***DISK SCHEDULING PROGRAMS***

1. FCFS Disk Scheduling
2. SSTF (Shortest Seek Time First)
3. SCAN
4. LOOK
5. C-SCAN
6. C-LOOK

**PROGRAMS 1: FCFCS DISK SCHEDULING**

import java.util.Scanner;

public class DiskScheduling {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of disk requests: ");

        int n = scanner.nextInt();

        int[] requests = new int[n];

        System.out.print("Enter the requests: ");

        for (int i = 0; i < n; i++) {

            requests[i] = scanner.nextInt();

        }

        System.out.print("Enter the initial head position: ");

        int head = scanner.nextInt();

        int totalSeekTime = Math.abs(head - requests[0]);

        for (int i = 1; i < n; i++) {

            totalSeekTime += Math.abs(requests[i] - requests[i - 1]);

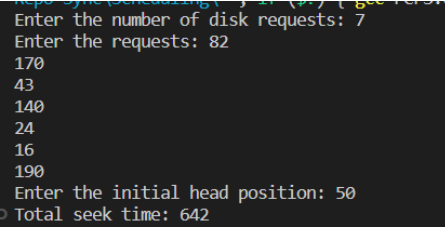
        }

        System.out.println("Total seek time: " + totalSeekTime);

    }

}

**OUTPUT:**

****

**PROGRAMS 2: SHORTEST SEEK TIME FIRST**

import java.util.Scanner;

public class DiskScheduling {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of disk requests: ");

        int n = scanner.nextInt();

        int[] requests = new int[n];

        int[] visited = new int[n];

        System.out.print("Enter the requests: ");

        for (int i = 0; i < n; i++) {

            requests[i] = scanner.nextInt();

            visited[i] = 0; // initialize all requests as unvisited

        }

        System.out.print("Enter the initial head position: ");

        int head = scanner.nextInt();

        int totalSeekTime = 0;

        for (int i = 0; i < n; i++) {

            int minDistance = Integer.MAX\_VALUE;

            int minIndex = -1;

            // find the request with the shortest seek time from the current head position

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

                if (visited[j] == 0) {

                    int distance = Math.abs(head - requests[j]);

                    if (distance < minDistance) {

                        minDistance = distance;

                        minIndex = j;

                    }

                }

            }

            visited[minIndex] = 1; // mark the selected request as visited

            totalSeekTime += minDistance; // add the seek time to the total

            head = requests[minIndex]; // update the head position

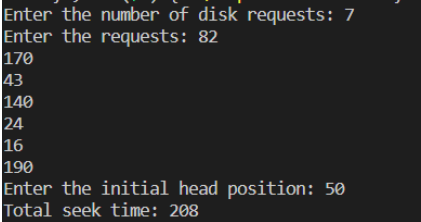
        }

        System.out.println("Total seek time: " + totalSeekTime);

    }

}

**OUTPUT:**

****

**PROGRAMS 3: SCAN**

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        int queue[] = new int[20], n, head, i, j, k, seek = 0, max, diff, temp, queue1[] = new int[20];

        int queue2[] = new int[20], temp1 = 0, temp2 = 0;

        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the max range of disk");

        max = scanner.nextInt();

        System.out.println("Enter the initial head position");

        head = scanner.nextInt();

        System.out.println("Enter the size of queue request");

        n = scanner.nextInt();

        System.out.println("Enter the queue of disk positions to be read");

        for (i = 1; i <= n; i++) {

            temp = scanner.nextInt();

            if (temp >= head) {

                queue1[temp1] = temp;

                temp1++;

            } else {

                queue2[temp2] = temp;

                temp2++;

            }

        }

        for (i = 0; i < temp1 - 1; i++) {

            for (j = i + 1; j < temp1; j++) {

                if (queue1[i] > queue1[j]) {

                    temp = queue1[i];

                    queue1[i] = queue1[j];

                    queue1[j] = temp;

                }

            }

        }

        for (i = 0; i < temp2 - 1; i++) {

            for (j = i + 1; j < temp2; j++) {

                if (queue2[i] < queue2[j]) {

                    temp = queue2[i];

                    queue2[i] = queue2[j];

                    queue2[j] = temp;

                }

            }

        }

        for (i = 1, j = 0; j < temp1; i++, j++) {

            queue[i] = queue1[j];

            queue[i] = max;

        }

        for (i = temp1 + 2, j = 0; j < temp2; i++, j++)

            queue[i] = queue2[j];

        queue[i] = 0;

        queue[0] = head;

        for (j = 0; j <= n ; j++) {

            diff = Math.abs(queue[j + 1] - queue[j]);

            seek += diff;

            System.out.printf("Disk head moves from %d to %d with seek %d\n", queue[j], queue[j + 1], diff);

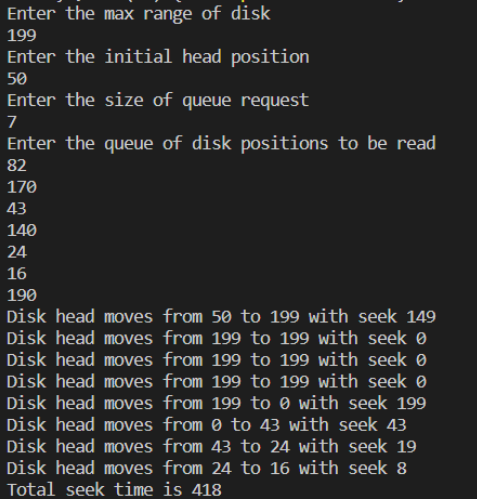
        }

        System.out.println("Total seek time is " + seek);

    }

}

**OUTPUT:**

****

**PROGRAMS 4: LOOK**

import java.util.Arrays;

import java.util.Scanner;

public class DiskScheduling {

    public static void main(String[] args) {

        int n, head, total\_seek\_time = 0;

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of disk requests: ");

        n = scanner.nextInt();

        int[] requests = new int[n];

        System.out.print("Enter the requests: ");

        for(int i = 0; i < n; i++) {

            requests[i] = scanner.nextInt();

        }

        System.out.print("Enter the initial head position: ");

        head = scanner.nextInt();

        // sort the requests in ascending order

        Arrays.sort(requests);

        // find the index of the initial head position in the sorted list

        int initial\_index = 0;

        for(int i = 0; i < n; i++) {

            if(requests[i] == head) {

                initial\_index = i;

                break;

            }

        }

        // handle the requests to the right of the initial head position

        for(int i = initial\_index; i < n; i++) {

            total\_seek\_time += Math.abs(requests[i] - head);

            head = requests[i];

        }

        // handle the requests to the left of the initial head position

        for(int i = initial\_index-1; i >= 0; i--) {

            total\_seek\_time += Math.abs(requests[i] - head);

            head = requests[i];

        }

        System.out.println("Total seek time: " + total\_seek\_time);

    }

}

**OUTPUT:**

**Text

Description automatically generated**

**PROGRAMS 5: C-SCAN**

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        int[] queue = new int[20];

        int n, head, i, j, k, seek = 0, max, diff, temp, temp1 = 0, temp2 = 0;

        float avg;

        System.out.println("Enter the max range of disk");

        max = input.nextInt();

        System.out.println("Enter the initial head position");

        head = input.nextInt();

        System.out.println("Enter the size of queue request");

        n = input.nextInt();

        int[] queue1 = new int[20];

        int[] queue2 = new int[20];

        System.out.println("Enter the queue of disk positions to be read");

        for (i = 1; i <= n; i++) {

            temp = input.nextInt();

            if (temp >= head) {

                queue1[temp1] = temp;

                temp1++;

            } else {

                queue2[temp2] = temp;

                temp2++;

            }

        }

        for (i = 0; i < temp1 - 1; i++) {

            for (j = i + 1; j < temp1; j++) {

                if (queue1[i] > queue1[j]) {

                    temp = queue1[i];

                    queue1[i] = queue1[j];

                    queue1[j] = temp;

                }

            }

        }

        for (i = 0; i < temp2 - 1; i++) {

            for (j = i + 1; j < temp2; j++) {

                if (queue2[i] > queue2[j]) {

                    temp = queue2[i];

                    queue2[i] = queue2[j];

                    queue2[j] = temp;

                }

            }

        }

        for (i = 1, j = 0; j < temp1; i++, j++)

            queue[i] = queue1[j];

        queue[i] = max;

        queue[i + 1] = 0;

        for (i = temp1 + 3, j = 0; j < temp2; i++, j++)

            queue[i] = queue2[j];

        queue[0] = head;

        for (j = 0; j <= n + 1; j++) {

            diff = Math.abs(queue[j + 1] - queue[j]);

            seek += diff;

            System.out.println("Disk head moves from " + queue[j] + " to " + queue[j + 1] + " with seek " + diff);

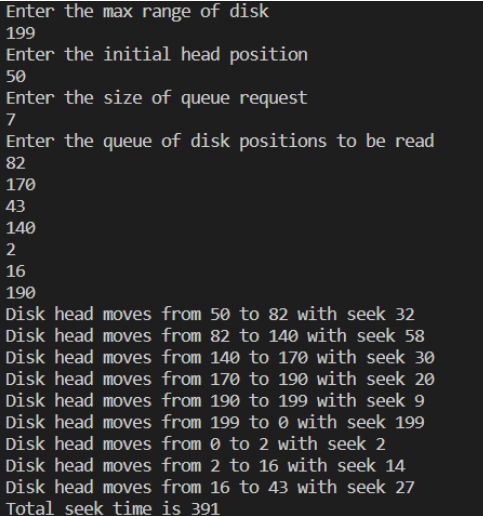
        }

        System.out.println("Total seek time is " + seek);

    }

}

**OUTPUT:**

****

**PROGRAMS 6: C-LOOK**

import java.util.Scanner;

public class DiskScheduling {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        int n, i, j, head, seektime = 0;

        int[] item = new int[20];

        int[] dst = new int[20];

        System.out.println("Enter no. of locations:");

        n = scanner.nextInt();

        System.out.println("Enter position of head:");

        head = scanner.nextInt();

        System.out.println("Enter elements of disk queue:");

        for (i = 0; i < n; i++) {

            item[i] = scanner.nextInt();

            dst[i] = head - item[i];

        }

        // Selection Sort

        for (i = 0; i < n - 1; i++) {

            for (j = i + 1; j < n; j++) {

                if (dst[j] > dst[i]) {

                    int temp = dst[j];

                    dst[j] = dst[i];

                    dst[i] = temp;

                    temp = item[i];

                    item[i] = item[j];

                    item[j] = temp;

                }

            }

        }

        for (i = 0; i < n; i++) {

            if (item[i] >= head) {

                j = i;

                break;

            }

        }

        System.out.println("j=" + j);

        System.out.println("\nOrder of disk allocation is as follows:");

        for (i = j; i < n; i++) {

            System.out.print(" -> " + item[i]);

            seektime += Math.abs(head - item[i]);

            head = item[i];

        }

        for (i = 0; i < j; i++) {

            System.out.print(" -> " + item[i]);

            seektime += Math.abs(head - item[i]);

            head = item[i];

        }

        System.out.println("\n\nSeek Time: " + seektime + "\n\n");

    }

}

**OUTPUT:**

**Text

Description automatically generated**