Name: Huỳnh Viết Tuấn Kiệt

ID: 20521494

Class: IT007.M13.2

OPERATING SYSTEM
LAB 6'S REPORT

SUMMARY

Task		**Status**	**Page**
Question	Ex 6.4: FIFO	Hoàn thành	2
	Ex 6.4: LRU	Hoàn thành	7
	Ex 6.4: OPT	Hoàn thành	12
	Ex 6.5: Bonus	Hoàn thành	18

Self-scrores: 10

Tham khảo sources code: Ubuntu-operating-system/Sources code Page Replacement at main · HiImKing1509/Ubuntu-operating-system (github.com)

** Mỗi giải thuật chạy 3 test case, trong đó:*

- ✓ *Case 1 khác nhau cho mỗi giải thuật*
- ✓ *Case 2 giống nhau cho các giải thuật để so sánh độ hiệu quả*
- ✓ *Case 3 giống nhau cho các giải thuật để so sánh độ hiệu quả*

1. Giải thuật thay trang FIFO

a. Ý tưởng thực hiện:

Bước 1: Khởi tạo

- Nhập số lượng page (`int`), mảng page truy xuất (**`vector<int>`**)
- Nhập số **`frame`**
- Khởi tạo **`vector page_fault`** chứa kí tự * hoặc khoảng trắng. Trong đó * biểu thị cho trang lỗi
- Khởi tạo **`vector FRAME`** chứa các page đang nằm trong các khung trang
- Khởi tạo ma trận **`vector<vector>`** chứa kết quả thay thế trang. Kiểu dữ liệu của ma trận là **`string`** để có thể biểu thị khoảng trống.

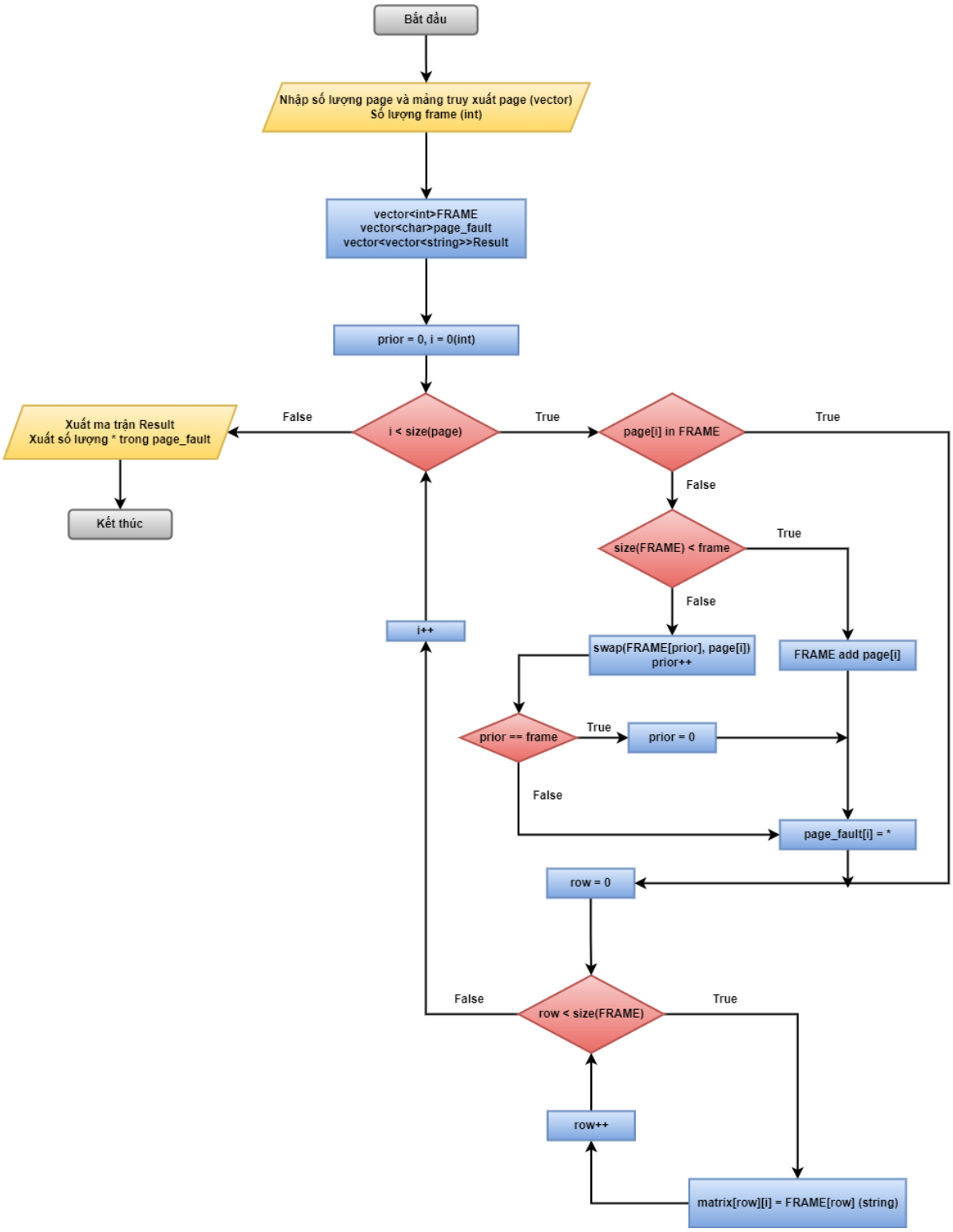
Bước 2: Xử lý

- Tạo biến **`prior = 0`**. Chức năng rất quan trọng, biến **`prior`** có tác dụng như một **`front(queue)`** cho biết page nào sẽ chọn thay thế tiếp theo. Phần tử **`FRAME[prior]`** sẽ là phần tử được chọn thay thế trực tiếp và không làm ảnh hưởng đến vị trí các phần tử còn lại trong **`FRAME`**
- Tạo vòng lặp **`for`** duyệt **`i`** từ 0 đến **`size(page)`**:
    - **`if page[i] in FRAME:`**
        - Đã có page đó trong **`FRAME`** thì không làm gì cả
    - **`else`**
        - **`if size(FRAME) < frame:`** Số page trong nằm trong **`FRAME`** ít hơn số frame được dùng. Thêm page mới vào **`FRAME`**
        - **`else`**
            - **`swap(FRAME[prior], page[i]):`** Thay page mới vào đúng vị trí của page được chọn trong **`FRAME`**
            - Tăng biến **`prior`** lên 1, page tiếp theo đó sẽ được chọn thay thế (thêm vào từ trên xuống thì độ ưu tiên cũng từ trên xuống)
            - Nếu giá trị **`prior`** vượt số lượng frame được cấp, gán lại **`prior = 0`** (page đầu tiên)
        - Gán **`page_fault[i] = *`** (lỗi trang)
    - Tạo vòng lặp đưa các phần tử có trong **`vector FRAME`** vào cột **`i`** của ma trận kết quả
- **`i = size(page):`** Duyệt hoàn tất, chuyển sang bước 3

Bước 3: Kết thúc

- In ra màn hình ma trận thay thế trang, **`vector page_fault`** và số lượng lỗi trang (bằng số lượng kí tự * trong **`vector page_fault`**

b. Lưu đồ giải thuật:



c. Hệ thống test case:

• Test 1

Chuỗi tham chiếu: 5 3 3 2 1 5 1 4 2 5 4 2 2 4 1

Số khung trang: 3

- **Chạy tay**

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

- **Thực thi chương trình**

```
kiet-20521494@kiet20521494-VirtualBox:~/LAB6$ ./bt1
Enter page number: 15
Manual input sequence: 5 3 3 2 1 5 1 4 2 5 4 2 2 4 1
Input page frames: 3

--- Page Replacement algorithm ---

0. Exit
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--> Algorithm selection or Exit (0->3): 1
        ------------------------------
        First in First out
        ------------------------------
Page:   5 3 3 2 1 5 1 4 2 5 4 2 2 4 1
        5 5 5 5 1 1 1 1 2 2 2 2 2 2 2
          3 3 3 3 5 5 5 5 5 5 5 5 5 1
              2 2 2 2 4 4 4 4 4 4 4 4
        ------------------------------
        * *   * * *   * *           *
Number of Page Fault: 8
```

- ## Test 2

Chuỗi tham chiếu: 0 2 4 0 3 1 6 4 5 5 3 3 2 2 5 3 2 5 3 0

Số khung trang: 4

- **Chạy tay**

0	2	4	0	3	1	6	4	5	5	3	3	2	2	5	3	2	5	3	0
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	3	3	3	3	3
	2	2	2	2	2	6	6	6	6	6	6	6	6	6	6	6	6	6	0
		4	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5	5	5
				3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2
*	*	*		*	*	*		*				*			*				*

Number of Page fault: 10

- **Thực thi chương trình**

```
kiet-20521494@kiet20521494-VirtualBox:~/LAB6$ ./bt1
Enter page number: 20
Manual input sequence: 0 2 4 0 3 1 6 4 5 5 3 3 2 2 5 3 2 5 3 0
Input page frames: 4

--- Page Replacement algorithm ---

0. Exit
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--> Algorithm selection or Exit (0->3): 1
        ----------------------------------------
        First in First out
        ----------------------------------------
Page:   0 2 4 0 3 1 6 4 5 5 3 3 2 2 5 3 2 5 3 0
        0 0 0 0 0 1 1 1 1 1 1 1 1 1 3 3 3 3 3
          2 2 2 2 2 6 6 6 6 6 6 6 6 6 6 6 6 6 0
            4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5
                3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2
        ----------------------------------------
        * * *   * * *   *       *     *       *
Number of Page Fault: 10
```

- # Test 3

Chuỗi tham chiếu: 1 3 0 5 5 6 4 3 0 6 5 1 2 3 0 4 0 2 5 1 4 3 0 6 4

Số khung trang: 5

- **Chạy tay**

1	3	0	5	5	6	4	3	0	6	5	1	2	3	0	4	0	2	5	1	4	3	0	6	4
1	1	1	1	1	1	4	4	4	4	4	4	4	4	4	4	4	4	5	5	5	5	5	5	5
	3	3	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1	1	1	4	4	4	4	4
		0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2	6	6
			5	5	5	5	5	5	5	5	5	5	3	3	3	3	3	3	3	3	3	3	3	3
					6	6	6	6	6	6	6	6	6	0	0	0	0	0	0	0	0	0	0	0
*	*	*	*		*	*					*	*	*	*				*		*			*		
Number of Page fault: 13																								

- **Thực thi chương trình**

```
kiet-20521494@kiet20521494-VirtualBox:~/LAB6$ ./bt1
Enter page number: 25
Manual input sequence: 1 3 0 5 5 6 4 3 0 6 5 1 2 3 0 4 0 2 5 1 4 3 0 6 4
Input page frames: 5

--- Page Replacement algorithm ---

0. Exit
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--> Algorithm selection or Exit (0->3): 1
        ------------------------------------------------
        First in First out
        ------------------------------------------------
Page:   1 3 0 5 5 6 4 3 0 6 5 1 2 3 0 4 0 2 5 1 4 3 0 6 4
        1 1 1 1 1 1 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5
          3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 4 4 4 4 4
            0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 6 6
              5 5 5 5 5 5 5 5 5 5 3 3 3 3 3 3 3 3 3 3 3
                  6 6 6 6 6 6 6 6 6 0 0 0 0 0 0 0 0 0 0
        ------------------------------------------------
        * * * *   * *         * * * *       *   *     *
Number of Page Fault: 13
```

2. Giải thuật thay trang LRU

a. Ý tưởng thực hiện

Bước 1: Khởi tạo

- Khởi tạo **vector<pair<int,int>>**page, trong đó mỗi phần tử lưu trữ 2 giá trị, **first** là chỉ số page truy xuất, **second** là thời gian page truy xuất (cách nhau 1 đơn vị)
- Nhập số lượng page **(int)**, mảng page truy xuất và gán thời gian các page truy xuất bắt đầu từ **0**
- Nhập số **frame**
- Khởi tạo **vector page_fault** chứa kí tự ***** hoặc khoảng trắng. Trong đó ***** biểu thị cho trang lỗi
- Khởi tạo **vector<pair<int,int>> FRAME** chứa các page đang nằm trong các khung trang
- Khởi tạo ma trận **vector<vector>** chứa kết quả thay thế trang. Kiểu dữ liệu của ma trận là **string** để có thể biểu thị khoảng trống.

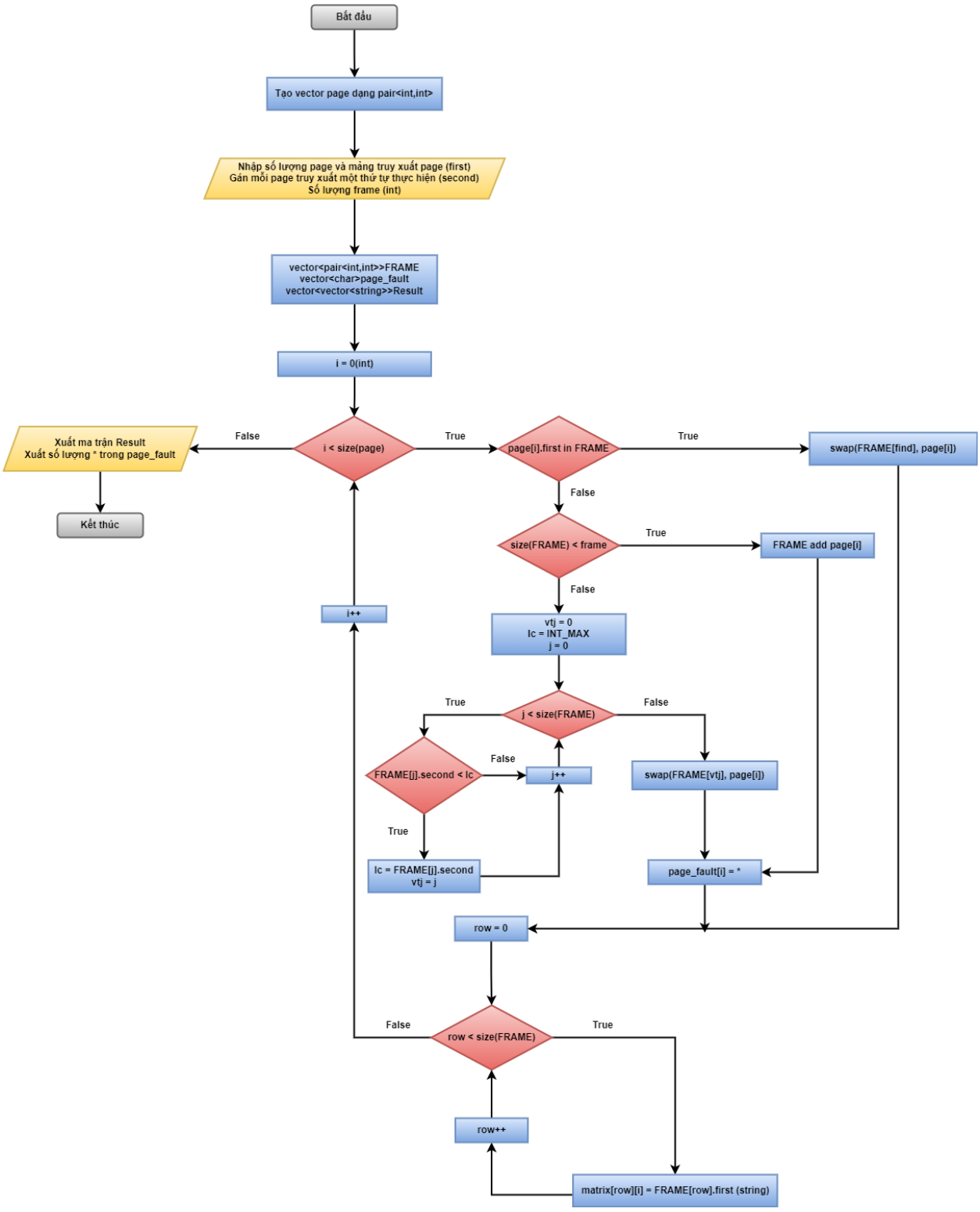
Bước 2: Xử lý

- Tạo vòng lặp **for** duyệt **i** từ **0** đến **size(page):**
    - **if page[i].first in FRAME:**
        - Đã có page đó trong **FRAME**, vẫn hoán vị để thay đổi mốc thời gian truy xuất
    - **else**
        - **if size(FRAME) < frame:** Số page trong nằm trong **FRAME** ít hơn số **frame** được dùng. Thêm page mới vào **FRAME**
        - **else**
            - Tạo vòng lặp tìm page có **page[j].second** nhỏ nhất trong **FRAME**. Vị trí page tìm được gán cho biến **vtj**
            - **swap(FRAME[vtj], page[i]):** Thay page mới vào đúng vị trí của page được chọn trong **FRAME**
        - Gán **page_fault[i] = *** (lỗi trang)
    - Tạo vòng lặp đưa các phần tử **first** có trong vector **FRAME** vào cột **i** của ma trận kết quả
- **i = size(page):** Duyệt hoàn tất, chuyển sang bước 3

Bước 3: Kết thúc

- In ra màn hình ma trận thay thế trang, **`vector page_fault`** và số lượng lỗi trang (bằng số lượng kí tự **`*`** trong **`vector page_faul`**

b. Lưu đồ giải thuật



c. Hệ thống test case

• Test 1

Chuỗi tham chiếu: 0 3 0 4 3 1 0 1 4 0 2 0 2 4 1

Số khung trang: 3

- **Chạy tay**

0	3	0	4	3	1	0	1	4	0	2	0	2	4	1
0	0	0	0	0	1	1	1	1	1	2	2	2	2	2
	3	3	3	3	3	3	3	4	4	4	4	4	4	4
			4	4	4	0	0	0	0	0	0	0	0	1
*	*		*		*	*		*		*				*
Number of Page Fault: 8														

- **Thực thi chương trình**

```
kiet-20521494@kiet20521494-VirtualBox:~/LAB6$ ./bt1
Enter page number: 15
Manual input sequence: 0 3 0 4 3 1 0 1 4 0 2 0 2 4 1
Input page frames: 3

--- Page Replacement algorithm ---

0. Exit
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--> Algorithm selection or Exit (0->3): 3
        ------------------------------
        Least Recent Used
        ------------------------------
Page:   0 3 0 4 3 1 0 1 4 0 2 0 2 4 1
        0 0 0 0 0 1 1 1 1 1 2 2 2 2 2
          3 3 3 3 3 3 3 4 4 4 4 4 4 4
              4 4 4 0 0 0 0 0 0 0 1
        ------------------------------
        * *   *   * *   *   *       *
Number of Page Fault: 8
```

- **Test 2**

Chuỗi tham chiếu: 0 2 4 0 3 1 6 4 5 5 3 3 2 2 5 3 2 5 3 0

Số khung trang: 4

- **Chạy tay**

0	2	4	0	3	1	6	4	5	5	3	3	2	2	5	3	2	5	3	0
0	0	0	0	0	0	0	4	4	4	4	4	4	4	4	4	4	4	4	0
	2	2	2	2	1	1	1	1	1	3	3	3	3	3	3	3	3	3	3
		4	4	4	4	6	6	6	6	6	6	2	2	2	2	2	2	2	2
				3	3	3	3	5	5	5	5	5	5	5	5	5	5	5	5
*	*	*		*	*	*	*	*		*		*							*

Number of Page fault: 11

- **Thực thi chương trình**

```
kiet-20521494@kiet20521494-VirtualBox:~/LAB6$ ./bt1
Enter page number: 20
Manual input sequence: 0 2 4 0 3 1 6 4 5 5 3 3 2 2 5 3 2 5 3 0
Input page frames: 4

--- Page Replacement algorithm ---

0. Exit
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--> Algorithm selection or Exit (0->3): 3
        ----------------------------------------
        Least Recent Used
        ----------------------------------------
Page:   0 2 4 0 3 1 6 4 5 5 3 3 2 2 5 3 2 5 3 0
        0 0 0 0 0 0 0 4 4 4 4 4 4 4 4 4 4 4 4 0
          2 2 2 2 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3
            4 4 4 4 6 6 6 6 6 6 2 2 2 2 2 2 2 2
                3 3 3 3 5 5 5 5 5 5 5 5 5 5 5 5 5
        ----------------------------------------
        * * *   * * * * *   *   *             *
Number of Page Fault: 11
```

- ## Test 3

Chuỗi tham chiếu: 1 3 0 5 5 6 4 3 0 6 5 1 2 3 0 4 0 2 5 1 4 3 0 6 4

Số khung trang: 5

- **Chạy tay**

1	3	0	5	5	6	4	3	0	6	5	1	2	3	0	4	0	2	5	1	4	3	0	6	4
1	1	1	1	1	1	4	4	4	4	4	1	1	1	1	1	1	1	5	5	5	5	5	6	6
	3	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	3	3	3	3	3	3	1	1	1	1	1	1
			5	5	5	5	5	5	5	5	5	5	5	5	4	4	4	4	4	4	4	4	4	4
					6	6	6	6	6	6	6	6	6	0	0	0	0	0	0	0	3	3	3	3
*	*	*	*		*	*					*	*	*	*	*			*	*		*	*	*	
Number of Page fault: 16																								

- **Thực thi chương trình**

```
kiet-20521494@kiet20521494-VirtualBox:~/LAB6$ ./bt1
Enter page number: 25
Manual input sequence: 1 3 0 5 5 6 4 3 0 6 5 1 2 3 0 4 0 2 5 1 4 3 0 6 4
Input page frames: 5

--- Page Replacement algorithm ---

0. Exit
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--> Algorithm selection or Exit (0->3): 3
        -------------------------------------------------
        Least Recent Used
        -------------------------------------------------
Page:   1 3 0 5 5 6 4 3 0 6 5 1 2 3 0 4 0 2 5 1 4 3 0 6 4
        1 1 1 1 1 1 4 4 4 4 4 1 1 1 1 1 1 1 5 5 5 5 5 6 6
          3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 0 0 0
            0 0 0 0 0 0 0 0 0 0 0 3 3 3 3 3 3 1 1 1 1 1 1
              5 5 5 5 5 5 5 5 5 5 5 5 4 4 4 4 4 4 4 4 4 4
                  6 6 6 6 6 6 6 6 6 0 0 0 0 0 0 0 3 3 3 3
        -------------------------------------------------
        * * * *   * *         * * * * *     * *   * * *
Number of Page Fault: 16
```

3. Giải thuật thay trang OPT

a. Ý tưởng thực hiện

Bước 1: Khởi tạo

- Nhập số lượng page **(int)**, mảng page truy xuất **(vector<int>)**
- Nhập số **frame**
- Khởi tạo **vector page_fault** chứa kí tự ***** hoặc khoảng trắng. Trong đó ***** biểu thị cho trang lỗi
- Khởi tạo **vector FRAME** chứa các page đang nằm trong các khung trang
- Khởi tạo ma trận **vector<vector>** chứa kết quả thay thế trang. Kiểu dữ liệu của ma trận là **string** để có thể biểu thị khoảng trống.

Bước 2: Xử lý

- Tạo vòng lặp **for** duyệt **i** từ **0** đến **size(page):**
    - **if page[i] in FRAME:**
        - Đã có page đó trong **FRAME** thì không làm gì cả
    - **else**
        - **if size(FRAME) < frame:** Số page trong nằm trong **FRAME** ít hơn số **frame** được dùng. Thêm page mới vào **FRAME**
        - **else**
            - Khởi tạo biến **order = 0**. Cho biết thứ tự đến trước hay sau của mỗi page nếu nó xuất hiện trong **FRAME** và được truy cập trong tương lai
            - Khởi tạo biến **prior = -1**. Biến này là vị trí của page sẽ được chọn để thay thế
            - Khởi tạo **vector<int>VEC** có kích thước bằng **frame** và khởi tạo ban đầu toàn giá trị **-1**. Có thể coi **vector VEC** là mảng cờ hiệu của vector **FRAME**. Nếu phần tử trong **FRAME** có xuất hiện trong tương lai, phần tử tại vị trí tương ứng trong **VEC** sẽ được bật lên để báo hiệu phần tử đã xuất hiện. (Xem tiếp để hiểu)
            - Duyệt **j = i + 1** đến **size(page)**. Đồng nghĩa với duyệt các phần tử còn lại của page chưa được truy xuất. Duyệt như sau:
                - Nếu **page[j]** nằm trong **FRAME**, đặt biến **vt** = vị trí của page đó trong **FRAME**. Nếu **VEC[vt] = -1**, tức là page đó xuất hiện lần đầu tiên trong tương lai
                    - Gán **VEC[vt] = order** biểu thị page đó đã xuất hiện và đồng thời cũng biết được thứ tự xuất hiện

của page đó so với các page còn lại trong **FRAME** nếu được truy xuất trong tương lai.

- **Prior = vt**. Tạm thời vị trí vt đang là vị trí được truy xuất trễ nhất
- **order++.** Tăng biến **order** lên để biểu thị các page được truy xuất trước hay sau. **order** càng lớn thì truy xuất càng trễ

- Ngược lại, nếu **page[j]** không nằm trong **FRAME** hoặc nằm trong **FRAME** nhưng đã được truy xuất trước đó (**VEC[vt] != -1**) thì bỏ qua.

- **If order = frame:** Tất cả các page trong **FRAME** đều được truy xuất trong tương lai. Lúc này **prior** vẫn đang giữ vị trí **vt** là vị trí của phần tử có thứ tự xuất hiện trễ nhất sau khi kết thúc vòng lặp trên
- **Else:** Có ít nhất 1 page hoặc tất cả các page trong **FRAME** không được truy xuất trong tương lai.
  - Duyệt vòng lặp hết **FRAME** để tìm page không xuất hiện. **page[j]** không xuất hiện trong **FRAME** tương ứng với **VEC[j] = -1**
    - Theo mặc định nếu nhiều page không được truy xuất trong tương lai thì chọn page trên cùng của FRAME. Do đó khi tìm được **VEC[j] = -1**:
      - Gán **prior = j**
      - Lệnh **break** dừng vòng lặp ngay lập tức
  - **Swap(FRAME[prior], page[i])**
- Gán **page_fault[i] = *** (lỗi trang)

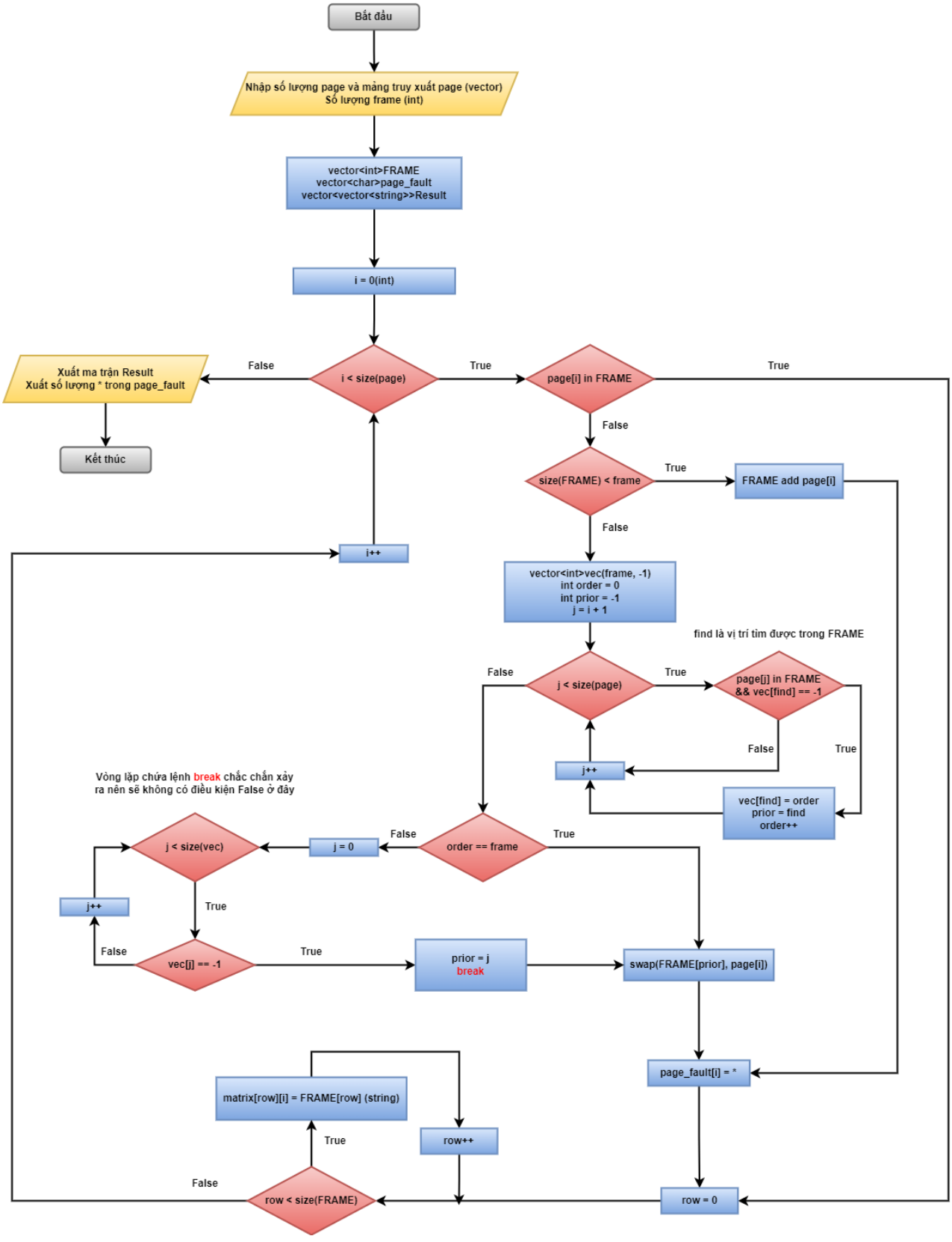
- Tạo vòng lặp đưa các phần tử có trong **vector FRAME** vào cột **i** của ma trận kết quả

- **i = size(page):** Duyệt hoàn tất, chuyển sang bước 3

Bước 3: Kết thúc

- In ra màn hình ma trận thay thế trang, **vector page_fault** và số lượng lỗi trang (bằng số lượng kí tự ***** trong **vector page_faul**

b. Lưu đồ giải thuật



c. Hệ thống test case

• Test 1

Chuỗi tham chiếu: 0 3 0 1 2 1 1 3 2 4 1 4 0 4 0

Số khung trang: 3

- **Chạy tay**

0	3	0	1	2	1	1	3	2	4	1	4	0	4	0
0	0	0	0	2	2	2	2	2	4	4	4	4	4	4
	3	3	3	3	3	3	3	3	3	3	3	0	0	0
			1	1	1	1	1	1	1	1	1	1	1	1
*	*		*	*					*			*		
Number of Page Fault: 6														

- **Thực thi chương trình**

```
kiet-20521494@kiet20521494-VirtualBox:~/LAB6$ ./bt1
Enter page number: 15
Manual input sequence: 0 3 0 1 2 1 1 3 2 4 1 4 0 4 0
Input page frames: 3

--- Page Replacement algorithm ---

0. Exit
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--> Algorithm selection or Exit (0->3): 2
        ------------------------------
        Optimal
        ------------------------------
Page:   0 3 0 1 2 1 1 3 2 4 1 4 0 4 0
        0 0 0 0 2 2 2 2 2 4 4 4 4 4 4
          3 3 3 3 3 3 3 3 3 3 0 0 0
              1 1 1 1 1 1 1 1 1 1 1
        ------------------------------
        * *   * *         *     *
Number of Page Fault: 6
```

- **Test 2**

Chuỗi tham chiếu: 0 2 4 0 3 1 6 4 5 5 3 3 2 2 5 3 2 5 3 0

Số khung trang: 4

- **Chạy tay**

0	2	4	0	3	1	6	4	5	5	3	3	2	2	5	3	2	5	3	0
0	0	0	0	0	1	6	6	5	5	5	5	5	5	5	5	5	5	5	0
	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	4	4	4	4	4	4	4	4
				3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
*	*	*		*	*	*		*											*
Number of Page fault: 8																			

- **Thực thi chương trình**

```
kiet-20521494@kiet20521494-VirtualBox:~/LAB6$ ./bt1
Enter page number: 20
Manual input sequence: 0 2 4 0 3 1 6 4 5 5 3 3 2 2 5 3 2 5 3 0
Input page frames: 4

--- Page Replacement algorithm ---

0. Exit
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--> Algorithm selection or Exit (0->3): 2
        ----------------------------------------
        Optimal
        ----------------------------------------
Page:   0 2 4 0 3 1 6 4 5 5 3 3 2 2 5 3 2 5 3 0
        0 0 0 0 0 1 6 6 5 5 5 5 5 5 5 5 5 5 5 0
          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 4 4 4 4 4 4 4 4
                3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
        ----------------------------------------
        * * *   * * *   *                     *
Number of Page Fault: 8
```

• Test 3

Chuỗi tham chiếu: 1 3 0 5 5 6 4 3 0 6 5 1 2 3 0 4 0 2 5 1 4 3 0 6 4

Số khung trang: 5

- **Chạy tay**

1	3	0	5	5	6	4	3	0	6	5	1	2	3	0	4	0	2	5	1	4	3	0	6	4
1	1	1	1	1	1	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	6	6
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
			5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	1	1	1	1	1	1
					6	6	6	6	6	6	1	2	2	2	2	2	2	2	2	2	2	2	2	2
*	*	*	*		*	*					*	*							*				*	
Number of Page fault: 10																								

- **Thực thi chương trình**

```
kiet-20521494@kiet20521494-VirtualBox:~/LAB6$ ./bt1
Enter page number: 25
Manual input sequence: 1 3 0 5 5 6 4 3 0 6 5 1 2 3 0 4 0 2 5 1 4 3 0 6 4
Input page frames: 5

--- Page Replacement algorithm ---

0. Exit
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--> Algorithm selection or Exit (0->3): 2
        ------------------------------------------------
        Optimal
        ------------------------------------------------
Page:   1 3 0 5 5 6 4 3 0 6 5 1 2 3 0 4 0 2 5 1 4 3 0 6 4
        1 1 1 1 1 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
          3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 6 6
            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
              5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 1 1 1 1 1 1
                6 6 6 6 6 6 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2
        ------------------------------------------------
        * * * *   * *         * *             *       *
Number of Page Fault: 10
```

4. Bài tập bonus

a. Nghịch lí belady là gì? Sử dụng chương trình đã viết trên để chứng minh nghịch lí này

- Số lượng lỗi trang xảy ra sẽ tăng lên khi số lượng khung trang sử dụng tăng. Hiện tượng này gọi là **nghịch lý Belady**.
- Ví dụ:

Chuỗi tham chiếu: 3 2 1 0 3 2 4 3 2 1 0 4
Số khung trang: 3 & 4
Giải thuật: FIFO

- **Kết quả thực thi trên 3 khung trang**

```
kiet-20521494@kiet20521494-VirtualBox:~/LAB6$ ./bt1
Enter page number: 12
Manual input sequence: 3 2 1 0 3 2 4 3 2 1 0 4
Input page frames: 3

--- Page Replacement algorithm ---

0. Exit
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--> Algorithm selection or Exit (0->3): 1
        ------------------------
        First in First out
        ------------------------
Page:   3 2 1 0 3 2 4 3 2 1 0 4
        3 3 3 0 0 0 4 4 4 4 4 4
          2 2 2 3 3 3 3 3 1 1 1
            1 1 1 2 2 2 2 2 0 0
        ------------------------
        * * * * * * *     * *
Number of Page Fault: 9
```

- **Kết quả thực thi trên 4 khung trang**

```
kiet-20521494@kiet20521494-VirtualBox:~/LAB6$ ./bt1
Enter page number: 12
Manual input sequence: 3 2 1 0 3 2 4 3 2 1 0 4
Input page frames: 4

--- Page Replacement algorithm ---

0. Exit
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--> Algorithm selection or Exit (0->3): 1
        ------------------------
        First in First out
        ------------------------
Page:   3 2 1 0 3 2 4 3 2 1 0 4
        3 3 3 3 3 3 4 4 4 4 0 0
          2 2 2 2 2 2 3 3 3 3 4
            1 1 1 1 1 1 2 2 2 2
              0 0 0 0 0 0 1 1 1
        ------------------------
        * * * *     * * * * * *
Number of Page Fault: 10
```

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

Giải thuật	Nhận xét
FIFO	Dễ cài đặt, dễ hiện thực nhưng hiệu quả đạt được không cao
LRU	Tương đối khó cài đặt, độ phức tạp cao, khá hiệu quả
OPT	Cực kì phức tạp, không khả thi, kết quả đạt được hiệu quả cao vì nó sinh ra để giải quyết nghịch lí Belady

- **Giải thuật nào là bất khả thi nhất? Vì sao?**
- **Giải thuật nào là phức tạp nhất? Vì sao?**

Giải thuật bất khả thi và phức tạp nhất
Giải thuật **OPT** bất khả thi nhất vì yêu cầu phải biết được các trang xuất hiện trong tương lai
Giải thuật **OPT** và **LRU** là giải thuật phức tạp nhất bởi vì khi xuất hiện lỗi trang, để tìm được trang thay thế thì phải phải xét đến toàn bộ chuỗi tham chiếu trước hay sau trang đó. (Với **LRU** thì xét chuỗi trước, **OPT** tham chiếu chuỗi sau)

HẾT