Observing the difference between memory access and experience swapping

Submitted by: Md Ishtiaq Ahmed

Abstract

This experiment's objective is to observe the effects of extensive physical memory consumption and swapping during program execution time. A ‘C’ program is designed which incrementally consumes all available physical memory and eventually forces the system to utilize swap space. The experiment is performed across three different platforms to understand the behavior of each platform as memory management is done differently in different platforms.

1. Introduction

SWAP memory or swap space is a section of a computer's hard disk or SSD that the OS uses to store data from RAM. This allows the OS to run a program even if the RAM is full [1]. If a process is too large for the available physical memory or if there are multiple processes that require more than the physical memory, the OS will utilize the swap space to allow the process to run. The OS does this by moving some segments of the memory which are not actively in use to swap space. This is called swapping out. When the process needs to access that data, the OS will swap out some data to swap space and swap in the needed data back into RAM. This is called swapping in. In this experiment I used a C program that grows by 500 MB in each iteration**.** Eventually the program consumes all the physical memory and when the physical memory is full then the swapping starts to give space to this program to run. Usually accessing swap space is more time consuming than accessing the physical memory. So, when swapping starts, the program execution experiences delay. By this experiment we will observe when swapping start and what is the execution time difference between when swapping is involved.

2. Methodology

I used a program that will allocate a memory for an array and fill up the array with integer ‘1’. The program runs in an infinite loop and in each loop, it allocates memory and fills up the array. In each iteration it consumes 500 MB of memory. Also, before creating and filling up a new array, the program revisited the older arrays to ensure all allocated memories are actively in use. Execution time is measured for creating the latest array. When the physical memory is free, there should not be any impact on the execution time but when the RAM is full and swapping starts, the time measured will include the delays caused by swapping. So, by measuring this time we can differentiate between the execution time before and after swapping starts.

3. Platforms & Experiment Design

The program is run on three different platforms to observe the difference of memory management by each platform.

Platforms used in this experiment:

i. Poobah.nmsu.edu
   Processor: Intel® Core™ i7-2600 CPU @ 3.40GHz
   Operating System: Linux
   Number of Cores: 4
   Total Physical Memory: 16 GB
   Total SWAP Space: 4 GB
   Available space is determined by running ‘top’ command

ii. Acer Nitro 5
   Processor: 12th Gen Intel® Core™ i7-12650
   Operating System: Windows 11 Home, 64 bit
   Number of Cores: 10
   Total Physical Memory: 16 GB
   Total SWAP Space/Virtual memory: 48 GB
   Available space is determined by running ‘systeminfo’ command

iii. MacBook pro
   Processor: M1 Operating System: OSX
   Number of Cores: 8
   Total Physical Memory: 16 GB
   Total SWAP Space: 4 GB
   Available space is from the system monitor tool

The C program used in this experiment; size of each array is configured as 131072000**.** The array is to be filled with integers and the size of an integer is 4 bytes. So, memory allocation for each array is 131072000 * 4 /1024 /1024 which is 500 MB. Before running the program, available physical memory and swap space is noted. Then I ran the program and observed the memory consumption as the program was being executed. In Linux it is possible to observe the live consumption of memory but in the Windows and Mac machine, we cannot observe the memory consumption from the command prompt live, instead we need to look at resource monitor which does not provide a pleasant view like what Linux system does.

4. Results

See Appendices for program output.

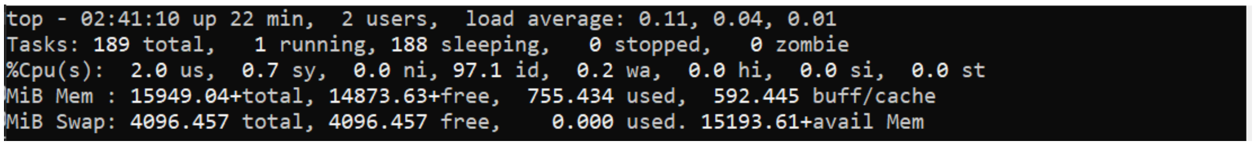
5. Data Analysis

The result demonstrates how the execution time for creating and filling up the latest array differs when swapping is involved. Ideally, when physical memory is free, the execution time is the same for each iteration. But when swapping starts, the execution time increases. I got the expected behavior from the program execution in Linux. In windows initially unable to get expected result but later found a way to get expected result. However, the behavior in Mac OS is different. As I have limited access to a Mac OS, I was unable to troubleshoot or analysis performance behavior extensively.

5.1 Performance on Linux Machine

Poobah.cs.nmsu.edu is used to conduct this experiment as a Linux platform. As soon as the program started running, the available physical memory started to reduce by the size around 500 MB which is like the space the program supposed to be consume in each iteration. During this time, the execution time to fill up the latest array is the same. Once the physical memory does not have enough space to accommodate the program, swapping started. And from this time, I observed significant delay is started and execution time is also higher. Once the swap space is full, the program was killed. The scenario is as expected as described in the article titled “Swap Memory: What It Is, How It Works, and How to Manage It” [1].

Available system resource before execution the program:
Command: top



```
top - 02:41:10 up 22 min,  2 users,  load average: 0.11, 0.04, 0.01
Tasks: 189 total,   1 running, 188 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.0 us,  0.7 sy,  0.0 ni, 97.1 id,  0.2 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem : 15949.04+total, 14873.63+free,  755.434 used,  592.445 buff/cache
MiB Swap: 4096.457 total, 4096.457 free,    0.000 used. 15193.61+avail Mem
```

Figure 1: Available Memory in Linux Machine

From the chart below, we can see that when the total consumption of memory reached the limit of physical memory, execution time increased. When the total memory was full, the program was killed.

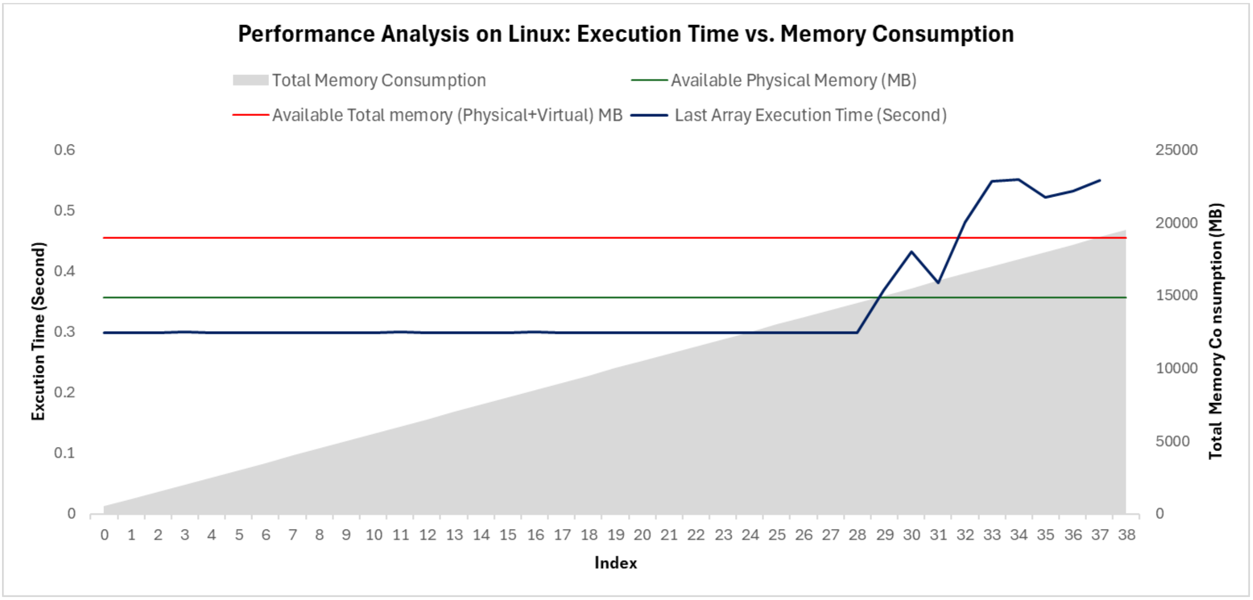
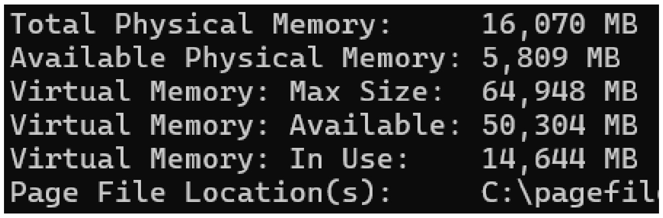


Figure 2: Performance Analysis on Linux

5.2 Performance on Windows

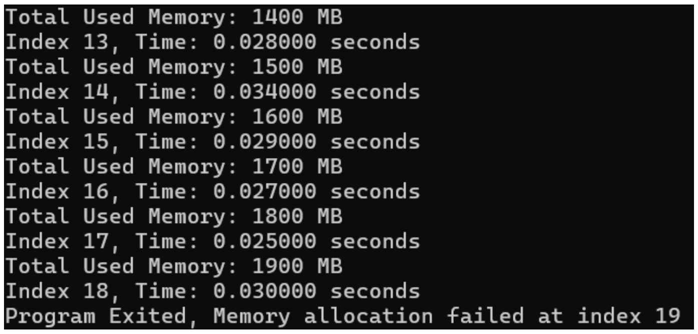
I observed a completely different outcome when running the program on my windows machine. Although my laptop has 16 GB of main memory and 64 GB of total memory (including physical memory), when the program consumed 2 GB of available memory it suddenly stopped. I tried with a different array size from smaller to larger, but the output was the same. I could not be able to use the rest of the physical memory or the swap space using the program. After some research I found the reason behind this is that the C compiler I was using is '32-bit gcc' which is compiling the program as a 32-bit process. As per the windows documentation [2], a single 32-bit process is limited to 2 GB space. In total max 4 GB can be allocated where 2 GB is reserved for user space and 2 GB for kernel space [2][3].

Below figure shows that the program was terminated after consuming around 2GB of memory while there was plenty of memory available.



```
Total Physical Memory:     16,070 MB
Available Physical Memory: 5,809 MB
Virtual Memory: Max Size:  64,948 MB
Virtual Memory: Available: 50,304 MB
Virtual Memory: In Use:    14,644 MB
Page File Location(s):     C:\pagefil
```

Figure 3: Available Memory

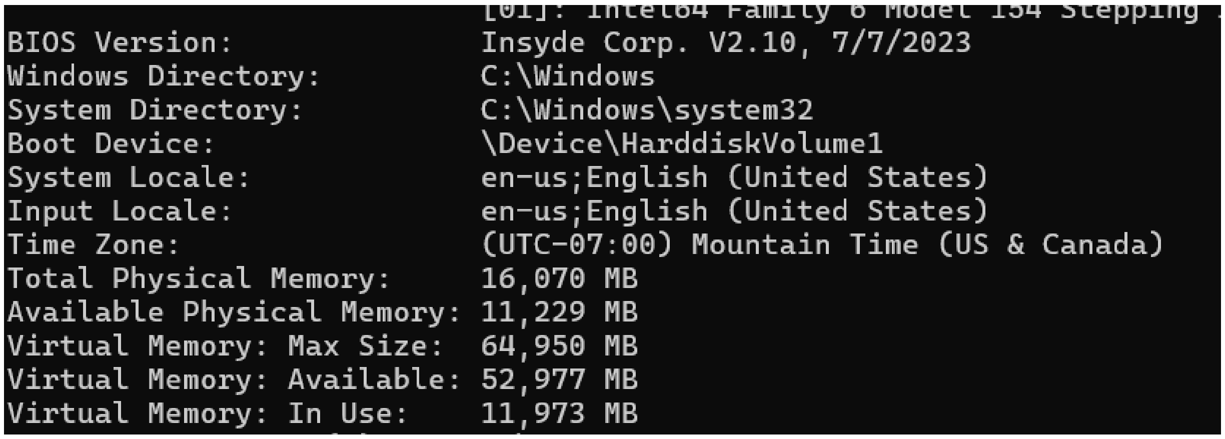


```
Total Used Memory: 1400 MB
Index 13, Time: 0.028000 seconds
Total Used Memory: 1500 MB
Index 14, Time: 0.034000 seconds
Total Used Memory: 1600 MB
Index 15, Time: 0.029000 seconds
Total Used Memory: 1700 MB
Index 16, Time: 0.027000 seconds
Total Used Memory: 1800 MB
Index 17, Time: 0.025000 seconds
Total Used Memory: 1900 MB
Index 18, Time: 0.030000 seconds
Program Exited, Memory allocation failed at index 19
```

Figure 4: program Terminated after consuming 1900 MB of Memory

Later I used a 64-bit compiler, and I got the expected result which is like the result I obtained from Linux platform.

Available system resource before execution the program:
Command: systeminfo



```
                          [01]: Intel64 Family 6 Model 154 Stepping
BIOS Version:             Insyde Corp. V2.10, 7/7/2023
Windows Directory:        C:\Windows
System Directory:         C:\Windows\system32
Boot Device:              \Device\HarddiskVolume1
System Locale:            en-us;English (United States)
Input Locale:             en-us;English (United States)
Time Zone:                (UTC-07:00) Mountain Time (US & Canada)
Total Physical Memory:    16,070 MB
Available Physical Memory: 11,229 MB
Virtual Memory: Max Size:  64,950 MB
Virtual Memory: Available: 52,977 MB
Virtual Memory: In Use:    11,973 MB
```

Figure 5: Available memory in Windows machine

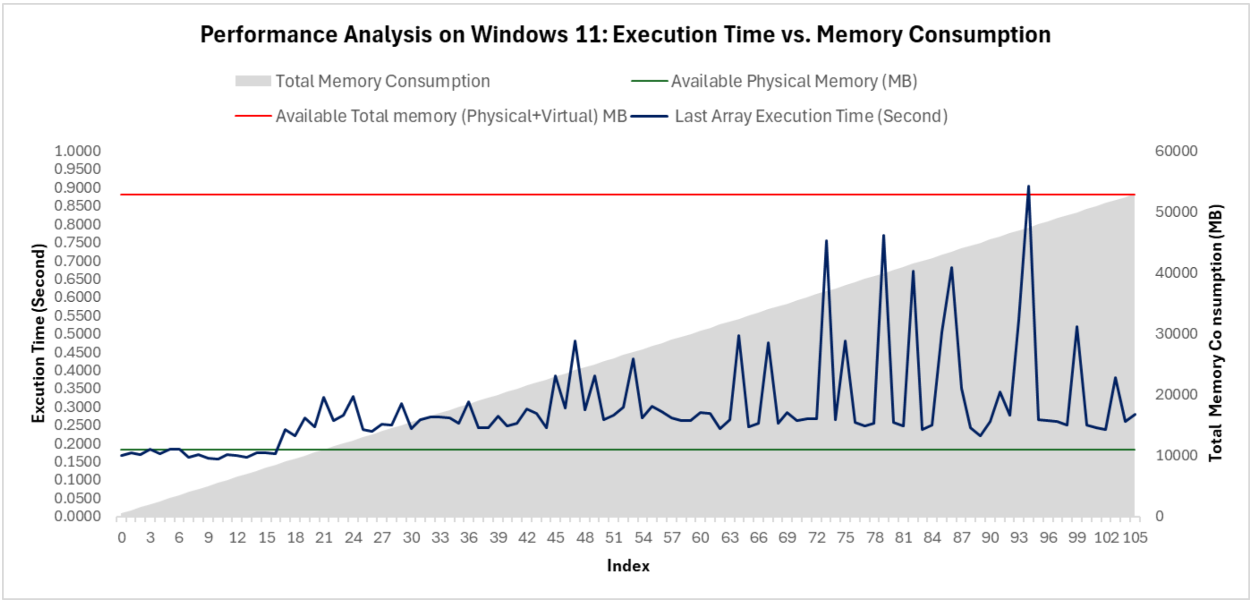


Figure 6: Performance analysis on Windows 11

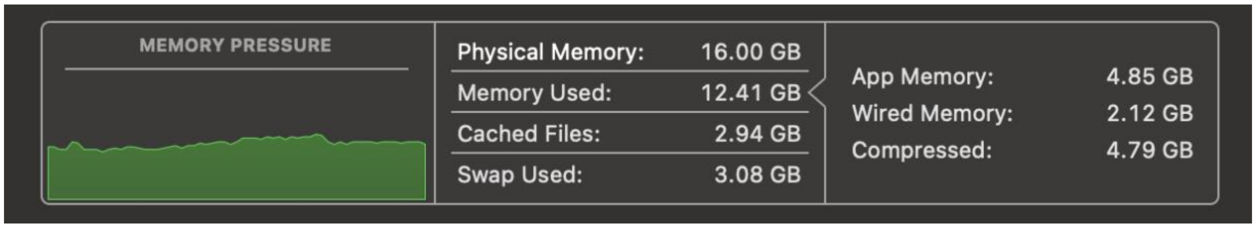
From the above chart, we can see that just before the memory consumption hit the total physical memory line, the execution time increased sharply. This is the time when swapping started. Swapping started a little bit earlier because some other background program consumed some memory after the program execution started. During, there are many ups and downs in the execution time, but it always remains higher than the time before swapping. From my observations from live memory monitoring, it is that, during swapping if more memory freed from the physical memory, then the execution time is lower if available physical memory is less, execution time is higher. When the total consumption reached the total available memory line, the program terminated.

5.3 Performance on Mac OS

I used one of my friend's computers to conduct the experiments in a Mac OS. My access to the system was limited hence I could not do much analysis using the Mac platform.

Available system resource before execution the program:

System monitor was used to see available memory



Physical Memory:	16.00 GB
Memory Used:	12.41 GB
Cached Files:	2.94 GB
Swap Used:	3.08 GB
App Memory:	4.85 GB
Wired Memory:	2.12 GB
Compressed:	4.79 GB

Figure 7: Available memory in Mac OS

Before starting the program, available physical memory was around 2.8 GB. When the available physical memory was consumed, it started to access the swap memory but no significant change was observed in the execution time. Most probable reason is that memory management in Mac OS is more efficient than Linux and Windows. According to one of the apple forums, Mac OS preemptively copies data from memory to swap, so if it needs more RAM, it can access the physical memory more quickly [4].

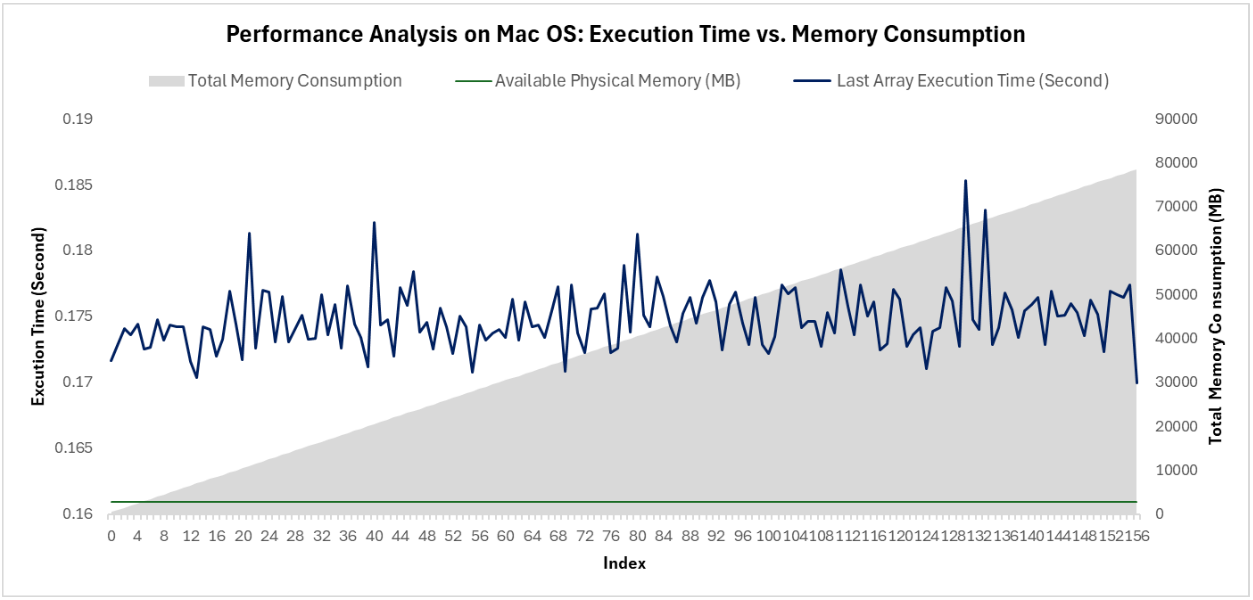
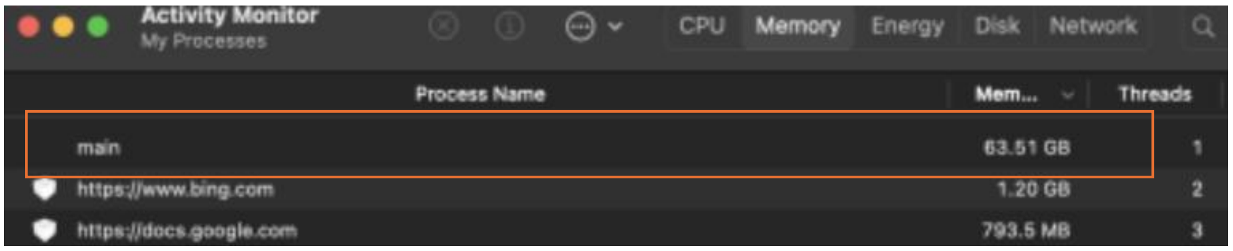


Figure 8: Performance analysis on Mac OS X

From the above plot we can also see that the total memory consumption by the program is around 80 GB which is confusing because total available memory from the system was much less than that. According to apple documentation Mac OS X uses a portion of the hard disk to hold data that are not currently in use, this is called backing store. OS X does not use a pre-allocated disk partition for the backing store. Instead, it uses all the available space on the machine’s boot partition [5]. It might be the reason why the program can access such a high volume of memory.



Process Name	Mem...	Threads
main	63.51 GB	1
https://www.bing.com	1.20 GB	2
https://docs.google.com	793.5 MB	3

Figure 9: From The activity monitor, program consumed more than 50 GB memory.

5.4 Performance Degradation

Another finding is that, when swapping started there was not only a delay in the execution time of filling up the last array but also results in an overall performance degradation when physical memory was almost full. To prove this, I edited the program to measure total execution time of each iteration which includes filling both the previous and current array with integers. I ran this program in my windows machine and below is the change in execution time:

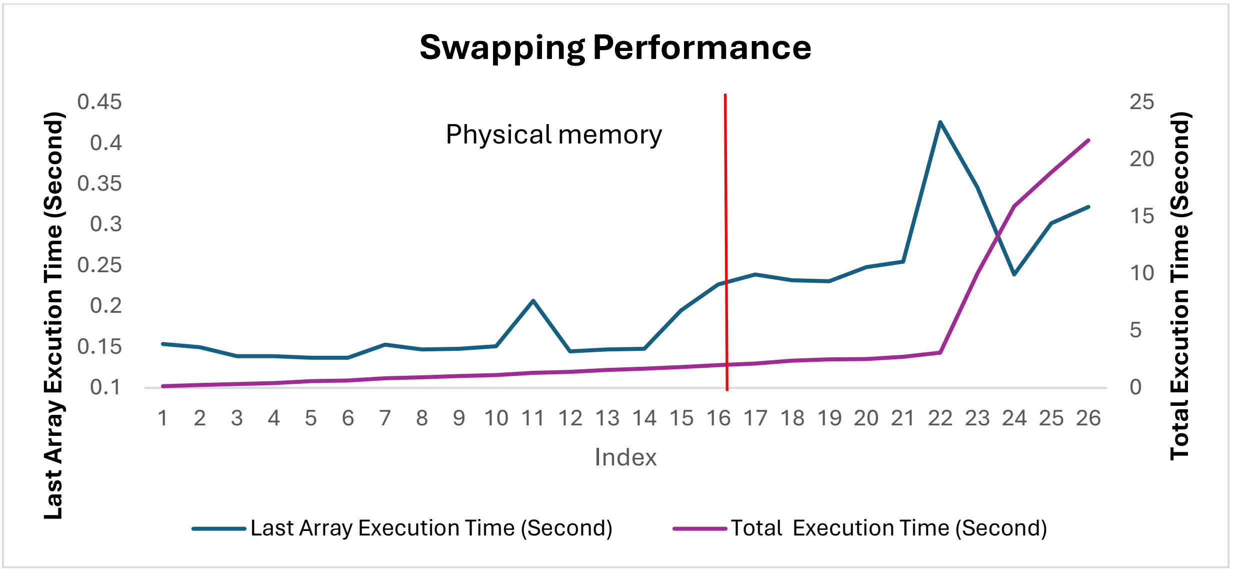


Figure 10: Total Execution time of Each Iteration

We can see from the chart, when physical memory is full, execution time increased for filling up the last array, as well as the total execution time of each iteration increased significantly.

Although swapping cause a delay in program execution it allows program to run which needs significant memory that is more than the physical memory. One of the advantages of swapping as per the article “Swapping: A Guide To Memory Management In Computer Operating Systems” [6] is that swapping allows handling of large workload. That is exactly what we observed here.

6. Conclusion

From the above observation we can see that the different platforms displayed different behavior. All three-systems started consuming physical memory and when physical memory was finished it started swapping. However, there was a change in the execution time between the systems as different OS managed their memory differently. For Linux and Windows, when the swapping started the program started to experience a significant delay. It is because when the swapping started, a chunk of data was transferred to the swap space to free up memory. Usually, the swap space is in the hard disk/SSD and read/write operation in hard disk takes significantly longer time than the read/write operation of ram. So, when the system starts swapping, it involves read/write operation in the disk which is slower than accessing RAN. This delay was added to the time measurement which was used to measure execution time. At the end, the program terminates when it has fully consumed all the available physical and swap memory and the OS can no longer allocate additional memory for the growing array. This condition arises when the MMU cannot find any more space to allocate which leads to a failure in the memory allocation (malloc) request.

References

[1] Swap Memory: What It Is & How It Works (phoenixnap.com)

[2] Memory Limits for Windows and Windows Server Releases - Win32 apps | Microsoft Learn

[3] What is the maximum memory available to a C++ application on 32-bit Windows? - Stack Overflow

[4] macbook pro - is this swap memory use normal? - Ask Different (stackexchange.com)

[5] About the Virtual Memory System (apple.com)

[6] Swapping: A Guide to Memory Management in Computer Operating Systems – Linux Bierwanderung

Appendices

A. [C code]

```
/*

Assignment 2: Observing the difference between memory access and experience swapping

Submitted by: Md Ishtiaq Ahmed

Objective: Run a program to observe the effects of extensive physical memory consumption and swapping during program execution time.

A C program is designed which incrementally consumes all available physical memory and eventually forces the system to utilize swap space.

*/
```

```
#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <unistd.h>


//#define SIZE 26214400  // size of each array (100*1024*1024) , when filled with
integer of size 4 byte total consumption will be 100 MB

#define SIZE 131072000  // size of each array, when filled with integer of size 4
byte total consumption will be 500 MB




int *A[1000]; // Max Number of array


// Pre Condition: Define a size which will be used for memory allocation of each
array, also define the array

// Post COndition: Program will run until it consumes all of the available memory


int main() {


    long long memoryAllocated = 0; // To keep track of the total allocated memory

    int index = 0; // initial index of array set as 0


    clock_t start, end;  // To calculate execution time

    double cpu_time_used;  // Variable to store execution time




    while (1) {

        A[index] = malloc(SIZE * sizeof(int));  // memory allocation for each array,
each array will have a size of 26214400 and filled by integer of 4 byte, total 100
MB
```

```
        if (A[index] == NULL) { // If out of memory, program will be exited
            printf("Program  Exited,  Memory  allocation  failed  at  index  %d\n",
index);
            break; // Exit if we can't allocate more memory
        }

        // Fill previous arrays with 1
        for (int i = 0; i < index; i++) {
            for (int j = 0; j < SIZE; j++) {
                A[i][j] = 1;
            }
        }
        memoryAllocated  +=  (SIZE  *  sizeof(int))/(1024*1024);  //  total  memory
allocated after each iteration
        printf("Total Used Memory: %lld MB\n", memoryAllocated);

        // Measure time to fill the latest array
        start = clock(); // Start time
        for (int j = 0; j < SIZE; j++) {  // Fill up the last array
            A[index][j] = 1;
        } // end for
        end = clock();  // End time

        cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
        printf("Index %d, Time: %f seconds\n", index, cpu_time_used);

        index++;
    } // end while

    // Free allocated memory
    for (int i = 0; i < index; i++) {
        free(A[i]);
    } // end for

    return 0;
} // end main
```

B. Program Output in Windows Platform

```
PS C:\Users\ishti\OneDrive\OS_II\Assignment2\Final Submission> ./memory
Total Used Memory: 500 MB
Index 0, Time: 0.168000 seconds
Total Used Memory: 1000 MB
Index 1, Time: 0.174000 seconds
Total Used Memory: 1500 MB
Index 2, Time: 0.170000 seconds
Total Used Memory: 2000 MB
Index 3, Time: 0.183000 seconds
Total Used Memory: 2500 MB
Index 4, Time: 0.172000 seconds
Total Used Memory: 3000 MB
Index 5, Time: 0.183000 seconds
Total Used Memory: 3500 MB
Index 6, Time: 0.184000 seconds
Total Used Memory: 4000 MB
Index 7, Time: 0.162000 seconds
Total Used Memory: 4500 MB
Index 8, Time: 0.170000 seconds
Total Used Memory: 5000 MB
Index 9, Time: 0.160000 seconds
Total Used Memory: 5500 MB
Index 10, Time: 0.158000 seconds
Total Used Memory: 6000 MB
Index 11, Time: 0.170000 seconds
Total Used Memory: 6500 MB
Index 12, Time: 0.166000 seconds
Total Used Memory: 7000 MB
Index 13, Time: 0.162000 seconds
Total Used Memory: 7500 MB
Index 14, Time: 0.175000 seconds
Total Used Memory: 8000 MB
Index 15, Time: 0.174000 seconds
Total Used Memory: 8500 MB
Index 16, Time: 0.173000 seconds
Total Used Memory: 9000 MB
Index 17, Time: 0.239000 seconds
Total Used Memory: 9500 MB
Index 18, Time: 0.221000 seconds
Total Used Memory: 10000 MB
Index 19, Time: 0.270000 seconds
Total Used Memory: 10500 MB
```

```
Index 20, Time: 0.245000 seconds
Total Used Memory: 11000 MB
Index 21, Time: 0.327000 seconds
Total Used Memory: 11500 MB
Index 22, Time: 0.262000 seconds
Total Used Memory: 12000 MB
Index 23, Time: 0.278000 seconds
Total Used Memory: 12500 MB
Index 24, Time: 0.328000 seconds
Total Used Memory: 13000 MB
Index 25, Time: 0.239000 seconds
Total Used Memory: 13500 MB
Index 26, Time: 0.233000 seconds
Total Used Memory: 14000 MB
Index 27, Time: 0.253000 seconds
Total Used Memory: 14500 MB
Index 28, Time: 0.251000 seconds
Total Used Memory: 15000 MB
Index 29, Time: 0.308000 seconds
Total Used Memory: 15500 MB
Index 30, Time: 0.240000 seconds
Total Used Memory: 16000 MB
Index 31, Time: 0.266000 seconds
Total Used Memory: 16500 MB
Index 32, Time: 0.272000 seconds
Total Used Memory: 17000 MB
Index 33, Time: 0.272000 seconds
Total Used Memory: 17500 MB
Index 34, Time: 0.270000 seconds
Total Used Memory: 18000 MB
Index 35, Time: 0.256000 seconds
Total Used Memory: 18500 MB
Index 36, Time: 0.313000 seconds
Total Used Memory: 19000 MB
Index 37, Time: 0.242000 seconds
Total Used Memory: 19500 MB
Index 38, Time: 0.243000 seconds
Total Used Memory: 20000 MB
Index 39, Time: 0.275000 seconds
Total Used Memory: 20500 MB
Index 40, Time: 0.249000 seconds
Total Used Memory: 21000 MB
Index 41, Time: 0.255000 seconds
Total Used Memory: 21500 MB
Index 42, Time: 0.294000 seconds
Total Used Memory: 22000 MB
Index 43, Time: 0.282000 seconds
Total Used Memory: 22500 MB
Index 44, Time: 0.242000 seconds
Total Used Memory: 23000 MB
Index 45, Time: 0.386000 seconds
Total Used Memory: 23500 MB
Index 46, Time: 0.296000 seconds
Total Used Memory: 24000 MB
Index 47, Time: 0.481000 seconds
Total Used Memory: 24500 MB
Index 48, Time: 0.291000 seconds
Total Used Memory: 25000 MB
```

```
Index 49, Time: 0.386000 seconds
Total Used Memory: 25500 MB
Index 50, Time: 0.264000 seconds
Total Used Memory: 26000 MB
Index 51, Time: 0.278000 seconds
Total Used Memory: 26500 MB
Index 52, Time: 0.299000 seconds
Total Used Memory: 27000 MB
Index 53, Time: 0.431000 seconds
Total Used Memory: 27500 MB
Index 54, Time: 0.270000 seconds
Total Used Memory: 28000 MB
Index 55, Time: 0.301000 seconds
Total Used Memory: 28500 MB
Index 56, Time: 0.286000 seconds
Total Used Memory: 29000 MB
Index 57, Time: 0.269000 seconds
Total Used Memory: 29500 MB
Index 58, Time: 0.262000 seconds
Total Used Memory: 30000 MB
Index 59, Time: 0.262000 seconds
Total Used Memory: 30500 MB
Index 60, Time: 0.284000 seconds
Total Used Memory: 31000 MB
Index 61, Time: 0.283000 seconds
Total Used Memory: 31500 MB
Index 62, Time: 0.240000 seconds
Total Used Memory: 32000 MB
Index 63, Time: 0.264000 seconds
Total Used Memory: 32500 MB
Index 64, Time: 0.495000 seconds
Total Used Memory: 33000 MB
Index 65, Time: 0.245000 seconds
Total Used Memory: 33500 MB
Index 66, Time: 0.254000 seconds
Total Used Memory: 34000 MB
Index 67, Time: 0.476000 seconds
Total Used Memory: 34500 MB
Index 68, Time: 0.254000 seconds
Total Used Memory: 35000 MB
Index 69, Time: 0.284000 seconds
Total Used Memory: 35500 MB
Index 70, Time: 0.263000 seconds
Total Used Memory: 36000 MB
Index 71, Time: 0.267000 seconds
Total Used Memory: 36500 MB
Index 72, Time: 0.267000 seconds
Total Used Memory: 37000 MB
Index 73, Time: 0.756000 seconds
Total Used Memory: 37500 MB
Index 74, Time: 0.266000 seconds
Total Used Memory: 38000 MB
Index 75, Time: 0.480000 seconds
Total Used Memory: 38500 MB
Index 76, Time: 0.257000 seconds
Total Used Memory: 39000 MB
Index 77, Time: 0.248000 seconds
Total Used Memory: 39500 MB
```

```
Index 78, Time: 0.255000 seconds
Total Used Memory: 40000 MB
Index 79, Time: 0.770000 seconds
Total Used Memory: 40500 MB
Index 80, Time: 0.257000 seconds
Total Used Memory: 41000 MB
Index 81, Time: 0.247000 seconds
Total Used Memory: 41500 MB
Index 82, Time: 0.672000 seconds
Total Used Memory: 42000 MB
Index 83, Time: 0.237000 seconds
Total Used Memory: 42500 MB
Index 84, Time: 0.251000 seconds
Total Used Memory: 43000 MB
Index 85, Time: 0.504000 seconds
Total Used Memory: 43500 MB
Index 86, Time: 0.681000 seconds
Total Used Memory: 44000 MB
Index 87, Time: 0.350000 seconds
Total Used Memory: 44500 MB
Index 88, Time: 0.244000 seconds
Total Used Memory: 45000 MB
Index 89, Time: 0.222000 seconds
Total Used Memory: 45500 MB
Index 90, Time: 0.260000 seconds
Total Used Memory: 46000 MB
Index 91, Time: 0.342000 seconds
Total Used Memory: 46500 MB
Index 92, Time: 0.277000 seconds
Total Used Memory: 47000 MB
Index 93, Time: 0.540000 seconds
Total Used Memory: 47500 MB
Index 94, Time: 0.905000 seconds
Total Used Memory: 48000 MB
Index 95, Time: 0.265000 seconds
Total Used Memory: 48500 MB
Index 96, Time: 0.262000 seconds
Total Used Memory: 49000 MB
Index 97, Time: 0.260000 seconds
Total Used Memory: 49500 MB
Index 98, Time: 0.250000 seconds
Total Used Memory: 50000 MB
Index 99, Time: 0.519000 seconds
Total Used Memory: 50500 MB
Index 100, Time: 0.250000 seconds
Total Used Memory: 51000 MB
Index 101, Time: 0.243000 seconds
Total Used Memory: 51500 MB
Index 102, Time: 0.238000 seconds
Total Used Memory: 52000 MB
Index 103, Time: 0.379000 seconds
Total Used Memory: 52500 MB
Index 104, Time: 0.261000 seconds
Total Used Memory: 53000 MB
Index 105, Time: 0.280000 seconds
Program Exited, Memory allocation failed at index 106
PS C:\Users\ishti\OneDrive\OS_II\Assignment2\Final Submission>
```

C. **Program Output in Linux Platform**

```
iahmed@poobah:~/os/Assignment2> ./memory
Total Used Memory: 500 MB
Index 0, Time: 0.298820 seconds
Total Used Memory: 1000 MB
Index 1, Time: 0.298883 seconds
Total Used Memory: 1500 MB
Index 2, Time: 0.298740 seconds
Total Used Memory: 2000 MB
Index 3, Time: 0.299594 seconds
Total Used Memory: 2500 MB
Index 4, Time: 0.298952 seconds
Total Used Memory: 3000 MB
Index 5, Time: 0.298484 seconds
Total Used Memory: 3500 MB
Index 6, Time: 0.298535 seconds
Total Used Memory: 4000 MB
Index 7, Time: 0.297956 seconds
Total Used Memory: 4500 MB
Index 8, Time: 0.298039 seconds
Total Used Memory: 5000 MB
Index 9, Time: 0.299376 seconds
Total Used Memory: 5500 MB
Index 10, Time: 0.298735 seconds
Total Used Memory: 6000 MB
Index 11, Time: 0.299491 seconds
Total Used Memory: 6500 MB
Index 12, Time: 0.299170 seconds
Total Used Memory: 7000 MB
Index 13, Time: 0.298490 seconds
Total Used Memory: 7500 MB
Index 14, Time: 0.298993 seconds
Total Used Memory: 8000 MB
Index 15, Time: 0.298457 seconds
Total Used Memory: 8500 MB
Index 16, Time: 0.299477 seconds
Total Used Memory: 9000 MB
Index 17, Time: 0.299356 seconds
Total Used Memory: 9500 MB
Index 18, Time: 0.298916 seconds
Total Used Memory: 10000 MB
Index 19, Time: 0.298703 seconds
Total Used Memory: 10500 MB
Index 20, Time: 0.298003 seconds
Total Used Memory: 11000 MB
Index 21, Time: 0.298855 seconds
Total Used Memory: 11500 MB
Index 22, Time: 0.298139 seconds
Total Used Memory: 12000 MB
Index 23, Time: 0.298038 seconds
Total Used Memory: 12500 MB
Index 24, Time: 0.298302 seconds
Total Used Memory: 13000 MB
Index 25, Time: 0.298731 seconds
Total Used Memory: 13500 MB
Index 26, Time: 0.298732 seconds
Total Used Memory: 14000 MB
```

```
Index 27, Time: 0.298729 seconds
Total Used Memory: 14500 MB
Index 28, Time: 0.299292 seconds
Total Used Memory: 15000 MB
Index 29, Time: 0.369831 seconds
Total Used Memory: 15500 MB
Index 30, Time: 0.433066 seconds
Total Used Memory: 16000 MB
Index 31, Time: 0.381503 seconds
Total Used Memory: 16500 MB
Index 32, Time: 0.480835 seconds
Total Used Memory: 17000 MB
Index 33, Time: 0.548557 seconds
Total Used Memory: 17500 MB
Index 34, Time: 0.551454 seconds
Total Used Memory: 18000 MB
Index 35, Time: 0.521973 seconds
Total Used Memory: 18500 MB
Index 36, Time: 0.532757 seconds
Total Used Memory: 19000 MB
Index 37, Time: 0.550760 seconds
Total Used Memory: 19500 MB
Killed
iahmed@poobah:~/os/Assignment2>
```

D. **Program Output in Mac Platform**

```
Total Used Memory: 500 MB
Index 0, Time: 0.171658 seconds
Total Used Memory: 1000 MB
Index 1, Time: 0.172784 seconds
Total Used Memory: 1500 MB
Index 2, Time: 0.174106 seconds
Total Used Memory: 2000 MB
Index 3, Time: 0.173579 seconds
Total Used Memory: 2500 MB
Index 4, Time: 0.174451 seconds
Total Used Memory: 3000 MB
Index 5, Time: 0.172561 seconds
Total Used Memory: 3500 MB
Index 6, Time: 0.172678 seconds
Total Used Memory: 4000 MB
Index 7, Time: 0.174778 seconds
Total Used Memory: 4500 MB
Index 8, Time: 0.173188 seconds
Total Used Memory: 5000 MB
Index 9, Time: 0.174362 seconds
Total Used Memory: 5500 MB
Index 10, Time: 0.174239 seconds
Total Used Memory: 6000 MB
Index 11, Time: 0.174202 seconds
Total Used Memory: 6500 MB
Index 12, Time: 0.171583 seconds
Total Used Memory: 7000 MB
Index 13, Time: 0.170393 seconds
Total Used Memory: 7500 MB
Index 14, Time: 0.174218 seconds
Total Used Memory: 8000 MB
```

```
Index 15, Time: 0.174000 seconds
Total Used Memory: 8500 MB
Index 16, Time: 0.171997 seconds
Total Used Memory: 9000 MB
Index 17, Time: 0.173253 seconds
Total Used Memory: 9500 MB
Index 18, Time: 0.176898 seconds
Total Used Memory: 10000 MB
Index 19, Time: 0.174413 seconds
Total Used Memory: 10500 MB
Index 20, Time: 0.171708 seconds
Total Used Memory: 11000 MB
Index 21, Time: 0.181336 seconds
Total Used Memory: 11500 MB
Index 22, Time: 0.172615 seconds
Total Used Memory: 12000 MB
Index 23, Time: 0.176956 seconds
Total Used Memory: 12500 MB
Index 24, Time: 0.176880 seconds
Total Used Memory: 13000 MB
Index 25, Time: 0.173075 seconds
Total Used Memory: 13500 MB
Index 26, Time: 0.176512 seconds
Total Used Memory: 14000 MB
Index 27, Time: 0.173090 seconds
Total Used Memory: 14500 MB
Index 28, Time: 0.173999 seconds
Total Used Memory: 15000 MB
Index 29, Time: 0.175087 seconds
Total Used Memory: 15500 MB
Index 30, Time: 0.173264 seconds
Total Used Memory: 16000 MB
Index 31, Time: 0.173360 seconds
Total Used Memory: 16500 MB
Index 32, Time: 0.176635 seconds
Total Used Memory: 17000 MB
Index 33, Time: 0.173634 seconds
Total Used Memory: 17500 MB
Index 34, Time: 0.175884 seconds
Total Used Memory: 18000 MB
Index 35, Time: 0.172585 seconds
Total Used Memory: 18500 MB
Index 36, Time: 0.177318 seconds
Total Used Memory: 19000 MB
Index 37, Time: 0.174446 seconds
Total Used Memory: 19500 MB
Index 38, Time: 0.173436 seconds
Total Used Memory: 20000 MB
Index 39, Time: 0.171188 seconds
Total Used Memory: 20500 MB
Index 40, Time: 0.182107 seconds
Total Used Memory: 21000 MB
Index 41, Time: 0.174331 seconds
Total Used Memory: 21500 MB
Index 42, Time: 0.174750 seconds
Total Used Memory: 22000 MB
Index 43, Time: 0.171967 seconds
Total Used Memory: 22500 MB
```

```
Index 44, Time: 0.177188 seconds
Total Used Memory: 23000 MB
Index 45, Time: 0.175867 seconds
Total Used Memory: 23500 MB
Index 46, Time: 0.178385 seconds
Total Used Memory: 24000 MB
Index 47, Time: 0.173834 seconds
Total Used Memory: 24500 MB
Index 48, Time: 0.174535 seconds
Total Used Memory: 25000 MB
Index 49, Time: 0.172547 seconds
Total Used Memory: 25500 MB
Index 50, Time: 0.175645 seconds
Total Used Memory: 26000 MB
Index 51, Time: 0.174218 seconds
Total Used Memory: 26500 MB
Index 52, Time: 0.172190 seconds
Total Used Memory: 27000 MB
Index 53, Time: 0.175055 seconds
Total Used Memory: 27500 MB
Index 54, Time: 0.174206 seconds
Total Used Memory: 28000 MB
Index 55, Time: 0.170772 seconds
Total Used Memory: 28500 MB
Index 56, Time: 0.174347 seconds
Total Used Memory: 29000 MB
Index 57, Time: 0.173183 seconds
Total Used Memory: 29500 MB
Index 58, Time: 0.173769 seconds
Total Used Memory: 30000 MB
Index 59, Time: 0.174022 seconds
Total Used Memory: 30500 MB
Index 60, Time: 0.173404 seconds
Total Used Memory: 31000 MB
Index 61, Time: 0.176316 seconds
Total Used Memory: 31500 MB
Index 62, Time: 0.173196 seconds
Total Used Memory: 32000 MB
Index 63, Time: 0.176119 seconds
Total Used Memory: 32500 MB
Index 64, Time: 0.174241 seconds
Total Used Memory: 33000 MB
Index 65, Time: 0.174380 seconds
Total Used Memory: 33500 MB
Index 66, Time: 0.173407 seconds
Total Used Memory: 34000 MB
Index 67, Time: 0.175420 seconds
Total Used Memory: 34500 MB
Index 68, Time: 0.177287 seconds
Total Used Memory: 35000 MB
Index 69, Time: 0.170812 seconds
Total Used Memory: 35500 MB
Index 70, Time: 0.177389 seconds
Total Used Memory: 36000 MB
Index 71, Time: 0.173736 seconds
Total Used Memory: 36500 MB
Index 72, Time: 0.172273 seconds
Total Used Memory: 37000 MB
```

```
Index 73, Time: 0.175564 seconds
Total Used Memory: 37500 MB
Index 74, Time: 0.175631 seconds
Total Used Memory: 38000 MB
Index 75, Time: 0.176721 seconds
Total Used Memory: 38500 MB
Index 76, Time: 0.172290 seconds
Total Used Memory: 39000 MB
Index 77, Time: 0.172589 seconds
Total Used Memory: 39500 MB
Index 78, Time: 0.178910 seconds
Total Used Memory: 40000 MB
Index 79, Time: 0.173815 seconds
Total Used Memory: 40500 MB
Index 80, Time: 0.181248 seconds
Total Used Memory: 41000 MB
Index 81, Time: 0.175128 seconds
Total Used Memory: 41500 MB
Index 82, Time: 0.174242 seconds
Total Used Memory: 42000 MB
Index 83, Time: 0.177970 seconds
Total Used Memory: 42500 MB
Index 84, Time: 0.176530 seconds
Total Used Memory: 43000 MB
Index 85, Time: 0.174265 seconds
Total Used Memory: 43500 MB
Index 86, Time: 0.173095 seconds
Total Used Memory: 44000 MB
Index 87, Time: 0.175220 seconds
Total Used Memory: 44500 MB
Index 88, Time: 0.176470 seconds
Total Used Memory: 45000 MB
Index 89, Time: 0.174457 seconds
Total Used Memory: 45500 MB
Index 90, Time: 0.176449 seconds
Total Used Memory: 46000 MB
Index 91, Time: 0.177722 seconds
Total Used Memory: 46500 MB
Index 92, Time: 0.176096 seconds
Total Used Memory: 47000 MB
Index 93, Time: 0.172445 seconds
Total Used Memory: 47500 MB
Index 94, Time: 0.175890 seconds
Total Used Memory: 48000 MB
Index 95, Time: 0.176872 seconds
Total Used Memory: 48500 MB
Index 96, Time: 0.174486 seconds
Total Used Memory: 49000 MB
Index 97, Time: 0.172851 seconds
Total Used Memory: 49500 MB
Index 98, Time: 0.176455 seconds
Total Used Memory: 50000 MB
Index 99, Time: 0.172899 seconds
Total Used Memory: 50500 MB
Index 100, Time: 0.172168 seconds
Total Used Memory: 51000 MB
Index 101, Time: 0.173479 seconds
Total Used Memory: 51500 MB
```

```
Index 102, Time: 0.177390 seconds
Total Used Memory: 52000 MB
Index 103, Time: 0.176702 seconds
Total Used Memory: 52500 MB
Index 104, Time: 0.177215 seconds
Total Used Memory: 53000 MB
Index 105, Time: 0.174141 seconds
Total Used Memory: 53500 MB
Index 106, Time: 0.174625 seconds
Total Used Memory: 54000 MB
Index 107, Time: 0.174601 seconds
Total Used Memory: 54500 MB
Index 108, Time: 0.172739 seconds
Total Used Memory: 55000 MB
Index 109, Time: 0.175319 seconds
Total Used Memory: 55500 MB
Index 110, Time: 0.173766 seconds
Total Used Memory: 56000 MB
Index 111, Time: 0.178514 seconds
Total Used Memory: 56500 MB
Index 112, Time: 0.175787 seconds
Total Used Memory: 57000 MB
Index 113, Time: 0.173594 seconds
Total Used Memory: 57500 MB
Index 114, Time: 0.177416 seconds
Total Used Memory: 58000 MB
Index 115, Time: 0.175024 seconds
Total Used Memory: 58500 MB
Index 116, Time: 0.176143 seconds
Total Used Memory: 59000 MB
Index 117, Time: 0.172488 seconds
Total Used Memory: 59500 MB
Index 118, Time: 0.172919 seconds
Total Used Memory: 60000 MB
Index 119, Time: 0.177025 seconds
Total Used Memory: 60500 MB
Index 120, Time: 0.176280 seconds
Total Used Memory: 61000 MB
Index 121, Time: 0.172747 seconds
Total Used Memory: 61500 MB
Index 122, Time: 0.173589 seconds
Total Used Memory: 62000 MB
Index 123, Time: 0.174168 seconds
Total Used Memory: 62500 MB
Index 124, Time: 0.171060 seconds
Total Used Memory: 63000 MB
Index 125, Time: 0.173855 seconds
Total Used Memory: 63500 MB
Index 126, Time: 0.174120 seconds
Total Used Memory: 64000 MB
Index 127, Time: 0.177171 seconds
Total Used Memory: 64500 MB
Index 128, Time: 0.176185 seconds
Total Used Memory: 65000 MB
Index 129, Time: 0.172757 seconds
Total Used Memory: 65500 MB
Index 130, Time: 0.185332 seconds
Total Used Memory: 66000 MB
```

```
Index 131, Time: 0.174728 seconds
Total Used Memory: 66500 MB
Index 132, Time: 0.174002 seconds
Total Used Memory: 67000 MB
Index 133, Time: 0.183042 seconds
Total Used Memory: 67500 MB
Index 134, Time: 0.172863 seconds
Total Used Memory: 68000 MB
Index 135, Time: 0.174153 seconds
Total Used Memory: 68500 MB
Index 136, Time: 0.176774 seconds
Total Used Memory: 69000 MB
Index 137, Time: 0.175489 seconds
Total Used Memory: 69500 MB
Index 138, Time: 0.173435 seconds
Total Used Memory: 70000 MB
Index 139, Time: 0.175408 seconds
Total Used Memory: 70500 MB
Index 140, Time: 0.175891 seconds
Total Used Memory: 71000 MB
Index 141, Time: 0.176429 seconds
Total Used Memory: 71500 MB
Index 142, Time: 0.172872 seconds
Total Used Memory: 72000 MB
Index 143, Time: 0.176914 seconds
Total Used Memory: 72500 MB
Index 144, Time: 0.175018 seconds
Total Used Memory: 73000 MB
Index 145, Time: 0.175068 seconds
Total Used Memory: 73500 MB
Index 146, Time: 0.175989 seconds
Total Used Memory: 74000 MB
Index 147, Time: 0.175278 seconds
Total Used Memory: 74500 MB
Index 148, Time: 0.173573 seconds
Total Used Memory: 75000 MB
Index 149, Time: 0.176249 seconds
Total Used Memory: 75500 MB
Index 150, Time: 0.175195 seconds
Total Used Memory: 76000 MB
Index 151, Time: 0.172313 seconds
Total Used Memory: 76500 MB
Index 152, Time: 0.176951 seconds
Total Used Memory: 77000 MB
Index 153, Time: 0.176663 seconds
Total Used Memory: 77500 MB
Index 154, Time: 0.176420 seconds
Total Used Memory: 78000 MB
Index 155, Time: 0.177399 seconds
Total Used Memory: 78500 MB
Index 156, Time: 0.169965 seconds
zsh: killed     ./main
```