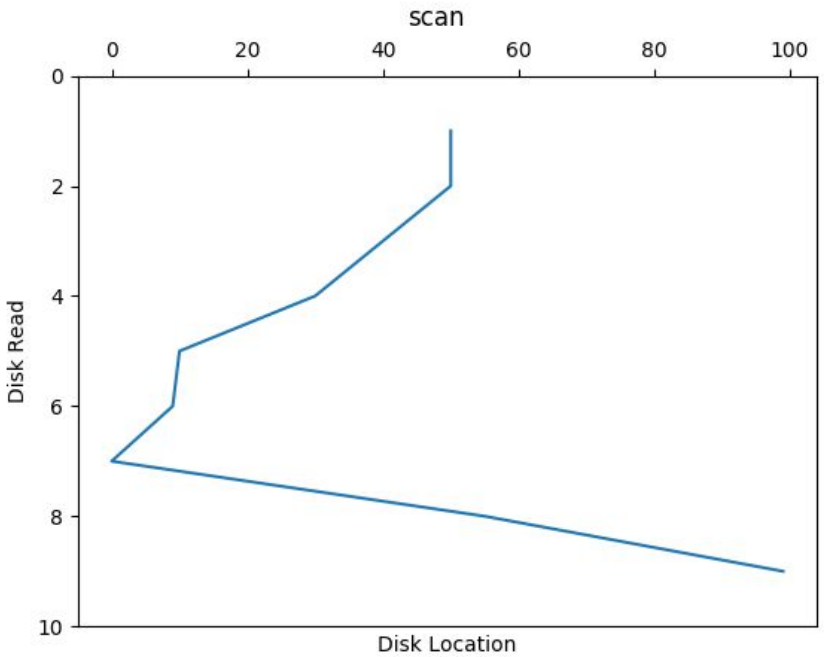
  50 55 10 99 40 30 50 9

- Let’s start going left from point 51.
- Order:

  50, 50, 40, 30, 10, 9, 55, 99

- Total:

  1 + 0 + 10 + 10 + 20 + 1 + 9 + 55 + 44 = 150



Look

- Given the following seeks:

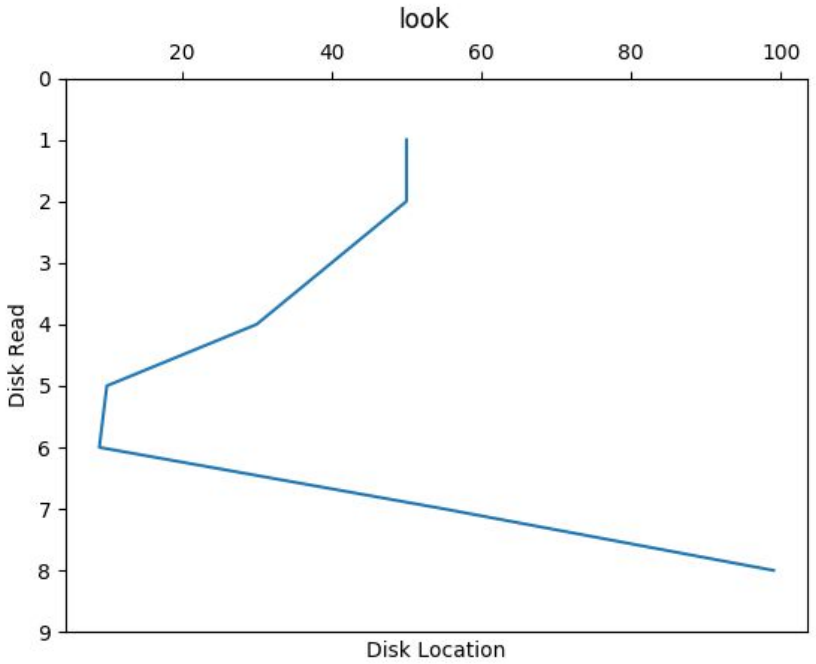
  50 55 10 99 40 30 50 9

- Let’s start going left from point 51.
- Order:

  50, 50, 40, 30, 10, 9, 55, 99

- Total:

  1 + 0 + 10 + 10 + 20 + 1 + 46 + 44 = 132



C-Scan

- Given the following seeks:

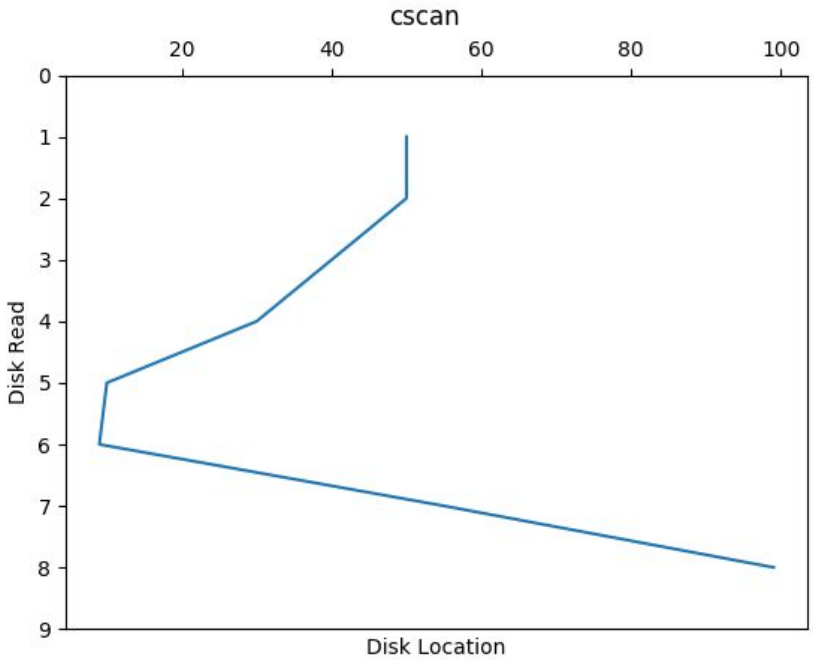
  50 55 10 99 40 30 50 9

- Let’s start going left from point 51.
- Order:

  50, 50, 40, 30, 10, 9, 55, 99

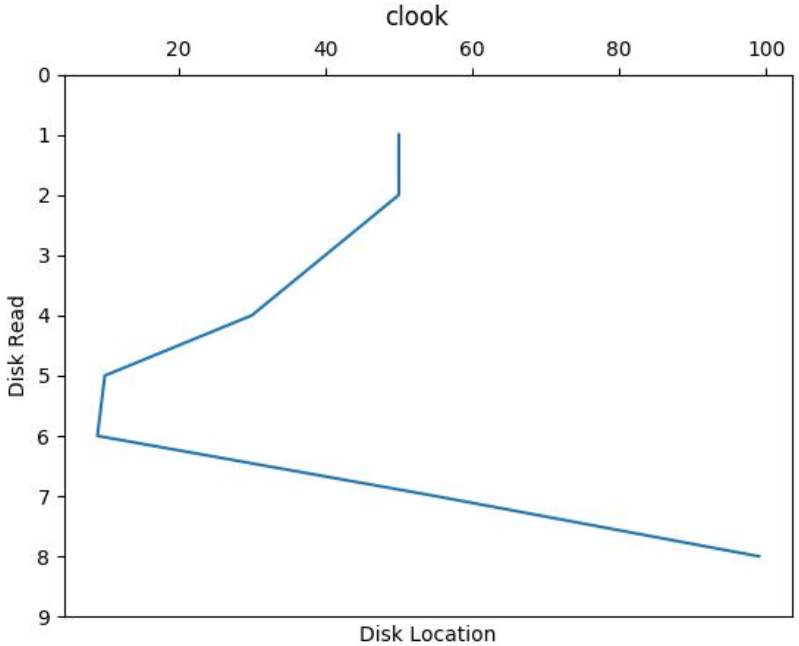
- Total:

  1 + 0 + 10 + 10 + 20 + 1 + 10 + 44 = 96



C-Look

- Given the following seeks:
  50 55 10 99 40 30 50 9
- Let’s start going left from point 51.
- Order:
  50 50 40 30 10 9 55 99
- Total:
  1 + 0 + 10 + 10 + 20 + 1 + 10 + 44 = 96
- Note: C-Look and C-Scan are the same in this scenario. This isnt always the case.



Reading and Writing Structs

- For your project, you will be storing data in your FS file.
- You will be dealing with data in blocks.
- A block is one of:
  - csc452_directory_entry
  - csc452_root_directory
  - A file (just 512 contiguous bytes of data!)
- You will need to load and store these blocks from your FS file.

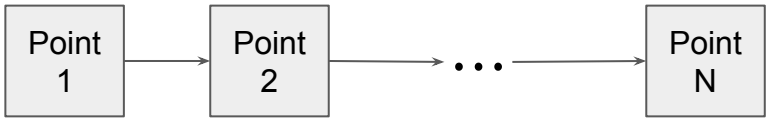
Framing the Problem

```
FILE *fp;

typedef struct Point {
        long x;
        long y;
        long z;
} Point;

void read_struct(FILE *fp, int index);
void write_struct(FILE *fp, int index,
    Point p);
```

- We wish to make a linked list of coordinates
- We should be able to arbitrarily write and read points from this list.



Reading from a File

```
void read_struct(FILE *fp, int index) {
        Point result;
        int offset = index * sizeof(Point);
        fseek(fp, offset, SEEK_SET);
        fread(&result, sizeof(Point), 1, fp);
        printf("Point:\n\tx: %ld\n\ty: %ld\n\tz: %ld\n",
                result.x, result.y, result.z);
}
```

- We just have to create memory to read into, seek to the correct index, and read into the struct!

Writing to a File

```
void write_struct(FILE *fp, int index, Point p) {
        int offset = index * sizeof(Point);
        fseek(fp, offset, SEEK_SET);
        fwrite(&p, sizeof(Point), 1, fp);
        printf("Finished logging point (%ld,%ld,%ld).\n",
             p.x, p.y, p.z);
}
```

- We just have to seek to the correct index and copy the struct into the file!