CSC 641

October 6, 2017

Disk Seek Time and Queueing Simulation

Richard Robinson

Professor: Jozo Dujmovic

Statement of Problem:
The purpose of this project is to model average disk seeking time, and disk seeking distance based on the number of elements in a queue. The model is load-dependent and performance will be improved as queue size increases.

The seek distance and seek time relative to a queue of only one element should decrease substantially as the queue size increases. Seek distances will decrease more rapidly than the time required for the seeking process as disk heads take time to accelerate and hit a maximum speed, after which they no longer possess a positive acceleration.

Program Structure:
This program consists of three primary elements:

-**DiskSimulation.cpp :** The entry point and driver of the simulation
-HeadMovement.cpp: The model data and methods
-**QueueGeneration.cpp:** Queue creation / Manipulation (also contains shortest seek time first method)
++**QueueGeneration.h:** Header file for the cpp

HeadMovement.cpp
This component contains a model that matches real-world data for disk access time robustly. Most models that attempt to capture the seeking behavior of disks fall short as they fail to break the problem down into distinct sections, have sharp transitions, or try to rely on square roots instead of adaptable exponents, and fail to adequately describe this process. The model used here captures disk behavior respectably well. The components used are clearly explained and labeled within the source if one wishes to see what every aspect of the model contains.

This model is based around seeking behaving in one fashion while accelerating, and another once the maximum speed is reached. It also factors in disk-specific elements such as the capacity and minimum seek time and contains an exponent that is not the same one used for every disk.

I also included a simple root model, which will be explained in the comments/additions section.

QueueGeneration.cpp / QueueGeneration.h
This creates an array of values representing a potential queue for a disk. When the queue is initialized it is passed the queue size. The queue is then filled with random, uniformly distributed values within the confines of a passed maximum value. In this case, the maximum value is the number of cylinders of our disk, 8057.

There is a method called moveToNext(). This method moves the head from its current location to the closest location listed within the queue. After the head moves this distance a new random value replaces the old one in the queue and the number of cylinders traveled is then returned.

DiskSimulation.cpp
This contains the environment to run the other methods from, as well as a data structure to store results. The average distance traveled per movement and the average time spent per movement are calculated and the results are displayed.

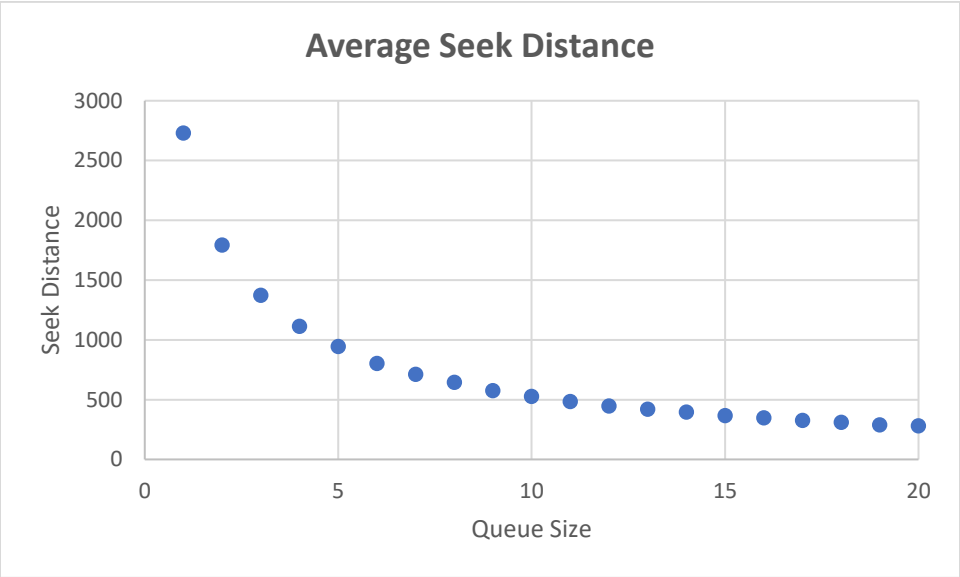
Sample Output:

```
Queue Size: 1   Avg Seek Distance: 2725.86      Avg Seek Time: 8.36222
Queue Size: 2   Avg Seek Distance: 1801.21      Avg Seek Time: 7.00084
Queue Size: 3   Avg Seek Distance: 1370.62      Avg Seek Time: 6.30806
Queue Size: 4   Avg Seek Distance: 1117.2       Avg Seek Time: 5.86838
Queue Size: 5   Avg Seek Distance: 941.685      Avg Seek Time: 5.54112
Queue Size: 6   Avg Seek Distance: 815.218      Avg Seek Time: 5.29078
Queue Size: 7   Avg Seek Distance: 718.037      Avg Seek Time: 5.08737
Queue Size: 8   Avg Seek Distance: 644.433      Avg Seek Time: 4.92338
Queue Size: 9   Avg Seek Distance: 583.293      Avg Seek Time: 4.78123
Queue Size: 10  Avg Seek Distance: 530.074      Avg Seek Time: 4.65218
Queue Size: 11  Avg Seek Distance: 488.987      Avg Seek Time: 4.54691
Queue Size: 12  Avg Seek Distance: 450.155      Avg Seek Time: 4.44433
Queue Size: 13  Avg Seek Distance: 419.402      Avg Seek Time: 4.35933
Queue Size: 14  Avg Seek Distance: 391.225      Avg Seek Time: 4.28169
Queue Size: 15  Avg Seek Distance: 367.954      Avg Seek Time: 4.20798
Queue Size: 16  Avg Seek Distance: 346.543      Avg Seek Time: 4.14458
Queue Size: 17  Avg Seek Distance: 324.723      Avg Seek Time: 4.07582
Queue Size: 18  Avg Seek Distance: 310.802      Avg Seek Time: 4.02786
Queue Size: 19  Avg Seek Distance: 293.984      Avg Seek Time: 3.97402
Queue Size: 20  Avg Seek Distance: 278.698      Avg Seek Time: 3.92175
```

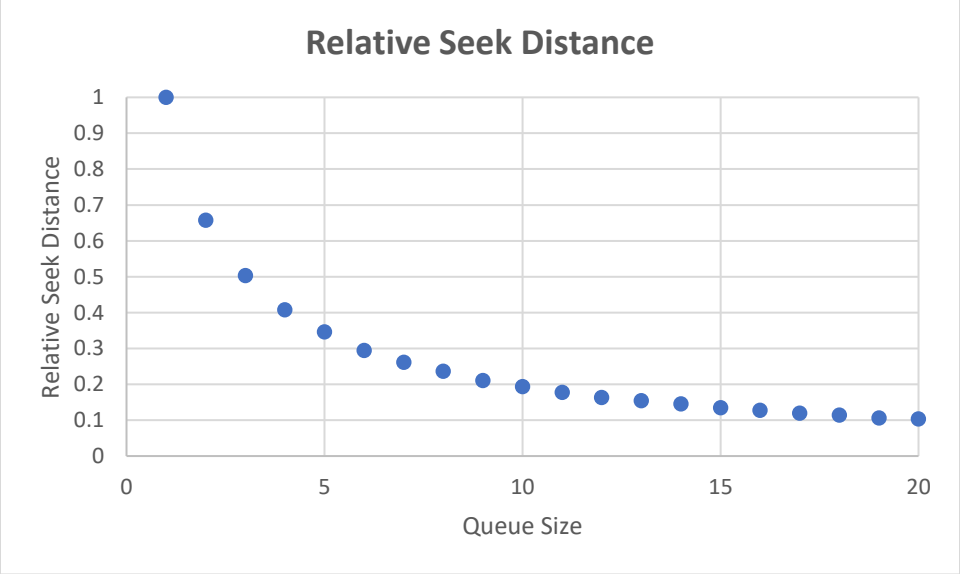
*I am aware that values after a certain cut-off are not significant. I used N = 100,000 for these results.

RESULTS:
The results followed the expected behavior.

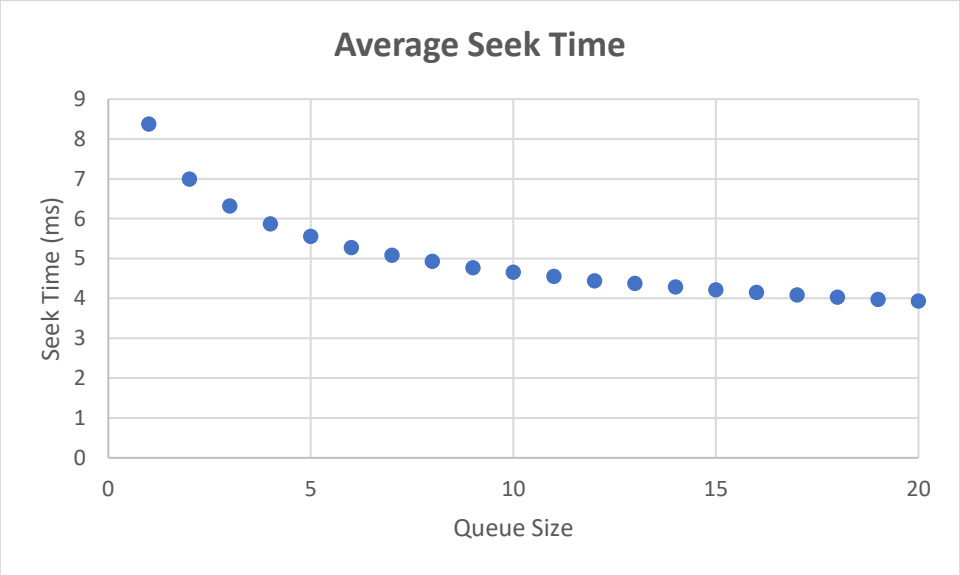
As queue size increases, the mean seek distance (in cylinders) decreases extremely quickly at first and then levels off as the queue size continues to increase.



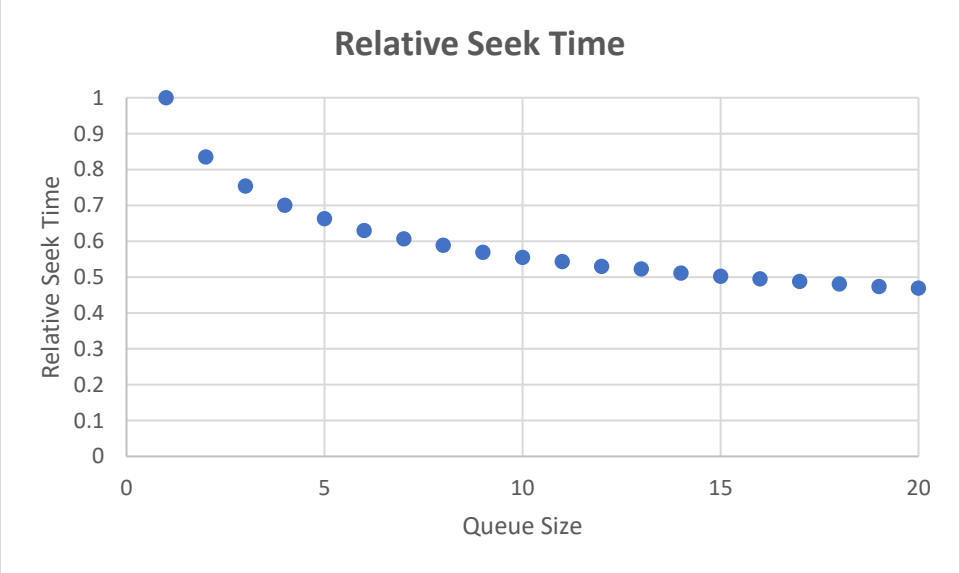
Seek Distance is Relative to A Queue With One Element



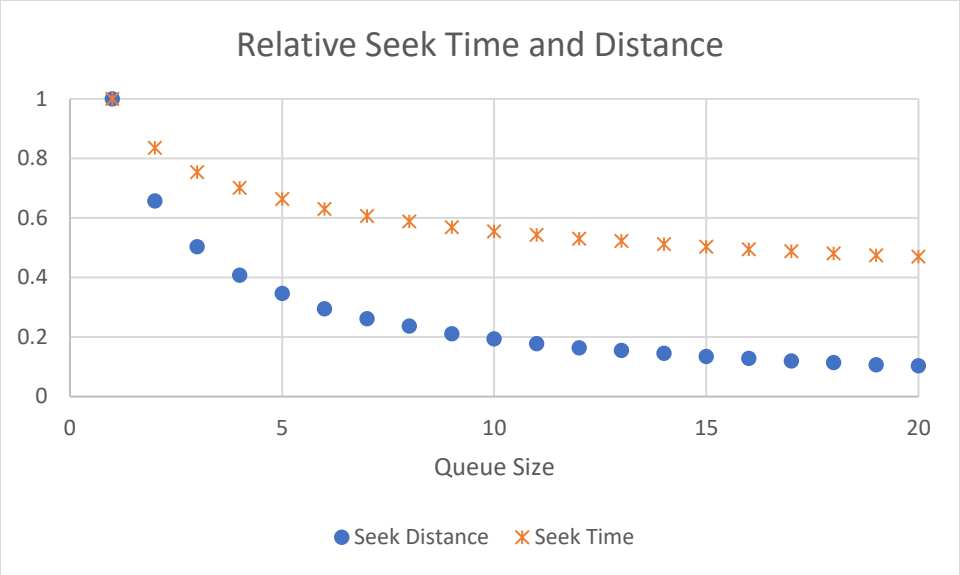
As the number of elements in a queue increase the average seek time decreases. The relative change in seek time decreases at a noticeably slower rate than the cylinder distance graphs decreased at. This is because there is a maximum achieved speed that the disk heads can achieve. Additionally, acceleration is not infinite (fortunately) so the heads take time to hit this maximum speed.



Seek Time is Relative to A Queue With One Element



While at first glance, the seek time may appear to be a simple exponential curve, but it most certainly is not. I plotted these against "best-fits" for linear, exponential, and moving average predictions for the data. Every model was a terrible prediction of the seeking behavior portrayed by this model; they were capable of matching some parts, but not the entire span.

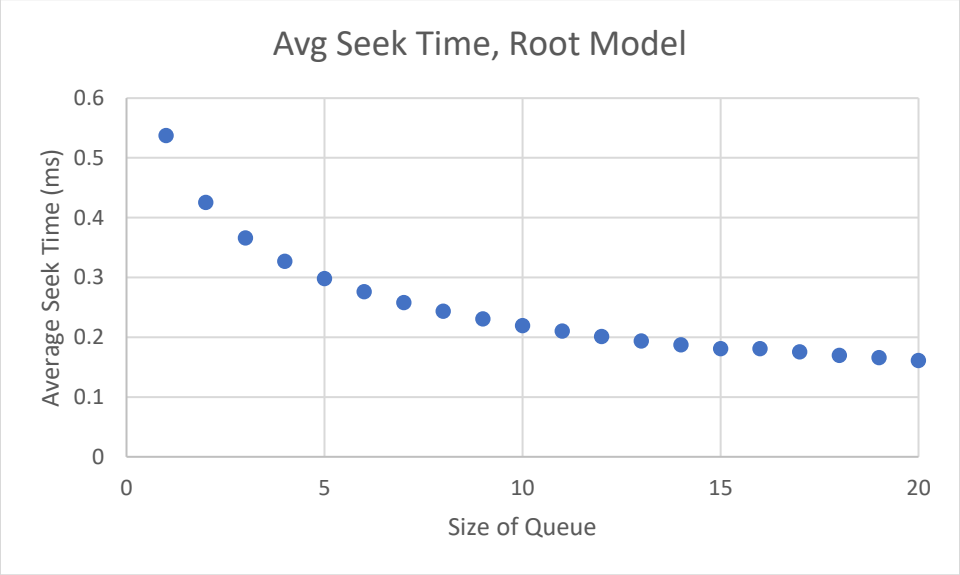


If we compare the seek time graph with the seek distance graph one can clearly see that the relative seek time graph follows a much more subtle curvature than the distance one. This is as expected. As the queue size increases the head should have to move significantly less and less to reach the seeking location. The seek time does not follow the distance trend because of the time it takes for the head to change speeds, its possession of a maximum speed limitation, and the presence of minimum seek-length. Moving over short distances is much slower in velocity than the head moving over large distances, where it can move at its optimal speed. Because of this disparity shorter distances (while still faster than longer distances) are reached at a slower value of cylinders/time. These results were exactly what I was expecting when beginning this project.

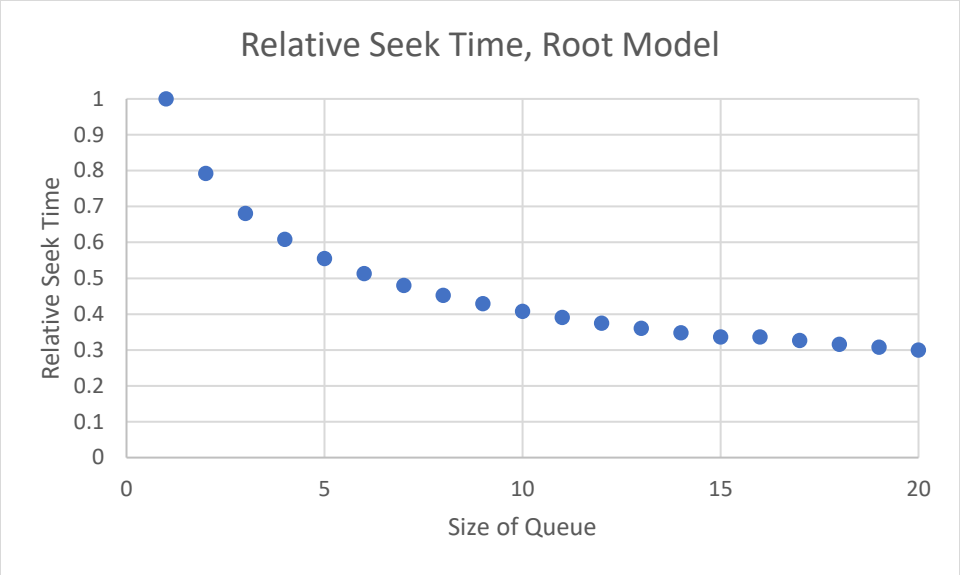
Discussion/Additions:

Analyzing the square root model.

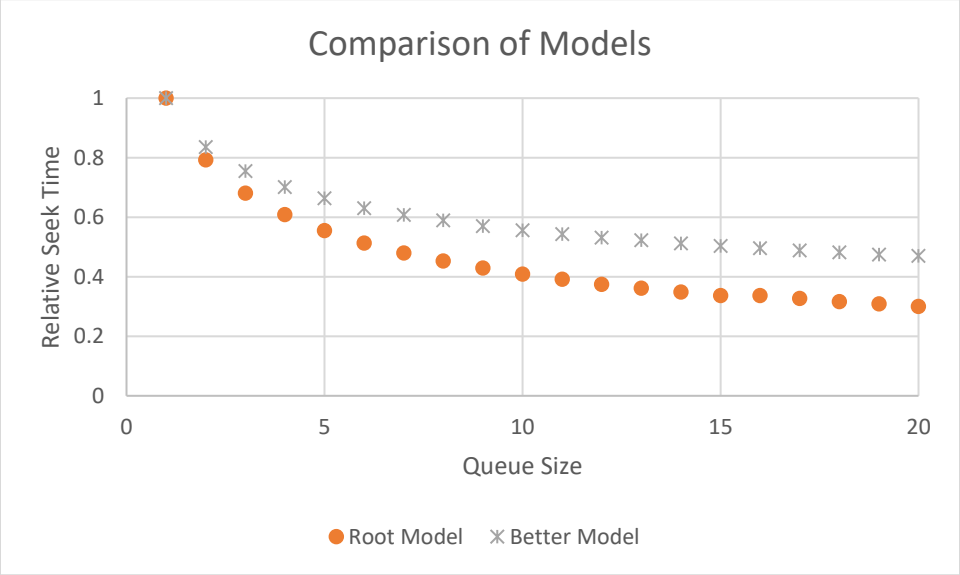
```
Queue Size: 1   Avg Seek Distance: 2725.86      Avg Seek Time: 0.537217
Queue Size: 2   Avg Seek Distance: 1801.21      Avg Seek Time: 0.425459
Queue Size: 3   Avg Seek Distance: 1370.62      Avg Seek Time: 0.365764
Queue Size: 4   Avg Seek Distance: 1117.2       Avg Seek Time: 0.327087
Queue Size: 5   Avg Seek Distance: 941.685      Avg Seek Time: 0.298018
Queue Size: 6   Avg Seek Distance: 815.218      Avg Seek Time: 0.275712
Queue Size: 7   Avg Seek Distance: 718.037      Avg Seek Time: 0.257633
Queue Size: 8   Avg Seek Distance: 644.433      Avg Seek Time: 0.243118
Queue Size: 9   Avg Seek Distance: 583.293      Avg Seek Time: 0.230519
Queue Size: 10  Avg Seek Distance: 530.074      Avg Seek Time: 0.219201
Queue Size: 11  Avg Seek Distance: 488.987      Avg Seek Time: 0.210049
Queue Size: 12  Avg Seek Distance: 450.155      Avg Seek Time: 0.201095
Queue Size: 13  Avg Seek Distance: 419.402      Avg Seek Time: 0.193737
Queue Size: 14  Avg Seek Distance: 391.225      Avg Seek Time: 0.187037
Queue Size: 15  Avg Seek Distance: 367.954      Avg Seek Time: 0.180837
Queue Size: 16  Avg Seek Distance: 346.543      Avg Seek Time: 0.175441
Queue Size: 17  Avg Seek Distance: 324.723      Avg Seek Time: 0.169589
Queue Size: 18  Avg Seek Distance: 310.802      Avg Seek Time: 0.165552
Queue Size: 19  Avg Seek Distance: 293.984      Avg Seek Time: 0.161048
Queue Size: 20  Avg Seek Distance: 278.698      Avg Seek Time: 0.156709
```



There was no maximum seek time given, so I will be comparing relative models and letting max seek time = 1 for simplicity. Now we can get into the more meaningful graphs. Since the root model follows the same trend no matter what the maximum seek time is a generated relative interpretation of the data will yield the exact same result.



Values are relative to a queue of size 1. To show the differences between this model and the other one I have graphed them in juxtaposition.



The root model changes in a significantly different way than the better model, the primary model used for this project, does. They start off rather similarly, but diverge quickly as more items are appended to the queue. The difference between the models is extremely significant towards the later values. At 20 places in the queue the root model's seek time shortens to 30% of the value when compared to a queue of 1 while the better model's seek time shortens to 47%. This difference is huge and should not be taken lightly.

The reason why this difference is so prevalent is that the root model treats the behavior of the head seek time as one simple equation, when this is not sufficient. It must be broken into sub-parts to describe the different regions. Disk heads have a maximum acceleration, a maximum speed, do not follow a power of .5 exactly, and have other factors that when summed together cause many inaccuracies with most (if not all) models that consist of only one contiguous component.

This is not to say that the root model is without use, however. It is useful for general estimations with small queue sizes and is very easy to remember, or to display a somewhat reasonable representation of seek time. The ease of remembering could be useful for introductions and to aid in developing a better understanding of disk behavior, from which can later be expanded upon with better models.

Comments:
This project was fairly interesting, straight-forward, and rewarding. There was nothing that was overly complicated or tedious for the sake of being tedious. It was immensely helpful for learning head movement behaviors and understanding the components behind the functions given. The work done was cohesive and I do not think I will ever forget what was covered here. In addition to all these things, it was a wonderful preparation to the mid-term. I believe projects are much more interesting and effective with learning material than brute forcing homework as you have to break the elements of the problem into its base components then expand upon them when making a program from scratch: one cannot easily through the pieces together to develop something usable without knowing how they operate.

This project also made me think more about how everything will likely switch over to SSD's foreseeable future and simply the evolution of computers. I have a strong desire to read more in depth about SSD components after doing this, which is something I would have taken for granted before.