| | |
|---|---|
| **Name:** Lê Minh Nguyệt<br>**ID:** 21521211 | **Name:** Trần Trung Tín<br>**ID:** 21522679 |
| **Name:** Hồ Đức Trưởng<br>**ID:** 21522730 | **Name:** Nguyễn Công Nguyên<br>**ID:** 21521200 |
| **Class:** IT007.N11 | **Group Name:** Team |


# OPERATING SYSTEM
# LAB 06'S REPORT

## SUMMARY

| | | | | |
|---|---|---|---|---|
| | Ex 2. Mức độ hiệu quả và tính khả thi của 3 giải thuật | 2.1. Giải thuật bất khả thi nhất | Done | 41 |
| | | 2.2. Giải thuật phức tạp nhất | | |

**Self-scrores: 9,5**

## Section 6.4

***Thực hiện Menu theo yêu cầu Section 6.4:***

- Tiến hành khai báo chuỗi tham chiếu mặc định def_seq gồm 11 phần tử (8 chữ số mã số sinh viên và 2 chữ số 0, 0, 7).
- Mảng seq chứa chuỗi tham chiếu muốn áp dụng giải thuật, n là số phần tử của chuỗi.
- Khai báo biến frame_num chứa số khung trang, do người dùng nhập vào.
- Tạo menu lựa chọn chuỗi tham chiếu: 1 để chọn chuỗi mặc định, 2 để tự nhập chuỗi vào từ bàn phím.
- Dùng vòng lặp do…while… yêu cầu người dùng nhập lại cho đến khi thỏa mãn một trong hai lựa chọn 1 hoặc 2.

```
int main()
{
// default referenced sequence
    int def_seq[11] = {2, 1, 5, 2, 2, 6, 7, 9, 0, 0, 7};
    int seq[MAX], n, i;

    int frame_num;

    int choice_seq;

    printf("--- Page Replacement algorithm ---");
    printf("\n1. Default referenced sequence");
    printf("\n2. Manual input sequence");

    do
    {
        printf("\n-------------------------");
        printf("\nYour choice: ");
        scanf("%d", &choice_seq);
    } while(choice_seq!=1 && choice_seq!=2);
```

*Hình 1. Tạo chuỗi tham chiếu mặc định và menu lựa chọn chuỗi tham chiếu*

- Nếu người dùng chọn 2 (choice_seq==2), cho phép người dùng nhập chuỗi tham chiếu từ bàn phím. Cụ thể số lượng phần tử lưu vào biến n, các giá trị trong chuỗi lưu vào mảng seq[].
- Nếu người dùng chọn chuỗi mặc định, lấy chuỗi trong mảng def_seq[] lưu vào seq[] và n=11.
- Yêu cầu người dùng nhập số khung trang, lưu vào biến frame_num.
- Tạo menu lựa chọn một trong ba giải thuật FIFO, OPT, LRU.

```
if(choice_seq==2)
{
    printf("\nEnter the size of your input sequence: ");
    scanf("%d", &n);
    printf("\nEnter your input sequence: ");
    for (i=0; i<n; i++)
        scanf("%d", &seq[i]);
}
else
{
    n=11;
    for (i=0; i<n; i++)
        seq[i]=def_seq[i];
}

printf("\n--- Page Replacement algorithm ---");
printf("\nInput page frames: "); scanf("%d", &frame_num);
printf("\n--- Select algorithm ---");
printf("\n1. FIFO algorithm");
printf("\n2. OPT algorithm");
printf("\n3. LRU algorithm");
```

*Hình 2. Nhập chuỗi tham chiếu và số lượng khung trang từ bàn phím và tạo menu lựa chọn giải thuật*

- Tương tự với lựa chọn chuỗi bên trên, ta dùng vòng lặp do…while… yêu cầu người dùng nhập lại cho đến khi thỏa mãn một trong ba lựa chọn 1, 2 hoặc 3 (choice_algo nhận các giá trị trong khoảng [1, 3]).

```
int choice_algo;
do
{
    printf("\n-------------------------");
    printf("\nYour choice: ");
    scanf("%d", &choice_algo);
} while(choice_algo<1 && choice_algo>3);

printf("\n--- Page Replacement algorithm---\n");
switch (choice_algo) {
    case 1:
        FIFO(seq, n, frame_num);
        break;
    case 2:
        OPT(seq, n, frame_num);
        break;
    case 3:
        LRU(seq, n, frame_num);
        break;
    }
return 0;
}
```

*Hình 3. Lựa chọn một trong ba giải thuật và thực thi*

- Nếu người dùng chọn 1, ta thực hiện giải thuật FIFO bằng cách gọi hàm FIFO() với các tham số truyền vào là chuỗi tham chiếu seq, số lượng phần tử n, và số lượng khung trang frame_num.
- Tương tự với lựa chọn 2 (giải thuật OPT) và 3 (giải thuật LRU).

```
#include <stdio.h>

#define MAX 100

// seq[n]: chuoi tham chieu n trang
// frame_num: so luong khung trang do nguoi dung nhap
// available: bien danh dau trang co trong khung trang hay khong
// replace_i: bien luu vi tri khung trang se thay the, [0, frame_num-1],
// khoi tao = 0
// fault_num: so luong loi trang
// faults[n]: mang danh dau loi xay ra khi tham chieu trang nao
// total_frames[frame_num][n]: mang 2 chieu voi cot la cac khung trang tai
// thoi diem tham chieu 1 trang, hang la khung trang tai vi tri nhat dinh tai
// nhieu thoi diem tham chieu trang khac nhau
```

*Hình 4. Một số chú thích trong lưu đồ và code của ba giải thuật*

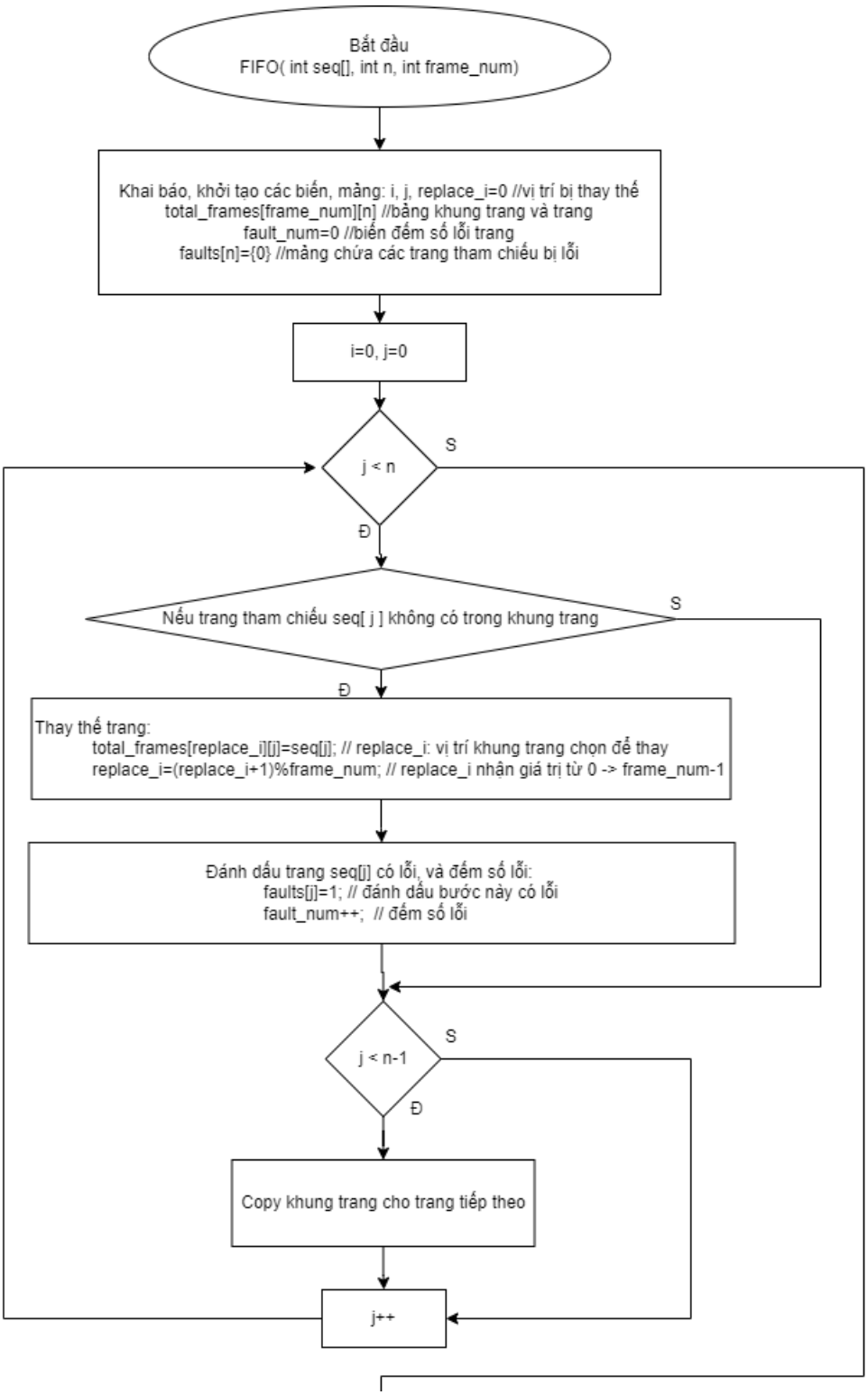
- Biến available đánh dấu trang có sẵn trong khung hay không, nhận giá trị 0 (không có sẵn, phải thay trang) hoặc 1 (có sẵn, không cần thay).
- Biến replace_i lưu vị trí khung trang sẽ bị thay ra, nhận giá trị trong khoảng [0, frame_num-1].
- Biến fault_num đếm số lượng lỗi trang.
- Mảng faults[n] đánh dấu mỗi trang có xảy ra lỗi hay không, mỗi phần tử ứng với mỗi trang, nhận giá trị 0 (không lỗi) hoặc 1 (có lỗi).
- Bảng total_frames[frame_num][n]:

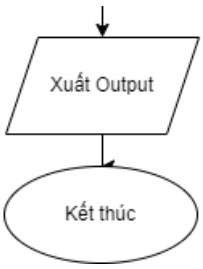
| Khung trang thứ i \ Chuỗi tham chiếu seq[j] | j=0 | j=1 | … | j=n-1 |
|---|---|---|---|---|
| i=0 | | | | |
| i=1 | | | | |
| … | | | | |
| i=frame_num-1 | | | | |

→ ***total_frames[i][j]***

## ***Ex 1.Giải thuật FIFO***

### 1.1. Vẽ lưu đồ giải thuật






*Hình 5. Lưu đồ giải thuật FIFO*

- Khởi tạo biến fault_num=0 và tất cả giá trị trong mảng faults[]=0, do ban đầu chưa có lỗi trang nào.
- replace_i khởi tạo=0: vị trí khung trang đầu tiên thêm vào.
- Ta thực hiện vòng lặp bắt đầu j=0 cho đến n-1 (j đi qua từng phần tử của chuỗi tham chiếu). Tại mỗi phần tử:
  + Tiến hành tìm trang tại vị trí j, xem có trong các khung trang hiện tại không. Nếu có, không thực hiện thay trang. Ngược lại thực hiện thay trang vào vị trí replace_i, sau đó tăng replace_i lên 1 đơn vị nhưng chỉ giới hạn trong khoảng [0, frame_num-1].
  + Tiến hành copy các khung trang hiện tại cho trang tiếp theo, nếu trang tiếp theo chưa phải trang cuối cùng (j<n-1).
  + Tăng j lên 1 đơn vị (chuyển đến trang tiếp theo).
  + Thực hiện vòng lặp cho đến khi j duyệt qua hết n phần tử trong chuỗi.
- Xuất ouput gồm bảng total_frames[][] và các lỗi trang.
- Kết thúc chương trình.

**1.2. Trình bày tính đúng đắn của lưu đồ bằng cách chạy tay ít nhất 01 test case tối thiểu 10 phần tử**

***a. Test case:***

INPUT

seq[10]={7,0,1,2,0,3,0,4,2,3}

n=10

frame_num=3

OUTPUT (Kết quả giải tay)

| **7** | **0** | **1** | **2** | **0** | **3** | **0** | **4** | **2** | **3** |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 7 | 7 | 2 | 2 | 2 | 2 | 4 | 4 | 4 |
| | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 2 | 2 |
| | | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 3 |
| * | * | * | * | | * | * | * | * | * |

Number of Page Fault: 9

***b. Chứng minh tính đúng đắn của lưu đồ bằng chạy tay test case trên:***

- Khai báo biến i, j, replace_i, fault_num

- Khởi tạo:
  + Mảng total_frames[3][10] =[[None*10]*3]
  + Mảng faults[10]=[0*10]
  + j=0, replace_i=0, fault_num=0
- Sử dụng vòng lặp cho đến khi j>=n (n=10):
  + Ở vòng lặp j=0 :
    - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][0] không chứa 7 (seq[0]).
    - total_frames[0][0]=seq[0]=7
    - replace_i=(0+1)%3=1
    - Đánh dấu trang j=0 có lỗi: faults[0]=1, tăng số lỗi fault_num=1
    - Ta kiểm tra thấy j<n-1(0<9) nên copy khung trang cho trang 1 tiếp theo: total_frames[i][1]=total_frames[i][0], i ∈ [0, 2]
    - Lúc này, total_frames[i][0]=total_frames[i][1]=[7,None,None], i ∈ [0, 2]
    - j=1
  + Ở vòng lặp j=1:
    - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][1] không chứa 0 (seq[1]).
    - total_frames[1][1]=seq[1]=0
    - replace_i=(1+1)%3=2
    - Đánh dấu trang j=1 có lỗi: faults[1]=1, tăng số lỗi fault_num=2
    - Ta kiểm tra thấy j<n-1(1<9) nên copy khung trang cho trang 2 tiếp theo: total_frames[i][2]=total_frames[i][1], i ∈ [0, 2]
    - Lúc này, total_frames[i][1]= total_frames[i][2]=[7,0,None], i ∈ [0, 2]
    - j=2
  + Ở vòng lặp j=2:
    - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][2] không chứa 1
    - total_frames[2][2]=seq[2]=1
    - replace_i=(2+1)%3=0
    - Đánh dấu trang j=2 có lỗi: faults[2]=1, tăng số lỗi fault_num=3
    - Ta kiểm tra thấy j<n-1(2<9) nên copy khung trang cho trang 3 tiếp theo: total_frames[i][3]=total_frames[i][2], i ∈ [0, 2]
    - Lúc này, total_frames[i][2]=total_frames[i][3]=[7,0,1], i ∈ [0, 2]
    - j=3
  + Ở vòng lặp j=3:
    - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][3] không chứa 2
    - total_frames[0][3]=seq[3]=2

- replace_i=(0+1)%3=1
- Đánh dấu trang j=3 có lỗi: faults[3]=1, tăng số lỗi fault_num=4
- Ta kiểm tra thấy j<n-1(3<9) nên copy khung trang cho trang 4 tiếp theo: total_frames[i][4]=total_frames[i][3], i ∈ [0, 2]
- Lúc này, total_frames[i][3]=total_frames[i][4]=[2,0,1], i ∈ [0, 2]
- j=4

+ Ở vòng lặp j=4:
  - Với i ∈ [0, 2], ta kiểm tra thấy total_frames[1][4]==0
  - Ta kiểm tra thấy j<n-1(4<9) nên copy khung trang cho trang 5 tiếp theo: total_frames[i][5]=total_frames[i][4], i ∈ [0, 2]
  - Lúc này, total_frames[i][4]=total_frames[i][5]=[2,0,1], i ∈ [0, 2]
  - j=5
+ Ở vòng lặp j=5:
  - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][5] không chứa 3
  - total_frames[1][5]=seq[5]=3
  - replace_i=(1+1)%3=2
  - Đánh dấu trang j=5 có lỗi: faults[5]=1, tăng số lỗi fault_num=5
  - Ta kiểm tra thấy j<n-1(5<9) nên copy khung trang cho trang 6 tiếp theo: total_frames[i][6]=total_frames[i][5], i ∈ [0, 2]
  - Lúc này, total_frames[i][5]=total_frames[i][6]=[2,3,1], i ∈ [0, 2]
  - j=6
+ Ở vòng lặp j=6:
  - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][6] không chứa 0
  - total_frames[2][6]=seq[6]=0
  - replace_i=(2+1)%3=0
  - Đánh dấu trang j=6 có lỗi: faults[6]=1, tăng số lỗi fault_num=6
  - Ta kiểm tra thấy j<n-1(6<9) nên copy khung trang cho trang 7 tiếp theo: total_frames[i][7]=total_frames[i][6], i ∈ [0, 2]
  - Lúc này, total_frames[i][6]=total_frames[i][7]=[2,3,0], i ∈ [0, 2]
  - j=7
+ Ở vòng lặp j=7:
  - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][7] không chứa 4
  - total_frames[0][7]=seq[7]=4
  - replace_i=(0+1)%3=1
  - Đánh dấu trang j=7 có lỗi: faults[7]=1, tăng số lỗi fault_num=7
  - Ta kiểm tra thấy j<n-1(7<9) nên copy khung trang cho trang 8 tiếp theo: total_frames[i][8]=total_frames[i][7], i ∈ [0, 2]
  - Lúc này, total_frames[i][7]=total_frames[i][8]=[4,3,0], i ∈ [0, 2]

- j=8

+ Ở vòng lặp j=8:
  - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][8] không chứa 2
  - total_frames[1][8]=seq[8]=2
  - replace_i=(1+1)%3=2
  - Đánh dấu trang j=8 có lỗi: faults[8]=1, tăng số lỗi fault_num=8
  - Ta kiểm tra thấy j<n-1(8<9) nên copy khung trang cho trang 9 tiếp theo: total_frames[i][9]=total_frames[i][8], i ∈ [0, 2]
  - Lúc này, total_frames[i][8]=total_frames[i][9]=[4,2,0], i ∈ [0, 2]
  - j=9
+ Ở vòng lặp j=9
  - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][9] không chứa 3
  - total_frames[2][9]=seq[9]=3
  - replace_i=(2+1)%3=0
  - Đánh dấu trang j=9 có lỗi: faults[9]=1, tăng số lỗi fault_num=9
  - Ta kiểm tra thấy j==n-1(9==9)
  - Lúc này, total_frames[i][9]=[4,2,3], i ∈ [0, 2]
  - j=10
+ Ở vòng lặp j=10 kiểm tra thấy j==n==10, thoát khỏi vòng lặp.

- Xuất output.
- Kết thúc chương trình.

***c. Minh họa output chạy tay trên lưu đồ:***

Number of Page Fault: 9 (fault_num)

| **7** | **0** | **1** | **2** | **0** | **3** | **0** | **4** | **2** | **3** |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 7 | 7 | 2 | 2 | 2 | 2 | 4 | 4 | 4 |
| None | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 2 | 2 |
| None | None | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 3 |
| **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** |

*Lưu ý*: Khi xuất ra màn hình, None được thay bằng khoảng trắng, và dòng cuối cùng của output (mảng faults[]) giá trị 1 được thay bằng "*", 0 được thay bằng khoảng trắng.

➢ ***Kết quả chạy tay trên lưu đồ giống với kết quả giải tay, chứng tỏ lưu đồ đúng.***

### 1.3. Thực hiện code cho giải thuật

- Khai báo và khởi tạo giá trị cho các biến replace_i, fault_num, mảng faults[].
- In chuỗi tham chiếu ban đầu seq[] ra màn hình.
- Khởi tạo tất cả ô bảng total_frames[][] với giá trị -1 (tương ứng với None trong phần 1.2).

```
void FIFO(int seq[], int n, int frame_num) {
    int i, j, available, replace_i=0, fault_num=0, faults[MAX];

    int total_frames[frame_num][n];
    for(j=0; j<n; j++) {
        printf("%d ", seq[j]); // in chuoi tham chieu ban dau
        faults[j]=0; // khoi tao ban dau khong trang nao co loi
    }
    printf("\n");

    for (i=0; i<frame_num; i++) {
        for (j=0; j<n; j++) total_frames[i][j]=-1;
    // khoi tao ban dau tat ca cac o trong bang = -1 (toan bo khung trang rong)
    }
```

*Hình 6. Thực hiện code cho giải thuật FIFO (1)*

- Ở mỗi vòng lặp j (mỗi trang trong chuỗi):
  + Khởi tạo giá trị biến available=0 (mặc định không tìm thấy trong khung trang).
  + Lặp qua các khung trang hiện tại, nếu tìm thấy gán available=1.
  + Sau khi tìm hết trong các khung trang, nếu available vẫn = 0 (không tìm thấy), ta tiến hành thay trang.
    - Gán trang hiện tại seq[j] cho ô total_frames[replace_i][j], khung trang tại vị trí replace_i trong cột trang hiện tại.
    - replace_i tăng lên 1, vẫn trong khoảng [0, frame_num-1]
    - Đánh dấu trang j có lỗi (faults[j]=1), tăng số lỗi thêm 1.
  + Copy các khung trang hiện tại cho trang kế tiếp.

```
    for(j=0; j<n; j++) {
    //moi lan tham chieu 1 trang, xem nhu chua tim thay trong khung trang
        available=0;
        // bat dau tim trong khung trang
        for(i=0; i<frame_num; i++) {
            if(seq[j]==total_frames[i][j]) available=1; // tim thay trang
        }

        if(available==0) { // khong tim thay -> thay trang
            total_frames[replace_i][j]=seq[j];
            replace_i=(replace_i+1)%frame_num;
            faults[j]=1; // danh dau buoc nay co loi
            fault_num++; // tang so loi
        }

        // copy khung trang tai thoi diem hien tai cho trang tiep theo
        if(j<n-1) {
            for(i=0; i<frame_num; i++)
                total_frames[i][j+1]=total_frames[i][j];
        }
    }
```

*Hình 7. Thực hiện code cho giải thuật FIFO (2)*

- Xuất ra màn hình các giá trị trong bảng total_frames[][], với giá trị bằng -1 (None/rỗng) xuất ra khoảng trắng thay thế.

- Xuất ra màn hình các đánh dấu lỗi trong mảng faults[], với giá trị 1 (có lỗi) xuất ra kí tự "*", với giá trị 0 xuất ra khoảng trắng.
- Xuất ra số lỗi trang lưu trong biến fault_num.

```
    // Xuat output
    // In bang khung trang
    for (i=0; i<frame_num; i++) {
        for(j=0; j<n; j++) {
            if(total_frames[i][j]!=-1)
                printf("%d ", total_frames[i][j]);
            else printf("  ");
        }
        printf("\n");
    }
    // In loi va so loi
    for(j=0; j<n; j++) {
        if(faults[j]==1)
            printf("* "); // in loi
        else printf("  ");
    }
    printf("\nNumber of Page Fault: %d\n", fault_num);
}
```

*Hình 8. Thực hiện code cho giải thuật FIFO (3)*

### 1.4. Trình bày tính đúng đắn của code bằng cách chạy ít nhất 03 test case

*Testcase mặc định:*

INPUT

n=11

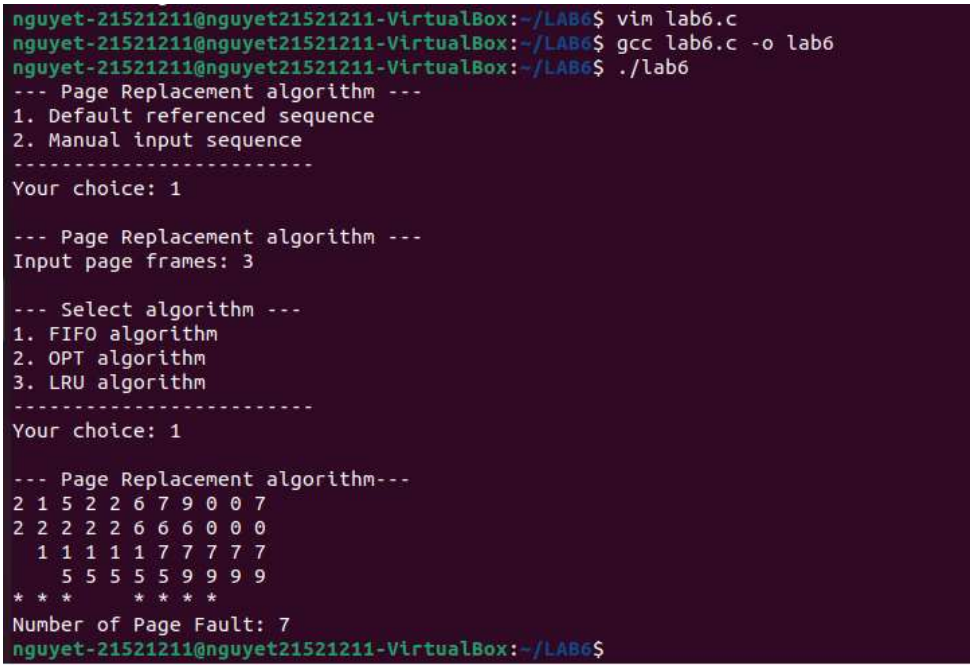
seq[]={2, 1, 5, 2, 2, 6, 7, 9, 0, 0, 7}

frame_num=3

OUTPUT (Kết quả giải tay)

| **2** | **1** | **5** | **2** | **2** | **6** | **7** | **9** | **0** | **0** | **7** |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 2 | 2 | 2 | 6 | 6 | 6 | 0 | 0 | 0 |
| | 1 | 1 | 1 | 1 | 1 | 7 | 7 | 7 | 7 | 7 |
| | | 5 | 5 | 5 | 5 | 5 | 9 | 9 | 9 | 9 |
| * | * | * | | | * | * | * | * | | |

Number of Page Fault: 7




*Hình 9. Chạy test case mặc định bằng code giải thuật FIFO*

➢ ***Kết quả thực hiện bằng code giống với kết quả giải tay.***

*Testcase nhập tay 1:*

INPUT

n=20

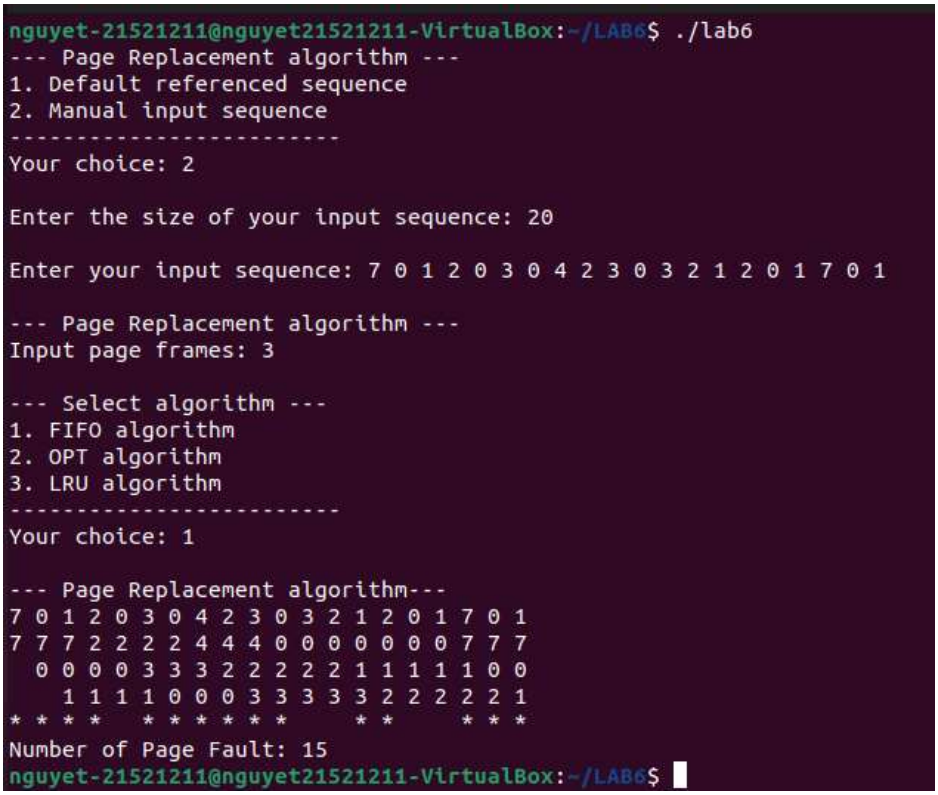
seq[]={7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1}

frame_num=3

OUTPUT (Kết quả giải tay)

| **7** | **0** | **1** | **2** | **0** | **3** | **0** | **4** | **2** | **3** | **0** | **3** | **2** | **1** | **2** | **0** | **1** | **7** | **0** | **1** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 7 | 7 | 2 | 2 | 2 | 2 | 4 | 4 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 7 | 7 |
| | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| | | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 1 |
| * | * | * | * | | * | * | * | * | * | * | | | * | * | | | * | * | * |

Number of Page Fault: 15




*Hình 10. Chạy test case nhập tay 1 bằng code giải thuật FIFO*

➢ ***Kết quả thực hiện bằng code giống với kết quả giải tay.***

*Testcase nhập tay 2:*

INPUT

n=20

seq[]={1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6}

frame_num=3

OUTPUT (Kết quả giải tay)

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

Number of Page Fault: 16

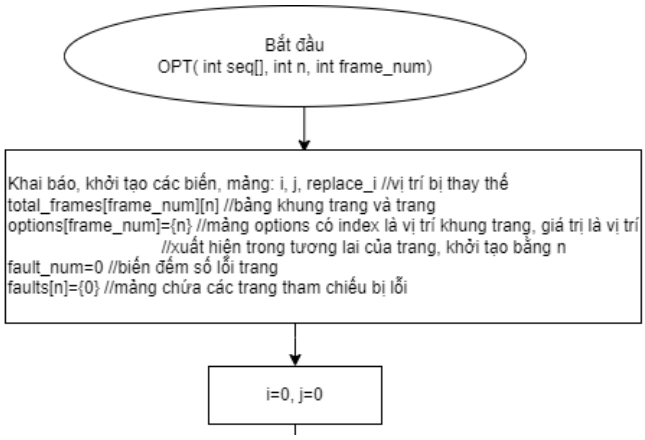
```
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB6$ ./lab6
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
-------------------------
Your choice: 2

Enter the size of your input sequence: 20

Enter your input sequence: 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

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

--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
-------------------------
Your choice: 1

--- Page Replacement algorithm---
1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6
1 1 1 4 4 4 4 6 6 6 6 3 3 3 3 2 2 2 2 6
  2 2 2 2 1 1 1 2 2 2 2 7 7 7 7 1 1 1 1
    3 3 3 3 5 5 5 1 1 1 1 6 6 6 6 6 3 3
* * * *   * * * * *   * * *   * *   * *
Number of Page Fault: 16
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB6$
```

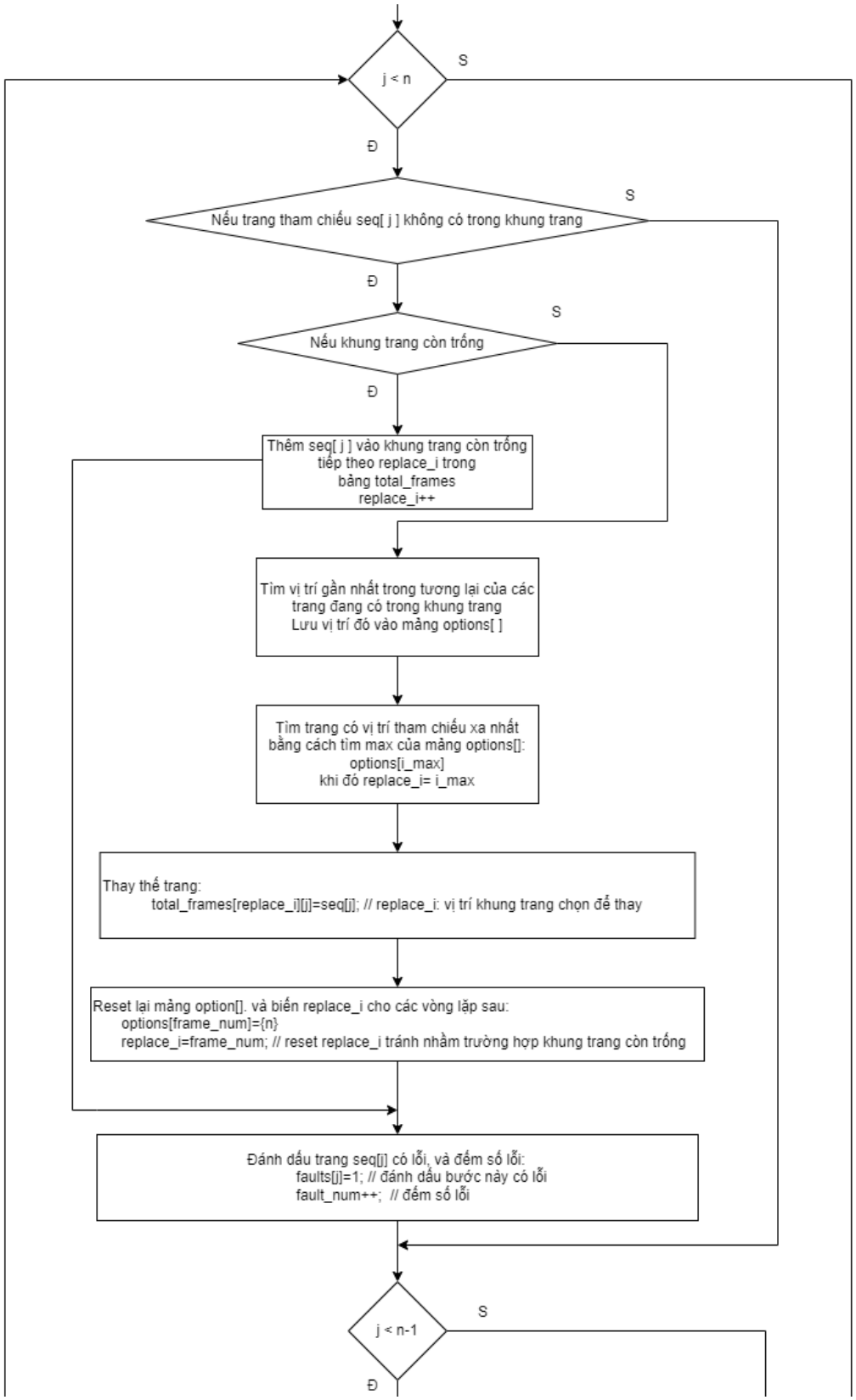
*Hình 11. Chạy test case nhập tay 2 bằng code giải thuật FIFO*

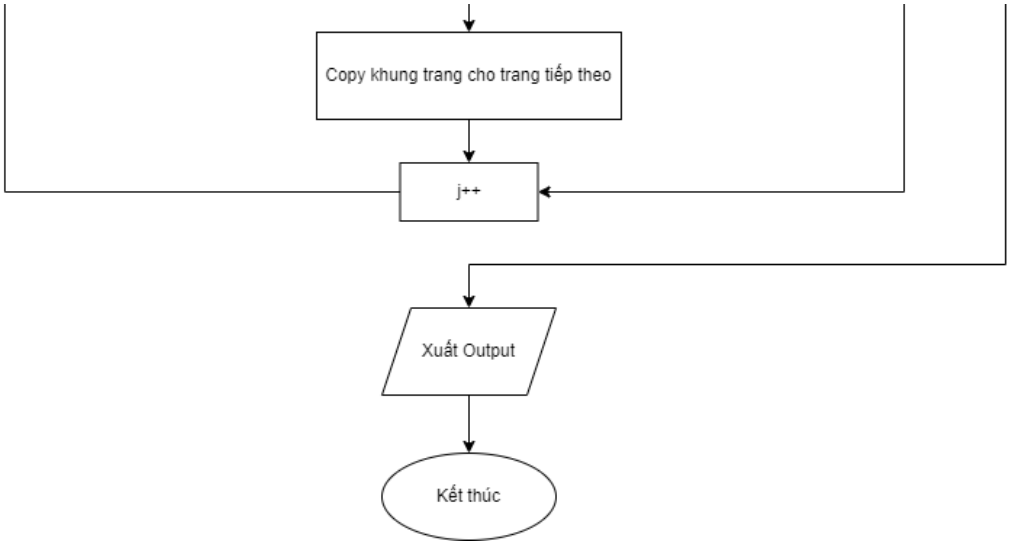
- ***Kết quả thực hiện bằng code giống với kết quả giải tay.***
- ***Cả 3 test case đều cho kết quả giống với kết quả giải tay. Kết luận code đúng.***

## ***Ex 2.Giải thuật OPT***

### 2.1. Vẽ lưu đồ giải thuật




j < n
S
Đ
Nếu trang tham chiếu seq[ j ] không có trong khung trang
S
Đ
Nếu khung trang còn trống
S
Đ
Thêm seq[ j ] vào khung trang còn trống
tiếp theo replace_i trong
bảng total_frames
replace_i++
Tìm vị trí gần nhất trong tương lai của các
trang đang có trong khung trang
Lưu vị trí đó vào mảng options[ ]
Tìm trang có vị trí tham chiếu xa nhất
bằng cách tìm max của mảng options[]:
options[i_max]
khi đó replace_i= i_max
Thay thế trang:
total_frames[replace_i][j]=seq[j]; // replace_i: vị trí khung trang chọn để thay
Reset lại mảng option[]. và biến replace_i cho các vòng lặp sau:
options[frame_num]={n}
replace_i=frame_num; // reset replace_i tránh nhầm trường hợp khung trang còn trống
Đánh dấu trang seq[j] có lỗi, và đếm số lỗi:
faults[j]=1; // đánh dấu bước này có lỗi
fault_num++; // đếm số lỗi
j < n-1
S
Đ




*Hình 12. Lưu đồ giải thuật OPT*

- Khởi tạo biến fault_num=0 và tất cả giá trị trong mảng faults[]=0, do ban đầu chưa có lỗi trang nào.
- replace_i khởi tạo=0: vị trí khung trang đầu tiên thêm vào.
- Khởi tạo mảng options[frame_num] là mảng chứa các lựa chọn để thay trang, lưu vị trí ***tương lai*** gần nhất trong chuỗi tham chiếu của các trang đang chiếm chỗ trong khung trang. Từ đó, lựa chọn được trang có vị trí xa nhất (index lớn nhất). Giá trị ban đầu của tất cả các phần tử trong options[] là n (nếu trang không được tham chiếu trong tương lai, giá trị của nó trong options[] vẫn = n, lớn hơn tất cả các trang được tham chiếu lần nữa (luôn <= n-1)).
- *Ví dụ: Ta có chuỗi tham chiếu {…,6,7,4,..,5}*
  *Khi trang thứ 4 (trang số 7) được tham chiếu, khung trang đầy, ta tiến hành lưu những vị trí tham chiếu tương lai của các trang trong khung trang (4, 5, 6) vào mảng options[]. Ban đầu, khởi tạo options[]=[n, n, n]. Ta giả sử trang số 6 không được tham chiếu nữa. Sau khi tìm trong chuỗi, options[]=[5, n-1, n] tương ứng với khung trang 0, 1, 2 và các trang số 4, 5, 6. Khi đó, tìm được giá trị lớn nhất trong options[] là n, vị trí khung trang chọn để thay tương ứng là vị trí 2 (thay trang số 6 ra).*

| *index* | **…** | **3** | **4** | **5** | **…** | **n-1** |
|---|---|---|---|---|---|---|
| | **…** | **6** | **7** | **4** | **…** | **5** |
| **0** | | 4 | 4 | | | |
| **1** | | 5 | 5 | | | |
| **2** | | 6 | 7 | | | |

- Ta thực hiện vòng lặp bắt đầu j=0 cho đến n-1 (j đi qua từng phần tử của chuỗi tham chiếu). Tại mỗi phần tử:
  + Tiến hành tìm trang tại vị trí j, xem có trong các khung trang hiện tại không. Nếu có, không thực hiện thay trang. Ngược lại thực hiện thay trang:

- Kiểm tra còn khung trang trống không. Nếu có, thêm trực tiếp vào vị trí còn trống nhỏ nhất (replace_i).
- Nếu không còn khung trang trống, thực hiện tìm vị trí tham chiếu gần nhất trong tương lai của từng trang trong khung trang hiện tại, lưu vào mảng options[]. Trong mảng options[], tìm giá trị lớn nhất (vị trí tham chiếu xa nhất), chọn khung trang tương ứng với giá trị này trong options[] để thay (replace_i=i_max). Reset mảng options[]={n}, replace_i=frame_num để tránh nhầm trường hợp khung trang còn trống.
- Đánh dấu trang thứ j có lỗi, tăng lỗi trang thêm 1.

+ Tiến hành copy các khung trang hiện tại cho trang tiếp theo, nếu trang tiếp theo chưa phải trang cuối cùng (j<n-1).

+ Tăng j lên 1 đơn vị (chuyển đến trang tiếp theo).

+ Thực hiện vòng lặp cho đến khi j duyệt qua hết n phần tử trong chuỗi.

- Xuất ouput gồm bảng total_frames[][] và các lỗi trang.
- Kết thúc chương trình.

**2.2. Trình bày tính đúng đắn của lưu đồ bằng cách chạy tay ít nhất 01 test case tối thiểu 10 phần tử**

***a. Test case:***

INPUT

seq[10]={7,0,1,2,0,3,0,4,2,3}

n=10

frame_num=3

OUTPUT (Kết quả giải tay)

| **7** | **0** | **1** | **2** | **0** | **3** | **0** | **4** | **2** | **3** |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 7 | 7 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 4 | 4 |
| | | 1 | 1 | 1 | 3 | 3 | 3 | 3 | 3 |
| * | * | * | * | | * | | * | | |

Number of Page Fault: 6

***b. Chứng minh tính đúng đắn của lưu đồ bằng chạy tay test case trên:***

- Khai báo biến i, j, replace_i, fault_num
- Khởi tạo:
  + Mảng total_frames[3][10] =[[None*10]*3]
  + Mảng faults[10]=[0*10]
  + Mảng options[3]=[10*3]
  + j=0, replace_i=0, fault_num=0
- Sử dụng vòng lặp cho đến khi j>=n (n=10):

+ Ở vòng lặp j=0 :
  - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][0] không chứa 7 (seq[0]).
  - Kiểm tra tiếp thấy khung trang còn trống (replace_i=0<3), vì vậy total_frames[0][0]=seq[0]=7, replace_i=1
  - Đánh dấu trang j=0 có lỗi: faults[0]=1, tăng số lỗi fault_num=1
  - Ta kiểm tra thấy j<n-1(0<9) nên copy khung trang cho trang 1 tiếp theo: total_frames[i][1]=total_frames[i][0], i ∈ [0, 2]
  - Lúc này, total_frames[i][0]=total_frames[i][1]=[7,None,None], i ∈ [0, 2]
  - j=1
+ Ở vòng lặp j=1:
  - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][1] không chứa 0 (seq[1]).
  - Kiểm tra tiếp thấy khung trang còn trống (replace_i=1<3), vì vậy total_frames[1][1]=seq[1]=0, replace_i=2
  - Đánh dấu trang j=1 có lỗi: faults[1]=1, tăng số lỗi fault_num=2
  - Ta kiểm tra thấy j<n-1(1<9) nên copy khung trang cho trang 2 tiếp theo: total_frames[i][2]=total_frames[i][1], i ∈ [0, 2]
  - Lúc này, total_frames[i][1]= total_frames[i][2]=[7,0,None], i ∈ [0, 2]
  - j=2
+ Ở vòng lặp j=2:
  - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][2] không chứa 1
  - Kiểm tra tiếp thấy khung trang còn trống (replace_i=2<3), vì vậy total_frames[2][2]=seq[2]=1, replace_i=3
  - Đánh dấu trang j=2 có lỗi: faults[2]=1, tăng số lỗi fault_num=3
  - Ta kiểm tra thấy j<n-1(2<9) nên copy khung trang cho trang 3 tiếp theo: total_frames[i][3]=total_frames[i][2], i ∈ [0, 2]
  - Lúc này, total_frames[i][2]=total_frames[i][3]=[7,0,1], i ∈ [0, 2]
  - j=3
+ Ở vòng lặp j=3:
  - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][3] không chứa 2
  - Kiểm tra tiếp thấy trong khung trang hết chỗ (replace_i==3)
    - Tìm vị trí gần nhất trong tương lai của các trang đang có trong khung trang total_frames[i][3], i ∈ [0, 2] và lưu vào mảng options[]. Mảng options[]=[10,4,10] do trang 0 (khung trang 1) được tham chiếu lần nữa tại vị trí thứ 4 trong chuỗi, trang 7 và 1 không được tham chiếu nữa.

        - Tìm vị trí của phần tử lớn nhất trong options[] (phần tử 10 ở vị trí 0), lưu vị trí đó vào replace_i(replace_i=0): total_frames[0][3]=seq[3]=2
        - Reset options[]=[10, 10, 10], replace_i=frame_num=3
    - Đánh dấu trang j=3 có lỗi: faults[3]=1, tăng số lỗi fault_num=4
    - Ta kiểm tra thấy j<n-1(3<9) nên copy khung trang cho trang 4 tiếp theo: total_frames[i][4]=total_frames[i][3], i ∈ [0, 2]
    - Lúc này, total_frames[i][3]=total_frames[i][4]=[2,0,1], i ∈ [0, 2]
    - j=4
+ Ở vòng lặp j=4:
    - Với i ∈ [0, 2], ta kiểm tra thấy total_frames[1][4]==0
    - Ta kiểm tra thấy j<n-1(4<9) nên copy khung trang cho trang 5 tiếp theo: total_frames[i][5]=total_frames[i][4], i ∈ [0, 2]
    - Lúc này, total_frames[i][4]=total_frames[i][5]=[2,0,1], i ∈ [0, 2]
    - j=5
+ Ở vòng lặp j=5:
    - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][5] không chứa 3
    - Kiểm tra tiếp thấy trong khung trang hết chỗ (replace_i==3)
        - Tìm vị trí gần nhất trong tương lai của các trang đang có trong khung trang total_frames[i][5], i ∈ [0, 2] và lưu vào mảng options[]. Mảng options[]=[8,6,10].
        - Tìm vị trí của phần tử lớn nhất trong options[] (phần tử 10 ở vị trí 2), lưu vị trí đó vào replace_i(replace_i=2): total_frames[2][5]=seq[5]=3
        - Reset options[]=[10, 10, 10], replace_i=frame_num=3
    - Đánh dấu trang j=5 có lỗi: faults[5]=1, tăng số lỗi fault_num=5
    - Ta kiểm tra thấy j<n-1(5<9) nên copy khung trang cho trang 6 tiếp theo: total_frames[i][6]=total_frames[i][5], i ∈ [0, 2]
    - Lúc này, total_frames[i][5]=total_frames[i][6]=[2,0,3], i ∈ [0, 2]
    - j=6
+ Ở vòng lặp j=6:
    - Với i ∈ [0, 2], ta kiểm tra thấy total_frames[1][6]==0
    - Ta kiểm tra thấy j<n-1(6<9) nên copy khung trang cho trang 7 tiếp theo: total_frames[i][7]=total_frames[i][6], i ∈ [0, 2]
    - Lúc này, total_frames[i][6]=total_frames[i][7]=[2,0,3], i ∈ [0, 2]
    - j=7
+ Ở vòng lặp j=7:
    - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][7] không chứa 4
    - Kiểm tra tiếp thấy trong khung trang hết chỗ (replace_i==3)

- Tìm vị trí gần nhất trong tương lai của các trang đang có trong khung trang total_frames[i][5], i ∈ [0, 2] và lưu vào mảng options[]. Mảng options[]=[8,10,9].
- Tìm vị trí của phần tử lớn nhất trong options[] (phần tử 10 ở vị trí 1), lưu vị trí đó vào replace_i(replace_i=1): total_frames[1][7]=seq[7]=4
- Reset options[]=[10, 10, 10], replace_i=frame_num=3

• Đánh dấu trang j=7 có lỗi: faults[7]=1, tăng số lỗi fault_num=6
• Ta kiểm tra thấy j<n-1(7<9) nên copy khung trang cho trang 8 tiếp theo: total_frames[i][8]=total_frames[i][7], i ∈ [0, 2]
• Lúc này, total_frames[i][7]=total_frames[i][8]=[2,4,3], i ∈ [0, 2]
• j=8

+ Ở vòng lặp j=8:
• Với i ∈ [0, 2], ta kiểm tra thấy total_frames[0][8]==2
• Ta kiểm tra thấy j<n-1(8<9) nên copy khung trang cho trang 9 tiếp theo: total_frames[i][9]=total_frames[i][8], i ∈ [0, 2]
• Lúc này, total_frames[i][8]=total_frames[i][9]=[2,4,3], i ∈ [0, 2]
• j=9

+ Ở vòng lặp j=9
• Với i ∈ [0, 2], ta kiểm tra thấy total_frames[2][9]==3
• Ta kiểm tra thấy j==n-1(9==9)
• Lúc này, total_frames[i][9]=[2,4,3], i ∈ [0, 2]
• j=10

+ Ở vòng lặp j=10 kiểm tra thấy j==n==10, thoát khỏi vòng lặp.

- Xuất output.
- Kết thúc chương trình.

***c. Minh họa output chạy tay trên lưu đồ:***

Number of Page Fault: 6 (fault_num)

| **7** | **0** | **1** | **2** | **0** | **3** | **0** | **4** | **2** | **3** |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 7 | 7 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| None | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 4 | 4 |
| None | None | 1 | 1 | 1 | 3 | 3 | 3 | 3 | 3 |
| **1** | **1** | **1** | **1** | **0** | **1** | **0** | **1** | **0** | **0** |

<u>*Lưu ý*</u>: Khi xuất ra màn hình, None được thay bằng khoảng trắng, và dòng cuối cùng của output (mảng faults[]) giá trị 1 được thay bằng "*", 0 được thay bằng khoảng trắng.

➢ ***Kết quả chạy tay trên lưu đồ giống với kết quả giải tay, chứng tỏ lưu đồ đúng.***

## 2.3. Thực hiện code cho giải thuật

```
// options[frame_num]: mang lua chon thay the trang,
// luu vi tri trong chuoi tham chieu cua cac trang tuong ung trong khung trang
hien tai

void OPT(int seq[], int n, int frame_num) {
    int i, j, k, available, replace_i=0, fault_num=0, faults[MAX];
    int total_frames[frame_num][n];

    int options[frame_num];

    for(j=0; j<n; j++) {
        printf("%d ", seq[j]);
        faults[j]=0;
    }
    printf("\n");

    for (i=0; i<frame_num; i++) {
        for (j=0; j<n; j++) total_frames[i][j]=-1;
        options[i]=n;
        // gia tri khoi tao = n (trang khong duoc tham chieu trong tuong lai)
    }
```

*Hình 13. Thực hiện code cho giải thuật OPT (1)*

- Khai báo và khởi tạo giá trị cho các biến replace_i, fault_num, mảng faults[].
- In chuỗi tham chiếu ban đầu seq[] ra màn hình.
- Khởi tạo tất cả ô bảng total_frames[][] với giá trị -1 (tương ứng với None trong phần 2.2). options[]={n}

```
    for(j=0; j<n; j++) {
        available=0;

        for(i=0; i<frame_num; i++) {
            if(seq[j]==total_frames[i][j]) available=1;
        }

        if(available==0) {
          // truong hop con trong khung trang thi chi can them vao
            if(replace_i<frame_num)
                total_frames[replace_i++][j]=seq[j];
// truong hop khong con trong khung trang -> tim trang tham chieu xa nhat trong
 tuong lai
            else {
                for(i=0; i<frame_num; i++) {
                    for(k=j+1; k<n; k++)
                        if(total_frames[i][j]==seq[k]) {
                            options[i]=k;
                            break;
                         // dung khi tim thay vi tri dau tien (gan nhat)
                        }
                }
```

*Hình 14. Thực hiện code cho giải thuật OPT (2)*

- Ở mỗi vòng lặp j (mỗi trang trong chuỗi):

+ Khởi tạo giá trị biến available=0 (mặc định không tìm thấy trong khung trang).
+ Lặp qua các khung trang hiện tại, nếu tìm thấy gán available=1.
+ Sau khi tìm hết trong các khung trang, nếu available vẫn = 0 (không tìm thấy), ta tiến hành thay trang.

- Kiểm tra khung trang còn trống (replace_i<frame_num). Nếu còn trống thêm trực tiếp vào vị trí replace_i, replace_i tăng lên 1. Nếu không còn trống:
  - Đối với mỗi trang trong khung trang i ∈ [0, frame_num-1], lặp qua tất cả các trang tương lai trong trong chuỗi (từ vị trí j+1 trở đi), tìm thấy tại vị trí nào thì lưu vào options[i]. Chỉ tìm vị trí gần nhất, dùng break dừng ngay khi tìm thấy vị trí đầu tiên.

```
            k=0;
    // k: bien trung gian de tim vi tri xa nhat (gia tri max cua options)
            for(i=0; i<frame_num; i++) {
                if(options[i]>k) {
                    k=options[i];
                    replace_i=i;
    // replace_i luu vi tri khung trang chon de thay = index của options
                }
            }

            for(i=0; i<frame_num; i++) options[i]=n;
            // reset options cho trang tiep theo

            total_frames[replace_i][j]=seq[j];
            // thay trang vao vi tri replace_i

            replace_i=frame_num;
            // reset replace_i tranh nham truong hop khung trang con trong
        }
        // danh dau va dem loi
        faults[j]=1;
        fault_num++;
    }
```

*Hình 15. Thực hiện code cho giải thuật OPT (3)*

  - Trong mảng options[] tìm phần tử có giá trị lớn nhất, lưu index của nó vào replace_i.
  - Thay trang tại vị trí replace_i: total_frames[replace_i][j]=seq[j]
  - Reset mảng options[]={n} và replace_i=frame_num.
- Đánh dấu trang j có lỗi (faults[j]=1), tăng số lỗi thêm 1.

+ Copy các khung trang hiện tại cho trang kế tiếp.

- Xuất ra màn hình các giá trị trong bảng total_frames[][], với giá trị bằng -1 (None/rỗng) xuất ra khoảng trắng thay thế.
- Xuất ra màn hình các đánh dấu lỗi trong mảng faults[], với giá trị 1 (có lỗi) xuất ra kí tự "*", với giá trị 0 xuất ra khoảng trắng.

- Xuất ra số lỗi trang lưu trong biến fault_num.

```
        // copy khung trang cho trang tiep theo
        if(j<n-1) {
            for(i=0; i<frame_num; i++)
                total_frames[i][j+1]=total_frames[i][j];
        }
    }
    // Xuat output
    for (i=0; i<frame_num; i++) {
        for(j=0; j<n; j++) {
            if(total_frames[i][j]!=-1)
                printf("%d ", total_frames[i][j]);
            else printf("  ");
        }
        printf("\n");
    }

    for(j=0; j<n; j++) {
        if(faults[j]==1)
            printf("* ");
        else printf("  ");
    }
    printf("\nNumber of Page Fault: %d\n", fault_num);
}
```

*Hình 16. Thực hiện code cho giải thuật OPT (4)*

## 2.4. Trình bày tính đúng đắn của code bằng cách chạy ít nhất 03 test case

*Testcase mặc định:*

INPUT

n=11

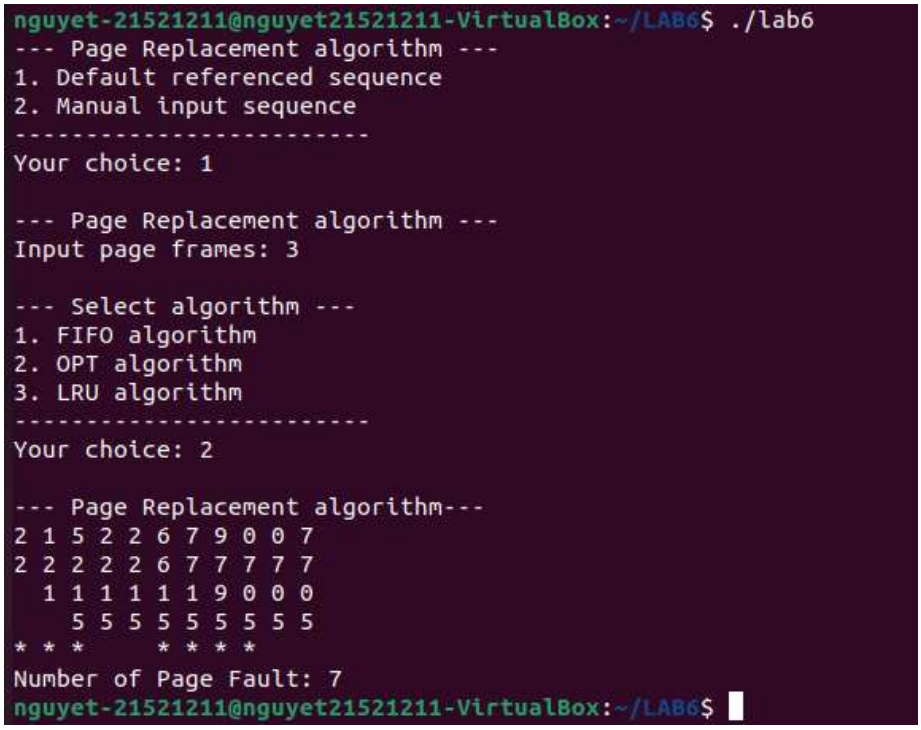
seq[]={2, 1, 5, 2, 2, 6, 7, 9, 0, 0, 7}

frame_num=3

OUTPUT (Kết quả giải tay)

| **2** | **1** | **5** | **2** | **2** | **6** | **7** | **9** | **0** | **0** | **7** |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 2 | 2 | 2 | 6 | 7 | 7 | 7 | 7 | 7 |
| | 1 | 1 | 1 | 1 | 1 | 1 | 9 | 0 | 0 | 0 |
| | | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| * | * | * | | | * | * | * | * | | |

Number of Page Fault: 7




*Hình 17. Chạy test case mặc định bằng code giải thuật OPT*

➢ ***Kết quả thực hiện bằng code giống với kết quả giải tay.***

*Testcase nhập tay 1:*

INPUT

n=20

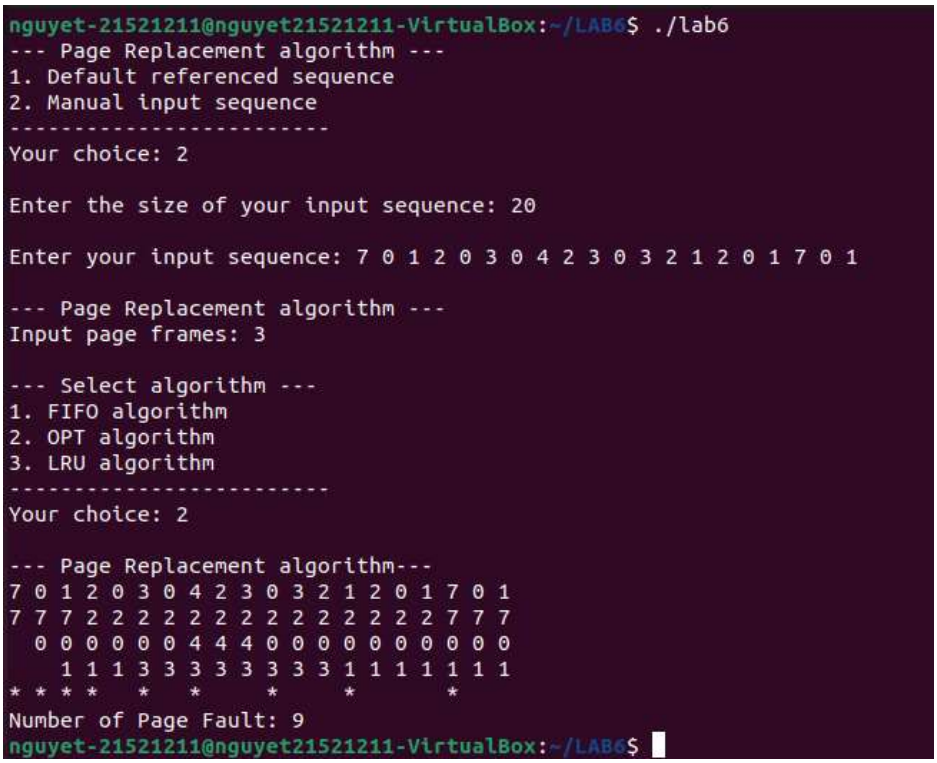
seq[]={7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1}

frame_num=3

OUTPUT (Kết quả giải tay)

| **7** | **0** | **1** | **2** | **0** | **3** | **0** | **4** | **2** | **3** | **0** | **3** | **2** | **1** | **2** | **0** | **1** | **7** | **0** | **1** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 7 | 7 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 7 | 7 | 7 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 4 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 1 | 1 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| * | * | * | * | | * | | * | | | * | | | * | | | | * | | |

Number of Page Fault: 9




*Hình 18. Chạy test case nhập tay 1 bằng code giải thuật OPT*

➢ ***Kết quả thực hiện bằng code giống với kết quả giải tay.***

*Testcase nhập tay 2:*

INPUT

n=20

seq[]={1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6}

frame_num=3

OUTPUT (Kết quả giải 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 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 6 |
| | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 7 | 7 | 7 | 2 | 2 | 2 | 2 | 2 |
| | | 3 | 4 | 4 | 4 | 5 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 1 | 1 | 1 | 1 |
| * | * | * | * | | | * | * | | | | * | * | | | * | * | | | * |

Number of Page Fault: 11

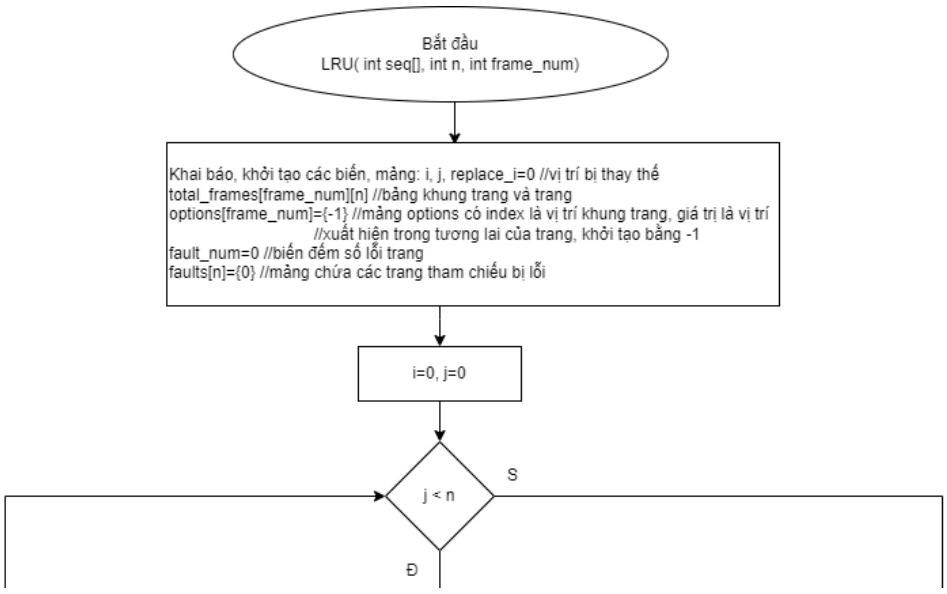
```
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB6$ ./lab6
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
-------------------------
Your choice: 2

Enter the size of your input sequence: 20

Enter your input sequence: 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

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

--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
-------------------------
Your choice: 2

--- Page Replacement algorithm---
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 3 3 3 3 3 3 3 3 6
  2 2 2 2 2 2 2 2 2 2 2 7 7 7 2 2 2 2 2
    3 4 4 4 5 6 6 6 6 6 6 6 6 6 1 1 1 1
* * * *     * *       * *     * *     *
Number of Page Fault: 11
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB6$
```

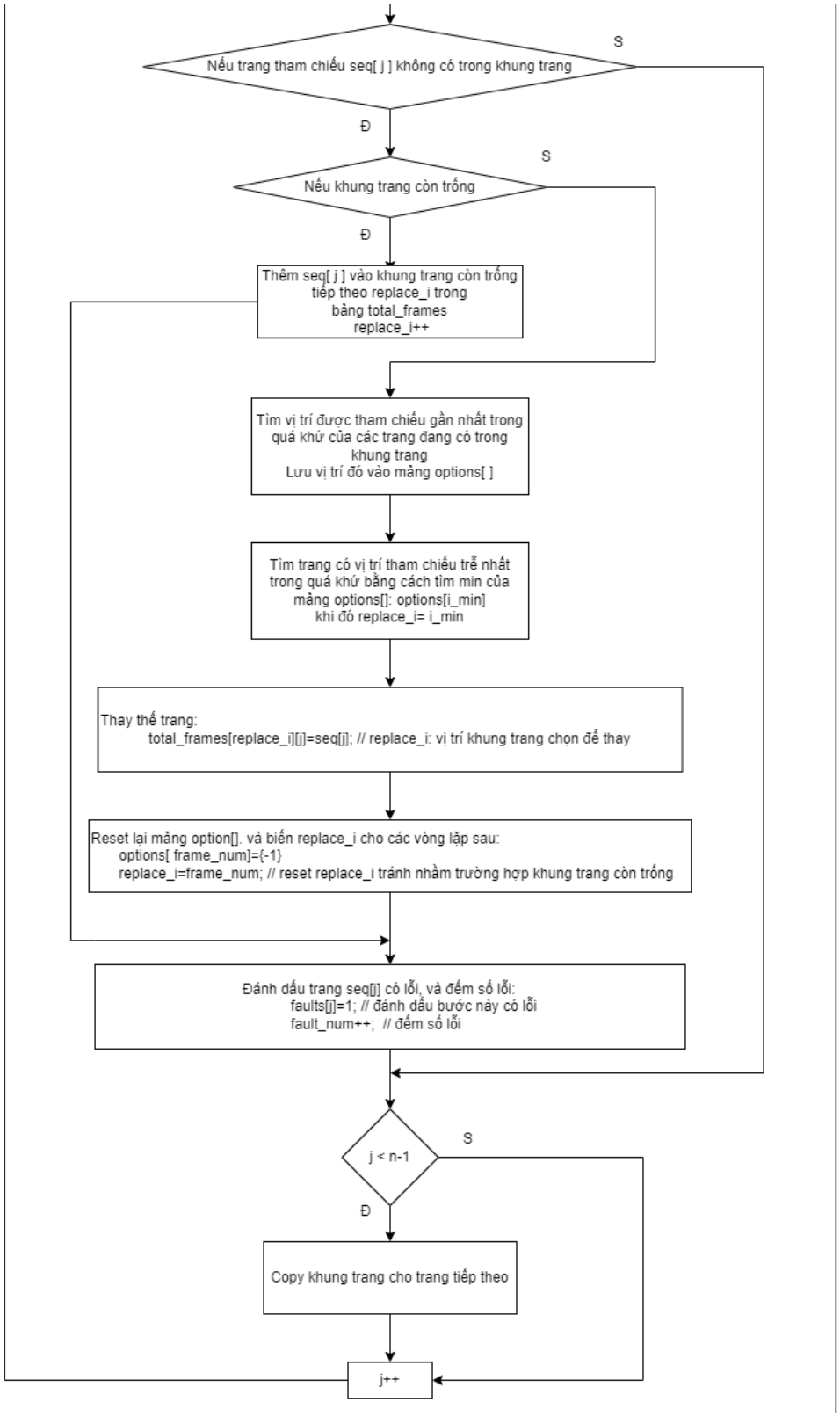
*Hình 19. Chạy test case nhập tay 2 bằng code giải thuật OPT*

- ***Kết quả thực hiện bằng code giống với kết quả giải tay.***
- ***Cả 3 test case đều cho kết quả giống với kết quả giải tay. Kết luận code đúng.***

## ***Ex 3.Giải thuật LRU***

### 3.1. Vẽ lưu đồ giải thuật




Nếu trang tham chiếu seq[ j ] không có trong khung trang
S
Đ
Nếu khung trang còn trống
S
Đ
Thêm seq[ j ] vào khung trang còn trống tiếp theo replace_i trong bảng total_frames
replace_i++
Tìm vị trí được tham chiếu gần nhất trong quá khứ của các trang đang có trong khung trang
Lưu vị trí đó vào mảng options[ ]
Tìm trang có vị trí tham chiếu trễ nhất trong quá khứ bằng cách tìm min của mảng options[]: options[i_min]
khi đó replace_i= i_min
Thay thế trang:
total_frames[replace_i][j]=seq[j]; // replace_i: vị trí khung trang chọn để thay
Reset lại mảng option[]. và biến replace_i cho các vòng lặp sau:
options[ frame_num]={-1}
replace_i=frame_num; // reset replace_i tránh nhầm trường hợp khung trang còn trống
Đánh dấu trang seq[j] có lỗi, và đếm số lỗi:
faults[j]=1; // đánh dấu bước này có lỗi
fault_num++; // đếm số lỗi
j < n-1
S
Đ
Copy khung trang cho trang tiếp theo
j++


*Hình 20. Lưu đồ giải thuật LRU*

- Khởi tạo biến fault_num=0 và tất cả giá trị trong mảng faults[]=0, do ban đầu chưa có lỗi trang nào.
- replace_i khởi tạo=0: vị trí khung trang đầu tiên thêm vào.
- Khởi tạo mảng options[frame_num] là mảng chứa các lựa chọn để thay trang, lưu vị trí ***quá khứ*** gần nhất trong chuỗi tham chiếu của các trang đang chiếm chỗ trong khung trang. Từ đó, lựa chọn được trang có thời điểm tham chiếu nhỏ nhất (index trong chuỗi tham chiếu nhỏ nhất). Giá trị ban đầu của tất cả các phần tử trong options[] là -1.
- Ta thực hiện vòng lặp bắt đầu j=0 cho đến n-1 (j đi qua từng phần tử của chuỗi tham chiếu). Tại mỗi phần tử:

  + Tiến hành tìm trang tại vị trí j, xem có trong các khung trang hiện tại không. Nếu có, không thực hiện thay trang. Ngược lại thực hiện thay trang:
    - Kiểm tra còn khung trang trống không. Nếu có, thêm trực tiếp vào vị trí còn trống nhỏ nhất (replace_i).
    - Nếu không còn khung trang trống, thực hiện tìm vị trí tham chiếu gần nhất trong quá khứ của từng trang trong khung trang hiện tại, lưu vào mảng options[]. Trong mảng options[], tìm giá trị nhỏ nhất (thời điểm tham chiếu nhỏ nhất), chọn khung trang tương ứng với giá trị này trong options[] để thay (replace_i=i_min). Reset mảng options[]={-1}, replace_i=frame_num để tránh nhầm trường hợp khung trang còn trống.
    - Đánh dấu trang thứ j có lỗi, tăng lỗi trang thêm 1.

  + Tiến hành copy các khung trang hiện tại cho trang tiếp theo, nếu trang tiếp theo chưa phải trang cuối cùng (j<n-1).

  + Tăng j lên 1 đơn vị (chuyển đến trang tiếp theo).

  + Thực hiện vòng lặp cho đến khi j duyệt qua hết n phần tử trong chuỗi.
- Xuất ouput gồm bảng total_frames[][] và các lỗi trang.
- Kết thúc chương trình.

**3.2. Trình bày tính đúng đắn của lưu đồ bằng cách chạy tay ít nhất 01 test case tối thiểu 10 phần tử**

***a. Test case:***

INPUT

seq[10]={7,0,1,2,0,3,0,4,2,3}

n=10

frame_num=3

OUTPUT (Kết quả giải tay)

| **7** | **0** | **1** | **2** | **0** | **3** | **0** | **4** | **2** | **3** |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 7 | 7 | 2 | 2 | 2 | 2 | 4 | 4 | 4 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| | | 1 | 1 | 1 | 3 | 3 | 3 | 2 | 2 |
| * | * | * | * | | * | | * | * | * |

Number of Page Fault: 8

***b. Chứng minh tính đúng đắn của lưu đồ bằng chạy tay test case trên:***

- Khai báo biến i, j, replace_i, fault_num
- Khởi tạo:
  - Mảng total_frames[3][10] =[[None*10]*3]
  - Mảng faults[10]=[0*10]
  - Mảng options[3]=[(-1)*3]
  - j=0, replace_i=0, fault_num=0
- Sử dụng vòng lặp cho đến khi j>=n (n=10):
  - Ở vòng lặp j=0 :
    - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][0] không chứa 7 (seq[0]).
    - Kiểm tra tiếp thấy khung trang còn trống (replace_i=0<3), vì vậy total_frames[0][0]=seq[0]=7, replace_i=1
    - Đánh dấu trang j=0 có lỗi: faults[0]=1, tăng số lỗi fault_num=1
    - Ta kiểm tra thấy j<n-1(0<9) nên copy khung trang cho trang 1 tiếp theo: total_frames[i][1]=total_frames[i][0], i ∈ [0, 2]
    - Lúc này, total_frames[i][0]=total_frames[i][1]=[7,None,None], i ∈ [0, 2]
    - j=1
  - Ở vòng lặp j=1:
    - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][1] không chứa 0 (seq[1]).
    - Kiểm tra tiếp thấy khung trang còn trống (replace_i=1<3), vì vậy total_frames[1][1]=seq[1]=0, replace_i=2
    - Đánh dấu trang j=1 có lỗi: faults[1]=1, tăng số lỗi fault_num=2

- Ta kiểm tra thấy j<n-1(1<9) nên copy khung trang cho trang 2 tiếp theo: total_frames[i][2]=total_frames[i][1], i ∈ [0, 2]
- Lúc này, total_frames[i][1]= total_frames[i][2]=[7,0,None], i ∈ [0, 2]
- j=2

+ Ở vòng lặp j=2:
  - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][2] không chứa 1
  - Kiểm tra tiếp thấy khung trang còn trống (replace_i=2<3), vì vậy total_frames[2][2]=seq[2]=1, replace_i=3
  - Đánh dấu trang j=2 có lỗi: faults[2]=1, tăng số lỗi fault_num=3
  - Ta kiểm tra thấy j<n-1(2<9) nên copy khung trang cho trang 3 tiếp theo: total_frames[i][3]=total_frames[i][2], i ∈ [0, 2]
  - Lúc này, total_frames[i][2]=total_frames[i][3]=[7,0,1], i ∈ [0, 2]
  - j=3
+ Ở vòng lặp j=3:
  - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][3] không chứa 2
  - Kiểm tra tiếp thấy trong khung trang hết chỗ (replace_i==3)
    - Tìm vị trí gần nhất trong quá khứ của các trang đang có trong khung trang total_frames[i][3], i ∈ [0, 2] và lưu vào mảng options[]. Mảng options[]=[0,1,2] do các trang 7,0,1 được tham chiếu gần nhất trong quá khứ tại vị trí 0,1,2 trong chuỗi.
    - Tìm vị trí của phần tử nhỏ nhất trong options[] (phần tử 0 ở vị trí 0), lưu vị trí đó vào replace_i(replace_i=0): total_frames[0][3]=seq[3]=2
    - Reset options[]=[-1,-1,-1], replace_i=frame_num=3
  - Đánh dấu trang j=3 có lỗi: faults[3]=1, tăng số lỗi fault_num=4
  - Ta kiểm tra thấy j<n-1(3<9) nên copy khung trang cho trang 4 tiếp theo: total_frames[i][4]=total_frames[i][3], i ∈ [0, 2]
  - Lúc này, total_frames[i][3]=total_frames[i][4]=[2,0,1], i ∈ [0, 2]
  - j=4
+ Ở vòng lặp j=4:
  - Với i ∈ [0, 2], ta kiểm tra thấy total_frames[1][4]==0
  - Ta kiểm tra thấy j<n-1(4<9) nên copy khung trang cho trang 5 tiếp theo: total_frames[i][5]=total_frames[i][4], i ∈ [0, 2]
  - Lúc này, total_frames[i][4]=total_frames[i][5]=[2,0,1], i ∈ [0, 2]
  - j=5
+ Ở vòng lặp j=5:
  - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][5] không chứa 3
  - Kiểm tra tiếp thấy trong khung trang hết chỗ (replace_i==3)

        - Tìm vị trí gần nhất trong quá khứ của các trang đang có trong khung trang total_frames[i][5], i ∈ [0, 2] và lưu vào mảng options[]. Mảng options[]=[3,4,2].
        - Tìm vị trí của phần tử nhỏ nhất trong options[] (phần tử 2 ở vị trí 2), lưu vị trí đó vào replace_i(replace_i=2): total_frames[2][5]=seq[5]=3
        - Reset options[]=[-1, -1, -1], replace_i=frame_num=3
    - Đánh dấu trang j=5 có lỗi: faults[5]=1, tăng số lỗi fault_num=5
    - Ta kiểm tra thấy j<n-1(5<9) nên copy khung trang cho trang 6 tiếp theo: total_frames[i][6]=total_frames[i][5], i ∈ [0, 2]
    - Lúc này, total_frames[i][5]=total_frames[i][6]=[2,0,3], i ∈ [0, 2]
    - j=6
+ Ở vòng lặp j=6:
    - Với i ∈ [0, 2], ta kiểm tra thấy total_frames[1][6]==0
    - Ta kiểm tra thấy j<n-1(6<9) nên copy khung trang cho trang 7 tiếp theo: total_frames[i][7]=total_frames[i][6], i ∈ [0, 2]
    - Lúc này, total_frames[i][6]=total_frames[i][7]=[2,0,3], i ∈ [0, 2]
    - j=7
+ Ở vòng lặp j=7:
    - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][7] không chứa 4
    - Kiểm tra tiếp thấy trong khung trang hết chỗ (replace_i==3)
        - Tìm vị trí gần nhất trong quá khứ của các trang đang có trong khung trang total_frames[i][7], i ∈ [0, 2] và lưu vào mảng options[]. Mảng options[]=[3,6,5].
        - Tìm vị trí của phần tử nhỏ nhất trong options[] (phần tử 3 ở vị trí 0), lưu vị trí đó vào replace_i(replace_i=0): total_frames[0][7]=seq[7]=4
        - Reset options[]=[-1, -1, -1], replace_i=frame_num=3
    - Đánh dấu trang j=7 có lỗi: faults[7]=1, tăng số lỗi fault_num=6
    - Ta kiểm tra thấy j<n-1(7<9) nên copy khung trang cho trang 8 tiếp theo: total_frames[i][8]=total_frames[i][7], i ∈ [0, 2]
    - Lúc này, total_frames[i][7]=total_frames[i][8]=[4,0,3], i ∈ [0, 2]
    - j=8
+ Ở vòng lặp j=8:
    - Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][8] không chứa 2
    - Kiểm tra tiếp thấy trong khung trang hết chỗ (replace_i==3)
        - Tìm vị trí gần nhất trong quá khứ của các trang đang có trong khung trang total_frames[i][8], i ∈ [0, 2] và lưu vào mảng options[]. Mảng options[]=[7,6,5].

- Tìm vị trí của phần tử nhỏ nhất trong options[] (phần tử 5 ở vị trí 2), lưu vị trí đó vào replace_i(replace_i=2): total_frames[2][8]=seq[8]=2
- Reset options[]=[-1, -1, -1], replace_i=frame_num=3

• Đánh dấu trang j=8 có lỗi: faults[8]=1, tăng số lỗi fault_num=7
• Ta kiểm tra thấy j<n-1(8<9) nên copy khung trang cho trang 9 tiếp theo: total_frames[i][9]=total_frames[i][8], i ∈ [0, 2]
• Lúc này, total_frames[i][8]=total_frames[i][9]=[4,0,2], i ∈ [0, 2]
• j=9

+ Ở vòng lặp j=9
  • Với i ∈ [0, 2], ta kiểm tra thấy trong total_frames[i][9] không chứa 3
  • Kiểm tra tiếp thấy trong khung trang hết chỗ (replace_i==3)
    - Tìm vị trí gần nhất trong quá khứ của các trang đang có trong khung trang total_frames[i][9], i ∈ [0, 2] và lưu vào mảng options[]. Mảng options[]=[7,6,8].
    - Tìm vị trí của phần tử nhỏ nhất trong options[][] (phần tử 8 ở vị trí 1), lưu vị trí đó vào replace_i(replace_i=1): total_frames[1][9]=seq[9]=3
    - Reset options[]=[-1, -1, -1], replace_i=frame_num=3
  • Đánh dấu trang j=9 có lỗi: faults[9]=1, tăng số lỗi fault_num=8
  • Ta kiểm tra thấy j==n-1(9==9)
  • Lúc này, total_frames[i][9]=[4,3,2], i ∈ [0, 2]
  • j=10

+ Ở vòng lặp j=10 kiểm tra thấy j==n==10, thoát khỏi vòng lặp.

– Xuất output.
– Kết thúc chương trình.

***c. Minh họa output chạy tay trên lưu đồ:***

Number of Page Fault: 8 (fault_num)

| **7** | **0** | **1** | **2** | **0** | **3** | **0** | **4** | **2** | **3** |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 7 | 7 | 2 | 2 | 2 | 2 | 4 | 4 | 4 |
| None | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| None | None | 1 | 1 | 1 | 3 | 3 | 3 | 2 | 2 |
| **1** | **1** | **1** | **1** | **0** | **1** | **0** | **1** | **1** | **1** |

<u>*Lưu ý*</u>: Khi xuất ra màn hình, None được thay bằng khoảng trắng, và dòng cuối cùng của output (mảng faults[]) giá trị 1 được thay bằng "*", 0 được thay bằng khoảng trắng.

➢ ***Kết quả chạy tay trên lưu đồ giống với kết quả giải tay, chứng tỏ lưu đồ đúng.***

### 3.3. Thực hiện code cho giải thuật

```
// LRU lam tuong tu OPT nhung voi chuoi tham chieu truoc vi tri hien tai
void LRU(int seq[], int n, int frame_num) {
    int i, j, k, available, replace_i=0, fault_num=0, faults[MAX];
    int total_frames[frame_num][n];

    int options[frame_num];

    for(j=0; j<n; j++) {
        printf("%d ", seq[j]);
        faults[j]=0;
    }
    printf("\n");

    for (i=0; i<frame_num; i++) {
        for (j=0; j<n; j++) total_frames[i][j]=-1;
        options[i]=-1;
    }
```

*Hình 21. Thực hiện code cho giải thuật LRU (1)*

- Khai báo và khởi tạo giá trị cho các biến replace_i, fault_num, mảng faults[].
- In chuỗi tham chiếu ban đầu seq[] ra màn hình.
- Khởi tạo tất cả ô bảng total_frames[][] với giá trị -1 (tương ứng với None trong phần 3.2). options[]={-1}

```
    for(j=0; j<n; j++) {
        available=0;
        for(i=0; i<frame_num; i++) {
            if(seq[j]==total_frames[i][j]) available=1;
        }
        if(available==0) {
            if(replace_i<frame_num)
                total_frames[replace_i++][j]=seq[j];
            else {
                for(i=0; i<frame_num; i++) {
                    for(k=j-1; k>=0; k--)
                        if(total_frames[i][j]==seq[k]) {
                            options[i]=k;
                            break;
                        }
                }

                k=n-1;
                for(i=0; i<frame_num; i++) {
                    if(options[i]<k) {
                        k=options[i];
                        replace_i=i;
                    }
                }
                for(i=0; i<frame_num; i++) options[i]=-1;
```

*Hình 22. Thực hiện code cho giải thuật LRU (2)*

- Ở mỗi vòng lặp j (mỗi trang trong chuỗi):
  + Khởi tạo giá trị biến available=0 (mặc định không tìm thấy trong khung trang).
  + Lặp qua các khung trang hiện tại, nếu tìm thấy gán available=1.
  + Sau khi tìm hết trong các khung trang, nếu available vẫn = 0 (không tìm thấy), ta tiến hành thay trang.
    - Kiểm tra khung trang còn trống (replace_i<frame_num). Nếu còn trống thêm trực tiếp vào vị trí replace_i, replace_i tăng lên 1. Nếu không còn trống:
      - Đối với mỗi trang trong khung trang i ∈ [0, frame_num-1], lặp qua tất cả các trang quá khứ trong trong chuỗi (từ vị trí j-1 ngược về trước), tìm thấy tại vị trí nào thì lưu vào options[i]. Chỉ tìm vị trí gần nhất, dùng break dừng ngay khi tìm thấy vị trí đầu tiên.
      - Trong mảng options[] tìm phần tử có giá trị nhỏ nhất, lưu index của nó vào replace_i.
      - Thay trang tại vị trí replace_i: total_frames[replace_i][j]=seq[j]
      - Reset mảng options[]={-1} và replace_i=frame_num.

```
                total_frames[replace_i][j]=seq[j];
                replace_i=frame_num;
            }
            faults[j]=1;
            fault_num++;
        }
        if(j<n-1) {
            for(i=0; i<frame_num; i++)
                total_frames[i][j+1]=total_frames[i][j];
        }
    }

    for (i=0; i<frame_num; i++) {
        for(j=0; j<n; j++) {
            if(total_frames[i][j]!=-1)
                printf("%d ", total_frames[i][j]);
            else printf("  ");
        }
        printf("\n");
    }

    for(j=0; j<n; j++) {
        if(faults[j]==1)
            printf("* ");
        else printf("  ");
    }
    printf("\nNumber of Page Fault: %d\n", fault_num);
}
```

*Hình 23. Thực hiện code cho giải thuật LRU (3)*

- Đánh dấu trang j có lỗi (faults[j]=1), tăng số lỗi thêm 1.

+ Copy các khung trang hiện tại cho trang kế tiếp.

- Xuất ra màn hình các giá trị trong bảng total_frames[][], với giá trị bằng -1 (None/rỗng) xuất ra khoảng trắng thay thế.
- Xuất ra màn hình các đánh dấu lỗi trong mảng faults[], với giá trị 1 (có lỗi) xuất ra kí tự "*", với giá trị 0 xuất ra khoảng trắng.
- Xuất ra số lỗi trang lưu trong biến fault_num.

### 3.4. Trình bày tính đúng đắn của code bằng cách chạy ít nhất 03 test case

*Testcase mặc định:*

INPUT

n=11

seq[]={2, 1, 5, 2, 2, 6, 7, 9, 0, 0, 7}

frame_num=3

OUTPUT (Kết quả giải tay)

| **2** | **1** | **5** | **2** | **2** | **6** | **7** | **9** | **0** | **0** | **7** |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 9 | 9 | 9 | 9 |
| | 1 | 1 | 1 | 1 | 6 | 6 | 6 | 0 | 0 | 0 |
| | | 5 | 5 | 5 | 5 | 7 | 7 | 7 | 7 | 7 |
| * | * | * | | | * | * | * | * | | |

Number of Page Fault: 7

```
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB6$ ./lab6
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
-------------------------
Your choice: 1

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

--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
-------------------------
Your choice: 3

--- Page Replacement algorithm---
2 1 5 2 2 6 7 9 0 0 7
2 2 2 2 2 2 2 9 9 9 9
  1 1 1 1 6 6 6 0 0 0
    5 5 5 5 7 7 7 7 7
* * *     * * * *
Number of Page Fault: 7
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB6$
```

*Hình 24. Chạy test case mặc định bằng code giải thuật LRU*

➢ ***Kết quả thực hiện bằng code giống với kết quả giải tay.***

*Testcase nhập tay 1:*

INPUT

n=20

seq[]={7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1}

frame_num=3

OUTPUT (Kết quả giải tay)

| **7** | **0** | **1** | **2** | **0** | **3** | **0** | **4** | **2** | **3** | **0** | **3** | **2** | **1** | **2** | **0** | **1** | **7** | **0** | **1** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 7 | 7 | 2 | 2 | 2 | 2 | 4 | 4 | 4 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 0 |
| | | 1 | 1 | 1 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 7 | 7 | 7 |
| * | * | * | * | | * | | * | * | * | * | | | * | | * | | * | | |

Number of Page Fault: 12

```
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB6$ ./lab6
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
------------------------
Your choice: 2

Enter the size of your input sequence: 20

Enter your input sequence: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

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

--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
------------------------
Your choice: 3

--- Page Replacement algorithm---
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
7 7 7 2 2 2 2 4 4 4 0 0 0 1 1 1 1 1 1 1
  0 0 0 0 0 0 0 0 3 3 3 3 3 3 0 0 0 0 0
    1 1 1 3 3 3 2 2 2 2 2 2 2 2 2 7 7 7
* * * *   *   * * * *     *   *   *
Number of Page Fault: 12
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB6$
```

*Hình 25. Chạy test case nhập tay 1 bằng code giải thuật LRU*

➢ ***Kết quả thực hiện bằng code giống với kết quả giải tay.***

*Testcase nhập tay 2:*

INPUT

n=20

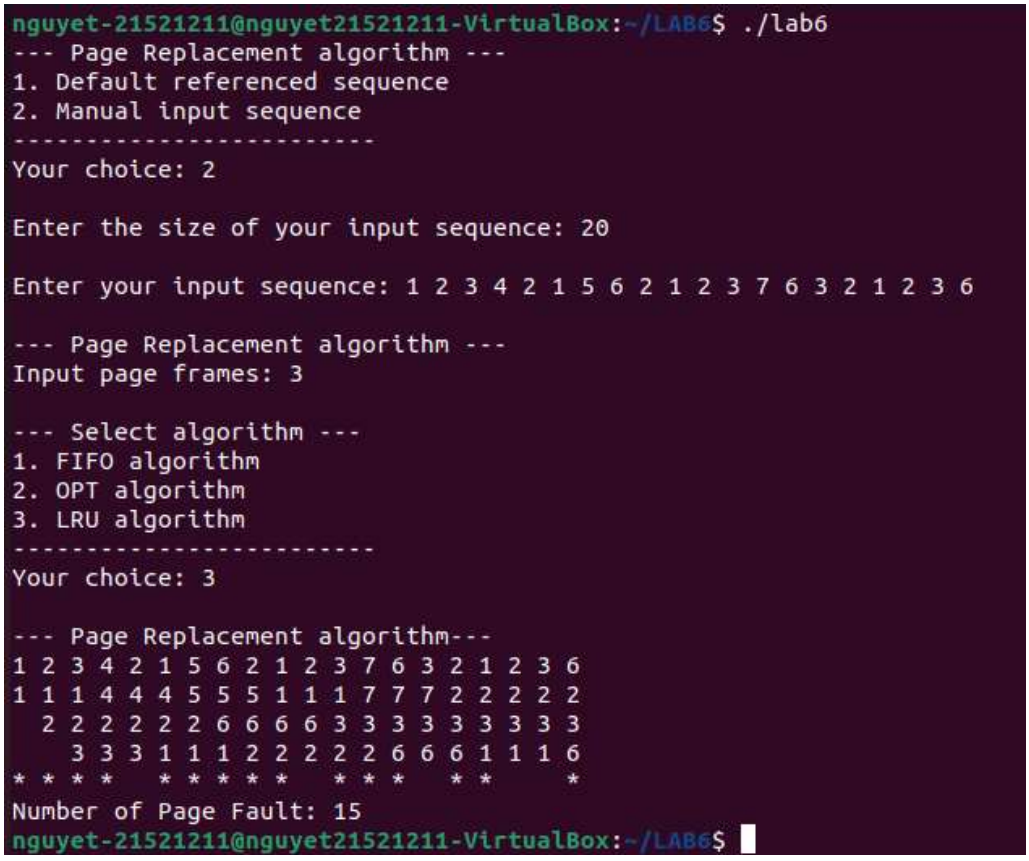
seq[]={1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6}

frame_num=3

OUTPUT (Kết quả giải tay)

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

Number of Page Fault: 15




*Hình 26. Chạy test case nhập tay 2 bằng code giải thuật LRU*

- ***Kết quả thực hiện bằng code giống với kết quả giải tay.***

- ***Cả 3 test case đều cho kết quả giống với kết quả giải tay. Kết luận code đúng.***

# Section 6.5

***Ex 1. Nghịch lý Belady***

### 1.1. Khái niệm nghịch lý Belady

Nghịch lý Belady là số page fault (lỗi trang) tăng mặc dù quá trình đã được cấp nhiều frame (khung trang) hơn lúc ban đầu.

### 1.2. Sử dụng chương trình đã viết chứng minh nghịch lý Belady

Test case:

n=12

seq[]={1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5}

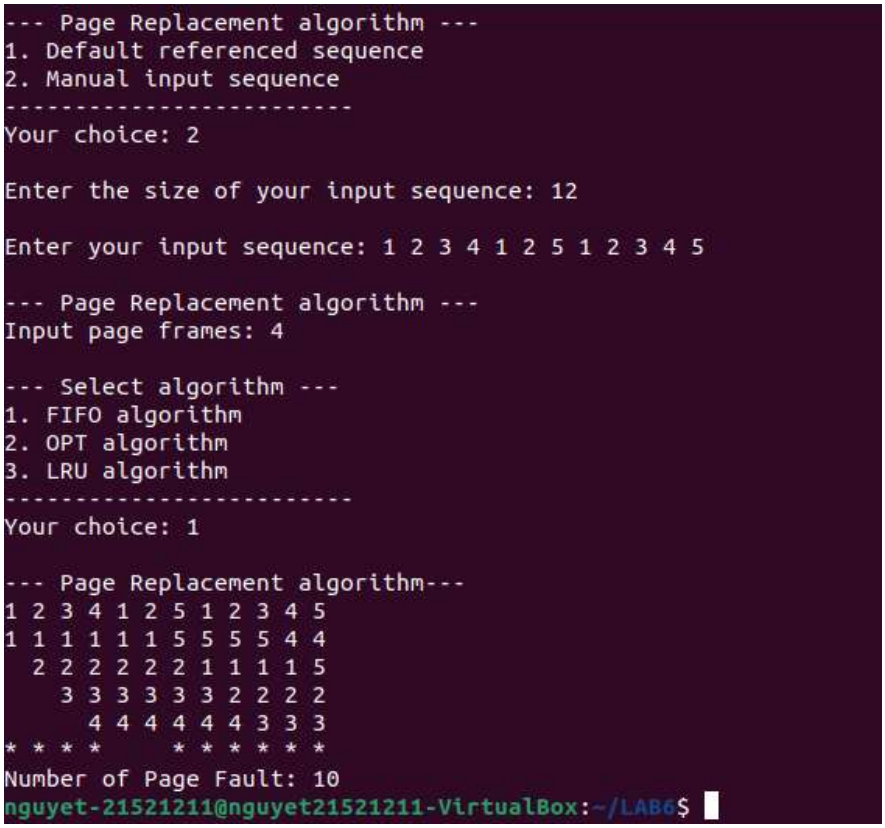
*Trường hợp 1: frame_num=3*

```
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB6$ ./lab6
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
-------------------------
Your choice: 2

Enter the size of your input sequence: 12

Enter your input sequence: 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
-------------------------
Your choice: 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
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB6$
```

*Hình 27. Chạy test case với frame_num=3*

Có thể thấy, khi sử dụng 3 khung trang, sẽ có 9 lỗi trang phát sinh.

*Trường hợp 2: frame_num=4*




*Hình 28. Chạy test case với frame_num=4*

Với cùng một chuỗi tham chiếu, khi sử dụng 4 khung trang, ta thấy có 10 lỗi trang phát sinh.

Như vậy, số lỗi trang đã tăng lên mặc dù quá trình được cấp nhiều khung trang hơn lúc ban đầu.

⇨ Nghịch lý Belady.

***Ex 2. Mức độ hiệu quả và tính khả thi của 3 giải thuật***

**2.1. Giải thuật bất khả thi nhất**

Giải thuật bất khả thi nhất là giải thuật OPT. Vì rất khó để biết được trang nào sẽ được sử dụng trong tương lai và trang nào không được sử dụng.

**2.2. Giải thuật phức tạp nhất**

Giải thuật phức tạp nhất là giải thuật LRU. Vì phải ghi nhận lại các thời điểm các trang đó truy xuất và việc này vô cùng phức tạp đòi hỏi phải có sự hỗ trợ của phần cứng để ghi nhận lại thời điểm truy xuất và cập nhật của các trang đó khi nào.