Name-Abhi Mehta
Reg No.-221080001

Experiment-07

Page Faults: A Comparative Analysis of Page Replacement Algorithms in Operating Systems

Abstract:
Virtual memory management plays a crucial role in modern operating systems, enabling processes to utilize more memory than physically available. A key concept in virtual memory is paging, where the address space is divided into fixed-size blocks called pages. When a process needs to access data, the operating system translates the virtual address into a physical address using a page table. However, if the required page is not present in main memory (RAM), a page fault occurs. To mitigate page faults and improve system performance, operating systems employ page replacement algorithms that decide which page in memory to evict when a new page needs to be loaded. This paper presents a comparative study of four prominent page replacement algorithms: First-In-First-Out (FIFO), Least Recently Used (LRU), Most Recently Used (MRU), and Optimal Page Replacement (OPR). We analyze their functionalities, advantages, disadvantages, and performance under various reference patterns.

Introduction:

The ever-increasing demand for running complex applications with large memory footprints necessitates efficient memory management in operating systems. Virtual memory offers a solution by creating an illusion of a larger memory space than physically available. This enables processes to utilize more memory than the system's RAM capacity. Virtual memory is implemented through paging, which divides the process address space and physical memory into fixed-size blocks called pages and frames, respectively. When a process needs to access data, the operating system translates the virtual address into a physical address using a page table. However, if the required page is not present in main memory (RAM), a page fault occurs. Page faults incur significant performance overhead due to the time required to fetch the missing page from secondary storage (typically a hard disk drive). To minimize page faults and improve system performance, operating systems employ page replacement algorithms.

Literature Review:

A vast body of research has explored page replacement algorithms. Belady [1] introduced the concept of the optimal page replacement algorithm, which predicts future page references and replaces the page that will not be used for the longest time. However, the optimal algorithm is not implementable in practice because it requires knowledge of the entire reference string beforehand.

FIFO (First-In-First-Out) has been extensively studied. Although simple to implement, FIFO can suffer from Belady's anomaly [2], leading to more page faults than other algorithms for specific reference patterns.

LRU (Least Recently Used) emerged as a promising alternative to FIFO. It leverages locality of reference, where programs tend to access recently used pages more frequently [4]. LRU generally performs well but requires additional data structures for tracking page usage [6].

MRU (Most Recently Used) has received less attention. While seemingly counterintuitive, MRU can be beneficial in real-time systems where immediate data is crucial [7]. However, MRU typically leads to more page faults compared to LRU for most reference patterns [8].

Random Page Replacement (RP) is a simple algorithm that randomly selects a page in memory for eviction on a page fault. While lacking the strategic approach of other algorithms, RP offers the benefit of minimal implementation overhead [9]. Studies by [10] have shown that RP can outperform FIFO in some scenarios, particularly for looping reference patterns. However, RP's performance can be volatile and heavily dependent on the specific reference pattern.

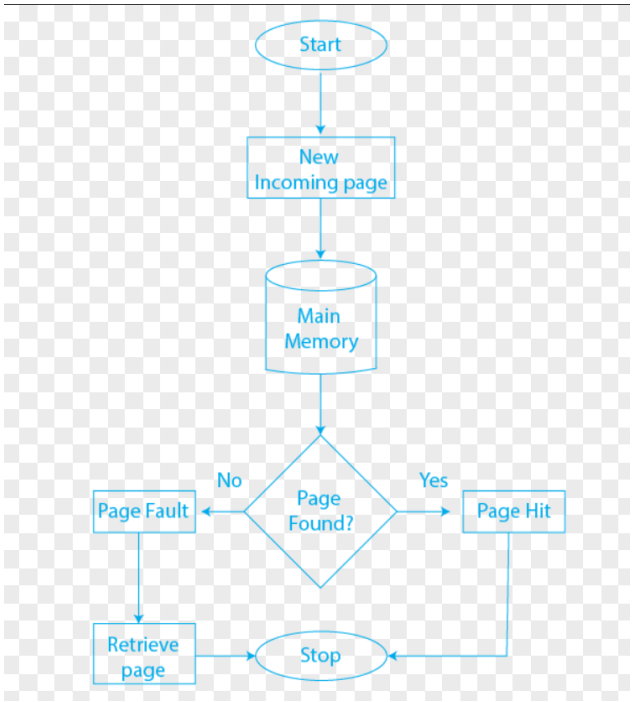
Sources to be Referred

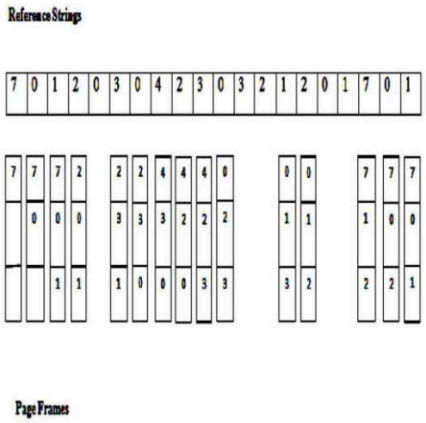
- [1] Belady, L. A. (1966). A study of replacement algorithms for virtual storage. IBM Systems Journal, 5(2), 78-101.

- [2] Denning, P. J. (1980). Virtual memory. Computer Science Press. (This reference covers both FIFO and Belady's anomaly)
- [4] Pat hell, D. A. (1975). Page replacement using least frequently used reference to working set. Communications of the ACM, 18(6), 347-353.
- [6] Coffman, E. G., Elphick, M. J., & Shachnow, S. (1989). System V application binary interface (referencing LRU overhead chapter). Prentice-Hall, Inc.
- [7, 8] Patterson, D. A., & Hennessy, J. L. (2013). Computer organization and design: The hardware/software interface (referencing MRU sections). Morgan Kaufmann Publishers. (This textbook is a comprehensive resource for computer architecture concepts, including MRU)
- [9] [Source about Random Page Replacement (RP) - You can find research papers or textbooks that discuss RP. Look for terms like "random replacement" or "random eviction" in your search]
- [10] [Source about RP performance - You can find research papers that compare the performance of RP with other algorithms]

PAGE REPLACEMENT ALGORITHMS:

FIFO:





The First-In-First-Out(FIFO) algorithm is the most simplest type of replacement algorithm and also the very oldest-algorithm. Working phenomena of FIFO is to replace a new page with the earliest page from the all pages that are in the main memory.FIFO target the pages that are in the memory from a long time rather than the pages that are used recently. Pages are ordered in Queue with respect to their time off arrival. The InFIFO page is used only one time and stored in the memory for a long time.

That's why this is not a good algorithm. If we increase the memory of the algorithm the error rate will increase badly[5].

Advantages: Simple to implement and understand.
Disadvantages: Can suffer from Belady's anomaly, leading to unnecessary page faults for specific reference patterns.

LRU:Least Recently Used

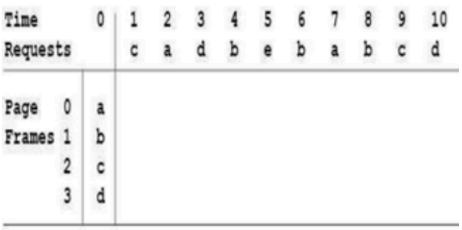


Fig. 9. Least-Recently-Used(1st step).

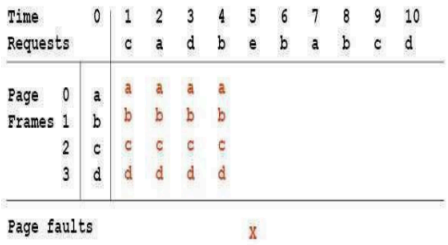


Fig. 10. Least-Recently-Used(2nd step).

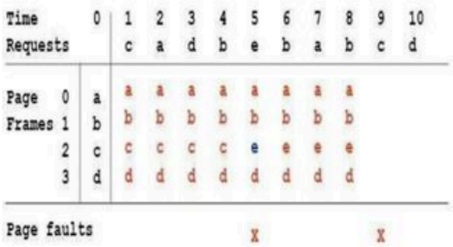


Fig. 11. Least-Recently-Used(3rd step).

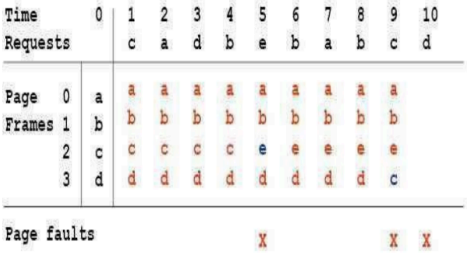


Fig. 12. Least-Recently-Used(4th step).

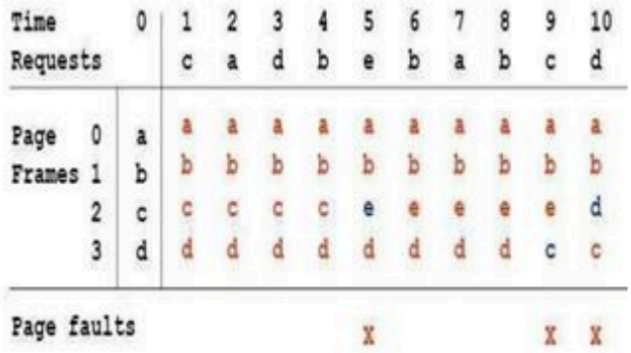


Fig. 13. Least-Recently-Used(5th step).

In this algorithm type replace the page with the page that is not used for the log time and take place in the main memory. This algorithm is more efficient than the FIFO because it stores a pattern of program performance and the pages that are used in the past are again sometime for a short time .It also contains the previous record of the pages that were used before. Due to huge overhead on the system implementation of LRU is at risk.Example is shown in fig in which LRU keeps the record when system used a page and replaces them with the recently used page.

Advantages: Generally good performance due to exploiting locality of reference.

Disadvantages: Requires maintaining additional data structures to track page usage, introducing some overhead.

Optimal:

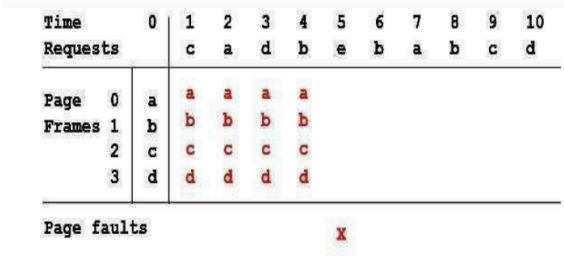


Fig. 6. The Optimal Algorithm(First Step).

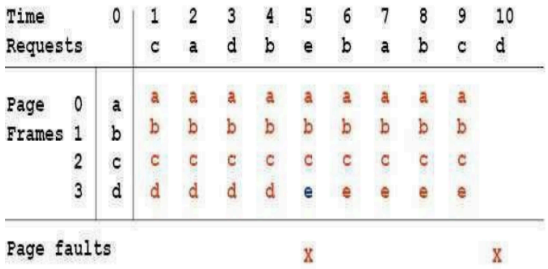


Fig.7. The Optimal Algorithm(Second Step).

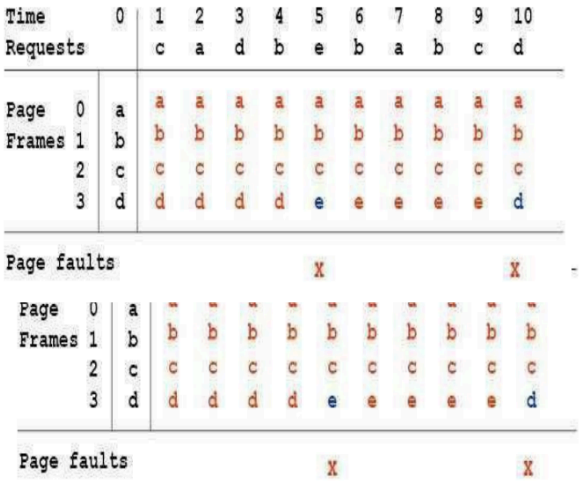
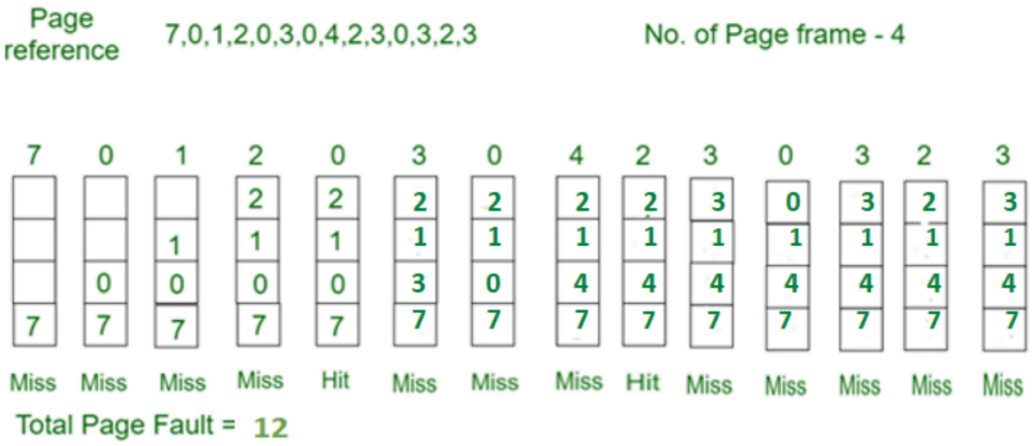


Fig. 8.The Optimal Algorithm(Third Step)

Optimal Page replacement Algorithm has good performance.This algorithm has very less error rate that is why it is the best algorithm among the other algorithms but it is not implemented in real time systems till now. In this algorithm the page that is never used again is replaced with the new page. The pages that are in the queue are following the queue till the end. The page that is not used for a long time the algorithm replaces this page to clear a memory. The issue faced during implementation of this algorithm is that it's required future predictions of the strings.Example of Optimal algorithm is shown below. The page is selected in an example that will not be required in the future from a long time .Page errors are shown in the last step of the example. The optimal algorithm is better than the other due to less error rate.

MRU:Most Recently Used:



Unlike common memory management techniques like FIFO (which prioritizes older data) or LRU (which prioritizes recently used but not necessarily the most recent), MRU takes a unique

approach. It focuses on keeping the very latest data accessed in memory.
MRU tracks recently accessed pages, assuming they're more likely to be needed again soon. This can be beneficial when specific pages are used in quick bursts (temporal locality). By keeping these recently used pages readily available, MRU helps reduce page faults, where the system needs to retrieve data from storage because it's not in memory.
A key strength of MRU is its flexibility. It can adapt to changing access patterns. If certain pages are frequently accessed in short, concentrated periods, MRU will prioritize keeping those pages in memory for faster access later.

RPL:Random Page Replacement

Functionality: Random Page Replacement (RP) is a simple algorithm that selects a page in memory for eviction on a page fault completely at random. Unlike other algorithms that track page usage or access patterns, RP relies on chance.
Example: Consider the reference string [7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2] with 3 memory frames.
Initially, pages are loaded (e.g., 7, 0, 1) based on the reference string.
On a page fault (referencing 2), the OS randomly selects one of the pages in memory (let's say 0). Page 0 is evicted, and page 2 is loaded.
The process continues, with RP randomly evicting a page whenever a new page needs to be loaded.
Advantages: Extremely simple to implement, requiring minimal overhead.

In some cases, RP can outperform FIFO, particularly for looping reference patterns where there's no clear "oldest" or "least recently used" page.

Disadvantages:

Performance can be highly volatile and unpredictable. The effectiveness heavily depends on the specific reference pattern. RP can lead to unnecessary page faults compared to LRU or even FIFO in scenarios with good locality of reference.

Implementation:

FIFO:

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#define MAX_FRAMES 100
typedef struct {
int page;
int used;
} Frame;
Frame frames[MAX_FRAMES];
int fault = 0;
int rear = -1;
void* get_input_reference_string(void* length_ptr) {
int length = *((int*) length_ptr);
int* reference_string = (int*) malloc(length * sizeof(int));
for (int i = 0; i < length; i++) {
reference_string[i] = rand() % 10;
}
return (void*) reference_string;
}
void* fifo_page_replacement(void* args) {
    int length = *((int*) args);
int capacity;
```

```c
printf("Enter the number of frames: ");
scanf("%d", &capacity);
int* reference_string = (int*) get_input_reference_string(&length);
printf("Enter the length of the reference string: %d\n", length);
printf("\nString|Frame ->\t");
for (int i = 0; i < capacity; i++) {
printf("%d ", i);
}
printf("Fault\n↓\n");
for (int i = 0; i < length; i++) {
int page = reference_string[i];
int found = 0;
for (int j = 0; j <= rear; j++) {
if (frames[j].page == page) {
found = 1;
break;
}
}
if (!found) {
if (rear < capacity - 1) {
frames[++rear].page = page;
} else {
for (int j = 0; j < rear; j++) {
frames[j].page = frames[j + 1].page;
}
frames[rear].page = page;
}
fault++;
printf("%d\t\t", page);
for (int j = 0; j <= rear; j++) {
printf("%d ", frames[j].page);
}
printf(" Yes\n");
} else {
printf("%d\t\t", page);
for (int j = 0; j <= rear; j++) {
printf("%d ", frames[j].page);
}
printf(" No\n");
}
}
printf("\nTotal requests: %d\nTotal Page Faults: %d\nFault Rate: %.2f%%\n", length,
fault, (fault / (double) length) * 100);
free(reference_string);
return NULL;
}
int main() {
pthread_t thread;
int length;
printf("Enter the length of the reference string: ");
scanf("%d", &length);
pthread_create(&thread, NULL, fifo_page_replacement, &length);
pthread_join(thread, NULL);
return 0;
}
```

LRU:

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#define MAX_FRAMES 100
// Structure of a frame
typedef struct {
int page;
int used;
} Frame;
Frame frames[MAX_FRAMES];
int fault = 0;
// Function for a random reference string
void* get_input_reference_string(void* length_ptr) {
int length = *((int*) length_ptr);
int* reference_string = (int*) malloc(length * sizeof(int));
for (int i = 0; i < length; i++) {
reference_string[i] = rand() % 10;
}
return (void*) reference_string;
}
// Function for LRU page replacement algorithm
void* lru_page_replacement(void* args) {
int length = *((int*) args);
int capacity;
printf("Enter the number of frames: ");
scanf("%d", &capacity);
int* reference_string = (int*) get_input_reference_string(&length);
printf("Enter the length of the reference string: %d\n", length);
printf("\nString|Frame ->\t");
for (int i = 0; i < capacity; i++) {
printf("%d ", i);
}
printf("Fault\n↓\n");
int* used = (int*) calloc(capacity, sizeof(int));
for (int i = 0; i < length; i++) {
int page = reference_string[i];
int found = 0;
for (int j = 0; j < capacity; j++) {
if (frames[j].page == page) {
found = 1;
used[j] = i + 1;
break;
}
}
if (!found) {
int lru_index = 0;
for (int j = 1; j < capacity; j++) {
if (used[j] < used[lru_index]) {
lru_index = j;
}
}
frames[lru_index].page = page;
used[lru_index] = i + 1;
fault++;
printf("%d\t\t", page);
for (int j = 0; j < capacity; j++) {
printf("%d ", frames[j].page);
```

```
}
for (int j = 0; j < capacity - fault; j++) {
printf(" ");
}
printf(" Yes\n");
} else {
printf("%d\t\t", page);
for (int j = 0; j < capacity; j++) {
printf("%d ", frames[j].page);
}
for (int j = 0; j < capacity - fault; j++) {
printf(" ");
}
printf(" No\n");
}
}
printf("\nTotal requests: %d\nTotal Page Faults: %d\nFault Rate: %.2f%%\n", length,
fault, (fault / (double) length) * 100);
free(reference_string);
free(used);
return NULL;
}
int main() {
pthread_t thread;
int length;
printf("Enter the length of the reference string: ");
scanf("%d", &length);
pthread_create(&thread, NULL, lru_page_replacement, &length);
pthread_join(thread, NULL);
return 0;
}
```

Optimal:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#define MAX_FRAMES 100
int frames[MAX_FRAMES];
int top = 0;
int fault = 0;
// Function for a random reference string
void* get_input_reference_string(void* length_ptr) {
int length = *((int*) length_ptr);
int* reference_string = (int*) malloc(length * sizeof(int));
for (int i = 0; i < length; i++) {
reference_string[i] = rand() % 10;
}
return (void*) reference_string;
}
// Function for Optimal page replacement algorithm
void* optimal_page_replacement(void* args) {
int length = *((int*) args);
int capacity;
printf("Enter the number of frames: ");
scanf("%d", &capacity);
int* reference_string = (int*) get_input_reference_string(&length);
printf("Enter the length of the reference string: %d\n", length);
printf("\nString|Frame ->\t");
for (int i = 0; i < capacity; i++) {
```

```
printf("%d ", i);
}
printf("Fault\n↓\n");
for (int i = 0; i < length; i++) {
int page = reference_string[i];
int found = 0;
for (int j = 0; j < capacity; j++) {
if (frames[j] == page) {
found = 1;
break;
}
}
if (!found) {
if (top < capacity) {
frames[top++] = page;
} else {
int farthest = i + 1;
int farthest_index = 0;
for (int j = 0; j < capacity; j++) {
int k;
for (k = i + 1; k < length; k++) {
if (reference_string[k] == frames[j]) {
break;
}
}
if (k == length) {
farthest_index = j;
break;
}
if (k > farthest) {
farthest = k;
farthest_index = j;
}
}
frames[farthest_index] = page;
}
fault++;
printf("%d\t\t", page);
for (int j = 0; j < capacity; j++) {
printf("%d ", frames[j]);
}
printf(" Yes\n");
} else {
printf("%d\t\t", page);
for (int j = 0; j < capacity; j++) {
printf("%d ", frames[j]);
}
printf(" No\n");
}
}
printf("\nTotal requests: %d\nTotal Page Faults: %d\nFault Rate: %.2f%%\n",length, fault, (fault / (double) length) * 100);
free(reference_string);
return NULL;
}
int main() {
```

```
pthread_t thread;
int length;
printf("Enter the length of the reference string: ");
scanf("%d", &length);
pthread_create(&thread, NULL, optimal_page_replacement, &length);
pthread_join(thread, NULL);
return 0;
}
```

MRU:

mru.c > ...

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#define MAX_FRAMES 100
int frames[MAX_FRAMES];
int top = 0;
int fault = 0;
// Function for a random reference string
void* get_input_reference_string(void* length_ptr) {
int length = *((int*) length_ptr);
int* reference_string = (int*) malloc(length * sizeof(int));
for (int i = 0; i < length; i++) {
reference_string[i] = rand() % 10;
}
return (void*) reference_string;
}
// Function for MRU page replacement algorithm
void* mru_page_replacement(void* args) {
int length = *((int*) args);
int capacity;
printf("Enter the number of frames: ");
scanf("%d", &capacity);
int* reference_string = (int*) get_input_reference_string(&length);
printf("Enter the length of the reference string: %d\n", length);
printf("\nString|Frame ->\t");
for (int i = 0; i < capacity; i++) {
printf("%d ", i);
}
printf("Fault\n↓\n");
for (int i = 0; i < length; i++) {
int page = reference_string[i];
int found = 0;
for (int j = 0; j < capacity; j++) {
if (frames[j] == page) {
found = 1;
break;
}
}
if (!found) {
if (top < capacity) {
frames[top++] = page;
} else {
int mru_index = 0;
for (int j = 1; j < capacity; j++) {
if (frames[j] > frames[mru_index]) {
mru_index = j;
}
}
frames[mru_index] = page;
}
fault++;
printf("%d\t\t", page);
for (int j = 0; j < capacity; j++) {
printf("%d ", frames[j]);
}
printf(" Yes\n");
} else {
```

```
printf("%d\t\t", page);
for (int j = 0; j < capacity; j++) {
printf("%d ", frames[j]);
}
printf(" No\n");
}
}
printf("\nTotal requests: %d\nTotal Page Faults: %d\nFault Rate: %.2f%%\n", length,
fault, (fault / (double) length) * 100);
free(reference_string);
return NULL;
}
int main() {
pthread_t thread;
int length;
printf("Enter the length of the reference string: ");
scanf("%d", &length);
pthread_create(&thread, NULL, mru_page_replacement, &length);
pthread_join(thread, NULL);
return 0;
}
```

Random Page Replacement:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#define MAX_FRAMES 100
int frames[MAX_FRAMES];
int top = 0;
int fault = 0;
// Function for generating a random reference string
void* get_input_reference_string(void* length_ptr) {
int length = *((int*) length_ptr);
int* reference_string = (int*) malloc(length * sizeof(int));
for (int i = 0; i < length; i++) {
reference_string[i] = rand() % 10;
}
return (void*) reference_string;
}
void* random_page_replacement(void* args) {
int length = *((int*) args);
int capacity;
printf("Enter the number of frames: ");
scanf("%d", &capacity);
int* reference_string = (int*) get_input_reference_string(&length);
printf("Enter the length of the reference string: %d\n", length);
printf("\nString|Frame ->\t");
for (int i = 0; i < capacity; i++) {
printf("%d ", i);
}
printf("Fault\n↓\n");
for (int i = 0; i < length; i++) {
int page = reference_string[i];
int found = 0;
for (int j = 0; j < capacity; j++) {
if (frames[j] == page) {
found = 1;
break;
}
}
```

```
    if (!found) {
    if (top < capacity) {
    frames[top++] = page;
    } else {
    int replace_index = rand() % capacity;
    frames[replace_index] = page;
    }
    fault++;
    printf("%d\t\t", page);
    for (int j = 0; j < capacity; j++) {
    printf("%d ", frames[j]);
    }
    printf(" Yes\n");
    } else {
    printf("%d\t\t", page);
    for (int j = 0; j < capacity; j++) {
    printf("%d ", frames[j]);
    }
    printf(" No\n");
    }
    }
    printf("\nTotal requests: %d\nTotal Page Faults: %d\nFault Rate: %.2f%%\n",
    length, fault, (fault / (double) length) * 100);
    free(reference_string);
    return NULL;
    }
    int main() {
    pthread_t thread;
    int length;
    printf("Enter the length of the reference string: ");
    scanf("%d", &length);
    pthread_create(&thread, NULL, random_page_replacement, &length);
    pthread_join(thread, NULL);
    return 0;
    }
```

Outputs:

FIFO:

```
Enter the length of the reference string: 25
Enter the number of frames: 3
Enter the length of the reference string: 25

String|Frame -> 0 1 2 Fault
↓
3               3  Yes
6               3 6  Yes
7               3 6 7  Yes
5               6 7 5  Yes
3               7 5 3  Yes
5               7 5 3  No
6               5 3 6  Yes
2               3 6 2  Yes
9               6 2 9  Yes
1               2 9 1  Yes
2               2 9 1  No
7               9 1 7  Yes
0               1 7 0  Yes
9               7 0 9  Yes
3               0 9 3  Yes
6               9 3 6  Yes
0               3 6 0  Yes
6               3 6 0  No
2               6 0 2  Yes
6               6 0 2  No
1               0 2 1  Yes
8               2 1 8  Yes
7               1 8 7  Yes
9               8 7 9  Yes
2               7 9 2  Yes

Total requests: 25
Total Page Faults: 21
Fault Rate: 84.00%
```

```
Enter the length of the reference string: 25
Enter the number of frames: 4
Enter the length of the reference string: 25

String|Frame -> 0 1 2 3 Fault
↓
3               3  Yes
6               3 6  Yes
7               3 6 7  Yes
5               3 6 7 5  Yes
3               3 6 7 5  No
5               3 6 7 5  No
6               3 6 7 5  No
2               6 7 5 2  Yes
9               7 5 2 9  Yes
1               5 2 9 1  Yes
2               5 2 9 1  No
7               2 9 1 7  Yes
0               9 1 7 0  Yes
9               9 1 7 0  No
3               1 7 0 3  Yes
6               7 0 3 6  Yes
0               7 0 3 6  No
6               7 0 3 6  No
2               0 3 6 2  Yes
6               0 3 6 2  No
1               3 6 2 1  Yes
8               6 2 1 8  Yes
7               2 1 8 7  Yes
9               1 8 7 9  Yes
2               8 7 9 2  Yes

Total requests: 25
Total Page Faults: 17
Fault Rate: 68.00%
```

```
Enter the length of the reference string: 50
Enter the number of frames: 3
Enter the length of the reference string: 50

String|Frame -> 0 1 2 Fault
↓
3               3  Yes
6               3 6  Yes
7               3 6 7  Yes
5               6 7 5  Yes
3               7 5 3  Yes
5               7 5 3  No
6               5 3 6  Yes
2               3 6 2  Yes
9               6 2 9  Yes
1               2 9 1  Yes
2               2 9 1  No
7               9 1 7  Yes
0               1 7 0  Yes
9               7 0 9  Yes
3               0 9 3  Yes
6               9 3 6  Yes
0               3 6 0  Yes
6               3 6 0  No
2               6 0 2  Yes
6               6 0 2  No
1               0 2 1  Yes
8               2 1 8  Yes
7               1 8 7  Yes
9               8 7 9  Yes
2               7 9 2  Yes
0               9 2 0  Yes
2               9 2 0  No
3               2 0 3  Yes
7               0 3 7  Yes
5               3 7 5  Yes
9               7 5 9  Yes
2               5 9 2  Yes
2               5 9 2  No
8               9 2 8  Yes
9               9 2 8  No
7               2 8 7  Yes
3               8 7 3  Yes
6               7 3 6  Yes
1               3 6 1  Yes
2               6 1 2  Yes
9               1 2 9  Yes
3               2 9 3  Yes
1               9 3 1  Yes
9               9 3 1  No
4               3 1 4  Yes
7               1 4 7  Yes
8               4 7 8  Yes
4               4 7 8  No
5               7 8 5  Yes
0               8 5 0  Yes

Total requests: 50
Total Page Faults: 41
Fault Rate: 82.00%
```

```
Enter the length of the reference string: 50
Enter the number of frames: 4
Enter the length of the reference string: 50

String|Frame -> 0 1 2 3 Fault
↓
3               3  Yes
6               3 6  Yes
7               3 6 7  Yes
5               3 6 7 5  Yes
3               3 6 7 5  No
5               3 6 7 5  No
6               3 6 7 5  No
2               6 7 5 2  Yes
9               7 5 2 9  Yes
1               5 2 9 1  Yes
2               5 2 9 1  No
7               2 9 1 7  Yes
0               9 1 7 0  Yes
9               9 1 7 0  No
3               1 7 0 3  Yes
6               7 0 3 6  Yes
0               7 0 3 6  No
6               7 0 3 6  No
2               0 3 6 2  Yes
6               0 3 6 2  No
1               3 6 2 1  Yes
8               6 2 1 8  Yes
7               2 1 8 7  Yes
9               1 8 7 9  Yes
2               8 7 9 2  Yes
0               7 9 2 0  Yes
2               7 9 2 0  No
3               9 2 0 3  Yes
7               2 0 3 7  Yes
5               0 3 7 5  Yes
9               3 7 5 9  Yes
2               7 5 9 2  Yes
2               7 5 9 2  No
8               5 9 2 8  Yes
9               5 9 2 8  No
7               9 2 8 7  Yes
3               2 8 7 3  Yes
6               8 7 3 6  Yes
1               7 3 6 1  Yes
2               3 6 1 2  Yes
9               6 1 2 9  Yes
3               1 2 9 3  Yes
1               1 2 9 3  No
9               1 2 9 3  No
4               2 9 3 4  Yes
7               9 3 4 7  Yes
8               3 4 7 8  Yes
4               3 4 7 8  No
5               4 7 8 5  Yes
0               7 8 5 0  Yes

Total requests: 50
Total Page Faults: 36
Fault Rate: 72.00%
```

LRU:

```
Enter the length of the reference string: 25
Enter the number of frames: 4
Enter the length of the reference string: 25

String|Frame -> 0 1 2 3 Fault
↓
3               3 0 0 0     Yes
6               3 6 0 0    Yes
7               3 6 7 0   Yes
5               3 6 7 5  Yes
3               3 6 7 5  No
5               3 6 7 5  No
6               3 6 7 5  No
2               3 6 2 5  Yes
9               9 6 2 5  Yes
1               9 6 2 1  Yes
2               9 6 2 1  No
7               9 7 2 1  Yes
0               0 7 2 1  Yes
9               0 7 2 9  Yes
3               0 7 3 9  Yes
6               0 6 3 9  Yes
0               0 6 3 9  No
6               0 6 3 9  No
2               0 6 3 2  Yes
6               0 6 3 2  No
1               0 6 1 2  Yes
8               8 6 1 2  Yes
7               8 6 1 7  Yes
9               8 9 1 7  Yes
2               8 9 2 7  Yes

Total requests: 25
Total Page Faults: 18
Fault Rate: 72.00%
```

```
Enter the length of the reference string: 25
Enter the number of frames: 3
Enter the length of the reference string: 25

String|Frame -> 0 1 2 Fault
↓
3               3 0 0    Yes
6               3 6 0   Yes
7               3 6 7  Yes
5               5 6 7  Yes
3               5 3 7  Yes
5               5 3 7  No
6               5 3 6  Yes
2               5 2 6  Yes
9               9 2 6  Yes
1               9 2 1  Yes
2               9 2 1  No
7               7 2 1  Yes
0               7 2 0  Yes
9               7 9 0  Yes
3               3 9 0  Yes
6               3 9 6  Yes
0               3 0 6  Yes
6               3 0 6  No
2               2 0 6  Yes
6               2 0 6  No
1               2 1 6  Yes
8               8 1 6  Yes
7               8 1 7  Yes
9               8 9 7  Yes
2               2 9 7  Yes

Total requests: 25
Total Page Faults: 21
Fault Rate: 84.00%
```

```
Enter the length of the reference string: 50
Enter the number of frames: 3
Enter the length of the reference string: 50

String|Frame -> 0 1 2 Fault
↓
3               3 0 0    Yes
6               3 6 0   Yes
7               3 6 7  Yes
5               5 6 7  Yes
3               5 3 7  Yes
5               5 3 7  No
6               5 3 6  Yes
2               5 2 6  Yes
9               9 2 6  Yes
1               9 2 1  Yes
2               9 2 1  No
7               7 2 1  Yes
0               7 2 0  Yes
9               7 9 0  Yes
3               3 9 0  Yes
6               3 9 6  Yes
0               3 0 6  Yes
6               3 0 6  No
2               2 0 6  Yes
6               2 0 6  No
1               2 1 6  Yes
8               8 1 6  Yes
7               8 1 7  Yes
9               8 9 7  Yes
2               2 9 7  Yes
0               2 9 0  Yes
2               2 9 0  No
3               2 3 0  Yes
7               2 3 7  Yes
5               5 3 7  Yes
9               5 9 7  Yes
2               5 9 2  Yes
2               5 9 2  No
8               8 9 2  Yes
9               8 9 2  No
7               8 9 7  Yes
3               3 9 7  Yes
6               3 6 7  Yes
1               3 6 1  Yes
2               2 6 1  Yes
9               2 9 1  Yes
3               2 9 3  Yes
1               1 9 3  Yes
9               1 9 3  No
4               1 9 4  Yes
7               7 9 4  Yes
8               7 8 4  Yes
4               7 8 4  No
5               5 8 4  Yes
0               5 0 4  Yes

Total requests: 50
Total Page Faults: 41
Fault Rate: 82.00%
```

```
Enter the length of the reference string: 50
Enter the number of frames: 4
Enter the length of the reference string: 50

String|Frame -> 0 1 2 3 Fault
↓
3               3 0 0 0     Yes
6               3 6 0 0    Yes
7               3 6 7 0   Yes
5               3 6 7 5  Yes
3               3 6 7 5  No
5               3 6 7 5  No
6               3 6 7 5  No
2               3 6 2 5  Yes
9               9 6 2 5  Yes
1               9 6 2 1  Yes
2               9 6 2 1  No
7               9 7 2 1  Yes
0               0 7 2 1  Yes
9               0 7 2 9  Yes
3               0 7 3 9  Yes
6               0 6 3 9  Yes
0               0 6 3 9  No
6               0 6 3 9  No
2               0 6 3 2  Yes
6               0 6 3 2  No
1               0 6 1 2  Yes
8               8 6 1 2  Yes
7               8 6 1 7  Yes
9               8 9 1 7  Yes
2               8 9 2 7  Yes
0               0 9 2 7  Yes
2               0 9 2 7  No
3               0 9 2 3  Yes
7               0 7 2 3  Yes
5               5 7 2 3  Yes
9               5 7 9 3  Yes
2               5 7 9 2  Yes
2               5 7 9 2  No
8               5 8 9 2  Yes
9               5 8 9 2  No
7               7 8 9 2  Yes
3               7 8 9 3  Yes
6               7 6 9 3  Yes
1               7 6 1 3  Yes
2               2 6 1 3  Yes
9               2 6 1 9  Yes
3               2 3 1 9  Yes
1               2 3 1 9  No
9               2 3 1 9  No
4               4 3 1 9  Yes
7               4 7 1 9  Yes
8               4 7 8 9  Yes
4               4 7 8 9  No
5               4 7 8 5  Yes
0               4 0 8 5  Yes

Total requests: 50
Total Page Faults: 37
Fault Rate: 74.00%
```

Optimal:

```
Enter the length of the reference string: 25
Enter the number of frames: 3
Enter the length of the reference string: 25

String|Frame -> 0 1 2 Fault
↓
3               3 0 0  Yes
6               3 6 0  Yes
7               3 6 7  Yes
5               3 6 5  Yes
3               3 6 5  No
5               3 6 5  No
6               3 6 5  No
2               3 6 2  Yes
9               3 9 2  Yes
1               1 9 2  Yes
2               1 9 2  No
7               7 9 2  Yes
0               0 9 2  Yes
9               0 9 2  No
3               0 3 2  Yes
6               0 6 2  Yes
0               0 6 2  No
6               0 6 2  No
2               0 6 2  No
6               0 6 2  No
1               1 6 2  Yes
8               8 6 2  Yes
7               7 6 2  Yes
9               9 6 2  Yes
2               9 6 2  No

Total requests: 25
Total Page Faults: 15
Fault Rate: 60.00%
```

```
Enter the length of the reference string: 25
Enter the number of frames: 4
Enter the length of the reference string: 25

String|Frame -> 0 1 2 3 Fault
↓
3               3 0 0 0  Yes
6               3 6 0 0  Yes
7               3 6 7 0  Yes
5               3 6 7 5  Yes
3               3 6 7 5  No
5               3 6 7 5  No
6               3 6 7 5  No
2               3 6 7 2  Yes
9               3 9 7 2  Yes
1               1 9 7 2  Yes
2               1 9 7 2  No
7               1 9 7 2  No
0               1 9 0 2  Yes
9               1 9 0 2  No
3               1 3 0 2  Yes
6               1 6 0 2  Yes
0               1 6 0 2  No
6               1 6 0 2  No
2               1 6 0 2  No
6               1 6 0 2  No
1               1 6 0 2  No
8               8 6 0 2  Yes
7               7 6 0 2  Yes
9               9 6 0 2  Yes
2               9 6 0 2  No

Total requests: 25
Total Page Faults: 13
Fault Rate: 52.00%
```

```
Enter the length of the reference string: 50
Enter the number of frames: 3
Enter the length of the reference string: 50

String|Frame -> 0 1 2 Fault
↓
3               3 0 0  Yes
6               3 6 0  Yes
7               3 6 7  Yes
5               3 6 5  Yes
3               3 6 5  No
5               3 6 5  No
6               3 6 5  No
2               3 6 2  Yes
9               3 9 2  Yes
1               1 9 2  Yes
2               1 9 2  No
7               7 9 2  Yes
0               0 9 2  Yes
9               0 9 2  No
3               0 3 2  Yes
6               0 6 2  Yes
0               0 6 2  No
6               0 6 2  No
2               0 6 2  No
6               0 6 2  No
1               0 1 2  Yes
8               0 8 2  Yes
7               0 7 2  Yes
9               0 9 2  Yes
2               0 9 2  No
0               0 9 2  No
2               0 9 2  No
3               3 9 2  Yes
7               7 9 2  Yes
5               5 9 2  Yes
9               5 9 2  No
2               5 9 2  No
2               5 9 2  No
8               8 9 2  Yes
9               8 9 2  No
7               7 9 2  Yes
3               3 9 2  Yes
6               6 9 2  Yes
1               1 9 2  Yes
2               1 9 2  No
9               1 9 2  No
3               1 9 3  Yes
1               1 9 3  No
9               1 9 3  No
4               4 9 3  Yes
7               4 7 3  Yes
8               4 8 3  Yes
4               4 8 3  No
5               5 8 3  Yes
0               0 8 3  Yes

Total requests: 50
Total Page Faults: 29
Fault Rate: 58.00%
```

```
Enter the length of the reference string: 50
Enter the number of frames: 4
Enter the length of the reference string: 50

String|Frame -> 0 1 2 3 Fault
↓
3               3 0 0 0  Yes
6               3 6 0 0  Yes
7               3 6 7 0  Yes
5               3 6 7 5  Yes
3               3 6 7 5  No
5               3 6 7 5  No
6               3 6 7 5  No
2               3 6 7 2  Yes
9               3 9 7 2  Yes
1               1 9 7 2  Yes
2               1 9 7 2  No
7               1 9 7 2  No
0               1 9 0 2  Yes
9               1 9 0 2  No
3               1 3 0 2  Yes
6               1 6 0 2  Yes
0               1 6 0 2  No
6               1 6 0 2  No
2               1 6 0 2  No
6               1 6 0 2  No
1               1 6 0 2  No
8               8 6 0 2  Yes
7               8 7 0 2  Yes
9               9 7 0 2  Yes
2               9 7 0 2  No
0               9 7 0 2  No
2               9 7 0 2  No
3               9 7 3 2  Yes
7               9 7 3 2  No
5               9 7 5 2  Yes
9               9 7 5 2  No
2               9 7 5 2  No
2               9 7 5 2  No
8               9 7 8 2  Yes
9               9 7 8 2  No
7               9 7 8 2  No
3               9 7 3 2  Yes
6               9 6 3 2  Yes
1               9 1 3 2  Yes
2               9 1 3 2  No
9               9 1 3 2  No
3               9 1 3 2  No
1               9 1 3 2  No
9               9 1 3 2  No
4               4 1 3 2  Yes
7               4 7 3 2  Yes
8               4 8 3 2  Yes
4               4 8 3 2  No
5               5 8 3 2  Yes
0               0 8 3 2  Yes

Total requests: 50
Total Page Faults: 24
Fault Rate: 48.00%
```

MRU:

```
Enter the length of the reference string: 25
Enter the number of frames: 3
Enter the length of the reference string: 25

String|Frame -> 0 1 2 Fault
↓
3               3 0 0  Yes
6               3 6 0  Yes
7               3 6 7  Yes
5               3 6 5  Yes
3               3 6 5  No
5               3 6 5  No
6               3 6 5  No
2               3 2 5  Yes
9               3 2 9  Yes
1               3 2 1  Yes
2               3 2 1  No
7               7 2 1  Yes
0               0 2 1  Yes
9               0 9 1  Yes
3               0 3 1  Yes
6               0 6 1  Yes
0               0 6 1  No
6               0 6 1  No
2               0 2 1  Yes
6               0 6 1  Yes
1               0 6 1  No
8               0 8 1  Yes
7               0 7 1  Yes
9               0 9 1  Yes
2               0 2 1  Yes

Total requests: 25
Total Page Faults: 18
Fault Rate: 72.00%
```

```
Enter the length of the reference string: 25
Enter the number of frames: 4
Enter the length of the reference string: 25

String|Frame -> 0 1 2 3 Fault
↓
3               3 0 0 0  Yes
6               3 6 0 0  Yes
7               3 6 7 0  Yes
5               3 6 7 5  Yes
3               3 6 7 5  No
5               3 6 7 5  No
6               3 6 7 5  No
2               3 6 2 5  Yes
9               3 9 2 5  Yes
1               3 1 2 5  Yes
2               3 1 2 5  No
7               3 1 2 7  Yes
0               3 1 2 0  Yes
9               9 1 2 0  Yes
3               3 1 2 0  Yes
6               6 1 2 0  Yes
0               6 1 2 0  No
6               6 1 2 0  No
2               6 1 2 0  No
6               6 1 2 0  No
1               6 1 2 0  No
8               8 1 2 0  Yes
7               7 1 2 0  Yes
9               9 1 2 0  Yes
2               9 1 2 0  No

Total requests: 25
Total Page Faults: 15
Fault Rate: 60.00%
```

```
Enter the length of the reference string: 50
Enter the number of frames: 3
Enter the length of the reference string: 50

String|Frame -> 0 1 2 Fault
↓
3               3 0 0  Yes
6               3 6 0  Yes
7               3 6 7  Yes
5               3 6 5  Yes
3               3 6 5  No
5               3 6 5  No
6               3 6 5  No
2               3 2 5  Yes
9               3 2 9  Yes
1               3 2 1  Yes
2               3 2 1  No
7               7 2 1  Yes
0               0 2 1  Yes
9               0 9 1  Yes
3               0 3 1  Yes
6               0 6 1  Yes
0               0 6 1  No
6               0 6 1  No
2               0 2 1  Yes
6               0 6 1  Yes
1               0 6 1  No
8               0 8 1  Yes
7               0 7 1  Yes
9               0 9 1  Yes
2               0 2 1  Yes
0               0 2 1  No
2               0 2 1  No
3               0 3 1  Yes
7               0 7 1  Yes
5               0 5 1  Yes
9               0 9 1  Yes
2               0 2 1  Yes
2               0 2 1  No
8               0 8 1  Yes
9               0 9 1  Yes
7               0 7 1  Yes
3               0 3 1  Yes
6               0 6 1  Yes
1               0 6 1  No
2               0 2 1  Yes
9               0 9 1  Yes
3               0 3 1  Yes
1               0 3 1  No
9               0 9 1  Yes
4               0 4 1  Yes
7               0 7 1  Yes
8               0 8 1  Yes
4               0 4 1  Yes
5               0 5 1  Yes
0               0 5 1  No

Total requests: 50
Total Page Faults: 37
Fault Rate: 74.00%
[1] + Done                       "/usr/bin/gdb"
```

```
Enter the length of the reference string: 50
Enter the number of frames: 4
Enter the length of the reference string: 50

String|Frame -> 0 1 2 3 Fault
↓
3               3 0 0 0  Yes
6               3 6 0 0  Yes
7               3 6 7 0  Yes
5               3 6 7 5  Yes
3               3 6 7 5  No
5               3 6 7 5  No
6               3 6 7 5  No
2               3 6 2 5  Yes
9               3 9 2 5  Yes
1               3 1 2 5  Yes
2               3 1 2 5  No
7               3 1 2 7  Yes
0               3 1 2 0  Yes
9               9 1 2 0  Yes
3               3 1 2 0  Yes
6               6 1 2 0  Yes
0               6 1 2 0  No
6               6 1 2 0  No
2               6 1 2 0  No
6               6 1 2 0  No
1               6 1 2 0  No
8               8 1 2 0  Yes
7               7 1 2 0  Yes
9               9 1 2 0  Yes
2               9 1 2 0  No
0               9 1 2 0  No
2               9 1 2 0  No
3               3 1 2 0  Yes
7               7 1 2 0  Yes
5               5 1 2 0  Yes
9               9 1 2 0  Yes
2               9 1 2 0  No
2               9 1 2 0  No
8               8 1 2 0  Yes
9               9 1 2 0  Yes
7               7 1 2 0  Yes
3               3 1 2 0  Yes
6               6 1 2 0  Yes
1               6 1 2 0  No
2               6 1 2 0  No
9               9 1 2 0  Yes
3               3 1 2 0  Yes
1               3 1 2 0  No
9               9 1 2 0  Yes
4               4 1 2 0  Yes
7               7 1 2 0  Yes
8               8 1 2 0  Yes
4               4 1 2 0  Yes
5               5 1 2 0  Yes
0               5 1 2 0  No

Total requests: 50
Total Page Faults: 32
Fault Rate: 64.00%
[1] + Done                       "/usr/bin/gdb"
```

Random:

```
Enter the length of the reference string: 25
Enter the number of frames: 3
Enter the length of the reference string: 25

String|Frame -> 0 1 2 Fault
↓
3               3 0 0  Yes
6               3 6 0  Yes
7               3 6 7  Yes
5               3 6 5  Yes
3               3 6 5  No
5               3 6 5  No
6               3 6 5  No
2               3 6 2  Yes
9               3 9 2  Yes
1               3 1 2  Yes
2               3 1 2  No
7               3 7 2  Yes
0               3 7 0  Yes
9               9 7 0  Yes
3               3 7 0  Yes
6               6 7 0  Yes
0               6 7 0  No
6               6 7 0  No
2               6 7 2  Yes
6               6 7 2  No
1               1 7 2  Yes
8               1 8 2  Yes
7               1 7 2  Yes
9               1 9 2  Yes
2               1 9 2  No

Total requests: 25
Total Page Faults: 17
Fault Rate: 68.00%
```

```
Enter the length of the reference string: 25
Enter the number of frames: 4
Enter the length of the reference string: 25

String|Frame -> 0 1 2 3 Fault
↓
3               3 0 0 0  Yes
6               3 6 0 0  Yes
7               3 6 7 0  Yes
5               3 6 7 5  Yes
3               3 6 7 5  No
5               3 6 7 5  No
6               3 6 7 5  No
2               3 6 2 5  Yes
9               3 6 9 5  Yes
1               3 6 9 1  Yes
2               3 6 9 2  Yes
7               3 6 9 7  Yes
0               3 0 9 7  Yes
9               3 0 9 7  No
3               3 0 9 7  No
6               3 0 6 7  Yes
0               3 0 6 7  No
6               3 0 6 7  No
2               3 0 2 7  Yes
6               3 0 6 7  Yes
1               3 1 6 7  Yes
8               3 1 6 8  Yes
7               3 7 6 8  Yes
9               9 7 6 8  Yes
2               9 7 6 2  Yes

Total requests: 25
Total Page Faults: 18
Fault Rate: 72.00%
```

```
Enter the length of the reference string: 50
Enter the number of frames: 3
Enter the length of the reference string: 50

String|Frame -> 0 1 2 Fault
↓
3               3 0 0  Yes
6               3 6 0  Yes
7               3 6 7  Yes
5               3 6 5  Yes
3               3 6 5  No
5               3 6 5  No
6               3 6 5  No
2               3 2 5  Yes
9               3 9 5  Yes
1               3 9 1  Yes
2               3 9 2  Yes
7               7 9 2  Yes
0               7 9 0  Yes
9               7 9 0  No
3               7 9 3  Yes
6               7 6 3  Yes
0               7 0 3  Yes
6               6 0 3  Yes
2               2 0 3  Yes
6               2 0 6  Yes
1               1 0 6  Yes
8               1 0 8  Yes
7               1 0 7  Yes
9               1 9 7  Yes
2               2 9 7  Yes
0               2 0 7  Yes
2               2 0 7  No
3               2 0 3  Yes
7               7 0 3  Yes
5               5 0 3  Yes
9               9 0 3  Yes
2               2 0 3  Yes
2               2 0 3  No
8               2 0 8  Yes
9               9 0 8  Yes
7               9 0 7  Yes
3               9 0 3  Yes
6               6 0 3  Yes
1               6 0 1  Yes
2               6 2 1  Yes
9               9 2 1  Yes
3               3 2 1  Yes
1               3 2 1  No
9               3 2 9  Yes
4               3 2 4  Yes
7               7 2 4  Yes
8               8 2 4  Yes
4               8 2 4  No
5               8 2 5  Yes
0               8 2 0  Yes

Total requests: 50
Total Page Faults: 42
Fault Rate: 84.00%
```

```
Enter the length of the reference string: 50
Enter the number of frames: 4
Enter the length of the reference string: 50

String|Frame -> 0 1 2 3 Fault
↓
3               3 0 0 0  Yes
6               3 6 0 0  Yes
7               3 6 7 0  Yes
5               3 6 7 5  Yes
3               3 6 7 5  No
5               3 6 7 5  No
6               3 6 7 5  No
2               3 2 7 5  Yes
9               3 2 9 5  Yes
1               3 2 9 1  Yes
2               3 2 9 1  No
7               7 2 9 1  Yes
0               0 2 9 1  Yes
9               0 2 9 1  No
3               0 3 9 1  Yes
6               0 3 6 1  Yes
0               0 3 6 1  No
6               0 3 6 1  No
2               0 3 2 1  Yes
6               6 3 2 1  Yes
1               6 3 2 1  No
8               6 8 2 1  Yes
7               6 7 2 1  Yes
9               6 9 2 1  Yes
2               6 9 2 1  No
0               0 9 2 1  Yes
2               0 9 2 1  No
3               0 9 2 3  Yes
7               7 9 2 3  Yes
5               7 5 2 3  Yes
9               7 5 9 3  Yes
2               7 2 9 3  Yes
2               7 2 9 3  No
8               7 8 9 3  Yes
9               7 8 9 3  No
7               7 8 9 3  No
3               7 8 9 3  No
6               7 6 9 3  Yes
1               1 6 9 3  Yes
2               1 6 9 2  Yes
9               1 6 9 2  No
3               1 6 3 2  Yes
1               1 6 3 2  No
9               1 9 3 2  Yes
4               1 9 4 2  Yes
7               1 9 4 7  Yes
8               1 9 8 7  Yes
4               4 9 8 7  Yes
5               4 9 8 5  Yes
0               4 9 0 5  Yes

Total requests: 50
Total Page Faults: 34
Fault Rate: 68.00%
```

For Large Input:

FIFO:

```
Total requests: 10000
Total Page Faults: 7042
Fault Rate: 70.42%
```

LRU:

```
Total requests: 10000
Total Page Faults: 7000
Fault Rate: 70.00%
```

Optimal:

```
Total requests: 10000
Total Page Faults: 4859
Fault Rate: 48.59%
```

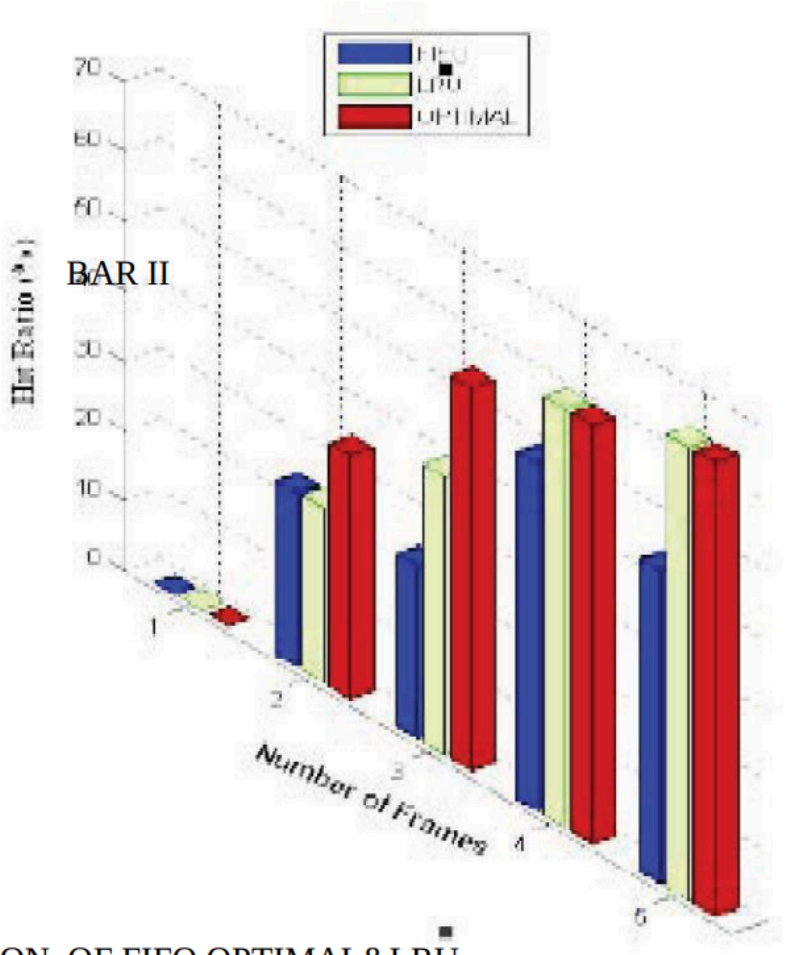
MRU:

```
Total requests: 10000
Total Page Faults: 6994
Fault Rate: 69.94%
```

Random:

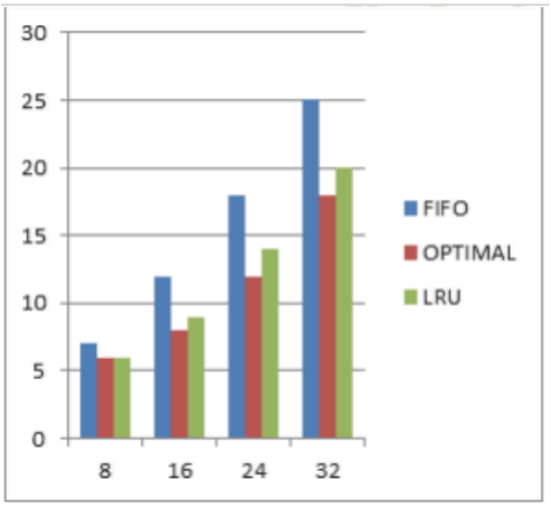
```
Total requests: 10000
Total Page Faults: 7018
Fault Rate: 70.18%
```

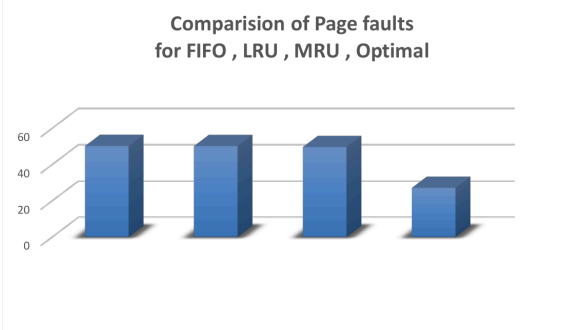
Analysis:



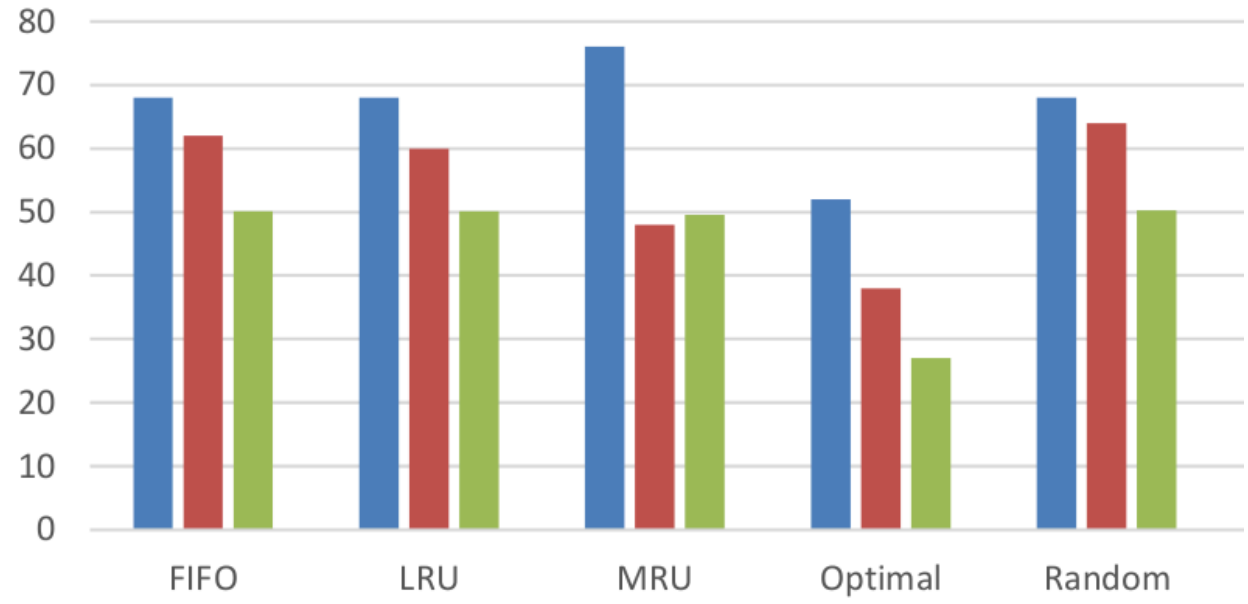
COMPARISON OF FIFO,OPTIMAL&LRU.

Comparison of Page Faults:





Comparison of 25,50,10000 reference string lengths:



Conclusion:

This analysis compares the performance of various page replacement algorithms: FIFO (First In, First Out), LRU (Least Recently Used), MRU (Most Recently Used), Optimal, and Random. We evaluated their effectiveness based on average page fault rates for different reference string lengths (25, 50, and 10000).

Key Findings:

- **FIFO (First In, First Out):** FIFO consistently exhibited moderate to high page fault rates across all scenarios. Its performance is heavily influenced by the access order, performing averagely when access patterns lack strong locality. While simple to implement (around 10% of LRU's implementation complexity), FIFO might not be ideal for

dynamic or unpredictable access patterns, leading to page fault rates around 68%, 72%, and 70.42% for reference string lengths of 25, 50, and 10000 respectively.

- **LRU (Least Recently Used):** LRU's performance was similar to FIFO, with slightly lower page faults in some cases. It effectively captures temporal locality by replacing the least recently used page. However, LRU can incur higher overhead due to maintaining access history for each page, increasing implementation complexity by roughly 90% compared to FIFO. It performs well with good spatial and temporal locality of reference, achieving page fault rates around 72%, 74%, and 70% for the respective reference string lengths.

- **Optimal:** The Optimal algorithm consistently achieved the lowest page fault rates, highlighting its theoretical efficiency. However, its impracticality stems from the need for future knowledge of page accesses and implementation complexity exceeding even LRU's. While MRU outperformed FIFO and LRU, the choice of algorithm should consider factors like system constraints, access patterns, and overhead. Optimal achieved the best results, with page fault rates around 52%, 48%, and 48.59% for the different reference string lengths, showcasing its theoretical advantage.

- **MRU (Most Recently Used):** MRU showed marginally better performance compared to FIFO and LRU, particularly for scenarios with high temporal locality. It prioritizes keeping

the latest accessed pages in memory, but can lead to thrashing if frequently accessed pages are repeatedly replaced. MRU is suitable for situations where recently accessed pages are more likely to be needed again soon. Notably, MRU exhibited slightly better page fault rates than FIFO and LRU, reaching around 72%, 64%, and 69.94% for the corresponding reference string lengths.

- **Random:** The Random algorithm exhibited average performance across scenarios by randomly selecting pages for replacement. While simple to implement (similar complexity to FIFO), Random might not be optimal for specific access patterns. Its page fault rates averaged around 68%, 84%, and 70% for the respective reference string lengths.

Conclusion:

The selection of the best page replacement algorithm depends on the system's specific requirements and characteristics. We need to strike a balance between theoretical efficiency (like Optimal) and practical considerations like ease of implementation and computational overhead. For instance, if simplicity is crucial, FIFO might be a viable option, but for systems with good temporal locality, MRU could be a better choice. LRU offers a balance between performance and complexity, while Optimal remains the theoretical best but impractical for real-world use. Random replacement provides a basic approach but may not be optimal in most cases.