# GTU Department of Computer Engineering

# CSE312-Spring 2022

# Homework 2 Report

**AHMET OKUR**

**1801042655**

# Problem Defination

Our task is to design page replacement algorithms for the operating system. Our operating system should give the opportunity to replace the given page list (with their page number) into the given number of frames that can be in memory at a time per process. Your operating system should have the following page replacement algorithms so the user can select one of them:

*FIFO (first-In-First-Out)

*Second Change

*LRU (Least Recently Used)

# Plan & Design

- I made my design inspired by the operating system written by Viktor Engelmann.
- First of all, I prepared my algorithm for getting input from the user. The working logic of this algorithm is as follows:

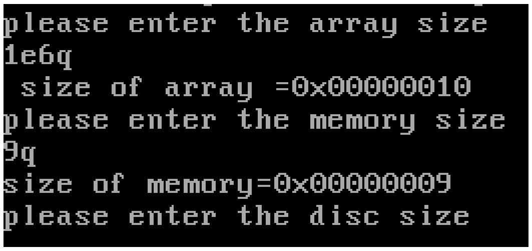
  For example, you want to enter the number 16. You must enter it as 1e6q. You must press 'e' after each digit. If our number is in the shape we want, you must press 'q'. (figure 1).
- We have 5 arrays. 1 of them is the array where we will use the sorting algorithm. The second one represents memory. the 3rd one represents the disk other arrays are LRU count and R bit. (figure 2)
- size of all arrays is taken from user (dynamic array)
- my program progresses with the user's choices.(figures 3 , 3.1)

# TIME Problem Solving

- I used time interrupt for time. First, I opened 2 empty processes. I put a counter in the first process. The other process also prints this counter. I ran the value in the second process 20 times and took the average. Now this is my concept of time. I named this time microphone.

- After this process, I have 2 programs, the first is the main program, and the second is the program I use to measure time.
- At the end of each operation in the main program, I take the value of the counter in the second program and divide it by the value I got averaged.
- this way my mic time shows up

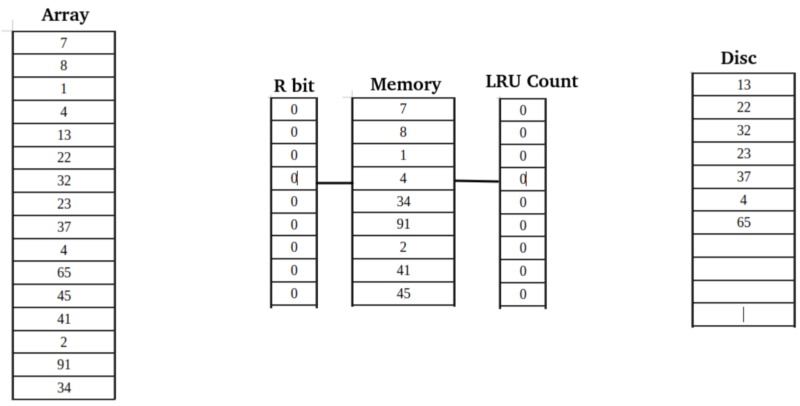
## Figures



```
please enter the array size
1e6q
 size of array =0x00000010
please enter the memory size
9q
size of memory=0x00000009
please enter the disc size
```

**Figure 1**

FIRST EXECUTION

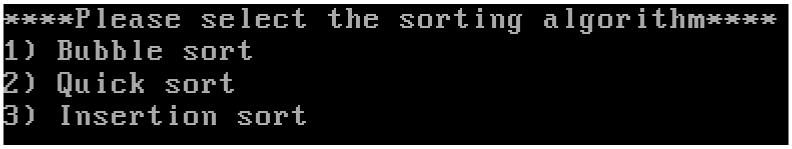


| Array |
|---|
| 7 |
| 8 |
| 1 |
| 4 |
| 13 |
| 22 |
| 32 |
| 23 |
| 37 |
| 4 |
| 65 |
| 45 |
| 41 |
| 2 |
| 91 |
| 34 |

| R bit | Memory | LRU Count |
|---|---|---|
| 0 | 7 | 0 |
| 0 | 8 | 0 |
| 0 | 1 | 0 |
| 0 | 4 | 0 |
| 0 | 34 | 0 |
| 0 | 91 | 0 |
| 0 | 2 | 0 |
| 0 | 41 | 0 |
| 0 | 45 | 0 |

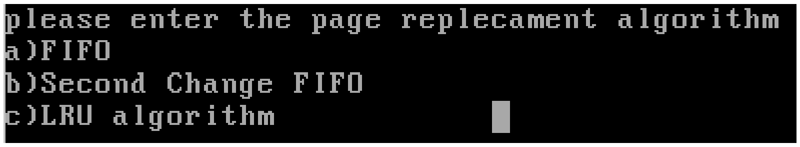
| Disc |
|---|
| 13 |
| 22 |
| 32 |
| 23 |
| 37 |
| 4 |
| 65 |
| |
| |
| |
| |

**Figure 2**



```
****Please select the sorting algorithm****
1) Bubble sort
2) Quick sort
3) Insertion sort
```

**Figure 3**



```
please enter the page replecament algorithm
a)FIFO
b)Second Change FIFO
c)LRU algorithm
```

**Figure 3.1**

# PAGE REPLACAMENT ALGORITHMS

## 1)FIFO

This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

I made it using circular array. I kept an iterator and incremented it in the page replecament and got its mod.

## 2)SECOND CHANGE FIFO

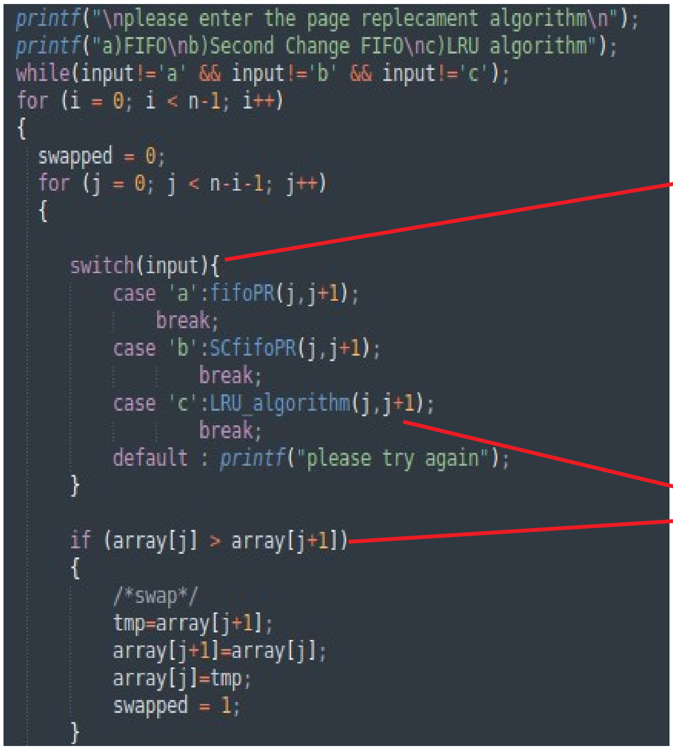
The second chance fifo algorithm has the same logic as the fifo algorithm, but here we need to keep R bits for each page. When a page is wanted to be changed, the R bit is checked. If the R bit is 1, the page is thrown to the end of the queue.

## 3)SECOND CHANGE FIFO

In **L**east **R**ecently **U**sed (LRU) algorithm is a Greedy algorithm where the page to be replaced is least recently used. The idea is based on locality of reference, the least recently used page is not likely

# Test Cases and Codes

## 1) BUBBLE SORT



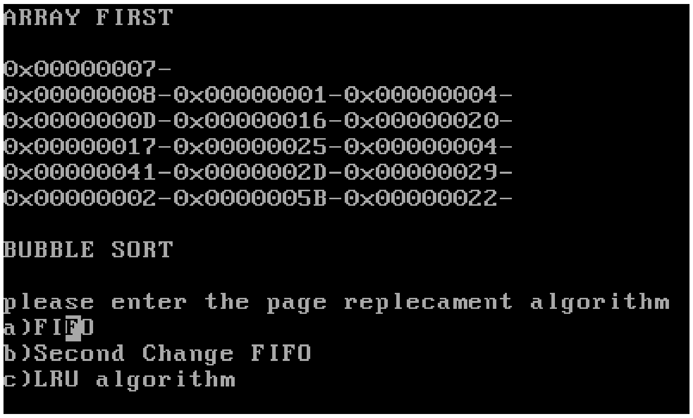
```
printf("\nplease enter the page replecament algorithm\n");
printf("a)FIFO\nb)Second Change FIFO\nc)LRU algorithm");
while(input!='a' && input!='b' && input!='c');
for (i = 0; i < n-1; i++)
{
  swapped = 0;
  for (j = 0; j < n-i-1; j++)
  {

     switch(input){
        case 'a':fifoPR(j,j+1);
            break;
        case 'b':SCfifoPR(j,j+1);
                break;
        case 'c':LRU_algorithm(j,j+1);
                break;
        default : printf("please try again");
     }

     if (array[j] > array[j+1])
     {
        /*swap*/
        tmp=array[j+1];
        array[j+1]=array[j];
        array[j]=tmp;
        swapped = 1;
     }
```

**This is some part of bubble sort algorithm**

In this section, we choose which page replecament algorithm we want the bubble sort to work with.

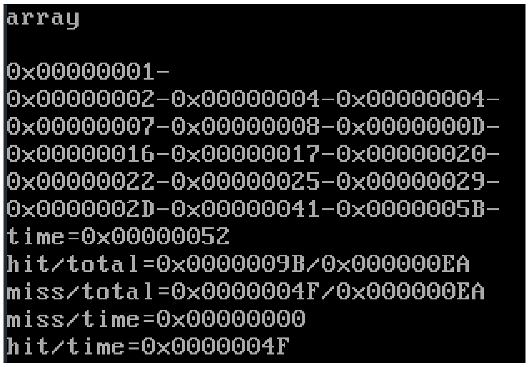
checking array[j] and array[j+1] in page replacement algorithm.

The following screen is the result of the code



```
ARRAY FIRST

0x00000007-
0x00000008-0x00000001-0x00000004-
0x0000000D-0x00000016-0x00000020-
0x00000017-0x00000025-0x00000004-
0x00000041-0x0000002D-0x00000029-
0x00000002-0x0000005B-0x00000022-

BUBBLE SORT

please enter the page replecament algorithm
a)FIFO
b)Second Change FIFO
c)LRU algorithm
```

## 1.1) Bubble sort and FIFO RESULT

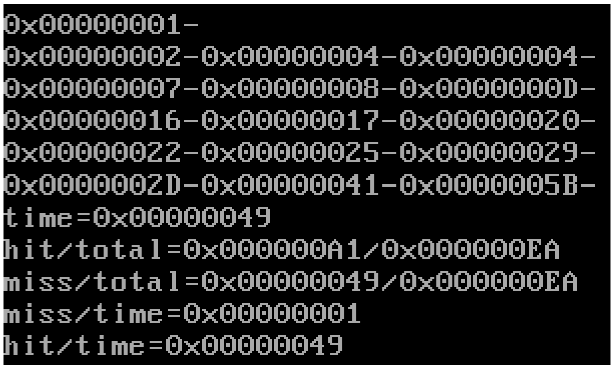


```
array

0x00000001-
0x00000002-0x00000004-0x00000004-
0x00000007-0x00000008-0x0000000D-
0x00000016-0x00000017-0x00000020-
0x00000022-0x00000025-0x00000029-
0x0000002D-0x00000041-0x0000005B-
time=0x00000052
hit/total=0x0000009B/0x000000EA
miss/total=0x0000004F/0x000000EA
miss/time=0x00000000
hit/time=0x0000004F
```

total=miss+hit

time=0x00000052 microfon

## 1.2) Bubble Sort and Second Change FIFO RESULT

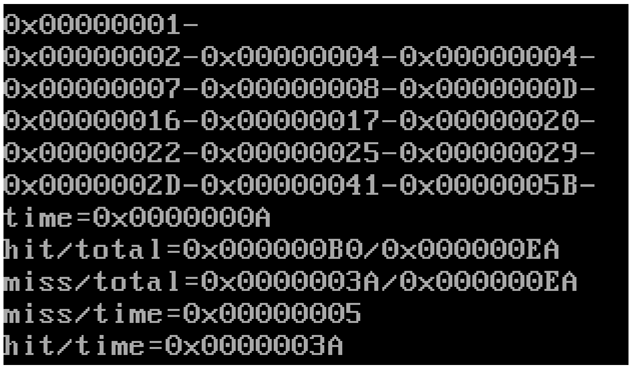


```
0x00000001-
0x00000002-0x00000004-0x00000004-
0x00000007-0x00000008-0x0000000D-
0x00000016-0x00000017-0x00000020-
0x00000022-0x00000025-0x00000029-
0x0000002D-0x00000041-0x0000005B-
time=0x00000049
hit/total=0x000000A1/0x000000EA
miss/total=0x00000049/0x000000EA
miss/time=0x00000001
hit/time=0x00000049
```

total=miss+hit

time=0x00000049 microfon

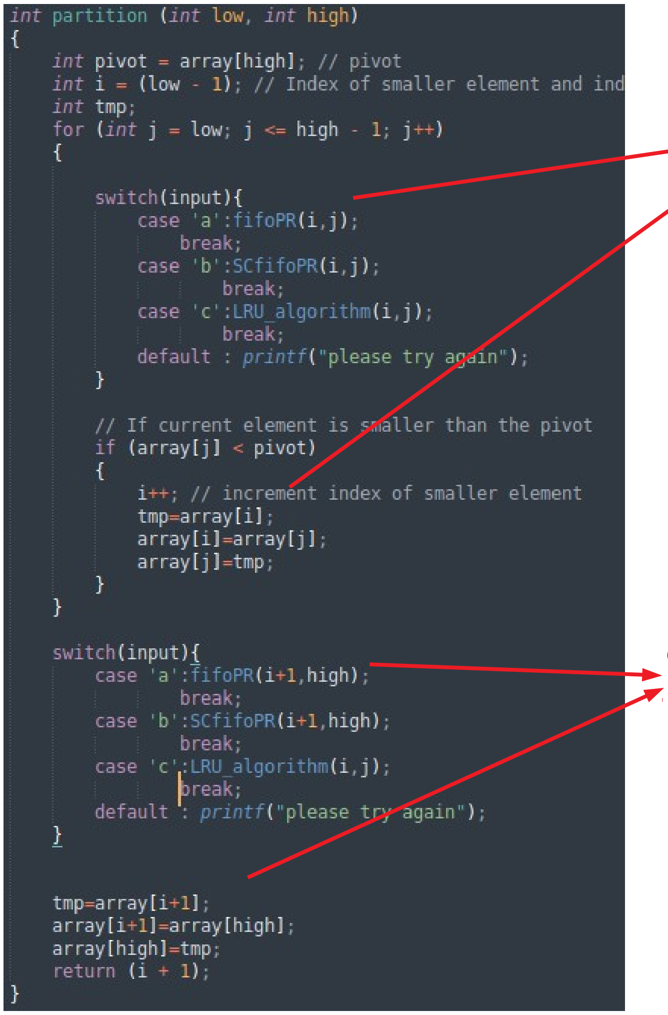
## 1.3) Bubble Sort and LRU RESULT



```
0x00000001-
0x00000002-0x00000004-0x00000004-
0x00000007-0x00000008-0x0000000D-
0x00000016-0x00000017-0x00000020-
0x00000022-0x00000025-0x00000029-
0x0000002D-0x00000041-0x0000005B-
time=0x0000000A
hit/total=0x000000B0/0x000000EA
miss/total=0x0000003A/0x000000EA
miss/time=0x00000005
hit/time=0x0000003A
```

total=miss+hit
time=0x0000000A microfon

## 2) QUICK SORT

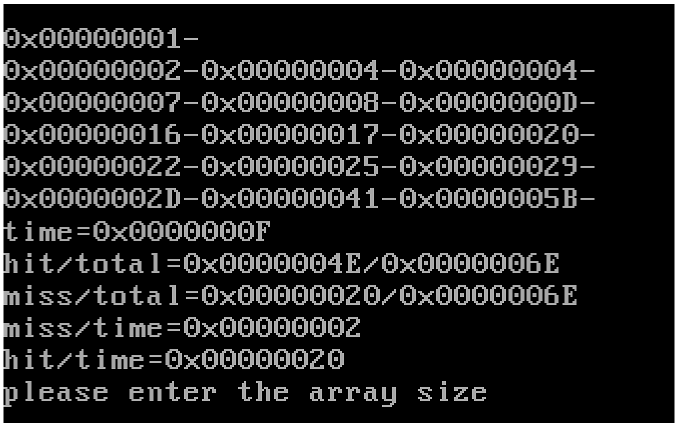


```
int partition (int low, int high)
{
    int pivot = array[high]; // pivot
    int i = (low - 1); // Index of smaller element and ind
    int tmp;
    for (int j = low; j <= high - 1; j++)
    {

        switch(input){
            case 'a':fifoPR(i,j);
                break;
            case 'b':SCfifoPR(i,j);
                    break;
            case 'c':LRU_algorithm(i,j);
                    break;
            default : printf("please try again");
        }

        // If current element is smaller than the pivot
        if (array[j] < pivot)
        {
            i++; // increment index of smaller element
            tmp=array[i];
            array[i]=array[j];
            array[j]=tmp;
        }
    }

    switch(input){
        case 'a':fifoPR(i+1,high);
                break;
        case 'b':SCfifoPR(i+1,high);
                break;
        case 'c':LRU_algorithm(i,j);
                break;
        default : printf("please try again");
    }


    tmp=array[i+1];
    array[i+1]=array[high];
    array[high]=tmp;
    return (i + 1);
}
```

checking array[i] and array[j]
in page replacement algorithm.

checking array[i+1] and array[high]
in page replacement algorithm.

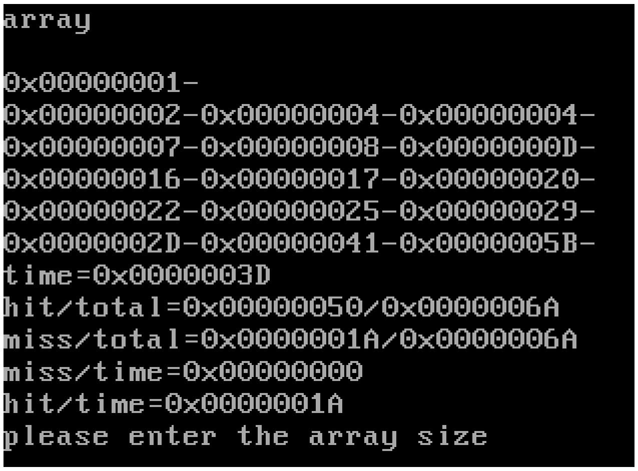
### 2.1) Quick Sort and FIFO RESULT



```
0x00000001-
0x00000002-0x00000004-0x00000004-
0x00000007-0x00000008-0x0000000D-
0x00000016-0x00000017-0x00000020-
0x00000022-0x00000025-0x00000029-
0x0000002D-0x00000041-0x0000005B-
time=0x0000000F
hit/total=0x0000004E/0x0000006E
miss/total=0x00000020/0x0000006E
miss/time=0x00000002
hit/time=0x00000020
please enter the array size
```

total=miss+hit
time=0x0000000F microfon

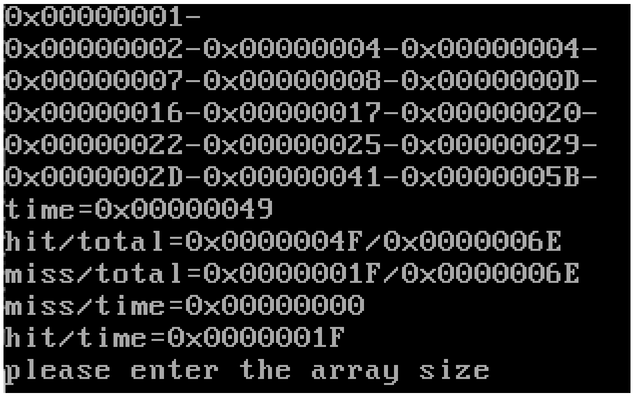
## 2.2) Quick Sort and SECOND CHANGE FIFO RESULT



```
array

0x00000001-
0x00000002-0x00000004-0x00000004-
0x00000007-0x00000008-0x0000000D-
0x00000016-0x00000017-0x00000020-
0x00000022-0x00000025-0x00000029-
0x0000002D-0x00000041-0x0000005B-
time=0x0000003D
hit/total=0x00000050/0x0000006A
miss/total=0x0000001A/0x0000006A
miss/time=0x00000000
hit/time=0x0000001A
please enter the array size
```

time=0x0000003D microfon

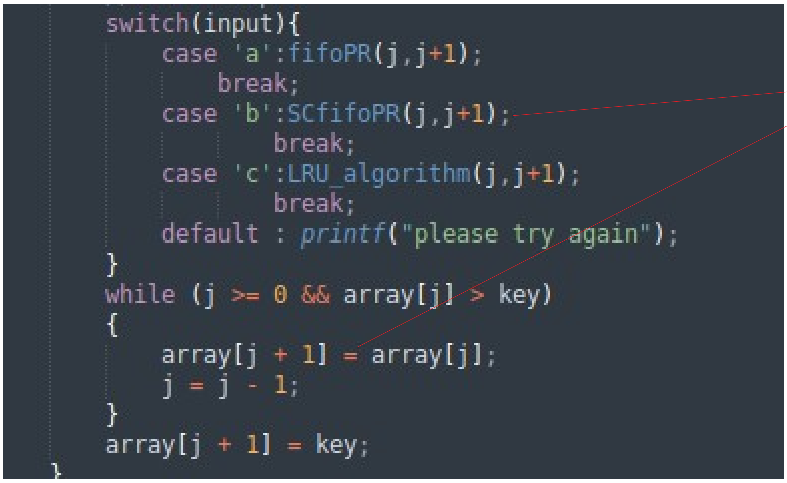
## 2.3) Quick Sort and SECOND CHANGE FIFO RESULT



```
0x00000001-
0x00000002-0x00000004-0x00000004-
0x00000007-0x00000008-0x0000000D-
0x00000016-0x00000017-0x00000020-
0x00000022-0x00000025-0x00000029-
0x0000002D-0x00000041-0x0000005B-
time=0x00000049
hit/total=0x0000004F/0x0000006E
miss/total=0x0000001F/0x0000006E
miss/time=0x00000000
hit/time=0x0000001F
please enter the array size
```

time=0x00000049 microfon

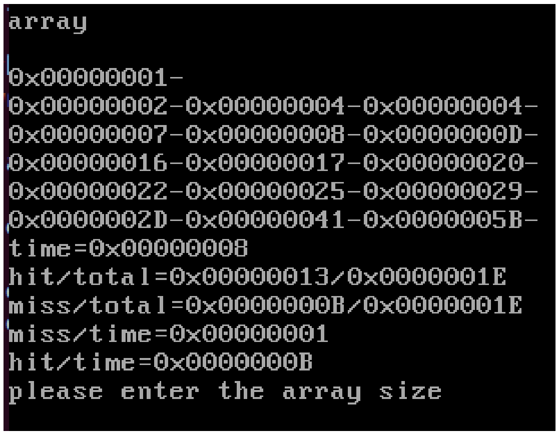
# 3) INSERTION SORT



```
    switch(input){
        case 'a':fifoPR(j,j+1);
            break;
        case 'b':SCfifoPR(j,j+1);
                break;
        case 'c':LRU_algorithm(j,j+1);
                break;
        default : printf("please try again");
    }
    while (j >= 0 && array[j] > key)
    {
        array[j + 1] = array[j];
        j = j - 1;
    }
    array[j + 1] = key;
}
```

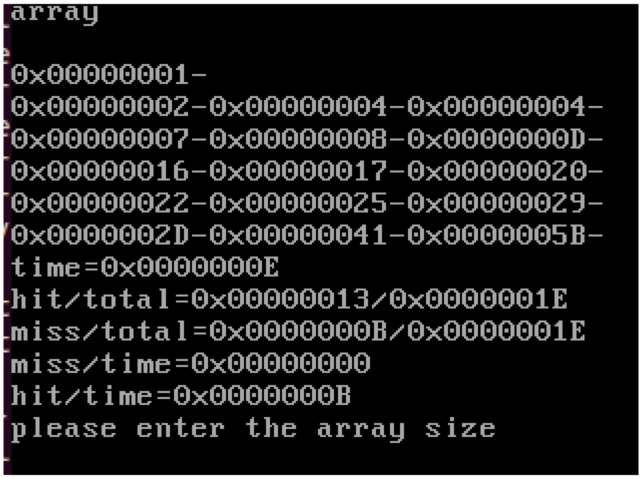
checking array[j] and array[j+1] in page replacement algorithm.

## 3.1) Insertion Sort and FIFO RESULT



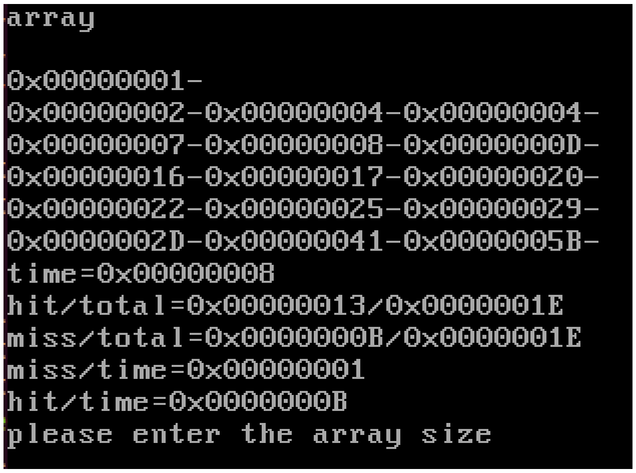
```
array

0x00000001-
0x00000002-0x00000004-0x00000004-
0x00000007-0x00000008-0x0000000D-
0x00000016-0x00000017-0x00000020-
0x00000022-0x00000025-0x00000029-
0x0000002D-0x00000041-0x0000005B-
time=0x00000008
hit/total=0x00000013/0x0000001E
miss/total=0x0000000B/0x0000001E
miss/time=0x00000001
hit/time=0x0000000B
please enter the array size
```

## 3.2) Insertion Sort and SECOND CHANGE FIFO RESULT



```
array

0x00000001-
0x00000002-0x00000004-0x00000004-
0x00000007-0x00000008-0x0000000D-
0x00000016-0x00000017-0x00000020-
0x00000022-0x00000025-0x00000029-
0x0000002D-0x00000041-0x0000005B-
time=0x0000000E
hit/total=0x00000013/0x0000001E
miss/total=0x0000000B/0x0000001E
miss/time=0x00000000
hit/time=0x0000000B
please enter the array size
```

## 3.3) Insertion Sort and SECOND CHANGE FIFO RESULT



```
array

0x00000001-
0x00000002-0x00000004-0x00000004-
0x00000007-0x00000008-0x0000000D-
0x00000016-0x00000017-0x00000020-
0x00000022-0x00000025-0x00000029-
0x0000002D-0x00000041-0x0000005B-
time=0x00000008
hit/total=0x00000013/0x0000001E
miss/total=0x0000000B/0x0000001E
miss/time=0x00000001
hit/time=0x0000000B
please enter the array size
```