

Group name: 4COS

ID: 21520497 – 21520472 – 21520213 - 21521924

Class:IT007.N11.1

OPERATING SYSTEM LAB 6'S REPORT

SUMMARY

Task		Status	Page
6.4	Ex 1	Hoàn thành	2
	Ex 2	Hoàn thành	10
	Ex 3	Hoàn thành	18
6.5	Ex 1	Hoàn thành	29
	Ex 2	Hoàn thành	30

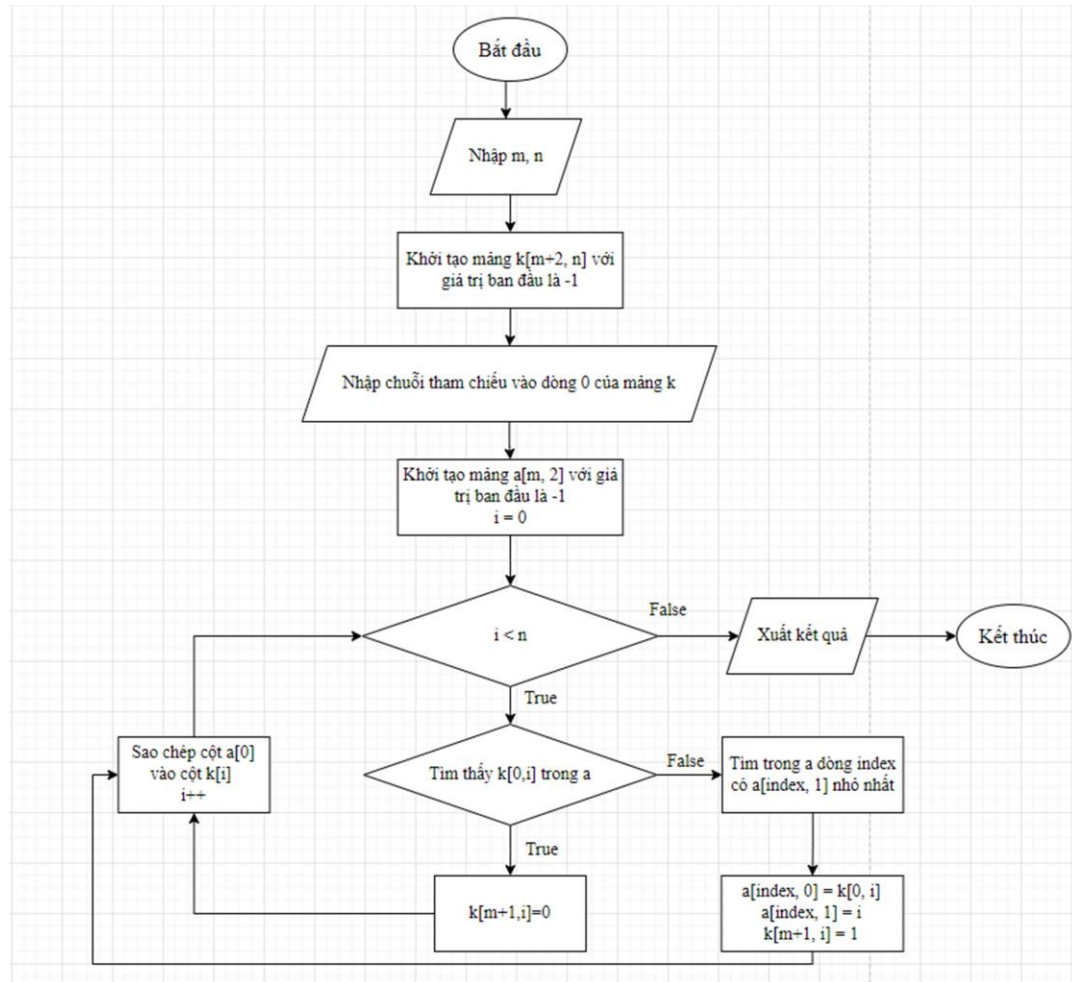
Self-scores: 10

Note: Export file to **PDF and name the file by following format:
21520497–21520472–21520213-21521924_LAB6.pdf*

6.4. Hướng dẫn thực hành

Câu 1: Vẽ lưu đồ, giải thích và hiện thực giải thuật FIFO

1. Lưu đồ giải thuật



Hình 1. 1 Lưu đồ giải thuật FIFO

2. Giải thích

- Cấu trúc dữ liệu

- Biến n lưu trữ độ dài chuỗi tham chiếu, biến m lưu trữ số lượng khung trang trống ban đầu
- Mảng k có vai trò như bảng trang, gồm có m + 2 dòng và n cột. Dòng đầu tiên chứa chuỗi tham chiếu, dòng cuối cùng đánh dấu lỗi trang (1) hay không (0).
- Mảng a có vai trò như một mảng lưu trữ trạng thái hiện thời của bảng trang tại thời điểm i, có 2 cột và m dòng tương ứng với m khung trang. Cột 0 chứa trang nhớ, cột 1 chứa thời gian mà trang nhớ đó được nạp vào trong bảng trang.

- Giải thuật

- Sau khi nhập dữ liệu, tiến hành duyệt lần lượt từng phần tử của chuỗi tham chiếu, tại lần duyệt thứ i , tiến hành làm việc với trang nhớ thứ i , tức là $k[0, i]$. Xem xét xem $k[0, i]$ có tồn tại trong mảng a (chứa trạng thái hiện thời) hay không, nếu có, tiến hành đánh dấu không gây ra lỗi trang và sao chép cột 0 của a vào cột i (từ dòng 1 đến dòng m) của k . Nếu không tìm thấy, tức xảy ra lỗi trang, tiến hành nạp trang nhớ $k[0, i]$ vào trong mảng a tại dòng có thời gian vào nhỏ nhất, đánh dấu thời điểm nó vào bảng trang tại cột 1 của bảng a . Do cột 1 đã gán giá trị bằng -1 nên việc nạp sẽ ưu tiên nạp vào khung trang trống trước, nếu không còn khung trang trống nên sẽ chọn dòng có thời gian vào nhỏ nhất để tiến hành thay trang, liên tục như vậy cho đến khi hết chuỗi tham chiếu thì tiến hành xuất bảng trang.

3. Hiện thực source code chương trình giải thuật FIFO

```
1 #include<iostream>
2 #include<math.h>
3 #include<algorithm>
4 using namespace std;
5
6 int m;
7 int n;
8 int k[100][100];
9 int a[100][2];
10
11 int main()
12 {
13     cout << "--- Page Replacement algorithm ---\n";
14     cout << "1. Default referenced sequence\n2. Manual input sequence\nEnter your selection: ";
15
16     int iluachon;
17     cin >> iluachon;
18
19     if (iluachon == 1)
20     {
21         n = 11;
22         k[0][0] = 2; k[0][1] = 1; k[0][2] = 5; k[0][3] = 2; k[0][4] = 0;
23         k[0][5] = 4; k[0][6] = 9; k[0][7] = 7; k[0][8] = 0; k[0][9] = 0; k[0][10] = 7;
24     }
25     else
26     {
27         cout << "Enter length of input sequence: ";
28         cin >> n;
29         cout << "Enter input sequence: \n";
30         for (int i = 0; i < n; i++)
31             cin >> k[0][i];
32     }
33
34     cout << "Input page frames: ";
35     cin >> m;
36
37     for (int i = 1; i <= m; i++)
38         for (int j = 1; j <= n; j++)
39             k[i][j] = -1;
40
41     for (int i = 0; i < n; i++)
42         for (int j = 0; j < 2; j++)
43             a[i][j] = -1;
44     for (int i = 0; i < n; i++)
45     {
46         bool b = false;
47         for (int j = 0; j < m; j++)
48             if (a[j][0] == k[0][i]) b = true;
```

```


49     if (b)
50         k[m + 1][i] = 0;
51     else
52     {
53         int index = 0;
54         for (int j = 1; j < m; j++)
55             if (a[j][1] < a[index][1]) index = j;
56         a[index][0] = k[0][i];
57         a[index][1] = i;
58         k[m + 1][i] = 1;
59     }
60     for (int j = 0; j < m; j++) k[j + 1][i] = a[j][0];
61 }
62
63 int count = 0;
64 for (int i = 0; i <= m; i++)
65 {
66     for (int j = 0; j < n; j++)
67     {
68         if (k[i][j] == -1) cout << " ";
69         else cout << k[i][j] << " ";
70     }
71     cout << "\n";
72 }
73 for (int i = 0; i < n; i++)
74     if (k[m + 1][i] == 1)
75     {
76         cout << "* ";
77         count++;
78     }
79     else cout << " ";
80
81
82 cout << "\nNumber of Page Fault: " << count << endl;
83 }
84

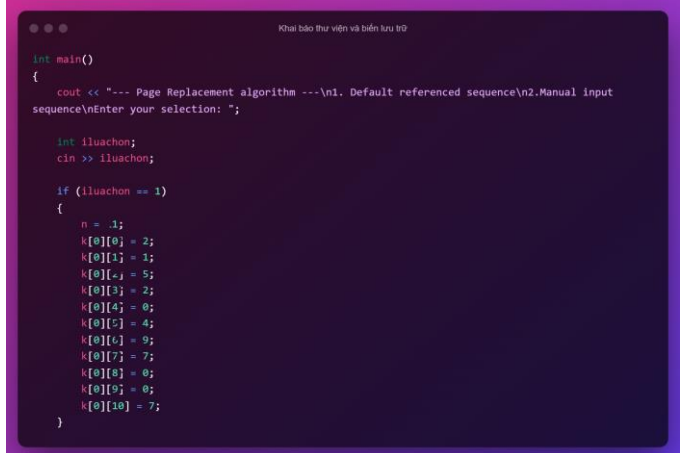
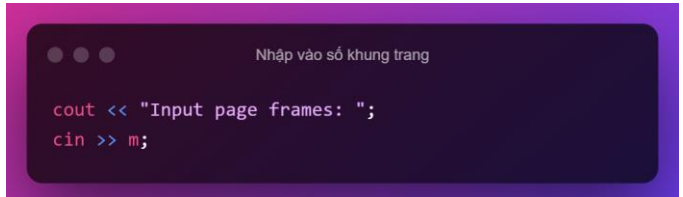


```

Hình 1. 2 Source code giải thuật FIFO

4. Kiểm tra tính đúng đắn

4.1. Kiểm tra bằng cách chạy tay

 <pre> #include <iostream> #include <math.h> #include <algorithm> using namespace std; int m; int n; int k[100][100]; int a[100][2]; </pre>	<p>Khai báo thư viện và các biến lưu trữ.</p>
---	---

 <pre> int main() { cout << "--- Page Replacement algorithm ---\n1. Default referenced sequence\n2.Manual input sequence\nEnter your selection: "; int iluachon; cin >> iluachon; if (iluachon == 1) { n = 11; k[0][0] = 2; k[0][1] = 1; k[0][2] = 5; k[0][3] = 2; k[0][4] = 0; k[0][5] = 4; k[0][6] = 9; k[0][7] = 7; k[0][8] = 0; k[0][9] = 0; k[0][10] = 7; } } </pre>	<p>Chúng ta sẽ chạy tay test case là Default referenced sequence(2,1,5,2,0,4,9,7,0,0,7).</p>
 <pre> cout << "Input page frames: "; cin >> m; </pre>	<p>Nhập vào số khung trang(page frames). Ở đây chúng ta chọn m = 3.</p>
 <pre> for (int i = 1; i <= m; i++) for (int j = 1; j <= n; j++) k[i][j] = -1; </pre>	<p>Duyệt toàn bộ ma trận k với m = 3 là số dòng và n = 11 là số cột.</p> <p>Gán giá trị -1 vào toàn bộ biến trong ma trận k.</p>
 <pre> for (int i = 0; i < n; i++) for (int j = 0; j < 2; j++) a[i][j] = -1; </pre>	<p>Duyệt toàn bộ ma trận a với n = 11 là số dòng và 2 là số cột.</p> <p>Gán giá trị -1 vào toàn bộ biến trong ma trận a.</p>

Thuật toán FIFO

```

for (int i = 0; i < n; i++)
{
    bool b = false;
    for (int j = 0; j < m; j++)
        if (a[j][0] == k[0][i])
            b = true;
    if (b)
        k[m + 1][i] = 0;
    else
    {
        int index = 0;
        for (int j = 1; j < m; j++)
            if (a[j][1] < a[index][1])
                index = j;
        a[index][0] = k[0][i];
        a[index][1] = i;
        k[m + 1][i] = 1;
    }
    for (int j = 0; j < m; j++)
        k[j + 1][i] = a[j][0];
}

```

Với $i = 0$, $b = \text{False}$, $\text{index} = 0$,
 $a[0][0] = 2$, $a[0][1] = 0$, $k[4][0] = 1$ (có xảy ra lỗi trắng), $k[1][0] = 2$,
 $k[2][0] = -1$, $k[3][0] = -1$

Với $i = 1$, $b = \text{False}$, $\text{index} = 1$,
 $a[1][0] = 1$, $a[1][1] = 1$, $k[4][1] = 1$ (có xảy ra lỗi trắng), $k[1][1] = 2$,
 $k[2][1] = 1$, $k[3][1] = -1$.

Với $i = 2$, $b = \text{False}$, $\text{index} = 2$,
 $a[2][0] = 5$, $a[2][1] = 2$, $k[4][2] = 1$ (có xảy ra lỗi trắng), $k[1][2] = 2$,
 $k[2][2] = 1$, $k[3][2] = 5$.

Với $i = 3$, $b = \text{False}$, $b = \text{True}$,
 $k[4][3] = 0$, $k[1][3] = 2$, $k[2][3] = 1$,
 $k[3][3] = 5$.

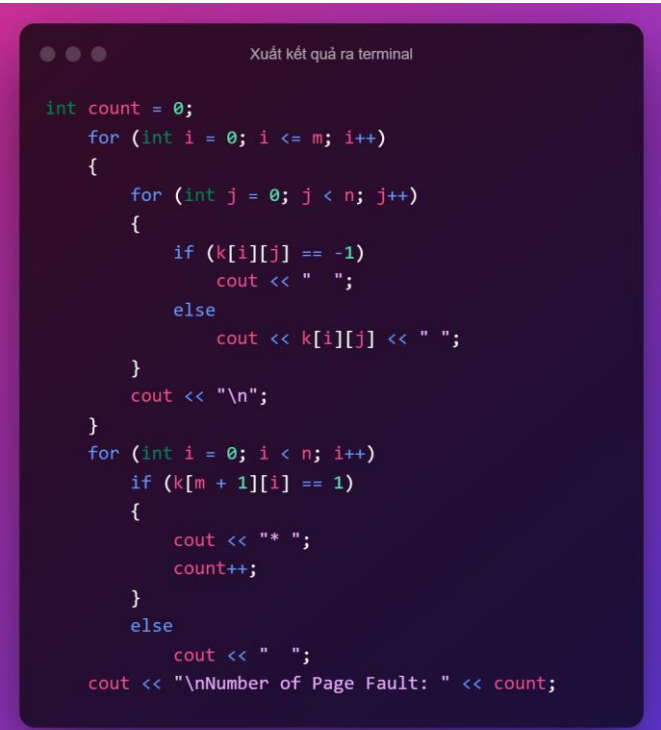
Với $i = 4$, $b = \text{False}$, $\text{index} = 0$,
 $a[0][0] = 0$, $a[0][1] = 4$, $k[4][4] = 1$ (có xảy ra lỗi trắng), $k[1][4] = 0$,
 $k[2][4] = 1$, $k[3][4] = 5$.

Với $i = 5$, $b = \text{False}$, $\text{index} = 1$,
 $a[1][0] = 4$, $a[1][1] = 5$, $k[4][5] = 1$ (có xảy ra lỗi trắng), $k[1][5] = 0$,
 $k[2][5] = 4$, $k[3][5] = 5$.

Với $i = 6$, $b = \text{False}$, $\text{index} = 2$,
 $a[2][0] = 9$, $a[2][1] = 6$, $k[4][6] = 1$ (có xảy ra lỗi trắng), $k[1][6] = 0$,
 $k[2][6] = 4$, $k[3][6] = 9$.

Với $i = 7$, $b = \text{False}$, $\text{index} = 0$,
 $a[0][0] = 7$, $a[0][1] = 7$, $k[4][7] = 1$ (có xảy ra lỗi trắng), $k[1][7] = 7$,
 $k[2][7] = 4$, $k[3][7] = 9$.

Với $i = 8$, $b = \text{False}$, $\text{index} = 1$,
 $a[1][0] = 0$, $a[1][1] = 8$, $k[4][8] = 1$ (có xảy ra lỗi trắng), $k[1][8] = 7$,
 $k[2][8] = 0$, $k[3][8] = 9$.

	<p>Với $i = 9$, $b = \text{False}$, $b = \text{True}$, $k[4][9] = 0$, $k[1][9] = 7$, $k[2][9] = 0$, $k[3][9] = 9$.</p> <p>Với $i = 10$, $b = \text{False}$, $b = \text{True}$, $k[4][10] = 0$, $k[1][10] = 7$, $k[2][10] = 0$, $k[3][10] = 9$.</p>
	<p>Kết quả được xuất ra terminal như sau:</p> <pre> 2 1 5 2 0 4 9 7 0 0 7 2 2 2 2 0 0 0 7 7 7 7 1 1 1 1 4 4 4 0 0 0 5 5 5 5 9 9 9 9 9 9 * * * * * Number of Page Fault: 8 </pre>

4.2. Kiểm tra bằng cách chạy ít nhất 3 test case

- Trường hợp Default, test case 1
- Chuỗi tham chiếu 2 1 5 2 0 4 7 2 0 0 7, frame: 3

2	1	5	2	0	4	9	7	0	0	7
2	2	2	2	0	0	0	7	7	7	7
	1	1	1	1	4	4	4	0	0	0
		5	5	5	5	9	9	9	9	9
*	*	*		*	*	*	*	*		
Num of Page Fault: 8										

Hình 1. 3 Kiểm tra chạy tay trường hợp Default

- Ta thấy, chương trình thực hiện đúng với trường hợp này

```

minhtriet-21520497@minhtriet21520497-VirtualBox: ~
minhtriet-21520497@minhtriet21520497-VirtualBox:~$ ./fifo
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
Enter your selection: 1
Input page frames: 3
2 1 5 2 0 4 9 7 0 0 7
2 2 2 2 0 0 0 7 7 7 7
1 1 1 1 4 4 4 0 0 0
5 5 5 5 9 9 9 9 9
* * * * *
Number of Page Fault: 8
minhtriet-21520497@minhtriet21520497-VirtualBox:~$

```

Hình 1. 4 Kiểm tra chạy chương trình trường hợp default

- Trường hợp Manual, test case 2
- Chuỗi tham chiếu 2 1 5 2 0 4 9 7 2 1 5 2, frame: 4

2	1	5	2	0	4	9	7	2	1	5	2
2	2	2	2	2	4	4	4	4	1	1	1
	1	1	1	1	1	9	9	9	9	5	5
		5	5	5	5	5	7	7	7	7	7
			0	0	0	0	0	2	2	2	2
*	*	*		*	*	*	*	*	*	*	
Num of Page Fault: 10											

Hình 1. 5 Kiểm tra chạy tay trường hợp test case 2

```

minhtriet-21520497@minhtriet21520497-VirtualBox:~$ ./fifo
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
Enter your selection: 2
Enter length of input sequence: 12
Enter input sequence:
2 1 5 2 0 4 9 7 2 1 5 2
Input page frames: 4
2 1 5 2 0 4 9 7 2 1 5 2
2 2 2 2 2 4 4 4 4 1 1 1
1 1 1 1 1 9 9 9 9 5 5
5 5 5 5 5 7 7 7 7 7
0 0 0 0 2 2 2 2
* * * * *
Number of Page Fault: 10
minhtriet-21520497@minhtriet21520497-VirtualBox:~$

```

Hình 1. 6 Kiểm tra chạy chương trình trường hợp test case 2

⇒ Ta thấy, chương trình thực hiện đúng với trường hợp này

- Trường hợp Manual, test case 3
- Chuỗi tham chiếu 2 1 5 2 0 4 7 2 0 4 7 2, frame: 4

2	1	5	2	0	4	7	2	0	4	7	2
2	2	2	2	2	4	4	4	4	4	4	4
	1	1	1	1	1	7	7	7	7	7	7
		5	5	5	5	5	2	2	2	2	2
			0	0	0	0	0	0	0	0	0
*	*	*		*	*	*	*				
Num of Page Fault:7											

Hình 1. 5 Kiểm tra chạy tay trường hợp test case 3

```

minhtriet-21520497@minhtriet21520497-VirtualBox:~$ ./fifo
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
Enter your selection: 2
Enter length of input sequence: 12
Enter input sequence:
2 1 5 2 0 4 7 2 0 4 7 2
Input page frames: 4
2 1 5 2 0 4 7 2 0 4 7 2
2 2 2 2 2 4 4 4 4 4 4 4
  1 1 1 1 1 7 7 7 7 7 7 7
    5 5 5 5 5 2 2 2 2 2 2
      0 0 0 0 0 0 0 0
* * * * *
Number of Page Fault: 7
minhtriet-21520497@minhtriet21520497-VirtualBox:~$

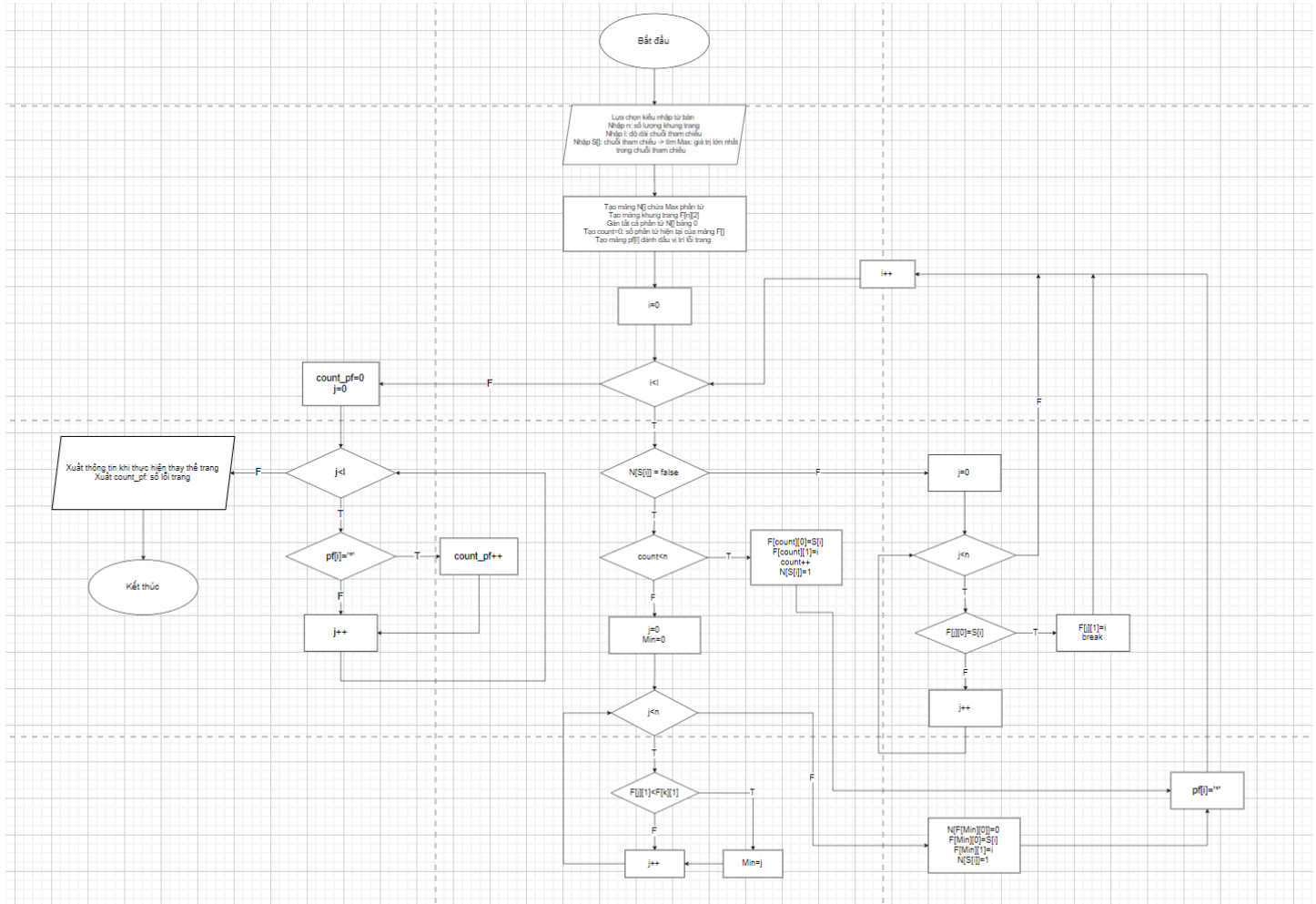
```

Hình 1. 6 Kiểm tra chạy chương trình trường hợp test case 3

⇒ Ta thấy, chương trình thực hiện đúng với trường hợp này

Câu 2: Vẽ lưu đồ, giải thích và hiện thực giải thuật LRU

1. Lưu đồ giải thuật



Hình 2. 1 Lưu đồ giải thuật LRU

2. Giải thích

- Cấu trúc dữ liệu

- n : số khung trang, l : độ dài chuỗi tham chiếu
- S: chuỗi tham chiếu
- F: khung trang
- Max : giá trị lớn nhất trong mảng S
- N : mảng chứa Max phần tử
- Count : số phần tử hiện tại của mảng F
- Pf : mảng dùng đánh dấu vị trí lỗi trang

- Count_pf : số lỗi trang
- Giải thuật
 - Đầu tiên, nhập 1 số thông tin cần thiết ($n, l, s[l]$), tìm Max là giá trị lớn nhất trong mảng S.
 - Khởi tạo mảng N là 1 mảng đánh dấu các phần tử có mặt trong mảng F, có Max phần tử được gán bằng 0; Mảng F là mảng khung trang, có n phần tử, $F[i][0]$ là giá trị của phần tử, $F[i][1]$ là thời điểm cuối cùng phần tử được sử dụng; count là số phần tử hiện hành trong F, được gán bằng 0; Mảng pf đánh dấu vị trí các lỗi trang.
 - Thực hiện 1 vòng lặp từ 0 đến l, xét từng phần tử $s[i]$ của mảng s
 - Nếu $N[s[i]] = 0$, tức là trong mảng F không có phần tử có giá trị $s[i]$ thì kiểm tra xem mảng F có còn trống hay không ($count < n$):
 - Nếu mảng F còn trống, thì thêm $s[i]$ vào F, thực hiện đánh dấu $N[s[i]] = 1$, tăng count, ghi lại giá trị $s[i]$ và thời gian i mà $s[i]$ được sử dụng vào $F[count]$.
 - Nếu mảng F đã đầy, thì tìm trong mảng F phần tử có thời gian sử dụng cuối cùng nhỏ nhất, thực hiện thay thế phần tử tìm được bằng $s[i]$.
 - Đánh dấu i là vị trí có xuất hiện lỗi trang.
 - Nếu $N[s[i]]$ bằng 1, tức là trong mảng F có phần tử có giá trị $s[i]$, ta thực hiện tìm phần tử đó trong mảng F và thay đổi thời gian cuối cùng phần tử đó được sử dụng thành i.
 - Kết thúc và xuất các thông tin cần thiết.

3. Hiện thực source code chương trình giải thuật FIFO

```
50         if (N[s[i]] == 0){
51             if (count < n)
52             {
53                 F[count][0] = s[i];
54                 F[count][1] = i;
55                 count++;
56                 N[s[i]] = 1;
57             }
58             else
59             {
60                 int Min = 0;
61                 for (int j = 0; j < n; j++){
62                     if (F[j][1] < F[Min][1])
63                         Min = j;
64                     N[F[Min][0]] = 0;
65                     N[s[i]] = 1;
66                     F[Min][0] = s[i];
67                     F[Min][1] = i;
68                 }
69                 pf[i] = '*';
70             }
71         }
72         else{
73             for (int j = 0; j < n; j++){
74                 if (F[j][0] == s[i])
75                 {
76                     F[j][1] = i;
77                     break;
78                 }
79             }
80             for (int j = 0; j < count; j++){
81                 result[j][i] = F[j][0];
82             }
83         }
84         for (int i = 0; i < l; i++)
85             cout << s[i] << ' ';
86         cout << '\n';
87         for (int i=0; i<n; i++)
88         {
89             for (int j = 0; j < l; j++){
90                 if (result[i][j] == -1)
91                     cout << " ";
92                 else
93                     cout << result[i][j] << ' ';
94             }
95             cout << '\n';
96         }
97         int count_pf = 0;
98         for (int i = 0; i < l; i++)
99         {
100             cout << pf[i] << " ";
101             if (pf[i] == '*')
102                 count_pf++;
103         }
104         cout << "\nNumber of Page Fault:" << count_pf<<'\n';
105     }
```

Hình 2. 2 Source code giải thuật LRU

4. Kiểm tra tính đúng đắn

4.1. Kiểm tra bằng cách chạy tay 1 test case

```

1 #include <iostream>
2 using namespace std;
3 int n, l, s[100] = { 2,1,5,2,0,4,7,2,0,0,7 }, N[1000], F[100][2], Max = 0;
4 int result[100][100];
5 char pf[100];
6 //s chưa chuỗi tham chiếu
7 // N là mảng lựa chọn
8 // F là bảng khung trang
9 void input()
10 {
11     cout << "1. Default Reference Sequence\n";
12     cout << "2. Manual Input Sequence\n";
13     int k;
14     cout << "Enter your selection: ";
15     cin >> k;
16     cout << "Input page frames: ";
17     cin >> n;
18     if (k == 2)
19     {
20         cout << "Length of Input Sequence:";
21         cin >> l;
22         cout << "Input Sequence:";
23         for (int i = 0; i < l; i++)
24         {
25             cin >> s[i];
26             Max = max(Max, s[i]);
27         }
28     }
29     else
30     {
31         l = 11;
32     }
33 }

```

- Chọn lựa : Nhấn 2 để nhập tay
- Nhập độ dài của chuỗi tham chiếu là 20
- Nhập chuỗi tham chiếu vào
- Nhập số khung trang : 3

```

34 void init()
35 {
36     for (int i = 0; i < Max; i++)
37         N[i] = 0;
38     for (int i = 0; i < n; i++)
39         F[i][0] = -1;
40     for (int i = 0; i < n; i++)
41     {
42         for (int j = 0; j < l; j++)
43             result[i][j] = -1;
44     }
45 }

```

- Khởi tạo mảng N có 7 phần tử = 0
- Tạo mảng khung trang F[3][2]
- Khởi tạo count = 0
- Khởi tạo mảng result có giá trị -1
- Khởi tạo mảng F[i][0] = -1

```

51     for (int i = 0; i < l; i++)
52     {
53         if (N[s[i]] == 0)
54         {
55             if (count < n)
56             {
57                 F[count][0] = s[i];
58                 F[count][1] = i;
59                 count++;
60                 N[s[i]] = 1;
61             }
62             else
63             {
64                 int Min = 0;
65                 for (int j = 0; j < n; j++)
66                     if (F[j][1] < F[Min][1])
67                         Min = j;
68                 N[F[Min][0]] = 0;
69                 N[s[i]] = 1;
70                 F[Min][0] = s[i];
71                 F[Min][1] = i;
72             }
73             pf[i] = '+';
74         }
75         else{
76             for (int j = 0; j < n; j++){
77                 if (F[j][0] == s[i])
78                 {
79                     F[j][1] = i;
80                     break;
81                 }
82             }
83             for (int j = 0; j < count; j++)
84             {
85                 result[j][i] = F[j][0];
86             }
87         }
88     }
89     for (int i = 0; i < l; i++)
90         cout << s[i] << " ";
91     cout << "\n";
92     for (int i=0;i<n;i++)
93     {
94         for (int j = 0; j < l; j++)
95             if (result[i][j] == -1)
96                 cout << " ";
97         else
98             cout << result[i][j] << " ";
99         cout << "\n";
100     }
101     int count_pf = 0;
102     for (int i = 0; i < l; i++)
103     {
104         cout << pf[i] << " ";
105         if (pf[i] == '+')
106             count_pf++;
107     }
108     cout << "\nNumber of Page Fault:" << count_pf << "\n";
109 }

```

Với i=0:
Ta có s[0]=7 và N[s[0]] = 0, count < n=3, F[1][0]=-1, F[1][1]=0, Phần tử thứ 0 của chuỗi có xảy ra lỗi trang.

Với i=1:
Ta có s[1]=0 và N[s[1]] = 0, count < n=3, F[2][0]=-1, F[2][1]=0, Phần tử thứ 1 của chuỗi có xảy ra lỗi trang.

Với i=2:
Ta có s[2]=1 và N[s[2]] = 0, count < n=3, F[3][0]=0, F[3][1]=0, Phần tử thứ 2 của chuỗi có xảy ra lỗi trang.

Với i=3:
Ta có s[3]=2 và N[s[3]] = 0, count = n=3, Khởi tạo Min = 0, Tìm chỉ số phần tử nhỏ nhất trong F, được Min = 0 có giá trị là 7, N[7] = 0, N[2] = 1, F[0][0] = 2, F[0][1] = 3, Phần tử thứ 3 của chuỗi có xảy ra lỗi trang.

Với i=4:
Ta có s[4]=0 và N[s[4]] = 1, F[1][0] = s[i] = 0, F[1][1] = 4

	<p>Với $i=5$: Ta có $s[5]=3$ và $N[s[5]] = 0$, $count = n=3$, Khởi tạo $Min = 0$, Tìm chỉ số phần tử nhỏ nhất trong F, được $Min = 2$ có giá trị là 1, $N[1] = 0$, $N[3] = 1$, $F[2][0] = 3$, $F[2][1] = 5$, Phần tử thứ 5 của chuỗi có xảy ra lỗi trang.</p> <p>Với $i=6$: Ta có $s[6]=0$ và $N[s[6]] = 1$, $F[1][0] = s[i] = 0$, $F[1][1] = 6$</p> <p>Với $i=7$: Ta có $s[7]=4$ và $N[s[7]] = 0$, $count = n=3$, Khởi tạo $Min = 0$, Tìm chỉ số phần tử nhỏ nhất trong F, được $Min = 0$ có giá trị là 2, $N[2] = 0$, $N[4] = 1$, $F[0][0] = 4$, $F[0][1] = 7$, Phần tử thứ 7 của chuỗi có xảy ra lỗi trang.</p> <p>Với $i=8$: Ta có $s[8]=2$ và $N[s[8]] = 0$, $count = n=3$, Khởi tạo $Min = 0$, Tìm chỉ số phần tử nhỏ nhất trong F, được $Min = 2$ có giá trị là 3, $N[3] = 0$, $N[2] = 1$, $F[2][0] = 2$, $F[2][1] = 8$, Phần tử thứ 8 của chuỗi có xảy ra lỗi trang.</p> <p>Với $i=9$: Ta có $s[9]=3$ và $N[s[9]] = 0$, $count = n=3$, Khởi tạo $Min = 0$, Tìm chỉ số phần tử nhỏ nhất trong F, được $Min = 1$ có giá trị là 0, $N[0] = 0$, $N[3] = 1$, $F[1][0] = 3$, $F[1][1] = 9$, Phần tử thứ 9 của chuỗi có xảy ra lỗi trang.</p> <p>Với $i=10$: Ta có $s[10]=0$ và $N[s[10]] = 0$, $count = n=3$, Khởi tạo $Min = 0$, Tìm chỉ số phần tử nhỏ nhất trong F, được $Min = 0$ có giá trị là 4, $N[4] = 0$, $N[0] = 1$, $F[0][0] = 0$, $F[0][1] = 10$, Phần tử thứ 10 của chuỗi có xảy ra lỗi trang</p> <p>Với $i=11$: Ta có $s[11]=3$ và $N[s[11]] = 1$, $F[1][0] = s[i] = 3$, $F[1][1] = 11$</p> <p>Với $i=12$: Ta có $s[12]=2$ và $N[s[12]] = 1$, $F[2][0] = s[i] = 2$, $F[2][1] = 12$</p>
--	---

	<p>Với $i=13$: Ta có $s[13]=1$ và $N[s[13]] = 0$, $count = n=3$, Khởi tạo $Min = 0$, Tìm chỉ số phần tử nhỏ nhất trong F, được $Min = 0$ có giá trị là 0, $N[0] = 0$, $N[1] = 1$, $F[0][0] = 1$, $F[0][1] = 13$, Phần tử thứ 13 của chuỗi có xảy ra lỗi trang.</p> <p>Với $i=14$: Ta có $s[14]=2$ và $N[s[14]] = 1$, $F[2][0] = s[i] = 2$, $F[2][1] = 14$</p> <p>Với $i=15$: Ta có $s[15]=0$ và $N[s[15]] = 0$, $count = n=3$, Khởi tạo $Min = 0$, Tìm chỉ số phần tử nhỏ nhất trong F, được $Min = 1$ có giá trị là 3, $N[3] = 0$, $N[0] = 1$, $F[1][0] = 0$, $F[1][1] = 15$, Phần tử thứ 15 của chuỗi có xảy ra lỗi trang.</p> <p>Với $i=16$: Ta có $s[16]=1$ và $N[s[16]] = 1$, $F[0][0] = s[i] = 1$, $F[0][1] = 16$</p> <p>Với $i=17$: Ta có $s[17]=7$ và $N[s[17]] = 0$, $count = n=3$, Khởi tạo $Min = 0$, Tìm chỉ số phần tử nhỏ nhất trong F, được $Min = 2$ có giá trị là 2, $N[2] = 0$, $N[7] = 1$, $F[2][0] = 7$, $F[2][1] = 17$, Phần tử thứ 17 của chuỗi có xảy ra lỗi trang</p> <p>Với $i=18$: Ta có $s[18]=0$ và $N[s[18]] = 1$, $F[1][0] = s[i] = 0$, $F[1][1] = 18$</p> <p>Với $i=19$: Ta có $s[19]=1$ và $N[s[19]] = 1$ Ta có $F[0][0] = s[i] = 1$ $F[0][1] = 19$</p>
--	---

<pre> 83 for (int i = 0; i < l; i++) 84 cout << s[i] << ' '; 85 cout << '\n'; 86 for (int i=0; i<n; i++) 87 { 88 for (int j = 0; j < l; j++) 89 if (result[i][j] == -1) 90 cout << " "; 91 else 92 cout << result[i][j] << ' '; 93 cout << '\n'; 94 } 95 int count_pf = 0; 96 for (int i = 0; i < l; i++) 97 { 98 cout << pf[i] << " "; 99 if (pf[i] == '+') 100 count_pf++; 101 } 102 cout << "\nNumber of Page Fault:" << count_pf<<'\n'; 103) </pre>	<ul style="list-style-type: none"> - Xuất bảng phân trang - Xuất số lỗi là 12
--	---

4.2. Kiểm tra bằng cách chạy 3 test case

	2	1	5	2	0	4	7	2	0	0	7
	2	2	2	2	2	2	2	2	2	2	2
		1	1	1	1	4	4	4	4	4	4
			5	5	5	5	7	7	7	7	7
					0	0	0	0	0	0	0
Number of Page Fault: 6	x	x	x		x	x	x				

Hình 2. 3 Kiểm tra chạy tay trường hợp Default

```

thu-21520472@thu21520472-VirtualBox:~$ gedit LRU.cpp
thu-21520472@thu21520472-VirtualBox:~$ g++ -o LRU LRU.cpp
thu-21520472@thu21520472-VirtualBox:~$ ./LRU
1. Default Reference Sequence
2. Manual Input Sequence
Enter your selection: 1
Input page frames: 4
2 1 5 2 0 4 7 2 0 0 7
2 2 2 2 2 2 2 2 2 2 2
  1 1 1 1 4 4 4 4 4 4
    5 5 5 5 7 7 7 7 7
      0 0 0 0 0 0 0
* * * * *
Number of Page Fault:6
thu-21520472@thu21520472-VirtualBox:~$

```

Hình 2. 4 Kiểm tra chạy chương trình trường hợp Default

- Trường hợp Default: chuỗi tham chiếu 2 1 5 2 0 4 7 2 0 0 7, frame: 3
- ⇒ Ta thấy, chương trình thực hiện đúng với trường hợp này

- Trường hợp Manual, test case 2
- Chuỗi tham chiếu 2 2 1 0 2 0 0 3 1 9 5, frame: 4

	2	2	1	0	2	0	0	3	1	9	5
	2	2	2	2	2	2	2	2	2	9	9
			1	1	1	1	1	1	1	1	1
				0	0	0	0	0	0	0	5
								3	3	3	3
Number of page fault: 6	x		x	x				x		x	x

Hình 2. 5 Kiểm tra chạy tay trường hợp test case 2

```

thu-21520472@thu21520472-VirtualBox:~$ ./LRU
1. Default Reference Sequence
2. Manual Input Sequence
Enter your selection: 2
Input page frames: 4
Length of Input Sequence:11
Input Sequence:2 2 1 0 2 0 0 3 1 9 5
2 2 1 0 2 0 0 3 1 9 5
2 2 2 2 2 2 2 2 2 9 9
    1 1 1 1 1 1 1 1 1
      0 0 0 0 0 0 0 5
        3 3 3 3
* * * * *
Number of Page Fault:6

```

Hình 2. 6 Kiểm tra chạy chương trình trường hợp test case 2

⇒ Ta thấy, chương trình thực hiện đúng với trường hợp này

- Trường hợp Manual, test case 3
- Chuỗi tham chiếu: 2 5 0 2 5 7 2 0 1 3 1 6 4, frame: 5

	2	5	0	2	5	7	2	0	1	3	1	6	4
	2	2	2	2	2	2	2	2	2	2	2	2	4
		5	5	5	5	5	5	5	5	3	3	3	3
			0	0	0	0	0	0	0	0	0	0	0
						7	7	7	7	7	7	6	6
									1	1	1	1	1
Number of page fault: 8	x	x	x			x			x	x		x	x

Hình 2. 7 Kiểm tra chạy tay trường hợp test case 3

```

thu-21520472@thu21520472-VirtualBox:~$ ./LRU
1. Default Reference Sequence
2. Manual Input Sequence
Enter your selection: 2
Input page frames: 5
Length of Input Sequence:13
Input Sequence:2 5 0 2 5 7 2 0 1 3 1 6 4
2 5 0 2 5 7 2 0 1 3 1 6 4
2 2 2 2 2 2 2 2 2 2 2 4
5 5 5 5 5 5 5 3 3 3 3
0 0 0 0 0 0 0 0 0 0
  7 7 7 7 7 6 6
    1 1 1 1 1
* * * * *
Number of Page Fault:8
thu-21520472@thu21520472-VirtualBox:~$

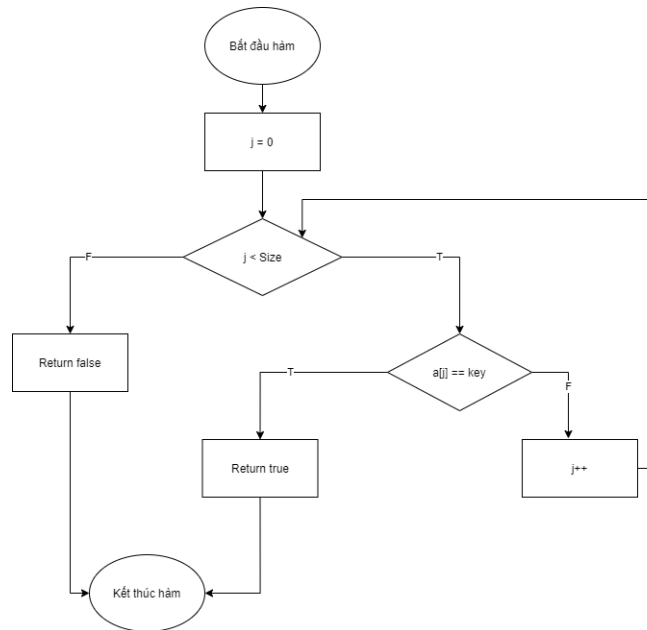
```

Hình 2. 8 Kiểm tra chạy chương trình trường hợp test case 3

Câu 3: Vẽ lưu đồ, giải thích và hiện thực giải thuật OPT

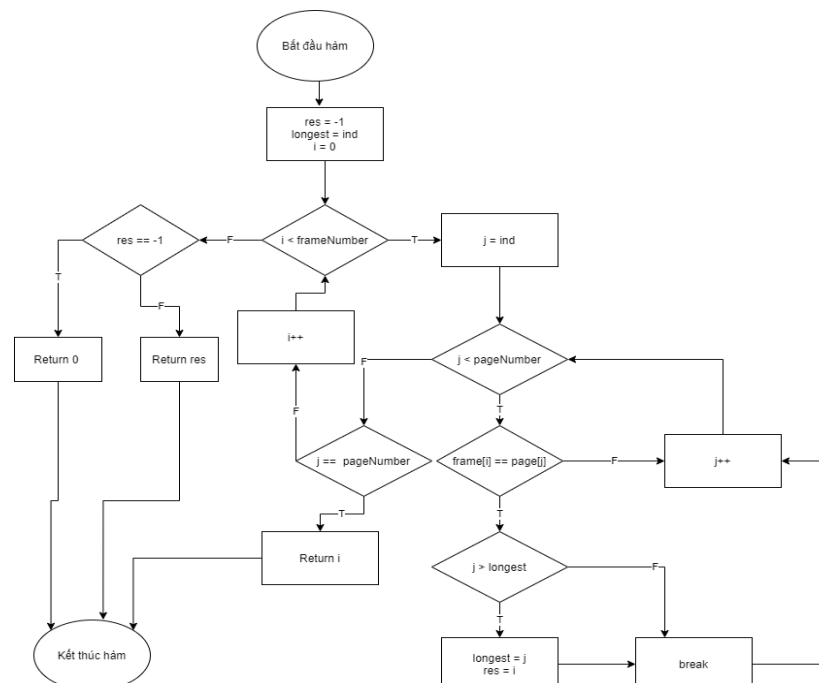
1. Lưu đồ giải thuật

a. Hàm Search(int key, int *a, int Size)



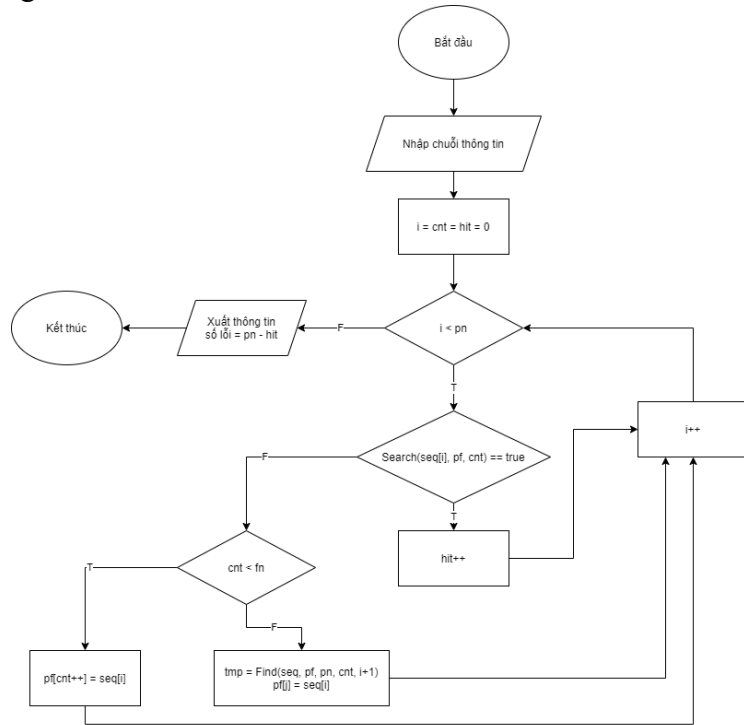
Hình 3. 1 Lưu đồ giải thuật hàm Search

b. Hàm Find(int *page, int *frame, int pageNumber, int frameNumber, int ind)



Hình 3. 2 Lưu đồ giải thuật Find

c. Chương trình chính



Hình 3. 3 Lưu đồ giải thuật chương trình chính giải thuật OPT

2. Giải thích

- Cấu trúc dữ liệu

- Hàm Search

- Biến key thể hiện khung trang nào đang được tìm kiếm
- Mảng a thể hiện thông tin các khung trang của một trang
- Size là số lượng khung trang của trang đang xét

- Hàm Find

- Page là mảng chứa chuỗi thông tin các trang
- Frame là mảng chứa thông tin các khung trang hiện tại
- pageNumber chứa số lượng trang
- frameNumber cho biết số lượng khung trang hiện tại đang có
- ind cho biết vị trí bắt đầu tìm từ trang bao nhiêu trong chuỗi trang tham chiếu
- longest cho biết vị trí trang được tham chiếu trễ nhất trong tương lai
- frame cho biết khung trang nào sẽ được thay thế

- Chương trình chính
 - Seq là mảng 1 chiều là chuỗi trang tham chiếu, pn cho biết số lượng trang trong bảng trang, Result mảng 2 chiều chứa bảng trang.
 - Hit cho biết số lượng trang không lỗi, cnt biến đếm để xét số khung trang đã không trống hiện tại có trong trang
- Giải thuật
 - Hàm Search
 - Mục đích là tìm xem trang đó đã xuất hiện tại dòng nào chưa nếu có trả về true ngược lại trả về false
 - Giải thuật sẽ xét xem giá trị key có trong mảng a hay không bằng cách dùng vòng lặp duyệt qua từng phần tử. Nếu có trả về true ngược lại trả về false
 - Hàm Find
 - Mục đích của hàm là tìm trang để thay thế.
 - Duyệt qua những khung trang hiện tại xem coi có trang nhớ nào trong tương lai xuất hiện lại không tức là $frame[i] == page[j]$. Nếu có xuất hiện thì tìm tất cả những trang có xuất hiện và lấy ra trang được tham chiếu tới lâu nhất. Nếu dòng thứ i đang xét trong tương lai không xuất hiện nữa thì trả về chính nó. Hàm trả về biến res.
 - Hàm chính
 - Mục đích của hàm chính là thực hiện giải thuật opt và lưu giá trị của bảng trang. Duyệt qua từng trang trong chuỗi tham chiếu. Nếu như trang đó có tồn tại trong một frame nhớ nào đó thì $hit++$ tức là trang đó không bị lỗi và cập nhật $result[fn][i] = -2$ (để khi xuất ra sẽ xuất trang đó không bị lỗi). Tiến hành sao chép từ pf sang result mà không có thay đổi gì
 - Ngược lại nếu không tồn tại một frame nhớ nào thì xảy ra lỗi trang. Tiến hành tìm trang nhớ để thay thế trang đang xét. Bằng cách nếu còn khung trang trống thì sẽ ưu tiên nạp vào khung trang trống đó trước. Nếu không còn khung trang trống nên sẽ phải chọn trang nào có thời gian tham chiếu trễ nhất để tiến hành thay trang. Liên tục như vậy cho đến khi hết chuỗi tham chiếu thì tiến hành xuất bảng trang.

3. Hiện thực source code chương trình giải thuật OPT

```
1 #include<iostream>
2 #include<math.h>
3 #include<algorithm>
4 using namespace std;
5
6 bool Search(int key, int *a, int Size)
7 {
8     for(int j = 0; j < Size; j++)
9         if(a[j] == key)
10             return true;
11     return false;
12 }
13
14 int Find(int *page, int *frame, int pageNumber, int frameNumber, int ind)
15 {
16     int res = -1, longest = ind;
17     for(int i = 0; i < frameNumber; i++)
18     {
19         int j;
20         for(j = ind; j < pageNumber; j++)
21             if(frame[i] == page[j])
22             {
23                 if(j > longest)
24                 {
25                     longest = j;
26                     res = i;
27                 }
28                 break;
29             }
30         if(j == pageNumber)
31             return i;
32     }
33     if(res == -1)
34         return 0;
35     return res;
36 }
```

Hình 3. 4 Source code hàm Search và Find

```

38 int main(){
39     int option;
40     int *seq, fn, pn;
41
42     cout << "--- Page Replacement algorithm ---\n";
43     cout << "1. Default referenced sequence\n";
44     cout << "2. Manual input sequence\n";
45     cout << "Enter your selection (1 or 2): ";
46     cin >> option;
47
48     if(option == 1)
49     {
50         pn = 11;
51         seq = new int[pn];
52         seq[0] = 2; seq[1] = 1; seq[2] = 5; seq[3] = 2; seq[4] = 0; seq[5] = 4;
53         seq[6] = 9; seq[7] = 7; seq[8] = 0; seq[9] = 0; seq[10] = 7;
54     }
55     else
56     {
57         cout << "Enter length of input sequence: ";
58         cin >> pn;
59         seq = new int[pn];
60         cout << "Enter input sequence: \n";
61         for(int i = 0; i < pn; i++)
62             cin >> seq[i];
63     }
64
65     cout << "Input page frames: ";
66     cin >> fn;
67
68     int **result = new int*[fn + 1];
69     for(int i = 0; i <= fn; i++)
70         result[i] = new int[pn];
71
72     int cnt = 0;
73     int *pf = new int[fn];
74     int hit = 0;
75     for(int i = 0; i < pn; i++){
76         if(Search(seq[i], pf, cnt))
77         {
78             hit++;
79             result[fn][i] = -2;
80             for(int j = 0; j < fn; j++)
81                 if(j < cnt)
82                     result[j][i] = pf[j];
83             else
84                 result[j][i] = -1;
85             continue;
86         }
87
88         if(cnt < fn)
89             pf[cnt++] = seq[i];
90         else
91         {
92             int j = Find(seq, pf, pn, cnt, i + 1);
93             pf[j] = seq[i];
94         }
95         result[fn][i] = -3;
96         for(int j = 0; j < fn; j++)
97             if(j < cnt)
98                 result[j][i] = pf[j];
99             else
100                 result[j][i] = -1;
101     }
102     for(int i = 0; i < pn; i++)
103         cout << seq[i] << " ";
104     cout << '\n';
105     for(int i = 0; i <= fn; i++)
106     {
107         for(int j = 0; j < pn; j++)
108             if((result[i][j] == -1) || (result[i][j] == -2))
109                 cout << " ";
110             else
111                 if(result[i][j] == -3)
112                     cout << "* ";
113             else
114                 cout << result[i][j] << " ";
115         cout << endl;
116     }
117
118     cout << "Number of Page Fault: " << pn - hit << endl;
119 }

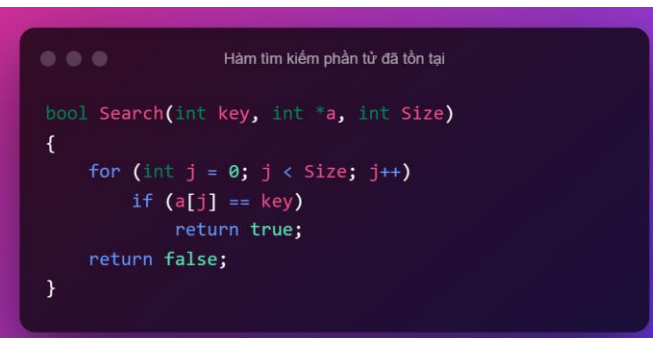
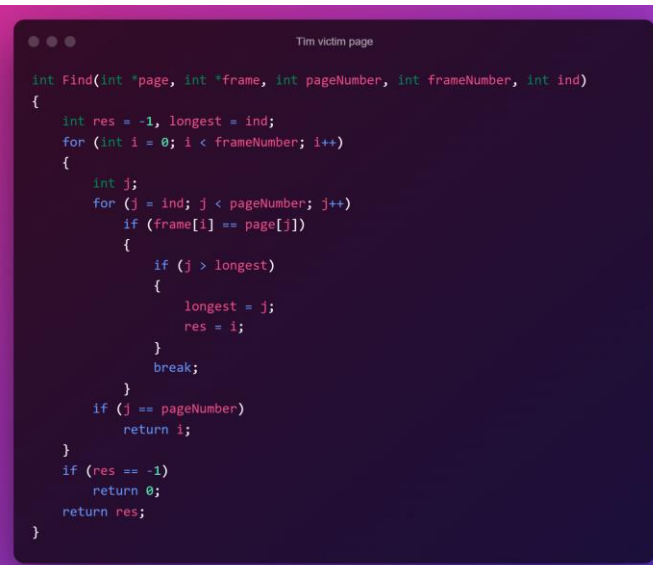
```

Hình 3. 5 Source code chương trình chính giải thuật OPT

4. Kiểm tra tính đúng đắn

4.1 Kiểm tra bằng cách chạy tay

 <pre>int main() { int option; int *seq, fn, pn; cout << "--- Page Replacement algorithm ---\n"; cout << "1. Default referenced sequence\n"; cout << "2. Manual input sequence\n"; cout << "Enter your selection (1 or 2): "; cin >> option; if (option == 1) { pn = 11; seq = new int[pn]; seq[0] = 2; seq[1] = 1; seq[2] = 5; seq[3] = 2; seq[4] = 0; seq[5] = 4; seq[6] = 9; seq[7] = 7; seq[8] = 0; seq[9] = 0; seq[10] = 7; } }</pre>	<p>Khai báo các biến lưu trữ.</p> <p>Ở đây chúng ta chọn lựa chọn chạy tay là Default referenced sequence với seq là (2,1,5,2,0,4,9,7,0,0,7) và pn = 11.</p>
 <pre>cout << "Input page frames: "; cin >> fn;</pre>	<p>Nhập vào số khung trang(page frames). Ở đây chúng ta chọn fn = 3</p>
 <pre>int **result = new int *[fn + 1]; for (int i = 0; i <= fn; i++) result[i] = new int[pn];</pre>	<p>Tạo ma trận lưu trữ kết quả.</p>

 <pre>bool Search(int key, int *a, int Size) { for (int j = 0; j < Size; j++) if (a[j] == key) return true; return false; }</pre>	<p>Hàm tìm kiếm các phần tử trên một cột của ma trận nếu đã tồn tại phần tử thì trả về kết quả true, ngược lại là false</p>
 <pre>int Find(int *page, int *frame, int pageNumber, int frameNumber, int ind) { int res = -1, longest = ind; for (int i = 0; i < frameNumber; i++) { int j; for (j = ind; j < pageNumber; j++) if (frame[i] == page[j]) { if (j > longest) { longest = j; res = i; } break; } if (j == pageNumber) return i; } if (res == -1) return 0; return res; }</pre>	<p>Tìm victim page để thay thế trang.</p>


```

Thuật toán OPT

int cnt = 0;
int *pf = new int[fn];
int hit = 0;
for (int i = 0; i < pn; i++)
{
    if (Search(seq[i], pf, cnt))
    {
        hit++;
        result[fn][i] = -2;
        for (int j = 0; j < fn; j++)
            if (j < cnt)
                result[j][i] = pf[j];
        else
            result[j][i] = -1;
        continue;
    }

    if (cnt < fn)
        pf[cnt++] = seq[i];
    else
    {
        int j = Find(seq, pf, pn, cnt, i + 1);
        pf[j] = seq[i];
    }
    result[fn][i] = -3;
    for (int j = 0; j < fn; j++)
        if (j < cnt)
            result[j][i] = pf[j];
        else
            result[j][i] = -1;
}

```

- Với $i = 0$, $\text{Seq}[0] = 2$, $\text{cnt} = 0$, $\text{hit} = 0$.

Hàm search trả về kết quả False, $\text{cnt} < \text{fn}$ trả về True nên $\text{pf}[0] = 2$, $\text{result}[3][0] = -3$, $\text{result}[0][0] = 2$, $\text{result}[1][0]$ và $\text{result}[2][0] = -1$.

- Với $i = 1$, $\text{seq}[1] = 1$, $\text{cnt} = 1$, $\text{hit} = 0$.

Hàm search trả về kết quả False, $\text{cnt} < \text{fn}$ trả về true nên $\text{pf}[1] = 1$, $\text{result}[3][1] = -3$, $\text{result}[0][1] = 2$, $\text{result}[1][1] = 1$, $\text{result}[2][1] = -1$

- Với $i = 2$, $\text{seq}[2] = 5$, $\text{cnt} = 3$, $\text{hit} = 0$.

Hàm search trả về kết quả False, $\text{cnt} < \text{fn}$ trả về False nên $j = 3$ (hàm find), $\text{pf}[2] = 5$, $\text{result}[3][2] = -3$, $\text{result}[0][2] = 2$, $\text{result}[1][2] = 1$, $\text{result}[2][2] = 5$.

- Với $i = 3$, $\text{seq}[3] = 2$, $\text{cnt} = 3$, $\text{hit} = 0$.

Hàm search trả về kết quả True, $\text{hit} = 1$, $\text{result}[3][3] = -2$, $\text{result}[0][3] = 2$, $\text{result}[1][3] = 1$, $\text{result}[2][3] = 5$.

- Với $i = 4$, $\text{seq}[4] = 0$, $\text{cnt} = 3$, $\text{hit} = 1$.

Hàm search trả về kết quả False, $\text{result}[3][4] = -3$, $j = 0$, $\text{result}[0][4] = 0$, $\text{result}[1][4] = 1$, $\text{result}[2][4] = 5$.

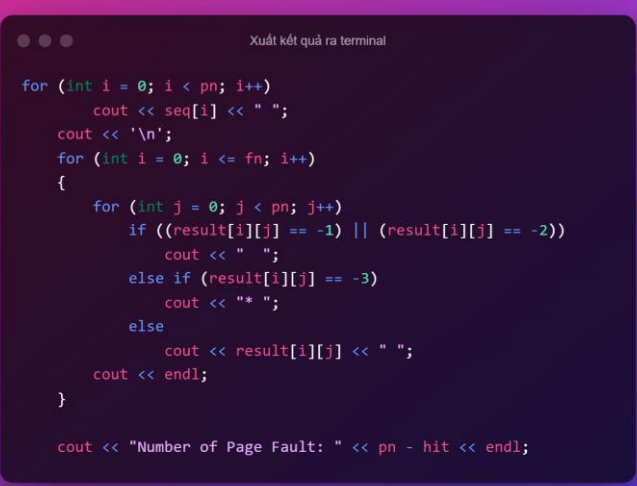
- Với $i = 5$, $\text{seq}[5] = 4$, $\text{cnt} = 3$, $\text{hit} = 2$.

Hàm search trả về kết quả False, $\text{cnt} < \text{fn}$ trả về False nên $j = 1$ (hàm find), $\text{pf}[1] = 4$, $\text{result}[3][5] = -3$, $\text{result}[0][5] = 0$, $\text{result}[1][5] = 4$, $\text{result}[2][5] = 5$.

- Với $i = 6$, $\text{seq}[6] = 9$, $\text{cnt} = 3$, $\text{hit} = 2$.

Hàm search trả về kết quả False, $\text{cnt} < \text{fn}$ trả về False nên $j = 1$ (hàm find), $\text{pf}[1] = 9$, $\text{result}[0][6] = 0$, $\text{result}[1][6] = 9$, $\text{result}[2][6] = 5$.

- Với $i = 7$, $\text{seq}[7] = 7$, $\text{cnt} = 3$, $\text{hit} = 2$.
Hàm search trả về kết quả False,

	<p>cnt < fn trả về False nên j = 1(hàm find), pf[1] = 7, result[0][7] = 0, result[1][7] = 7, result[2][7] = 5.</p> <p>-Với i = 8, seq[8] = 0, cnt = 3, hit = 2. Hàm search trả về kết quả True, hit = 3, result[3][8] = -2, result[0][8] = 0, result[1][8] = 7, result[2][8] = 5.</p> <p>-Với i = 9, seq[9] = 0, cnt = 3, hit = 3. Hàm search trả về kết quả True, hit = 4, result[3][9] = -2, result[0][9] = 0, result[1][9] = 7, result[2][9] = 5.</p> <p>-Với i = 10, seq[10] = 7, cnt = 3, hit = 4. Hàm search trả về kết quả True, hit = 5, result[3][10] = -2, result[0][10] = 0, result[1][10] = 7, result[2][10] = 5.</p>
 <pre>for (int i = 0; i < pn; i++) cout << seq[i] << " "; cout << '\n'; for (int i = 0; i <= fn; i++) { for (int j = 0; j < pn; j++) if ((result[i][j] == -1) (result[i][j] == -2)) cout << " "; else if (result[i][j] == -3) cout << "** "; else cout << result[i][j] << " "; cout << endl; } cout << "Number of Page Fault: " << pn - hit << endl;</pre>	<p>Kết quả được xuất ra terminal như sau:</p> <p>2 1 5 2 0 4 9 0 0 0 7 2 2 2 2 0 0 0 0 0 0 7 1 1 1 1 4 9 9 9 9 9 5 5 5 5 5 5 5 5 5 * * * * * Number of Page Fault: 7</p>

4.1 Kiểm tra bằng cách chạy tay

- Trường hợp Default: chuỗi tham chiếu 2 1 5 2 0 4 9 7 0 0 7, frame: 3

2	1	5	2	0	4	9	7	0	0	7
2	2	2	2	0	0	0	0	0	0	0
	1	1	1	1	4	9	7	7	7	7
		5	5	5	5	5	5	5	5	5
*	*	*		*	*	*	*			
Number of Page Fault: 7										

Hình 3. 6 Kết quả chạy tay trường hợp Default

```
minhtriet-21520497@minhtriet21520497-VirtualBox:~$ ./opt
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
Enter your selection (1 or 2): 1
Input page frames: 3
2 1 5 2 0 4 9 7 0 0 7
2 2 2 2 0 0 0 0 0 0 0
  1 1 1 1 4 9 7 7 7 7
    5 5 5 5 5 5 5 5 5
  * * * * *
Number of Page Fault: 7
```

Hình 3. 7 Kết quả chạy chương trình trường hợp Default

⇒ Ta thấy, chương trình thực hiện đúng với trường hợp này

- Trường hợp Manual, test case 2
- Chuỗi tham chiếu : 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1, frame: 3

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	1	1	1	1	1	1	1	1
Number of Page Fault: 9																			

Hình 3. 8 Kết quả chạy tay trường hợp test case 2

```
minhtriet-21520497@minhtriet21520497-VirtualBox:~$ ./opt
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
Enter your selection (1 or 2): 2
Enter length of input sequence: 20
Enter input sequence:
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Input page frames: 3
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 3 1 1 1 1 1 1 1
  * * * * *
Number of Page Fault: 9
```

Hình 3. 9 Kết quả chạy chương trình trường hợp test case 2

- Trường hợp Manual, test case 3
- Chuỗi tham chiếu: 6 2 4 4 5 6 3 1 4 2 3 7 5 6 7 2 4 3 5 1, frame: 3

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	7	7	7	7	7	4	4	4	1
	2	2	2	5	5	5	1	1	2	2	2	2	2	2	2	2	3	3	3
.	.	4	4	4	4	4	4	4	4	4	4	5	6	6	6	6	6	5	5
Number of Page Fault: 14																			

Hình 3. 10 Kết quả chạy tay trường hợp test case 3

```

minhtriet-21520497@minhtriet21520497-VirtualBox: ~
minhtriet-21520497@minhtriet21520497-VirtualBox:~$ ./opt
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
Enter your selection (1 or 2): 2
Enter length of input sequence: 20
Enter input sequence:
6 2 4 4 5 6 3 1 4 2 3 7 5 6 7 2 4 3 5 1
Input page frames: 3
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 7 7 7 7 7 4 3 5 1
  2 2 2 5 5 5 1 1 2 2 2 2 2 2 2 2 2 2 2 2
    4 4 4 4 4 4 4 4 4 4 5 6 6 6 6 6 6 6 6 6
* * * * *
Number of Page Fault: 14
minhtriet-21520497@minhtriet21520497-VirtualBox:~$

```

Hình 3. 11 Kết quả chạy chương trình test case 3

=> Ta thấy, chương trình thực hiện đúng với trường hợp này

6.5. Bài tập ôn tập

Câu 1: Nghịch lý Belady là gì? Sử dụng chương trình đã viết trên để chứng minh nghịch lý.

- Khái niệm: nghịch lý Belady là hiện tượng số lỗi trang tăng lên mặc dù số khung trang được sử dụng tăng lên.
- Chứng minh: Xét chuỗi tham chiếu 1 2 3 4 1 2 5 1 2 3 4 5, số frame lần lượt là 3 và 4 giải thuật thay trang được sử dụng là FIFO, ta có kết quả như sau:
- Với kết quả trên, ta thấy rằng với số frame là 3, ta có tất cả 9 lỗi trang

```
minhtriet-21520497@minhtriet21520497-VirtualBox: ~  
minhtriet-21520497@minhtriet21520497-VirtualBox:~$ gedit FIFO.cpp  
minhtriet-21520497@minhtriet21520497-VirtualBox:~$ g++ FIFO.cpp -o fifo  
minhtriet-21520497@minhtriet21520497-VirtualBox:~$ ./fifo  
--- Page Replacement algorithm ---  
1. Default referenced sequence  
2. Manual input sequence  
Enter your selection: 2  
Enter length of input sequence: 12  
Enter input sequence:  
1 2 3 4 1 2 5 1 2 3 4 5  
Input page frames: 3  
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  
minhtriet-21520497@minhtriet21520497-VirtualBox:~$
```

Hình 4. 1 Thực hiện trên chuỗi trên với số frame là 3

- Với kết quả trên, ta thấy rằng với số frame là 4, ta có tất cả 10 lỗi trang.

```
minhtriet-21520497@minhtriet21520497-VirtualBox:~$ ./fifo  
--- Page Replacement algorithm ---  
1. Default referenced sequence  
2. Manual input sequence  
Enter your selection: 2  
Enter length of input sequence: 12  
Enter input sequence:  
1 2 3 4 1 2 5 1 2 3 4 5  
Input page frames: 4  
1 2 3 4 1 2 5 1 2 3 4 5  
1 1 1 1 1 1 5 5 5 5 4 4  
  2 2 2 2 2 2 1 1 1 1 5  
    3 3 3 3 3 3 2 2 2 2  
      4 4 4 4 4 4 3 3 3  
* * * * *  
Number of Page Fault: 10  
minhtriet-21520497@minhtriet21520497-VirtualBox:~$
```

Hình 4. 2 Thực hiện chuỗi trên với số frame là 4

- Từ kết quả thực tiễn, ta có thể kết luận rằng nghịch lý Belady có tồn tại.

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

Về mặt lý thuyết, giải thuật thay trang có chức năng chính là tìm ra một trang hy sinh sao cho tổng số lỗi trang sinh ra là ít nhất, do đó có thể nhận xét mức độ hiệu quả và tính khả thi của các giải thuật trên như sau:

- FIFO: về mặt cài đặt rất dễ được thực hiện, tính khả thi cao. Tuy nhiên xét về độ hiệu quả thì FIFO dường như cho độ hiệu quả kém nhất khi chọn trang hy sinh là trang được nạp vào đầu tiên nên không đánh giá được mức độ quan trọng, tần số sử dụng của trang, do đó giải thuật cho tính hiệu quả kém.
- LRU: Về mặt cài đặt phức tạp, sử dụng nhiều phần cứng và cần sự hỗ trợ rất nhiều từ hệ điều hành nên tính khả thi khá thấp. Tuy nhiên giải thuật này có phần hiệu quả hơn FIFO do nó xác định trang thay thế dựa vào thời gian tham chiếu, qua đó có thể đánh giá được mức độ quan trọng của trang hay tần số sử dụng của nó, giúp hạn chế việc thay thế nhầm trang.
- OPT: về mặt cài đặt có vẻ không khả thi do phải chọn thời gian tham chiếu ở tương lai. Tuy nhiên, về mặt hiệu quả đây là giải thuật hiệu quả nhất, nó đánh giá đúng việc chọn trang hy sinh sao cho tối ưu nhất, do đó có thể nói đây là giải thuật hiệu quả nhất.

❖ Giải thuật nào là bất khả thi nhất? Vì sao?

Giải thuật bất khả thi nhất là OPT, bởi lẽ do không phải lúc nào ta cũng biết trước được chuỗi tham chiếu để thực thi giải thuật này, trong trường hợp không biết trước thì việc chọn trang hy sinh sẽ rất rủi ro và dường như không thể làm được.

❖ Giải thuật nào là phức tạp nhất? Vì sao?

Giải thuật phức tạp nhất là LRU, do LRU cần sự hỗ trợ của phần cứng và chi phí cho việc tìm kiếm là cao. Ít CPU cung cấp đủ sự hỗ trợ phần cứng cho giải thuật LRU.

.....HẾT.....