Name: Hà Danh Tuấn

ID: 20522109

Class: IT007.M13

OPERATING SYSTEM
LAB 6'S REPORT

SUMMARY

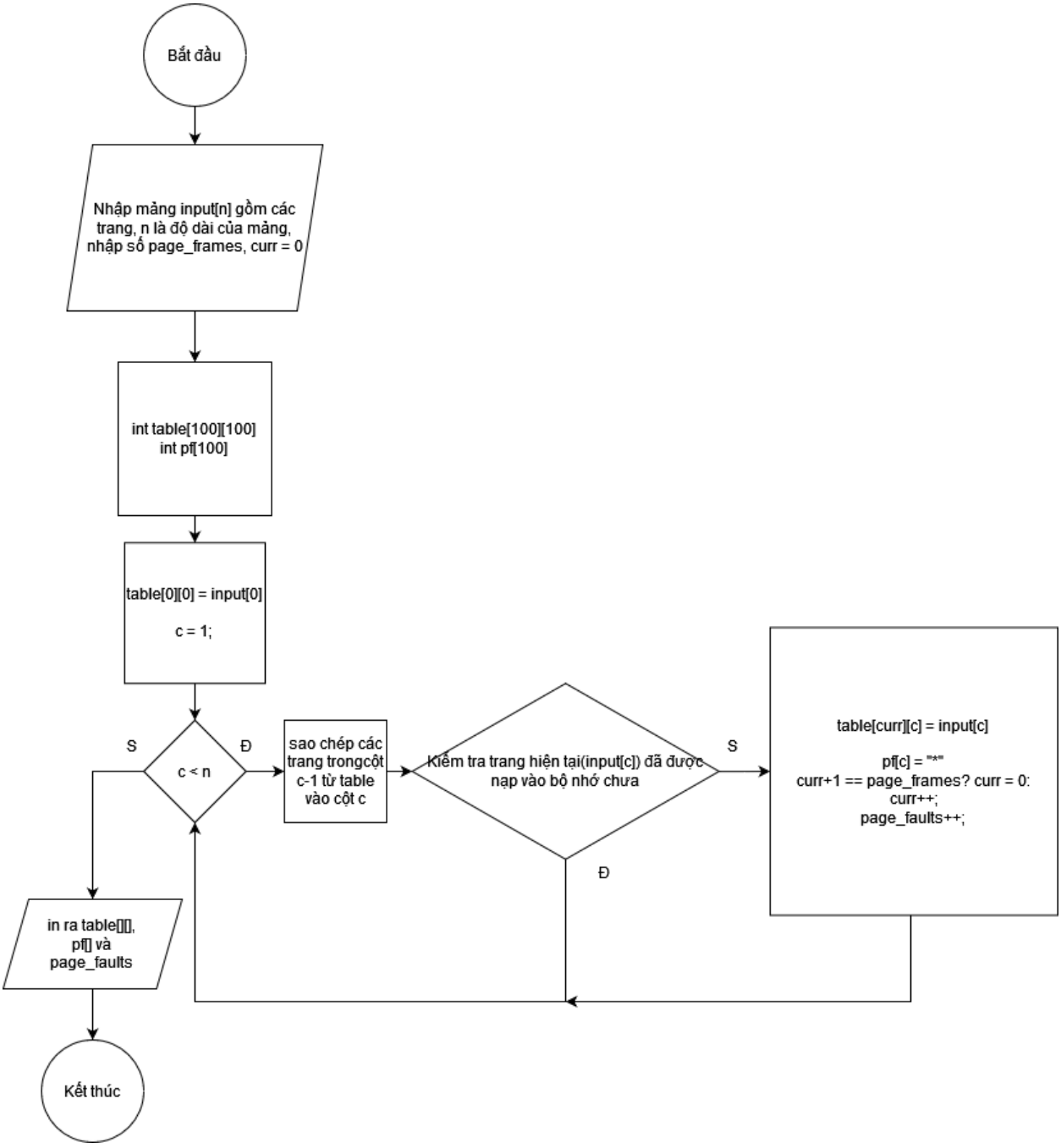
Task		Status	Page
6.4	Lưu đồ	Hoàn thành	2
	Hiện thực thuật toán	Hoàn thành	5
	Giải thích	Hoàn thành	11
	Test case	Hoàn thành	13
6.5	Câu 1	Hoàn thành	20
	Câu 2	Hoàn thành	23

Self-scrores: 8

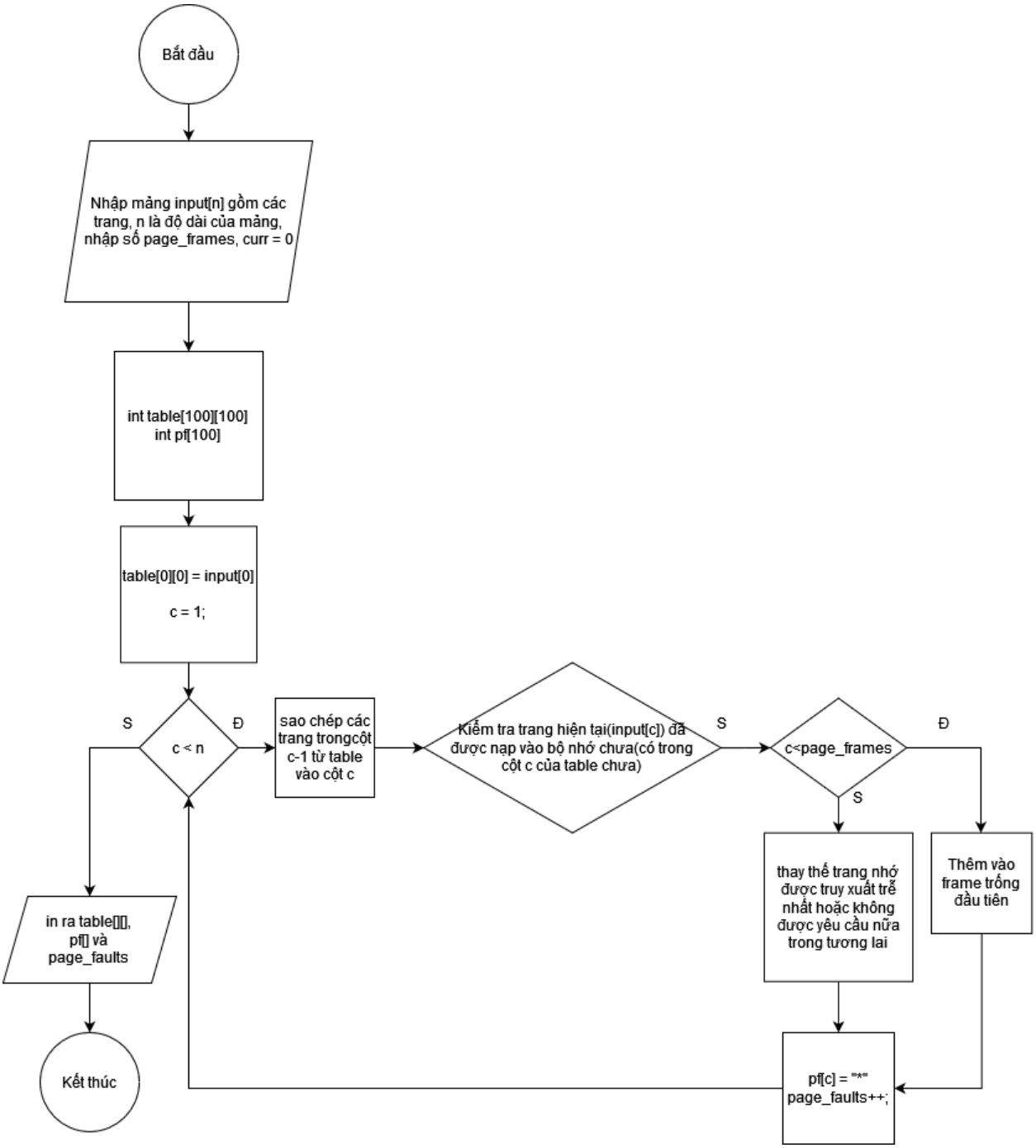
6.4

Lưu đồ thuật toán

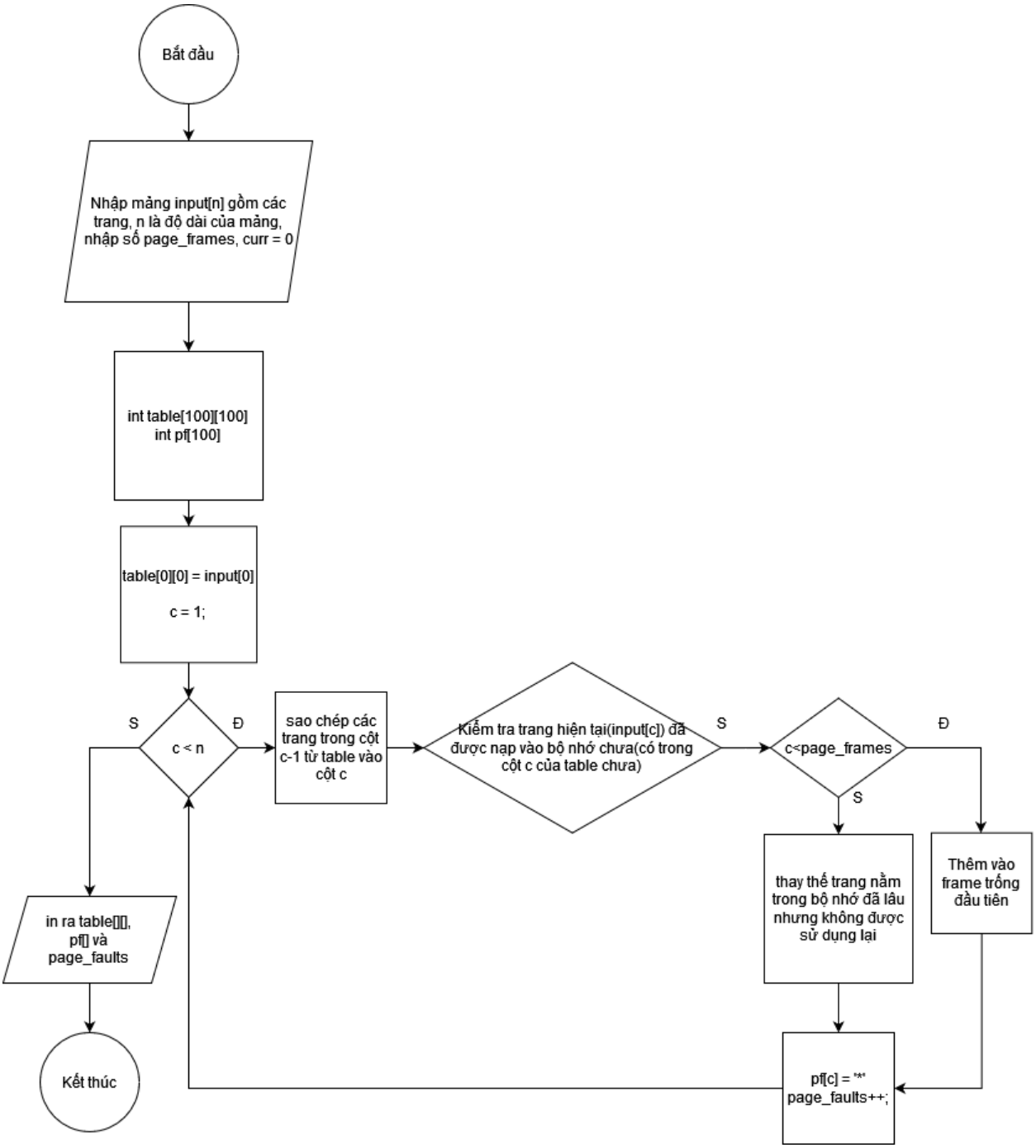
FIFO:



OPT:


Bắt đầu
Nhập mảng input[n] gồm các trang, n là độ dài của mảng, nhập số page_frames, curr = 0
int table[100][100]
int pf[100]
table[0][0] = input[0]
c = 1;
S
c < n
Đ
sao chép các trang trongcột c-1 từ table vào cột c
Kiểm tra trang hiện tại(input[c]) đã được nạp vào bộ nhớ chưa(có trong cột c của table chưa)
S
c<page_frames
Đ
S
thay thế trang nhớ được truy xuất trễ nhất hoặc không được yêu cầu nữa trong tương lai
Thêm vào frame trống đầu tiên
in ra table[][], pf[] và page_faults
Kết thúc
pf[c] = "*"
page_faults++;

LRU:


Bắt đầu
Nhập mảng input[n] gồm các trang, n là độ dài của mảng, nhập số page_frames, curr = 0
int table[100][100]
int pf[100]
table[0][0] = input[0]
c = 1;
S
c < n
Đ
sao chép các trang trong cột c-1 từ table vào cột c
Kiểm tra trang hiện tại(input[c]) đã được nạp vào bộ nhớ chưa(có trong cột c của table chưa)
S
c<page_frames
Đ
S
thay thế trang nằm trong bộ nhớ đã lâu nhưng không được sử dụng lại
Thêm vào frame trống đầu tiên
in ra table[][], pf[] và page_faults
Kết thúc
pf[c] = '*'
page_faults++;

Hiện thực 3 giải thuật:


```cpp
#include <iostream>
#include <climits>
using namespace std;

char pf[100];  // page fault array
int table[100][100];
int page_faults = 0; // page frames, ref length, page faults

int findInCol(int x, int c, int page_frames)
{
        for(int i=0; i<page_frames; i++)
                if(table[i][c] == x)
                        return i;
        return -1;
}

void FIFO(int input[], int n, int page_frames)
{

        int curr= 0;
        table[0][0] = input[0];
        page_faults++;
        pf[0] = '*';

        for(int i=1; i<page_frames; i++)
        {
                table[i][0] = -1;
        }


        for(int c=1; c<n; c++)
        {
                for(int r=0; r<page_frames; r++)
                {
                        table[r][c] = table[r][c-1];
                }

                bool flag = false;
                pf[c] = ' ';

                if(findInCol(input[c], c, page_frames) == -1)
                        flag = true;

                if(flag)
                {
                        page_faults++;
                        pf[c] = '*';
```

```
                if(flag)
                {
                        page_faults++;
                        pf[c] = '*';
                        curr + 1 == page_frames ? curr = 0 : curr++;

                        table[curr][c] = input[c];
                }

        }


}

void OPT(int input[], int n, int page_frames)
{

        table[0][0] = input[0];
        page_faults++;
        pf[0] = '*';
        int curr = 0;

        for(int i=1; i<page_frames; i++)
        {
                table[i][0] = -1;
        }


        for(int c=1; c<n; c++)
        {
                for(int r=0; r<page_frames; r++)
                {
                        table[r][c] = table[r][c-1];
                }

                bool flag = false;
                pf[c] = ' ';

                if(findInCol(input[c], c, page_frames) == -1)
                        flag = true;

                if(flag)
                {
                        if(c < page_frames)
                        {
                                for(int i=0; i<page_frames; i++)
```

```
                if(c < page_frames)
                {
                        for(int i=0; i<page_frames; i++)
                                if(table[i][c] == -1)
                                {
                                        table[i][c] = input[c];
                                        break;
                                }
                }

                else
                {
                        //tim phan tu duoc truy xuat tre nhat  nhat ben tay phai
                        int max = -1;
                        int pos = -1;
                        bool check[100];
                        for(int i=0; i<page_frames; i++)
                        {
                                check[i] = false;
                                for(int j=c+1; j<n; j++)
                                        if(table[i][c] == input[j])
                                        {       if(max < j)
                                                {
                                                        max = j;
                                                        pos = i;
                                                }
                                                check[i] = true;
                                                break;
                                        }
                        }

                        for(int i=0; i<page_frames; i++)
                                if(!check[i])
                                {
                                        pos = i;
                                        break;
                                }

                        table[pos][c] = input[c];

                }
                page_faults++;
                pf[c] = '*';

        }
```

```
                }

        }


}

void LRU(int input[], int n, int page_frames)
{


        table[0][0] = input[0];
        page_faults++;
        pf[0] = '*';


        for(int i=1; i<page_frames; i++)
        {
                table[i][0] = -1;
        }


        for(int c=1; c<n; c++)
        {
                for(int r=0; r<page_frames; r++)
                {
                        table[r][c] = table[r][c-1];
                }

                bool flag = false;
                pf[c] = ' ';

                if(findInCol(input[c], c, page_frames) == -1)
                        flag = true;


                if(flag)
                {
                        if(c < page_frames)
                        {
                                for(int i=0; i<page_frames; i++)
                                        if(table[i][c] == -1)
                                        {
                                                table[i][c] = input[c];
                                                break;
                                        }
```

```
                                        break;
                                }
                        }

                        else
                        {
                                //tim phan tu khong duoc su dung tu lau ben tay trai
                                int min = INT_MAX;
                                int pos = -1;
                                bool check[100];
                                for(int i=0; i<page_frames; i++)
                                {
                                        check[i] = false;
                                        for(int j=c-1; j>=0; j--)
                                                if(table[i][c] == input[j])
                                                {       if(min > j)
                                                        {
                                                                min = j;
                                                                pos = i;
                                                        }
                                                        check[i] = true;
                                                        break;
                                                }
                                }

                                for(int i=0; i<page_frames; i++)
                                        if(!check[i])
                                        {
                                                pos = i;
                                                break;
                                        }

                                table[pos][c] = input[c];

                        }
                        page_faults++;
                        pf[c] = '*';
                }
        }
}
```

```
int main()
{
        int choice;
        int *input;
        int n, page_frames;

        cout<<"--Page Replacement algorithm--";
        cout<<"\n1. Default referenced sequence";
        cout<<"\n2. Manual input sequence"<<endl;
        cin>>choice;

        switch(choice)
        {
                case 1:
                        n = 11;
                        input = new int[11]{2,0,5,2,2,1,0,9,0,0,7};
                        break;

                case 2:
                        cout<<"\nnhap n: ";
                        cin>>n;

                        input = new int[n];
                        cout<<"\nnhap cac trang: ";
                        for(int i=0; i<n; i++)
                                cin>>input[i];
        }


        cout<<"\n\n--Page Replacement algorithm--";

        cout<<"\nInput page frames: ";
        cin>>page_frames;

        cout<<"\n\n1.FIFO algorithm";
        cout<<"\n2.OPT algorithm";
        cout<<"\n3.LRU algorithm"<<endl;

        cin>>choice;

        switch(choice)
        {
                case 1:
                        FIFO(input, n, page_frames);
                        break;

                case 2:
```

```
        switch(choice)
        {
                case 1:
                        FIFO(input, n, page_frames);
                        break;

                case 2:
                        OPT(input, n, page_frames);
                        break;

                case 3:
                        LRU(input, n, page_frames);
                        break;

        }


        cout<<endl;
        for(int i=0; i<n; i++)
                cout<<input[i]<<' ';
        cout<<endl<<endl;

        for(int r=0; r<page_frames; r++)
        {
                for(int c=0; c<n; c++)
                        if(table[r][c] == -1)
                                cout<<' '<<' ';
                        else
                                cout<<table[r][c]<<' ';

                cout<<endl;
        }

        for(int i=0; i<n; i++)
                cout<<pf[i]<<' ';


        cout<<"\nNumber of Page Fault: "<< page_faults<<endl;
        return 1;
}
```

Giải thích

Thuật toán FIFO: Tạo một mảng 2 chiều table để lưu lại tình trạng thay thế trang nhớ, mảng pf(page_fault) để lưu lại những thời điểm bị lỗi trang. Nhập danh sách các trang tham chiếu input[n] và nhập số khung trang page_frames từ bàn phím. Cột đầu tiên của ma trận table lưu trang được tham chiếu đầu tiên vào bộ nhớ, những khung trang còn

trống thì sẽ có giá trị -1. Dùng biến curr với giá trị khởi tạo bằng 0 để lưu vị trí khung trang có trang nhớ được nạp vào lâu nhất.
Với mỗi trang được tham chiếu thì các cột trong table sẽ sao chép các bảng trong cột trước và kiểm tra xem trang đó đã được nạp vào bộ nhớ(có trong cột hiện tại của table chưa). Nếu đã có rồi thì sẽ không phát sinh lỗi trang và chuyển tới tham chiếu trang tiếp theo. Nếu chưa có thì sẽ nạp vào khung trang có vị trí curr cụ thể là table[curr][c] với c là cột hiện tại.

Thuật toán OPT: Tạo một mảng 2 chiều table để lưu lại tình trạng thay thế trang nhớ, mảng pf(page_fault) để lưu lại những thời điểm bị lỗi trang. Nhập danh sách các trang input[n] và nhập số frame từ bàn phím. Cột đầu tiên của ma trận table lưu trang được tham chiếu đầu tiên vào bộ nhớ, những khung trang còn trống thì sẽ có giá trị -1.
Với mỗi trang được tham chiếu thì các cột trong table sẽ sao chép các bảng trong cột trước và kiểm tra xem trang đó đã được nạp vào bộ nhớ(có trong cột c của table chưa). Nếu đã có rồi thì sẽ không phát sinh lỗi trang và chuyển tới tham chiếu trang tiếp theo. Nếu chưa có thì có 2 trường hợp. Trường hợp 1 là vẫn còn frame trống thì sẽ nạp trang đó vào, trường hợp 2 là hết frame trống thì thay thế trang nhớ ở vị trí khung trang có trang nhớ được truy xuất trễ nhất hoặc không được yêu cầu nữa trong tương lai. Nếu chỉ dùng biến max thì chỉ có thể tìm được frame có trang sẽ được truy xuất trễ nhất, phải dùng thêm mảng phụ bool check[] để kiểm tra trong trường hợp trang nhớ đó không còn được truy xuất trong tương lai nữa. Sau khi tìm được thì sẽ thay trang vào vị trí đó.

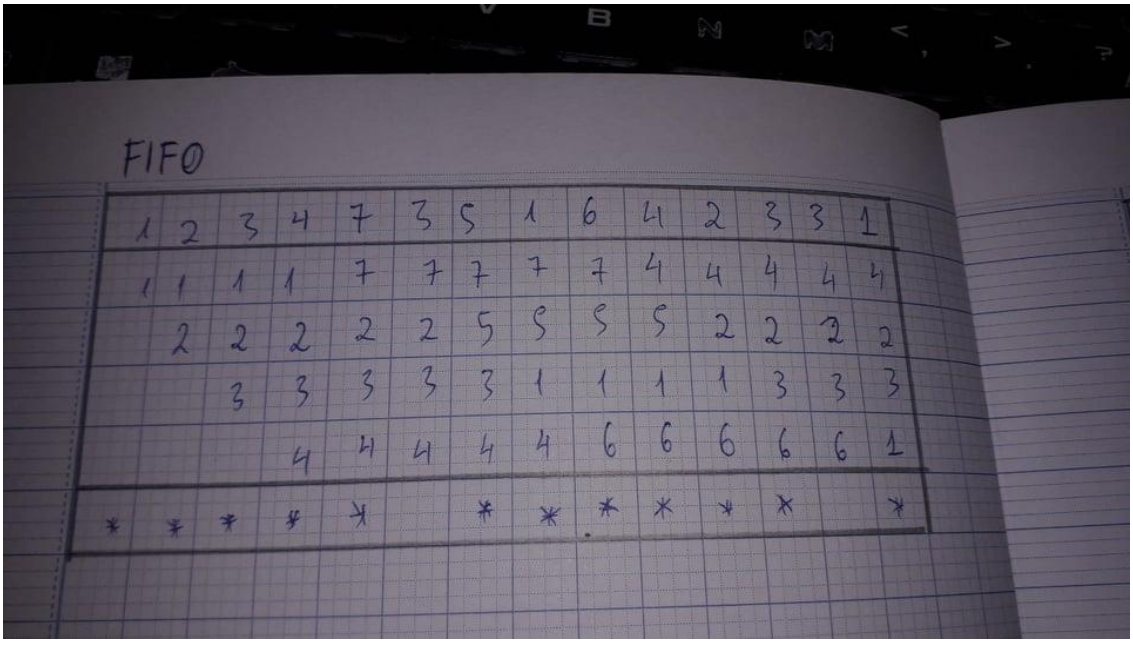
Thuật toán LRU: cũng tương tự như giải thuật OPT chỉ khác ở chỗ là trong trường hợp trang nhớ được tham chiếu không có trong bộ nhớ và đã hết frame trống thì sẽ quan tâm đến thời điểm mà trang nhớ được sử dụng để từ đó chọn trang nhớ bị thay thế. Tức là trang nhớ có thời điểm sử dụng lâu nhất (đã lâu nhưng không được sử dụng lại) sẽ là trang nhớ bị thay thế.

Test case:

FIFO:

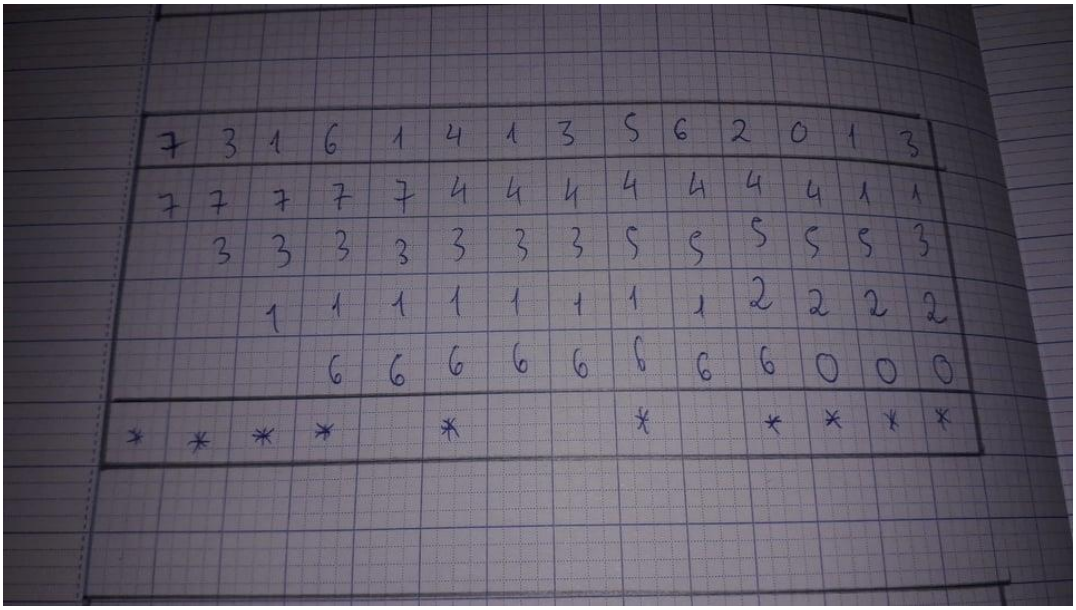
1

```
tuan_20522109@tuan-VirtualBox:~/Desktop/lab6$ g++ bai1.cpp -o bai1
tuan_20522109@tuan-VirtualBox:~/Desktop/lab6$ ./bai1
--Page Replacement algorithm--
1. Default referenced sequence
2. Manual input sequence
2

nhap n: 14

nhap cac trang: 1 2 3 4 7 3 5 1 6 4 2 3 3 1


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


1.FIFO algorithm
2.OPT algorithm
3.LRU algorithm
1

1 2 3 4 7 3 5 1 6 4 2 3 3 1

1 1 1 1 7 7 7 7 7 4 4 4 4 4
  2 2 2 2 2 5 5 5 5 2 2 2 2
    3 3 3 3 3 1 1 1 1 3 3 3
      4 4 4 4 4 6 6 6 6 6 1
* * * * *   * * * * * *   *
Number of Page Fault: 12
```



2

```
tuan_20522109@tuan-VirtualBox:~/Desktop/lab6$ ./bai1
--Page Replacement algorithm--
1. Default referenced sequence
2. Manual input sequence
2

nhap n: 14

nhap cac trang: 7 3 1 6 1 4 1 3 5 6 2 0 1 3


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


1.FIFO algorithm
2.OPT algorithm
3.LRU algorithm
1

7 3 1 6 1 4 1 3 5 6 2 0 1 3

7 7 7 7 7 4 4 4 4 4 4 4 1 1
  3 3 3 3 3 3 3 5 5 5 5 5 3
    1 1 1 1 1 1 1 1 2 2 2 2
      6 6 6 6 6 6 6 6 0 0 0
* * * *   *     *   * * * *
Number of Page Fault: 10
```



3

```
tuan_20522109@tuan-VirtualBox:~/Desktop/lab6$ ./bai1
--Page Replacement algorithm--
1. Default referenced sequence
2. Manual input sequence
2

nhap n: 14

nhap cac trang: 1 2 3 4 2 3 1 5 1 4 2 3 4 1


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


1.FIFO algorithm
2.OPT algorithm
3.LRU algorithm
1

1 2 3 4 2 3 1 5 1 4 2 3 4 1

1 1 1 4 4 4 4 4 4 4 2 2 2 1
  2 2 2 2 2 1 1 1 1 1 3 3 3
    3 3 3 3 3 5 5 5 5 5 4 4
* * * *     * *     * * * *
Number of Page Fault: 10
```

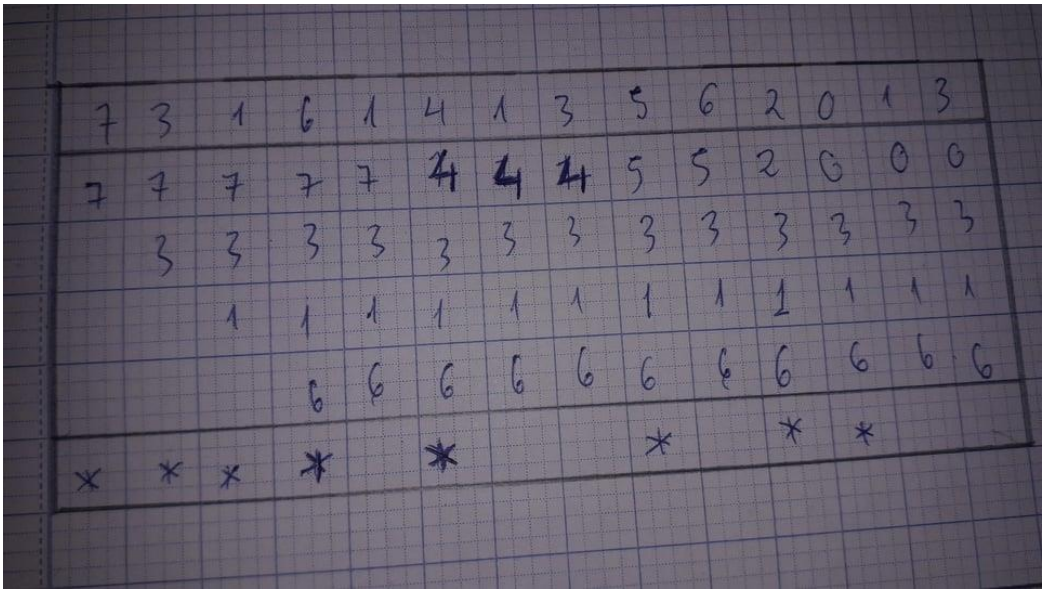
OPT:

1

```
tuan_20522109@tuan-VirtualBox:~/Desktop/lab6$ ./bai1
--Page Replacement algorithm--
1. Default referenced sequence
2. Manual input sequence
2

nhap n: 14

nhap cac trang: 1 2 3 4 7 3 5 1 6 4 2 3 3 1


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


1.FIFO algorithm
2.OPT algorithm
3.LRU algorithm
2

1 2 3 4 7 3 5 1 6 4 2 3 3 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1
  2 2 2 7 7 5 5 6 6 2 2 2 2
    3 3 3 3 3 3 3 3 3 3 3 3
      4 4 4 4 4 4 4 4 4 4 4
* * * * *   *   *   *
Number of Page Fault: 8
```

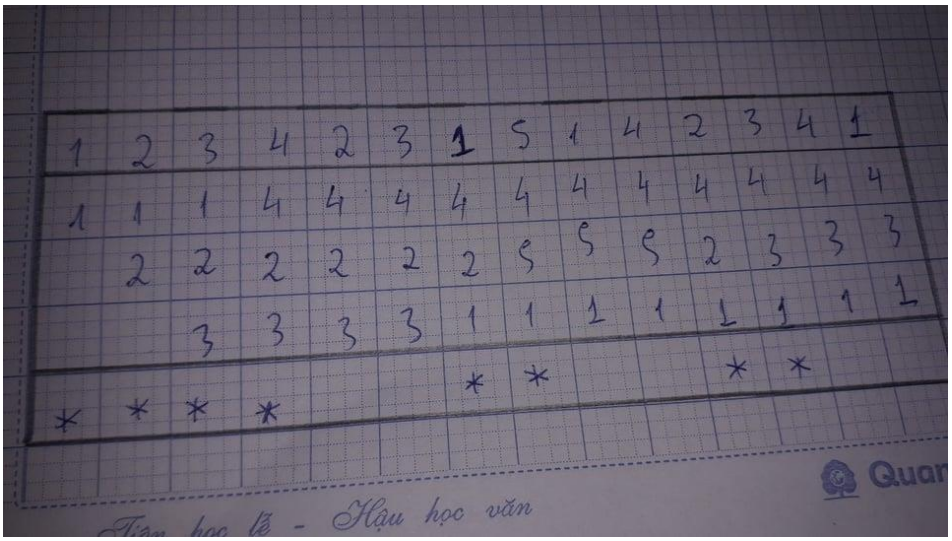
2

```
tuan_20522109@tuan-VirtualBox:~/Desktop/lab6$ ./bai1
--Page Replacement algorithm--
1. Default referenced sequence
2. Manual input sequence
2

nhap n: 14

nhap cac trang: 7 3 1 6 1 4 1 3 5 6 2 0 1 3


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


1.FIFO algorithm
2.OPT algorithm
3.LRU algorithm
2

7 3 1 6 1 4 1 3 5 6 2 0 1 3

7 7 7 7 7 4 4 4 5 5 2 0 0 0
  3 3 3 3 3 3 3 3 3 3 3 3 3
    1 1 1 1 1 1 1 1 1 1 1 1
      6 6 6 6 6 6 6 6 6 6 6
* * * *   *     *   * *
Number of Page Fault: 8
```



3

```
tuan_20522109@tuan-VirtualBox:~/Desktop/lab6$ ./bai1
--Page Replacement algorithm--
1. Default referenced sequence
2. Manual input sequence
2

nhap n: 14

nhap cac trang: 1 2 3 4 2 3 1 5 1 4 2 3 4 1


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


1.FIFO algorithm
2.OPT algorithm
3.LRU algorithm
2

1 2 3 4 2 3 1 5 1 4 2 3 4 1

1 1 1 4 4 4 4 4 4 4 4 4 4 4
  2 2 2 2 2 2 5 5 5 2 3 3 3
    3 3 3 3 1 1 1 1 1 1 1 1
* * * *     * *     * *
Number of Page Fault: 8
```



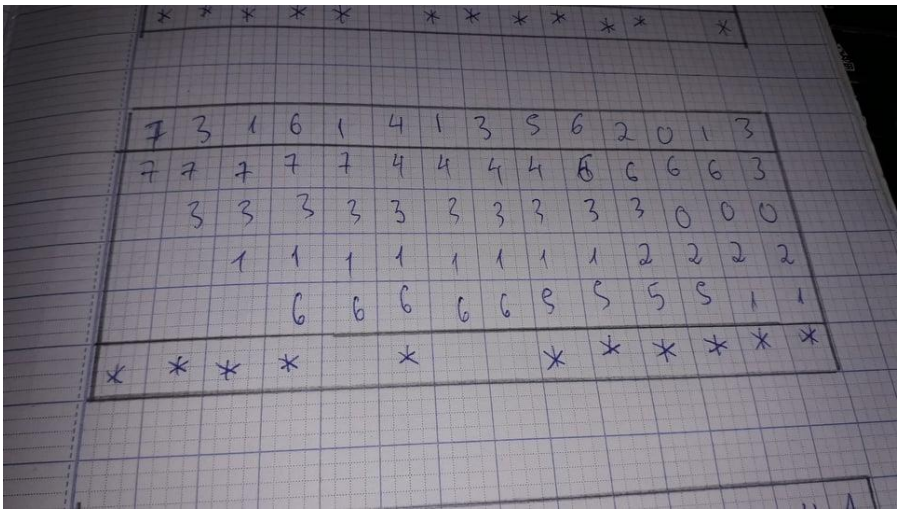
LRU:

1

```
tuan_20522109@tuan-VirtualBox:~/Desktop/lab6$ ./bai1
--Page Replacement algorithm--
1. Default referenced sequence
2. Manual input sequence
2

nhap n: 14

nhap cac trang: 1 2 3 4 7 3 5 1 6 4 2 3 3 1


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


1.FIFO algorithm
2.OPT algorithm
3.LRU algorithm
3

1 2 3 4 7 3 5 1 6 4 2 3 3 1

1 1 1 1 7 7 7 7 6 6 6 6 6 1
  2 2 2 2 2 5 5 5 5 2 2 2 2
    3 3 3 3 3 3 3 4 4 4 4 4
      4 4 4 4 1 1 1 1 3 3 3
* * * * *   * * * * * *   *
Number of Page Fault: 12
```

2

```
tuan_20522109@tuan-VirtualBox:~/Desktop/lab6$ ./bai1
--Page Replacement algorithm--
1. Default referenced sequence
2. Manual input sequence
2

nhap n: 14

nhap cac trang: 7 3 1 6 1 4 1 3 5 6 2 0 1 3


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


1.FIFO algorithm
2.OPT algorithm
3.LRU algorithm
3

7 3 1 6 1 4 1 3 5 6 2 0 1 3

7 7 7 7 7 4 4 4 4 6 6 6 6 3
  3 3 3 3 3 3 3 3 3 3 0 0 0 0
    1 1 1 1 1 1 1 1 2 2 2 2
      6 6 6 6 6 5 5 5 5 1 1
* * * *   *     * * * * * *
Number of Page Fault: 11
```



3

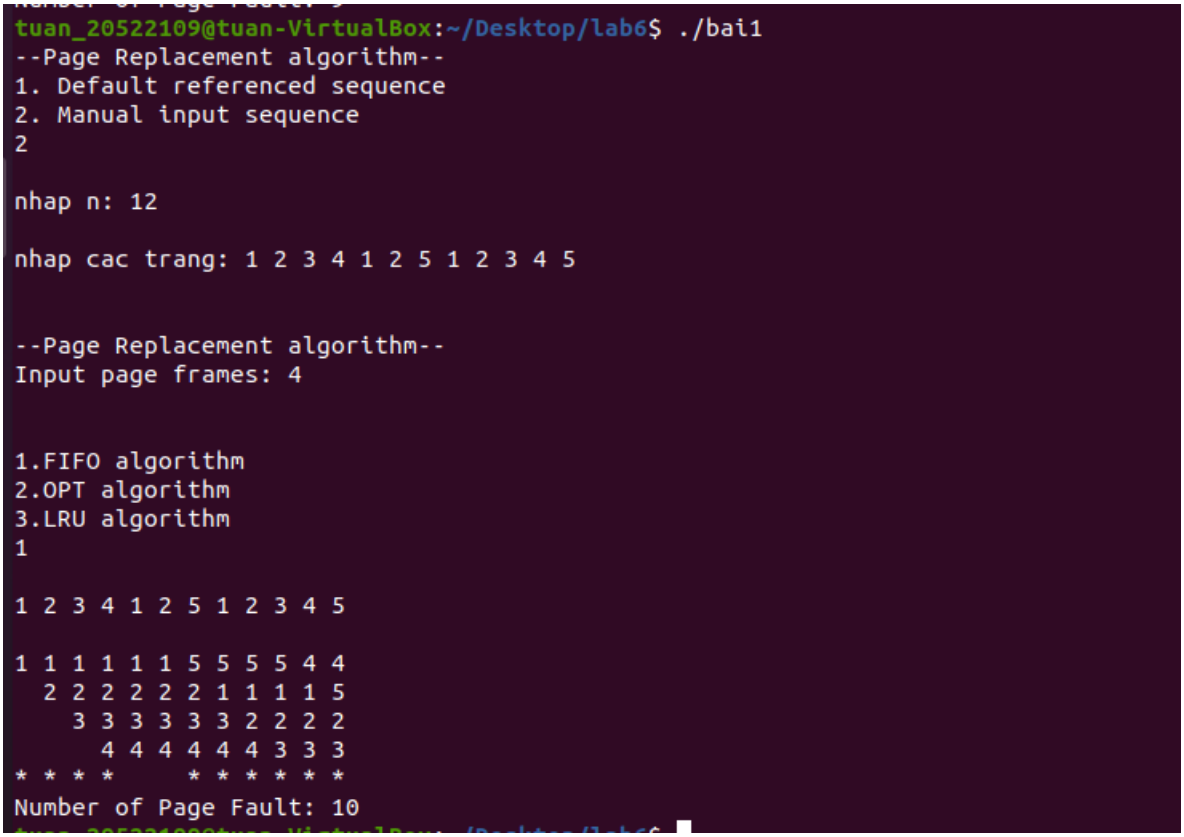
```
tuan_20522109@tuan-VirtualBox:~/Desktop/lab6$ ./bai1
--Page Replacement algorithm--
1. Default referenced sequence
2. Manual input sequence
2

nhap n: 14

nhap cac trang: 1 2 3 4 2 3 1 5 1 4 2 3 4 1


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


1.FIFO algorithm
2.OPT algorithm
3.LRU algorithm
3

1 2 3 4 2 3 1 5 1 4 2 3 4 1

1 1 1 4 4 4 1 1 1 1 1 3 3 3
  2 2 2 2 2 2 5 5 5 2 2 2 1
    3 3 3 3 3 3 3 4 4 4 4 4 4
* * * *     * *   * * *   *
Number of Page Fault: 10
```

6.5 Bài tập ôn tập

1. Nghịch lý Belady là gì? Sử dụng chương trình đã viết trên để chứng minh nghịch lý này.

Nghịch lý Belady: nghịch lý số page fault tăng mặc dù quá trình đã được cấp nhiều frame hơn.

```
tuan_20522109@tuan-VirtualBox:~/Desktop/lab6$ ./bai1
--Page Replacement algorithm--
1. Default referenced sequence
2. Manual input sequence
2

nhap n: 12

nhap cac trang: 1 2 3 4 1 2 5 1 2 3 4 5


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


1.FIFO algorithm
2.OPT algorithm
3.LRU algorithm
1

1 2 3 4 1 2 5 1 2 3 4 5

1 1 1 4 4 4 5 5 5 5 5 5
  2 2 2 1 1 1 1 1 3 3 3
    3 3 3 2 2 2 2 2 4 4
* * * * * * *     * *
Number of Page Fault: 9
```



Cùng một chuỗi tham chiếu nhưng khi cấp 3 khung trang thì có 9 lỗi và khi cấp 4 khung trang thì sẽ có 10 lỗi phát sinh.

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

❖ Giải thuật nào là bất khả thi nhất? Vì sao?

Giải thuật OPT là bất khả thi nhất vì không thể biết được thứ tự các trang sẽ được tham chiếu trong tương lai.

❖ Giải thuật nào là phức tạp nhất? Vì sao?

Giải thuật LRU phức tạp nhất do chi phí cho việc tìm kiếm là rất cao và thuật toán này đòi hỏi phải được cơ chế phần cứng hỗ trợ để xác định một thứ tự cho các trang theo thời điểm truy xuất cuối cùng. Ít CPU cung cấp đủ sự hỗ trợ phần cứng đó.