

Programming Assignment 1

Report/Analysis

Hard Patel

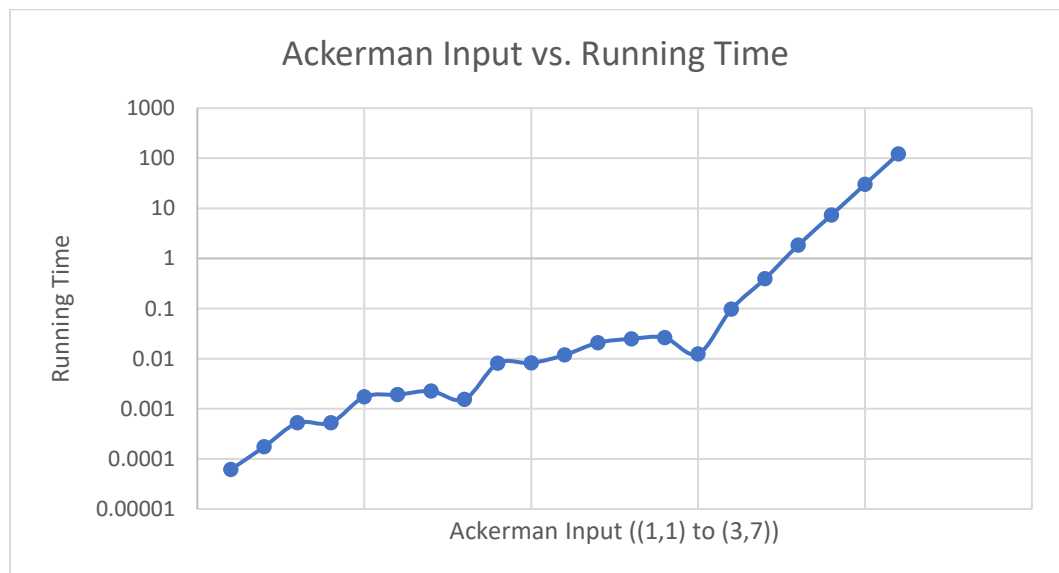
1. Introduction

- a. This assignment is designed to help understand how memory can be used to allocate free memory and free allocated memory. It required us to complete coding the operations `alloc()` and `free()`, including the other operations that may be used by these two operations. Ackerman was used to test the program by creating n number of `alloc/free` memory cycles to obtain the time taken for the different inputs of values 'n' and 'm'. Overall, the assignment required a concrete understanding of how memory works.

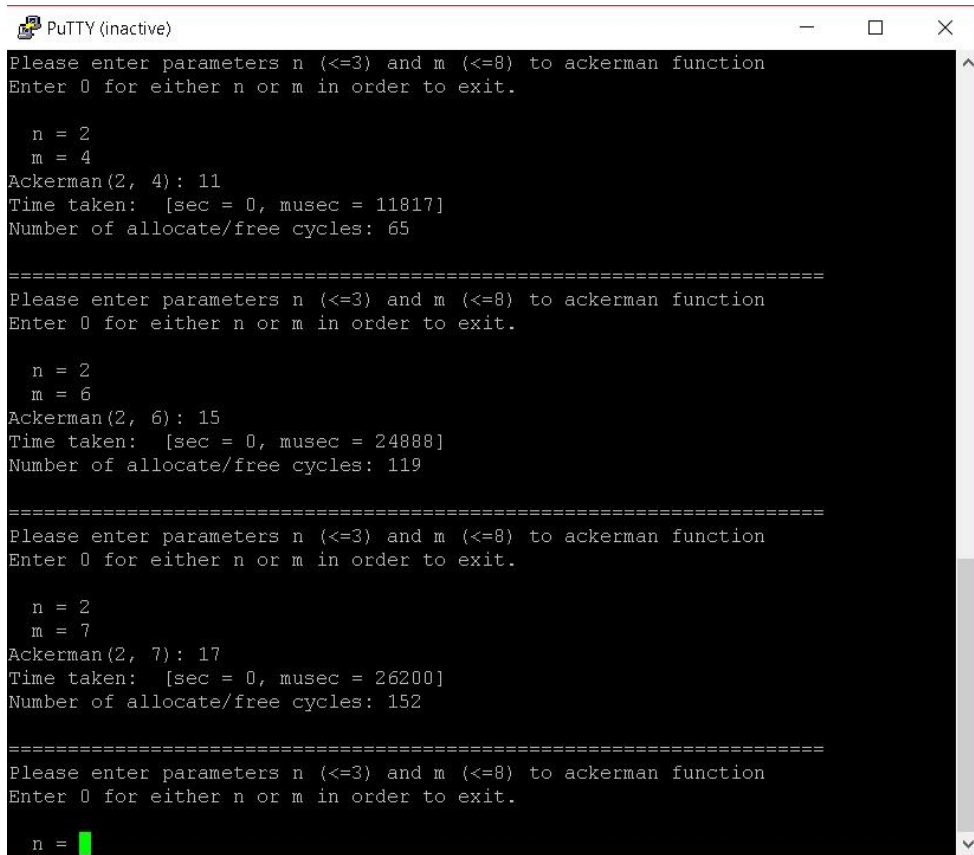
2. Table

m\n(Running Time)	1	2	3
1	0.000061	0.001540	0.012434
2	0.000174	0.008045	0.097101
3	0.000522	0.008217	0.390667
4	0.000528	0.011817	1.844606
5	0.001730	0.020808	7.385863
6	0.001914	0.024888	30.022602
7	0.002256	0.026200	121.037995

3. Graph



4. Example (Command Window)



```
PuTTY (inactive)
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

n = 2
m = 4
Ackerman(2, 4): 11
Time taken: [sec = 0, musec = 11817]
Number of allocate/free cycles: 65

=====
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

n = 2
m = 6
Ackerman(2, 6): 15
Time taken: [sec = 0, musec = 24888]
Number of allocate/free cycles: 119

=====
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

n = 2
m = 7
Ackerman(2, 7): 17
Time taken: [sec = 0, musec = 26200]
Number of allocate/free cycles: 152

=====
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

n = 
```

5. Analysis

- a. The increase in the number of cycles had a direct effect on the performance of the allocator. The running time was greater for higher number of cycles as can also be seen in the graph above. The increase in the number of cycles was an effect of the higher input 'n' and 'm' values. The bottleneck in this case is that both operations alloc() and free() can only run one at a time and therefore the high running time. For example, free has to wait for alloc to use the memory before it can free it.
- b. The Ackerman was giving issues since the memory was trying to be accessed out of bounce. Although, those errors were fixed and with the right basic_block_size/total_memory_size, the Ackerman ran in perfect condition and outputted the right values.
- c. The getopt is implemented in the Main.cpp and is built so that the input for basic_block_size/total_memory_size can be given from the command line. The default is set as: basic_block_size = 128 B, total_memory_size = 512 kB.

6. Conclusion

- a. In conclusion, the assignment was a good practice to help understand how memory allocation works and the benefits of keeping of the space used to be more efficient.