ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

HỆ ĐIỀU HÀNH
LỚP: IT007.O18
THỰC HÀNH LAB 6

Tên: Lê Minh Nhựt

MSSV: 22521060

6.3.3. Câu hỏi chuẩn bị

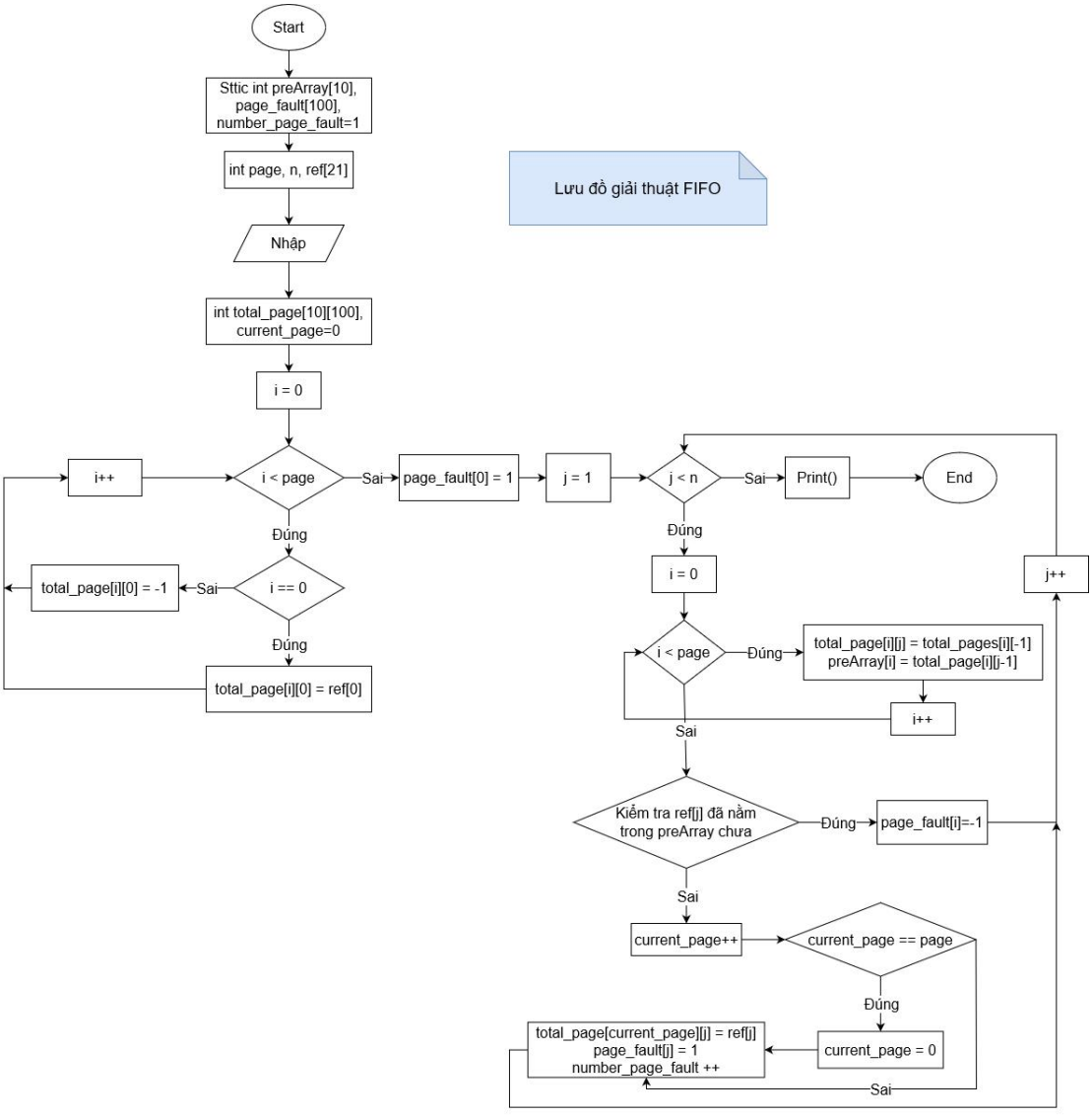
3. Vẽ lưu đồ thuật toán mô tả cách thức xử lý của 3 giải thuật thay thế trang: FIFO, OPT, LRU? Vẽ sơ đồ trình bày thay thế trang bằng 3 giải thuật trên với chuỗi tham chiếu:

0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Giải sử có 3 khung trang và các khung trang ban đầu là rỗng. Xác định số Page Fault từ đó đưa ra đánh giá các giải thuật.

* Giải thuật FIFO:

- **Lưu đồ thuật toán**



Hình 1: Lưu đồ thuật toán FIFO

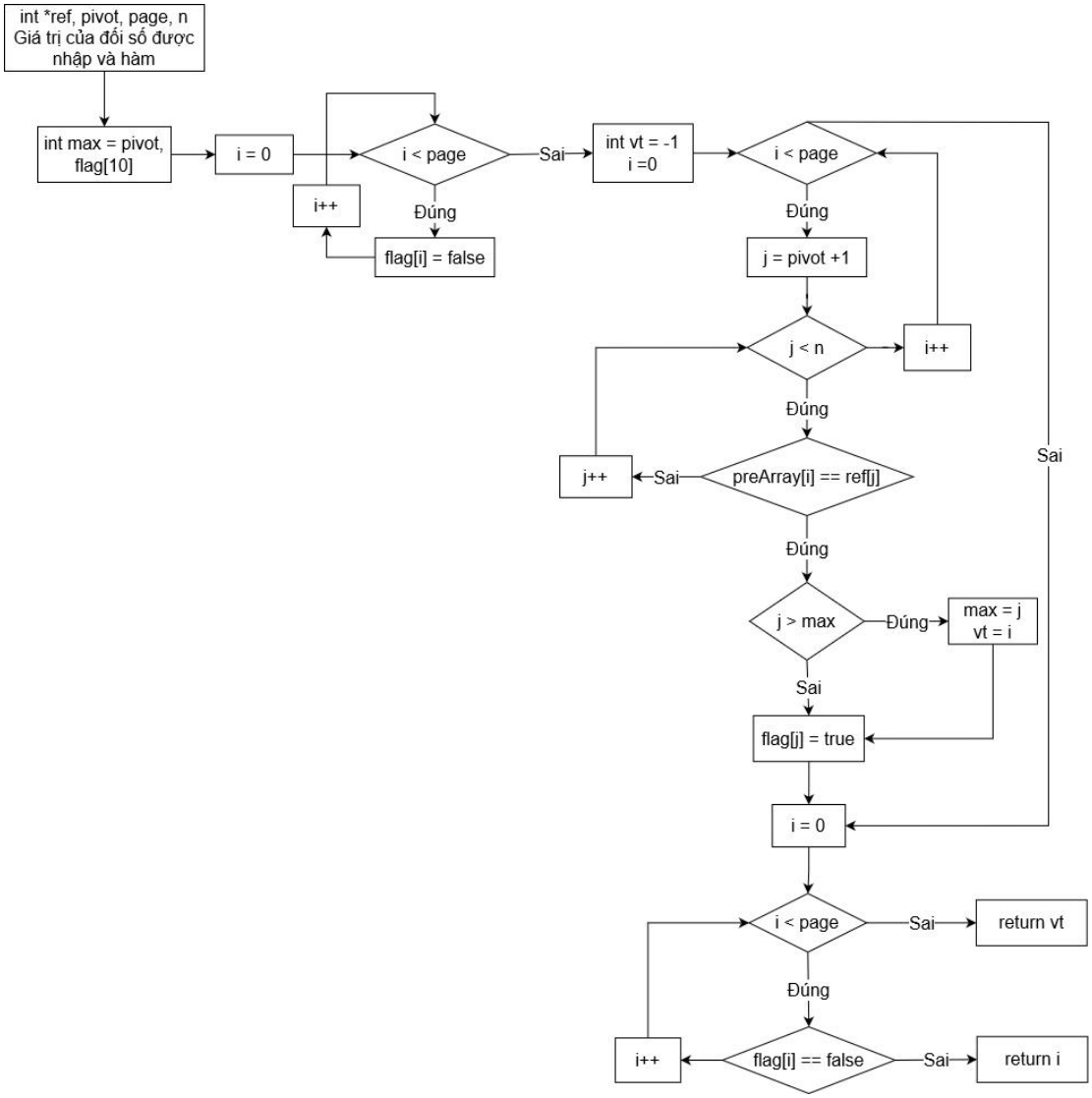
➢ **Lời giải:**

0	2	1	6	4	0	1	0	3	1	2	1
0	0	0	6	6	6	1	1	1	1	1	1
	2	2	2	4	4	4	4	3	3	3	3
		1	1	1	0	0	0	0	0	2	2
*	*	*	*	*	*	*		*		*	

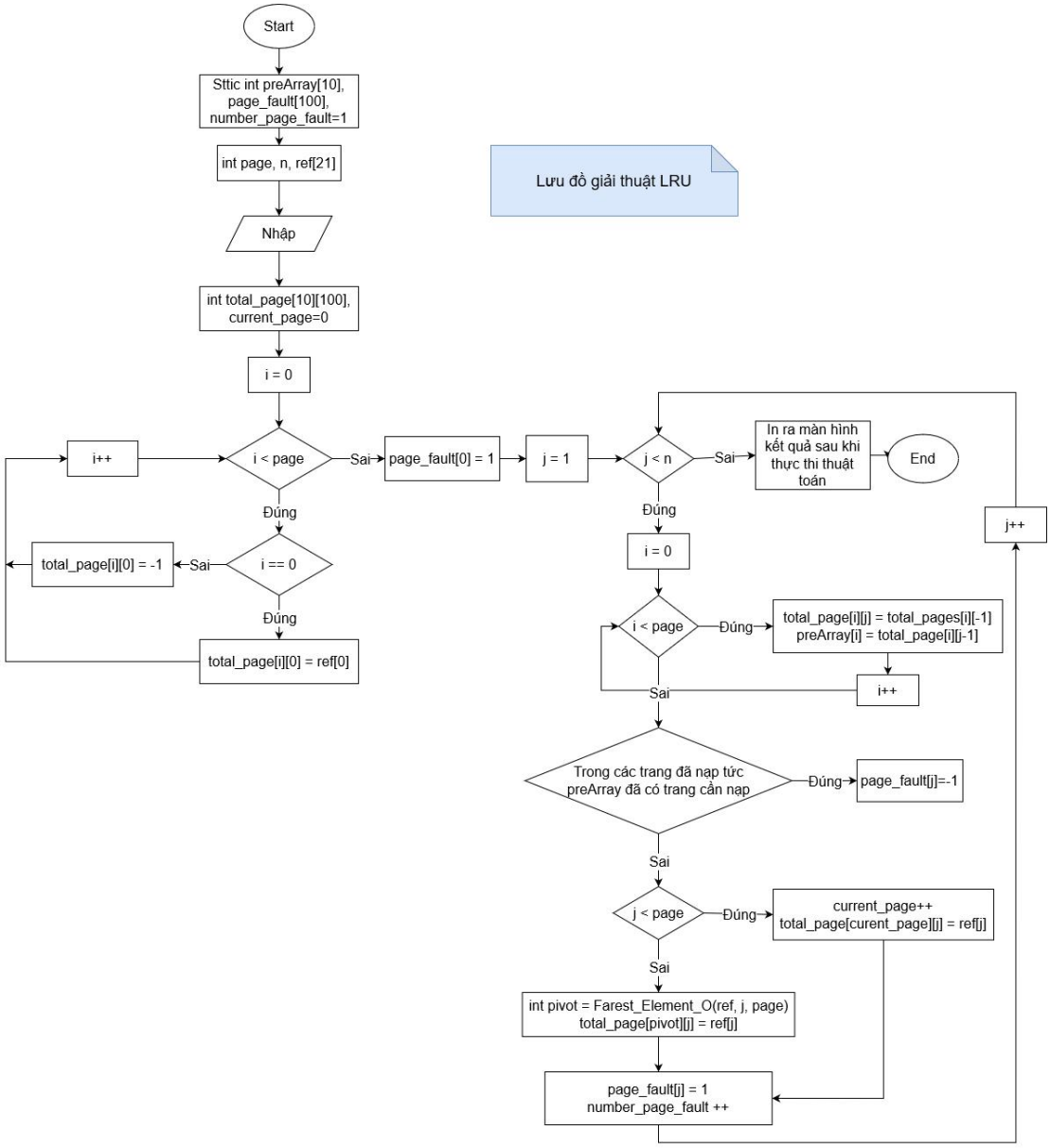
=> Tổng cộng có 9 lỗi trang.

* Giải thuật LRU

➢ **Lưu đồ thuật toán**



Hình 2: Lưu đồ hàm tìm ra trang được gói sớm nhất trong quá khứ



Hình 3: Lưu đồ thuật toán LRU.

- **Lời giải:**

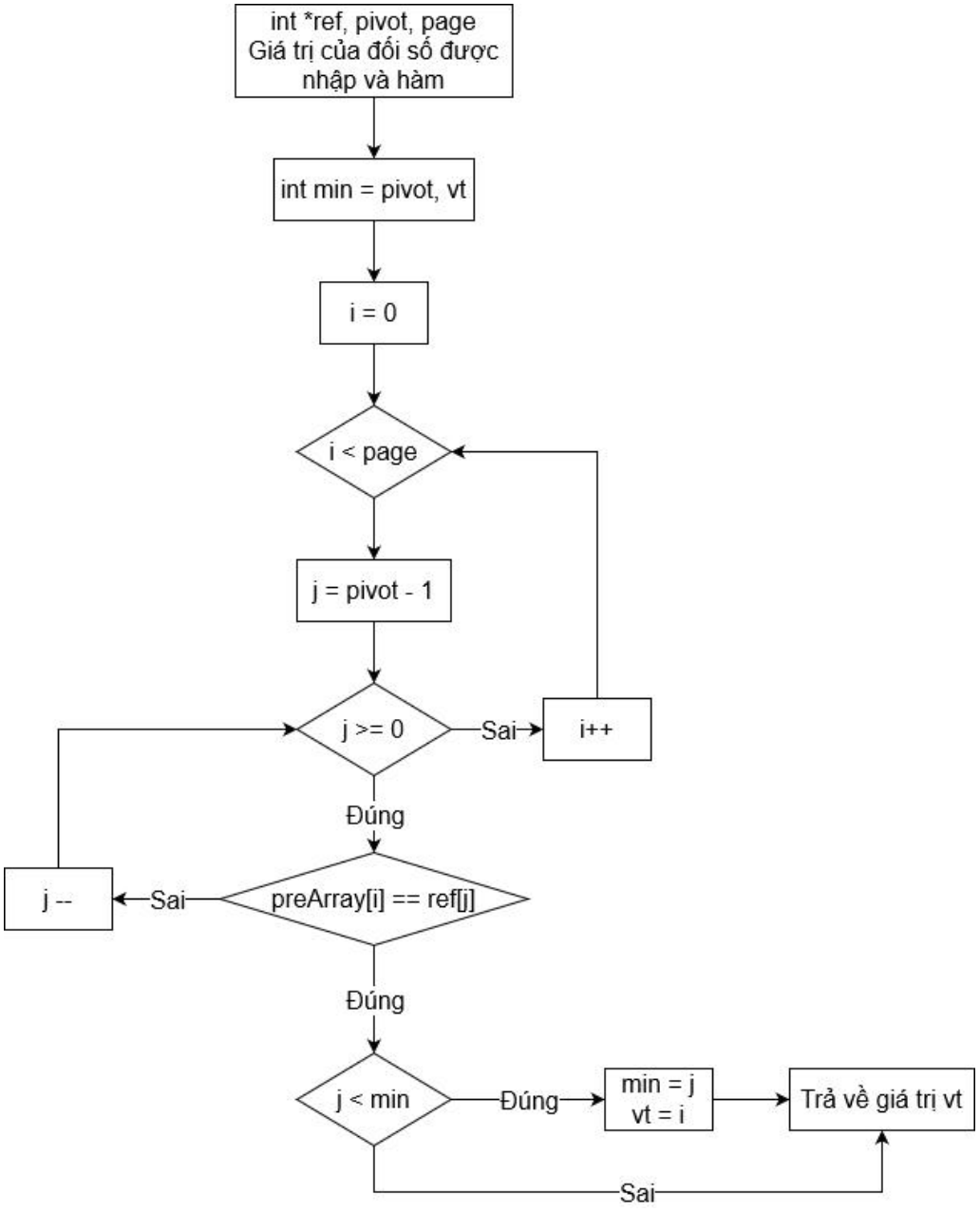
0	2	1	6	4	0	1	0	3	1	2	1
0	0	0	6	6	6	1	1	1	1	1	1
	2	2	2	4	4	4	4	3	3	3	3

		1	1	1	0	0	0	0	0	2	2
*	*	*	*	*	*	*		*		*	

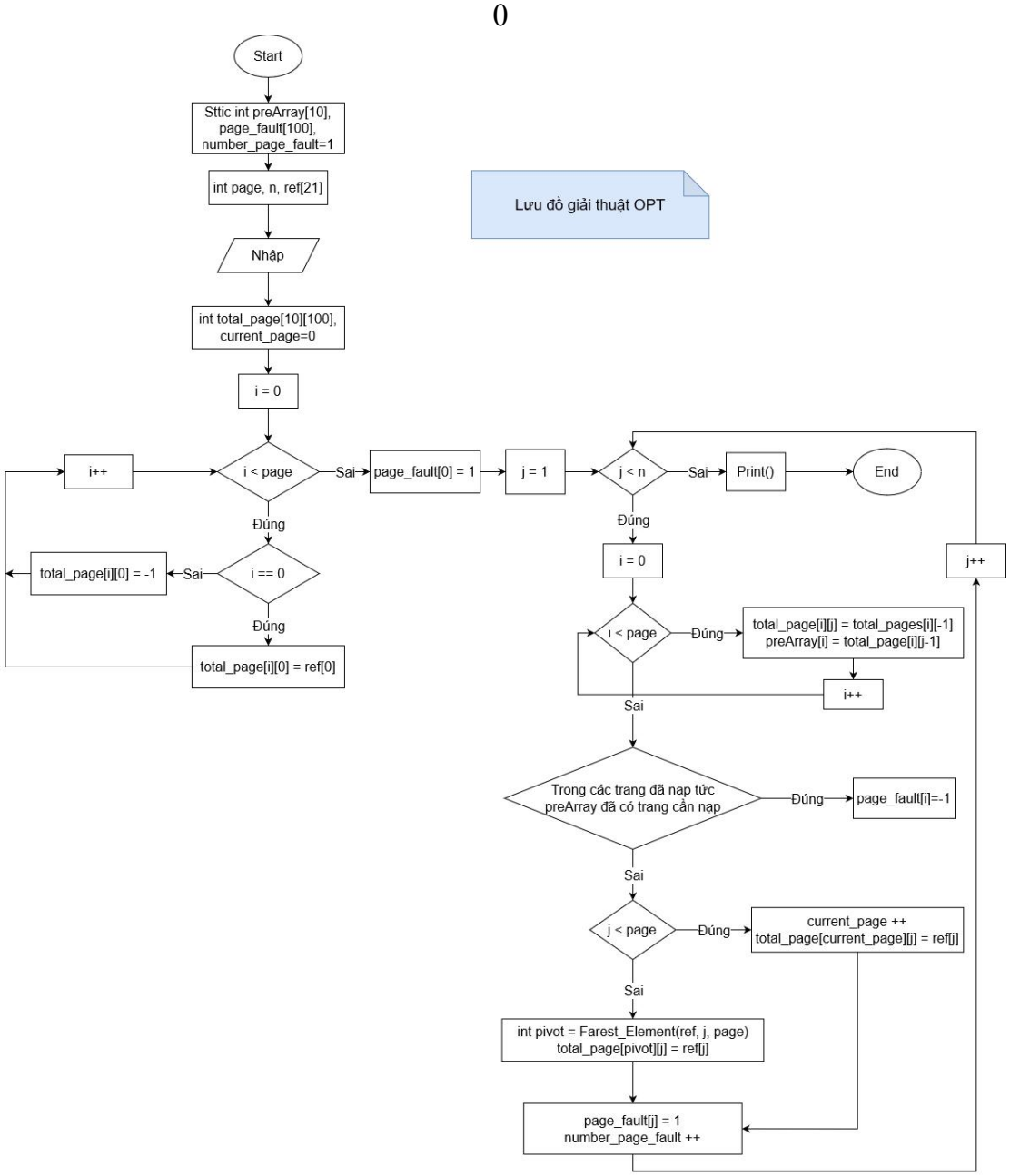
=> Tổng cộng có 9 lỗi trang.

* Giải thuật OPT

- **Lưu đồ thuật toán**



Hình 4: Lưu đồ hàm tìm ra vị trí cần được thay thế.



Hình 5: Lưu đồ thuật toán OPT.

- **Lời giải:**

0	2	1	6	4	0	1	0	3	1	2	1

0	0	0	0	0	0	0	0	3	3	2	2
	2	2	6	4	4	4	4	4	4	4	4
		1	1	1	1	1	1	1	1	1	1
*	*	*	*	*				*		*	

=> Tổng cộng có 7 lỗi trang.

- **Nhận xét chung:** giải thuật FIFO dễ thực hiện nhưng độ hiệu quả không cao, giải thuật LRU phức tạp hơn và sẽ hiệu quả hơn trong nhiều trường hợp. Trong khi đó, OPT cho lỗi ít nhất nhưng bất khả thi trong thực tế.

6.4. Hướng dẫn thực hành.

- **Soucre Code Của Chương Trình**

```
#include <iostream>
using namespace std;
static int preArray[10];
static int page_fault[100];
static int number_page_fault = 1;

int Is_in_preArray(int page, int value) {
    for (int i = 0; i < page; i++) {
        if (value == preArray[i]) return i;
    }
    return -1;
}

int Farest_Element(int *ref, int pivot, int page) {
    int min = pivot;
    int vt;
    for (int i = 0; i < page; i++) {
        for (int j = pivot - 1; j >= 0; j--) {
```

```
            if (preArray[i] == ref[j]) {
                if (j < min) {
                    min = j;
                    vt = i;
                }

                break;
            }

        }
    }
    return vt;
}

int Farest_Element_Oppsite(int *ref, int pivot, int page, int n)
{
    int max = pivot;
    int flag[10];
    for (int i = 0; i < page; i++) {
        flag[i] = false;
    }
    int vt = -1;
    for (int i = 0; i < page; i++) {
        for (int j = pivot + 1; j < n; j++) {
            if (preArray[i] == ref[j]) {
                if (j > max) {
                    max = j;
                    vt = i;
```

```
                }
                flag[i] = true;
                break;
            }
        }
    }

    for (int i = 0; i < page; i++) {
        if (flag[i] == false) return i;
    }
    return vt;
}

void Print(int total_page[10][100], int n, int page, int ref[100])
{
    // Print
    for (int i = 0; i < n; i++) {
        cout << ref[i] << " ";
    }
    cout << endl;
    for (int i = 0; i < page; i++) {
        for (int j = 0; j < n; j++) {
            if (total_page[i][j] != -1) {
                cout << total_page[i][j] << " ";
            }
            else {
                cout << "   ";
            }
```

```
        }
        cout << endl;
    }
    for (int i = 0; i < n; i++) {
        if (page_fault[i] == 1) cout << "* ";
        else {
            cout << "  ";
        }
    }
    cout << endl;
    cout << "Number of Page Fault: " << number_page_fault << endl;
}

void FIFO(int ref[], int n, int page)
{
    bool IsFault;
    int total_page[10][100];
    int current_page = 0;
    for (int i = 0; i < page; i++) {
        if (i == 0) { total_page[i][0] = ref[0]; }
        else {
            total_page[i][0] = -1;
        }
    }
    page_fault[0] = 1;
    for (int j = 1; j < n; j++) {
        for (int i = 0; i < page; i++) {
```

```
            total_page[i][j] = total_page[i][j-1];
            preArray[i] = total_page[i][j - 1];
        }
        if (Is_in_preArray(page, ref[j]) != -1) {
            page_fault[j] = -1;
        }
        else {
            current_page++;
            if (current_page == page) current_page = 0;
            total_page[current_page][j] = ref[j];
            page_fault[j] = 1;
            number_page_fault++;

        }
    }
    Print(total_page, n, page, ref);
}

void LRU(int ref[], int n, int page)
{
    bool IsFault;
    int total_page[10][100];
    int current_page = 0;
    for (int i = 0; i < page; i++) {
        if (i == 0) { total_page[i][0] = ref[0]; }
        else {
            total_page[i][0] = -1;
        }
```

```
    }
    page_fault[0] = 1;
    for (int j = 1; j < n; j++)
    {
        for (int i = 0; i < page; i++) {
            total_page[i][j] = total_page[i][j - 1];
            preArray[i] = total_page[i][j - 1];
        }
        ///////
        if (Is_in_preArray(page, ref[j]) != -1) {
            page_fault[j] = 0;
        }
        else {
            if (j < page) {
                current_page++;
                total_page[current_page][j] = ref[j];
            }
            else {

                int pivot = Farest_Element(ref, j, page);
                total_page[pivot][j] = ref[j];
            }
            page_fault[j] = 1;
            number_page_fault++;
        }
        /////
    }
    Print(total_page, n, page, ref);
```

```
}

void OPT(int ref[], int n, int page)
{
    bool IsFault;
    int total_page[10][100];
    int current_page = 0;
    for (int i = 0; i < page; i++) {
        if (i == 0) { total_page[i][0] = ref[0]; }
        else {
            total_page[i][0] = 0;
        }
    }
    page_fault[0] = 1;
    for (int j = 1; j < n; j++)
    {
        for (int i = 0; i < page; i++) {
            total_page[i][j] = total_page[i][j - 1];
            preArray[i] = total_page[i][j - 1];
        }
        ///////
        if (Is_in_preArray(page, ref[j]) != -1) {
            page_fault[j] = 0;
        }
        else {
            if (j < page) {
                current_page++;
                total_page[current_page][j] = ref[j];
```

```
            }
            else {
                int pivot = Farest_Element_Oppsite(ref, j, page,
n);
                total_page[pivot][j] = ref[j];
            }
            page_fault[j] = 1;
            number_page_fault++;
        }
        /////
    }
    Print(total_page, n, page, ref);
}

int main() {
    int page; int temp;
    int ref[11] = { 2, 2, 5, 2, 1, 0, 6, 0, 0, 0, 7 };;
    int n = 11;
    cout << "--- Page Replacement algorithm ---" << endl;
    cout << "1. Default referenced sequence" << endl;
    cout << "2. Manual input sequence" << endl;
    cin >> temp;
    switch (temp) {
    case 1:

        break;
    case 2:
        cout << "Nhap so luong: "; cin >> n;
```

```
        cout << "Nhap danh sach trang: ";
        for (int i = 0; i < n; i++) {
            cin >> ref[i];
        }
    }
    cout << "--- Page Replacement algorithm ---" << endl;
    cout << "Input page frames: "; cin >> page;
    cout << "--- Select algorithm ---" << endl;
    cout << "1. FIFO algorithm" << endl;
    cout << "2. LRU algorithm" << endl;
    cout << "3. OPT algorithm" << endl;
    cout << "--- Enter input ---" << endl;

    cin >> temp;
    cout << "--- Page Replacement algorithm--- " << endl;
    switch (temp) {
    case 1:
        FIFO(ref, n, page);
        break;
    case 2:
        LRU(ref, n, page);
        break;
    case 3:
        OPT(ref, n, page);
        break;
    }

    system("pause");
```

```
    return 0;
}
```

- **Mô phỏng giải thuật trong phần 6.3.3**

- Chuỗi: 0 2 1 6 4 0 1 0 3 1 2 1

- Frame: 3

*** FIFO:**

```
leeminsun@leeminsun-VirtualBox:~/Desktop$ ./LAB6
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhap so luong: 12
Nhap danh sach trang: 0 2 1 6 4 0 1 0 3 1 2 1
--- Page Replacement algorithm ---
Input page frames: 3
--- Select algorithm ---
1. FIFO algorithm
2. LRU algorithm
3. OPT algorithm
--- Enter input ---
1
--- Page Replacement algorithm---
0 2 1 6 4 0 1 0 3 1 2 1
0 0 0 6 6 6 1 1 1 1 1 1
  2 2 2 4 4 4 4 3 3 3 3
    1 1 1 0 0 0 0 0 2 2
* * * * * * *   *   *
Number of Page Fault: 9
sh: 1: pause: not found
```

*** LRU:**

```
leeminsun@leeminsun-VirtualBox:~/Desktop$ ./LAB6
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhap so luong: 12
Nhap danh sach trang: 0 2 1 6 4 0 1 0 3 1 2 1
--- Page Replacement algorithm ---
Input page frames: 3
--- Select algorithm ---
1. FIFO algorithm
2. LRU algorithm
3. OPT algorithm
--- Enter input ---
2
--- Page Replacement algorithm---
0 2 1 6 4 0 1 0 3 1 2 1
0 0 0 6 6 6 1 1 1 1 1 1
  2 2 2 4 4 4 4 3 3 3 3
    1 1 1 0 0 0 0 0 2 2
* * * * * * *   *   *
Number of Page Fault: 9
sh: 1: pause: not found
```

*** OPT:**

```
leeminsun@leeminsun-VirtualBox:~/Desktop$ ./LAB6
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhap so luong: 12
Nhap danh sach trang: 0 2 1 6 4 0 1 0 3 1 2 1
--- Page Replacement algorithm ---
Input page frames: 3
--- Select algorithm ---
1. FIFO algorithm
2. LRU algorithm
3. OPT algorithm
--- Enter input ---
3
--- Page Replacement algorithm---
0 2 1 6 4 0 1 0 3 1 2 1
0 0 0 0 0 0 0 0 3 3 2 2
0 2 2 6 4 4 4 4 4 4 4 4
0 0 1 1 1 1 1 1 1 1 1 1
* * * * *       *   *
Number of Page Fault: 7
sh: 1: pause: not found
```

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 là hiện tượng số lỗi trang tăng lên khi tăng số frame.
- Chứng minh: Với chuỗi *1 2 3 4 1 2 5 1 2 3 4 5* (12 phần tử) và thuật toán *FIFO* ta có:

```
leeminsun@leeminsun-VirtualBox:~$ cd Desktop
leeminsun@leeminsun-VirtualBox:~/Desktop$ g++ LAB6.cpp -o LAB6
leeminsun@leeminsun-VirtualBox:~/Desktop$ ./LAB6
--- 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. LRU algorithm
3. OPT 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
sh: 1: pause: not found
leeminsun@leeminsun-VirtualBox:~/Desktop$ ./LAB6
--- 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. LRU algorithm
3. OPT 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
sh: 1: pause: not found
```

Chứng minh nghịch lý Belady

- *Với 3 frame có 9 lỗi trang, 4 frame lại có tới 10 lỗi trang.*

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

- Nhận xét:
  - Giải thuật FIFO: dễ cài đặt, dễ hiện thực, hiệu quả kém
  - Giải thuật LRU: khó cài đặt, phức tạp, hiệu quả
  - Giải thuật OPT: không khả thi, nhưng hiệu quả nhất
- Giải thuật bất khả thi nhất là **OPT** vì việc biết trước những trang nào có thể được truy xuất tiếp theo gần như là điều không thể.
- Giải thuật phức tạp nhất là **OPT** và **LRU** 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.