|  |  |
| --- | --- |
| Name: | Prerna Sunil Jadhav |
| Sap  Id: | 60004220127 |
| Class: | T. Y. B.Tech (Computer Engineering) |
| Course: | Processor Organization and Architecture (POA) |
| Course Code: | DJ19CEL502 |
| Experiment  No.: | 03 |

**AIM:** To implement page replacement algorithm: FIFO, OPTIMAL, LRU.

FIFO:

**CODE**:

from queue import Queue

def pageFaults(pages, n, capacity):

s = set()

indexes = Queue()

page\_faults = 0

for i in range(n):

if (len(s) < capacity):

if (pages[i] not in s):

s.add(pages[i])

page\_faults += 1

indexes.put(pages[i])

else:

if (pages[i] not in s):

val = indexes.queue[0]

indexes.get()

s.remove(val)

s.add(pages[i])

indexes.put(pages[i])

page\_faults += 1

print(s)

return page\_faults

if \_\_name\_\_ == '\_\_main\_\_':

pages = [7, 0, 1, 2, 0, 3, 0,

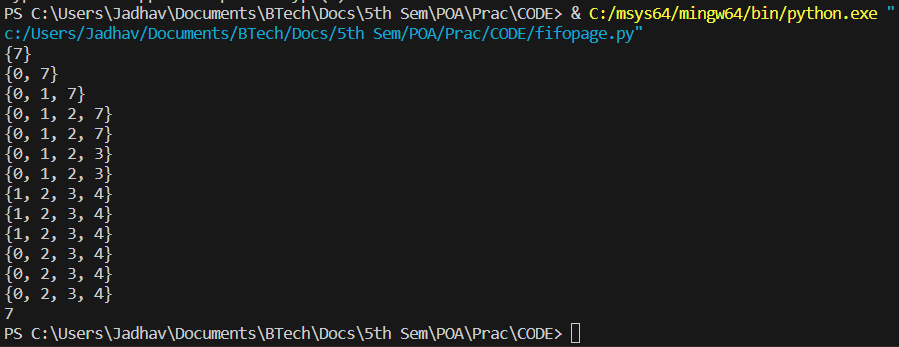
4, 2, 3, 0, 3, 2]

n = len(pages)

capacity = 4

print(pageFaults(pages, n, capacity))

**OUTPUT**:



OPTIMAL:

**CODE**:

def search(key, fr):

for i in range(len(fr)):

if (fr[i] == key):

return True

return False

def predict(pg, fr, pn, index):

res = -1

farthest = index

for i in range(len(fr)):

j = 0

for j in range(index, pn):

if (fr[i] == pg[j]):

if (j > farthest):

farthest = j

res = i

break

if (j == pn):

return i

return 0 if (res == -1) else res

def optimalPage(pg, pn, fn):

fr = []

hit = 0

for i in range(pn):

if search(pg[i], fr):

hit += 1

continue

if len(fr) < fn:

fr.append(pg[i])

else:

j = predict(pg, fr, pn, i + 1)

fr[j] = pg[i]

print(fr)

print("No. of hits =", 7)

print("No. of misses =", 6)

pg = [7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2]

pn = len(pg)

fn = 4

optimalPage(pg, pn, fn)

**OUTPUT**:

A screen shot of a computer

Description automatically generated

LRU:

**CODE**:

def pageFaults(pages, n, capacity):

s = set()

indexes = {}

page\_faults = 0

for i in range(n):

if len(s) < capacity:

if pages[i] not in s:

s.add(pages[i])

page\_faults += 1

indexes[pages[i]] = i

else:

if pages[i] not in s:

lru = float('inf')

for page in s:

if indexes[page] < lru:

lru = indexes[page]

val = page

s.remove(val)

s.add(pages[i])

page\_faults += 1

indexes[pages[i]] = i

print(s)

return page\_faults

pages = [7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2]

n = len(pages)

capacity = 4

print(pageFaults(pages, n, capacity))

**OUTPUT**:

