Experiment – 6

Page Replacement Algorithms

The Page replacement algorithm decides which memory page is to be replaced. The process of replacement is sometimes called swap out or write to disk. Page replacement is done when the requested page is not found in the main mem. This is also called Page fault. A page fault happens when a running program accesses a memory page (tries to access) which has been swapped out of memory. Since actual physical memory is much smaller than virtual memory so page fault happens. Hence the operating system has to use an algorithm (Page replacement Algorithm) to decide which pages are to swapped in and out of physical memory to minimize page faults.

1. First in First out (FIFO)

This is the simplest page replacement algorithm. In this algo. the OS keeps a track of oldest page using a queue. When a page needs to be replaced, the oldest page is selected for removal at the front of queue.

2. Optimal Page Replacement

In this algorithm, pages are replaced which would not be used for the longest duration of time in the future. Optimal page replacement is perfect, but not possible as in practise an operating system cannot know future requests. It is a benchmark algorithm used to compare the the performance of other (practical) algorithms.

3. Least Recently Used (LRU)

It is one of the most widely used and practical algorithm (LRU cache). In this algorithm the least recently used page is replaced. It is opposite of optimal page replacement Algorithm, in the sense that here we look for page in the past instead of future.

ALGORITHM

First in First out

1. Maintain a FIFO queue on the order of arrival of frames
2. If page fault occurs, swap out oldest element from the queue
3. Else if space is available then add the frame at the end of the queue

Optimal Page Replacement

1. If page fault occurs and space is available in buffer then add it to buffer
2. If space is not available, look at future pages and swap out the one which would be used last.

Least Recently Used (LRU)

1. Maintain a mapping of least recently pages in memory of the last used time
2. If page fault occurs and space is available then no swaps are required
3. If space is not available then swap out least recently used frame
4. Update the mapping for every new coming frame and hits

Numerical

Q1 Concider the page reference string 7,0,1,2,0,3,0,4,2,3,0 with 4 page frames

Ans First in First out

7	0	1	2	0	3	0	4	2	3	0
			2	2	2	2	2	2	2	2
		1	1	1	1	1	1	1	1	0
	0	0	0	0	0	0	4	4	4	4
7	7	7	7	7	3	3	3	3	3	3
*	*	*	*		*		*			*

Queue: 7, 0, 1, 2, 3

Number of page faults = 7

Optimal Page Replacement

7	0	1	2	0	3	0	4	2	3	0
			2	2	2	2	2	2	2	2
		1	1	1	1	1	4	4	4	4
	0	0	0	0	0	0	0	0	0	0
7	7	7	7	7	3	3	3	3	3	3
*	*	*	*		*		*			

Number of Page Faults = 6

Least Recently Used (LRU)

7	0	1	2	0	3	0	4	2	3	0
			2	2	2	2	2	2	2	2
		1	1	1	1	1	4	4	4	4
	0	0	0	0	0	0	0	0	0	0
7	7	7	7	7	3	3	3	3	3	3
*	*	*	*		*		*			

Number of page faults = 6

Q2. Concider the page reference string with 3 page frames - 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1

Ans First in First out

1	2	3	4	1	2	3	4	1	2	3	4	1
	3	3	3	3	2	2	2	1	1	1	4	4
	2	2	2	1	1	1	4	4	4	3	3	3
1	1	1	4	4	4	3	3	3	2	2	2	1
*	*	*	*	*	*	*	*	*	*	*	*	*

Queue : 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1

Page Faults = 13

Optimal Page Replacement

1	2	3	4	1	2	3	4	1	2	3	4	1
		3	4	4	4	4	4	4	4	4	4	4
	2	2	2	2	2	3	3	3	3	3	3	3
1	1	1	1	1	1	1	1	1	2	2	2	1
*	*	*	*			*			*			*

Page faults = 7

Least recently used (LRU)

1	2	3	4	1	2	3	4	1	2	3	4	1
		3	3	3	2	2	2	1	1	1	4	4
	2	2	2	1	1	1	4	4	4	3	3	3
1	1	1	4	4	4	3	3	3	2	2	2	1
*	*	*	*	*	*	*	*	*	*	*	*	*

Page faults = 13

Q3 Concider the following page reffernce string with 3 Page frames - 1 2 3 4 5 4 3 2 1 2 3 4 5 4

Ans First in First out

1	2	3	4	5	4	3	2	1	2	3	4	5	4
		3	3	3	3	3	2	2	2	2	4	4	4
	2	2	2	5	5	5	5	5	5	3	3	3	3
1	1	1	4	4	4	4	4	1	1	1	1	5	5
*	*	*	*	*			*	*		*	*	*	

Queue : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Page faults = 10

Optimal Page Replacement

1	2	3	4	5	4	3	2	1	2	3	4	5	4
		3	3	3	3	3	3	3	3	3	3	3	3
	2	2	2	5	5	5	2	2	2	2	2	5	5
1	1	1	4	4	4	4	4	1	1	1	4	4	4
*	*	*	*	*			*	*			*	*	

Page Faults = 9

Least Recently Used (LRU)

1	2	3	4	5	4	3	2	1	2	3	4	5	4
		3	3	3	3	3	3	3	3	3	3	3	
	2	2	2	5	5	5	2	2	2	2	2	5	
1	1	1	4	4	4	4	4	1	1	1	4	4	
*	*	*	*	*			*	*			*	*	

Page Faults = 9

Code

```python
# Abstract Classes
from abc import ABCMeta, abstractmethod

# Copy lists
from copy import deepcopy

# Output
from rich.console import Console
from rich.table import Table
from rich.style import Style


def me():
    print("\n")
    console = Console(width=20)
    console.print(
        "Deekshant Wadhwa\nIT-EVE\n[not bold white]01296303118",
        style="white on dark_blue",
        justify="center",
    )


class PageReplacement(metaclass=ABCMeta):
    def __init__(self, size, pages, name):
        self.size = size
        self.pages = pages
        self.name = name

        self.hits = 0
        self.miss = 0

        # records hit or miss
        self.record = []

        # records buffer history
        self.history = []

        # Execute
        self.process()
        self.output()

    # Page Replacement Stratergy
    @abstractmethod
```

```
    def stratergy(self, page):
        ...

    def newPage(self, i, page):
        ...

    def Hit(self):
        self.hits += 1
        self.record.append("Hit")

    def Miss(self):
        self.miss += 1
        self.record.append("Miss")

    def process(self):
        for i, page in enumerate(self.pages):
            # individual processing of all pages
            self.newPage(i, page)

            # first page
            if not self.history:
                self.Miss()
                self.history.append([page])
                continue

            ## further pages –

            # page found in buffer
            if page in self.history[-1]:
                self.Hit()
                self.history.append(deepcopy(self.history[-1]))

            # page not found in buffer
            else:
                self.Miss()

                # space available in buffer
                if len(self.history[-1]) < self.size:
                    newBuffer = deepcopy(self.history[-1])
                    newBuffer.append(page)

                # buffer is full
                else:
                    newBuffer = self.stratergy(page)
```

```
                self.history.append(newBuffer)

    def output(self):
        # Cleaning Data
        for i, h in enumerate(self.history):
            if len(h) < self.size:
                self.history[i] = h + [" "] * (self.size - len(h))
            self.history[i] = list(reversed(self.history[i]))

        # Making Output
        console = Console()
        console.print(f"\n\n {self.name} ", style="black on white")
        table = Table(show_header=True, show_footer=True, box=box.SQUARE)

        # Table Formatting
        miss_footer_style = Style(bold=False, color="red1")
        hit_footer_style = Style(bold=False, color="chartreuse1")
        empty_footer_style = Style(bold=False, color="grey93")
        cell_style = "yellow1"
        header_style = Style(bold=True, color="cyan1")

        # Adding columns
        table.add_column(
            header="",
            footer="Empty",
            justify="center",
            footer_style=empty_footer_style
        )
        for i in range(len(self.history)):
            footer_style = hit_footer_style if self.record[i] == "Hit" else miss_
footer_style
            table.add_column(
                header=str(pages[i]),
                footer=self.record[i],
                justify="center",
                width=5,
                header_style=header_style,
                footer_style=footer_style,
                style=cell_style,
            )

        # Adding rows
        for i in range(len(self.history[0])):
            row = [""]
            for j in range(len(self.history)):
```

```
                row.append(str(self.history[j][i]))
            table.add_row(*row)

        console.print(table)
        console.print(f"[green]Hits: {str(self.hits)}")
        console.print(f"[red]Miss: {str(self.miss)}")


class FIFO(PageReplacement):
    # page not in buffer, and buffer is full
    def stratergy(self, page):
        prev = deepcopy(self.history[-1])
        prev.pop(0)
        prev.append(page)
        return prev

    def __init__(self, size, pages):
        name = "First in First Out (FIFO)"
        super().__init__(size, pages, name)


class LRU(PageReplacement):
    # page not in buffer, and buffer is full
    def stratergy(self, page):
        prev = deepcopy(self.history[-1])

        # page which was accessed earliest
        removePage = min(prev, key=self.accessMapping.get)

        # replacing old page with new pageS
        prev = [page if x == removePage else x for x in prev]
        return prev

    def __init__(self, size, pages):
        # map access times
        self.accessMapping = {}
        name = "Least Recently Used (LRU)"
        super().__init__(size, pages, name)

    def newPage(self, i, page):
        self.accessMapping[page] = i


class Optimal(PageReplacement):
    def stratergy(self, page):
```

```
        prev = deepcopy(self.history[-1])

        # checking future pages
        pages: list = deepcopy(self.history[-1])
        for i in range(self.currentPage, len(self.pages)):
            if len(pages) == 1:
                break
            curr = self.pages[i]
            if curr in pages:
                pages.remove(curr)

        # replace page
        prev = [page if x == pages[0] else x for x in prev]
        return prev

    def __init__(self, size, pages):
        self.currentPage = -1
        name = "Optimal Page replacement"
        super().__init__(size, pages, name)

    def newPage(self, i, page):
        self.currentPage = i

#inputs
size = int(input("Size: "))
pages = list(map(int, input("Frames: ").split()))
me()

# Page replacement algorithms
fifo = FIFO(size, pages)
lru = LRU(size, pages)
opt = Optimal(size, pages)

me()
print("\n")
```

Output

```
PS D:\Drive\Sem 6\OS\lab> python .\pageReplacement.py
Size: 4
Frames: 7 0 1 2 0 3 0 4 2 3 0


  Deekshant Wadhwa
       IT-EVE
    01296303118
```

First In First Out (FIFO)

	7	0	1	2	0	3	0	4	2	3	0
	7	0 7	1 0 7	2 1 0 7	2 1 0 7	3 2 1 0	3 2 1 0	4 3 2 1	4 3 2 1	4 3 2 1	0 4 3 2
Empty	Miss	Miss	Miss	Miss	Hit	Miss	Hit	Miss	Hit	Hit	Miss

```
Hits: 4
Miss: 7
```

Least Recently Used (LRU)

	7	0	1	2	0	3	0	4	2	3	0
	7	0 7	1 0 7	2 1 0 7	2 1 0 7	2 1 0 3	2 1 0 3	2 4 0 3	2 4 0 3	2 4 0 3	2 4 0 3
Empty	Miss	Miss	Miss	Miss	Hit	Miss	Hit	Miss	Hit	Hit	Hit

```
Hits: 5
Miss: 6
```

Optimal Page replacement

	7	0	1	2	0	3	0	4	2	3	0
	7	0 7	1 0 7	2 1 0 7	2 1 0 7	2 1 0 3	2 1 0 3	2 4 0 3	2 4 0 3	2 4 0 3	2 4 0 3
Empty	Miss	Miss	Miss	Miss	Hit	Miss	Hit	Miss	Hit	Hit	Hit

```
Hits: 5
Miss: 6


  Deekshant Wadhwa
       IT-EVE
    01296303118
```

```
PS D:\Drive\Sem 6\OS\lab> python .\pageReplacement.py
Size: 3
Frames: 1 2 3 4 5 4 3 2 1 2

  Deekshant Wadhwa
       IT-EVE
    01296303118

 First In First Out (FIFO)
```

	1	2	3	4	5	4	3	2	1	2
	1	2 1	3 2 1	4 3 2	5 4 3	5 4 3	5 4 3	2 5 4	1 2 5	1 2 5
Empty	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Miss	Hit

```
Hits: 3
Miss: 7

 Least Recently Used (LRU)
```

	1	2	3	4	5	4	3	2	1	2
	1	2 1	3 2 1	3 2 4	3 5 4	3 5 4	3 5 4	3 2 4	3 2 1	3 2 1
Empty	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Miss	Hit

```
Hits: 3
Miss: 7

 Optimal Page replacement
```

	1	2	3	4	5	4	3	2	1	2
	1	2 1	3 2 1	3 2 4	3 5 4	3 5 4	3 5 4	3 5 2	3 1 2	3 1 2
Empty	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Miss	Hit

```
Hits: 3
Miss: 7

  Deekshant Wadhwa
       IT-EVE
    01296303118

PS D:\Drive\Sem 6\OS\lab>
```

```
Size: 5
Frames: 3 5 6 8 3 1 2 4 5 6 4 4 5 4 3 2 1 2

  Deekshant Wadhwa
      IT-EVE
   01296303118
```

First In First Out (FIFO)

	3	5	6	8	3	1	2	4	5	6	4	4	5	4	3	2	1	2
	3	5 3	6 5 3	8 6 5 3	8 6 5 3	1 8 6 5 3	2 1 8 6 5	4 2 1 8 6	5 4 2 1 8	6 5 4 2 1	6 5 4 2 1	6 5 4 2 1	6 5 4 2 1	6 5 4 2 1	3 6 5 4 2	3 6 5 4 2	1 3 6 5 4	2 1 3 6 5
Empty	Miss	Miss	Miss	Miss	Hit	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Hit	Hit	Miss	Hit	Miss	Miss

```
Hits: 6
Miss: 12
```

Least Recently Used (LRU)

	3	5	6	8	3	1	2	4	5	6	4	4	5	4	3	2	1	2
	3	5 3	6 5 3	8 6 5 3	8 6 5 3	1 8 6 5 3	1 8 6 2 3	1 8 4 2 3	1 5 4 2 3	1 5 4 2 6	1 5 4 2 6	1 5 4 2 6	1 5 4 2 6	1 5 4 2 6	3 5 4 2 6	3 5 4 2 6	3 5 4 2 1	3 5 4 2 1
Empty	Miss	Miss	Miss	Miss	Hit	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Hit	Hit	Miss	Hit	Miss	Hit

```
Hits: 7
Miss: 11
```

Optimal Page replacement

	3	5	6	8	3	1	2	4	5	6	4	4	5	4	3	2	1	2
	3	5 3	6 5 3	8 6 5 3	8 6 5 3	1 8 6 5 3	1 2 6 5 3	4 2 6 5 3	4 2 6 5 3	4 2 6 5 3	4 2 6 5 3	4 2 6 5 3	4 2 6 5 3	4 2 6 5 3	4 2 6 5 3	4 2 6 5 3	4 2 6 5 1	4 2 6 5 1
Empty	Miss	Miss	Miss	Miss	Hit	Miss	Miss	Miss	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Miss	Hit

```
Hits: 10
Miss: 8

  Deekshant Wadhwa
      IT-EVE
   01296303118
```