**Page Replacement Algorithm**

* Fifo -> No. of Frame = 3

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **8** | **5** | **7** | **8** | **0** | **2** | **1** | **5** | **4** | **2** | **0** | **1** | **7** |
| 8 | 8 | 8 | 8 | 0 | 0 | 0 | 5 | 5 | 5 | 0 | 0 | 0 |
|  | 5 | 5 | 5 | 5 | 2 | 2 | 2 | 4 | 4 | 4 | 1 | 1 |
|  |  | 7 | 7 | 7 | 7 | 1 | 1 | 1 | 2 | 2 | 2 | 7 |

**8,5,7,8,0,2,1,5,4,2,0,1,7**

Page Fault = No. of Page/Reference String-Page Hit

= 13-1

= 12

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **8** | **5** | **7** | **3** | **0** | **2** | **1** | **5** | **4** | **2** | **8** | **1** | **7** |
| 8 | 8 | 8 | 8 | 8 | 8 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
|  | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 4 | 8 | 8 | 8 |
|  |  | 7 | 3 | 0 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

**Optimal ->**

Page Fault = No. of Page/Reference String-Page Hit

= 13-3

= 10

**LRU (Least Recently Used) ->**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **8** | **5** | **7** | **8** | **0** | **2** | **1** | **5** | **4** | **2** | **8** | **1** | **7** |
| 8 | 8 | 8 | 8 | 8 | 8 | 1 | 1 | 1 | 2 | 2 | 2 | 7 |
|  | 5 | 5 | 5 | 0 | 0 | 0 | 5 | 5 | 5 | 8 | 8 | 8 |
|  |  | 7 | 7 | 7 | 2 | 2 | 2 | 4 | 4 | 4 | 1 | 1 |

Page Fault = No. of Page/Reference String-Page Hit

= 13-1

= 12

**Source Code:**

**Fifo ->**

#include <stdio.h>

#define MAX\_FRAMES 4

void fifoPageReplacement(int pages[], int n, int frameCount)

{

int frames[MAX\_FRAMES];

int pageFaults = 0;

int pageIndex = 0;

int i, j;

for (i = 0; i < frameCount; i++)

{

frames[i] = -1;

}

printf("Page Reference String: ");

for (i = 0; i < n; i++)

{

printf("%d ", pages[i]);

}

printf("\n");

for (i = 0; i < n; i++)

{

int page = pages[i];

int found = 0;

for (j = 0; j < frameCount; j++)

{

if (frames[j] == page)

{

found = 1;

break;

}

}

if (!found)

{

frames[pageIndex] = page;

pageIndex = (pageIndex + 1) % frameCount;

pageFaults++;

printf("Frames after page %d:\n", page);

for (j = 0; j < frameCount; j++)

{

if (frames[j] != -1)

{

printf("%d ", frames[j]);

}

else

{

printf("- ");

}

}

printf("\n");

}

}

printf("Total page faults: %d\n", pageFaults);

}

int main()

{

int n, frameCount, i;

printf("Enter the number of frames: ");

scanf("%d", &frameCount);

printf("Enter the number of pages: ");

scanf("%d", &n);

int pages[n];

printf("Enter the page reference string (space-separated): ");

for (i = 0; i < n; i++)

{

scanf("%d", &pages[i]);

}

fifoPageReplacement(pages, n, frameCount);

return 0;

}

**Optimal ->**

#include <stdio.h>

#include <limits.h>

#define MAX\_FRAMES 4

void optimalPageReplacement(int pages[], int n, int frameCount)

{

int frames[MAX\_FRAMES];

int pageFaults = 0;

int i, j, k;

for (i = 0; i < frameCount; i++)

{

frames[i] = -1;

}

for (i = 0; i < n; i++)

{

int currentPage = pages[i];

int pageFound = 0;

for (j = 0; j < frameCount; j++)

{

if (frames[j] == currentPage)

{

pageFound = 1;

break;

}

}

if (!pageFound)

{

int farthest = -1;

int pageToReplace = -1;

for (j = 0; j < frameCount; j++)

{

int nextUse = -1;

for (k = i + 1; k < n; k++)

{

if (frames[j] == pages[k])

{

nextUse = k;

break;

}

}

if (nextUse == -1)

{

pageToReplace = frames[j];

break;

}

if (nextUse > farthest)

{

farthest = nextUse;

pageToReplace = frames[j];

}

}

for (j = 0; j < frameCount; j++)

{

if (frames[j] == pageToReplace)

{

frames[j] = currentPage;

break;

}

}

pageFaults++;

}

printf("Frames after page %d: ", currentPage);

for (j = 0; j < frameCount; j++)

{

if (frames[j] != -1)

{

printf("%d ", frames[j]);

}

else

{

printf("- ");

}

}

printf("\n");

}

printf("Total page faults: %d\n", pageFaults);

}

int main()

{

int n, frameCount, i;

printf("Enter the number of frames: ");

scanf("%d", &frameCount);

printf("Enter the number of pages: ");

scanf("%d", &n);

int pages[n];

printf("Enter the page reference string (space-separated): ");

for (i = 0; i < n; i++)

{

scanf("%d", &pages[i]);

}

optimalPageReplacement(pages, n, frameCount);

return 0;

}

**Least Recently Used (LRU) ->**

#include <stdio.h>

#include <stdlib.h>

#define MAX\_FRAMES 10

int isPageInFrame(int page, int frames[], int n)

{

int i;

for (i = 0; i < n; i++)

{

if (frames[i] == page)

{

return 1; // Page found in frame

}

}

return 0;

}

int findLRU(int pages[], int pageIndex, int frames[], int n)

{

int i, j;

int lastUsed[MAX\_FRAMES];

for (i = 0; i < n; i++)

{

lastUsed[i] = -1;

}

for (i = 0; i < n; i++)

{

for (j = pageIndex - 1; j >= 0; j--)

{

if (frames[i] == pages[j])

{

lastUsed[i] = j;

break;

}

}

}

int lru = 0;

for (i = 1; i < n; i++)

{

if (lastUsed[i] < lastUsed[lru])

{

lru = i;

}

}

return lru;

}

void LRU(int pages[], int numPages, int numFrames)

{

int frames[MAX\_FRAMES] = {-1, -1, -1};

int pageFaults = 0;

int i, j;

printf("Page reference string: ");

for (i = 0; i < numPages; i++)

{

printf("%d ", pages[i]);

}

printf("\n\n");

printf("Frames: ");

for (i = 0; i < numFrames; i++)

{

printf(" - ");

}

printf("\n");

for (i = 0; i < numPages; i++)

{

printf("\nAccessing page %d\n", pages[i]);

if (!isPageInFrame(pages[i], frames, numFrames))

{

int lruFrame = findLRU(pages, i, frames, numFrames);

frames[lruFrame] = pages[i];

pageFaults++;

}

printf("Frames: ");

for (j = 0; j < numFrames; j++)

{

if (frames[j] != -1)

{

printf("%d ", frames[j]);

}

else

{

printf("- ");

}

}

printf("\n");

}

printf("\nTotal page faults: %d\n", pageFaults);

}

int main()

{

int numPages, numFrames;

int i;

printf("Enter the number of frames: ");

scanf("%d", &numFrames);

printf("Enter the number of pages: ");

scanf("%d", &numPages);

int pages[numPages];

printf("Enter the page reference string: ");

for (i = 0; i < numPages; i++)

{

scanf("%d", &pages[i]);

}

LRU(pages, numPages, numFrames);

return 0;

}