✅ **Theory: SSTF (Shortest Seek Time First) Disk Scheduling Algorithm**

🔹 **Definition**

SSTF is a disk scheduling algorithm that **selects the disk I/O request closest to the current head position**. It reduces the total seek time compared to FIFO, but **does not guarantee fairness** (some requests may starve).

🔹 **How SSTF Works**

1. Start from the **initial head position**.
2. Calculate the **absolute distance** between the current head position and each pending request.
3. Select the **closest request** (minimum seek time).
4. Move the head to that position.
5. Mark that request as **completed** and repeat until all requests are processed.

🔹 **Advantages**

✅ Better performance than FCFS (less head movement)
✅ Reduces average seek time

🔹 **Disadvantages**

❌ May lead to starvation
❌ More complex to implement than FCFS

✅ **Code Explanation (Line by Line)**

```
#include <stdio.h>

#include <stdlib.h>
```

- stdio.h is for input/output functions like printf, scanf
- stdlib.h is used for abs() function (absolute value)

```
int main() {

  int n, i, current, total = 0;

  int requests[100], done[100] = {0};
```

- n = number of disk requests
- requests[] stores the request positions (track numbers)
- done[] marks whether a request is already processed (0 = pending, 1 = done)
- current = current head position
- total = total head movement

---

printf("Enter number of disk requests: ");

scanf("%d", &n);

Asks user how many disk requests exist.

---

printf("Enter disk requests: ");

for (i = 0; i < n; i++)

  scanf("%d", &requests[i]);

Reads the **sequence of disk requests** from user input.

---

printf("Enter initial head position: ");

scanf("%d", &current);

Reads **starting position of disk head**.

---

printf("\nSSTF Disk Scheduling:\n");

printf("Sequence: %d", current);

Displays the start of output and prints the initial head position.

✅ **Main Logic (SSTF Searching Loop)**

for (int count = 0; count < n; count++) {

Loop runs **n times**, processing one request per iteration.

---

int minDist = 9999, pos = -1;

- minDist stores minimum seek distance (initialized to large value)
- pos stores index of nearest request

---

for (i = 0; i < n; i++) {

  if (!done[i]) {

    int distance = abs(current - requests[i]);

    if (distance < minDist) {

      minDist = distance;

```
            pos = i;
        }
    }
}
```

- Checks every **unfinished request**
- Computes distance from current head position
- Selects **request with minimum distance**

```
total += minDist;        // Add movement
current = requests[pos];   // Move head
done[pos] = 1;           // Mark request as completed

printf(" -> %d", current);
```

- Adds seek distance to total
- Moves head to selected track
- Marks request as completed
- Prints movement sequence

```
printf("\nTotal head movement: %d\n", total);
return 0;
}
```

Final output: total seek time