Republic of the Philippines

Technological University of the Philippines

COLLEGE OF SCIENCE

Manila

COMPUTER STUDIES DEPARTMENT
BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY

Project 3: Disk Scheduling Algorithms

[illegible]

Victoria, Jerico M.

[illegible]

Matullano, Raymond T.

[illegible]

Panaligan, Francis Edian M.

[illegible]

Santos, Ramino Jake H.

TABLE OF CONTENTS

Background

Disk Scheduling Algorithms

As a computer deals with multiple processes over a period of time, a list of requests to access the disk builds up. Disk Scheduling Algorithms is a technique that the operating system uses to determine which requests to satisfy first. The purpose of disk scheduling algorithms is to reduce the total seek time. We developed this program **Shortest Seek Time First (SSTF)** and **LOOK** algorithms. SSTF selects the request with the minimum seek time from the current head position. LOOK similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only.

Objectives

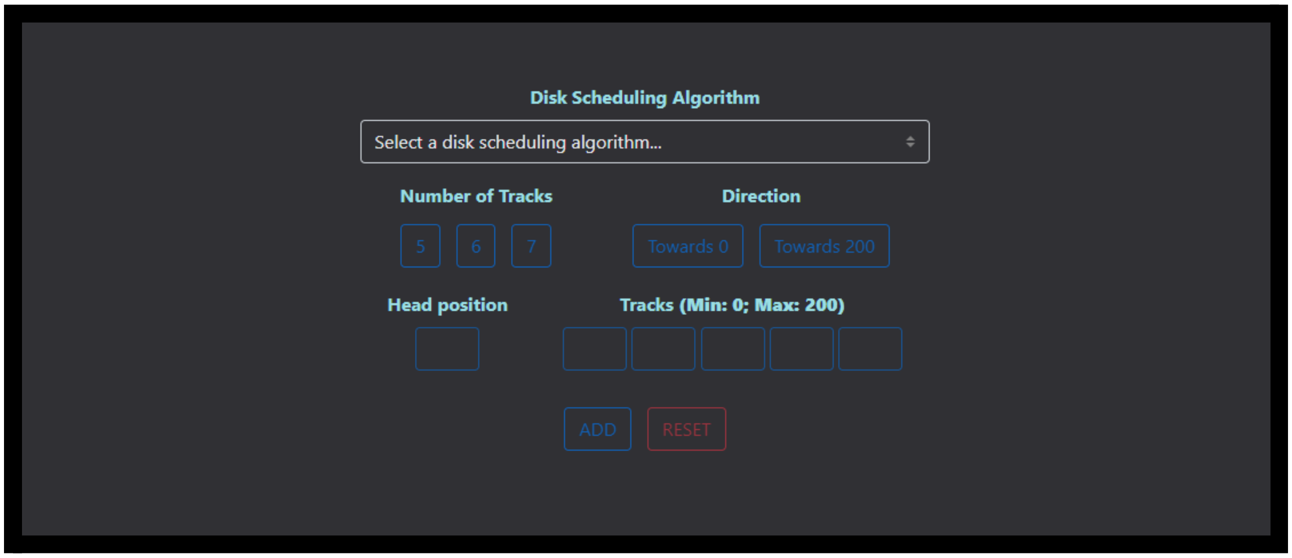
Our team's goal is to create a program that can compute according to user input and generate the table or vertical line chart. The Objective of our program is to calculate the exact total seek time of each algorithm. SSTF and LOOK were implemented using JavaScript programming language and We used CSS, HTML, and Bootstrap for the design and architecture of the program.

Scope and Delimitations

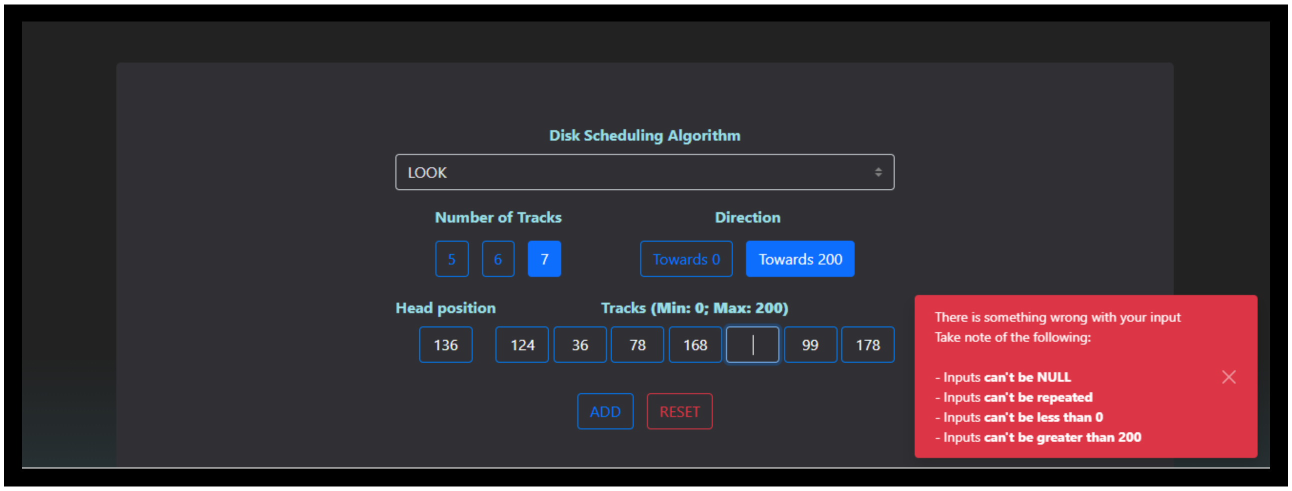
The algorithms that we implemented are SSTF and LOOK. We set the minimum number of tracks to 5 and a maximum of 7, the number of each track can start to 0 up to 200, and a direction specification for LOOK is available in the program; it's either towards zero (0) or towards higher value (200). Our program will compute the total seek time, display the list of traveled tracks in proper sequence, and generate the graph showing the arrows and tracks in their proper positions.

Results and Discussions

After testing our systems for several times, we can say that SSTF and LOOK do their job and process appropriately. SSTF and LOOK can compute the exact total seek time as well as generate the disk scheduler chart which is easy to understand. It depends on the user whether the number of tracks are 5, 6, or 7 then input the value of those tracks and also the head position or the starting point in the chart. After that, users can double check in the chart below if the head position and tracks are in their right positions and if the computation of total seek time is correct. But included in LOOK what direction of tracks will be either towards zero or towards higher value.


Disk Scheduling Algorithm
Select a disk scheduling algorithm...
Number of Tracks
Direction
5
6
7
Towards 0
Towards 200
Head position
Tracks (Min: 0; Max: 200)
ADD
RESET

Here is what the program looks like at the beginning. Let's select a disk scheduling algorithm and proceed to start the program.



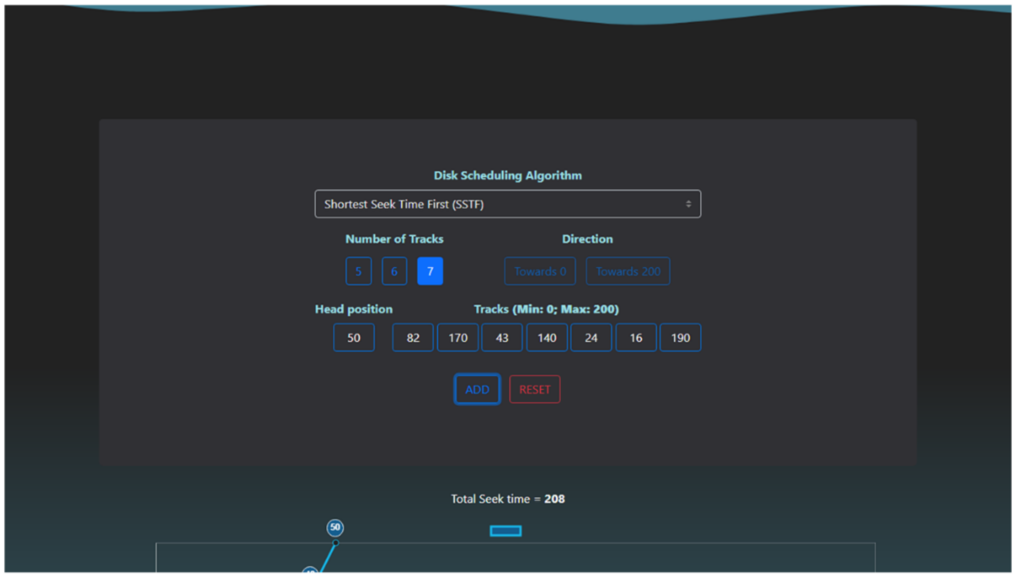
There is something wrong with your input
Take note of the following:

- Inputs **can't be NULL**
- Inputs **can't be repeated**
- Inputs **can't be less than 0**
- Inputs **can't be greater than 200**

As you can see, the user input can't be NULL, repeated, or less than 0 or greater than 200. The validations will make sure that users will do only the following to run the program continuously and avoid interruption. Once the user input is done it will proceed to the graph below.

Let's start with the four data samples for each algorithm. We'll start with SSTF and then move on to LOOK.

Algorithm: Shortest Seek Time
Head Position: 50
Tracks: 82, 170, 43, 140, 24, 16, 190



Disk Scheduling Algorithm

Shortest Seek Time First (SSTF)

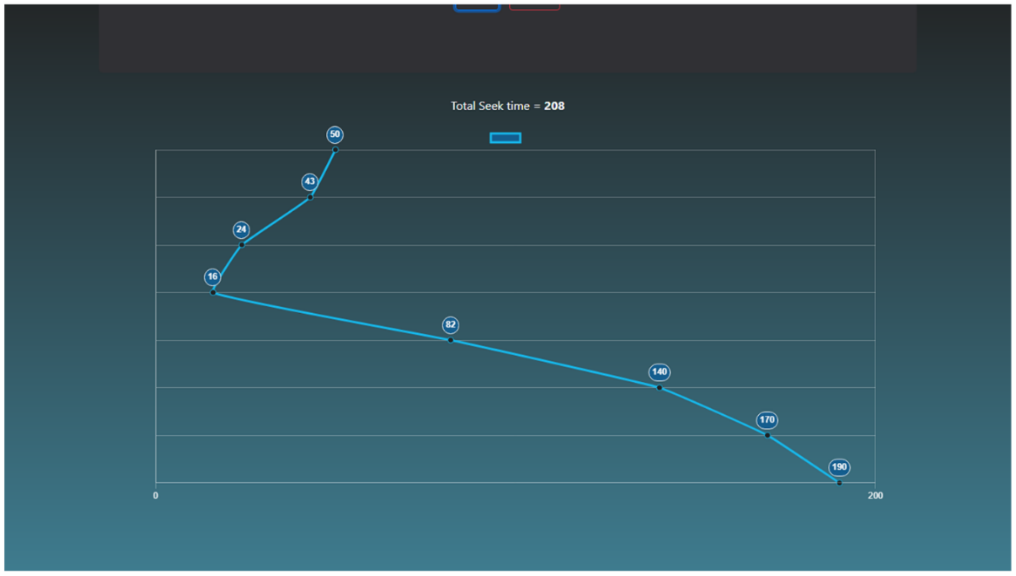
Number of Tracks: 5 6 [✓ 7]

Direction: Towards 0 Towards 200

Head position: 50

Tracks (Min: 0; Max: 200): 82 170 43 140 24 16 190

Total Seek time = **208**

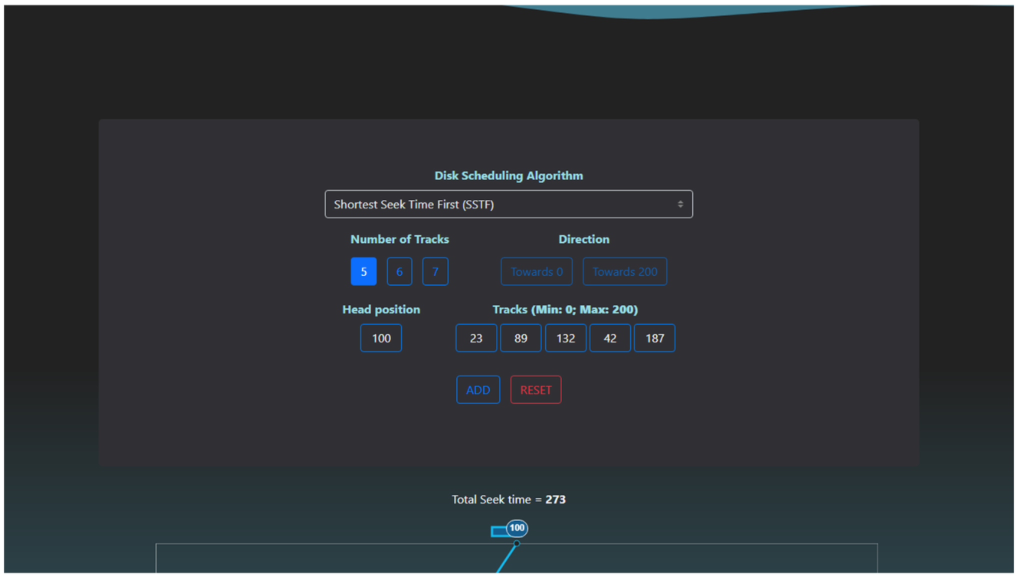


Total Seek time = **208**

Total Seek Time: 208

Figure 1.1: SSTF

Algorithm: Shortest Seek Time
Head Position: 100
Tracks: 23, 89, 132, 42, 187



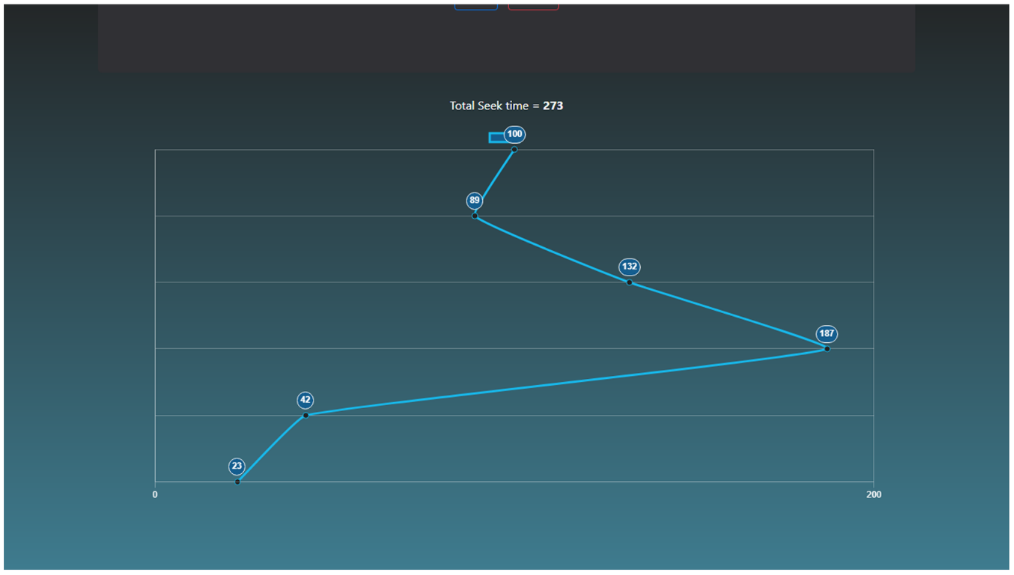
Disk Scheduling Algorithm: Shortest Seek Time First (SSTF)

Number of Tracks: 5

Head position: 100

Tracks (Min: 0; Max: 200): 23, 89, 132, 42, 187

Total Seek time = **273**

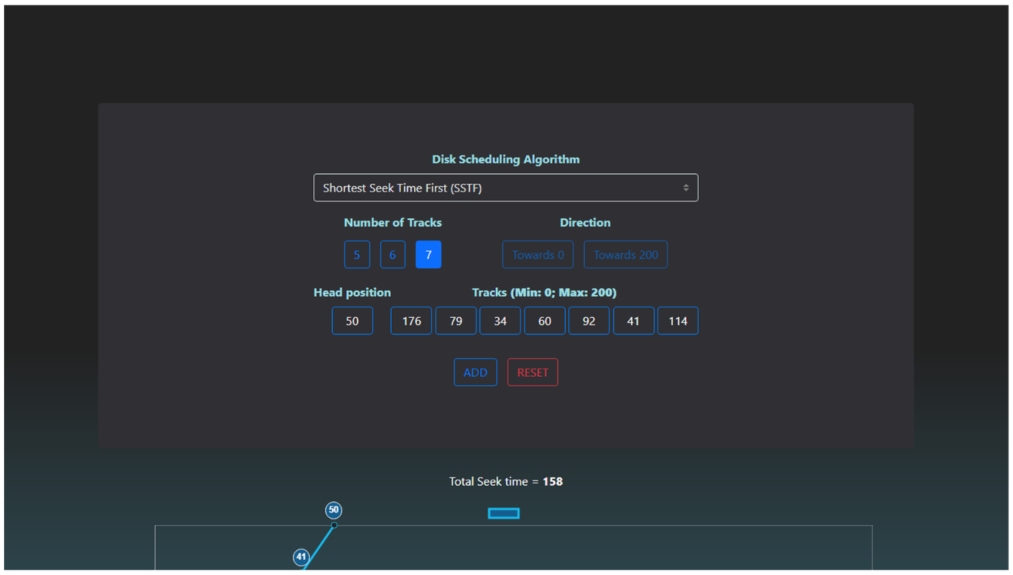


Total Seek time = **273**

Total Seek Time: 273

Figure 1.2: SSTF

Algorithm: Shortest Seek Time
Head Position: 53
Tracks: 98, 183, 40, 122, 10, 124, 65



Disk Scheduling Algorithm

Shortest Seek Time First (SSTF)

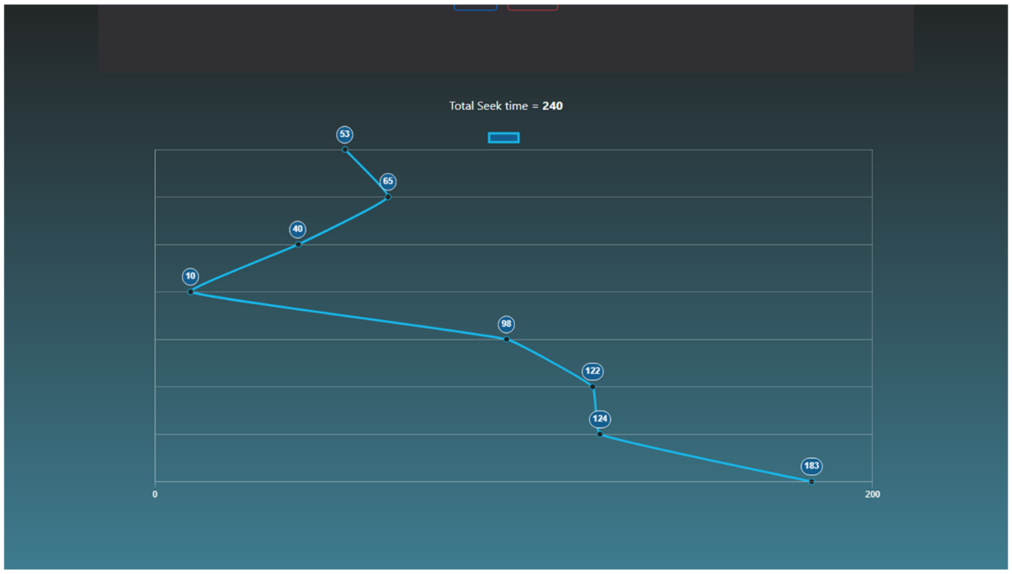
Number of Tracks: 5 6 [✓ 7]

Direction: Towards 0 Towards 200

Head position: 50

Tracks (Min: 0; Max: 200): 176 79 34 60 92 41 114

Total Seek time = **158**

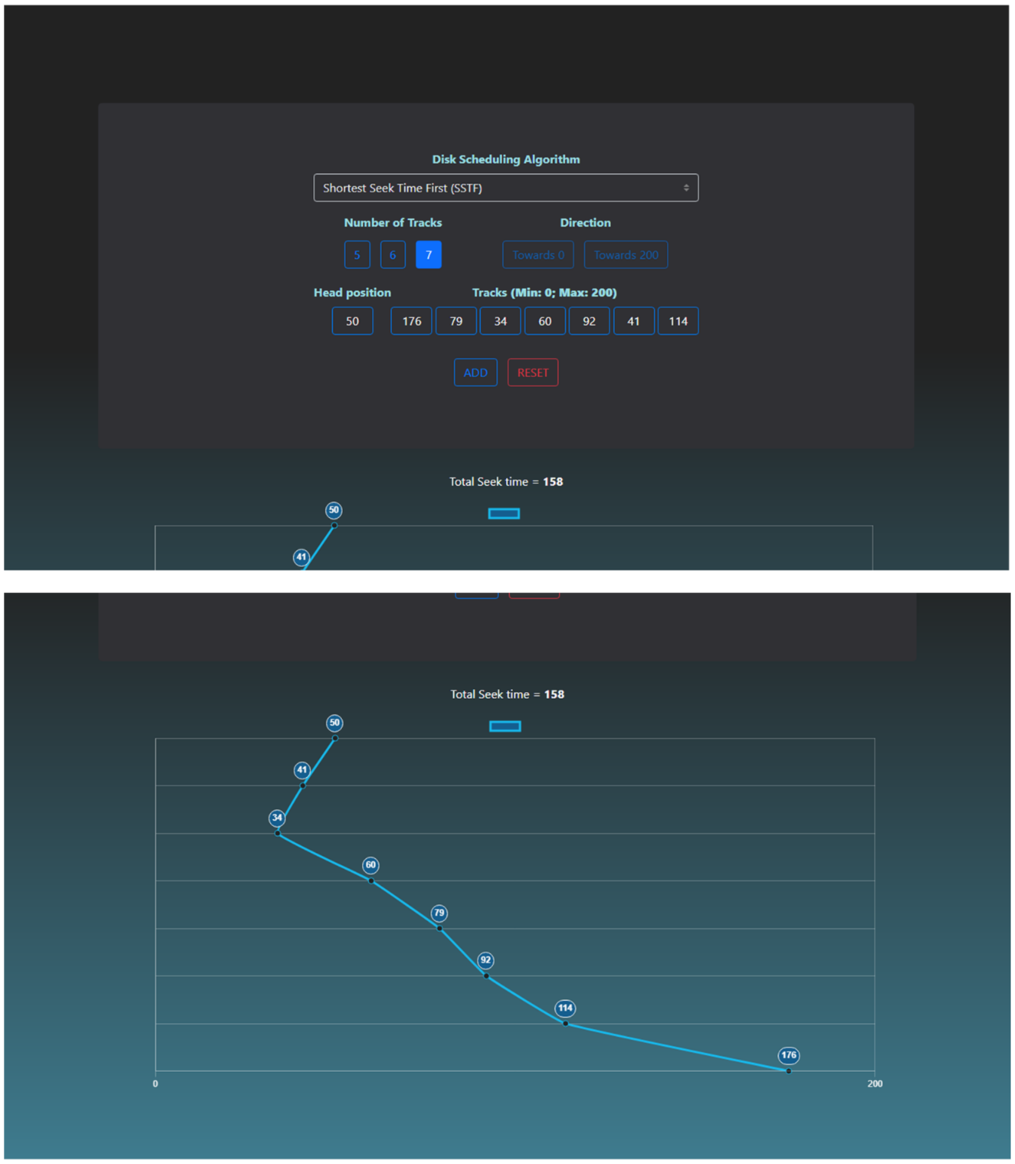


Total Seek time = **240**

Total Seek Time: 240

Figure 1.3: SSTF

Algorithm: Shortest Seek Time
Head Position: 50
Tracks: 176, 79, 34, 60, 92, 41, 114



Disk Scheduling Algorithm

Shortest Seek Time First (SSTF)

Number of Tracks: 5 6 [✓ 7]

Direction: Towards 0 Towards 200

Head position: 50

Tracks (Min: 0; Max: 200): 176 79 34 60 92 41 114

ADD RESET

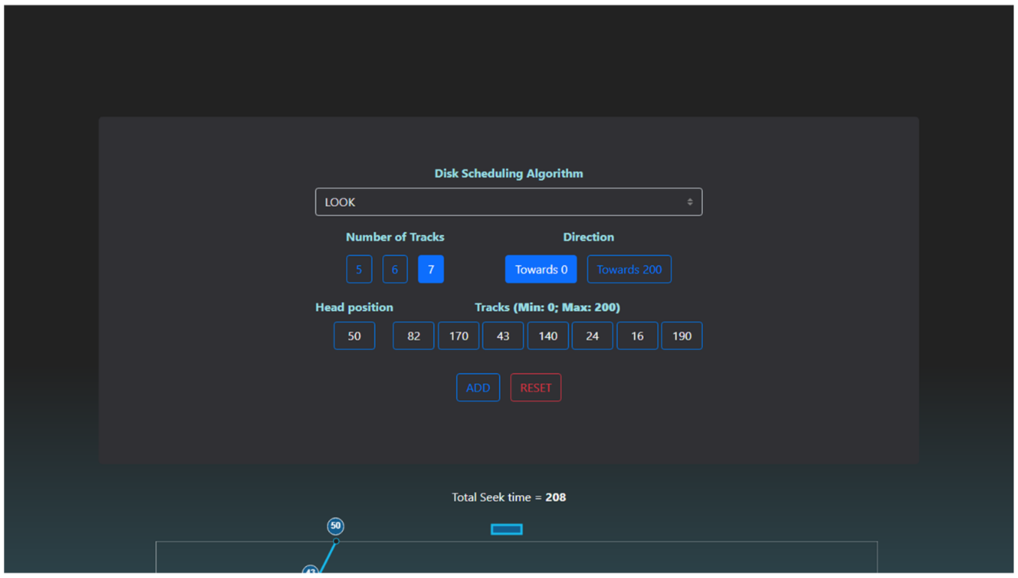
Total Seek time = **158**

Total Seek time = **158**

Total Seek Time: 158

Figure 1.4: SSTF

Algorithm: LOOK
Head Position: 50
Tracks: 82, 170, 43, 140, 24, 16, 190



Disk Scheduling Algorithm

LOOK

Number of Tracks: 5 6 [✓ 7]

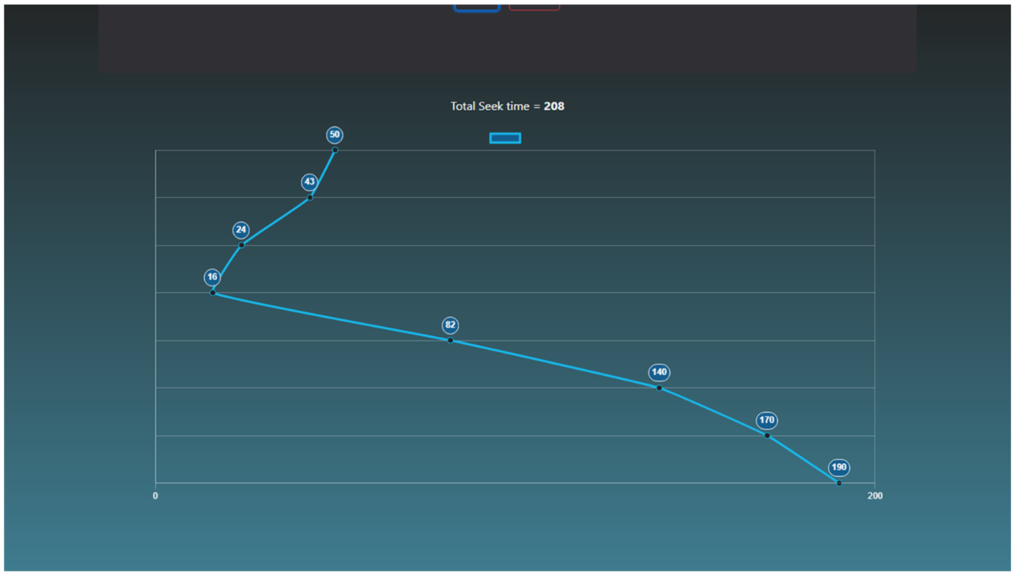
Direction: [✓ Towards 0] Towards 200

Head position: 50

Tracks (Min: 0; Max: 200): 82 170 43 140 24 16 190

ADD RESET

Total Seek time = **208**

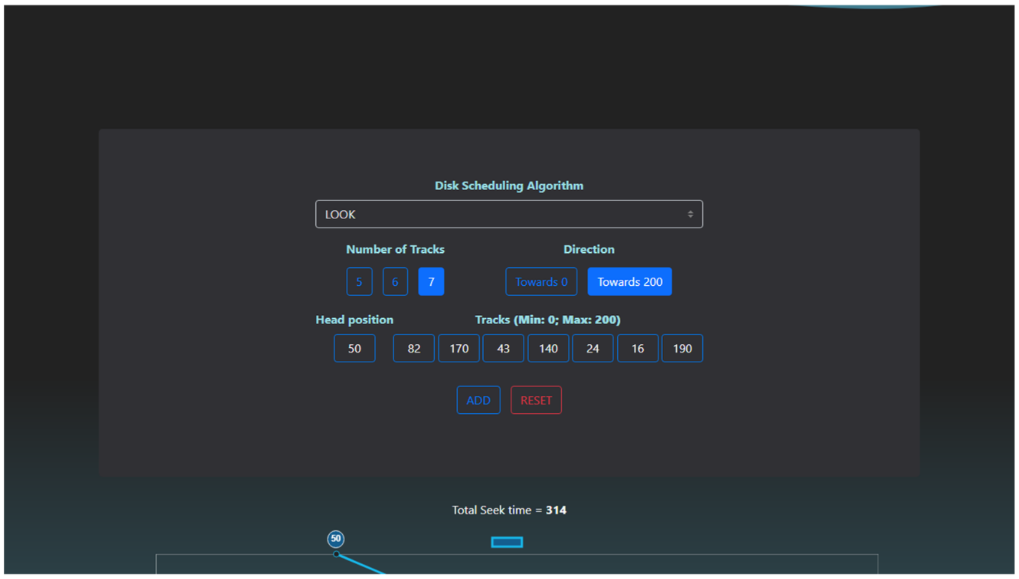


Total Seek time = **208**

Total Seek Time: 208

Figure 2.1: LOOK (Towards 0)

Algorithm: LOOK
Head Position: 50
Tracks: 82, 170, 43, 140, 24, 16, 190



Disk Scheduling Algorithm

LOOK

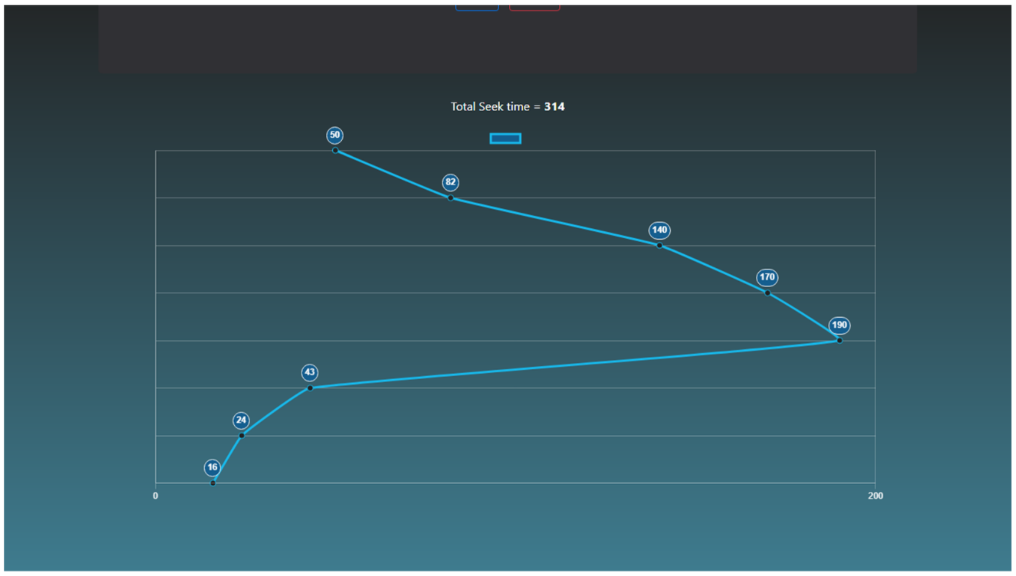
Number of Tracks: 5 6 [✓ 7]

Direction: Towards 0 [✓ Towards 200]

Head position: 50

Tracks (Min: 0; Max: 200): 82 170 43 140 24 16 190

Total Seek time = **314**

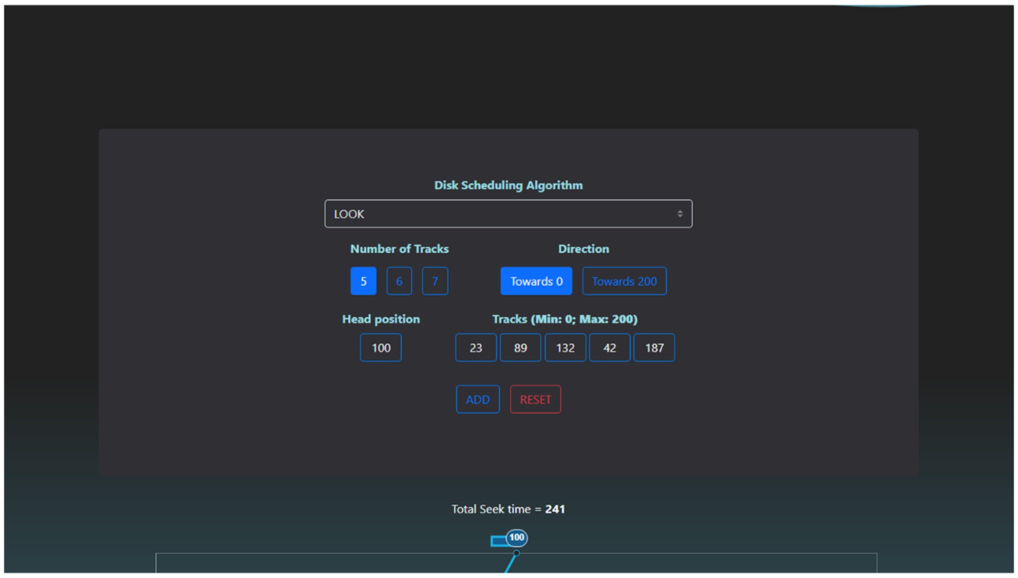


Total Seek time = **314**

Total Seek Time: 314

Figure 2.2: LOOK (Towards 200)

Algorithm: LOOK
Head Position: 100
Tracks: 23, 89, 132, 42, 187



Disk Scheduling Algorithm

LOOK

Number of Tracks: 5 | 6 | 7 [✓ 5]

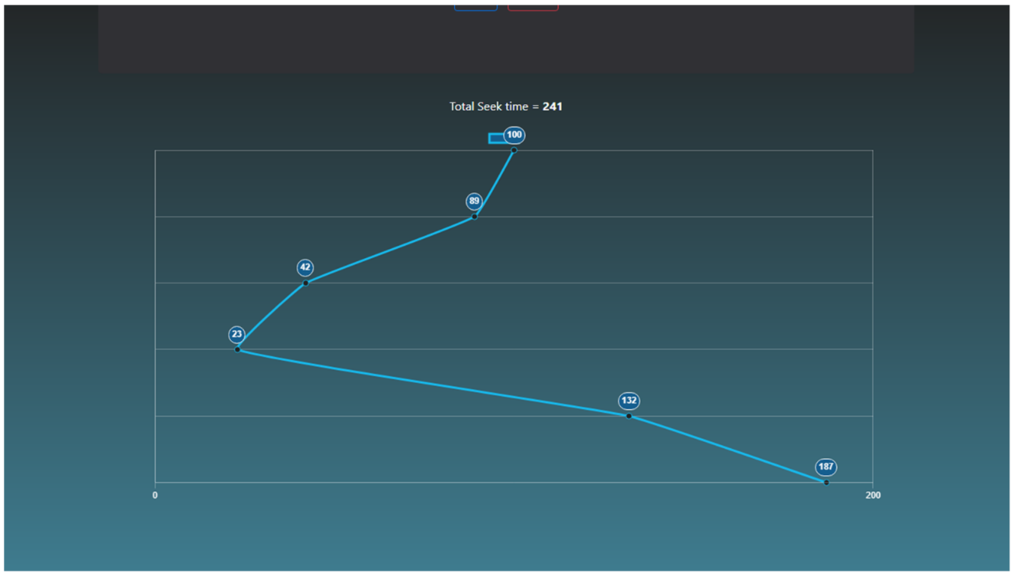
Direction: Towards 0 | Towards 200 [✓ Towards 0]

Head position: 100

Tracks (Min: 0; Max: 200): 23 | 89 | 132 | 42 | 187

ADD RESET

Total Seek time = **241**

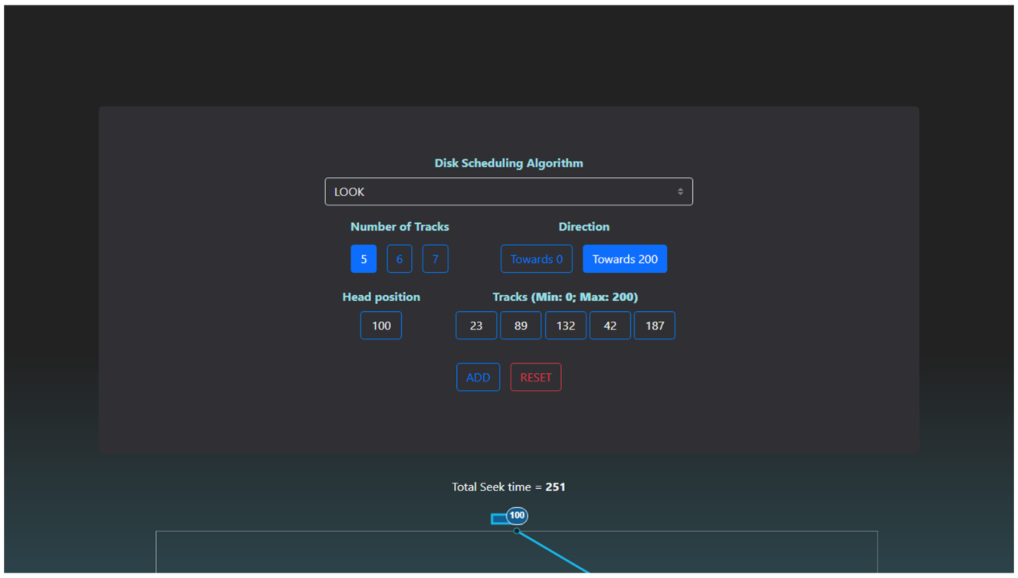


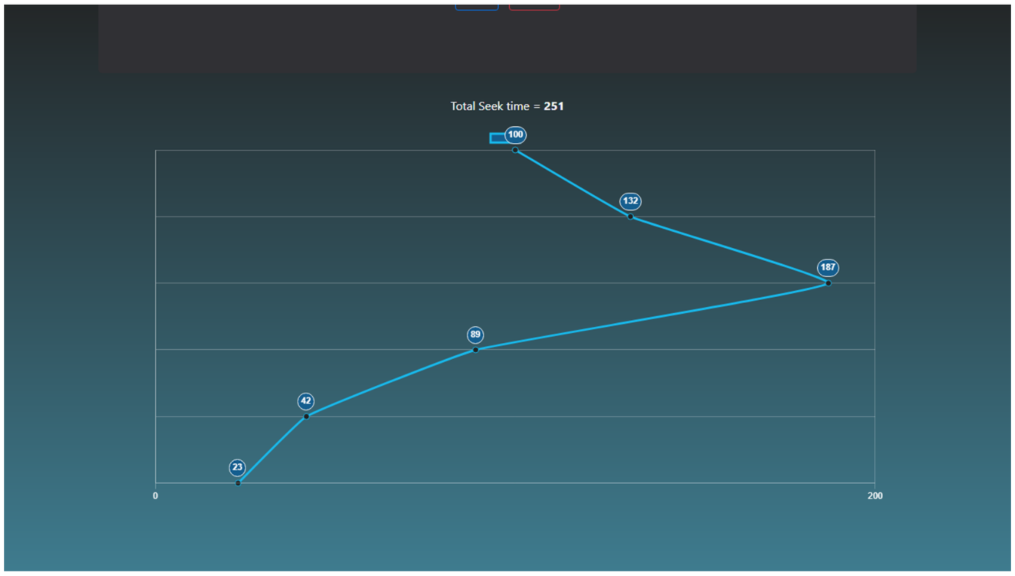
Total Seek time = **241**

Total Seek Time: 241

Figure 2.3: LOOK (Towards 0)

Algorithm: LOOK
Head Position: 100
Tracks: 23, 89, 132, 42, 187

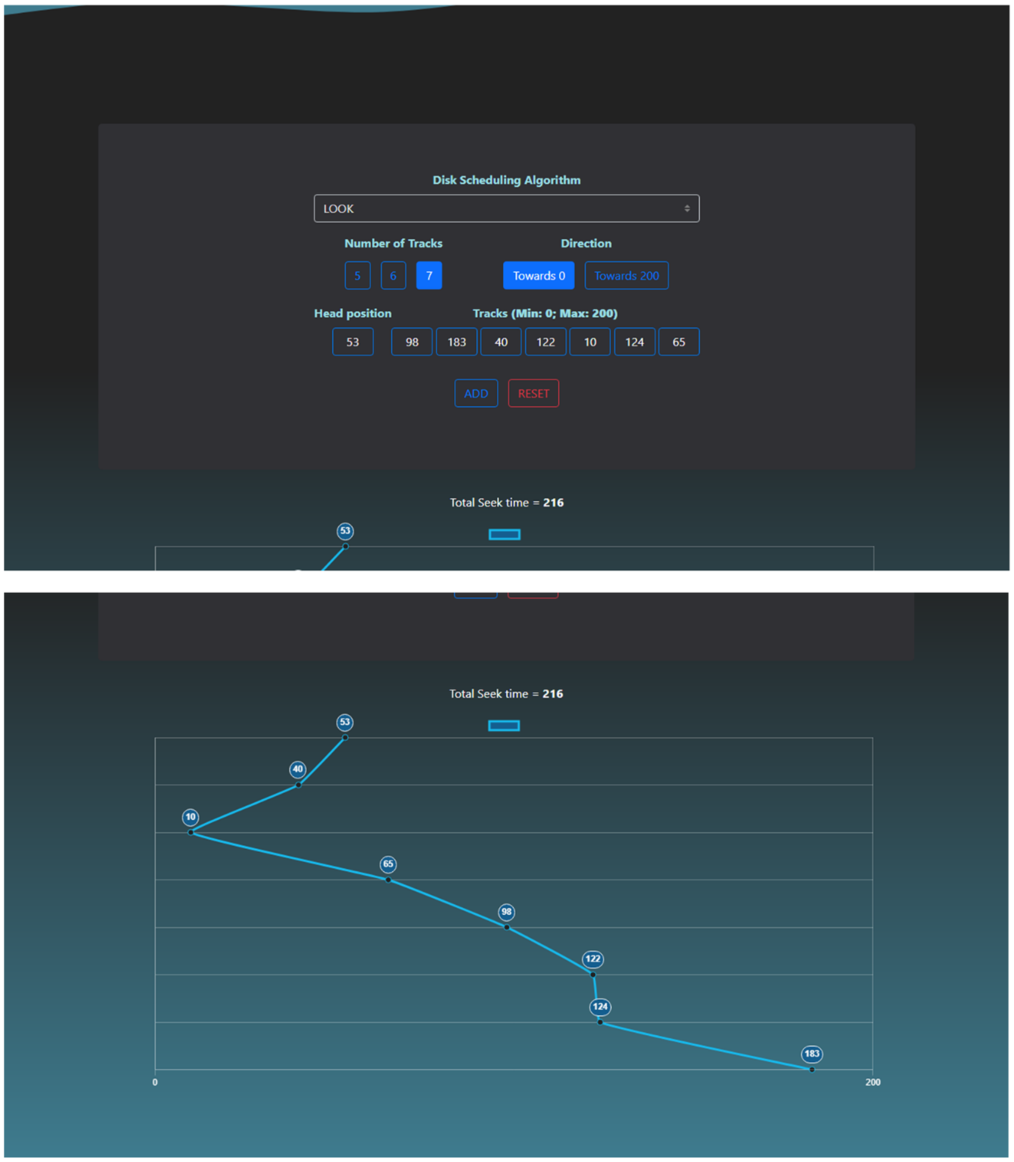




Total Seek Time: 251

Figure 2.4: LOOK (Towards 200)

Algorithm: LOOK
Head Position: 53
Tracks: 98, 183, 40, 122, 10, 124, 65



Disk Scheduling Algorithm

LOOK

Number of Tracks: 5 | 6 | [✓ 7]

Direction: [✓ Towards 0] | Towards 200

Head position: 53

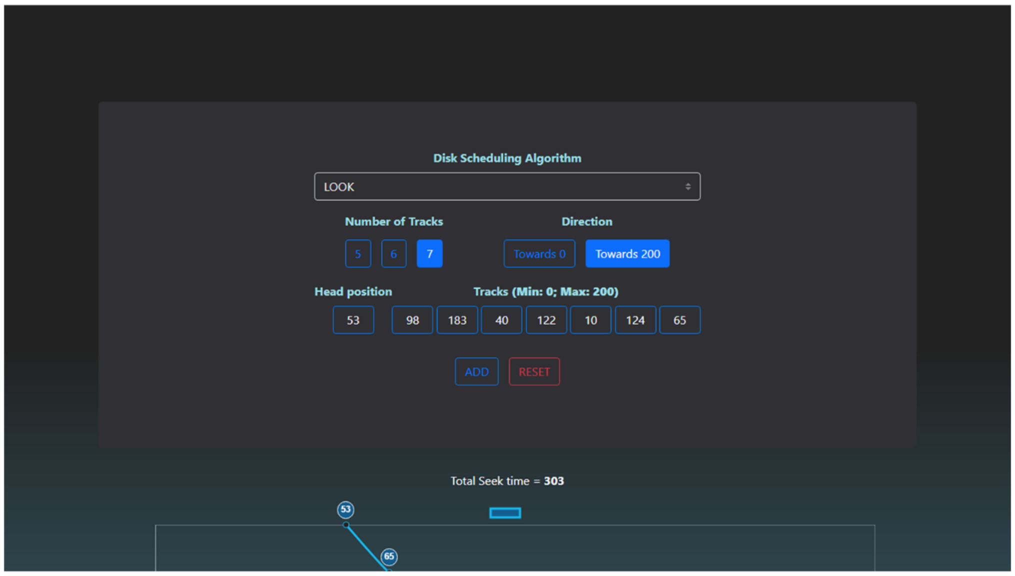
Tracks (Min: 0; Max: 200): 98, 183, 40, 122, 10, 124, 65

Total Seek time = **216**

Total Seek Time: 216

Figure 2.5: LOOK (Towards 0)

Algorithm: LOOK
Head Position: 53
Tracks: 98, 183, 40, 122, 10, 124, 65



Disk Scheduling Algorithm

LOOK

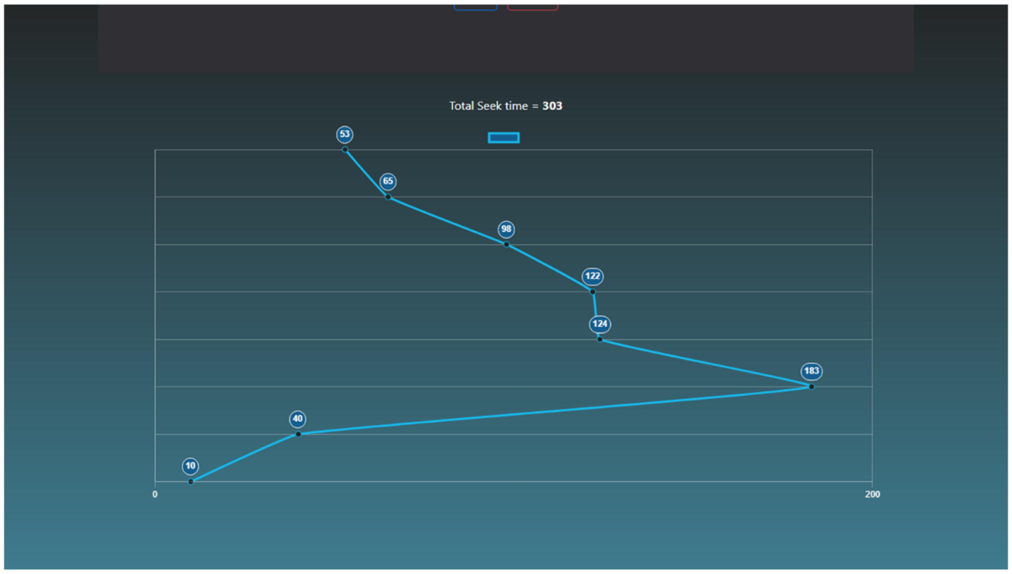
Number of Tracks: 5 6 [✓ 7]

Direction: Towards 0 [✓ Towards 200]

Head position: 53

Tracks (Min: 0; Max: 200): 98 183 40 122 10 124 65

Total Seek time = **303**

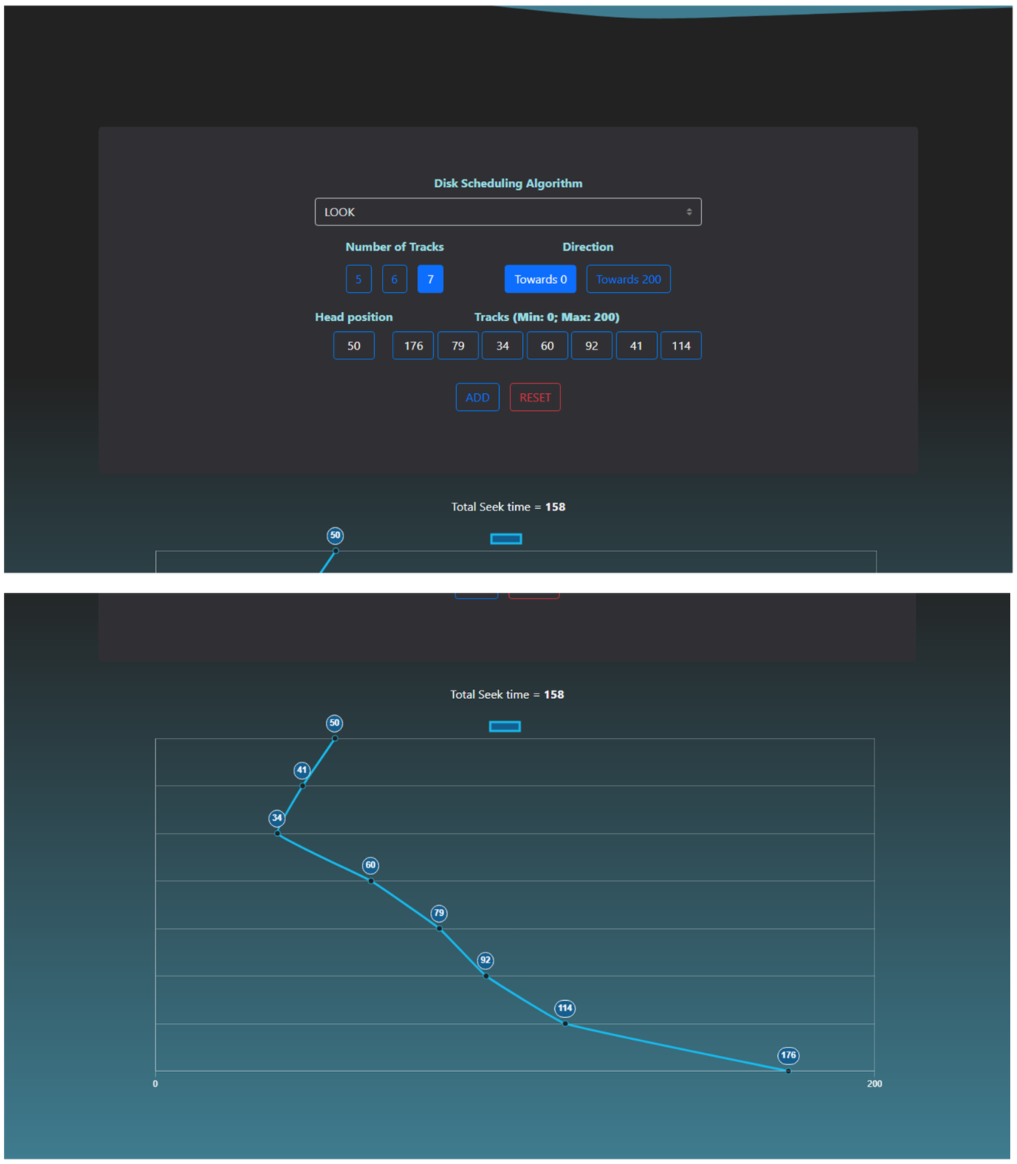


Total Seek time = **303**

Total Seek Time: 303

Figure 2.6: LOOK (Towards 200)

Algorithm: LOOK
Head Position: 50
Tracks: 176, 79, 34, 60, 92, 41, 114


Disk Scheduling Algorithm

LOOK

Number of Tracks: 5 6 [✓ 7]

Direction: [✓ Towards 0] Towards 200

Head position: 50

Tracks (Min: 0; Max: 200): 176 79 34 60 92 41 114

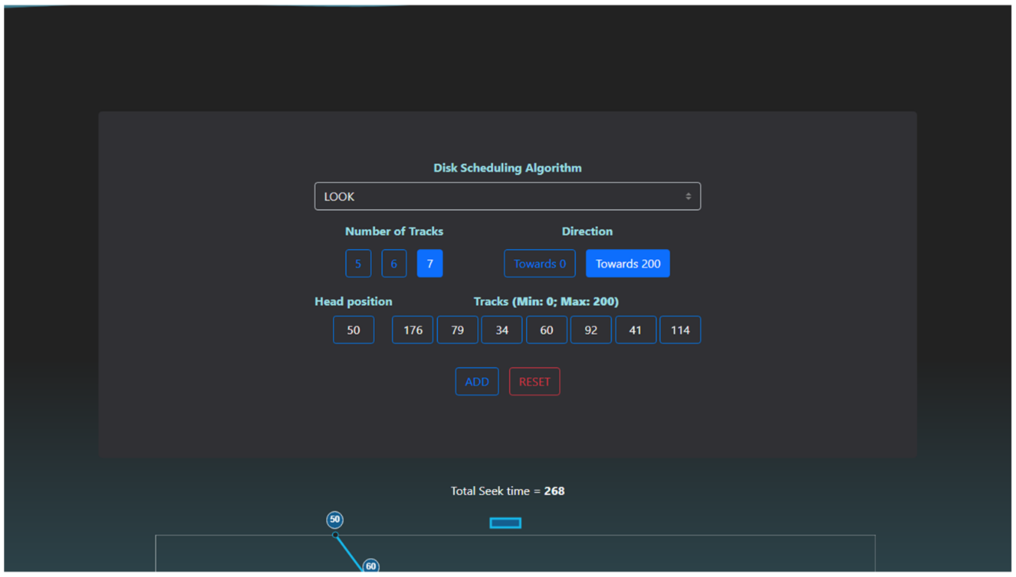
ADD RESET

Total Seek time = **158**

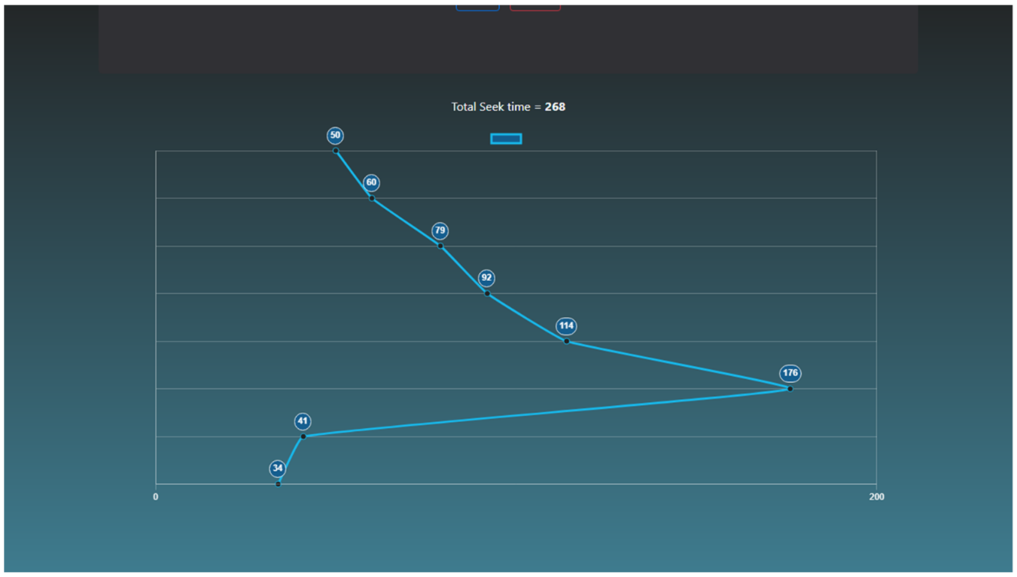
Total Seek Time: 158

Figure 2.7: LOOK (Towards 0)

Algorithm: LOOK
Head Position: 50
Tracks: 176, 79, 34, 60, 92, 41, 114


Disk Scheduling Algorithm
LOOK
Number of Tracks: 5 6 [✓ 7]
Direction: Towards 0 [✓ Towards 200]
Head position: 50
Tracks (Min: 0; Max: 200): 176 79 34 60 92 41 114

Total Seek time = **268**


Total Seek time = **268**

Total Seek Time: 268

Figure 2.8: LOOK (Towards 200)

Technical Difficulties:

Here is a list of the technical difficulties that we have encountered:

1. The two programmers of our team have technical difficulties in terms of the number of tracks on how to adjust the number of tracks that will be used by the user whether 5, 6, or 7. It was solved by implementing a button wherein the user can click it to adjust the number of tracks.
2. There is a visual problem with the chart wherein the labels were shown at the left side of the chart instead of below it. We fixed this issue by implementing *chart.js.labels* plugin so now, it looks better than the first chart. It took so many attempts and adjustments to put the position of tracks inside the chart.
3. The total seek time was not accurate, sometimes it resulted in NAN or negative infinity when the Algorithm is LOOK but we managed to solve and fix this kind of problem by using the same calculation method with SSTF.

Group Evaluation:

- ❑ **Functionality(25pts):** Our team is pleased with our system because of its inherent accuracy as we evaluate the generated output seek time that our program generates. It passed four tests to determine whether or not it was accurate. As a result, we will award our team 25 points.
- ❑ **Reliability(25pts):** Our program is simple to use; users can easily understand the UI because it is straightforward and easy to operate. To avoid errors, we added some input validations. If there is an error at the inputs, a toast message will appear to notify the user and provide them with necessary instructions; otherwise, a toast message will appear stating that it is successful. We will give 25 points for our Reliability.
- ❑ **Usability(25pts):** When it comes to the presentation of the system, we can say that it is user friendly. Users can easily recognize the purpose of the input boxes or option buttons. The user will also learn because he or she will be able to examine the chart generated by our program. As a result, we will give 25 points for our Usability.
- ❑ **Efficiency(23pts):** In terms of efficiency, our program code was vastly long and if carefully examined, it takes up a lot of JavaScript memory. Despite the fact that the execution time is short, we believe that our algorithm can be improved, allowing us to save more resources. We will give 23 points for Efficiency.

FUNCTIONALITY	**25 pts.**
RELIABILITY	**25 pts.**
USABILITY	**25 pts.**
EFFICIENCY	**23 pts.**
TOTAL	**98 pts**

Conclusion and Recommendation:

Our team can say that this program does its job after we tried several times with no encounter of error or wrong output. The user can choose the number of tracks, direction, input the value of tracks, and the head position. The user needs to make sure that their inputs will not be below 0 and greater than 200 to run the program smoothly. It will show the chart at the end part wherein the user will see the positions of the head and other tracks. The total seek time computation is also accurate. It's simple to use because it is a website. This program will benefit both professors and students.

We will improve our program someday, especially in the future. We need to upgrade the algorithms whereby it will execute faster and ensure that no bugs or errors appear in order to achieve 100 percent accuracy which is the main goal of our program.

Appendices:

Job Distribution:

- Victoria, Jerico M.
    - Lead Programmer
- Panaligan, Francis Edian M.
    - Programmer
    - Front End Designer
- Santos, Ramino Jake H.
    - Programmer
    - Quality Assurance
- Matullano, Raymond T.
    - Programmer
    - Documentation

Source Code:

disk.js

```
var ctx = document.getElementById('lineChart');
var addNum;
var addLabel = "edi wow";
var track_count;
var resetButton, addButton;
var tracksInputField = [];
var direction;
var selectedOpts;
var validation = true;

Chart.register(ChartDataLabels);

var data = new Chart(ctx, {
  type: 'line',
  data: {
    labels: [],
    datasets: [{
      label: "",
      fill: false,
      lineTension: 0.1,
      backgroundColor: "#145e8f",
      borderColor: "#18baed",
      borderCapStyle: 'butt',
      borderDash: [],
      borderDashOffset: 0.0,
      borderJoinStyle: 'miter',
      color: "#fff",
      pointStyle: 'circle',
      pointRotation: 60,
      pointRadius: 4,
      pointBorderColor: "#18baed",
      pointBackgroundColor: "#222222",
      pointBorderWidth: 1,
      pointHoverRadius: 5,
      pointHitRadius: 10,
      data: [],
      datalabels: {
        align: 'end',
        anchor: 'end'
      }
    }],
  },
  options: {
    tooltips: {
      enabled: false
    },
    hover: {
      mode: null
    },
    drawBorder: true,
    indexAxis: 'y',
    scales: {
      xAxes: {
        color: "rgba(255,255,255,1)",
        suggestedMin: 0,
        suggestedMax: 200,
        ticks: {
          autoSkip: false,
          count: 2,
          stepSize: 1,
          maxTicksLimit: 200,
          color: 'white'
        },
        grid: {
          borderColor: 'rgba(255,255,255,.3)',
          color: 'rgba(255,255,255,.3)'
        }
      },
      yAxes: {
        suggestedMax: 200,
        ticks: {
          color: 'transparent'
        },
        grid: {
          borderColor: 'rgba(255,255,255,.3)',
          color: 'rgba(255,255,255,.3)',
          tickColor: 'transparent',
        }
      }
    },
    plugins: {
      datalabels: {
        backgroundColor: function(context) {
          return context.dataset.backgroundColor;
        },
        borderColor: 'white',
        borderRadius: 25,
        borderWidth: 1,
        color: 'white',
        font: {
          weight: 'bold'
        },
        formatter: Math.round,
        padding: 4
      }
    },
  }
});

var toastElList = [].slice.call(document.querySelectorAll('.toast'))
var toastList = toastElList.map(function(toastEl) {
  return new bootstrap.Toast(toastEl, toastOption)
})

var toastOption = {
  animation: true,
  delay: 2000
}


function see() {
  for (let i = 0; i < toastList.length; i++) {
    toastList[i].hide();
  }
  if (validation == false)
    toastList[0].show();
  else
    toastList[1].show();
}

Array.prototype.max = function() {
  return Math.max.apply(null, this);
};

Array.prototype.min = function() {
  return Math.min.apply(null, this);
};

function checkIfArrayIsUnique(myArray) {
  return myArray.length === new Set(myArray).size;
}

function addData() {
  let tracksInputField = document.getElementsByName('track-input');
  let trackValues = [];
  let headValue;
  let seekTime = 0;
  let lowest;
  let highest;
  validation = true;
  removeData();

  for (let i = 0; i <= track_count; i++) {
    if (tracksInputField[i].value == "" || tracksInputField[i].value > 200 ||
tracksInputField[i].value < 0) {
      validation = false;
      see();
      break;
    }
    trackValues.push(parseInt(tracksInputField[i].value));
  }

  if(checkIfArrayIsUnique(trackValues) == false){
    validation = false;
    see();
  }

  if (validation != false) {
    if (selectedOpts == 'look') {
      headValue = trackValues.shift();
      trackValues.sort((a, b) => b - a); // For descending sort
      var lowerThanHead = [];
      var higherThanHead = [];
      for (let i = 0; i < trackValues.length; i++) {
        if (trackValues[i] < headValue)
          lowerThanHead.push(trackValues[i]);
        else
          higherThanHead.push(trackValues[i]);

        lowerThanHead.sort((a, b) => a - b); // For ascending sort
        higherThanHead.sort((a, b) => b - a); // For descending sort
      }
      lowest = lowerThanHead.min(); //get the lowest number
      highest = higherThanHead.max(); //get the highest number

      trackValues.length = 0;
      if (direction == 't_low') {
        while (lowerThanHead.length != 0) {
          trackValues.push(lowerThanHead.pop());
        }
        while (higherThanHead.length != 0) {
          trackValues.push(higherThanHead.pop());
        }
      } else if (direction == 't_high') {
        while (higherThanHead.length != 0) {
          trackValues.push(higherThanHead.pop());
        }
        while (lowerThanHead.length != 0) {
          trackValues.push(lowerThanHead.pop());
        }
      }
      trackValues.unshift(headValue);
      for (let i = 1; i < trackValues.length; i++) {
        seekTime += Math.abs(trackValues[i - 1] - trackValues[i]);
      }
    } else {

      headValue = trackValues.shift();
      trackValues.sort((a, b) => a - b); // For descending sort
      trackValues.unshift(headValue);

      let sstfSet = [];
      let lowest = 0;
      let temp;
      let previous;
      let tempCompare = trackValues.shift();
      while (trackValues.length != 0) {
        for (let j = 0; j < trackValues.length; j++) {
          temp = Math.abs(tempCompare - trackValues[j]);
          if (lowest == 0 || lowest > temp) {
            lowest = temp;
            previous = trackValues[j];
          }
        }
        tempCompare = previous;
        sstfSet.push(previous);
        trackValues = trackValues.filter(function(item) {
          return item != previous;
        })
        lowest = 0;
      }

      sstfSet = sstfSet.filter(function(item) {
        return item != headValue;
      })
      sstfSet.unshift(headValue)
      sstfSet.reverse();
      while (sstfSet.length != 0) {
        trackValues.push(sstfSet.pop());
      }

      for (let i = 1; i < trackValues.length; i++) {
        seekTime += Math.abs(trackValues[i - 1] - trackValues[i]);
      }
    }

    for (let i = 0; i <= track_count; i++) {
      data.data.labels.push(trackValues[i]);
      data.data.datasets.forEach((dataset) => {
        dataset.data.push(trackValues[i]);
      });
    }
    data.update();
    document.getElementById('seekTime').innerHTML = "Total Seek time =
<strong>" + seekTime + "</strong>";
    document.getElementById("seekTime").scrollIntoView();
    see();
  }
}

function removeData() {
  data.data.labels.length = 0;
  data.data.datasets.forEach((dataset) => {
    dataset.data.length = 0;
  });
  data.update();
}

function resetData() {
  var tracksInputField = document.getElementsByName('track-input');
  for (let i = 0; i < tracksInputField.length; i++) {
    tracksInputField[i].value = "";
  }
  data.data.labels.length = 0;
  data.data.datasets.forEach((dataset) => {
    dataset.data.length = 0;
  });
  data.update();
  document.getElementById('seekTime').innerHTML = "";
  $('#exampleModal').modal('hide')
}

function clearInput(tracksInputField) {
  tracksInputField.value = "";
```

```
}

function lengthInput(trackInputField) {
  if (trackInputField.value.length > trackInputField.maxLength)
    trackInputField.value = trackInputField.value.slice(0,
trackInputField.maxLength);
}

function numTracks(opts) {
  selectedOpts = opts.value;
  var numTracksCheck = document.getElementsByName('tracks');
  var directionCheck = document.getElementsByName('direction');
  var tracksInputField = document.getElementsByName('track-input');
  addButton.disabled = false;
  resetButton.disabled = false;

  if (numTracksCheck[0].disabled == true) {
    numTracksCheck[0].checked = true;
  }

  if (selectedOpts == 'sstf' || selectedOpts == 'look') {
    for (let i = 0; i < tracksInputField.length; i++) {
      tracksInputField[i].disabled = false;
    }
  }

  switch (selectedOpts) {
    case "sstf":
      for (i = 0; i < numTracksCheck.length; i++) {
        numTracksCheck[i].disabled = false;
      }
      for (i = 0; i < directionCheck.length; i++) {
        directionCheck[i].disabled = true;
        directionCheck[i].checked = false;
      }
      break;
    case "look":
      direction = 't_low';
      for (i = 0; i < numTracksCheck.length; i++) {
        numTracksCheck[i].disabled = false;
      }
      for (i = 0; i < directionCheck.length; i++) {
        directionCheck[i].disabled = false;
      }
      directionCheck[0].checked = true;
      break;
    default:
      for (i = 0; i < numTracksCheck.length; i++) {
        numTracksCheck[i].disabled = true;
        numTracksCheck[i].checked = false;
      }
      for (i = 0; i < directionCheck.length; i++) {
        directionCheck[i].disabled = true;
        directionCheck[i].checked = false;
      }
  }
```

```
}

function track_controller(track_handler) {
  track_count = parseInt(track_handler.value);
  var textFields = document.getElementsByName('track-input');
  switch (track_count) {
    case 5:
      textFields[6].style.display = "none";
      textFields[7].style.display = "none";
      break;
    case 6:
      textFields[6].style.display = "inline";
      textFields[7].style.display = "none";
      break;
    default:
      textFields[7].style.display = "inline";
      textFields[6].style.display = "inline";
  }
}

function track_direction(track_handler) {
  direction = track_handler.value;
}

function start_body() {
  resetButton = document.getElementById('resetButton');
  addButton = document.getElementById('addButton');
  addButton.disabled = true;
  resetButton.disabled = true;
  var textFields = document.getElementsByName('track-input');
  track_count = 5;
  direction = 't_low';
  textFields[6].style.display = "none";
  textFields[7].style.display = "none";
}
```

“God doesn’t give you what you want… He creates the opportunity for us to do so.”