
Project-02 보고서

[Virtual Memory Management 기법 구현]

2023. 05. 28

0. 프로젝트 개요

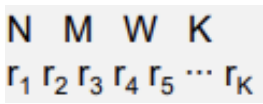
Demand paging system을 위한 page replacement 기법을 구현하고 검증하는 프로젝트이다. 주어진 page reference string을 입력 받아서 MIN replacement, FIFO replacement, LRU replacement, LFU replacement, WS memory management 기법으로 처리했을 경우의 memory residence set의 변화 과정과 page fault 발생 과정을 출력한다.

1. 구현 설계/방법

1) 구현 환경

- python 버전: 3.11
- 사용한 라이브러리: 없음
- 개발 환경: Pycharm Community Edition 2013.1.2

2) 입력 파일 형식

 입력으로 사용할 데이터는 .txt 파일에 저장되어 있으며, 파일은 총 2 줄의 text로 이루어져 있다. 첫 번째 줄의 N, M, W, K는 정수 값으로 입력되어야 하며, blank로 구분한다.

- N: process가 갖는 page의 개수로 100보다 큰 정수가 입력될 경우 ValueError가 발생한다.
- M: 할당 page frame 개수로 20보다 큰 정수가 입력될 경우 ValueError가 발생하며, Fixed Allocation에서만 사용된다.
- W: window size로 100보다 큰 정수가 입력될 경우 ValueError가 발생하며, Working Set 기법에서만 사용된다.
- K: page reference string의 길이로 1,000보다 큰 정수가 입력도리 경우 ValueError가 발생한다.
- $r_1, r_2, r_3, \dots, r_k$: page reference string으로 각 r_i 는 각 page의 번호를 의미하며, blank로 구분한다.

3) page replacement 기법

① MIN replacement

Fixed Allocation 방식에서 사용 가능한 기법으로 page fault를 최소화하는 알고리즘이다. replace 되는 page는 현재 시점에서 가장 오래 실행되지 않을 page이다. 즉, replace 될 page는 forward distance에 의해 결정된다. 이 때, page가 앞으로 reference되지 않을 경우 forward distance 값은 무한 대로 설정되는데, 현재 시점에서 여러 개의 page가 무한 대의

값을 가지게 된다면, tie-breaking rule을 설정하여 replace될 page를 선정한다. 이 프로젝트에서는 간단한 구현을 위해 맨 앞의 page frame에 load된 page가 선정되도록 하였다.

② FIFO replacement

Fixed Allocation 방식에서 사용 가능한 기법으로, memory에 가장 먼저 load된 page를 replace한다. 각 page가 memory에 load된 시간을 기록하는 timestamping이 필요하지만, 이를 대신하여 queue 방식을 사용하기도 한다. 이 프로젝트는 queue 아이디어를 가져와 구현하였다.

③ LRU replacement

Fixed Allocation 방식에서 사용 가능한 기법으로 MIN 기법과는 반대의 방식을 사용하여 replace될 page를 선정한다. 가장 최근에 reference된 시점이 제일 오래된 page를 replace하는 방식으로, backward distance를 계산하여 가장 큰 값을 가지는 page를 선정한다. 이 알고리즘 역시 timestamping을 필요로 하지만, stack을 변형한 LRU stack을 이용하여 구현하기도 한다.

④ LFU replacement

Fixed Allocation 방식에서 사용 가능한 기법으로 page가 reference된 횟수를 세어서 가장 적게 reference된 page가 replace된다. 따라서 각 page에 대한 reference 횟수가 축적되어야 한다. 만약 가장 적게 reference 된 page가 여러 개 존재할 경우 tie-breaking rule으로 LRU 방식을 사용한다.

⑤ Working set memory management

Variable Allocation 방식에서 사용 가능한 기법으로 window size를 사용하여, 특정 시간 간격동안 reference된 page의 집합이 memory에 할당된다. 다른 방식들과 달리, 시간 간격에 따라 page가 할당되므로, page fault와 무관하게 page replacement가 일어난다.

4) page replacement 구현 방법

5개의 page replacement는 page_replacement.py 파일의 class에 있는 method로 구현되어 있다. 각 class의 이름은 MINReplacement, FIFOReplacement, LRUReplacement, LFUReplacement, WSMemoryManagement이며, 각 class의 method들을 이용하여 page replacement 결과를 확인할 수 있다. 이 중 MIN, FIFO, LRU, LFU 방식은 Fixed Allocation 방식을 사용하므로, 이를 고려하여 FixedAllocation 클래스를 상속받도록 구현하였다. 구현 시 초기 할당된 page frame들은 모두 비어 있는 것으로 가정하였다.

① 입력 데이터 불러오기

입력 데이터 텍스트 파일에 있는 텍스트를 한 줄 씩 불러와 변수나 리스트에 저장되며, 이는 각 page replacement method들의 parameter로 사용된다.

첫 번째 줄에 있는 데이터들은 blank를 기준으로 나눠서 각각의 변수에 저장한다. Process가 갖는 page의 개수는 int의 형태로 변수 n에 저장되며, 할당 page frame의 개수는 int의 형태로 변수 m에, window size는 int 형태로 변수 w에 저장되고, page reference string의 길이는 int의 형태로 변수 k에 저장된다.

두 번째 줄에 있는 데이터들 역시 blank를 기준으로 나누어지며, 줄 전체가 data라는 리스트에 저장된다. 이 때, 리스트에 저장되는 page 번호들은 int가 아닌 string의 형태 그대로 저장된다.

② 예외 구현하기

입력 값의 범위와 입력 값에 대한 정보가 불일치할 경우, 예외를 구현하여 주어야 한다. 이 프로젝트에서는 FixedAllocation과 WSMemoryManagement의 생성자 부분에서 각각 예외를 구현하였다.

```
if n > 100 or m > 20 or k > 1000:
    raise ValueError("입력 값의 범위가 올바르지 않습니다.")

elif n < 0 or m < 0 or k < 0:
    raise ValueError("입력 값의 범위가 올바르지 않습니다.")

if len(self.data) != k:
    raise Exception("page reference string의 길이와 주어진 page reference string 입력값이 일치하지 않음")
```

FixedAllocation 클래스에 구현된 예외 처리

```
if n > 100 or w > 100 or k > 1000:
    raise ValueError("입력 값의 범위가 올바르지 않습니다.")

elif n < 0 or w < 0 or k < 0:
    raise ValueError("입력 값의 범위가 올바르지 않습니다.")

if len(self.data) != k:
    raise Exception("page reference string의 길이와 주어진 page reference string 입력값이 일치하지 않음")
```

WSMemoryManagement 클래스에 구현된 예외 처리

③ page fault 확인 method 구현하기

현재 시점에서 page fault가 발생했는 지를 확인하기 위해 check_pf(page) method를 구현하였다. 이는 모든 page replacement가 사용하는 method로, FixedAllocation 클래스와 WSMemoryManagement 클래스에 구현되어 있다. 이 method는 page fault가 발생하면, "F"를, page fault가 발생하지 않으면 " "를 반환한다.

```
def check_pf(self, page):
    if page in self.memory_state: # page fault 발생 하지 않은 경우
        return " "
    else: # page fault 발생한 경우
        return "F"
```

④ MIN replacement method 구현하기

MIN replacement는 page fault가 발생한 경우 발생하며, memory state의 변화 과정과 총 page fault 발생 횟수를 확인하기 위해 min_replacement() method를 구현하였다. 이 함수에서는 pf_cnt 변수에 총 page fault 발생 횟수를 저장하며, mr 변수에 각 시점에서의 메모리 상태를 문자열 형태로 축적하여 저장한다.

MIN replacement는 page fault가 발생한 경우, 현재 메모리에 있는 각 page 별로 현재 시점에서 다음에 reference되기까지 걸리는 시간 즉, forward distance를 계산하여 가장 큰 값을 가진 page를 새로운 page로 replace한다. 가장 큰 값을 가진 page가 여러 개일 경우 가장 작은 page frame number에 저장된 memory가 replace된다.

```
def min_replacement(self):
    pf_cnt = 0
    mr = ""

    for i, page in enumerate(self.data):
        pf = super().check_pf(page) # page fault 발생 했는 지 확인

        if pf == "F": # page fault 발생 하지 않은 경우
            pf_cnt += 1 # page fault 발생 횟수 count

            # memory assign
            if len(self.memory_state) < self.m: # 메모리 할당 가능한 공간 남아 있는 경우
                self.memory_state.append(page) # 메모리 page 할당

            # page replacement
            else: # victim 선정 하는 경우
                # 각 page 별 현재 시점 에서 다음에 reference 되는 시점 까지의 시간 계산
                distance = [] # 각 page 별 거리 저장 하는 리스트
                for temp in self.memory_state:
                    if temp in self.data[i:]:
                        d = min(list(x for x, y in enumerate(self.data[i:]) if y == temp))
                    else:
                        d = 999
                    distance.append(d)

                # tie-breaking rule: smallest page frame number
                self.memory_state[distance.index(max(distance))] = page

        # print(f"t = {i+1}일 때 memory state:", self.memory_state, " / referenced page: ", page)
        mr += "t = " + str(i+1) + "일 때 memory state: " + str(self.memory_state) + " " + pf + "\n"

    print("총 page fault 횟수", pf_cnt)
    print("메모리 상태 변화 과정 (page fault 발생 위치 표시)")
    print(mr)
```

⑤ FIFO replacement method 구현하기

FIFO replacement는 page fault가 발생한 경우 일어나며, memory state의 변화 과정과 총 page fault 발생 횟수를 확인하기 위해 fifo_replacement() method를 구현하였다. 이 함수에서는 queue의 아이디어를 사용하여 가장 먼저 메모리에 할당된 page가 replace 되도록 한다. pf_cnt 변수에는 총 page fault 발생 횟수가 저장되며, mr 변수에는 각 시점에서의 메모리 상태 변화 과정이 축적되어 문자열로 저장되고, queue 리스트에는 page가 할당된 순서가 저장되며, 갱신된다.

```
def fifo_replacement(self):
    pf_cnt = 0
    mr = ""
    queue = []

    for i, page in enumerate(self.data):
        pf = super().check_pf(page)

        if pf == "F": # page fault 발생한 경우
            pf_cnt += 1

            # memory assign
            if len(self.memory_state) < self.m:
                self.memory_state.append(page)
                queue.append(page) # 새롭게 할당된 page enqueue

            # replacement
            else:
                replace_page = queue.pop(0) # 가장 먼저 할당된 page dequeue
                replace_index = self.memory_state.index(replace_page)
                self.memory_state[replace_index] = page # page replacement
                queue.append(page) # 새롭게 할당된 page enqueue

        mr += "t = " + str(i+1) + "일 때 memory state: " + str(self.memory_state) + " " + pf + "\n"

    print("총 page fault 횟수", pf_cnt)
    print("메모리 상태 변화 과정 (page fault 발생 위치 표시)")
    print(mr)
```

⑥ LRU replacement method 구현하기

LRU replacement는 page fault가 발생한 경우 일어나며, memory state의 변화 과정과 총 page fault 발생 횟수를 확인하기 위해 lru_replacement() method를 구현하였다. 이 함수는 pf_cnt 변수에 총 page fault 발생 횟수를 저장하며, mr 변수에는 각 시점에서의 메모리 상태 변화 과정이 축적되어 문자열로 저장된다.

LRU replacement는 MIN replacement 방식과 반대로, distance 리스트를 사용하여 메모리에 할당되어 있는 각 page 별로 가장 최근에 reference된 시점을 계산하여, 가장 오래된 page가 replace된다.

```

def lru_replacement(self):
    pf_cnt = 0
    mr = ""

    for i, page in enumerate(self.data):
        pf = super().check_pf(page)

        if pf == "F": # page fault 발생한 경우
            pf_cnt += 1

            # memory assign
            if len(self.memory_state) < self.m:
                self.memory_state.append(page)

            # page replacement
            else:
                distance = []
                for temp in self.memory_state:
                    # 가장 최근에 reference 되었던 시점 계산
                    d = max(list(x for x, y in enumerate(self.data[:i]) if y == temp))
                    distance.append(d)

                self.memory_state[distance.index(min(distance))] = page

        mr += "t = " + str(i+1) + "일 때 memory state: " + str(self.memory_state) + " " + pf + "\n"

    print("총 page fault 횟수", pf_cnt)
    print("메모리 상태 변화 과정 (page fault 발생 위치 표시)")
    print(mr)

```

⑦ LFU replacement method 구현하기

LFU replacement는 page fault가 발생한 경우 일어나며, memory state의 변화 과정과 총 page fault 발생 횟수를 확인하기 위해 lfu_replacement() method를 구현하였다. 이 함수는 pf_cnt 변수에 총 page fault 발생 횟수를 저장하며, mr 변수에 각 시점에서의 메모리 상태 변화 과정을 축적하여 문자열 형태로 저장한다.

또한 LFU algorithm은 현재 memory에 있는 page들이 여태까지 reference 되었던 횟수를 이용하여 replace를 결정하므로 refer_cnt에 각 page의 reference frequency를 저장하고, tie-breaking을 고려하여 LRU 알고리즘을 위한 가장 최근 reference 되었던 시점 또한 distance 리스트에 저장하여 둔다. 만약, refer_cnt의 최솟값을 갖는 page가 둘 이상이라면, LRU 알고리즘을 사용하여 tie-breaking을 한다.

```

def lfu_replacement(self):
    pf_cnt = 0
    mr = ""

    for i, page in enumerate(self.data):
        pf = super().check_pf(page)

        if pf == "F": # page fault 발생한 경우
            pf_cnt += 1

```

```

# memory assign
if len(self.memory_state) < self.m:
    self.memory_state.append(page)

# page replacement
else:
    refer_cnt = [] # page reference count list
    distance = [] # tie-breaking 위한 LRU 리스트
    for temp in self.memory_state:
        # page reference count 계산
        cnt = self.data[:i].count(temp)
        refer_cnt.append(cnt)
        # tie-breaking 위해 가장 최근에 reference 되었던 시점 계산
        d = max(list(x for x, y in enumerate(self.data[:i]) if y == temp))
        distance.append(d)

    if refer_cnt.count(min(refer_cnt)) > 1: # tie 발생한 경우
        self.memory_state[distance.index(min(distance))] = page # LRU Algorithm
    else: # tie 발생하지 않은 경우
        self.memory_state[refer_cnt.index(min(refer_cnt))] = page # LFU Algorithm

mr += "t = " + str(i + 1) + "일 때 memory state: " + str(self.memory_state) + " " + pf + "\n"

print("총 page fault 횟수", pf_cnt)
print("메모리 상태 변화 과정 (page fault 발생 위치 표시)")
print(mr)

```

⑧ Working set memory management method 구현하기

Working set memory management 방식은 다른 page replacement 방식들과 달리 page fault에 의해 page replacement가 발생하지 않고, window size로 설정된 시간 간격에 의해 memory state가 변화한다. 이 프로젝트에서는 working set memory management 방식의 총 page fault 발생 횟수와 memory state의 변화 과정을 확인하기 위해 ws_memory_management() method를 구현하였다.

```

def ws_memory_management(self):
    pf_cnt = 0
    mr = ""

    for i, page in enumerate(self.data):
        pf = self.check_pf(page)

        # page fault check
        if pf == "F":
            pf_cnt += 1

        # memory assign
        if i < self.w:
            self.memory_state = sorted(list(set(self.data[:i+1])))

        # memory management
        else:
            self.memory_state = sorted(list(set(self.data[i-self.w:i+1])))

    mr += "t = " + str(i + 1) + "일 때 memory state: " + str(self.memory_state) + " " + pf + "\n"

```



```
print("총 page fault 횟수", pf_cnt)
print("메모리 상태 변화 과정 (page fault 발생 위치 표시)")
print(mr)
```

2. 다양한 입력에 대한 실행 결과

1) input1.txt: 기본 예시

[illegible]

2) input2.txt: 교재 기본 예시

≡ input2.txt ×

[illegible]

3) input3.txt: FIFO anomaly 교재 예시

1	5 4 3 12
2	1 2 3 4 1 2 5 1 2 3 4 5

[illegible]

4) input4.txt: k와 page reference string의 길이가 다른 경우

input4.txt ×

1	5	4	3	13								
2	1	2	3	4	1	2	5	1	2	3	4	5

```
Traceback (most recent call last):
  File "C:\Users\yeonl\OneDrive\Desktop\과제\운영체제\project 2\main.py", line 15, in <module>
    min_rp = MINReplacement(n, m, k, data)
  File "C:\Users\yeonl\OneDrive\Desktop\과제\운영체제\project 2\page_replacement.py", line 29, in __init__
    super().__init__(n, m, k, data)
  File "C:\Users\yeonl\OneDrive\Desktop\과제\운영체제\project 2\page_replacement.py", line 16, in __init__
    raise Exception("page reference string의 길이와 주어진 page reference string 입력값이 일치하지 않음")
Exception: page reference string의 길이와 주어진 page reference string 입력값이 일치하지 않음
```

5) input5.txt: 입력 값의 범위가 잘못 설정된 경우

input5.txt ×												
1	-5 4 3 12											
2	1 2 3 4 1 2 5 1 2 3 4 5											

```
Traceback (most recent call last):
  File "C:\Users\yeonl\OneDrive\Desktop\과제\운영체제\project 2\main.py", line 15, in <module>
    min_rp = MINReplacement(n, m, k, data)
  File "C:\Users\yeonl\OneDrive\Desktop\과제\운영체제\project 2\page_replacement.py", line 29, in __init__
    super().__init__(n, m, k, data)
  File "C:\Users\yeonl\OneDrive\Desktop\과제\운영체제\project 2\page_replacement.py", line 13, in __init__
    raise ValueError("입력 값의 범위가 올바르지 않습니다.")
ValueError: 입력 값의 범위가 올바르지 않습니다.
```

6) input6.txt: loop을 도는 데이터가 입력된 경우

input6.txt										
1	4	3	2	10						
2	1	2	3	4	1	2	3	4	1	2

MIN	FIFO	LRU	LFU	WS
총 page fault 횟수 6 메모리 상태 변화 과정 (page fault 발생 위치 표시) t = 1일 때 memory state: ['1'] F t = 2일 때 memory state: ['1', '2'] F t = 3일 때 memory state: ['1', '2', '3'] F t = 4일 때 memory state: ['1', '2', '4'] F t = 5일 때 memory state: ['1', '2', '4'] t = 6일 때 memory state: ['1', '2', '4'] t = 7일 때 memory state: ['1', '3', '4'] t = 8일 때 memory state: ['1', '3', '4'] t = 9일 때 memory state: ['1', '3', '4'] t = 10일 때 memory state: ['2', '3', '4'] F	총 page fault 횟수 10 메모리 상태 변화 과정 (page fault 발생 위치 표시) t = 1일 때 memory state: ['1'] F t = 2일 때 memory state: ['1', '2'] F t = 3일 때 memory state: ['1', '2', '3'] F t = 4일 때 memory state: ['4', '2', '3'] F t = 5일 때 memory state: ['4', '1', '3'] F t = 6일 때 memory state: ['4', '1', '2'] F t = 7일 때 memory state: ['3', '1', '2'] F t = 8일 때 memory state: ['3', '4', '2'] F t = 9일 때 memory state: ['3', '4', '1'] F t = 10일 때 memory state: ['2', '4', '1'] F	총 page fault 횟수 10 메모리 상태 변화 과정 (page fault 발생 위치 표시) t = 1일 때 memory state: ['1'] F t = 2일 때 memory state: ['1', '2'] F t = 3일 때 memory state: ['1', '2', '3'] F t = 4일 때 memory state: ['4', '2', '3'] F t = 5일 때 memory state: ['4', '1', '3'] F t = 6일 때 memory state: ['4', '1', '2'] F t = 7일 때 memory state: ['3', '1', '2'] F t = 8일 때 memory state: ['3', '4', '2'] F t = 9일 때 memory state: ['3', '4', '1'] F t = 10일 때 memory state: ['2', '4', '1'] F	총 page fault 횟수 10 메모리 상태 변화 과정 (page fault 발생 위치 표시) t = 1일 때 memory state: ['1'] F t = 2일 때 memory state: ['1', '2'] F t = 3일 때 memory state: ['1', '2', '3'] F t = 4일 때 memory state: ['4', '2', '3'] F t = 5일 때 memory state: ['4', '1', '3'] F t = 6일 때 memory state: ['4', '1', '2'] F t = 7일 때 memory state: ['3', '1', '2'] F t = 8일 때 memory state: ['3', '4', '2'] F t = 9일 때 memory state: ['3', '4', '1'] F t = 10일 때 memory state: ['2', '4', '1'] F	총 page fault 횟수 10 메모리 상태 변화 과정 (page fault 발생 위치 표시) t = 1일 때 memory state: ['1'] F t = 2일 때 memory state: ['1', '2'] F t = 3일 때 memory state: ['1', '2', '3'] F t = 4일 때 memory state: ['2', '3', '4'] F t = 5일 때 memory state: ['1', '3', '4'] F t = 6일 때 memory state: ['1', '2', '4'] F t = 7일 때 memory state: ['1', '2', '3'] F t = 8일 때 memory state: ['2', '3', '4'] F t = 9일 때 memory state: ['1', '3', '4'] F t = 10일 때 memory state: ['1', '2', '4'] F

7) input7.txt: 하나의 page만 reference되는 경우

1	1	4	2	5
2	1	1	1	1

MIN 총 page fault 횟수 1 메모리 상태 변화 과정 (page fault 발생 위치 표시) t = 1일 때 memory state: ['1'] F t = 2일 때 memory state: ['1'] t = 3일 때 memory state: ['1'] t = 4일 때 memory state: ['1'] t = 5일 때 memory state: ['1']	LRU 총 page fault 횟수 1 메모리 상태 변화 과정 (page fault 발생 위치 표시) t = 1일 때 memory state: ['1'] F t = 2일 때 memory state: ['1'] t = 3일 때 memory state: ['1'] t = 4일 때 memory state: ['1'] t = 5일 때 memory state: ['1']	WS 총 page fault 횟수 1 메모리 상태 변화 과정 (page fault 발생 위치 표시) t = 1일 때 memory state: ['1'] F t = 2일 때 memory state: ['1'] t = 3일 때 memory state: ['1'] t = 4일 때 memory state: ['1'] t = 5일 때 memory state: ['1']
FIFO 총 page fault 횟수 1 메모리 상태 변화 과정 (page fault 발생 위치 표시) t = 1일 때 memory state: ['1'] F t = 2일 때 memory state: ['1'] t = 3일 때 memory state: ['1'] t = 4일 때 memory state: ['1'] t = 5일 때 memory state: ['1']	LFU 총 page fault 횟수 1 메모리 상태 변화 과정 (page fault 발생 위치 표시) t = 1일 때 memory state: ['1'] F t = 2일 때 memory state: ['1'] t = 3일 때 memory state: ['1'] t = 4일 때 memory state: ['1'] t = 5일 때 memory state: ['1']	

8) input8.txt: page number가 숫자가 아닌 경우

input8.txt

1	5	4	3	11							
2	a	c	b	d	c	d	e	a	d	a	b

MIN 총 page fault 횟수 5 메모리 상태 변화 과정 (page fault 발생 위치 표시) t = 1일 때 memory state: ['a'] F t = 2일 때 memory state: ['a', 'c'] F t = 3일 때 memory state: ['a', 'c', 'b'] F t = 4일 때 memory state: ['a', 'c', 'b', 'd'] F t = 5일 때 memory state: ['a', 'c', 'b', 'd'] t = 6일 때 memory state: ['a', 'c', 'b', 'd'] t = 7일 때 memory state: ['a', 'c', 'b', 'd'] F t = 8일 때 memory state: ['a', 'c', 'b', 'd'] t = 9일 때 memory state: ['a', 'c', 'b', 'd'] t = 10일 때 memory state: ['a', 'c', 'b', 'd'] t = 11일 때 memory state: ['a', 'c', 'b', 'd']	FIFO 총 page fault 횟수 5 메모리 상태 변화 과정 (page fault 발생 위치 표시) t = 1일 때 memory state: ['a'] F t = 2일 때 memory state: ['a', 'c'] F t = 3일 때 memory state: ['a', 'c', 'b'] F t = 4일 때 memory state: ['a', 'c', 'b', 'd'] F t = 5일 때 memory state: ['a', 'c', 'b', 'd'] t = 6일 때 memory state: ['a', 'c', 'b', 'd'] t = 7일 때 memory state: ['a', 'c', 'b', 'd'] F t = 8일 때 memory state: ['a', 'c', 'b', 'd'] t = 9일 때 memory state: ['a', 'c', 'b', 'd'] t = 10일 때 memory state: ['a', 'c', 'b', 'd'] t = 11일 때 memory state: ['a', 'c', 'b', 'd']	LRU 총 page fault 횟수 7 메모리 상태 변화 과정 (page fault 발생 위치 표시) t = 1일 때 memory state: ['a'] F t = 2일 때 memory state: ['a', 'c'] F t = 3일 때 memory state: ['a', 'c', 'b'] F t = 4일 때 memory state: ['a', 'c', 'b', 'd'] F t = 5일 때 memory state: ['a', 'c', 'b', 'd'] t = 6일 때 memory state: ['a', 'c', 'b', 'd'] t = 7일 때 memory state: ['a', 'c', 'b', 'd'] F t = 8일 때 memory state: ['a', 'c', 'b', 'd'] F t = 9일 때 memory state: ['a', 'c', 'b', 'd'] t = 10일 때 memory state: ['a', 'c', 'b', 'd'] t = 11일 때 memory state: ['a', 'c', 'b', 'd']	LFU 총 page fault 횟수 7 메모리 상태 변화 과정 (page fault 발생 위치 표시) t = 1일 때 memory state: ['a'] F t = 2일 때 memory state: ['a', 'c'] F t = 3일 때 memory state: ['a', 'c', 'b'] F t = 4일 때 memory state: ['a', 'c', 'b', 'd'] F t = 5일 때 memory state: ['a', 'c', 'b', 'd'] t = 6일 때 memory state: ['a', 'c', 'b', 'd'] t = 7일 때 memory state: ['a', 'c', 'b', 'd'] F t = 8일 때 memory state: ['a', 'c', 'b', 'd'] F t = 9일 때 memory state: ['a', 'c', 'b', 'd'] t = 10일 때 memory state: ['a', 'c', 'b', 'd'] t = 11일 때 memory state: ['a', 'c', 'b', 'd']	WS 총 page fault 횟수 7 메모리 상태 변화 과정 (page fault 발생 위치 표시) t = 1일 때 memory state: ['a'] F t = 2일 때 memory state: ['a', 'c'] F t = 3일 때 memory state: ['a', 'b', 'c'] F t = 4일 때 memory state: ['a', 'b', 'c', 'd'] F t = 5일 때 memory state: ['b', 'c', 'd'] t = 6일 때 memory state: ['b', 'c', 'd'] t = 7일 때 memory state: ['c', 'd', 'e'] F t = 8일 때 memory state: ['a', 'c', 'd', 'e'] F t = 9일 때 memory state: ['a', 'd', 'e'] t = 10일 때 memory state: ['a', 'd', 'e'] t = 11일 때 memory state: ['a', 'b', 'd', 'e'] F
--	---	--	--	--

3. 실행 결과 확인 방법

Page_replacement.py 파일에 있는 클래스를 이용하여 각 page replacement 기법에 대한 객체를 생성하고, method를 사용하여 출력 결과를 확인할 수 있다. 파일의 파일 이름을 입력하면, 각 page replacement 기법 별로 총 page fault가 발생한 횟수와 memory state 변화 과정이 출력된다.

이 프로젝트에서는 예시로 총 8개의 input text file을 사용하였으며, 2023-1-OS01-2021311828-박수연-P02.py 파일을 실행할 때, 파일명을 입력하여 해당 파일 입력 값에 대한 결과를 확인할 수 있다.

다음은 2023-1-OS01-2021311828-박수연-P02.py 파일에서 input1.txt 파일을 입력하였을 때 출력된 결과이다.

파일명(.txt)을 입력하세요: input1.txt

MIN

총 page fault 횟수 7

메모리 상태 변화 과정 (page fault 발생 위치 표시)

```
t = 1초 때 memory state: [0] F
t = 2초 때 memory state: [0, 1] F
t = 3초 때 memory state: [0, 1, 2] F
t = 4초 때 memory state: [0, 1, 2] F
t = 5초 때 memory state: [0, 1, 2] F
t = 6초 때 memory state: [0, 1, 2] F
t = 7초 때 memory state: [0, 1, 4] F
t = 8초 때 memory state: [0, 1, 4] F
t = 9초 때 memory state: [0, 1, 4] F
t = 10초 때 memory state: [0, 1, 4] F
t = 11초 때 memory state: [0, 3, 4] F
t = 12초 때 memory state: [0, 3, 4] F
t = 13초 때 memory state: [0, 3, 4] F
t = 14초 때 memory state: [0, 3, 4] F
t = 15초 때 memory state: [0, 3, 4] F
```

FIFO

총 page fault 횟수 10

메모리 상태 변화 과정 (page fault 발생 위치 표시)

```
t = 1초 때 memory state: [0] F
t = 2초 때 memory state: [0, 1] F
t = 3초 때 memory state: [0, 1, 2] F
t = 4초 때 memory state: [0, 1, 2] F
t = 5초 때 memory state: [0, 1, 2] F
t = 6초 때 memory state: [0, 1, 2] F
t = 7초 때 memory state: [0, 4, 2] F
t = 8초 때 memory state: [0, 4, 5] F
t = 9초 때 memory state: [3, 4, 5] F
t = 10초 때 memory state: [1, 4, 5] F
t = 11초 때 memory state: [1, 3, 5] F
t = 12초 때 memory state: [1, 3, 4] F
t = 13초 때 memory state: [1, 3, 4] F
t = 14초 때 memory state: [1, 3, 4] F
t = 15초 때 memory state: [5, 3, 4] F
```

LRU

총 page fault 횟수 9

메모리 상태 변화 과정 (page fault 발생 위치 표시)

```
t = 1초 때 memory state: [0] F
t = 2초 때 memory state: [0, 1] F
t = 3초 때 memory state: [0, 1, 2] F
t = 4초 때 memory state: [0, 1, 2] F
t = 5초 때 memory state: [0, 1, 2] F
t = 6초 때 memory state: [0, 1, 2] F
t = 7초 때 memory state: [0, 4, 2] F
t = 8초 때 memory state: [0, 4, 5] F
t = 9초 때 memory state: [3, 4, 5] F
t = 10초 때 memory state: [1, 4, 5] F
t = 11초 때 memory state: [1, 4, 3] F
t = 12초 때 memory state: [1, 4, 3] F
t = 13초 때 memory state: [1, 4, 3] F
t = 14초 때 memory state: [1, 4, 3] F
t = 15초 때 memory state: [5, 4, 3] F
```

LFU

총 page fault 횟수 9

메모리 상태 변화 과정 (page fault 발생 위치 표시)

```
t = 1초 때 memory state: [0] F
t = 2초 때 memory state: [0, 1] F
t = 3초 때 memory state: [0, 1, 2] F
t = 4초 때 memory state: [0, 1, 2] F
t = 5초 때 memory state: [0, 1, 2] F
t = 6초 때 memory state: [0, 1, 2] F
t = 7초 때 memory state: [0, 4, 2] F
t = 8초 때 memory state: [0, 5, 2] F
t = 9초 때 memory state: [0, 4, 2] F
t = 10초 때 memory state: [3, 4, 1] F
t = 11초 때 memory state: [3, 4, 1] F
t = 12초 때 memory state: [3, 4, 1] F
t = 13초 때 memory state: [3, 4, 1] F
t = 14초 때 memory state: [3, 4, 1] F
t = 15초 때 memory state: [3, 4, 5] F
```

WS

총 page fault 횟수 9

메모리 상태 변화 과정 (page fault 발생 위치 표시)

```
t = 1초 때 memory state: [0] F
t = 2초 때 memory state: [0, 1] F
t = 3초 때 memory state: [0, 1, 2] F
t = 4초 때 memory state: [0, 1, 2, 3] F
t = 5초 때 memory state: [1, 2, 3] F
t = 6초 때 memory state: [2, 3] F
t = 7초 때 memory state: [2, 3, 4] F
t = 8초 때 memory state: [2, 3, 4, 5] F
t = 9초 때 memory state: [3, 4, 5] F
t = 10초 때 memory state: [1, 4, 5] F
t = 11초 때 memory state: [1, 3, 4, 5] F
t = 12초 때 memory state: [1, 3, 4] F
t = 13초 때 memory state: [1, 3, 4] F
t = 14초 때 memory state: [3, 4] F
t = 15초 때 memory state: [3, 4, 5] F
```

1) 총 page fault 횟수

각 page replacement 기법의 이름의 바로 아래 줄에서 총 page fault가 발생한 횟수를 확인할 수 있다.

2) memory state 변화 과정

시점 t 에서의 memory state를 리스트 형태로 확인할 수 있으며, 리스트 안에 있는 page number가 시점 t 에 할당되어 있는 page이다. 또한, 각 시점에서 page fault가 발생했는 지의 여부는 memory state 리스트의 오른쪽의 "F" 표시를 통해 확인할 수 있다.

Memory state 리스트의 오른쪽에 아무 표시가 없으면, 시점 t 에서 page fault가 발생하지 않았음을 나타내고, "F" 표시가 있으면, page fault가 발생하였음을 의미한다.