Page replacement Algorithms

이름 : 권영준

소속 명세 : Student, Dept. of Computer Engineering, Yeungnam University, Gyeongsan, Korea

[요 약]

본 프로젝트에서는 Page Replacement 알고리즘이 필요한 이유와 그 종류를 설명하고 각각의 알고리즘의 특징들을 시뮬레이터를 통해 만들어진 자료를 통해 분석하여 알 수 있도록 한다.

▸ **주제어** : Page Replacement 알고리즘, FIFO, Optimal, LRU, Second-Chance, LFU, SCLRU

I. 서론

본 프로젝트의 개요 및 의의

본 프로젝트의 주요 목표

본 프로젝트의 핵심 제공 기능

II. 배경 지식 및 관련 기술

1. 배경 지식

본 프로젝트를 진행하기 위해 필요한 배경 지식

본 프로젝트 문서를 참고하는 사람이
기본적으로 알고 있어야 하는 배경 지식 (소양)

2. 관련 기술

본 프로젝트에서 분석하는 기술에 대한 현 상황

유사 기술 혹은 이전 기술에 대한 소개

III. (본론) 주요 주제명

1. 주요 주제의 개요

본 프로젝트에서 수행하고자 하는 핵심 주제의 개요

주제 설명을 위한 구조도 제시

2. 주요 주제의 핵심 알고리즘 및 기능

주요 주제의 기능 및 동작 명세

Pseudo 코드(혹 순서도)를 활용한 알고리즘 구체화

알고리즘 이해를 위한 수식 제공 (필요시)

3. 주요 주제의 알고리즘 동작 사례

제공한 알고리즘의 구체적이고 다양한 동작 사례 제시

다이어그램등을 활용한 동작 절차 소개

알고리즘 기반 성능 예측 및 성능 추이에 대한
정량적 분석 (수식 분석을 통한 성능 예측)

IV. 성능 평가

1. 실험 환경

본 프로젝트의 실험 환경 명세
실험 도구, 구현 환경, 구현 언어 등

실험 제약 사항 명세
본 실험만의 특징적 고려 사항 혹은 제약 사항 명세

성능 평가를 위한 입력 요소
해당 입력 요소 선정의 적합성 및 합리성 서술

성능 평가를 위한 접근 방법, 즉 출력 요소
성능 평가 기준 제시
해당 기준 선정의 적합성 및 합리성 서술

2. 실험 결과 및 분석

제공된 실험 환경을 통해 도출된 구체적인 실험 결과
엑셀 차트 혹은 표를 활용한 정량적 결과 제시

그래프나 표로 제공된 실험 결과에 대한 분석 서술
결과의 변화 추이에 대한 원인 분석
가변 인자 변경에 따른 결과 변화도 상관 분석
정성적 결과와 정량적 결과의 차이 및 상관도 분석

V. 결론

본 프로젝트의 목표, 기능, 알고리즘, 실험 결과에 대한 간략한 재명세

본 프로젝트에서 도출된 주요 핵심 결론 제시

향후 알고리즘 개선 방안(가능성) 및 추가 실험 요소(필요시) 제시

참고 자료

[1] 본 문서의 주요 참고 자료 제공
[2] 도서, 웹 싸이트, 기술 문서등
문서 내부에 index를 붙여 직접 참조가 가능하도록 편집

I. 서론

운영체제는 메모리를 관리한다. 메모리 공간은 한정적이기 때문에 최대한 적게 사용하는 것이 좋다. 그래서 가상 메모리라는 기법이 사용되었다. 보조 기억장치의 일부를 주기억장치인 것처럼 사용하는 방법으로 용량이 작은 주기억장치를 마치 더 큰 용량을 갖은 것처럼하여 주기억장치보다 더 큰 용량의 프로그램을 실행하기 위하여 주로 사용하는 방법이다. 이를 위해서 가상기억장치의 주소를 주기억장치의 주소로 변환하는 주소 사상화(Mapping) 과정이 필요하다. 이를 구현하기 위해 페이징(Paging) 기법과, 세그멘테이션(Segmentation) 기법을 사용하게 된다. 본 프로젝트는 페이징 기법을 사용하였을 때 page fault가 발생하였을 때 가상기억장치의 필요한 페이지를 주기억장치의 어떤 프레임을 선택하고 교체해야하는 가를 결정하는 Page replacement Algorithms을 설명한다. 알고리즘의 종류로는 FIFO, Optimal, LRU, Second-Chance 등 있으며 직접 구현하여 알고리즘을 이해하고 또 새로운 알고리즘을 고안해내는 것을 목표로 한다.

II. 배경 지식 및 관련 기술

1. 배경 지식

본 프로젝트의 필요한 배경 지식으로는 가상 메모리와 Demand Paging, Page Fault, Page replacement 알고리즘의 개념이다.

가상 메모리는 주기억 장치의 용량의 제약을 극복하기 위해 하드디스크나 SSD같와 같은 보조기억장치의 용량을 사용하여 사용자가 커다란 기억장소를 갖고 있는 것처럼 가상의 메모리 영역을 생성하는 것이다. 이를 통하여 얻는 장점으로는 프로그램이 기억 공간에 제약을 받지 않고 가상 주소공간을 활용할 수 있고 각 사용자들의 입출력이 적어지므로 속도가 향상된다는 것이다. 하지만 단점 또한 존재하는데 지나친 자료의 이동을 하게 되면 컴퓨터 속도가 느려질 수 있다는 것이다.

Demand Paging 시스템은 프로세스가 특정 페이지를 요구할 때 해당 페이지를 물리 메모리에 로딩한다. 메모리에 필요한 페이지가 있을 때는 hit가 발생하여 잘 진행되지만, 없을 경우에는 Page Fault라는 문제가 생긴다. 프로세스가 필요로 하는 페이지가 없는 경우 보조기억장치에서 페이지를 찾아 빈 프레임에 로딩한다. 여기서 또다시 페이지를 올릴 빈 프레임이 없을 경우 또다른 문제에 직면할 수 있다. 이때 사용하는 것이 새로 올릴 페이지와 교체할 희생 프레임을 찾는 알고리즘, 페이지 교체 알고리즘이다

2. 관련 기술

Operating System Concepts에서 설명하는 Page replacement 알고리즘의 종류로는 FIFO, Optimal Algorithm, Least Recently Used Algorithm, Additional Reference Bits Algorithm, Second Chance Algorithm, Enhanced Second-Chance Algorithm, Least Frequently Used Algorithm, most frequently used Algorithm이 있다. 대표적인 알고리즘의 관련 설명은 아래 표와 같다.

	동작 방식	특징
FIFO	가장 오래된 페이지를 제거	구현하기 쉬움 Belady's anomaly 현상 발생
Optimal	앞으로 가장 오랫동안 사용되지 않을 페이지를 제거	가장 낮은 Page Fault 앞으로 들어올 페이지를 알 수 없음 다른 알고리즘의 성능 평가하는 데에 쓰임
LRU	가장 오랫동안 사용되지 않은 페이지를 제거	구현이 쉽지 않음 근사화를 통해 구현
Second-Chance	Reference bit가 0인 것을 제거	Reference bit가 필요함 LRU 근사화 알고리즘 중 제일 유명
LFU	가장 사용 빈도가 적은 페이지 제거	참조 횟수를 기준으로 함
MFU	가장 사용 빈도가 많은 페이지 제거	참조 횟수를 기준으로 함

표 1 Page Replacement 알고지름의 종류

Page Replacement는 계속 연구되고 있는 분야로 이외에도 수많은 정책들이 있다.

III. Page Replacement Algorithms

1. Page Replacement Algorithms의 개요

Page Replacement Algorithms은 페이지를 효율적으로 교체하기 위해 필요하다. Page Fault가 한번이라도 나는 경우 성능이 현저하게 낮아진다. 무슨 페이지를 교체할지 기준 없이 막무가내로 교체해버리면 Page Fault의 발생횟수가 늘어나게 된다. 그러므로 앞서 기술한 FIFO, Optimal, LRU, LFU 등 수많은 알고리즘이 개발되었다,

본 프로젝트의 알고리즘의 성능 평가는 hit 횟수, Page Fault 횟수를 기준으로 평가한다. 시뮬레이터의 순서도는 다음 그림과 같다.

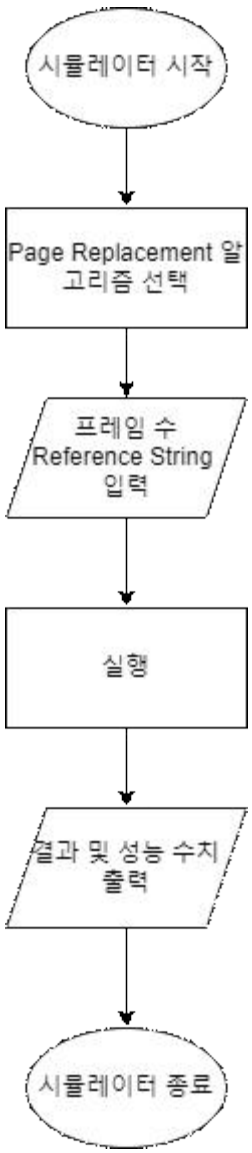


그림 1 시뮬레이터의 전체 순서도

2. Page Replacement Algorithms 핵심 알고리즘 및 기능

Page Replacement 알고리즘의 종류로는 FIFO, Optimal Algorithm, Least Recently Used Algorithm, Additional Reference Bits Algorithm, Second Chance Algorithm, Least Frequently Used Algorithm, most frequently used Algorithm 등이 있다. FIFO는 가장 오래된 페이지를 교체한다. Optimal 알고리즘은 앞으로 가장 오랫동안 사용되지 않을 페이지를 교체한다. 앞으로 들어올 페이지가 무엇인지 알 수 없으므로 다른 알고리즘의 성능을 평가하는데에 쓰인다. LRU 알고리즘은 각 페이지

마다 마지막 사용 시간을 유지하고 교체 시 가장 오랫동안 사용되지 않은 페이지를 교체한다. Optimal 알고리즘과 달리 과거 시간에 대해 적용한 최적 교체 정책이다. Additional Reference Bits 알고리즘은 LRU의 구현이 어려워 근사화를 하여 나온 알고리즘으로 Reference Bit가 0인 페이지 중에서 교체한다. 참조되는 경우 1로 바뀌게 된다. Second Chance 알고리즘은 페이지가 참조되면 Reference bit를 조사하고 0이면 교체하고 1이면 0으로 바꾼 후 다음 페이지를 참조한다. LFU 알고리즘은 시간을 기준으로 한 이전 알고리즘과 달리 참조 횟수를 기준으로 한다. 이전에 사용 빈도가 가장 적은 페이지를 교체한다. 이전에도 가장 적게 쓰였으니 앞으로도 안 쓰이겠다는 논리로 만들어진 알고리즘이다. 반대로 MFU 알고리즘은 이전에 사용 빈도가 가장 많은 페이지를 교체한다. 이전에 가장 많이 썼으니 앞으로는 쓰일 일이 없겠다는 논리로 만들어진 알고리즘이다.

각 알고리즘의 Pseudo 코드와 순서도는 다음과 같다.

//FIFO의 Pseudo 코드
1. 새로운 페이지 도착
2. 메인 메모리에 페이지가 존재하는지 찾는다.
3. 있으면 Page Hit
4. 없으면 Page Fault
5. Page Fault의 경우 들어갈 공간이 있는지 찾는다.
6. 있으면 남은 공간에 들어간다.
7. 없으면 가장 오래된 페이지와 교체한다.

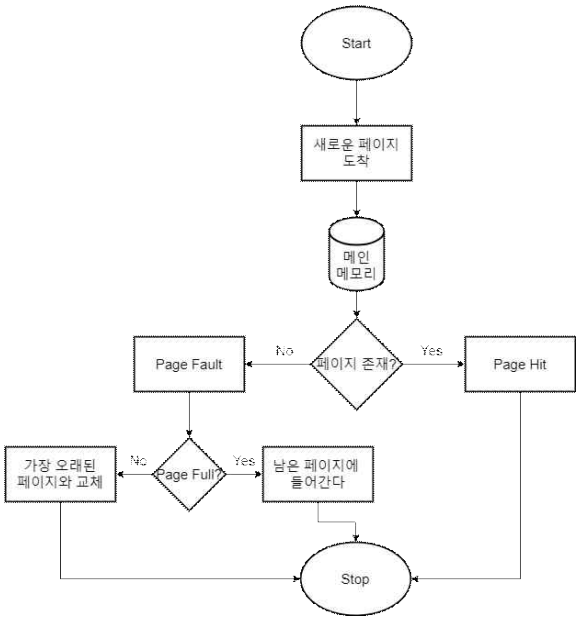


그림 2 FIFO의 순서도

//Optimal 알고리즘의 Pseudo 코드
1. 새로운 페이지 도착
2. 메인 메모리에 페이지가 존재하는지 찾는다.
3. 있으면 Page Hit
4. 없으면 Page Fault
5. Page Fault의 경우 들어갈 공간이 있는지 찾는다.
6. 있으면 남은 공간에 들어간다.
7. 없으면 앞으로 가장 오랫동안 사용되지 않을 페이지와 교체한다.

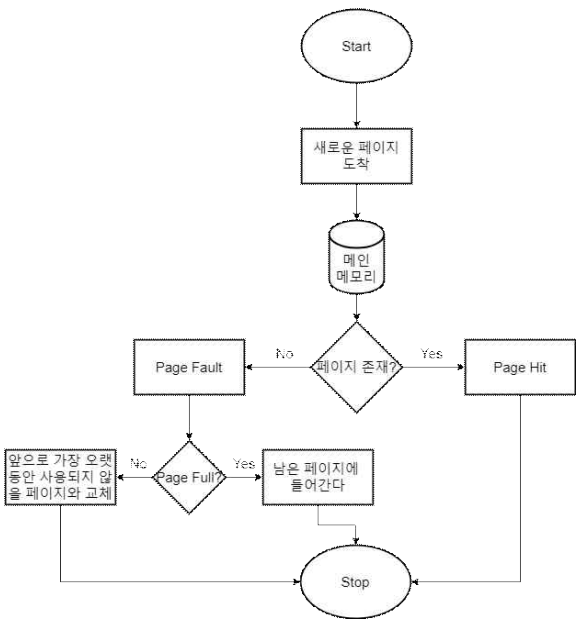


그림 3 Optimal 알고리즘의 순서도

//LRU 알고리즘의 Pseudo 코드
1. 새로운 페이지 도착
2. 메인 메모리에 페이지가 존재하는지 찾는다.
3. 있으면 Page Hit
4. 없으면 Page Fault
5. Page Fault의 경우 들어갈 공간이 있는지 찾는다.
6. 있으면 남은 공간에 들어간다.
7. 없으면 앞으로 가장 오랫동안 사용되지 않을 페이지와 교체한다.

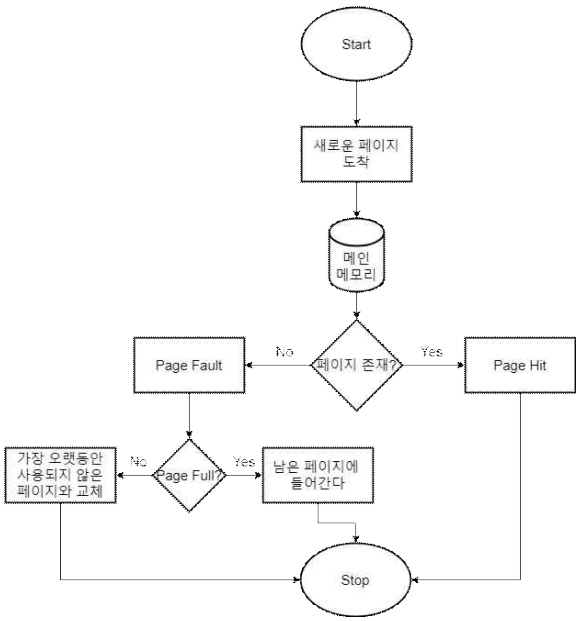


그림 4 LRU 알고리즘의 순서도

//Second Chance 알고리즘의 Pseudo 코드
1. 새로운 페이지 도착
2. 메인 메모리에 페이지가 존재하는지 찾는다.
3. 있으면 Page Hit
4. 없으면 Page Fault
5. Page Fault의 경우 들어갈 공간이 있는지 찾는다.
6. 있으면 남은 공간에 들어간다.
7. 없으면 Reference bit가 0인 페이지와 교체한다.

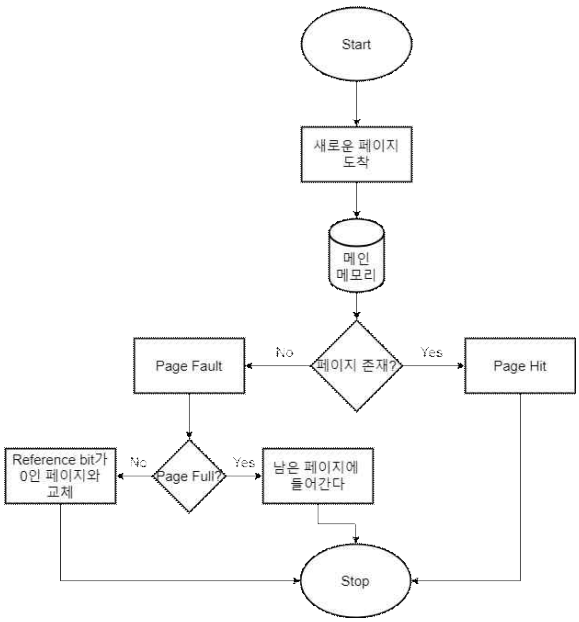


그림 5 Second Chance 알고리즘의 순서도

//LFU 알고리즘의 Pseudo 코드
1. 새로운 페이지 도착
2. 메인 메모리에 페이지가 존재하는지 찾는다.
3. 있으면 Page Hit
4. 없으면 Page Fault
5. Page Fault의 경우 들어갈 공간이 있는지 찾는다.
6. 있으면 남은 공간에 들어간다.
7. 없으면 사용 빈도가 가장 적은 페이지와 교체한다.

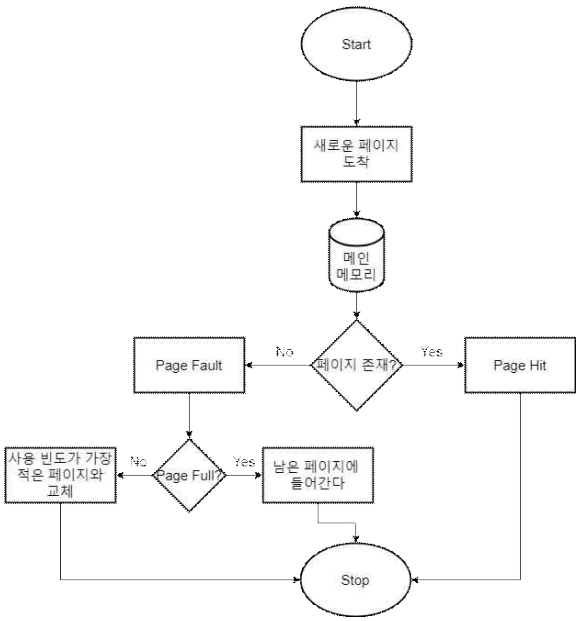


그림 6 LFU 알고리즘의 순서도

//MFU 알고리즘의 Pseudo 코드
1. 새로운 페이지 도착
2. 메인 메모리에 페이지가 존재하는지 찾는다.
3. 있으면 Page Hit
4. 없으면 Page Fault
5. Page Fault의 경우 들어갈 공간이 있는지 찾는다.
6. 있으면 남은 공간에 들어간다.
7. 없으면 사용 빈도가 가장 많은 페이지와 교체한다.

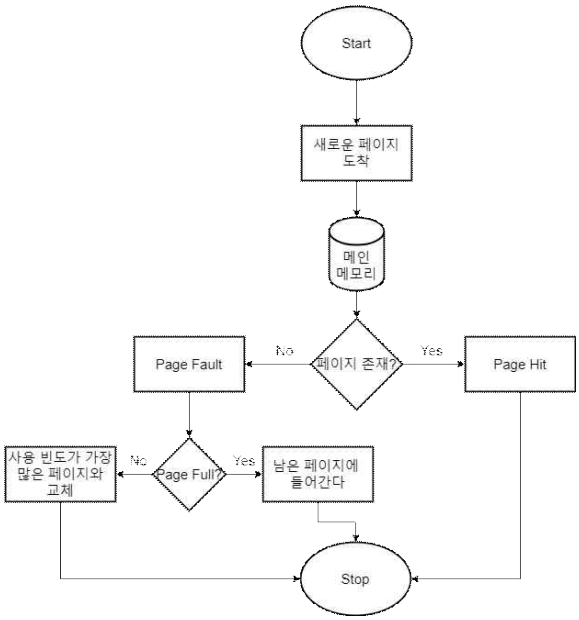


그림 7 MFU 알고리즘의 순서도

본 프로젝트의 목적 중 새로운 알고리즘을 구현하는 것이 있다. 다음은 새로운 알고리즘의 Pseudo 코드이다. 이름은 Second Chance Least Frequently Used로 SCLFU으로 칭한다.

//SCLRU 알고리즘의 Pseudo 코드
1. 새로운 페이지 도착
2. 메인 메모리에 페이지가 존재하는지 찾는다.
3. 있으면 Page Hit
4. 없으면 Page Fault
5. Page Fault의 경우 들어갈 공간이 있는지 찾는다.
6. 있으면 남은 공간에 들어간다.
7. 없으면 Reference bit가 0인 페이지들 중
   가장 오랫동안 사용되지 않은 페이지와 교체한다.

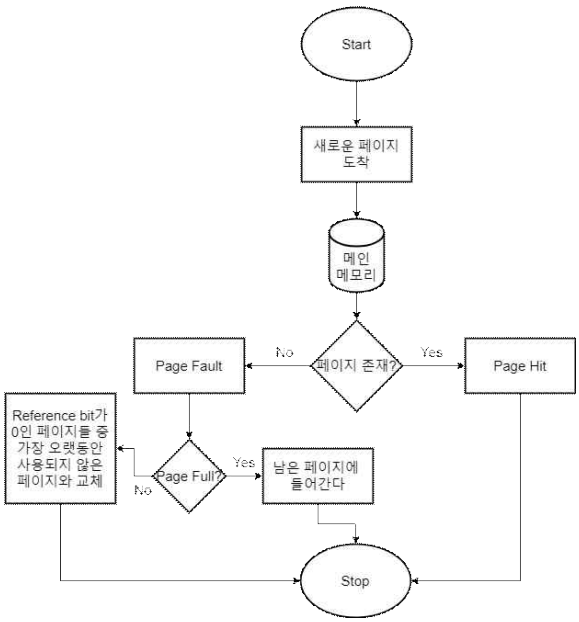


그림 8 SCLRU 알고리즘의 순서도

3. Page Replacement 알고리즘 동작 사례

Reference String가 “123412512345”이고 프레임의 크기가 4인 경우 알고리즘 별 동작 결과는 다음과 같다.

	1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1	1	1	5	5	5	5	4	4
2		2	2	2	2	2	2	1	1	1	1	5
3			3	3	3	3	3	3	2	2	2	2
4				4	4	4	4	4	4	3	3	3
Fault	↑	↑	↑	↑			↑	↑	↑	↑	↑	↑
Hit					↑	↑						

표 2 FIFO의 동작 결과

	1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1	1	1	1	1	1	1	4	4
2		2	2	2	2	2	2	2	2	2	2	2
3			3	3	3	3	3	3	3	3	3	3
4				4	4	4	5	5	5	5	5	5
Fault	↑	↑	↑	↑			↑				↑	
Hit					↑	↑		↑	↑	↑		↑

표 3 Optimal 알고리즘의 동작 결과

	1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1	1	1	1	1	1	1	1	5
2		2	2	2	2	2	2	2	2	2	2	2
3			3	3	3	3	5	5	5	5	4	4
4				4	4	4	4	4	4	3	3	3
Fault	↑	↑	↑	↑			↑			↑	↑	↑
Hit					↑	↑		↑	↑			

표 4 LRU 알고리즘의 동작 결과

	1		2		3		4		1		2		5		1		2		3		4		5	
1	1	1	1	1	1	1	1	1	1	1	1	1	5	1	5	1	5	1	5	1	4	1	4	1
2			2	1	2	1	2	1	2	1	2	1	2	0	1	1	1	1	1	1	1	0	5	1
3					3	1	3	1	3	1	3	1	5	0	3	0	2	1	2	1	2	0	2	0
4							4	1	4	1	4	1	4	0	4	0	4	0	3	1	3	0	3	0
Fault	↑		↑		↑		↑						↑		↑		↑		↑		↑		↑	
Hit									↑		↑													

표 5 Second Chance 알고리즘의 동작 결과

	1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1	1	1	1	1	1	1	1	1
2		2	2	2	2	2	2	2	2	2	2	2
3			3	3	3	3	5	5	5	5	4	4
4				4	4	4	4	4	4	3	3	5
Fault	↑	↑	↑	↑			↑			↑	↑	↑
Hit					↑	↑		↑	↑			

표 6 LFU 알고리즘의 동작 결과

	1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1	1	1	5	5	5	5	5	5
2		2	2	2	2	2	2	1	2	2	2	2
3			3	3	3	3	3	3	3	3	3	3
4				4	4	4	4	4	4	4	4	4
Fault	↑	↑	↑	↑			↑	↑	↑			
Hit					↑	↑				↑	↑	↑

표 7 MFU 알고리즘의 동작 결과

	1		2		3		4		1		2		5		1		2		3		4		5	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	5	1
2			2	1	2	1	2	1	2	1	2	1	2	0	2	0	2	1	2	1	2	0	2	0
3					3	1	3	1	3	1	3	1	5	1	5	1	5	1	5	1	4	1	4	1
4							4	1	4	1	4	1	4	0	4	0	4	0	3	1	3	0	3	0
Fault	↑		↑		↑		↑						↑						↑		↑		↑	
Hit									↑		↑				↑		↑							

표 8 SCLRU 알고리즘의 동작 결과

IV. 성능 평가

1. 실험 환경

이 절에서는 시뮬레이터의 구현환경을 설명한다. 개발 언어는 JAVA를 선정하였고 개발 환경은 IntelliJ IDEA와 JDK 16에서 개발하였다. Reference String의 양식은 실행 시 문자열을 입력하면 된다.

실험용 데이터는 https://www.random.org/strings 에서 뽑은 20자리의 문자열 3개와 임의로 입력한 50자리의 숫자 3개이다. 문자열은 대문자만 체크하고 중복 허용하여 뽑았다. 자세한 내용은 아래 표와 같다. 프레임의 수는 문자열은 3, 4, 5, 10을 진행하였다. 숫자는 프레임의 수를 3, 4, 5로 진행하였고 10을 넣지 않은 이유는 숫자는 서로 다른 개수가 10개이므로 프레임이 10이면 최초에 한번만 page fault가 발생하고 그 뒤로는 발생하지 않기 때문이다.

문자열	data1 ZRJIGRCIJZVCQFWKHSPD	data2 OOOPDIFIUSLVVDDOUXEN	data3 IUVICMQOLXSLUFDRYHUU
숫자	data4 75681087611250086725636482480968789786096092905183	data5 16258260415683758157734142782050783709818366305187	data6 29691739408158348343509269318351486282465744328339

표 9 실험용 데이터

성능 평가는 page fault의 수만 고려할 것이다. 그 이유는 Page replacement 알고리즘의 목표는 page fault의 수를 최소화시키는 것이기 때문이다.

2. 실험 결과 및 분석

위의 실험용 데이터를 기반으로 FIFO, Optimal 알고리즘, LRU 알고리즘, Second Chance 알고리즘, LFU, SCLRU 알고리즘을 시뮬레이터를 통해 실행하였다.

실험용 데이터의 페이지 테이블의 결과는 다음 그림과 같다. 6개의 알고리즘과 4개의 프레임 크기, 또 6개의 데이터로 나타날 수 있는 페이지 테이블의 결과는 144개이므로 페이지 테이블은 data1의 프레임 크기가 3인 경우의 알고리즘별로 출력하겠다.

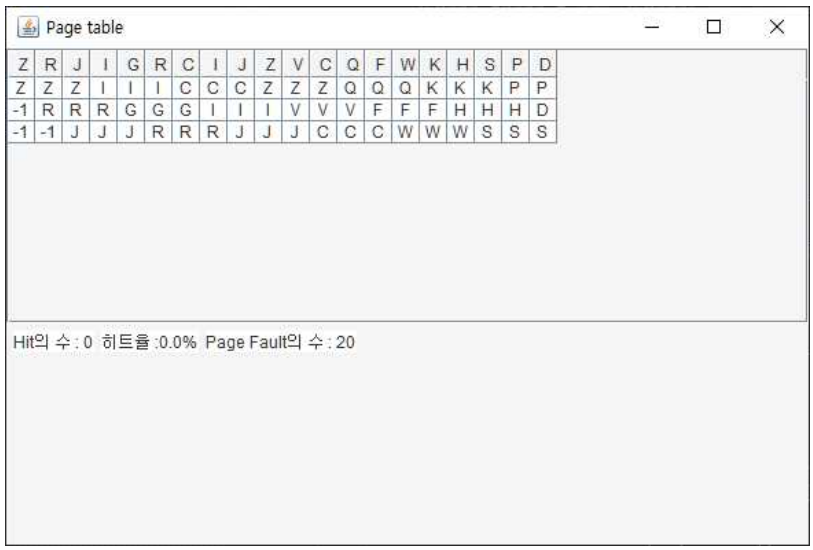


그림 9 data1의 프레임 크기가 3인 경우의 FIFO

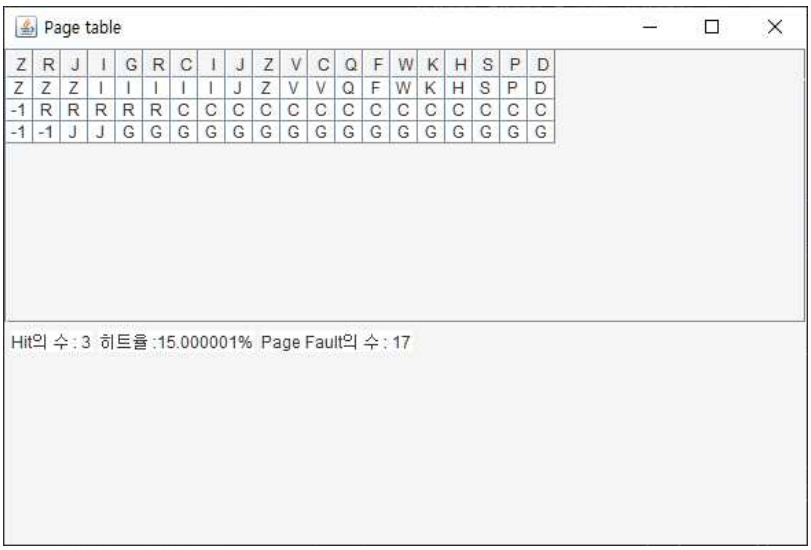


그림 10 data1의 프레임 크기가 3인 경우의 Optimal

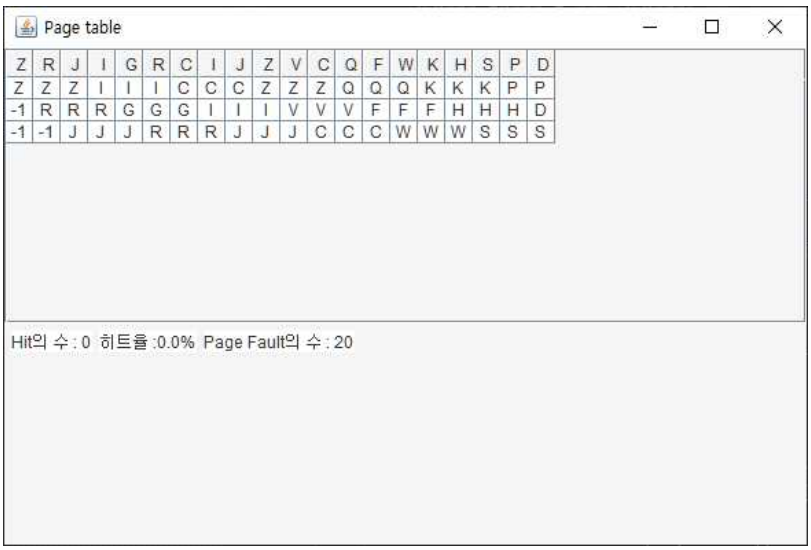


그림 11 data1의 프레임 크기가 3인 경우의 LRU

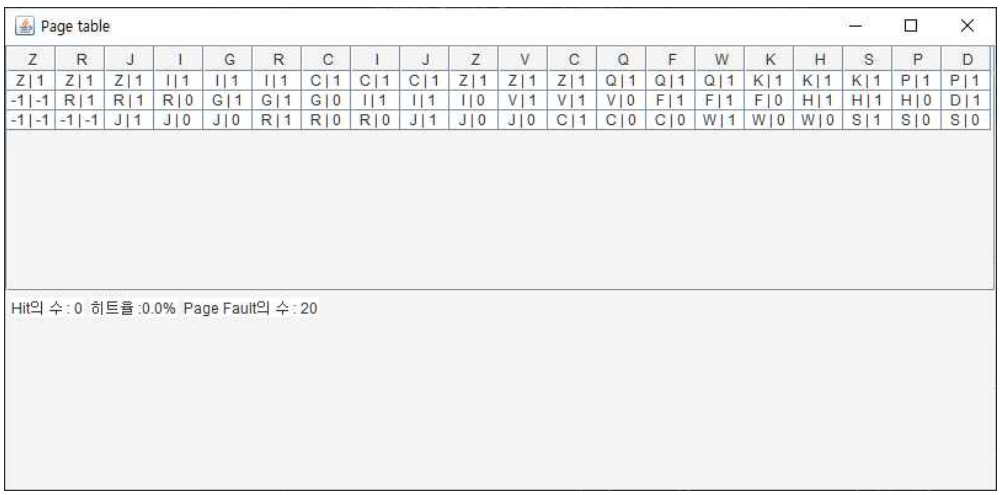


그림 12 data1의 프레임 크기가 3인 경우의 Second-chance

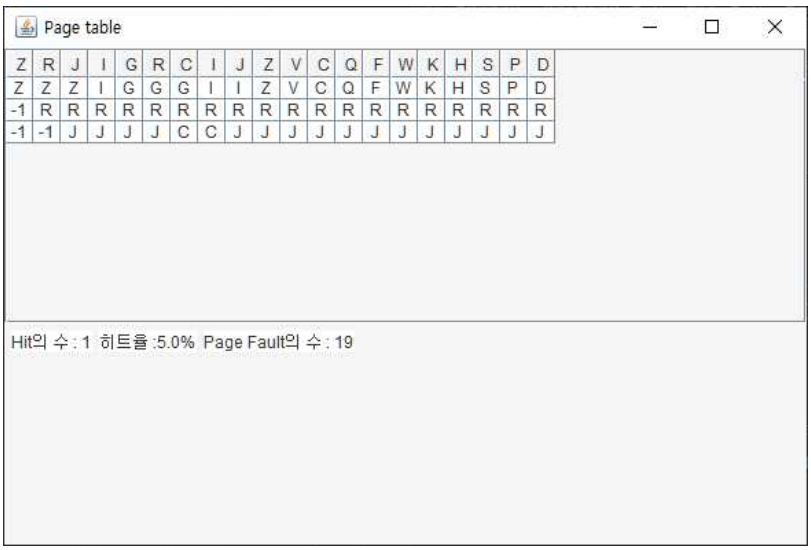


그림 13 data1의 프레임 크기가 3인 경우의 LFU

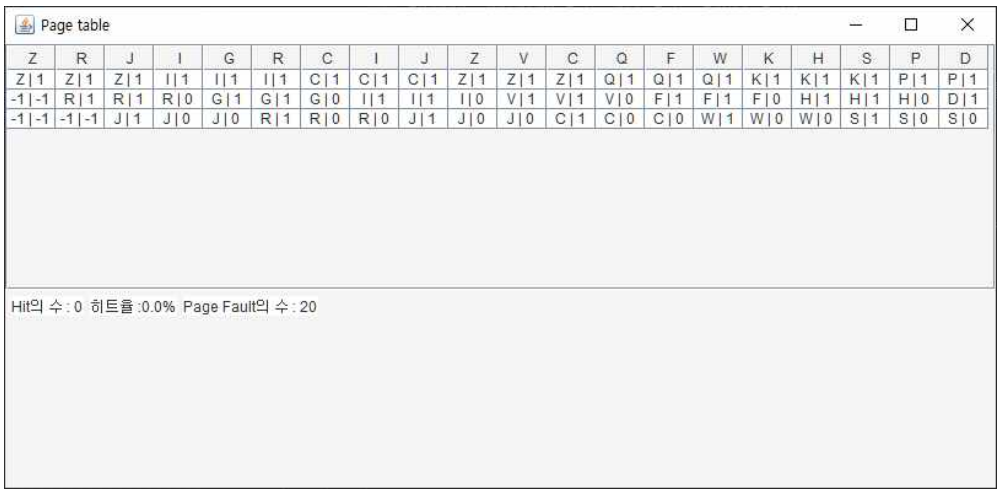
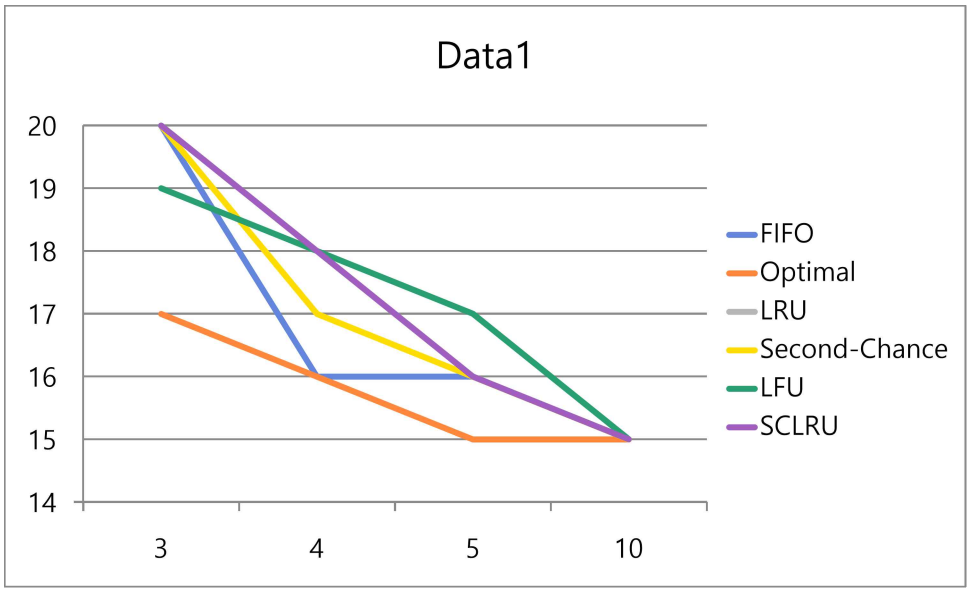
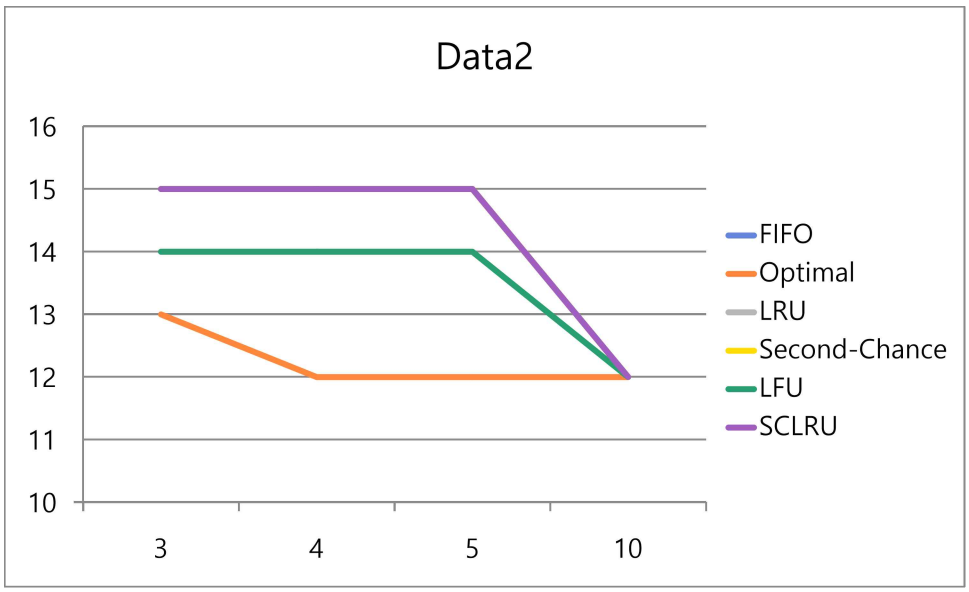


그림 14 data1의 프레임 크기가 3인 경우의 SCLRU

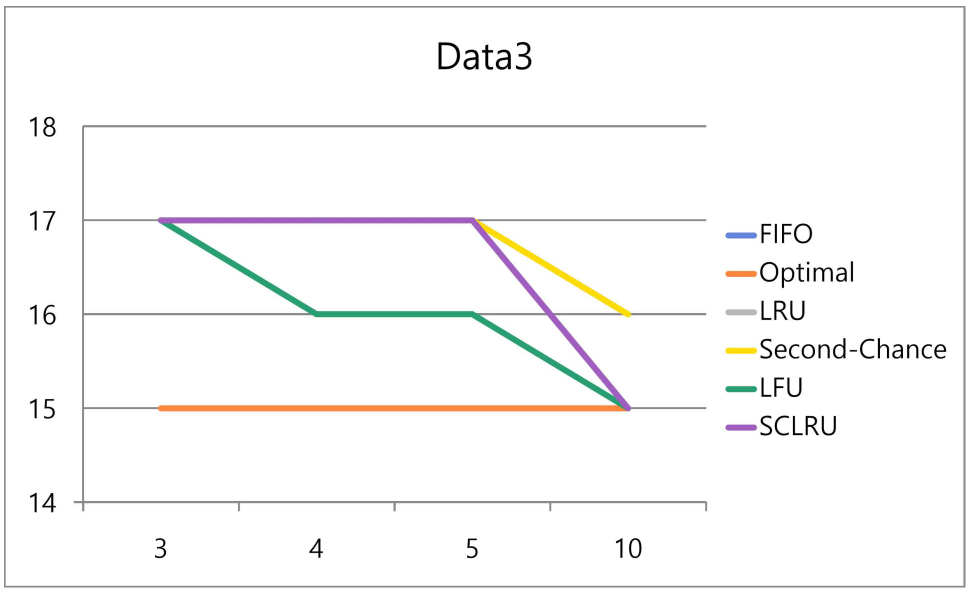
실험용 데이터의 결과를 평가 지표를 기준으로 그린 그래프는 다음과 같다. 가로축은 프레임의 크기이며 세로축은 page fault의 횟수이다.



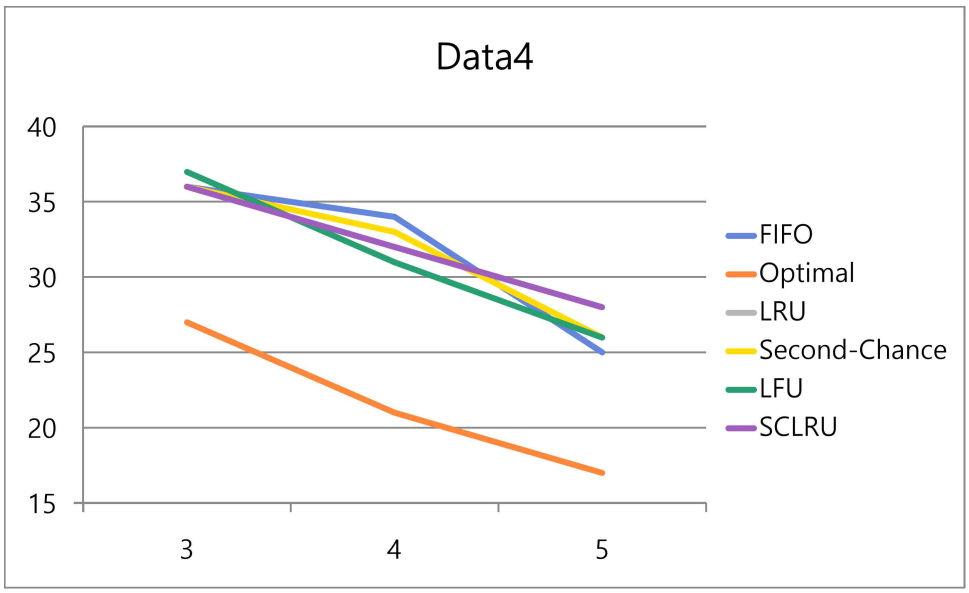
그래프 1 data1의 결과를 표현한 그래프



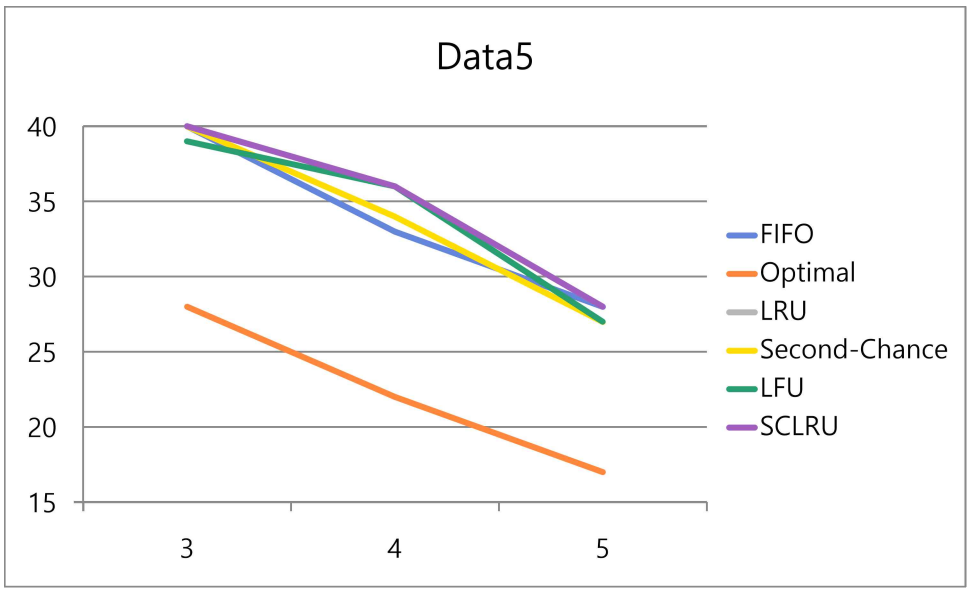
그래프 2 data2의 결과를 표현한 그래프



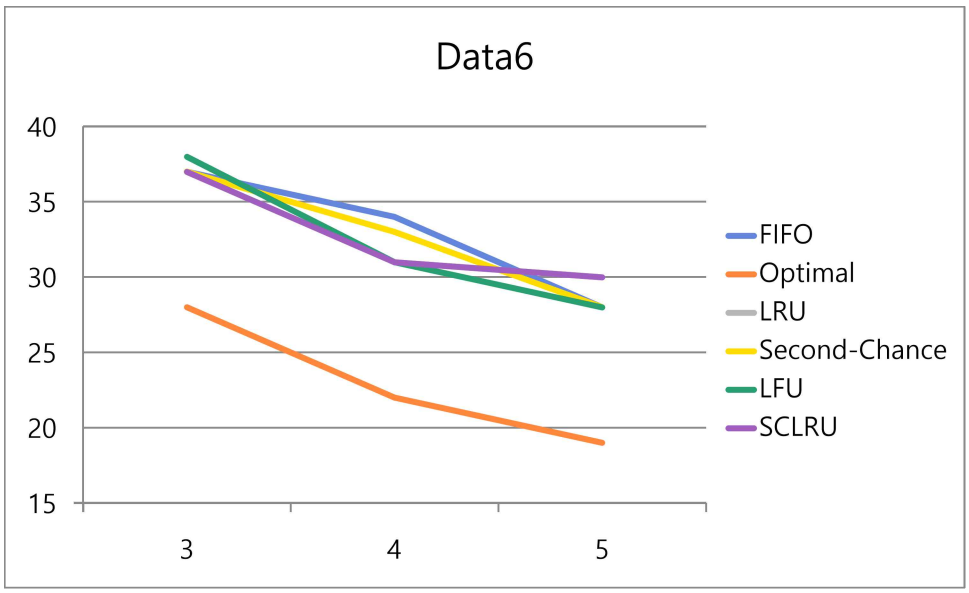
그래프 3 data3의 결과를 표현한 그래프



그래프 4 data4의 결과를 표현한 그래프



그래프 5 data5의 결과를 표현한 그래프



그래프 6 data6의 결과를 표현한 그래프

위의 그래프를 보고 알 수 있는 사실은 Optimal 알고리즘의 Page fault가 다른 알고리즘에 비해 훨씬 적다. 또 프레임의 크기가 커질수록 Page fault의 수가 줄어드는 것을 볼 수 있다. 또 과거의 데이터를 보는 알고리즘들은 결과가 거의 비슷하였다.

새롭게 만든 SCLRU의 경우 LRU의 값과 일치하였다. 그 이유를 분석해본 결과 Second-Chance의 reference bit가 0인 페이지들 중 안 쓴지 가장 오래된 페이지를 교체하는 것은 결국 LRU의 동작과정과 완전히 일치하기 때문이다. 처음에 SCLRU를 개발하게 된 이유는 reference bit중 0인 페이지들 중 안 쓴지 가장 오래된 페이지를 교체하면 성능이 향상하지 않을까라는 생각에서 시작했지만 결과론적으로 그 알고리즘은 LRU와 동일한 것이다.

V. 결론

본 프로젝트는 Page Replacement 알고리즘을 설명하여 이해도 증진을 위해 페이지가 교체되는 과정을 실시간으로 보여주는 시뮬레이터를 설계하였따. 실험용 데이터를 통하여 시뮬레이션을 실행하여 Page Replacement 알고리즘 간의 특징과 결과값을 그래프를 통하여 이해도를 높였다. 이를 통해 도출된 결과는 Optimal 알고리즘이 Page fault의 수가 가장 낮으며 프레임의 크기가 커질수록 Page fault의 수가 줄어든다는 것이다. 또 과거의 데이터를 보는 알고리즘들은 대체로 결과값이 비슷하다는 것이다.

본 프로젝트를 위해 구현한 시뮬레이터의 GUI의 환경은 사용자의 불편함을 느낄 수 있는 구조일 수 있다. 향후 GUI 화면을 잘 구성하여 사용자의 편리성을 높이도록 할 것이다. 또 Page Fault가 난 경우 화면에 표시되는 글씨를 다르게 하여 Page Fault가 발생한 시점을 눈에 띄게 하여 시인성을 더할 것이다. 이를 통해 지속적으로 기능을 개선하면 활용 효과가 극대화될 것이다.

VI. 참고문헌

Abraham Silberschatz, Peter B. Galvin, Greg Gagne, Operating System Concepts 10th Edition
문자열 추출 - https://www.random.org/strings