Operating Systems and Systems Programming Lab
(15B11CI412)

Comparative analysis of Memory Management Algorithms

—

Group Members:

Shashank Agarwal	19104018	B11
Nakshatra Maharishi	19104026	B11
Sarthak Gupta	19104002	B11
Akshat Aggarwal	19104059	B11
Shubham Mishra	19104009	B11

—

Submitted To:

Dr Dharamveer Singh Rajpoot

Problem Statement

Execution time is critical in Operating Systems. A practical demonstration to prove the relevance of using certain algorithms in certain situations is thus helpful in understanding such decisions.

We aim to provide a comparative analysis of the different memory management algorithms, for a better understanding of how memory allocation works in an Operating System.

We have implemented and compared the execution time three basic memory management algorithms: first-fit, best-fit and worst-fit. Thus, we've tried to demonstrate the algorithm's working and contrasted them with relevant memory block count vs execution time graph.

Introduction

Memory management is a type of resource management that is used to manage the memory of a computer. A memory management technique must be able to dynamically assign sections of memory to programmes at their request, and then free it for reuse when it is no longer required.

Following are the basic algorithms used, which we have implemented in out project:

<u>First-Fit</u>

- This algorithm finds the first free partition or memory block large enough to accommodate the process.
- Advantage: It is the fastest algorithm because it stops the search as soon as it finds a suitable block.
- Disadvantage: May lead to too much internal fragmentation. As a result, requests for more RAM are unable to be fulfilled.

<u>Best-Fit</u>

- This algorithm is concerned with allocating the smallest free partition that matches the requesting process's requirements. This algorithm loops through the whole list of available partitions for the smallest suitable hole.
- Advantage: Much better memory utilization than first-fit because it searches the smallest free partition.
- Disadvantage: Extremely tiny holes may be left which will be unusable. Also, it is a time consuming method, as it iterates through all the partitions.

<u>Worst-Fit</u>

- This approach locates the largest available free partition and allocates the process there. It can be said to be the reverse of the best-fit approach.
- Advantage: Reduces the pace at which tiny gaps or partitions are produced.
- Disadvantage: Similarly to best-fit, it is a time consuming method. Besides that, if a process appears requiring a large memory, it is likely that such a partition has been made unavailable due to its nature.

Proposed work with tools used

Our memory allocation simulator is built with C++. It simulates the following common memory allocation strategies:

- First-Fit
- Best-Fit
- Worst-Fit

The result is presented in a table for convenience in recognizing the allocation. We have also written a separate code which will be used to evaluate execution time of the algorithms at various number of memory blocks, increased linearly.

Tools used:

- <chrono> library in C++ to evaluate execution time of respective algorithms.
- MatPlotLib library in Python to plot memory block count vs execution time line graph.

Working code:

- `memMngmt.cpp` :
  Contains the main code for our memory allocation simulator.
- `timeCalc.cpp` :
  Calculates the execution time of each method with block count as input.
- `plot.py` :
  Presents a line graph for the data collected from timeCalc.cpp.

The code has been hosted here:
https://github.com/Corbe30/Comparative-analysis-of-Memory-Management-Algorithms

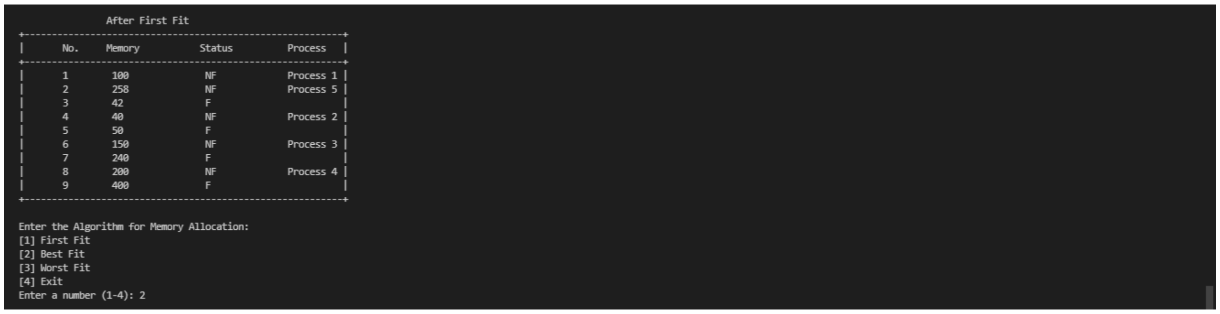
Result

Our simulator was successfully able to demonstrate the three memory allocation algorithms. The function for first-fit correctly allocates the input process in the first block with sufficient size, then stops immediately. Similarly, function for best-fit and worst-fit loop through the whole memory and print a table with expected values.

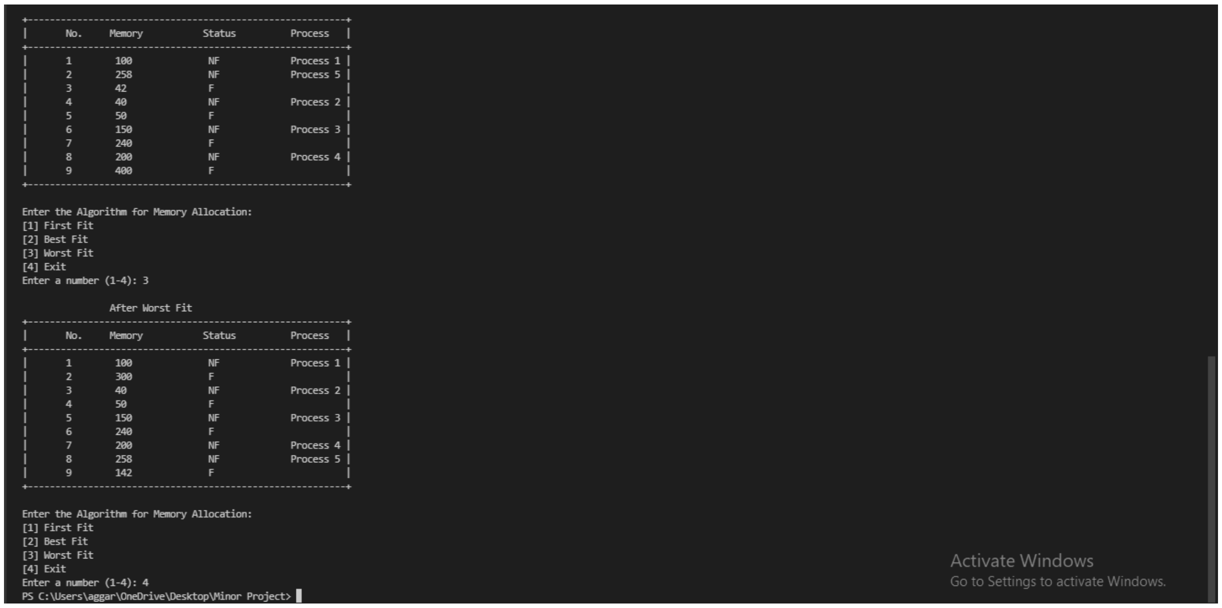
`timeCalc.cpp` showed a linear increase in best-fit and worse-fit function's execution time, thus practically demonstrating their disadvantage as a time consuming allocation strategy. This result has also been highlighted in the line graph we implemented in `plot.py` using MatPlotLib library.

Snapshots of Results

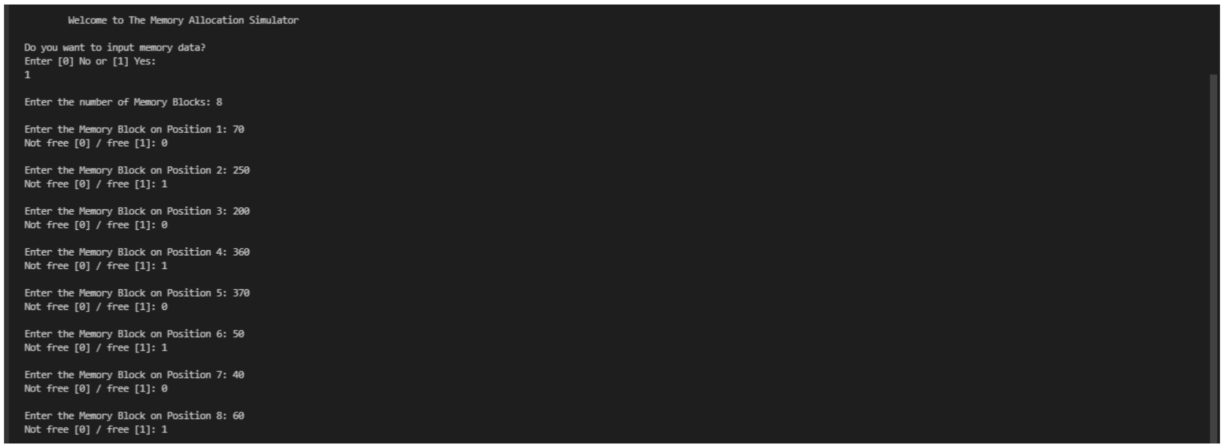
Output of memMngmt.cpp:



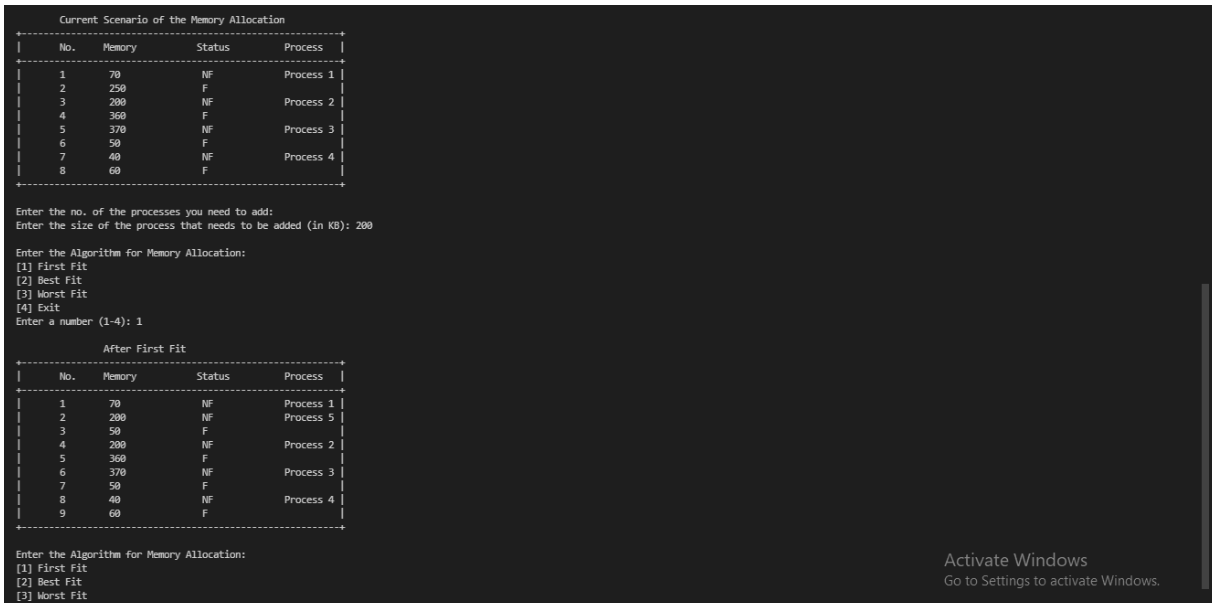
```
              After First Fit
+------------------------------------------------------+
|      No.    Memory          Status         Process   |
+------------------------------------------------------+
|      1       100             NF            Process 1 |
|      2       258             NF            Process 5 |
|      3       42              F                       |
|      4       40              NF            Process 2 |
|      5       50              F                       |
|      6       150             NF            Process 3 |
|      7       240             F                       |
|      8       200             NF            Process 4 |
|      9       400             F                       |
+------------------------------------------------------+

Enter the Algorithm for Memory Allocation:
[1] First Fit
[2] Best Fit
[3] Worst Fit
[4] Exit
Enter a number (1-4): 2
```



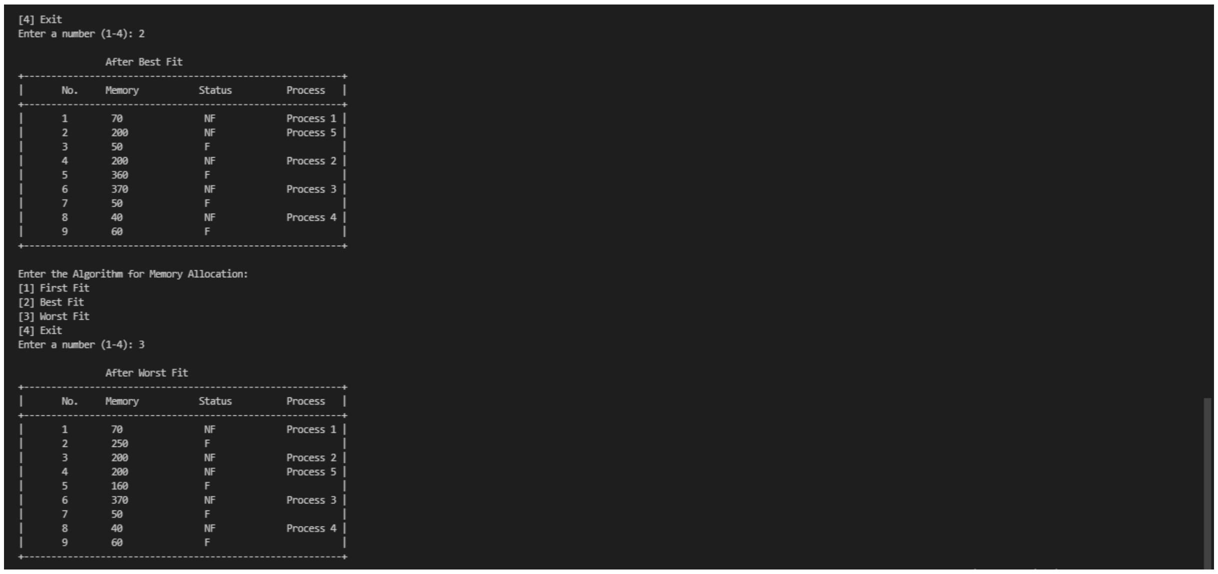
```
+------------------------------------------------------+
|      No.    Memory          Status         Process   |
+------------------------------------------------------+
|      1       100             NF            Process 1 |
|      2       258             NF            Process 5 |
|      3       42              F                       |
|      4       40              NF            Process 2 |
|      5       50              F                       |
|      6       150             NF            Process 3 |
|      7       240             F                       |
|      8       200             NF            Process 4 |
|      9       400             F                       |
+------------------------------------------------------+

Enter the Algorithm for Memory Allocation:
[1] First Fit
[2] Best Fit
[3] Worst Fit
[4] Exit
Enter a number (1-4): 3

              After Worst Fit
+------------------------------------------------------+
|      No.    Memory          Status         Process   |
+------------------------------------------------------+
|      1       100             NF            Process 1 |
|      2       300             F                       |
|      3       40              NF            Process 2 |
|      4       50              F                       |
|      5       150             NF            Process 3 |
|      6       240             F                       |
|      7       200             NF            Process 4 |
|      8       258             NF            Process 5 |
|      9       142             F                       |
+------------------------------------------------------+

Enter the Algorithm for Memory Allocation:
[1] First Fit
[2] Best Fit
[3] Worst Fit
[4] Exit
Enter a number (1-4): 4
PS C:\Users\aggar\OneDrive\Desktop\Minor Project>
```



```
        Welcome to The Memory Allocation Simulator

Do you want to input memory data?
Enter [0] No or [1] Yes:
1

Enter the number of Memory Blocks: 8

Enter the Memory Block on Position 1: 70
Not free [0] / free [1]: 0

Enter the Memory Block on Position 2: 250
Not free [0] / free [1]: 1

Enter the Memory Block on Position 3: 200
Not free [0] / free [1]: 0

Enter the Memory Block on Position 4: 360
Not free [0] / free [1]: 1

Enter the Memory Block on Position 5: 370
Not free [0] / free [1]: 0

Enter the Memory Block on Position 6: 50
Not free [0] / free [1]: 1

Enter the Memory Block on Position 7: 40
Not free [0] / free [1]: 0

Enter the Memory Block on Position 8: 60
Not free [0] / free [1]: 1
```

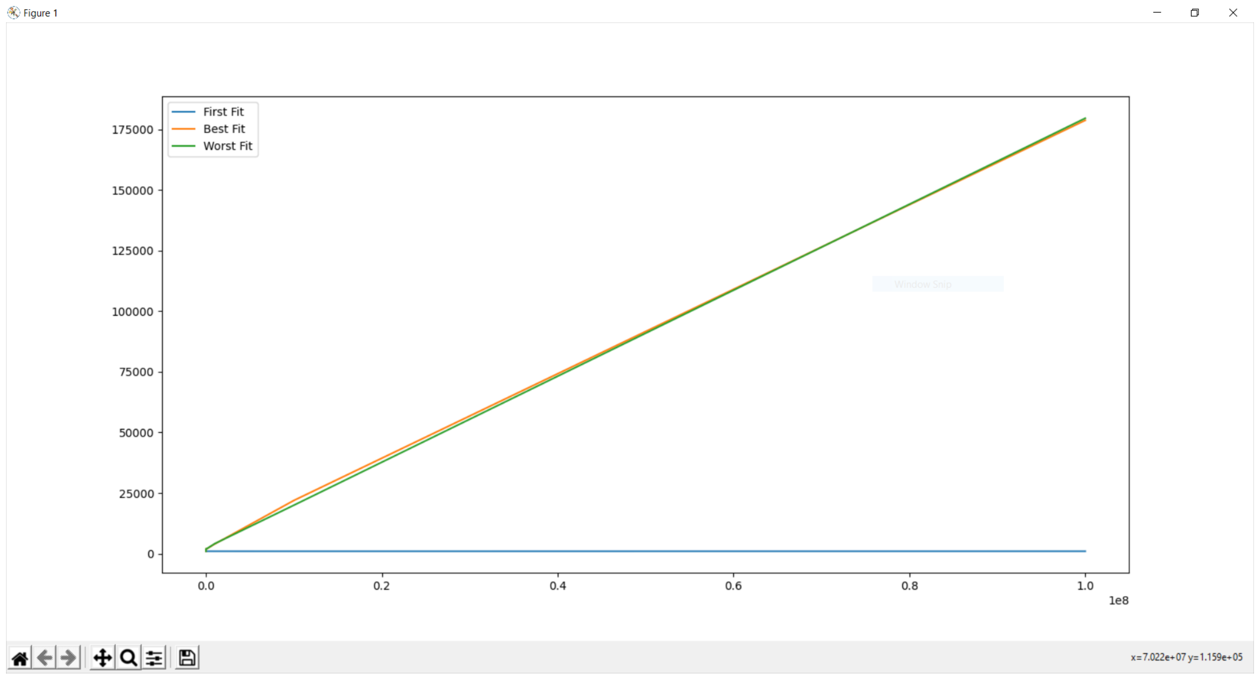


```
        Current Scenario of the Memory Allocation
+--------------------------------------------------------+
|       No.    Memory          Status         Process   |
+--------------------------------------------------------+
|       1       70              NF            Process 1 |
|       2       250             F                       |
|       3       200             NF            Process 2 |
|       4       360             F                       |
|       5       370             NF            Process 3 |
|       6       50              F                       |
|       7       40              NF            Process 4 |
|       8       60              F                       |
+--------------------------------------------------------+

Enter the no. of the processes you need to add:
Enter the size of the process that needs to be added (in KB): 200

Enter the Algorithm for Memory Allocation:
[1] First Fit
[2] Best Fit
[3] Worst Fit
[4] Exit
Enter a number (1-4): 1

                After First Fit
+--------------------------------------------------------+
|       No.    Memory          Status         Process   |
+--------------------------------------------------------+
|       1       70              NF            Process 1 |
|       2       200             NF            Process 5 |
|       3       50              F                       |
|       4       200             NF            Process 2 |
|       5       360             F                       |
|       6       370             NF            Process 3 |
|       7       50              F                       |
|       8       40              NF            Process 4 |
|       9       60              F                       |
+--------------------------------------------------------+

Enter the Algorithm for Memory Allocation:
[1] First Fit
[2] Best Fit
[3] Worst Fit
```



```
[4] Exit
Enter a number (1-4): 2

                After Best Fit
+--------------------------------------------------------+
|       No.    Memory          Status         Process   |
+--------------------------------------------------------+
|       1       70              NF            Process 1 |
|       2       200             NF            Process 5 |
|       3       50              F                       |
|       4       200             NF            Process 2 |
|       5       360             F                       |
|       6       370             NF            Process 3 |
|       7       50              F                       |
|       8       40              NF            Process 4 |
|       9       60              F                       |
+--------------------------------------------------------+

Enter the Algorithm for Memory Allocation:
[1] First Fit
[2] Best Fit
[3] Worst Fit
[4] Exit
Enter a number (1-4): 3

                After Worst Fit
+--------------------------------------------------------+
|       No.    Memory          Status         Process   |
+--------------------------------------------------------+
|       1       70              NF            Process 1 |
|       2       250             F                       |
|       3       200             NF            Process 2 |
|       4       200             NF            Process 5 |
|       5       160             F                       |
|       6       370             NF            Process 3 |
|       7       50              F                       |
|       8       40              NF            Process 4 |
|       9       60              F                       |
+--------------------------------------------------------+
```

Line Graph:
x-axis: memory block count
y-axis: execution time (in ms)



Conclusions

From the project, we conclude that the first-fit algorithm works best out of the three- the first fit, worst fit, and best fit in terms of execution time. This fact is evident from the graph, which shows linear increase of execution time in best-fit and worst-fit technique.

Although the memory utilization in best-fit is better than the first-fit and worst-fit. Likewise, worst-fit reduces the pace at which tiny gaps or partitions are produced.
But the overall first-fit algorithm is better than best-fit and first-fit because of its much better execution time which is the most important factor when talking about the real world performance of a machine.