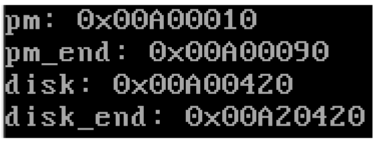
# CSE312 – Operating Systems

# Homework #2 Report

Ömer Faruk Bitikçioğlu

161044010

First, we make some space to hold physical memory and disk:



```
pm: 0x00A00010
pm_end: 0x00A00090
disk: 0x00A00420
disk_end: 0x00A20420
```

You can see the physical memory and disk addresses. They both hold in the heap.

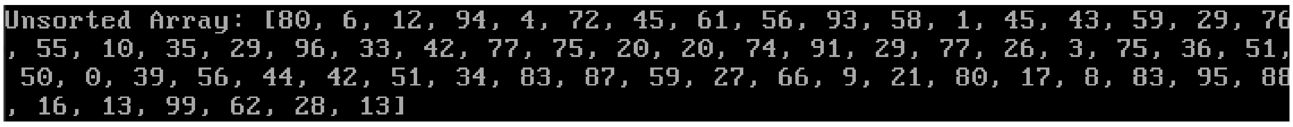
User will give a relative size to physical memory. It should be bigger than 1, so the array will not fit into the physical memory. Physical memory can hold 128 byte = 32 int value. If we choose 2 as relative size, then the array size will be 256 byte = 64 int values. On the other hand disk can hold 2^15 int values.

When we doing sorting in this array, CPU will think that it all fit into memory, because it sees the memory as a big virtual memory. Some part of the array will be hold into physical memory and some part will be hold in disk.

When the sorting program is running, at first physical memory will be empty. So every call to an array element will be a miss. After misses the operating system will load the array elements one by one to the physical memory.

Physical memory capacity will be insufficient at some point. All the pages will be full and need to be replaced. In this situation the page replacement algorithm the user choose will show up and replace the page that need to be replaced.

We have an unsorted integer array with 64 elements:



```
Unsorted Array: [80, 6, 12, 94, 4, 72, 45, 61, 56, 93, 58, 1, 45, 43, 59, 29, 76
, 55, 10, 35, 29, 96, 33, 42, 77, 75, 20, 20, 74, 91, 29, 77, 26, 3, 75, 36, 51,
 50, 0, 39, 56, 44, 42, 51, 34, 83, 87, 59, 27, 66, 9, 21, 80, 17, 8, 83, 95, 88
, 16, 13, 99, 62, 28, 13]
```

Our main goal here is to sort this array. We will have the option of using three basic sorting algorithms. Bubble sort, insertion sort, and quick sort. User will chose one of them and try to see how is page replacement efficiency changes accordingly. We also see which page replacement algorithm is good for which sorting algorithm.

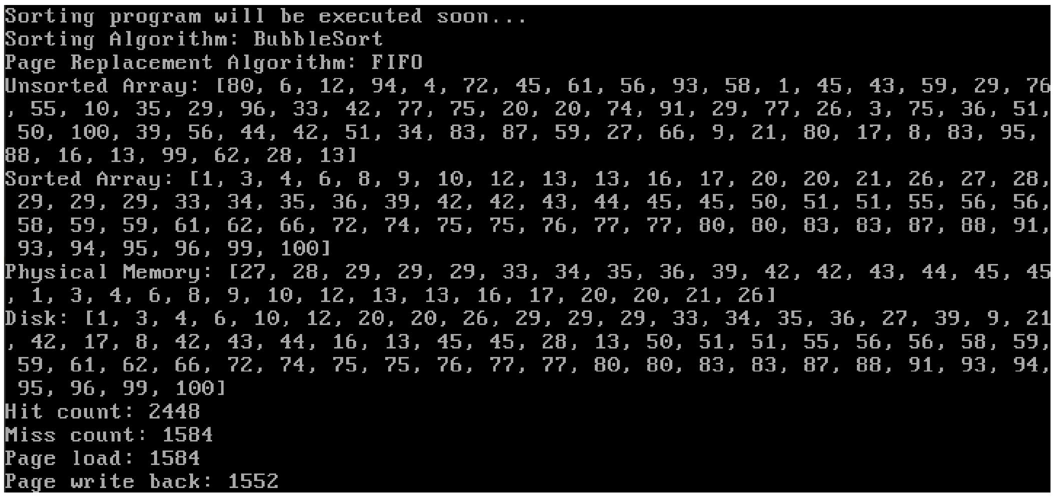
This array is placed to disk array. We will place elements from disk to physical memory when they needed. We will keep records of miss and hit rates. If the physical memory capacity is insufficient, then we will replace some of the pages of physical memory with the needed pages. We also keep the records of page replacement efficiency.

I implemented FIFO, Second Chance and LRU algorithms. For FIFO I implemented a basic queue model holding front and rear nodes of the physical memory addresses. If a memory page is need to be replaced, according to FIFO the front page should be replaced since it is the one that first enters to physical memory.

For second chance algorithm, we use refined version of FIFO. If a node is referanced we give it a second chance and make its referanced bit 0, put the page to the end of the queue. If the page is not referenced, then it is the page we should replace. If all the nodes are referanced then it is normal FIFO.

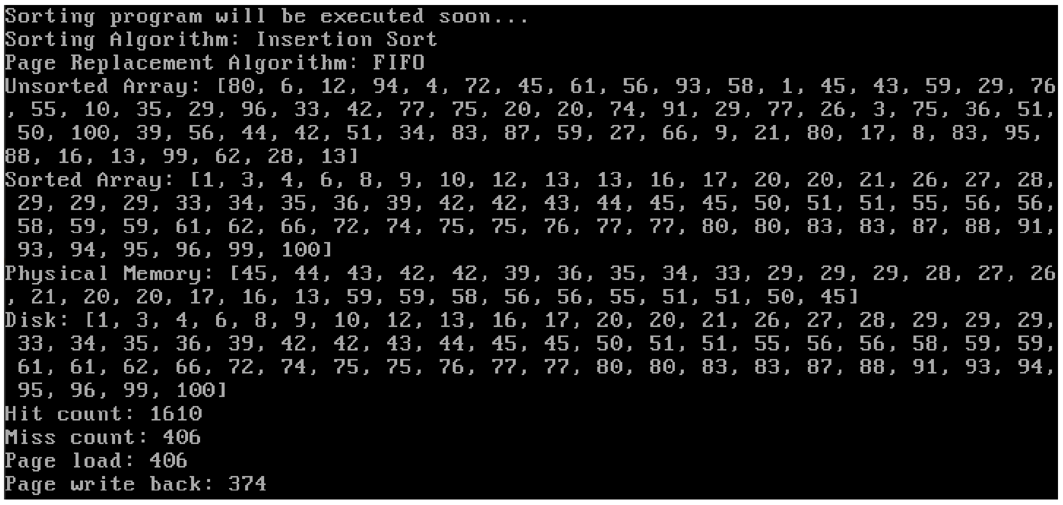
For LRU we keep an array of recently used pages. We fill the array like FIFO queue. If a page is referanced we put it back to the array. Least recently used page is the 0th element of the array.

When we try FIFO with Bubble Sort, we got a result like the below:



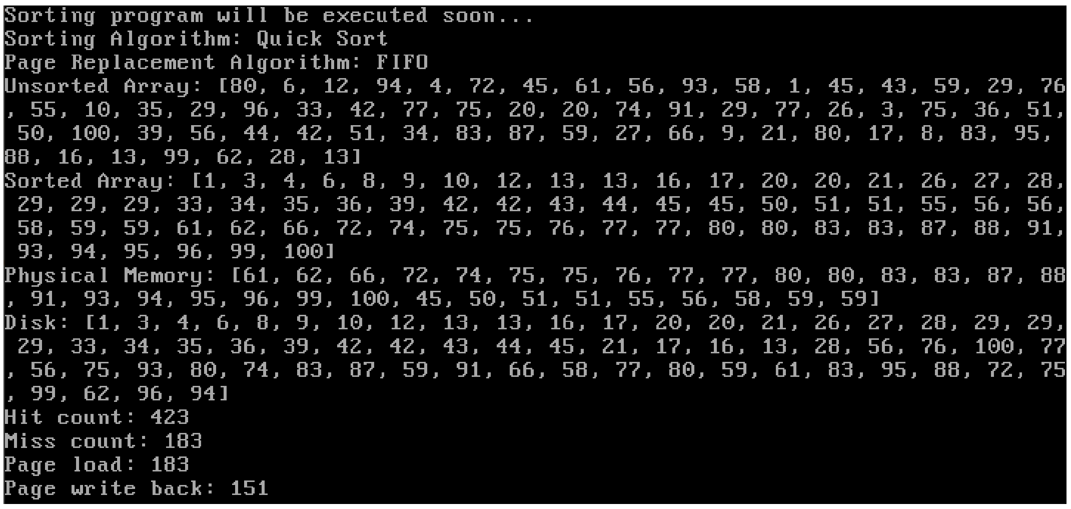
```
Sorting program will be executed soon...
Sorting Algorithm: BubbleSort
Page Replacement Algorithm: FIFO
Unsorted Array: [80, 6, 12, 94, 4, 72, 45, 61, 56, 93, 58, 1, 45, 43, 59, 29, 76
, 55, 10, 35, 29, 96, 33, 42, 77, 75, 20, 20, 74, 91, 29, 77, 26, 3, 75, 36, 51,
 50, 100, 39, 56, 44, 42, 51, 34, 83, 87, 59, 27, 66, 9, 21, 80, 17, 8, 83, 95,
88, 16, 13, 99, 62, 28, 13]
Sorted Array: [1, 3, 4, 6, 8, 9, 10, 12, 13, 13, 16, 17, 20, 20, 21, 26, 27, 28,
 29, 29, 29, 33, 34, 35, 36, 39, 42, 42, 43, 44, 45, 45, 50, 51, 51, 55, 56, 56,
 58, 59, 59, 61, 62, 66, 72, 74, 75, 75, 76, 77, 77, 80, 80, 83, 83, 87, 88, 91,
 93, 94, 95, 96, 99, 100]
Physical Memory: [27, 28, 29, 29, 29, 33, 34, 35, 36, 39, 42, 42, 43, 44, 45, 45
, 1, 3, 4, 6, 8, 9, 10, 12, 13, 13, 16, 17, 20, 20, 21, 26]
Disk: [1, 3, 4, 6, 10, 12, 20, 20, 26, 29, 29, 29, 33, 34, 35, 36, 27, 39, 9, 21
, 42, 17, 8, 42, 43, 44, 16, 13, 45, 45, 28, 13, 50, 51, 51, 55, 56, 56, 58, 59,
 59, 61, 62, 66, 72, 74, 75, 75, 76, 77, 77, 80, 80, 83, 83, 87, 88, 91, 93, 94,
 95, 96, 99, 100]
Hit count: 2448
Miss count: 1584
Page load: 1584
Page write back: 1552
```

When we try FIFO with Insertion Sort, we got a result like the below:



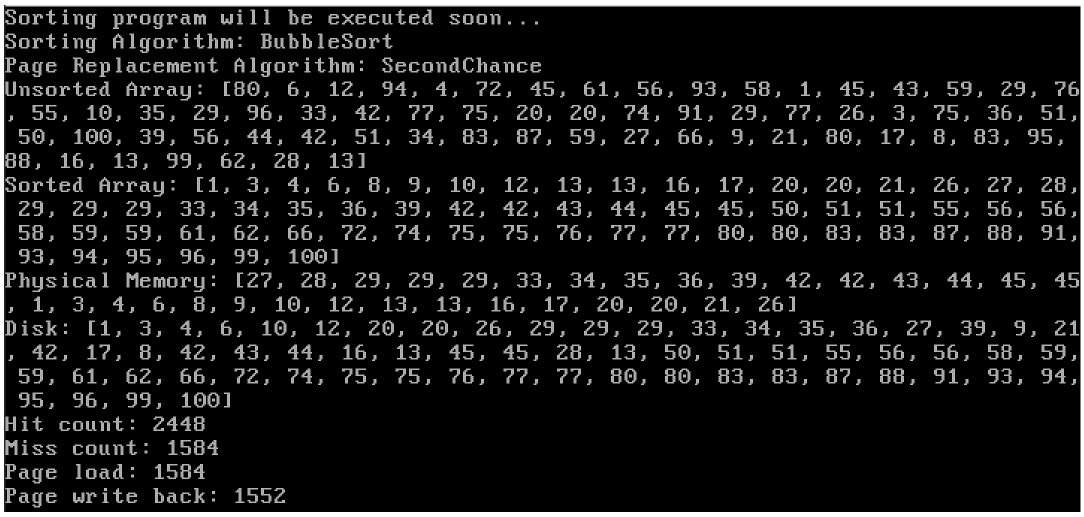
```
Sorting program will be executed soon...
Sorting Algorithm: Insertion Sort
Page Replacement Algorithm: FIFO
Unsorted Array: [80, 6, 12, 94, 4, 72, 45, 61, 56, 93, 58, 1, 45, 43, 59, 29, 76
, 55, 10, 35, 29, 96, 33, 42, 77, 75, 20, 20, 74, 91, 29, 77, 26, 3, 75, 36, 51,
 50, 100, 39, 56, 44, 42, 51, 34, 83, 87, 59, 27, 66, 9, 21, 80, 17, 8, 83, 95,
88, 16, 13, 99, 62, 28, 13]
Sorted Array: [1, 3, 4, 6, 8, 9, 10, 12, 13, 13, 16, 17, 20, 20, 21, 26, 27, 28,
 29, 29, 29, 33, 34, 35, 36, 39, 42, 42, 43, 44, 45, 45, 50, 51, 51, 55, 56, 56,
 58, 59, 59, 61, 62, 66, 72, 74, 75, 75, 76, 77, 77, 80, 80, 83, 83, 87, 88, 91,
 93, 94, 95, 96, 99, 100]
Physical Memory: [45, 44, 43, 42, 42, 39, 36, 35, 34, 33, 29, 29, 29, 28, 27, 26
, 21, 20, 20, 17, 16, 13, 59, 59, 58, 56, 56, 55, 51, 51, 50, 45]
Disk: [1, 3, 4, 6, 8, 9, 10, 12, 13, 16, 17, 20, 20, 21, 26, 27, 28, 29, 29, 29,
 33, 34, 35, 36, 39, 42, 42, 43, 44, 45, 45, 50, 51, 51, 55, 56, 56, 58, 59, 59,
 61, 61, 62, 66, 72, 74, 75, 75, 76, 77, 77, 80, 80, 83, 83, 87, 88, 91, 93, 94,
 95, 96, 99, 100]
Hit count: 1610
Miss count: 406
Page load: 406
Page write back: 374
```

When we try FIFO with Quick Sort, we got a result like the below:



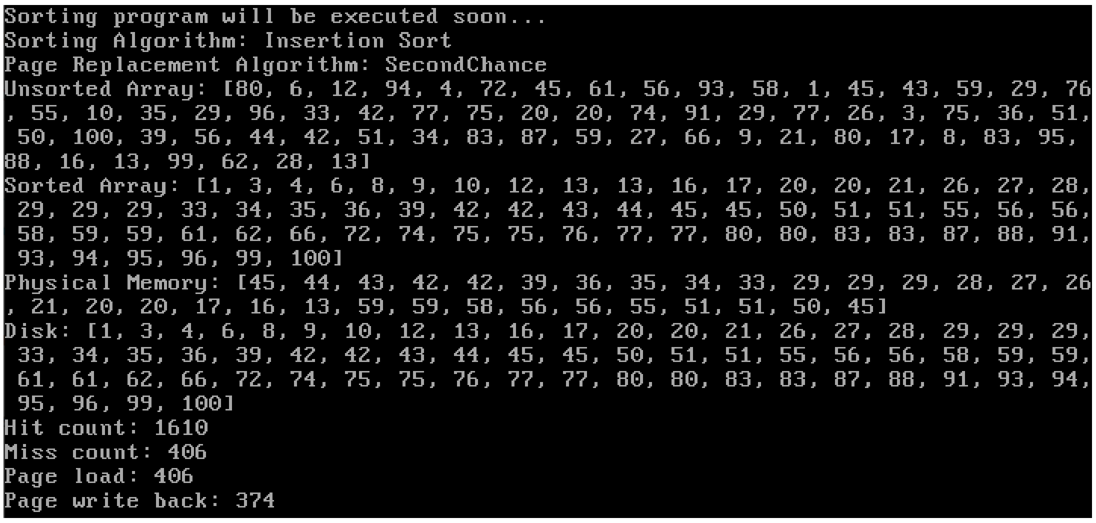
```
Sorting program will be executed soon...
Sorting Algorithm: Quick Sort
Page Replacement Algorithm: FIFO
Unsorted Array: [80, 6, 12, 94, 4, 72, 45, 61, 56, 93, 58, 1, 45, 43, 59, 29, 76
, 55, 10, 35, 29, 96, 33, 42, 77, 75, 20, 20, 74, 91, 29, 77, 26, 3, 75, 36, 51,
 50, 100, 39, 56, 44, 42, 51, 34, 83, 87, 59, 27, 66, 9, 21, 80, 17, 8, 83, 95,
88, 16, 13, 99, 62, 28, 13]
Sorted Array: [1, 3, 4, 6, 8, 9, 10, 12, 13, 13, 16, 17, 20, 20, 21, 26, 27, 28,
 29, 29, 29, 33, 34, 35, 36, 39, 42, 42, 43, 44, 45, 45, 50, 51, 51, 55, 56, 56,
 58, 59, 59, 61, 62, 66, 72, 74, 75, 75, 76, 77, 77, 80, 80, 83, 83, 87, 88, 91,
 93, 94, 95, 96, 99, 100]
Physical Memory: [61, 62, 66, 72, 74, 75, 75, 76, 77, 77, 80, 80, 83, 83, 87, 88
, 91, 93, 94, 95, 96, 99, 100, 45, 50, 51, 51, 55, 56, 58, 59, 59]
Disk: [1, 3, 4, 6, 8, 9, 10, 12, 13, 13, 16, 17, 20, 20, 21, 26, 27, 28, 29, 29,
 29, 33, 34, 35, 36, 39, 42, 42, 43, 44, 45, 21, 17, 16, 13, 28, 56, 76, 100, 77
, 56, 75, 93, 80, 74, 83, 87, 59, 91, 66, 58, 77, 80, 59, 61, 83, 95, 88, 72, 75
, 99, 62, 96, 94]
Hit count: 423
Miss count: 183
Page load: 183
Page write back: 151
```

When we try Second Chance with Bubble Sort, we got a result like the below:



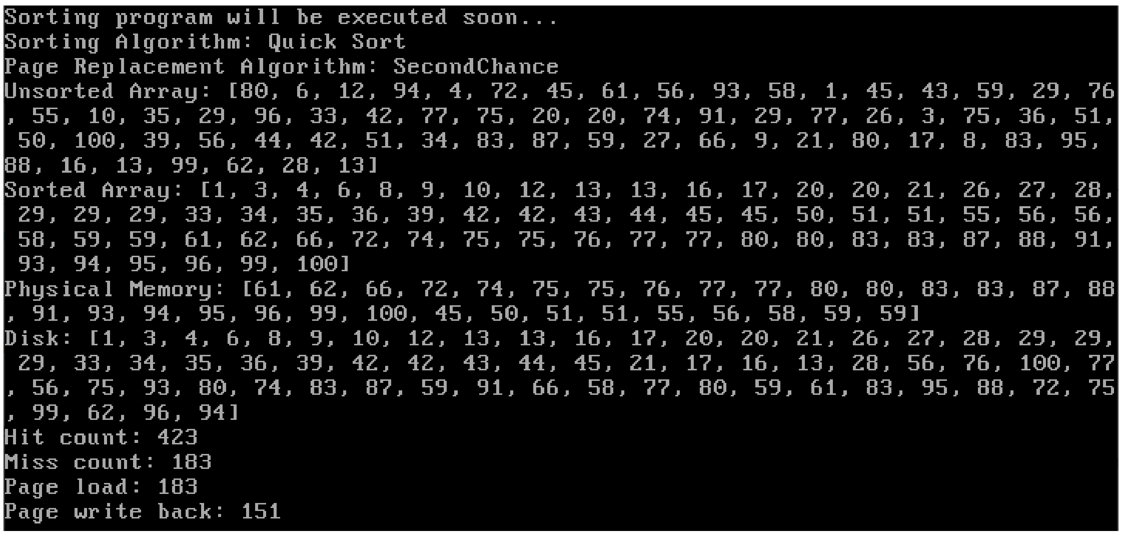
```
Sorting program will be executed soon...
Sorting Algorithm: BubbleSort
Page Replacement Algorithm: SecondChance
Unsorted Array: [80, 6, 12, 94, 4, 72, 45, 61, 56, 93, 58, 1, 45, 43, 59, 29, 76
, 55, 10, 35, 29, 96, 33, 42, 77, 75, 20, 20, 74, 91, 29, 77, 26, 3, 75, 36, 51,
 50, 100, 39, 56, 44, 42, 51, 34, 83, 87, 59, 27, 66, 9, 21, 80, 17, 8, 83, 95,
88, 16, 13, 99, 62, 28, 13]
Sorted Array: [1, 3, 4, 6, 8, 9, 10, 12, 13, 13, 16, 17, 20, 20, 21, 26, 27, 28,
 29, 29, 29, 33, 34, 35, 36, 39, 42, 42, 43, 44, 45, 45, 50, 51, 51, 55, 56, 56,
 58, 59, 59, 61, 62, 66, 72, 74, 75, 75, 76, 77, 77, 80, 80, 83, 83, 87, 88, 91,
 93, 94, 95, 96, 99, 100]
Physical Memory: [27, 28, 29, 29, 29, 33, 34, 35, 36, 39, 42, 42, 43, 44, 45, 45
, 1, 3, 4, 6, 8, 9, 10, 12, 13, 13, 16, 17, 20, 20, 21, 26]
Disk: [1, 3, 4, 6, 10, 12, 20, 20, 26, 29, 29, 29, 33, 34, 35, 36, 27, 39, 9, 21
, 42, 17, 8, 42, 43, 44, 16, 13, 45, 45, 28, 13, 50, 51, 51, 55, 56, 56, 58, 59,
 59, 61, 62, 66, 72, 74, 75, 75, 76, 77, 77, 80, 80, 83, 83, 87, 88, 91, 93, 94,
 95, 96, 99, 100]
Hit count: 2448
Miss count: 1584
Page load: 1584
Page write back: 1552
```

When we try Second Chance with Insertion Sort, we got a result like the below:



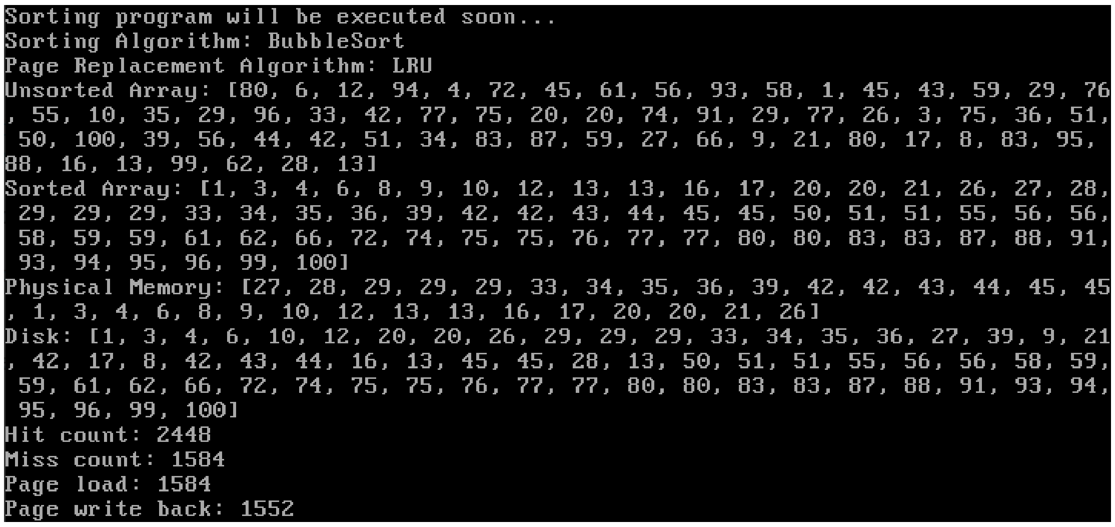
```
Sorting program will be executed soon...
Sorting Algorithm: Insertion Sort
Page Replacement Algorithm: SecondChance
Unsorted Array: [80, 6, 12, 94, 4, 72, 45, 61, 56, 93, 58, 1, 45, 43, 59, 29, 76
, 55, 10, 35, 29, 96, 33, 42, 77, 75, 20, 20, 74, 91, 29, 77, 26, 3, 75, 36, 51,
 50, 100, 39, 56, 44, 42, 51, 34, 83, 87, 59, 27, 66, 9, 21, 80, 17, 8, 83, 95,
88, 16, 13, 99, 62, 28, 13]
Sorted Array: [1, 3, 4, 6, 8, 9, 10, 12, 13, 13, 16, 17, 20, 20, 21, 26, 27, 28,
 29, 29, 29, 33, 34, 35, 36, 39, 42, 42, 43, 44, 45, 45, 50, 51, 51, 55, 56, 56,
 58, 59, 59, 61, 62, 66, 72, 74, 75, 75, 76, 77, 77, 80, 80, 83, 83, 87, 88, 91,
 93, 94, 95, 96, 99, 100]
Physical Memory: [45, 44, 43, 42, 42, 39, 36, 35, 34, 33, 29, 29, 29, 28, 27, 26
, 21, 20, 20, 17, 16, 13, 59, 59, 58, 56, 56, 55, 51, 51, 50, 45]
Disk: [1, 3, 4, 6, 8, 9, 10, 12, 13, 16, 17, 20, 20, 21, 26, 27, 28, 29, 29, 29,
 33, 34, 35, 36, 39, 42, 42, 43, 44, 45, 45, 50, 51, 51, 55, 56, 56, 58, 59, 59,
 61, 61, 62, 66, 72, 74, 75, 75, 76, 77, 77, 80, 80, 83, 83, 87, 88, 91, 93, 94,
 95, 96, 99, 100]
Hit count: 1610
Miss count: 406
Page load: 406
Page write back: 374
```

When we try Second Chance with Quick Sort, we got a result like the below:



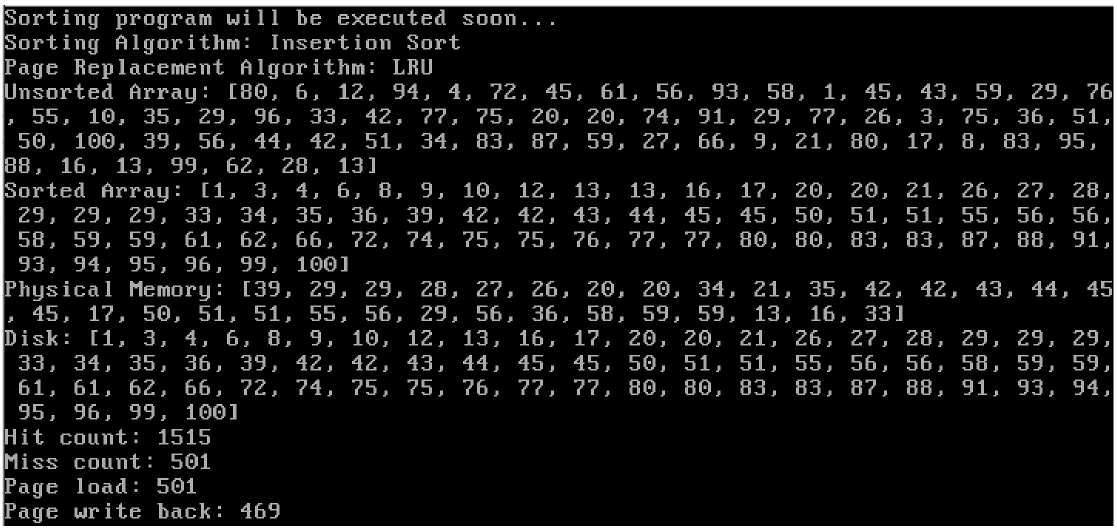
```
Sorting program will be executed soon...
Sorting Algorithm: Quick Sort
Page Replacement Algorithm: SecondChance
Unsorted Array: [80, 6, 12, 94, 4, 72, 45, 61, 56, 93, 58, 1, 45, 43, 59, 29, 76
, 55, 10, 35, 29, 96, 33, 42, 77, 75, 20, 20, 74, 91, 29, 77, 26, 3, 75, 36, 51,
 50, 100, 39, 56, 44, 42, 51, 34, 83, 87, 59, 27, 66, 9, 21, 80, 17, 8, 83, 95,
88, 16, 13, 99, 62, 28, 13]
Sorted Array: [1, 3, 4, 6, 8, 9, 10, 12, 13, 13, 16, 17, 20, 20, 21, 26, 27, 28,
 29, 29, 29, 33, 34, 35, 36, 39, 42, 42, 43, 44, 45, 45, 50, 51, 51, 55, 56, 56,
 58, 59, 59, 61, 62, 66, 72, 74, 75, 75, 76, 77, 77, 80, 80, 83, 83, 87, 88, 91,
 93, 94, 95, 96, 99, 100]
Physical Memory: [61, 62, 66, 72, 74, 75, 75, 76, 77, 77, 80, 80, 83, 83, 87, 88
, 91, 93, 94, 95, 96, 99, 100, 45, 50, 51, 51, 55, 56, 58, 59, 59]
Disk: [1, 3, 4, 6, 8, 9, 10, 12, 13, 13, 16, 17, 20, 20, 21, 26, 27, 28, 29, 29,
 29, 33, 34, 35, 36, 39, 42, 42, 43, 44, 45, 21, 17, 16, 13, 28, 56, 76, 100, 77
, 56, 75, 93, 80, 74, 83, 87, 59, 91, 66, 58, 77, 80, 59, 61, 83, 95, 88, 72, 75
, 99, 62, 96, 94]
Hit count: 423
Miss count: 183
Page load: 183
Page write back: 151
```

When we try LRU with Bubble Sort, we got a result like the below:



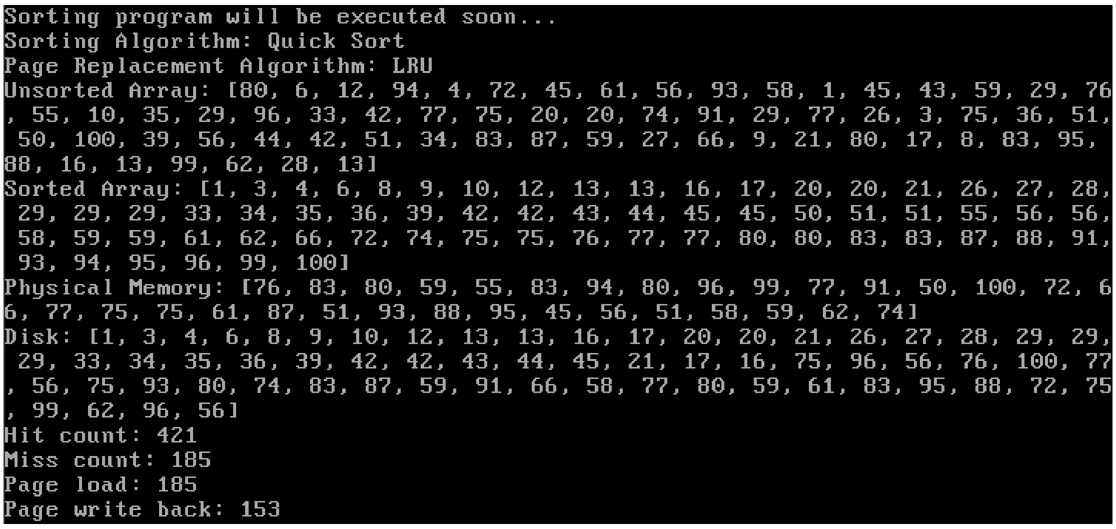
```
Sorting program will be executed soon...
Sorting Algorithm: BubbleSort
Page Replacement Algorithm: LRU
Unsorted Array: [80, 6, 12, 94, 4, 72, 45, 61, 56, 93, 58, 1, 45, 43, 59, 29, 76
, 55, 10, 35, 29, 96, 33, 42, 77, 75, 20, 20, 74, 91, 29, 77, 26, 3, 75, 36, 51,
 50, 100, 39, 56, 44, 42, 51, 34, 83, 87, 59, 27, 66, 9, 21, 80, 17, 8, 83, 95,
88, 16, 13, 99, 62, 28, 13]
Sorted Array: [1, 3, 4, 6, 8, 9, 10, 12, 13, 13, 16, 17, 20, 20, 21, 26, 27, 28,
 29, 29, 29, 33, 34, 35, 36, 39, 42, 42, 43, 44, 45, 45, 50, 51, 51, 55, 56, 56,
 58, 59, 59, 61, 62, 66, 72, 74, 75, 75, 76, 77, 77, 80, 80, 83, 83, 87, 88, 91,
 93, 94, 95, 96, 99, 100]
Physical Memory: [27, 28, 29, 29, 29, 33, 34, 35, 36, 39, 42, 42, 43, 44, 45, 45
, 1, 3, 4, 6, 8, 9, 10, 12, 13, 13, 16, 17, 20, 20, 21, 26]
Disk: [1, 3, 4, 6, 10, 12, 20, 20, 26, 29, 29, 29, 33, 34, 35, 36, 27, 39, 9, 21
, 42, 17, 8, 42, 43, 44, 16, 13, 45, 45, 28, 13, 50, 51, 51, 55, 56, 56, 58, 59,
 59, 61, 62, 66, 72, 74, 75, 75, 76, 77, 77, 80, 80, 83, 83, 87, 88, 91, 93, 94,
 95, 96, 99, 100]
Hit count: 2448
Miss count: 1584
Page load: 1584
Page write back: 1552
```

When we try LRU with Insertion Sort, we got a result like the below:



```
Sorting program will be executed soon...
Sorting Algorithm: Insertion Sort
Page Replacement Algorithm: LRU
Unsorted Array: [80, 6, 12, 94, 4, 72, 45, 61, 56, 93, 58, 1, 45, 43, 59, 29, 76
, 55, 10, 35, 29, 96, 33, 42, 77, 75, 20, 20, 74, 91, 29, 77, 26, 3, 75, 36, 51,
 50, 100, 39, 56, 44, 42, 51, 34, 83, 87, 59, 27, 66, 9, 21, 80, 17, 8, 83, 95,
88, 16, 13, 99, 62, 28, 13]
Sorted Array: [1, 3, 4, 6, 8, 9, 10, 12, 13, 13, 16, 17, 20, 20, 21, 26, 27, 28,
 29, 29, 29, 33, 34, 35, 36, 39, 42, 42, 43, 44, 45, 45, 50, 51, 51, 55, 56, 56,
 58, 59, 59, 61, 62, 66, 72, 74, 75, 75, 76, 77, 77, 80, 80, 83, 83, 87, 88, 91,
 93, 94, 95, 96, 99, 100]
Physical Memory: [39, 29, 29, 28, 27, 26, 20, 20, 34, 21, 35, 42, 42, 43, 44, 45
, 45, 17, 50, 51, 51, 55, 56, 29, 56, 36, 58, 59, 59, 13, 16, 33]
Disk: [1, 3, 4, 6, 8, 9, 10, 12, 13, 16, 17, 20, 20, 21, 26, 27, 28, 29, 29, 29,
 33, 34, 35, 36, 39, 42, 42, 43, 44, 45, 45, 50, 51, 51, 55, 56, 56, 58, 59, 59,
 61, 61, 62, 66, 72, 74, 75, 75, 76, 77, 77, 80, 80, 83, 83, 87, 88, 91, 93, 94,
 95, 96, 99, 100]
Hit count: 1515
Miss count: 501
Page load: 501
Page write back: 469
```

When we try LRU with Quick Sort, we got a result like the below:



```
Sorting program will be executed soon...
Sorting Algorithm: Quick Sort
Page Replacement Algorithm: LRU
Unsorted Array: [80, 6, 12, 94, 4, 72, 45, 61, 56, 93, 58, 1, 45, 43, 59, 29, 76
, 55, 10, 35, 29, 96, 33, 42, 77, 75, 20, 20, 74, 91, 29, 77, 26, 3, 75, 36, 51,
 50, 100, 39, 56, 44, 42, 51, 34, 83, 87, 59, 27, 66, 9, 21, 80, 17, 8, 83, 95,
88, 16, 13, 99, 62, 28, 13]
Sorted Array: [1, 3, 4, 6, 8, 9, 10, 12, 13, 13, 16, 17, 20, 20, 21, 26, 27, 28,
 29, 29, 29, 33, 34, 35, 36, 39, 42, 42, 43, 44, 45, 45, 50, 51, 51, 55, 56, 56,
 58, 59, 59, 61, 62, 66, 72, 74, 75, 75, 76, 77, 77, 80, 80, 83, 83, 87, 88, 91,
 93, 94, 95, 96, 99, 100]
Physical Memory: [76, 83, 80, 59, 55, 83, 94, 80, 96, 99, 77, 91, 50, 100, 72, 6
6, 77, 75, 75, 61, 87, 51, 93, 88, 95, 45, 56, 51, 58, 59, 62, 74]
Disk: [1, 3, 4, 6, 8, 9, 10, 12, 13, 13, 16, 17, 20, 20, 21, 26, 27, 28, 29, 29,
 29, 33, 34, 35, 36, 39, 42, 42, 43, 44, 45, 21, 17, 16, 75, 96, 56, 76, 100, 77
, 56, 75, 93, 80, 74, 83, 87, 59, 91, 66, 58, 77, 80, 59, 61, 83, 95, 88, 72, 75
, 99, 62, 96, 56]
Hit count: 421
Miss count: 185
Page load: 185
Page write back: 153
```