Name: Phạm Mai Dung

ID: 19520477

Class: IT007.L21.1

OPERATING SYSTEM
LAB 6'S REPORT

SUMMARY

Self-scrores: 9

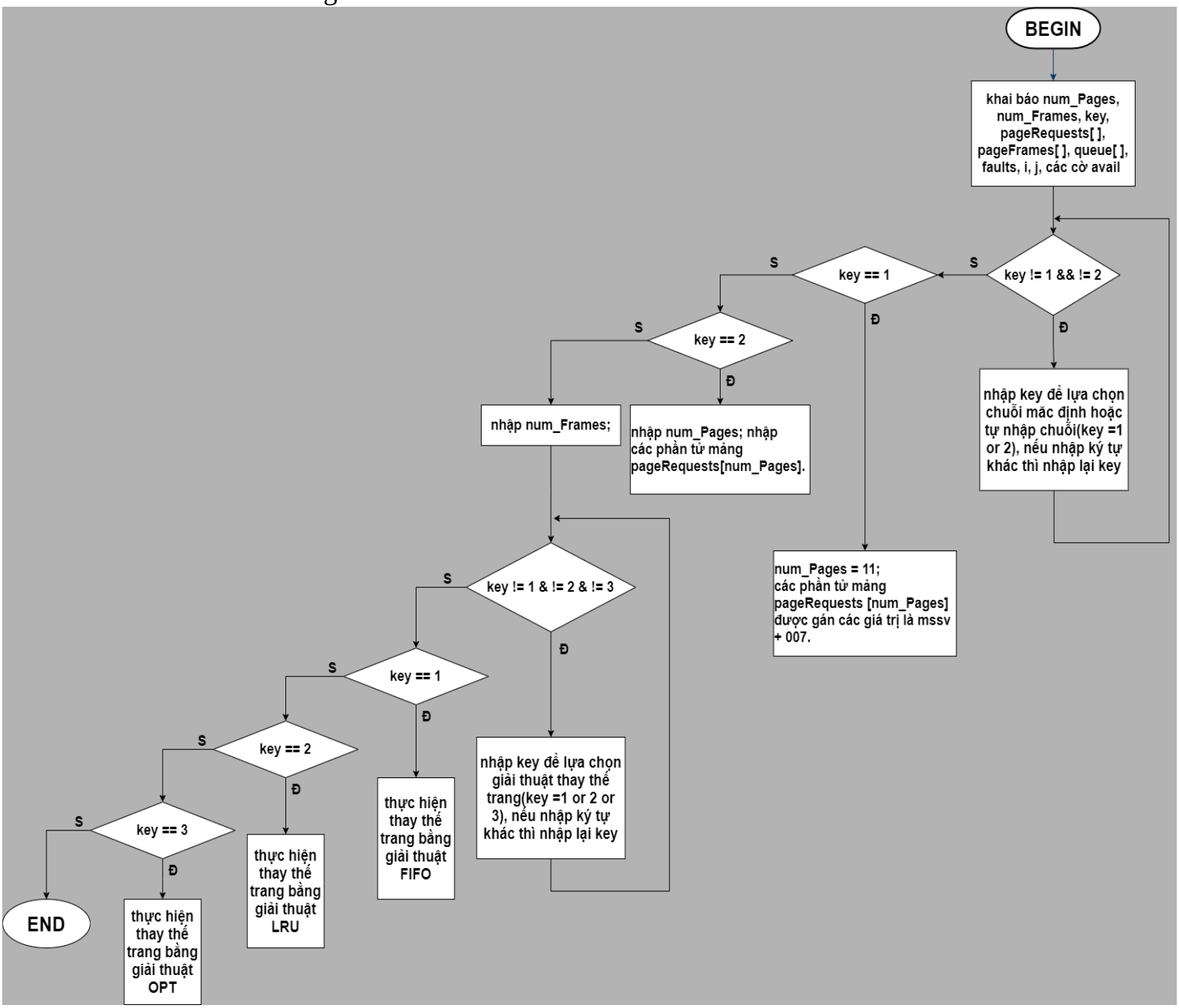
Note: Export file to **PDF and name the file by following format:*
LAB X – <Student ID>.pdf

Section 1.5

1. **Task name 1: Sử dụng ngôn ngữ lập trình C viết chương trình mô phỏng các giải thuật thay thế trang đã nêu trong câu số 3 mục 6.3.3 với các yêu cầu trong Lab6.pdf .**

- Thực hiện chương trình:

+ Lưu đồ chung:



Hình 1 _ Lưu đồ thực hiện chương trình

+ Source code:

```c
/*#####################################
# University of Information Technology #
# IT007 Operating System               #
# Pham Mai Dung, 19520477              #
# File: page_Replacement_Algorithm.c   #
#####################################*/

#include<stdio.h>

int pos_EarliestPage_LRU(int array[], int n)
{
    int i, min = array[0], pos = 0;
    for(i = 1; i < n; ++i)
    {
        if(array[i] < min)
        {
            min = array[i];
            pos = i;
        }
    }
    return pos;
}

int main()
{
    int key;
    int numOfPages, numOfFrames;
    int pageRequests[100], pageFrames[10], queue[10];
    int i, j;
    int faults = 0;
    int avail1, avail2, avail3;
    //choose Default referenced sequence or Manual input sequence
    do
    {
        printf("\n--- Page Replacement algorithm ---\n1. Default referenced sequence\n2. Manual input sequence\n");
        printf("Enter your selection: ");
        scanf("%d", &key);
        if (key != 1 && key != 2)
        {
            printf("\nInvalid syntax. Please re-enter!\n");
        }
    } while (key != 1 && key != 2);

```

```c
    //Default referenced sequence: MSSV + 007. [1, 9, 5, 2, 0, 4, 7, 7, 0, 0, 7]
    if(key == 1)
    {
        numOfPages = 11;
        pageRequests[0] = 1;
        pageRequests[1] = 9;
        pageRequests[2] = 5;
        pageRequests[3] = 2;
        pageRequests[4] = 0;
        pageRequests[5] = 4;
        pageRequests[6] = 7;
        pageRequests[7] = 7;
        pageRequests[8] = 0;
        pageRequests[9] = 0;
        pageRequests[10] = 7;
    }

    //Manual input sequence
    if(key == 2)
    {
        printf("\nEnter number of page requests: ");
        scanf("%d", &numOfPages);
        printf("Input the page requests separated by space: \n");
        for (i = 0; i < numOfPages; i++)
        {
            printf("Value No. [%d]:\t", i + 1);
            scanf("%d", &pageRequests[i]);
        }
    }

    //Enter page frames
    printf("\n--- Page Replacement algorithm ---\nEnter page frames: ");
    scanf("%d", &numOfFrames);

    //choose the page replacement algorithm
    do
    {
        printf("\n--- Page Replacement algorithm ---\n1. FIFO algorithm\n2. LRU algorithm\n3. OPT algorithm\n");
        printf("Enter your selection: ");
        scanf("%d", &key);
        if (key != 1 && key != 2 && key != 3)
        {
            printf("\nInvalid syntax. Please re-enter!\n");
        }
    } while (key != 1 && key != 2 && key != 3);

    //FIFO page replacement algorithm
    if (key == 1)
    {
        printf("\n--- Page Replacement algorithm ---\nFIFO algorithm:\n");
        int index;
        for (i = 0; i < numOfFrames; i++)
        {
            pageFrames[i] = -1;
        }
        index = 0;
        printf("\nPages\t|\t\tFrames\t\t\t|\n");
        for (i = 0; i < numOfPages; i++)
        {
            printf("%d\t\t", pageRequests[i]);
            avail1 = 0;
            for(j = 0; j < numOfFrames; j++)
            {
                if(pageFrames[j] == pageRequests[i])
                {
                    avail1 = 1;
                }
            }
```

```
            if(avail1 == 0)
            {
                pageFrames[index] = pageRequests[i];
                index = (index + 1) % numOfFrames;
                faults++;
                for(j = 0; j < numOfFrames; j++)
                {
                    printf("%d\t", pageFrames[j]);
                }
                printf("\t*");
            }
            else
            {
                for(j = 0; j < numOfFrames; j++)
                {
                    printf("%d\t", pageFrames[j]);
                }
            }
            printf("\n");
        }
        printf("\nNumber of Page faults : %d\n", faults);
    }

    //LRU page replacement algorithm
    if  (key == 2)
    {
        printf("\n--- Page Replacement algorithm ---\nLRU algorithm:\n");
        int count = 0, pos;
        for (i = 0; i < numOfFrames; ++i)
        {
            pageFrames[i] = -1;
        }
        printf("\nPages\t|\t\tFrames\t\t\t|\n");
        for (i = 0; i < numOfPages; i++)
        {
            avail1 = avail2 = 0;
            for(j = 0; j < numOfFrames; ++j)
            {
                if(pageFrames[j] == pageRequests[i])
                {
                count++;
                queue[j] = count;
                avail1 = avail2 = 1;
                break;
                }
            }
            if(avail1 == 0)
            {
                for(j = 0; j < numOfFrames; ++j)
                {
                    if(queue[j] == -1)
                    {
                        count++;
                        faults++;
                        pageFrames[j] = pageRequests[i];
                    queue[j] = count;
                    avail2 = 1;
                    break;
                    }
                }
            }
```

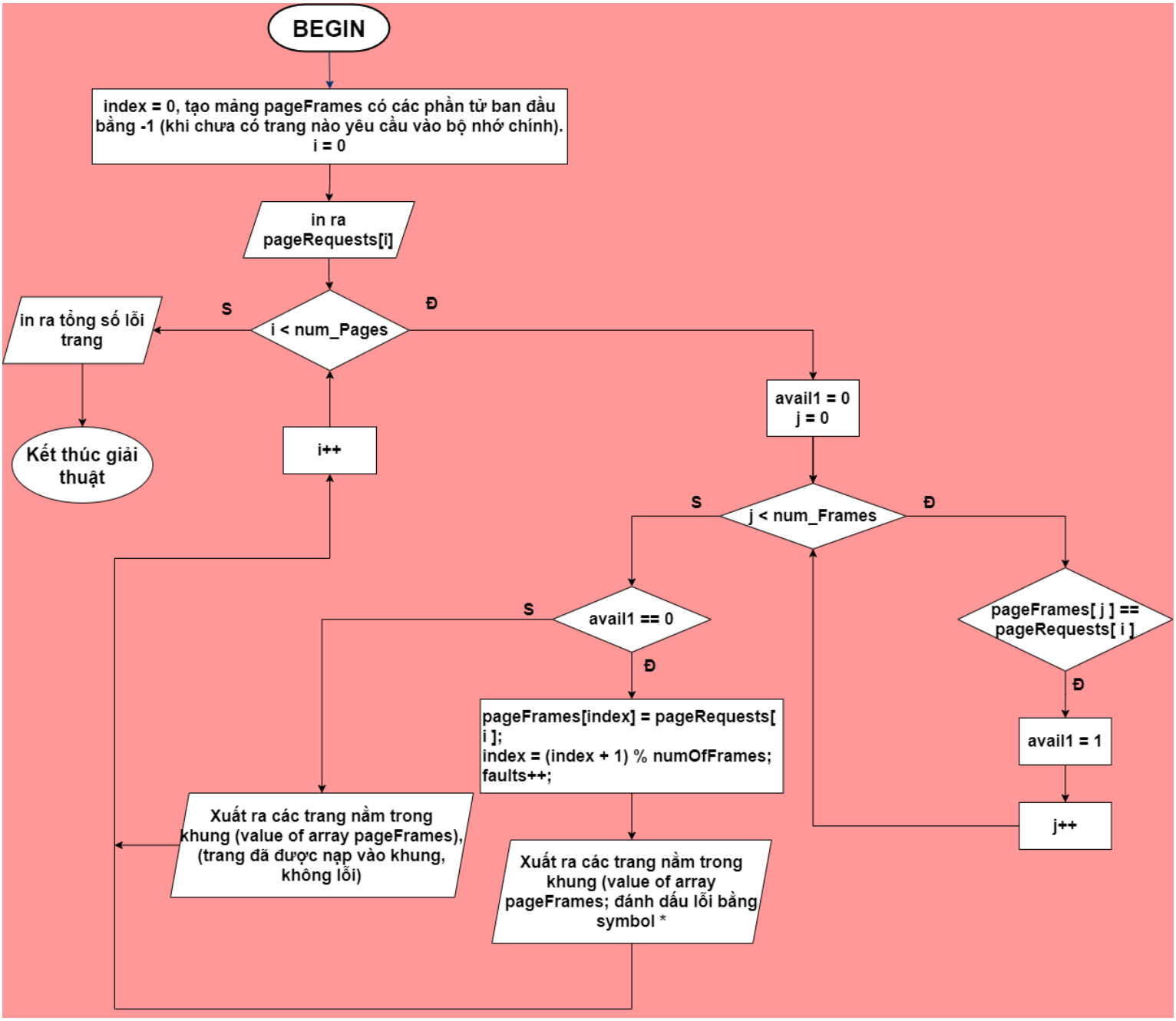
```
216             if(avail1 == 0)
217             {
218                 for(j = 0; j < numOfFrames; ++j)
219                 {
220                     if(pageFrames[j] == -1)
221                     {
222                         faults++;
223                         pageFrames[j] = pageRequests[i];
224                     avail2 = 1;
225                     break;
226                     }
227                 }
228             }
229             if(avail2 == 0)
230             {
231             avail3 = 0;
232             for(j = 0; j < numOfFrames; ++j)
233             {
234                 queue[j] = -1;
235                 for(m = i + 1; m < numOfPages; ++m)
236                 {
237                     if(pageFrames[j] == pageRequests[m])
238                     {
239                         queue[j] = m;
240                         break;
241                     }
242                 }
243             }
244             for(j = 0; j < numOfFrames; ++j)
245             {
246                 if(queue[j] == -1)
247                 {
248                     pos = j;
249                     avail3 = 1;
250                     break;
251                 }
252             }
253             if(avail3 == 0)
254             {
255                 latest = queue[0];
256                 pos = 0;
257                 for(j = 1; j < numOfFrames; ++j)
258                 {
259                     if(queue[j] > latest)
260                     {
261                         latest = queue[j];
262                         pos = j;
263                     }
264                 }
265             }
266             pageFrames[pos] = pageRequests[i];
267             faults++;
268             }
```

```
269             printf("\n");
270             printf("%d\t\t", pageRequests[i]);
271             for(j = 0; j < numOfFrames; j++)
272             {
273                 printf("%d\t", pageFrames[j]);
274             }
275             if(avail1 == 0 || avail2 == 0)
276             {
277                 printf("\t*");
278             }
279         }
280         printf("\nNumber of Page faults : %d\n", faults);
281     }
282     return 0;
283 }
284
```

Hình 2_ Source code chương trình các giải thuật thay thế trang

a. Giải thuật thay thế trang FIFO:

+ Lưu đồ giải thuật:



Hình 3_ Lưu đồ FIFO Algorithm

+ Giải thích:

- Trước khi duyệt mảng pageRequests để các trang đi vào bộ nhớ chính, ta gán cho các phần tử của mảng pageFrames (có num_Frames phần tử là số khung trang) giá trị -1.

- Duyệt qua các phần tử của mảng trang yêu cầu, nếu trang chưa có trong khung thì đặt cờ avail1 = 0 để thêm trang đó vào khung, đồng thời đánh dấu lỗi trang, đếm lỗi trang lên 1, “index = (index+1)%num_Frames” sẽ xét khi khung đã đầy thì trang mới vào tiếp theo sẽ thay cho trang vào trước nhất trong số num_Frames. (ví dụ có 3 khung trang, duyệt qua i sẽ là 0, 1, 2, sau khi frames[2] = pages[2] thì index = (2 + 1)% 3 = 0 thì sẽ thay trang tại frames [index == 0], vòng lặp tiếp theo sẽ thay tại frames[index == 1]). Sau đó xuất ra khung trang và đánh dấu * để báo hiệu xảy ra lỗi trang. Nếu trang đó đã có trong khung thì cờ avail = 1, không có lỗi nên chỉ cần xuất ra khung trang.
- Khi đã duyệt hết các phần tử mảng pageRequests, xuất ra tổng số lỗi trang là tổng số faults đã đếm được và kết thúc chương trình.

+ Kết quả:

- Chuỗi mặc định: [19520477 + 007]:

```
dung-19520477@dung19520477-VirtualBox:~/lab6-os$ gcc page_Replacement_Algorithm
.c -o Lab06
dung-19520477@dung19520477-VirtualBox:~/lab6-os$ ./Lab06

--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
Enter your selection: 1

--- Page Replacement algorithm ---
1. FIFO algorithm
2. LRU algorithm
3. OPT algorithm
Enter your selection: 1

--- Page Replacement algorithm ---
FIFO algorithm:

Pages   |              Frames                |
1               1       -1      -1              *
9               1       9       -1              *
5               1       9       5               *
2               2       9       5               *
0               2       0       5               *
4               2       0       4               *
7               7       0       4               *
7               7       0       4
0               7       0       4
0               7       0       4
7               7       0       4
```

Hình 4 _ Kết quả khi thực hiện FIFO với 3 frames cho chuỗi mặc định.

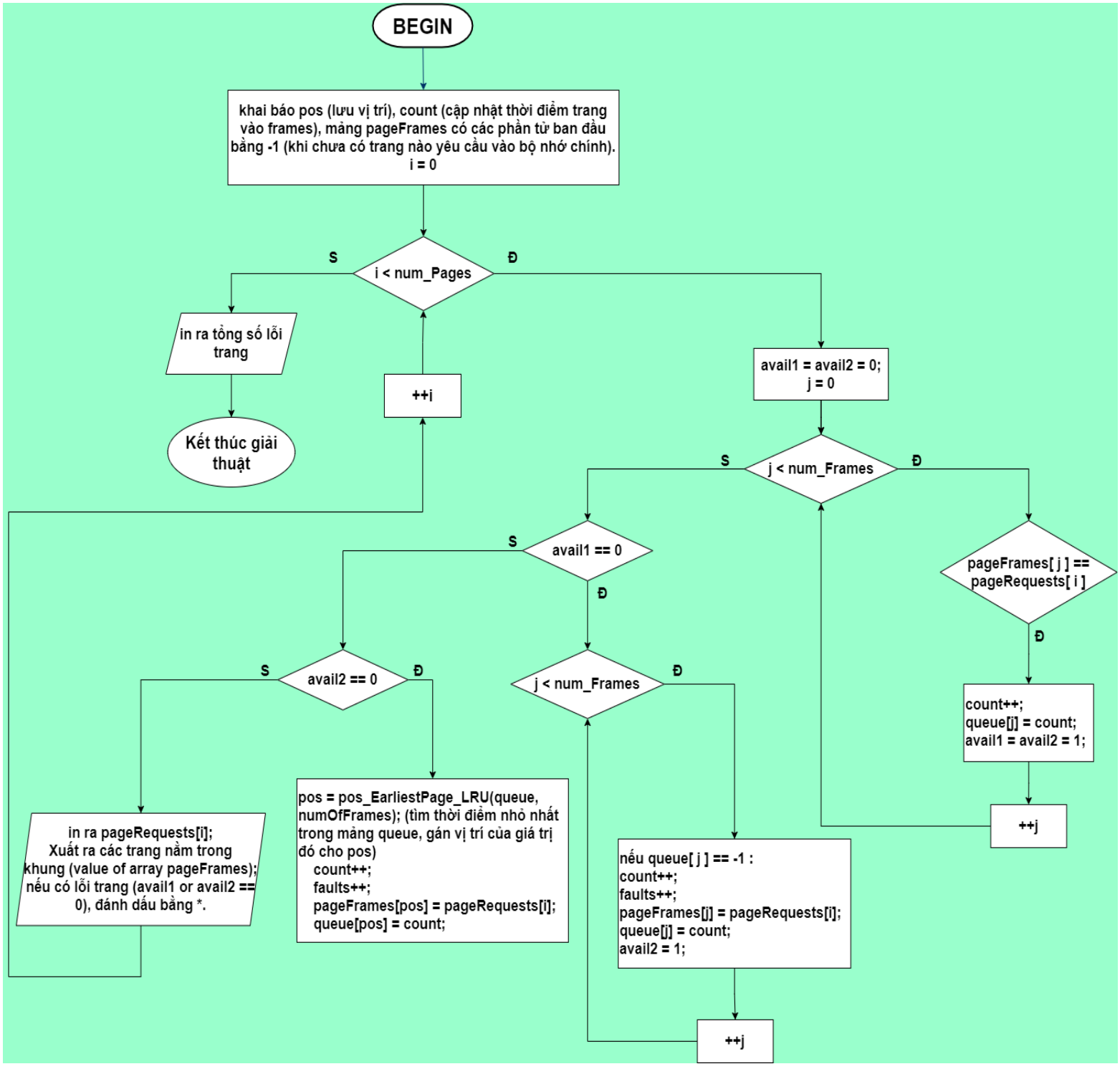
- Nhập chuỗi: (chuỗi trong ví dụ của bài lab):

```
dung-19520477@dung19520477-VirtualBox:~/lab6-os$ ./Lab06

--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
Enter your selection: 2

Enter number of page requests: 12
Input the page requests separated by space:
Value No. [1]:  1
Value No. [2]:  2
Value No. [3]:  3
Value No. [4]:  4
Value No. [5]:  1
Value No. [6]:  2
Value No. [7]:  5
Value No. [8]:  1
Value No. [9]:  2
Value No. [10]: 3
Value No. [11]: 4
Value No. [12]: 5

--- Page Replacement algorithm ---
Enter page frames: 3

--- Page Replacement algorithm ---
1. FIFO algorithm
2. LRU algorithm
3. OPT algorithm
Enter your selection: 1

--- Page Replacement algorithm ---
FIFO algorithm:

Pages   |              Frames                |
1               1       -1      -1              *
2               1       2       -1              *
3               1       2       3               *
4               4       2       3               *
1               4       1       3               *
2               4       1       2               *
5               5       1       2               *
1               5       1       2
2               5       1       2
3               5       3       2               *
4               5       3       4               *
5               5       3       4

Number of Page faults : 9
dung-19520477@dung19520477-VirtualBox:~/lab6-os$
```

Hình 5 _ Kết quả thực hiện FIFO với 3 frames cho chuỗi đã nhập.

b. Giải thuật thay thế trang LRU:

+ Lưu đồ giải thuật:



Hình 6_ Lưu đồ LRU Algorithm

+ Giải thích:

- Trước khi duyệt mảng pageRequests để các trang đi vào bộ nhớ chính, ta gán cho các phần tử của mảng pageFrames (có num_Frames phần tử là số khung trang) giá trị -1; mảng queue (có num_Frames phần tử) sẽ lưu thời điểm pages đi vào frames. Thời điểm sẽ tăng dần theo số pages vào, được đếm bằng biến count. Đặt các cờ avail1, avail2 = 0.
- Duyệt qua các phần tử của mảng trang yêu cầu, nếu trang chưa có trong khung (avail1 = 0) thì thêm trang đó vào khung, đồng thời đếm lỗi trang và biến lưu thời điểm lên 1, gán thời điểm cho phần tử j của mảng queue. Đặt cờ avail2 = 1.
- Nếu cờ avail2 = 0, mảng frames đầy mà trang mới vào lại không có trong frames, tìm thời điểm nhỏ nhất trong mảng queue, gán phần tử có thời điểm nhỏ nhất đó cho pos. Sau đó tiếp tục tăng thời điểm, đếm lỗi lên 1, frames[pos] = pages[i] và queue[pos] = count (gán trang mới vào vị trí pos trong mảng frames và cập nhật thời điểm mới ở vị trí pos).
- Nếu trang đó đã có trong frames, cập nhật thời điểm và đặt các cờ bằng 1 (không xảy ra lỗi trang).
- Sau mỗi vòng lặp thay trang, in ra số trang đó và khung của nó, nếu có lỗi trang thì đánh dấu * báo hiệu. Khi đã duyệt hết các phần tử mảng pageRequests, xuất ra tổng số lỗi trang là tổng số faults đã đếm được và kết thúc chương trình.

+ Kết quả:

- Chuỗi mặc định: [19520477 + 007]:

```
dung-19520477@dung19520477-VirtualBox:~/lab6-os$ ./Lab06

--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
Enter your selection: 1

--- Page Replacement algorithm ---
Enter page frames: 4

--- Page Replacement algorithm ---
1. FIFO algorithm
2. LRU algorithm
3. OPT algorithm
Enter your selection: 2
--- Page Replacement algorithm ---
LRU algorithm:

Pages   |               Frames                  |

1               1       -1      -1      -1              *
9               1       9       -1      -1              *
5               1       9       5       -1              *
2               1       9       5       2               *
0               0       9       5       2               *
4               0       4       5       2               *
7               0       4       7       2               *
7               0       4       7       2
0               0       4       7       2
0               0       4       7       2
7               0       4       7       2
Number of Page faults : 7
dung-19520477@dung19520477-VirtualBox:~/lab6-os$
```

Hình 7 _ Kết quả khi thực hiện LRU với 4 frames cho chuỗi mặc định.

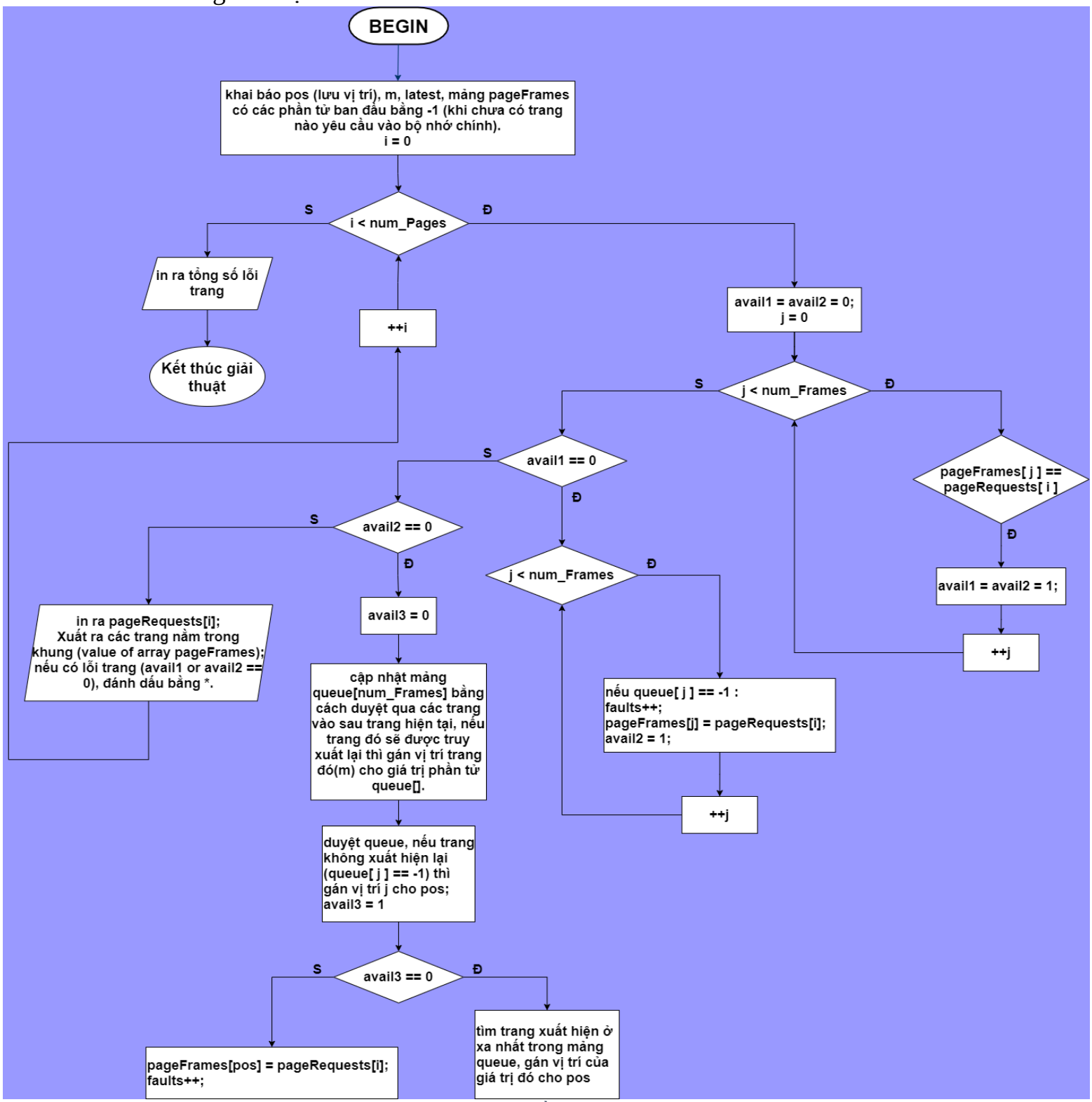
- Nhập chuỗi: (chuỗi trong ví dụ của bài lab):

```
dung-19520477@dung19520477-VirtualBox:~/lab6-os$ ./Lab06

--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
Enter your selection: 2

Enter number of page requests: 12
Input the page requests separated by space:
Value No. [1]:  1
Value No. [2]:  2
Value No. [3]:  3
Value No. [4]:  4
Value No. [5]:  1
Value No. [6]:  2
Value No. [7]:  5
Value No. [8]:  1
Value No. [9]:  2
Value No. [10]: 3
Value No. [11]: 4
Value No. [12]: 5

--- Page Replacement algorithm ---
Enter page frames: 3

--- Page Replacement algorithm ---
1. FIFO algorithm
2. LRU algorithm
3. OPT algorithm
Enter your selection: 2

--- Page Replacement algorithm ---
LRU algorithm:

Pages   |              Frames                  |

1               1       -1      -1              *
2               1       2       -1              *
3               1       2       3               *
4               4       2       3               *
1               4       1       3               *
2               4       1       2               *
5               5       1       2               *
1               5       1       2
2               5       1       2
3               3       1       2               *
4               3       4       2               *
5               3       4       5               *
Number of Page faults : 10
dung-19520477@dung19520477-VirtualBox:~/lab6-os$
```

Hình 8 _ Kết quả thực hiện LRU với 3 frames cho chuỗi đã nhập.

c. Giải thuật thay thế trang OPT:

+ Lưu đồ giải thuật:



Hình 9 _ Lưu đồ OPT Algorithm

+ Giải thích:

- Trước khi duyệt mảng pageRequests để các trang đi vào bộ nhớ chính, ta gán cho các phần tử của mảng pageFrames (có num_Frames phần tử là số khung trang) giá trị -1; mảng queue (có num_Frames phần tử) sẽ lưu vị trí pages sau page hiện tại sẽ đi vào frames. Biến m sẽ cập nhật giá trị cho các phần tử mảng queue. Đặt các cờ avail1, avail2 = 0.
- Duyệt qua các phần tử của mảng trang yêu cầu, nếu trang chưa có trong khung (avail1 = 0) thì thêm trang đó vào khung, đồng thời đếm lỗi trang lên 1, đặt cờ avail2 = 1.
- Nếu cờ avail2 = 0, mảng frames đầy, đặt cờ avail3 = 0. Cập nhật mảng queue bằng cách duyệt qua các trang vào sau trang hiện tại, nếu trang đó sẽ được truy xuất lại thì gán vị trí trang đó(m) cho phần tử queue[j]. Duyệt queue, nếu trang không xuất hiện lại thì gán vị trí j cho pos. Nếu trang sẽ xuất hiện lại (avail3 == 0), tìm trang xuất hiện ở xa nhất trong mảng queue, gán vị trí của giá trị đó cho pos. Sau mỗi lần pos được cập nhật, gán trang mới vào vị trí pos trong mảng frames và đếm biến lỗi lên 1.
- Nếu trang đó đã có trong frames, đặt các cờ bằng 1 (không xảy ra lỗi trang).
- Sau mỗi vòng lặp thay trang, in ra số trang đó và khung của nó, nếu có lỗi trang thì đánh dấu * báo hiệu. Khi đã duyệt hết các phần tử mảng pageRequests, xuất ra tổng số lỗi trang là tổng số faults đã đếm được và kết thúc chương trình.

+ Kết quả:

- Chuỗi mặc định: [19520477 + 007]:

```
dung-19520477@dung19520477-VirtualBox:~/lab6-os$ ./Lab06

--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
Enter your selection: 1

--- Page Replacement algorithm ---
Enter page frames: 4

--- Page Replacement algorithm ---
1. FIFO algorithm
2. LRU algorithm
3. OPT algorithm
Enter your selection: 3

--- Page Replacement algorithm ---
OPT algorithm:

Pages   |               Frames                  |

1               1       -1      -1      -1              *
9               1       9       -1      -1              *
5               1       9       5       -1              *
2               1       9       5       2               *
0               0       9       5       2               *
4               0       4       5       2               *
7               0       7       5       2               *
7               0       7       5       2
0               0       7       5       2
0               0       7       5       2
7               0       7       5       2
Number of Page faults : 7
dung-19520477@dung19520477-VirtualBox:~/lab6-os$
```

Hình 10 _ Kết quả khi thực hiện OPT với 4 frames cho chuỗi mặc định.

- Nhập chuỗi: (chuỗi trong ví dụ của bài lab):

```
dung-19520477@dung19520477-VirtualBox:~/lab6-os$ ./Lab06

--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
Enter your selection: 2

Enter number of page requests: 12
Input the page requests separated by space:
Value No. [1]:  1
Value No. [2]:  2
Value No. [3]:  3
Value No. [4]:  4
Value No. [5]:  1
Value No. [6]:  2
Value No. [7]:  5
Value No. [8]:  1
Value No. [9]:  2
Value No. [10]: 3
Value No. [11]: 4
Value No. [12]: 5

--- Page Replacement algorithm ---
Enter page frames: 3

--- Page Replacement algorithm ---
1. FIFO algorithm
2. LRU algorithm
3. OPT algorithm
Enter your selection: 3

--- Page Replacement algorithm ---
OPT algorithm:

Pages   |              Frames                  |

1               1       -1      -1              *
2               1       2       -1              *
3               1       2       3               *
4               1       2       4               *
1               1       2       4
2               1       2       4
5               1       2       5               *
1               1       2       5
2               1       2       5
3               3       2       5               *
4               4       2       5               *
5               4       2       5
Number of Page faults : 7
dung-19520477@dung19520477-VirtualBox:~/lab6-os$
```

Hình 11 _ Kết quả thực hiện OPT với 3 frames cho chuỗi đã nhập.

2. Task name 2: Nhận xét về mức độ hiệu quả và tính khả thi của các giải thuật FIFO, OPT, LRU.

- Giải thuật OPT là giải thuật tối ưu nhất, ít xảy ra lỗi trang nhất trong 3 giải thuật trên nhưng OPT cũng là giải thuật bất khả thi nhất. Vì ta không thể biết người dùng sẽ yêu cầu truy xuất trang nào trước trang nào sau ở tương lai, thậm chí người dùng cũng không thể kiểm soát được việc đó.
- Giải thuật FIFO đơn giản, dễ lập trình, ít tốn tài nguyên máy nhưng không tối ưu vì lỗi trang nhiều và 1 vài trường hợp gặp nghịch lý Belady.
- 2 giải thuật LRU và OPT là 2 giải thuật tối ưu hơn nhưng phức tạp hơn vì cần nhiều tài nguyên máy và cài đặt thuật toán cũng phức tạp hơn.

3. Task name 3: Nghịch lý Belady:

- Nghịch lý Belady: số page faults tăng mặc dù tiến trình đã được cấp nhiều frames hơn.
- Nghịch lý này xuất hiện trong FIFO algorithm: (sử dụng ví dụ trong bài lab)

```
--- Page Replacement algorithm ---
1. FIFO algorithm
2. LRU algorithm
3. OPT algorithm
Enter your selection: 1

--- Page Replacement algorithm ---
FIFO algorithm:

Pages   |               Frames                  |
1               1       -1      -1              *
2               1       2       -1              *
3               1       2       3               *
4               4       2       3               *
1               4       1       3               *
2               4       1       2               *
5               5       1       2               *
1               5       1       2
2               5       1       2
3               5       3       2               *
4               5       3       4               *
5               5       3       4

Number of Page faults : 9
dung-19520477@dung19520477-VirtualBox:~/lab6-os$
```

Hình 12 _ Khi cấp 3 frames, giải thuật FIFO xảy ra 9 lỗi trang

```
--- Page Replacement algorithm ---
Enter page frames: 4

--- Page Replacement algorithm ---
1. FIFO algorithm
2. LRU algorithm
3. OPT algorithm
Enter your selection: 1

--- Page Replacement algorithm ---
FIFO algorithm:

Pages   |               Frames                    |
1               1       -1      -1      -1              *
2               1       2       -1      -1              *
3               1       2       3       -1              *
4               1       2       3       4               *
1               1       2       3       4
2               1       2       3       4
5               5       2       3       4               *
1               5       1       3       4               *
2               5       1       2       4               *
3               5       1       2       3               *
4               4       1       2       3               *
5               4       5       2       3               *

Number of Page faults : 10
dung-19520477@dung19520477-VirtualBox:~/lab6-os$
```

Hình 13 _ Khi được cấp 4 frames, lỗi trang lại nhiều hơn (xảy ra 10 lỗi trang)

- Trong khi đó, OPT khắc phục được nghịch lý này:

```
--- Page Replacement algorithm ---
1. FIFO algorithm
2. LRU algorithm
3. OPT algorithm
Enter your selection: 3

--- Page Replacement algorithm ---
OPT algorithm:

Pages   |               Frames                  |

1               1       -1      -1              *
2               1       2       -1              *
3               1       2       3               *
4               1       2       4               *
1               1       2       4
2               1       2       4
5               1       2       5               *
1               1       2       5
2               1       2       5
3               3       2       5               *
4               4       2       5               *
5               4       2       5
Number of Page faults : 7
dung-19520477@dung19520477-VirtualBox:~/lab6-os$
```

Hình 14 _ Giải thuật OPT chỉ xảy ra 7 lỗi trang khi cùng thực hiện 1 chuỗi trang yêu cầu với 3 frames được cấp.

```
--- Page Replacement algorithm ---
Enter page frames: 4

--- Page Replacement algorithm ---
1. FIFO algorithm
2. LRU algorithm
3. OPT algorithm
Enter your selection: 3

--- Page Replacement algorithm ---
OPT algorithm:

Pages   |               Frames                  |

1               1       -1      -1      -1              *
2               1       2       -1      -1              *
3               1       2       3       -1              *
4               1       2       3       4               *
1               1       2       3       4
2               1       2       3       4
5               1       2       3       5               *
1               1       2       3       5
2               1       2       3       5
3               1       2       3       5
4               4       2       3       5               *
5               4       2       3       5
Number of Page faults : 6
dung-19520477@dung19520477-VirtualBox:~/lab6-os$
```

Hình 15_ Khi được cấp 4 frames, lỗi trang ít hơn (xảy ra 6 lỗi trang) với giải thuật OPT