|  |
| --- |
| Name: Phạm Đức Thể  ID: 19522253  Class: IT007.M14.2 |

OPERATING SYSTEM  
LAB 6 REPORT

**SUMMARY**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | | **Status** | **Page** |
| Section 6.4 | FIFO | Hoàn thành | 2 |
| LRU | Hoàn thành | 7 |
| OPT | Hoàn thành | 10 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Section 6.5 | Ex 1 | Hoàn thành | 21 |
| Ex 2 | Hoàn thành | 22 |

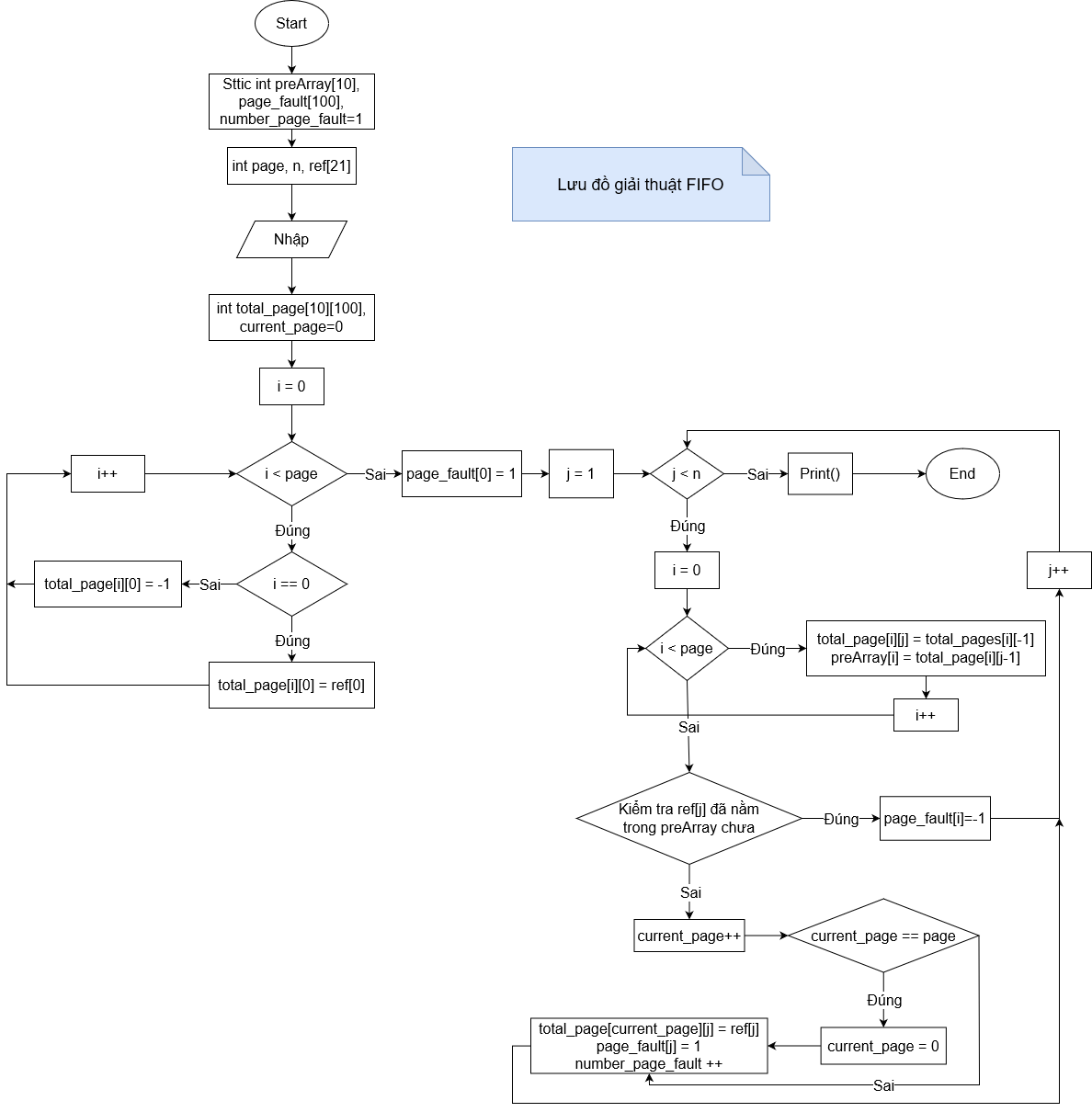
**Self-scrores: 10/10**

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

# Section 6.4

## Task name 1: Giải thuật FIFO

* **Lưu đồ thuật toán**



Hình 1: Lưu đồ thuật toán FIFO

* **Giải thích**
* Bước 1: Khai báo các biến cần thiết như là: ref[100] là mảng các trang, n là số trang (cùng là chiều dài của mảng ref), page là số khung trang. Sau đó tiến hành xây dựng hàm nhập để nhập các thống số vào 3 biến này. Ngoài ra còn cần các biến nhu 1 mảng chứa các thông tin của một cột, 1 mảng để lưu các trang bị lỗi và 1 biến đếm các trang bị lỗi.
* Bước 2: Tạo mảng 2 chiều total\_page[][] là cái bảng lưu trữ thông tin, là biến curent\_page để tác định trang mình đang làm việc.
* Bước 3: Điền chứ số đầu tiên của mảng ref vào cộ đầu tiên của mảng 2 chiều, các trang còn lại của cột mà ko có giá trị sẽ được điền bằng -1.
* Bước 4: Tiến hành xét các giá trị tiếp theo của mảng.
* Bước 5: Sao chép cột đang xét bằng cột ở trước đó, Đồng thời sao chép là mảng preArry[], cũng là giá trị của cột trước đó để tiến hành xét với giá trị trong mảng là ref[j] đang xét. Nếu giá trị ref[j] nằm ở trong mảng ròi thì tiến hành j++ để thực hiện các tiến trình tiếp theo. Nếu sai thì xét từng giá trị của page để tìm ra vị trí cần thay thế và đánh dấu lỗi trang lúc, tiến hành xét các giá trị tiếp theo cho đến hết.
* Bước 6: Sau khi lặp hết các giá trị trong mảng ref[] thì tiến hành in bảng total\_page và các giá trị lỗi sai.
* **Test Case**
* Ví dụ 1: Xét chuỗi truy xuất bộ nhớ sau: 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6. Có bao nhiêu lỗi trang xảy ra khi sử dụng thuật toán thay thế FIFO, giả sử có 5 khung trang?

+ Lời giải:

Giải bằng tay:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 2 | 1 | 5 | 6 | 2 | 1 | 2 | 3 | 7 | 6 | 3 | 2 | 1 | 2 | 3 | 6 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
|  | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|  |  | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
|  |  |  | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
|  |  |  |  |  |  | 5 | 5 | 5 | 5 | 5 | 5 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| \* | \* | \* | \* |  |  | \* | \* |  | \* | \* | \* | \* |  |  |  |  |  |  |  |

=> Tổng cộng có 10 lỗi trang.

Giải bằng code:

Text

Description automatically generated

Hình 2: Kết quả giải ví dụ 1 bằng thuật toán FIFO

* Ví dụ 2: Xét chuỗi truy xuất bộ nhớ sau: 1, 3, 4, 5, 2, 3, 6, 3. Có bao nhiêu lỗi trang xảy ra khi sử dụng thuật toán thay thế FIFO, giả sử có 4 khung trang?

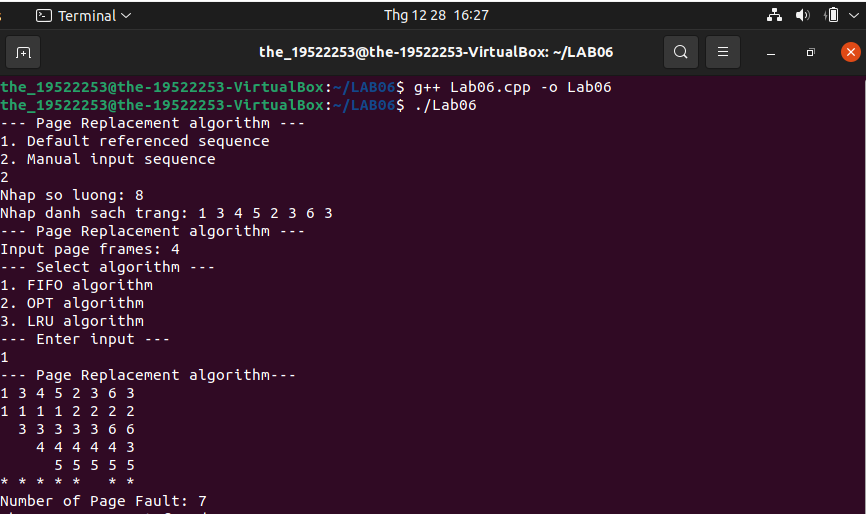
+ Lời giải:

Giải bằng tay:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 5 | 2 | 3 | 6 | 3 |
| 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 |
|  | 3 | 3 | 3 | 3 | 3 | 6 | 6 |
|  |  | 4 | 4 | 4 | 4 | 4 | 3 |
|  |  |  | 5 | 5 | 5 | 5 | 5 |
| \* | \* | \* | \* | \* |  | \* | \* |

=> Tổng cộng có 7 lỗi trang.

Giải bằng code:



Hình 3: Kết quả giải ví dụ 2 bằng thuật toán FIFO

* Ví dụ 3: Xét chuỗi truy xuất bộ nhớ sau: 6, 2, 4, 4, 5, 6, 3, 1, 4, 2, 3, 7, 5, 6, 7, 2, 4, 3, 5, 1. Có bao nhiêu lỗi trang xảy ra khi sử dụng thuật toán thay thế FIFO, giả sử có 4 khung trang?

+ Lời giải:

Giải bằng tay:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | 2 | 4 | 4 | 5 | 6 | 3 | 1 | 4 | 2 | 3 | 7 | 5 | 6 | 7 | 2 | 4 | 3 | 5 | 1 |
| 6 | 6 | 6 | 6 | 6 | 6 | 3 | 3 | 3 | 3 | 3 | 3 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 1 |
|  | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
|  |  | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 4 | 4 | 4 | 4 |
|  |  |  |  | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 7 | 7 | 7 | 7 | 7 | 7 | 3 | 3 | 3 |
| \* | \* | \* |  | \* |  | \* | \* |  | \* |  | \* | \* | \* |  |  | \* | \* |  | \* |

=> Tổng cộng có 13 lỗi trang.

Giải bằng code:

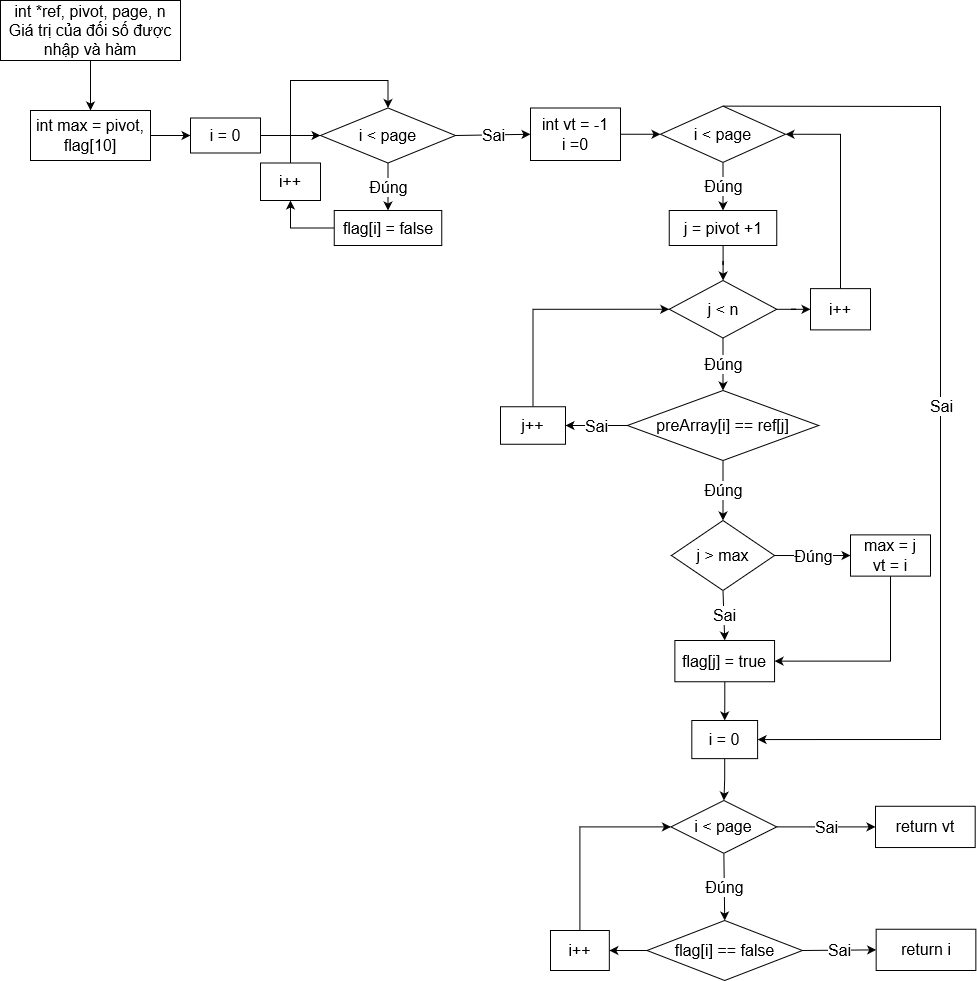
Text

Description automatically generated

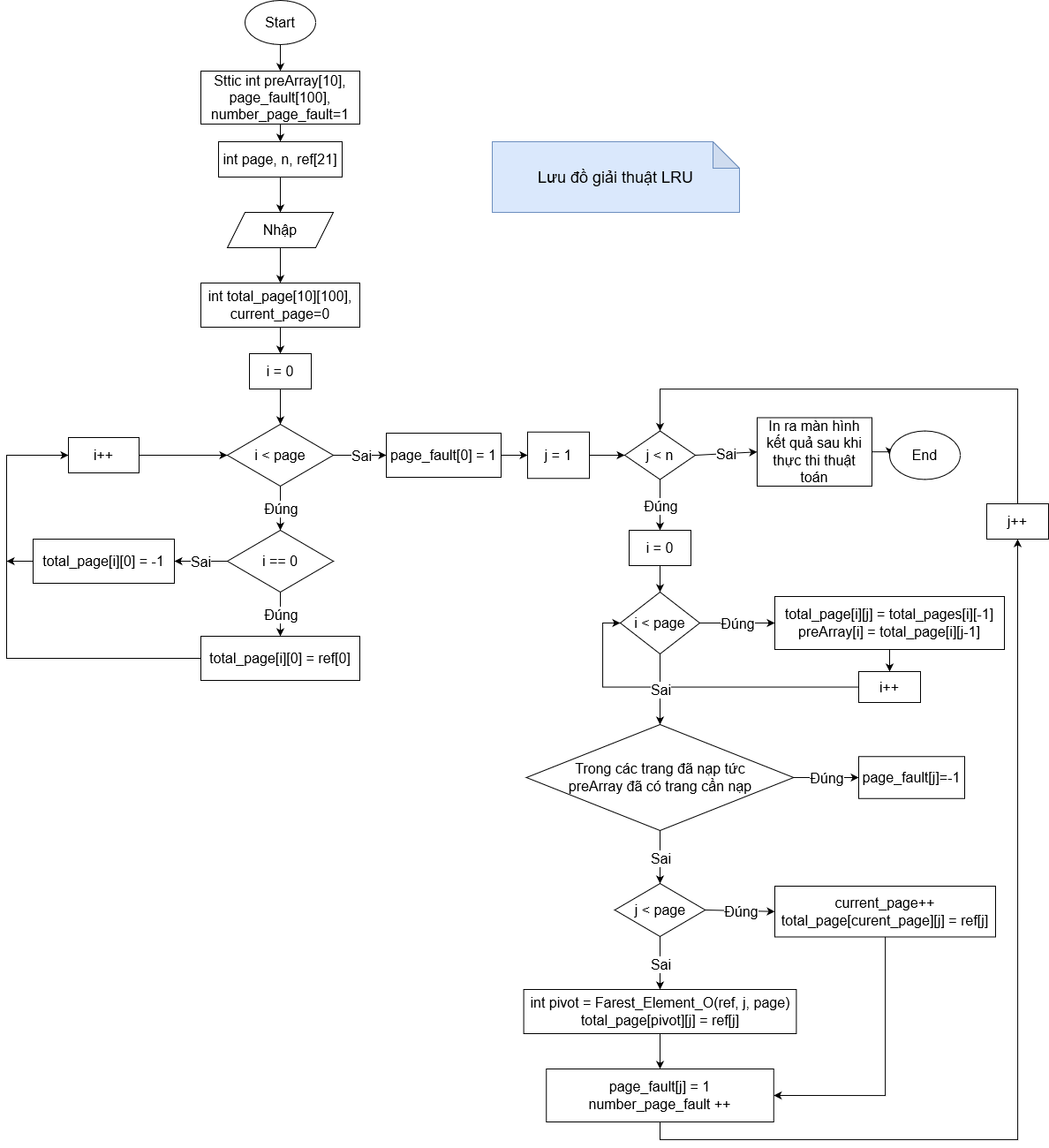
Hình 4: Kết quả giải ví dụ 3 bằng thuật toán FIFO

## Task name 2: Giải thuật LRU

* **Lưu đồ thuật toán**



Hình 5: Lưu đồ hàm tìm ra trang được gói sớm nhất trong quá khứ



Hình 6: Lưu đồ thuật toán LRU.

* **Giải thích**
* Hình 5: Tiến hành lọc quá tất cả các trang có trong cột và từ đó xét với mảng ref để xem trang nào được truy xuất sớm nhất trong quá khứ.
* Hình 6: Cũng tương tự như thuật toán OPT thì ở Bước 5, để chọn ra vị trí page cần thay thế thì ta sẽ sử dụng hàm **Farest\_Element\_O** để xác định ví trị của trang được truy xuất sớm nhất trong quá khứ. Còn lại tất cả các bước thì như nhau.
* **Test Case**
* Ví dụ 1: Xét chuỗi truy xuất bộ nhớ sau: 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6. Có bao nhiêu lỗi trang xảy ra khi sử dụng thuật toán thay thế LRU, giả sử có 5 khung trang?

+ Lời giải:

Giải bằng tay:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 2 | 1 | 5 | 6 | 2 | 1 | 2 | 3 | 7 | 6 | 3 | 2 | 1 | 2 | 3 | 6 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|  | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
|  |  | 3 | 3 | 3 | 3 | 3 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
|  |  |  | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
|  |  |  |  |  |  | 5 | 5 | 5 | 5 | 5 | 5 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| \* | \* | \* | \* |  |  | \* | \* |  |  |  | \* | \* |  |  |  |  |  |  |  |

=> Tổng cộng có 8 lỗi trang.

Giải bằng code:

Text

Description automatically generated

Hình 7: Kết quả giải ví dụ 1 bằng thuật toán LRU

* Ví dụ 2: Xét chuỗi truy xuất bộ nhớ sau: 1, 3, 4, 5, 2, 3, 6, 3. Có bao nhiêu lỗi trang xảy ra khi sử dụng thuật toán thay thế LRU, giả sử có 4 khung trang?

+ Lời giải:

Giải bằng tay:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 5 | 2 | 3 | 6 | 3 |
| 1 | 1 | 1 | 1 | 2 | 2 | 6 | 6 |
|  | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
|  |  | 4 | 4 | 4 | 4 | 4 | 4 |
|  |  |  | 5 | 5 | 5 | 5 | 5 |
| \* | \* | \* | \* | \* |  | \* |  |

=> Tổng cộng có 6 lỗi trang.

Giải bằng code:

Text

Description automatically generated

Hình 8: Kết quả giải ví dụ 2 bằng thuật toán LRU

* Ví dụ 3: Xét chuỗi truy xuất bộ nhớ sau: 6, 2, 4, 4, 5, 6, 3, 1, 4, 2, 3, 7, 5, 6, 7, 2, 4, 3, 5, 1. Có bao nhiêu lỗi trang xảy ra khi sử dụng thuật toán thay thế LRU, giả sử có 4 khung trang?

+ Lời giải:

Giải bằng tay:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | 2 | 4 | 4 | 5 | 6 | 3 | 1 | 4 | 2 | 3 | 7 | 5 | 6 | 7 | 2 | 4 | 3 | 5 | 1 |
| 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 2 | 2 | 2 | 2 | 6 | 6 | 6 | 6 | 3 | 3 | 3 |
|  | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 1 |
|  |  | 4 | 4 | 4 | 4 | 4 | 1 | 1 | 1 | 1 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 5 | 5 |
|  |  |  |  | 5 | 5 | 5 | 5 | 4 | 4 | 4 | 4 | 5 | 5 | 5 | 5 | 4 | 4 | 4 | 4 |
| \* | \* | \* |  | \* |  | \* | \* | \* | \* |  | \* | \* | \* |  | \* | \* | \* | \* | \* |

=> Tổng cộng có 16 lỗi trang.

Giải bằng code:

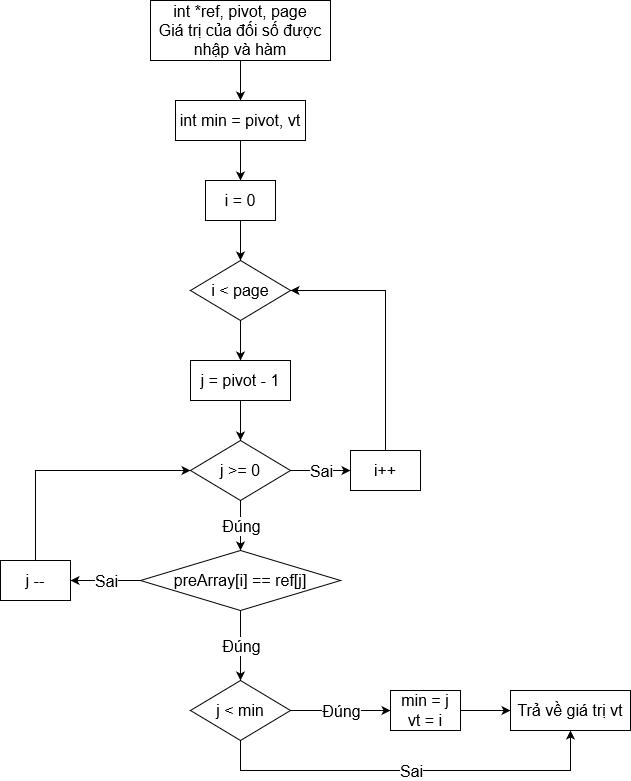
Text

Description automatically generated

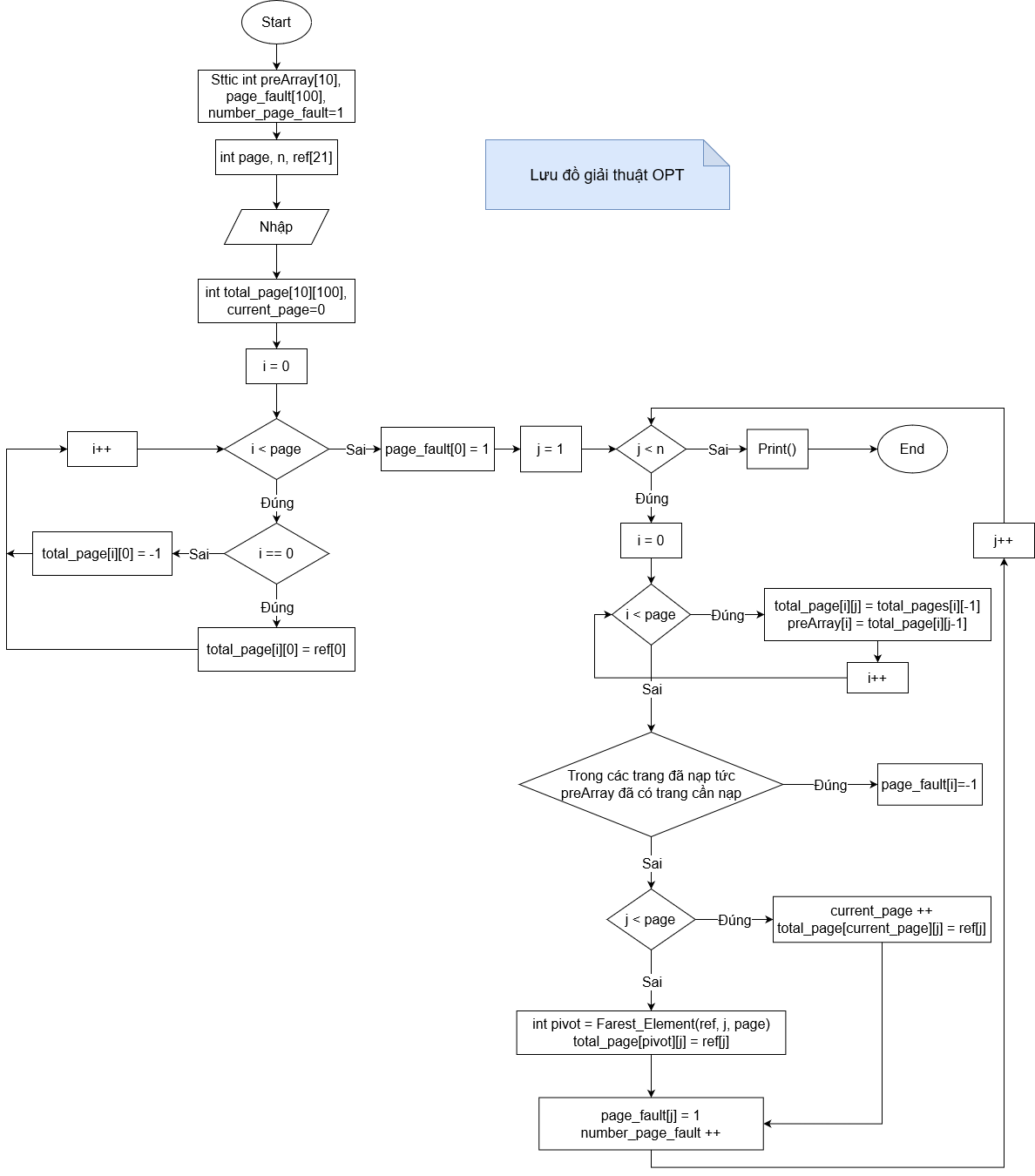
Hình 9: Kết quả giải ví dụ 3 bằng thuật toán LRU

## Task name 3: Giải thuật OPT

* **Lưu đồ thuật toán**



Hình 10: Lưu đồ hàm tìm ra vị trí cần được thay thế.



Hình 11: Lưu đồ thuật toán OPT.

* **Giải thích**
* Hình 10: Hàm sẽ tiến hành duyệt qua tất cả các giá trị trong cột và đi tìm vị trí trong mảng của mảng ref[] để tìm ra trang sẽ được gọi lại muộn nhất trong tương lai.
* Hình 11: Tương tự như các bước của thuật toán FIFO, ở thuật toán OPT tại bước 5 để xác đinh vị trí page cần thay đế ta sẽ tiến hành gọi hàm **Farest\_Element** (xác định ví trị trang cần thay thế) ròi tất cả mọi bước thì được thực hiện như FIFO.
* **Test Case**
* Ví dụ 1: Xét chuỗi truy xuất bộ nhớ sau: 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6. Có bao nhiêu lỗi trang xảy ra khi sử dụng thuật toán thay thế OPT, giả sử có 5 khung trang?

+ Lời giải:

Giải bằng tay:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 2 | 1 | 5 | 6 | 2 | 1 | 2 | 3 | 7 | 6 | 3 | 2 | 1 | 2 | 3 | 6 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|  | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
|  |  | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
|  |  |  | 4 | 4 | 4 | 4 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
|  |  |  |  |  |  | 5 | 5 | 5 | 5 | 5 | 5 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| \* | \* | \* | \* |  |  | \* | \* |  |  |  |  | \* |  |  |  |  |  |  |  |

=> Tổng cộng có 7 lỗi trang.

Giải bằng code:

Text

Description automatically generated

Hình 12: Kết quả giải ví dụ 1 bằng thuật toán OPT

* Ví dụ 2: Xét chuỗi truy xuất bộ nhớ sau: 1, 3, 4, 5, 2, 3, 6, 3. Có bao nhiêu lỗi trang xảy ra khi sử dụng thuật toán thay thế OPT, giả sử có 4 khung trang?

+ Lời giải:

Giải bằng tay:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 5 | 2 | 3 | 6 | 3 |
| 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 |
|  | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
|  |  | 4 | 4 | 4 | 4 | 6 | 6 |
|  |  |  | 5 | 5 | 5 | 5 | 5 |
| \* | \* | \* | \* | \* |  | \* |  |

=> Tổng cộng có 6 lỗi trang.

Giải bằng code:

Text

Description automatically generated

Hình 13: Kết quả giải ví dụ 2 bằng thuật toán OPT

* Ví dụ 3: Xét chuỗi truy xuất bộ nhớ sau: 6, 2, 4, 4, 5, 6, 3, 1, 4, 2, 3, 7, 5, 6, 7, 2, 4, 3, 5, 1. Có bao nhiêu lỗi trang xảy ra khi sử dụng thuật toán thay thế OPT, giả sử có 4 khung trang?

+ Lời giải:

Giải bằng tay:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | 2 | 4 | 4 | 5 | 6 | 3 | 1 | 4 | 2 | 3 | 7 | 5 | 6 | 7 | 2 | 4 | 3 | 5 | 1 |
| 6 | 6 | 6 | 6 | 6 | 6 | 3 | 3 | 3 | 3 | 3 | 3 | 5 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
|  | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 |
|  |  | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 5 | 5 |
|  |  |  |  | 5 | 5 | 5 | 1 | 1 | 1 | 1 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 1 |
| \* | \* | \* |  | \* |  | \* | \* |  |  |  | \* | \* | \* |  |  |  | \* | \* | \* |

=> Tổng cộng có 12 lỗi trang.

Giải bằng code:

Text

Description automatically generated

Hình 14: Kết quả giải ví dụ 3 bằng thuật toán OPT

* Soucre Code Của Chương Trình

Text

Description automatically generated

Hình 15: Soucre code chương trình từ dòng 0-68

Text

Description automatically generated

Hình 16: Soucre code chương trình từ dòng 69-129

Text

Description automatically generated

Hình 17: Soucre code chương trình từ dòng 130-195

Text

Description automatically generated

Hình 18: Soucre code chương trình từ dòng 196-264

# Section 6.5

## Task name 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:

Text

Description automatically generated

Hình 19: Chứng minh nghịch lý Belady

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

## Task name 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.