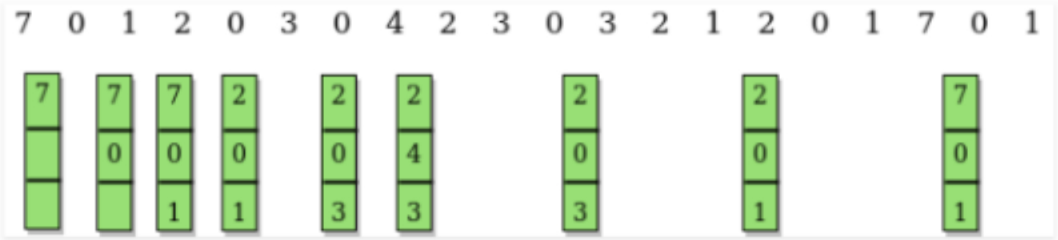
Aim:

Aim: Simulate Optimal page replacement algorithms.

Description: In operating systems, whenever a new page is referred to and not present in memory, a page fault occurs and Operating System replaces one of the existing pages with a newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce the number of page faults.
In this algorithm, OS replaces the page that will not be used for the longest period of time in future.



Algorithm:

1. Start
2. Read the number of frames
3. Read the number of pages
4. Read the page numbers
5. Initialize the values in frames to -1
6. Allocate the pages in to frames by selecting the page that will not be used for the longest period of time.
7. Display the number of page faults.
8. Stop

Source Code:

OptimalPage.c

```
#include<stdio.h>
#include<conio.h>
int i,j,nof,nor,flag=0,ref[50],frm[50],pf=0,victim=-1;
int recent[10],optcal[50],count=0;
int optvictim();
void main()
{
   printf("OPTIMAL PAGE REPLACEMENT ALGORITHM");
   printf("\nEnter the no.of frames: ");
   scanf("%d",&nof);
   printf("Enter the no.of reference string: ");
   scanf("%d",&nor);
   printf("Enter the reference string: ");
   for(i=0;i<nor;i++)
      scanf("%d",&ref[i]);
   printf("OPTIMAL PAGE REPLACEMENT ALGORITHM");
   printf("\nThe given string:");
   for(i=0;i<nor;i++)
      printf("%4d",ref[i]);
   for(i=0;i<nof;i++)
```

```
      {
         frm[i]=-1;
         optcal[i]=0;
      }
   for(i=0;i<10;i++)
      recent[i]=0;
   printf("\n");
   for(i=0;i<nor;i++)
      {
         flag=0;
         printf("Reference NO %d ->\t",ref[i]);
         for(j=0;j<nof; j++)
            {
               if(frm[j]==ref[i])
               {
                  flag=1;
                  break;
               }
            }
         if(flag==0)
         {
            count++;
            if(count<=nof)
               victim++;
            else
               victim=optvictim(i);
            pf++;
            frm[victim]=ref[i];
            for(j=0; j<nof; j++)
               printf("%4d",frm[j]);
            printf("\n");
         }
      }
   printf("Number of page faults: %d",pf);
}
int optvictim(index)
{
   int i,j,temp,notfound;
   for(i=0;i<nof;i++)
      {
         notfound=1;
         for(j=index; j<nor; j++)
            if(frm[i]==ref[j])
            {
               notfound=0;
               optcal[i]=j;
               break;
            }
         if(notfound==1)
            return i;
      }
   temp=optcal[0];
   for(i=1; i<nof; i++)
      if(temp<optcal[i])
         temp=optcal[i];
   for(i=0; i<nof; i++)
```

```
      if(frm[temp]==frm[i])
         return i;
   return 0;
}
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
OPTIMAL PAGE REPLACEMENT ALGORITHM 3
Enter the no.of frames: 3
Enter the no.of reference string: 6
Enter the reference string: 6 5 4 2 3 1
OPTIMAL PAGE REPLACEMENT ALGORITHM
The given string: 6 5 4 2 3 1
Reference NO 6 -> 6 -1 -1
Reference NO 5 -> 6 5 -1
Reference NO 4 -> 6 5 4
Reference NO 2 -> 2 5 4
Reference NO 3 -> 3 5 4
Reference NO 1 -> 1 5 4
Number of page faults: 6

Test Case - 2
User Output
OPTIMAL PAGE REPLACEMENT ALGORITHM 4
Enter the no.of frames: 4
Enter the no.of reference string: 5
Enter the reference string: 4 2 5 8 9
OPTIMAL PAGE REPLACEMENT ALGORITHM
The given string: 4 2 5 8 9
Reference NO 4 -> 4 -1 -1 -1
Reference NO 2 -> 4 2 -1 -1
Reference NO 5 -> 4 2 5 -1
Reference NO 8 -> 4 2 5 8
Reference NO 9 -> 9 2 5 8
Number of page faults: 5