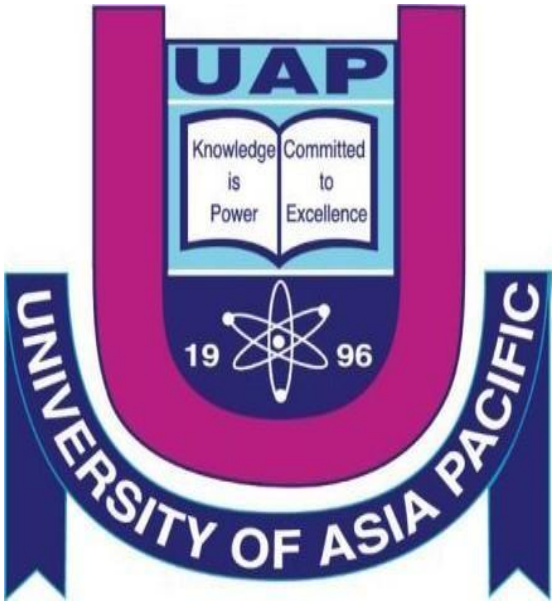

Department of Computer Science & Engineering

Course Title: Operating System Lab

Course Code: CSE 406

Lab Report No: 11

Lab Report On, LRU Page Replacement

Submitted To:
Atia Rahman Orthi
Lecturer,
Department of CSE, UAP

Submitted By:
Name: Amirul Islam Papon
Reg No: 21201076
Sec: B2

Problem Statement

In operating systems, when multiple processes run simultaneously, they require memory to execute. The main memory (RAM) is limited in size, so when it becomes full, some pages need to be swapped out to make space for new ones. The challenge is to decide which page to replace in order to minimize the number of page faults (misses).

The **Least Recently Used (LRU)** page replacement algorithm is a commonly used strategy that aims to reduce page faults by removing the least recently used page from memory.

Algorithm Steps

The LRU algorithm works by keeping track of the order in which pages are accessed. When a page must be replaced, the one that hasn't been used for the longest time is removed.

Steps:

1. Initialize an empty page frame of a given capacity.
2. Iterate over the page reference string.
3. For each page:
    - If the page is already in memory, mark it as recently used.
    - If the page is not in memory:
        - A page fault occurs.
        - If the memory is full, evict the least recently used page.
        - Load the new page into memory.
4. Track and count the number of page faults.

Code

LRU_page_replacement.py > ...

```python
from collections import OrderedDict

def lru_page_replacement(pages, capacity):
    memory = OrderedDict()
    page_faults = 0

    for page in pages:
        if page not in memory:
            page_faults += 1
            if len(memory) == capacity:
                memory.popitem(last=False)
        else:
            memory.move_to_end(page)

        memory[page] = True
        print(f"Page: {page} -> Memory: {list(memory.keys())}")

    print(f"\nTotal Page Faults: {page_faults}")
    return page_faults

pages = [7, 0, 1, 2, 0, 3, 0, 4]
capacity = 3
lru_page_replacement(pages, capacity)
```

Output

```
[Running] python -u "d:\Operating-System\LRU_page_replacement.py"
Page: 7 -> Memory: [7]
Page: 0 -> Memory: [7, 0]
Page: 1 -> Memory: [7, 0, 1]
Page: 2 -> Memory: [0, 1, 2]
Page: 0 -> Memory: [1, 2, 0]
Page: 3 -> Memory: [2, 0, 3]
Page: 0 -> Memory: [2, 3, 0]
Page: 4 -> Memory: [3, 0, 4]

Total Page Faults: 6

[Done] exited with code=0 in 0.058 seconds
```

Discussion

Input Parameters:

- Page reference string: [7, 0, 1, 2, 0, 3, 0, 4]
- Page frame capacity: 3

Execution Trace:

Step	Page	Memory State	Page Fault
1	7	[7]	Yes
2	0	[7, 0]	Yes
3	1	[7, 0, 1]	Yes
4	2	[0, 1, 2]	Yes
5	0	[1, 2, 0]	No
6	3	[2, 0, 3]	Yes
7	0	[2, 3, 0]	No
8	4	[3, 0, 4]	Yes

Total Page Faults: 6

Analysis:

- LRU maintains a history of usage to make intelligent replacement decisions.
- In the above trace, we see that even frequently used pages (like 0) are not removed unless absolutely necessary.
- This makes LRU more efficient than simpler algorithms like FIFO in many cases.

Conclusion

The **Least Recently Used (LRU)** page replacement algorithm is a widely-used strategy due to its logical assumption that recently accessed pages are likely to be used again soon. It typically results in fewer page faults compared to simpler algorithms like FIFO or Random Replacement.

Advantages:

- Provides a good approximation of optimal performance.
- Reduces unnecessary page swaps by prioritizing active pages.

Disadvantages:

- Requires tracking usage history, which can add overhead in terms of memory and time.

Source Code Link: https://github.com/Amirul-Islam-Papon/Operating-System/blob/main/LRU_page_replacement.py