Fit Algorithm Comparison

I. Problem Statement

The assignment was to compare four different types of scheduling algorithms (First Fit, Best Fit, Worst Fit, and Next Fit). We also had to generate a 1000-element task stream, each with a size and time integer between 1-16. Then we were to run the task stream through each of the algorithms 1000 times and compare the results.

II. Initial Approach

Section 7.2 of the textbook gave some examples for all the algorithms except Next Fit. I studied the three given and then researched online for Next Fit.

The normal distribution used to randomly generate the numbers was given to us in the assignment. I noticed that the given code didn’t seed random() and I thought about including this. In the end, however, I decided against it.

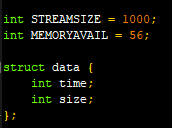
I decided to use an array of structures with time and size integers within them to make up the task stream. The memory would be generated and worked through inside of the functions with only the task stream being passed in with the parameters.

I started with First Fit. It was simple to set up, however, I wish I could go back and rewrite the function to be more like how I wrote the other three. For this assignment, this will suffice.

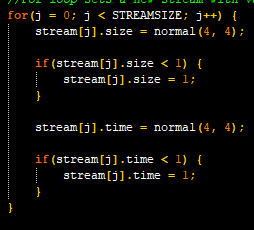
Best Fit and Worst Fit were next. Except for a few lines of code, they were pretty much the same. It was at this point where I decided to mark the spot in the array that I was currently checking with ‘index’ instead of using a boolean to see if it was all 0’s like I did with First Fit.

Next Fit may not have been done the correct way. Everywhere I looked for information, the pointer was just said to be within the partition. It never mentioned whether that was the beginning or the end of the partition. I assumed the end of the partition and set the pointer there.

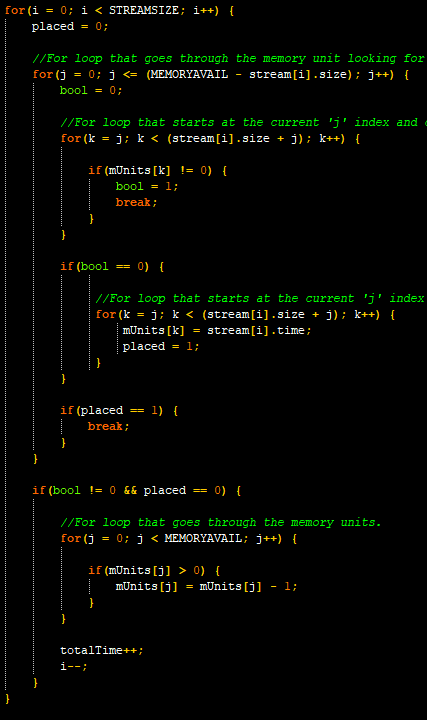
III. Process/Problems



**Structure:** I decided to use structures to keep the size and time information together. The size of the stream and the size of the memory were set by global variables so that it could be changed if needed.

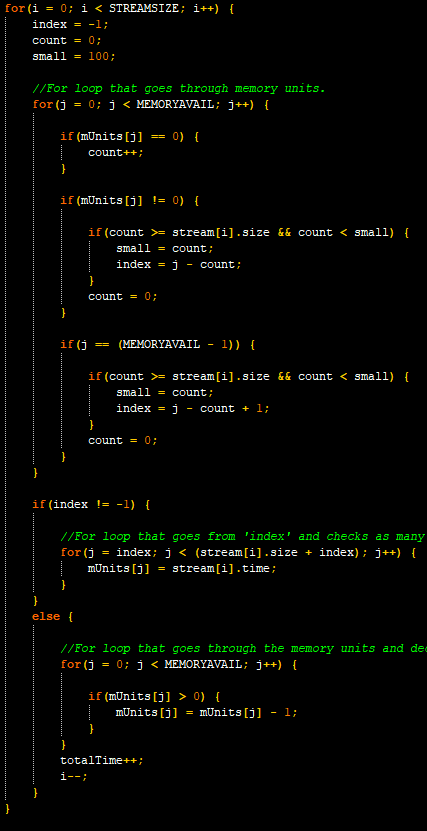


**Generate task stream:** This was given to us for the assignment. All size and time values will be between 1 and 16.

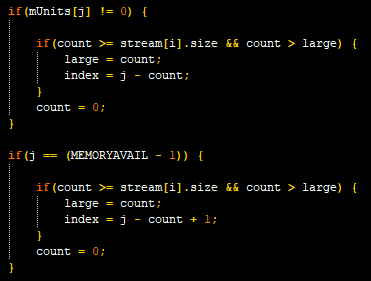


**First Fit:** This was the simplest and first function that I wrote. Problems that came up were mostly my not thinking of certain things that could happen. Using an online debugger to watch the values at each step and writing temporary code to print out values at every task, I was able to figure out my mistakes.

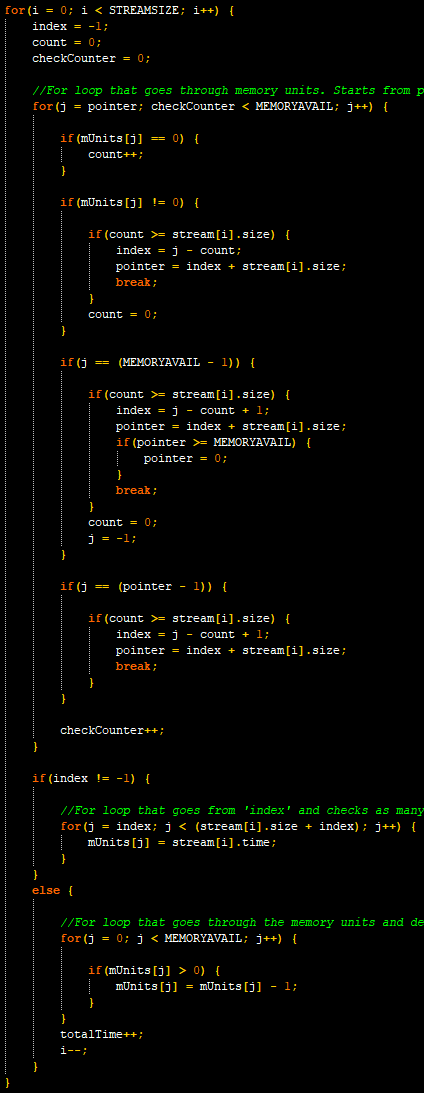
Note that compared to my later functions I used boolean values (‘bool’ and ‘placed’) instead of an ‘index’ value like the other functions.

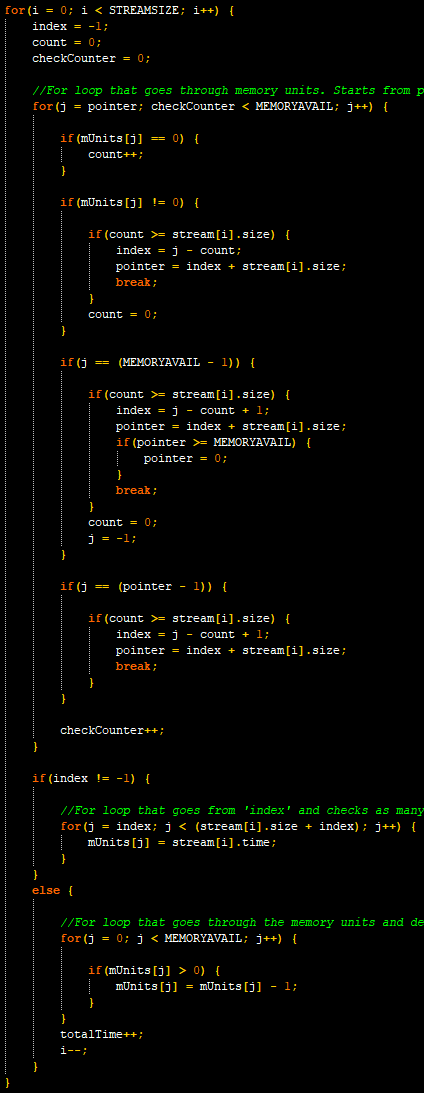


**Best Fit:** This function counts the units with a value of 0 until it reaches the end or reaches a non-zero. Then it compares how many it counted with the current ‘small’ value. This value must be larger than the size of the current task to count, though.

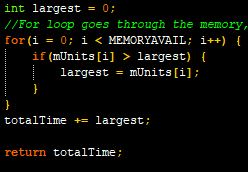


**Worst Fit:** This function is almost exactly like Best Fit except for the section pictured. Instead of looking for the smallest value that fits the size of the task, we look for the largest value that fits the size of the task.



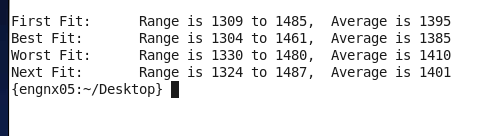


**Next Fit:** Next fit needed more IFs to check for the extra edge cases caused by the scan through the array starting at the pointer instead of at 0. To make sure I only check one way around I used ‘checkCounter’ to keep up with how many loops I did.



**Add rest of time:** Each of the functions has this at the end. It looks through the memory for the largest amount of time left and adds it to the ‘totalTime’ variable.

IV. Build/Final Thoughts



I decided to include the range of each algorithm in the output. Since the random() is unseeded the results are the same every time the program is ran.

According to my results, Best Fit is the best. However, I feel like Next Fit should be better than it ended up being. This could be due to my decision to set the pointer at the end of the partition.

Another issue with my implementation of Next Fit is that if the pointer was in the middle of a section of 0’s (after a few times through without placing a task) it wouldn’t count because it would stop counting once it reached the pointer. I could have maybe set up something that kept the count going from one pass to the next if it didn’t end with 0 but I thought this was against the idea of Next Fit.

In the end, the four algorithms are closer than I expected. This could be because they are all O(n) algorithms in the first place. It could also be because we only had 56 memory units in the first place. Most of the time only one section was opening up in the first place, so it didn’t matter where we put the new task.

If I were asked to use one, I would probably use First Fit or Best Fit. First Fit was easier to implement and had the second fastest times. Best fit was faster, and I would be OK with using it, too.

In the end I would probably use First Fit if given the choice. While it tends to load up the “front” of the array more than the “back”, I think this is a minor drawback for an easy to implement and reasonably fast algorithm.