Name: Phạm Nguyễn Hải Anh

ID: 21520586

Class: IT007.ANTN.1

OPERATING SYSTEM
LAB 6'S REPORT

SUMMARY

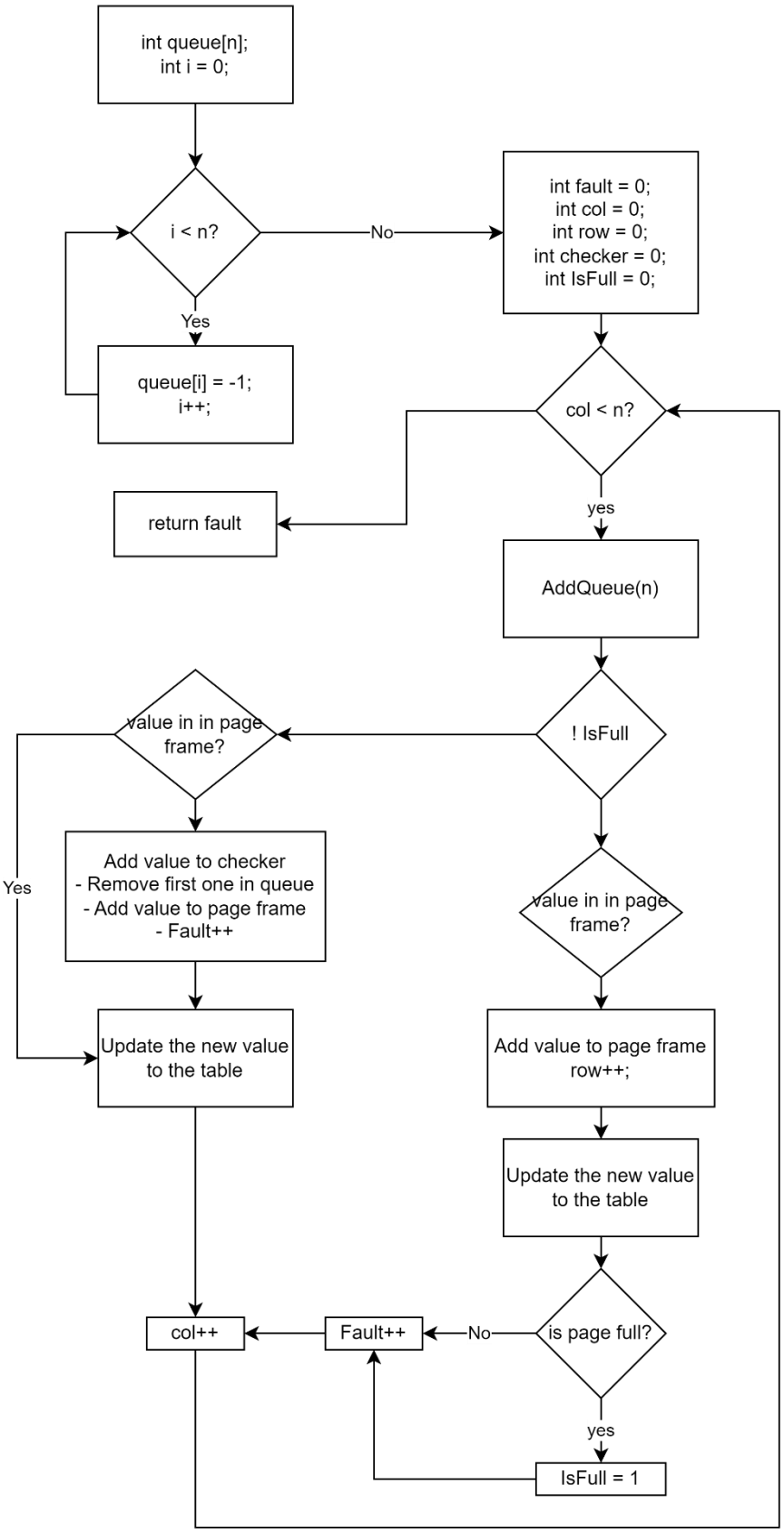
Self-scrores: 9

Note: Export file to **PDF and name the file by following format:*
Student ID_LABx.pdf

Section 6.4

1. FIFO:

Lưu đồ:


int queue[n];
int i = 0;
i < n?
No
Yes
queue[i] = -1;
i++;
int fault = 0;
int col = 0;
int row = 0;
int checker = 0;
int IsFull = 0;
col < n?
yes
return fault
AddQueue(n)
! IsFull
value in in page frame?
Yes
Add value to checker
- Remove first one in queue
- Add value to page frame
- Fault++
Update the new value to the table
value in in page frame?
Add value to page frame
row++;
Update the new value to the table
is page full?
No
Fault++
col++
yes
IsFull = 1

Chạy tay 1 test case ít nhất 10 phần tử:

Chuỗi truy xuất bộ nhớ: 1 2 4 6 8 4 3 6 4 4 3 2 5 6 7 3 2 5 6 7 1 2 3 4 5 với 4 khung trang vật lý: page fault = 20

1	2	4	6	8	4	3	6	4	4	3	2	5	6	7	3	2	5	6	7	1	2	3	4	5
1	1	1	1	8	8	8	8	8	8	8	8	8	6	6	6	6	5	5	5	5	2	2	2	2
	2	2	2	2	2	3	3	3	3	3	3	3	3	7	7	7	7	6	6	6	6	3	3	3
		4	4	4	4	4	4	4	4	4	2	2	2	2	3	3	3	3	7	7	7	7	4	4
			6	6	6	6	6	6	6	6	6	5	5	5	5	2	2	2	2	1	1	1	1	5
*	*	*	*	*		*					*	*	*	*	*	*	*	*	*	*	*	*	*	*

Code cho giải thuật:

```
int FIFO(int *input, int **result, int n, int page)
{
    int *queue = (int *)malloc(n * sizeof(int));
    int i = 0;
    for (; i < n; ){
        queue[i] = -1;
        i++;
    }

    int fault = 0;
    int row = 0;
    int checker = 0;
    int IsFull = 0;
    for (int col = 0; col < n; )
    {
        AddQueue(queue, input[col], n);
        if (!IsFull)
        {
            if (CheckPage(result, page, col, input[col]))
            {
                result[row][col] = input[col];
                row++;
            }
            Copy(result, col, row, n);
            if (row ==page)
                IsFull = 1;
            fault++;
            result[page][col] = -2;
            }
        else
        {
            if (CheckPage(result, page, col, input[col]))
            {
                checker = Find(result, queue[0], page, col);
                DeleteEle(queue, n);
                result[checker][col] = input[col];
                fault++;
                result[page][col] = -2;
            }
            Copy(result, col, row, n);
        }
        col++;
    }
    return fault;
}
```

Chạy test case mặc định:

```
pngha@pngha20:~$ ./Lab06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
1
--- Page Replacement algorithm ---
Input page frames: 3
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
1
--- Page Replacement algorithm---
2 1 5 2 0 5 8 6 0 0 7
2 2 2 2 0 0 0 0 0 0 7
  1 1 1 1 1 8 8 8 8 8
    5 5 5 5 5 6 6 6 6
* * *   *   * *     *
Number of Page Fault: 7
```

Chạy test case nhập tay 1: n = 10, page frame = 3 , input = 1 3 4 3 2 5 1 3 1 2:

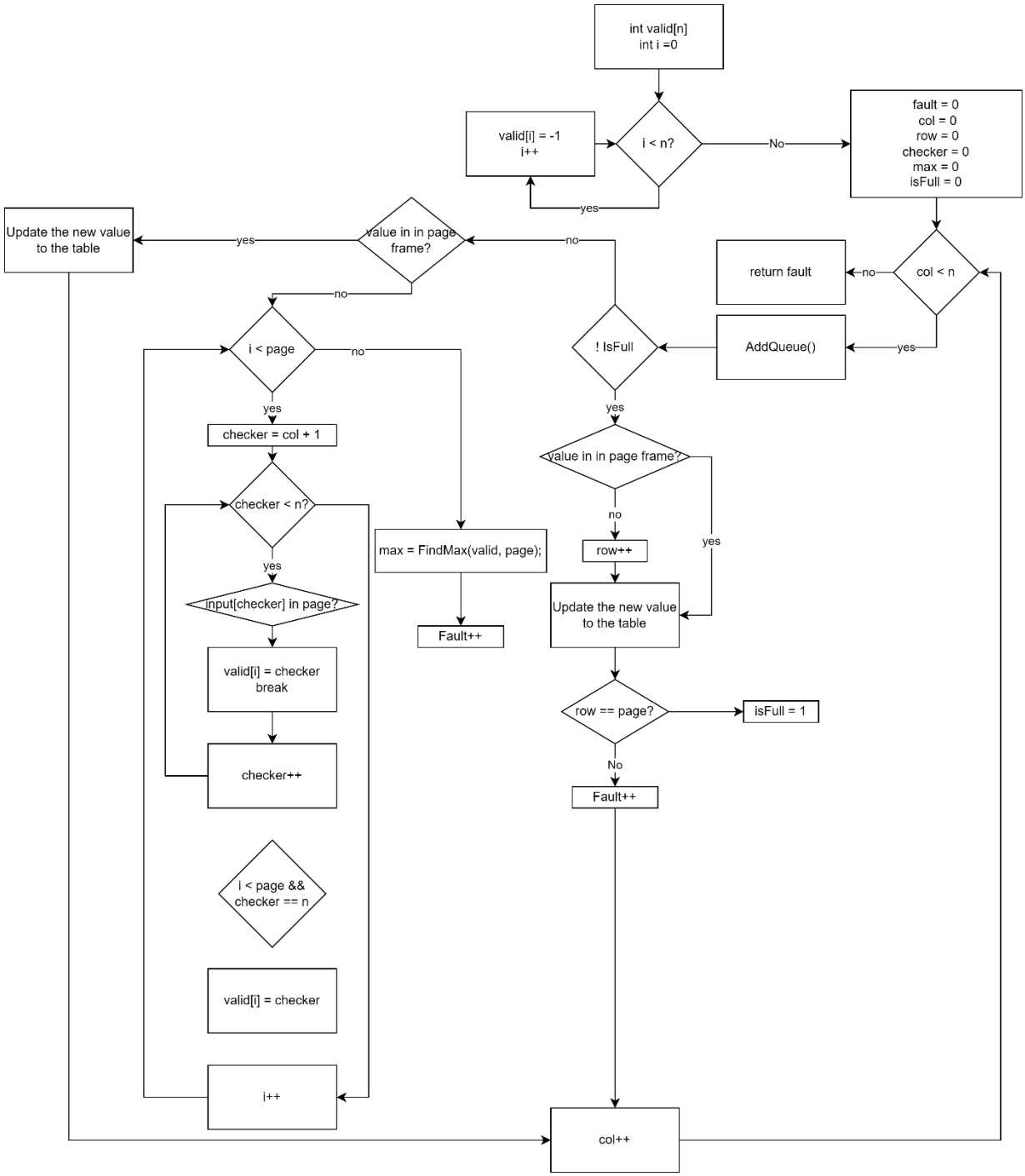
```
pngha@pngha20:~$ ./Lab06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhap so luong: 10
Nhap danh sach trang: 1 3 4 3 2 5 1 3 1 2
--- Page Replacement algorithm ---
Input page frames: 3
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
1
--- Page Replacement algorithm---
1 3 4 3 2 5 1 3 1 2
1 1 1 1 2 2 2 3 3 3
  3 3 3 3 5 5 5 5 2
    4 4 4 4 1 1 1 1
* * *   * * * *   *
Number of Page Fault: 8
```

Chạy test case nhập tay: n = 10 , page frame = 4, input = 5 3 1 2 3 1 0 4 2 1:

```
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhap so luong: 10
Nhap danh sach trang: 5 3 1 2 3 1 0 4 2 1
--- Page Replacement algorithm ---
Input page frames: 4
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
1
--- Page Replacement algorithm---
5 3 1 2 3 1 0 4 2 1
5 5 5 5 5 5 0 0 0 0
  3 3 3 3 3 3 4 4 4
    1 1 1 1 1 1 1 1
      2 2 2 2 2 2 2
* * * *     * *
Number of Page Fault: 6
```

2. OPT:

Lưu đồ:



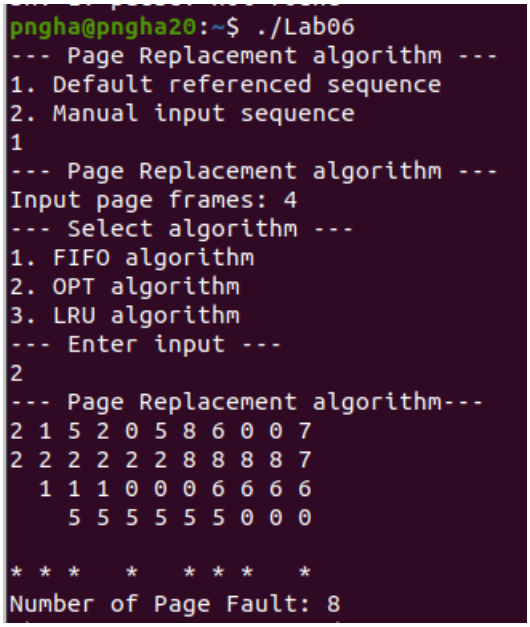
Chuỗi truy xuất bộ nhớ: 1 2 4 6 8 4 3 6 4 4 3 2 5 6 7 3 2 5 6 7 1 2 3 4 5 với 4 khung trang vật lý: page fault = 12

1	2	4	6	8	4	3	6	4	4	3	2	5	6	7	3	2	5	6	7	1	2	3	4	5
1	1	1	1	8	8	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	4	5
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		4	4	4	4	4	4	4	4	4	4	5	5	5	5	5	5	6	6	1	1	1	1	1
			6	6	6	6	6	6	6	6	6	6	6	7	7	7	7	7	7	7	7	7	7	7
F	F	F	F	F		F						F		F				F		F			F	F

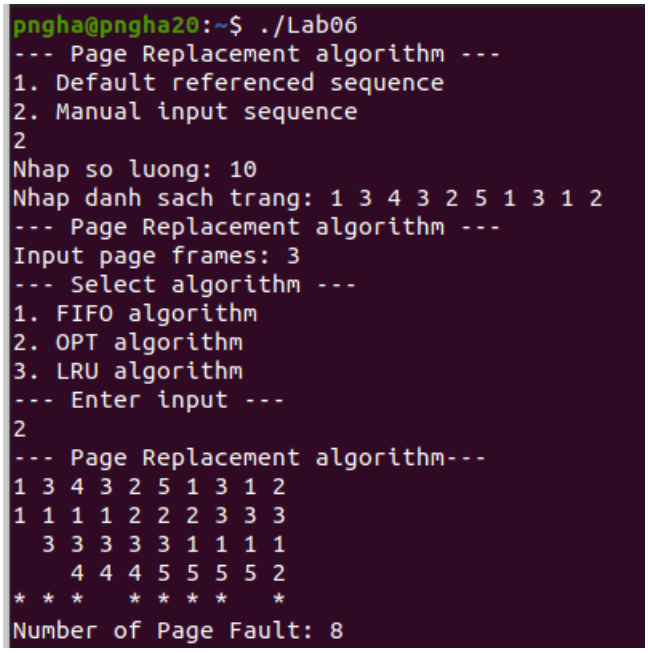
Code:

```
int OPT(int *input, int **result, int n, int page)
{
    int *valid = (int *)malloc(page * sizeof(int));
    int i;
    for (i = 0; i < n; i++)
    valid[i] = -1;
    int fault = 0;
    int col = 0;
    int row = 0;
    int checker = 0;
    int max = 0;
    int IsFull = 0;
    for (; col < n; col++)
    {
        if (!IsFull)
        {
            if (CheckPage(result, page, col, input[col]))
            {
                result[row][col] = input[col];
                row++;
            }
            Copy(result, col, row, n);
            if (row == page)
                IsFull = 1;
            fault++;
            result[page][col] = -2;
            }
        else
        {
            if (CheckPage(result, page, col, input[col]))
            {
                for (i = 0; i < page; i++)
                {
                    for (checker = col + 1; checker < n; checker++)
                    {
                        if (input[checker] == result[i][col])
                        {
                            valid[i] = checker;
                            break;
                        }
                    }
                    if (i < page && checker == n)
                    valid[i] = checker;
                }
                max = FindMax(valid, page);
                result[max][col] = input[col];
                fault++;
                result[page][col] = -2;
            }
            Copy(result, col, row, n);
        }
    }
    return fault;
}
```

Chạy test case mặc định:



Chạy test case nhập tay 1: n = 10, page frame = 3 , input = 1 3 4 3 2 5 1 3 1 2:

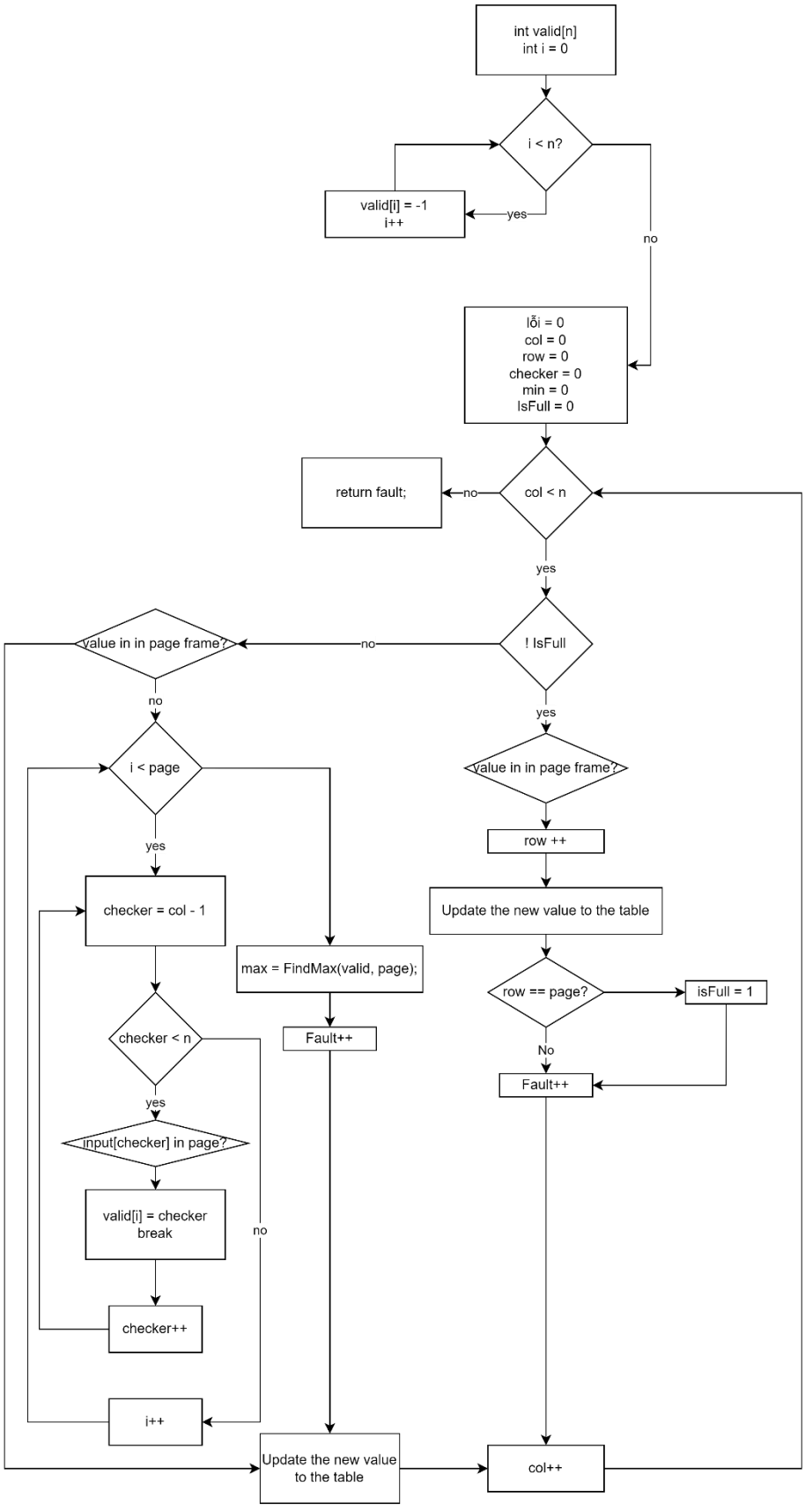


Chạy test case nhập tay: n = 10 , page frame = 4, input = 5 3 1 2 3 1 0 4 2 1:

```
pngha@pngha20:~$ ./Lab06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhap so luong: 10
Nhap danh sach trang: 5 3 1 2 3 1 0 4 2 1
--- Page Replacement algorithm ---
Input page frames: 4
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
2
--- Page Replacement algorithm---
5 3 1 2 3 1 0 4 2 1
5 5 5 5 5 5 0 0 0 0
  3 3 3 3 3 3 3 2 2
    1 1 1 1 1 1 1 1
      2 2 2 2 4 4 4
* * * *     * * *
Number of Page Fault: 7
```

3. LRU

Lưu đồ:


int valid[n]
int i = 0
i < n?
yes
valid[i] = -1
i++
no
lǒi = 0
col = 0
row = 0
checker = 0
min = 0
IsFull = 0
col < n
no
return fault;
yes
! IsFull
no
value in in page frame?
yes
value in in page frame?
no
i < page
yes
checker = col - 1
checker < n
yes
input[checker] in page?
valid[i] = checker
break
checker++
no
i++
max = FindMax(valid, page);
Fault++
row ++
Update the new value to the table
row == page?
isFull = 1
No
Fault++
Update the new value
to the table
col++

Test case chạy tay: n= 25, 4 khung, chuỗi: 1 2 4 6 8 4 3 6 4 4 3 2 5 6 7 3 2 5 6 7 1 2 3 4 5 page fault = 20

1	2	4	6	8	4	3	6	4	4	3	2	5	6	7	3	2	5	6	7	1	2	3	4	5
1	1	1	1	8	8	8	8	8	8	8	2	2	2	2	3	3	3	3	7	7	7	7	4	4
	2	2	2	2	2	3	3	3	3	3	3	3	3	7	7	7	7	6	6	6	6	3	3	3
		4	4	4	4	4	4	4	4	4	4	4	6	6	6	6	5	5	5	5	2	2	2	2
			6	6	6	6	6	6	6	6	6	5	5	5	5	2	2	2	2	1	1	1	1	5
F	F	F	F	F		F					F	F	F	F	F	F	F	F	F	F	F	F	F	F

Code:

```
int LRU(int *input, int **result, int n, int page)
{
    int *valid = (int *)malloc(page * sizeof(int));
    int i;
    for (i = 0; i < n; i++)
        valid[i] = -1;
        int fault = 0;
        int col = 0;
        int row = 0;
        int checker = 0;
        int min = 0;
        int IsFull = 0;
        for (; col < n; col++)
        {
            if (!IsFull)
            {
                if (CheckPage(result, page, col, input[col]))
                {
                    result[row][col] = input[col];
                    row++;
                }
                Copy(result, col, row, n);
                if (row == page)
                    IsFull = 1;
                fault++;
                result[page][col] = -2;
            }
            else
            {
                if (CheckPage(result, page, col, input[col]))
                {
                    for (i = 0; i < page; i++)
                    {
                        for (checker = col - 1; checker > 0; checker--)
                        {
                            if (input[checker] == result[i][col])
                            {
                                valid[i] = checker;
                                break;
                            }
                        }
                    }
                    min = FindMin(valid, page);
                    result[min][col] = input[col];
                    fault++;
                    result[page][col] = -2;
                }
                Copy(result, col, row, n);
            }
        }
    return fault;
}
```

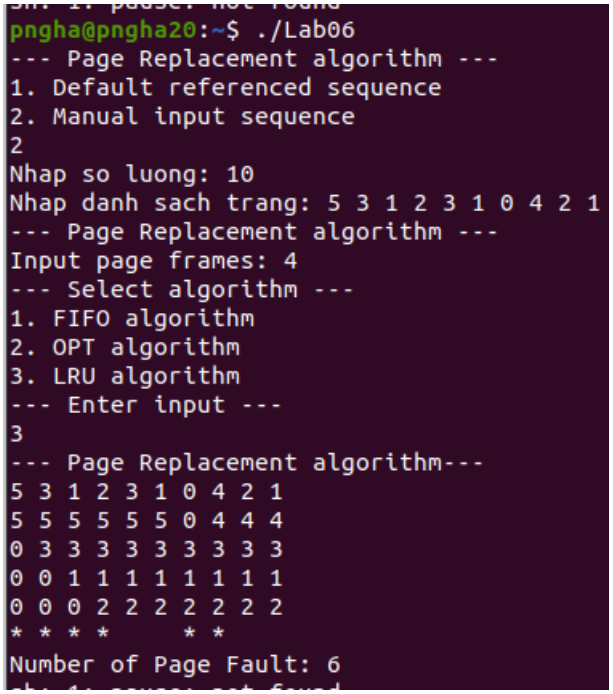
Test case mặc định:

```
pngha@pngha20:~$ ./Lab06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
1
--- Page Replacement algorithm ---
Input page frames: 3
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
3
--- Page Replacement algorithm---
2 1 5 2 0 5 8 6 0 0 7
2 2 2 2 0 0 0 0 0 0 7
0 1 1 1 1 1 8 6 6 6 6
0 0 5 5 5 5 5 5 5 5 5
* * *   *   * *     *
Number of Page Fault: 7
```

Test case nhập tay 1: n = 10, page frame = 3 , input = 1 3 4 3 2 5 1 3 1 2:

```
pngha@pngha20:~$ ./Lab06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhap so luong: 10
Nhap danh sach trang: 1 3 4 3 2 5 1 3 1 2
--- Page Replacement algorithm ---
Input page frames: 3
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
3
--- Page Replacement algorithm---
1 3 4 3 2 5 1 3 1 2
1 1 1 1 1 1 1 1 1 2
0 3 3 3 3 3 3 3 3 3
0 0 4 4 2 5 5 5 5 5
* * *   * *       *
Number of Page Fault: 6
```

Test case nhập tay 2: n=10, page frame = 4, input = 5 3 1 2 3 1 0 4 2 1:



Section 6.5:

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

Khái niệm:

frame được cấp phát thêm nhưng lỗi trang lại tăng thay vì giảm. Thường xảy ra ở các thuật toán thay trang không tối ưu, tức thường xuyên có tỉ lệ trang lỗi cao, như FIFO.

Chứng minh:

Khi số frame là 3, chuỗi là 12 thì có 9 lỗi trang:

```
pngha@pngha20:~$ ./Lab06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhap so luong: 12
Nhap danh sach trang: 1 2 3 4 1 2 5 1 2 3 4 5
--- Page Replacement algorithm ---
Input page frames: 3
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
1
--- Page Replacement algorithm---
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
```

Nhưng khi tăng frame lên 4, thì số lỗi trang lại tăng thành 10:

```
pngha@pngha20:~$ ./Lab06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhap so luong: 12
Nhap danh sach trang: 1 2 3 4 1 2 5 1 2 3 4 5
--- Page Replacement algorithm ---
Input page frames: 4
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
1
--- Page Replacement algorithm---
1 2 3 4 1 2 5 1 2 3 4 5
1 1 1 1 1 1 5 5 5 5 4 4
  2 2 2 2 2 2 1 1 1 1 5
    3 3 3 3 3 3 2 2 2 2
      4 4 4 4 4 4 3 3 3
* * * *     * * * * * *
Number of Page Fault: 10
```

5. 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 nào là phức tạp nhất? Vì sao?

Giải thuật FIFO: hiệu quả kém nhưng dễ hiện thực,

Giải thuật LRU: khó cài đặt, phức tạp, hiệu quả

Giải thuật OPT: không khả thi, nhưng hiệu quả nhất, vì không thể xác định thứ tự các trang trong tương lai.

Giải thuật OPT và LRU là phức tạp nhất bởi vì mỗi lần lỗi trang, khi tìm khung trang thích hợp để thay thế thì phải xét đến toàn bộ chuỗi tham chiếu trước / sau nó