1. 개요 및 구현방법

일곱 가지 페이지 교체 알고리즘을 C 프로그래밍 언어로 프로그래밍하고 시스템에서 자동 입력과 파일 입력을 사용해 페이지 폴트 횟수를 측정하고 optimal 알고리즘과 비교하라.

페이지 번호는 1번부터 30번까지 존재하고, 참조 페이지 스트림은 프로그램 내부적으로 랜덤하게 생성할 수 있고, 파일로 입력 받을 수 있다. 페이지 프레임 수는 최소 3개에서 최대 10개의 프레임을 사용자에게 입력 받는다. 시뮬레이션 결과는 표준 출력이 되고, 파일로도 저장하는 기능이 제공된다.

다음은 프로그램의 UI 메뉴이다.

1. 프로그램을 실행하면 무슨 알고리즘으로 시뮬레이션을 진행할 것인지 묻는다.

```
Select page replacement algorithm simulator (maximum 3 but (8) can only be selected by itself)
(1) OPTIMAL  (2) FIFO  (3) LIFO  (4) LRU  (5) LFU  (6) SC  (7) ESC  (8) ALL
>
```

2. 3에서 10 사이의 페이지 프레임 수를 입력받는다.

```
Enter number of page frame (3~10)
>
```

3. 데이터를 어떻게 입력할 것인지 선택한다.
1번은 내부적으로 랜덤하게 생성하고, 2번은 파일을 입력으로 받는다.

```
Select how to input data
(1) Randomly  (2) File
>
```

3-1. 1번, 랜덤하게 생성하기를 누르면 스트링 사이즈를 입력받는다.

```
Enter number of string size
>
```

3-2. 2번, 파일로 입력받기를 누르면 파일의 이름을 입력받는다.

```
Enter file name
>
```

4. 각 항목에 맞는 질문 후에는 공통적으로 결과 내용을 output.txt에 저장할 것인지 묻는다.

```
Do you want to save the output to output.txt? (Y/N)
>
```

5. 선택된 시뮬레이터로 시뮬레이션을 한 뒤 시뮬레이션을 계속할 것인지 묻는다.

```
Do you want to be continue the simulator? (Y/N)
>
```

사용자는 파일로 데이터를 입력 시, ESC를 선택하지 않을 경우 [페이지 번호] [페이지 번호] ...로 구성되도록, ESC를 선택할 경우 [페이지 번호] [R/W 여부 - 0 입력 시 read, 1 입력 시 write] ...로 구성되도록 파일의 내용을 구성해야 한다. 예를 들어, ESC 알고리즘을 선택하지 않을 경우 26 3 15 29...와 같이 페이지 번호를 띄어쓰기로 구분한 스트링으로 입력해야 하고, ESC 알고리즘을 선택할 경우 26 1 3 0 15 1 29 1...와 같이 페이지 번호 입력 후 read/write 여부의 형식으로 스트링을 입력해야 한다.

구현한 페이지 교체 알고리즘의 개념

1. FIFO

페이지 프레임에 가장 먼저 들어온, 즉 현재 프레임에서 가장 오래된 페이지를 교체 대상으로 삼는다. 페이지 프레임 배열을 가르키는 포인터를 0부터 마지막 인덱스까지 원형으로 순차적으로 증가시킨다. 현재 포인터가 가르키고 있는 곳에 새로운 페이지를 넣는다.
(참고한 문헌 : 교안 chap6. p24)

2. LIFO

페이지 프레임에 가장 나중에 들어온, 즉 현재 프레임에서 가장 들어온지 얼마 안된 페이지를 교체 대상

으로 삼는다. 페이지 프레임 배열에 0번부터 순차적으로 페이지를 넣으면 결국 마지막 인덱스까지 꽉 채우게 된다. 다음 페이지 폴트가 발생했을 때 가장 나중에 들어온 페이지는 마지막 인덱스의 페이지이다. 그다음 페이지 폴트가 발생해도 가장 나중에 들어온 페이지는 마지막 인덱스의 페이지이다. 따라서 페이지 프레임이 꽉 차고 난 후에는 마지막 인덱스에서만 페이지 폴트가 발생한다.
(참고한 문헌 : https://www.youtube.com/watch?v=QEyNUdcBCfA)

3. LRU

페이지 프레임에서 가장 최근에 참조된 페이지를 교체 대상으로 삼는다. LRU에서는 페이지 프레임을 참조순으로 정렬해 유지한다. 참조한지 오래되면 될수록 0번에 가깝게 위치한다. 즉, 배열의 0번 인덱스에는 참조한지 가장 오래된 페이지가 위치하고, 마지막 인덱스에는 가장 최근에 참조한 페이지가 위치한다. LRU 알고리즘을 사용하면 페이지 폴트가 발생해도 히트가 발생해도 매번 프레임 값의 이동이 있다. 페이지 폴트가 발생하면 0번 인덱스의 페이지를 삭제하고 1번부터 마지막 전 인덱스까지의 페이지를 한 칸 앞으로 이동한다. 이후 마지막 인덱스에 새로운 페이지를 삽입한다. 페이지 히트가 발생하면 hit인 페이지 뒤에 있는 페이지들은 한 칸 앞으로 이동한다. 이후 히트인 페이지를 페이지 프레임의 마지막 인덱스로 이동한다.
(참고한 문헌 : 교안 chap6. p30)

4. LFU

페이지 프레임에서 참조된 횟수가 가장 적은 페이지를 교체 대상으로 삼는다. 가장 적은 참조 횟수를 가진 페이지들이 여러 개일 경우 프레임에 들어온지 가장 오래된 페이지를 선택한다. 즉, 페이지 프레임 내에서 가장 적은 참조 횟수를 가진 페이지들을 교체 대상 후보로 갖고, 이들 중 참조한지 가장 오래된 페이지를 선택하는 것이다.
(참고한 문헌 : https://www.geeksforgeeks.org/page-faults-in-lfu-implementation/
https://www.youtube.com/watch?v=uL0xP57negc)

5. SC

FIFO를 기반으로 두고 있는 알고리즘이다. 기본적으로 페이지 프레임을 원형으로 순회하면서 페이지를 교체한다. 그러나 페이지 hit가 발생한 적이 있는 페이지에게는 교체하지 않고 한 번 더 기회를 준다. 페이지 프레임 entry는 각각 reference bit를 갖고 있는다. 페이지 히트가 발생하면 해당 페이지의 reference bit를 1로 바꿔준다. 페이지 프레임을 순회하다 reference bit가 1인 페이지는 교체하지 않고 reference bit를 0으로 바꿔준 후 다음 페이지를 검사한다. reference bit가 0인 페이지는 교체한다.

6. ESC

reference bit, dirty bit 두 개의 비트를 기반으로 페이지 교체 대상을 정한다. 페이지 스트링이 해당 페이지를 read했는지, write했는지와 함께 입력된다. 예를 들어, 26 R 4 W 15 R... 등의 형식으로 입력된다. 해당 페이지를 read했으면 reference bit가 1로, write했으면 dirty bit를 1로 바꿔준다. 페이지 폴트가 발생하면 페이지 프레임 안의 모든 페이지의 우선순위를 reference bit와 dirty bit로 판단한다. reference bit와 dirty bit가 각각 0과 0, 0과 1, 1과 0, 1과 1순으로 우선순위를 매긴다. reference bit와 dirty bit가 0과 0인 페이지가 가장 교체될 우선순위가 높다. 우선순위가 같은 페이지들 중에서는 circular queue 방식으로 순회하며 가장 먼저 만나는 페이지를 선택한다. reference bit를 주기적으로 0으로 바꿔주기도 하나, 교안에 있는 알고리즘에 따라 현재 페이지 프레임 안의 reference bit, dirty bit를 기반으로 교체 대상을 선정한다.
(참고한 문헌 : 교안 chap6. p33)

7. Optimal

페이지 교체 알고리즘 중 가장 최적화된 알고리즘이다. 페이지 폴트가 발생하면 가장 나중에 참조될 페이지를 교체 대상으로 정한다. 페이지 프레임에 있는 페이지들 각각 다음에 참조될 위치를 계산하고, 이 중 가장 나중에 참조될 페이지를 지정한다.
(참고한 문헌 : 교안 chap6. p27)

2. 결과

1. 입력 에러 처리

번호에 속하지 않는 값을 입력했을 경우

```
Select page replacement algorithm simulator (maximum 3 but (8) can only be selected by itself)
(1) OPTIMAL  (2) FIFO  (3) LIFO  (4) LRU  (5) LFU  (6) SC  (7) ESC  (8) ALL
>9
USAGE : please select 1~8 simulator
Program ended with exit code: 0
```

```
Select page replacement algorithm simulator (maximum 3 but (8) can only be selected by itself)
(1) OPTIMAL  (2) FIFO  (3) LIFO  (4) LRU  (5) LFU  (6) SC  (7) ESC  (8) ALL
>12
USAGE : please select 1~8 simulator
Program ended with exit code: 0
```

```
Select page replacement algorithm simulator (maximum 3 but (8) can only be selected by itself)
(1) OPTIMAL  (2) FIFO  (3) LIFO  (4) LRU  (5) LFU  (6) SC  (7) ESC  (8) ALL
>-3
USAGE : please select 1~8 simulator
Program ended with exit code: 0
```

3개 이상 입력했을 경우

```
Select page replacement algorithm simulator (maximum 3 but (8) can only be selected by itself)
(1) OPTIMAL  (2) FIFO  (3) LIFO  (4) LRU  (5) LFU  (6) SC  (7) ESC  (8) ALL
>1 3 4 5
USAGE : maximum number of choices is 3
Program ended with exit code: 0
```

전체 (8)과 다른 알고리즘을 함께 선택했을 경우

```
Select page replacement algorithm simulator (maximum 3 but (8) can only be selected by itself)
(1) OPTIMAL  (2) FIFO  (3) LIFO  (4) LRU  (5) LFU  (6) SC  (7) ESC  (8) ALL
>1 8
USAGE : (8) should only be selected by itslef
Program ended with exit code: 0
```

알고리즘을 중복으로 선택했을 경우

```
Select page replacement algorithm simulator (maximum 3 but (8) can only be selected by itself)
(1) OPTIMAL  (2) FIFO  (3) LIFO  (4) LRU  (5) LFU  (6) SC  (7) ESC  (8) ALL
>1 2 1
USAGE : duplicated number is not allowed
Program ended with exit code: 0
```

페이지 프레임을 3에서 10 사이로 입력하지 않았을 경우

```
Enter number of page frame (3~10)
>1
USAGE : number of page frame should be between 3 and 10
Program ended with exit code: 0
```

데이터를 입력하는 방식을 1과 2 이외의 수를 입력했을 경우

```
Select how to input data
(1) Randomly  (2) File
>3
USAGE : selecet 1 way or 2 way
Program ended with exit code: 0
```

Y와 N 이외의 문자를 입력했을 경우 다시 입력을 받는다.

```
Do you want to save the output to output.txt? (Y/N)
>Z
USAGE : selecet Y or N

Do you want to save the output to output.txt? (Y/N)
>
```

시뮬레이션이 끝난 후 Y와 N 이외의 문자를 입력했을 경우 다시 입력을 받는다.

```
Do you want to be continue the simulator? (Y/N)
>Z
USAGE : selecet Y or N
Do you want to be continue the simulator? (Y/N)
>
```

2. 시뮬레이션 결과

reference string 7 8 1 2 8 3 8 4 2 3 8 3 8 3 2 1 2 8 1 7 8 1으로 비교적 짧은 reference string으로 알고리즘이 제대로 작동하는지 살펴보았다.

FIFO

```
Select page replacement algorithm simulator (maximum 3 but (8) can only be selected by itself)
(1) OPTIMAL  (2) FIFO  (3) LIFO  (4) LRU  (5) LFU  (6) SC  (7) ESC  (8) ALL
>2

Enter number of page frame (3~10)
>3

Select how to input data
(1) Randomly  (2) File
>2

Enter file name
>input.txt

reference string :
7 8 1 2 8 3 8 4 2 3 8 3 8 3 2 1 2 8 1 7 8 1

Do you want to save the output to output.txt? (Y/N)
>N


==FIFO simulation start==

#1     page : 7     frame :    7    0    0    F
#2     page : 8     frame :    7    8    0    F
#3     page : 1     frame :    7    8    1    F
#4     page : 2     frame :    2    8    1    F
#5     page : 8     frame :    2    8    1
#6     page : 3     frame :    2    3    1    F
#7     page : 8     frame :    2    3    8    F
#8     page : 4     frame :    4    3    8    F
#9     page : 2     frame :    4    2    8    F
#10    page : 3     frame :    4    2    3    F
#11    page : 8     frame :    8    2    3    F
#12    page : 3     frame :    8    2    3
#13    page : 8     frame :    8    2    3
#14    page : 3     frame :    8    2    3
#15    page : 2     frame :    8    2    3
#16    page : 1     frame :    8    1    3    F
#17    page : 2     frame :    8    1    2    F
#18    page : 8     frame :    8    1    2
#19    page : 1     frame :    8    1    2
#20    page : 7     frame :    7    1    2    F
#21    page : 8     frame :    7    8    2    F
#22    page : 1     frame :    7    8    1    F

Total number of page fault 15
==FIFO simulation end==


Do you want to be continue the simulator? (Y/N)
>N
Program ended with exit code: 0
```

페이지에서 참조된지 가장 오래된 페이지를 교체한다. 페이지 7, 8, 1이 삽입된 후 페이지 1을 삽입할 때 가장 오래된 페이지인 7을 제거하고 1을 삽입한다.
교안 chap6. p24 FIFO의 예시와 똑같이 작동하는 것을 확인할 수 있다.

LIFO

```
Select page replacement algorithm simulator (maximum 3 but (8) can only be selected by itself)
(1) OPTIMAL  (2) FIFO  (3) LIFO  (4) LRU  (5) LFU  (6) SC  (7) ESC  (8) ALL
>2

Enter number of page frame (3~10)
>3

Select how to input data
(1) Randomly  (2) File
>2

Enter file name
>input.txt

reference string :
7 8 1 2 8 3 8 4 2 3 8 3 8 3 2 1 2 8 1 7 8 1

Do you want to save the output to output.txt? (Y/N)
>N


==FIFO simulation start==

#1     page : 7     frame :    7    0    0    F
#2     page : 8     frame :    7    8    0    F
#3     page : 1     frame :    7    8    1    F
#4     page : 2     frame :    2    8    1    F
#5     page : 8     frame :    2    8    1
#6     page : 3     frame :    2    3    1    F
#7     page : 8     frame :    2    3    8    F
#8     page : 4     frame :    4    3    8    F
#9     page : 2     frame :    4    2    8    F
#10    page : 3     frame :    4    2    3    F
#11    page : 8     frame :    8    2    3    F
#12    page : 3     frame :    8    2    3
#13    page : 8     frame :    8    2    3
#14    page : 3     frame :    8    2    3
#15    page : 2     frame :    8    2    3
#16    page : 1     frame :    8    1    3    F
#17    page : 2     frame :    8    1    2    F
#18    page : 8     frame :    8    1    2
#19    page : 1     frame :    8    1    2
#20    page : 7     frame :    7    1    2    F
#21    page : 8     frame :    7    8    2    F
#22    page : 1     frame :    7    8    1    F

Total number of page fault 15
==FIFO simulation end==


Do you want to be continue the simulator? (Y/N)
>N
Program ended with exit code: 0
```

페이지 프레임이 꽉 찬 이후에는 페이지 프레임의 마지막 인덱스에서만 페이지 폴트가 발생한다.

LRU

```
Select page replacement algorithm simulator (maximum 3 but (8) can only be selected by itself)
(1) OPTIMAL  (2) FIFO  (3) LIFO  (4) LRU  (5) LFU  (6) SC  (7) ESC  (8) ALL
>4

Enter number of page frame (3~10)
>3

Select how to input data
(1) Randomly  (2) File
>2

Enter file name
>input.txt

reference string :
7 8 1 2 8 3 8 4 2 3 8 3 8 3 2 1 2 8 1 7 8 1

Do you want to save the output to output.txt? (Y/N)
>N


==LRU simulation start==

#1     page : 7     frame :    7    0    0    F
#2     page : 8     frame :    7    8    0    F
#3     page : 1     frame :    7    8    1    F
#4     page : 2     frame :    8    1    2    F
#5     page : 8     frame :    1    2    8
#6     page : 3     frame :    2    8    3    F
#7     page : 8     frame :    2    3    8
#8     page : 4     frame :    3    8    4    F
#9     page : 2     frame :    8    4    2    F
#10    page : 3     frame :    4    2    3    F
#11    page : 8     frame :    2    3    8    F
#12    page : 3     frame :    2    8    3
#13    page : 8     frame :    2    3    8
#14    page : 3     frame :    2    8    3
#15    page : 2     frame :    8    3    2
#16    page : 1     frame :    3    2    1    F
#17    page : 2     frame :    3    1    2
#18    page : 8     frame :    1    2    8    F
#19    page : 1     frame :    2    8    1
#20    page : 7     frame :    8    1    7    F
#21    page : 8     frame :    1    7    8
#22    page : 1     frame :    7    8    1

Total number of page fault 12
==LRU simulation end==


Do you want to be continue the simulator? (Y/N)
>N
Program ended with exit code: 0
```

페이지 프레임을 참조된 순으로 정렬한다.
7, 8, 1 페이지가 들어온 후 페이지 2를 참조하기 위해 페이지 7을 제거하고 8과 1을 한 칸 앞으로 이동시킨 뒤 페이지 2를 맨 마지막 인덱스에 삽입하고 있다. 이후 페이지 8이 히트하면서 참조순이 1, 2, 8로 바뀌었다. 페이지 3을 삽입하기 위해서 마찬가지로 맨 앞의 가장 오래된 페이지를 삭제하고 맨 뒤에 3을 삽입한다.
교안 chap6. p28과 같이 페이지 폴트가 12번 발생한다.

LFU

```
Select page replacement algorithm simulator (maximum 3 but (8) can only be selected by itself)
(1) OPTIMAL  (2) FIFO  (3) LIFO  (4) LRU  (5) LFU  (6) SC  (7) ESC  (8) ALL
>5

Enter number of page frame (3~10)
>3

Select how to input data
(1) Randomly  (2) File
>2

Enter file name
>input.txt

reference string :
7 8 1 2 8 3 8 4 2 3 8 3 8 3 2 1 2 8 1 7 8 1

Do you want to save the output to output.txt? (Y/N)
>N


==LFU simulation start==

#1     page : 7     frame :    7    0    0    F
#2     page : 8     frame :    7    8    0    F
#3     page : 1     frame :    7    8    1    F
#4     page : 2     frame :    2    8    1    F
#5     page : 8     frame :    2    8    1
#6     page : 3     frame :    2    8    3    F
#7     page : 8     frame :    2    8    3
#8     page : 4     frame :    4    8    3    F
#9     page : 2     frame :    4    8    2    F
#10    page : 3     frame :    3    8    2    F
#11    page : 8     frame :    3    8    2
#12    page : 3     frame :    3    8    2
#13    page : 8     frame :    3    8    2
#14    page : 3     frame :    3    8    2
#15    page : 2     frame :    3    8    2
#16    page : 1     frame :    3    8    1    F
#17    page : 2     frame :    3    8    2    F
#18    page : 8     frame :    3    8    2
#19    page : 1     frame :    3    8    1    F
#20    page : 7     frame :    3    8    7    F
#21    page : 8     frame :    3    8    7
#22    page : 1     frame :    3    8    1    F

Total number of page fault 13
==LFU simulation end==


Do you want to be continue the simulator? (Y/N)
>N
Program ended with exit code: 0
```

페이지 7, 8, 1이 들어온 후 페이지 2번을 참조하기 위해 참조한지 가장 오래된 페이지 7번을 제거하고 2번을 삽입한다. 페이지 8번이 히트되면서 참조 횟수가 2로 증가한다. 그래서 다음 교체 대상은 2, 8, 1 중 참조 횟수가 적은 2와 1로 줄어들고, 2와 1 중 참조한지 더 오래된 1이 교체 대상이 된다.

SC

```
Select page replacement algorithm simulator (maximum 3 but (8) can only be selected by itself)
(1) OPTIMAL  (2) FIFO  (3) LIFO  (4) LRU  (5) LFU  (6) SC  (7) ESC  (8) ALL
>6

Enter number of page frame (3~10)
>3

Select how to input data
(1) Randomly  (2) File
>2

Enter file name
>input.txt

reference string :
7 8 1 2 8 3 8 4 2 3 8 3 8 3 2 1 2 8 1 7 8 1

Do you want to save the output to output.txt? (Y/N)
>N


==SC simulation start==

#1     page : 7     frame :    7    0    0    F
#2     page : 8     frame :    7    8    0    F
#3     page : 1     frame :    7    8    1    F
#4     page : 2     frame :    2    8    1    F
#5     page : 8     frame :    2    8    1
#6     page : 3     frame :    2    8    3    F
#7     page : 8     frame :    2    8    3
#8     page : 4     frame :    4    8    3    F
#9     page : 2     frame :    4    8    2    F
#10    page : 3     frame :    3    8    2    F
#11    page : 8     frame :    3    8    2
#12    page : 3     frame :    3    8    2
#13    page : 8     frame :    3    8    2
#14    page : 3     frame :    3    8    2
#15    page : 2     frame :    3    8    2
#16    page : 1     frame :    3    1    2    F
#17    page : 2     frame :    3    1    2
#18    page : 8     frame :    8    1    2    F
#19    page : 1     frame :    8    1    2
#20    page : 7     frame :    8    1    7    F
#21    page : 8     frame :    8    1    7
#22    page : 1     frame :    8    1    7

Total number of page fault 11
==SC simulation end==


Do you want to be continue the simulator? (Y/N)
>N
Program ended with exit code: 0
```

SC에서는 페이지 히트가 일어나면 reference bit를 1로 변경한다. FIFO로 페이지 프레임을 원형으로 순회하며 reference bit가 0인 페이지를 교체 대상으로 삼는다. 7, 8, 1 페이지가 삽입된 후 2번 페이지는 참조한지 가장 오래된 7번 페이지와 교체된다. 이후 8번 페이지가 히트하면 reference bit를 1로 바꾼다. 따라서 다음 페이지 3번은 2, 8, 3 중 8번이 아닌 1번 페이지와 교체된다. 8번 페이지가 또다시 히트하면 reference bit를 다시 0에서 1로 바꾼다. 다음 페이지 4번은 2번과 교체된다. 다음 페이지 2번은 8번의 reference bit가 1이므로 3번과 교체된다.
11번째 페이지 프레임 3, 8, 2에서 다음 교체 대상인 페이지는 8번 페이지이며, 3, 8, 2 페이지 모두 히트가 일어나 reference bit가 1로 바뀐다. 페이지 1번에서 폴트가 발생할 때 교체 대상인 8번 페이지의 reference bit가 1이므로 0으로 바꿔준 뒤 다음 페이지를 검사한다. 2번 또한 reference bit가 1이므로 0으로 바꿔준 뒤 다음 페이지를 검사한다. 3번 페이지 역시 reference bit가 1이므로 0으로 바꿔준 뒤 다음 페이지인 8번 페이지를 검사한

다. 1바퀴를 순회한 뒤 8번 페이지의 reference bit는 0이므로 8번을 교체 대상으로 삼는다.

ESC

```
Select page replacement algorithm simulator (maximum 3 but (8) can only be selected by itself)
(1) OPTIMAL  (2) FIFO  (3) LIFO  (4) LRU  (5) LFU  (6) SC  (7) ESC  (8) ALL
>7

Enter number of page frame (3~10)
>3

Select how to input data
(1) Randomly  (2) File
>2

Enter file name
>input.txt

reference string :
7 W 8 R 1 W 2 R 8 W 3 W 8 R 4 W 2 W 3 R 8 R 3 W 8 W 3 W 2 R 1 W 2 R 8 W 1 R 7 W 8 R 1 W

Do you want to save the output to output.txt? (Y/N)
>N

==ESC simulation start==

#1     page : 7    bit : W   frame :     7    0    0    F
#2     page : 8    bit : R   frame :     7    8    0    F
#3     page : 1    bit : W   frame :     7    8    1    F
#4     page : 2    bit : R   frame :     2    8    1    F
#5     page : 8    bit : W   frame :     2    8    1
#6     page : 3    bit : W   frame :     2    8    3    F
#7     page : 8    bit : R   frame :     2    8    3
#8     page : 4    bit : W   frame :     2    8    4    F
#9     page : 2    bit : W   frame :     2    8    4
#10    page : 3    bit : R   frame :     2    8    3    F
#11    page : 8    bit : R   frame :     2    8    3
#12    page : 3    bit : W   frame :     2    8    3
#13    page : 8    bit : W   frame :     2    8    3
#14    page : 3    bit : W   frame :     2    8    3
#15    page : 2    bit : R   frame :     2    8    3
#16    page : 1    bit : W   frame :     1    8    3    F
#17    page : 2    bit : R   frame :     2    8    3    F
#18    page : 8    bit : W   frame :     2    8    3
#19    page : 1    bit : R   frame :     1    8    3    F
#20    page : 7    bit : W   frame :     7    8    3    F
#21    page : 8    bit : R   frame :     7    8    3
#22    page : 1    bit : W   frame :     1    8    3    F

Total number of page fault 12
==ESC simulation end==


Do you want to be continue the simulator? (Y/N)
>N
Program ended with exit code: 0
```

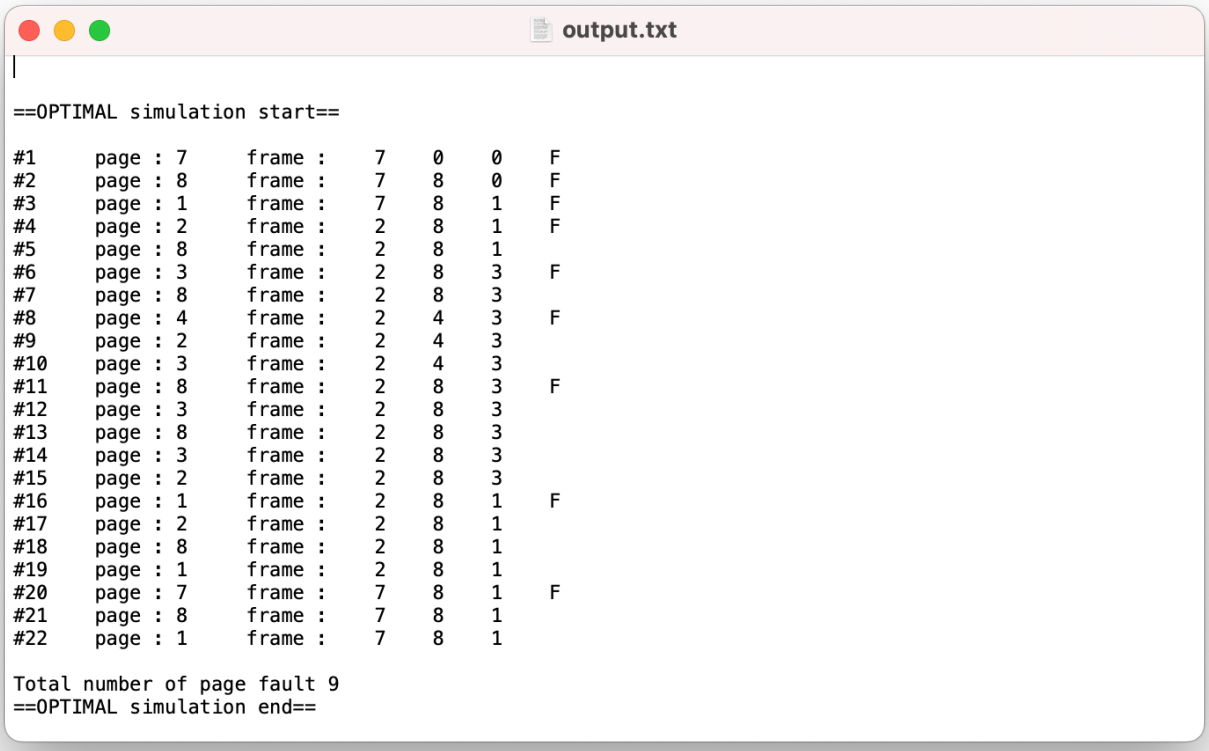
7, 8, 1 페이지의 우선순위는 각각 1, 2, 1과 같다. 우선순위가 같은 페이지 중 먼저 들어온 7번이 교체 대상인 따라서 7번 페이지와 다음 2번 페이지가 교체된다. 8번 페이지가 히트하면서 reference bit가 1로 바뀌어 우선순위가 3으로 바뀐다. 2, 8, 1 페이지의 우선순위는 각각 2, 3, 1과 같다. 따라서 다음 페이지 3번은 1번 페이지와 교체한다.

Optimal

```
Select page replacement algorithm simulator (maximum 3 but (8) can only be selected by itself)
(1) OPTIMAL  (2) FIFO  (3) LIFO  (4) LRU  (5) LFU  (6) SC  (7) ESC  (8) ALL
>1

Enter number of page frame (3~10)
>3

Select how to input data
(1) Randomly  (2) File
>2

Enter file name
>input.txt

reference string :
7 8 1 2 8 3 8 4 2 3 8 3 8 3 2 1 2 8 1 7 8 1

Do you want to save the output to output.txt? (Y/N)
>Y


==OPTIMAL simulation start==

#1     page : 7     frame :    7    0    0    F
#2     page : 8     frame :    7    8    0    F
#3     page : 1     frame :    7    8    1    F
#4     page : 2     frame :    2    8    1    F
#5     page : 8     frame :    2    8    1
#6     page : 3     frame :    2    8    3    F
#7     page : 8     frame :    2    8    3
#8     page : 4     frame :    2    4    3    F
#9     page : 2     frame :    2    4    3
#10    page : 3     frame :    2    4    3
#11    page : 8     frame :    2    8    3    F
#12    page : 3     frame :    2    8    3
#13    page : 8     frame :    2    8    3
#14    page : 3     frame :    2    8    3
#15    page : 2     frame :    2    8    3
#16    page : 1     frame :    2    8    1    F
#17    page : 2     frame :    2    8    1
#18    page : 8     frame :    2    8    1
#19    page : 1     frame :    2    8    1
#20    page : 7     frame :    7    8    1    F
#21    page : 8     frame :    7    8    1
#22    page : 1     frame :    7    8    1

Total number of page fault 9
==OPTIMAL simulation end==


Do you want to be continue the simulator? (Y/N)
>N
Program ended with exit code: 0
```

optimal 알고리즘은 페이지 폴트 수가 가장 적게 일어나도록 최적화된 알고리즘이다. 7, 8, 1 페이지들 중 가장 나중에 다시 참조되는 7번 페이지를 교체 대상으로 삼는다. 2, 8, 1 페이지들 중에는 1번 페이지를 가장 나중에 다시 참조한다. 따라서 1번 페이지와 다음 페이지를 교체한다.

다음은 output.txt 파일의 내용이다. output.txt 파일로 알고리즘 시뮬레이션 결과가 저장된다.



두 가지, 세 가지 시뮬레이션을 선택했을 경우

입력 string이 최소 500개로 설정된 것을 알고 있으나, 출력 결과를 보고서에 첨부하기 위해 작게 설정하였다.

```
Select page replacement algorithm simulator (maximum 3 but (8) can only be selected by itself)
(1) OPTIMAL  (2) FIFO  (3) LIFO  (4) LRU  (5) LFU  (6) SC  (7) ESC  (8) ALL
>1 2

Enter number of page frame (3~10)
>4

Select how to input data
(1) Randomly  (2) File
>1

Enter number of string size
>10

reference string :
24 2 20 26 14 12 14 29 24 3

Do you want to save the output to output.txt? (Y/N)
>N
USAGE : selecet Y or N

Do you want to save the output to output.txt? (Y/N)
>USAGE : selecet Y or N

Do you want to save the output to output.txt? (Y/N)
>

==OPTIMAL simulation start==

#1     page : 24    frame :   24    0    0    0    F
#2     page : 2     frame :   24    2    0    0    F
#3     page : 20    frame :   24    2   20    0    F
#4     page : 26    frame :   24    2   20   26    F
#5     page : 14    frame :   24   14   20   26    F
#6     page : 12    frame :   24   14   12   26    F
#7     page : 14    frame :   24   14   12   26
#8     page : 29    frame :   24   29   12   26    F
#9     page : 24    frame :   24   29   12   26
#10    page : 3     frame :    3   29   12   26    F

Total number of page fault 8
==OPTIMAL simulation end==




==FIFO simulation start==

#1     page : 24    frame :   24    0    0    0    F
#2     page : 2     frame :   24    2    0    0    F
#3     page : 20    frame :   24    2   20    0    F
#4     page : 26    frame :   24    2   20   26    F
#5     page : 14    frame :   14    2   20   26    F
#6     page : 12    frame :   14   12   20   26    F
#7     page : 14    frame :   14   12   20   26
#8     page : 29    frame :   14   12   29   26    F
#9     page : 24    frame :   14   12   29   24    F
#10    page : 3     frame :    3   12   29   24    F

Total number of page fault 9
==FIFO simulation end==


Do you want to be continue the simulator? (Y/N)
```

```
Select page replacement algorithm simulator (maximum 3 but (8) can only be selected by itself)
(1) OPTIMAL  (2) FIFO  (3) LIFO  (4) LRU  (5) LFU  (6) SC  (7) ESC  (8) ALL
>7 4 2

Enter number of page frame (3~10)
>5

Select how to input data
(1) Randomly  (2) File
>1

Enter number of string size
>10

reference string :
29 R 25 W 3 R 6 W 10 W 29 R 5 R 4 R 23 R 17 R

Do you want to save the output to output.txt? (Y/N)
>N

==ESC simulation start==

#1     page : 29   bit : R   frame :    29    0    0    0    0    F
#2     page : 25   bit : W   frame :    29   25    0    0    0    F
#3     page : 3    bit : R   frame :    29   25    3    0    0    F
#4     page : 6    bit : W   frame :    29   25    3    6    0    F
#5     page : 10   bit : W   frame :    29   25    3    6   10    F
#6     page : 29   bit : R   frame :    29   25    3    6   10
#7     page : 5    bit : R   frame :    29    5    3    6   10    F
#8     page : 4    bit : R   frame :    29    5    3    4   10    F
#9     page : 23   bit : R   frame :    29    5    3    4   23    F
#10    page : 17   bit : R   frame :    17    5    3    4   23    F

Total number of page fault 9
==ESC simulation end==



==LRU simulation start==

#1     page : 29    frame :   29    0    0    0    0    F
#2     page : 25    frame :   29   25    0    0    0    F
#3     page : 3     frame :   29   25    3    0    0    F
#4     page : 6     frame :   29   25    3    6    0    F
#5     page : 10    frame :   29   25    3    6   10    F
#6     page : 29    frame :   25    3    6   10   29
#7     page : 5     frame :    3    6   10   29    5    F
#8     page : 4     frame :    6   10   29    5    4    F
#9     page : 23    frame :   10   29    5    4   23    F
#10    page : 17    frame :   29    5    4   23   17    F

Total number of page fault 9
==LRU simulation end==
```

```
==FIFO simulation start==

#1     page : 29    frame :   29    0    0    0    0    F
#2     page : 25    frame :   29   25    0    0    0    F
#3     page : 3     frame :   29   25    3    0    0    F
#4     page : 6     frame :   29   25    3    6    0    F
#5     page : 10    frame :   29   25    3    6   10    F
#6     page : 29    frame :   29   25    3    6   10
#7     page : 5     frame :    5   25    3    6   10    F
#8     page : 4     frame :    5    4    3    6   10    F
#9     page : 23    frame :    5    4   23    6   10    F
#10    page : 17    frame :    5    4   23   17   10    F

Total number of page fault 9
==FIFO simulation end==


Do you want to be continue the simulator? (Y/N)
>N
Program ended with exit code: 0
```

두가지, 세가지 경우를 선택했을 때에도 시뮬레이션 결과가 잘 선택되는 것을 확인할 수 있다.

구현한 알고리즘의 성능 비교 분석

reference string을 500개로 설정해 optimal 알고리즘과 각각의 알고리즘의 페이지 폴트 횟수를 비교해보았다.

```
Select page replacement algorithm simulator (maximum 3 but (8) can only be selected by itself)
(1) OPTIMAL  (2) FIFO  (3) LIFO  (4) LRU  (5) LFU  (6) SC  (7) ESC  (8) ALL
>8

Enter number of page frame (3~10)
>5

Select how to input data
(1) Randomly  (2) File
>1

Enter number of string size
>500

reference string :
4 W 1 R 2 W 7 W 26 R 22 W 29 R 2 R 21 R 18 W 30 R 7 W 19 W 29 R 11 W 7 W 24 W 12 W 28 R 23 W 8 R 16 R 23 R 16 W 13 W 12 W
    28 W 13 W 14 R 5 R 4 R 18 W 13 W 19 R 20 R 5 W 5 W 21 R 21 R 14 R 21 W 13 R 2 W 18 R 11 R 17 R 25 W 22 R 3 R 16 R 30
    R 24 R 17 W 13 R 8 R 20 W 20 R 2 R 25 W 25 R 15 W 17 W 26 R 30 R 13 W 14 R 9 R 21 R 29 W 30 R 30 R 6 W 27 R 25 R 26 R
    26 W 17 R 3 R 7 R 16 R 8 R 9 R 13 W 23 W 13 R 20 R 5 W 3 R 28 R 13 R 27 R 23 W 11 W 22 R 25 R 5 R 29 W 12 R 9 W 9 W
    20 W 22 W 13 R 14 R 5 W 28 W 28 R 27 R 6 R 12 R 4 W 16 W 5 W 8 W 6 R 5 W 28 W 3 R 25 R 10 R 13 R 26 R 18 W 29 R 14 W
    9 R 28 W 15 R 12 W 20 R 14 R 9 W 24 R 10 R 11 W 11 W 25 W 15 R 22 R 14 W 27 R 1 R 21 R 6 R 30 R 25 R 3 R 6 R 4 R 14 W
    23 R 5 R 10 W 7 R 14 W 5 R 22 R 16 R 27 R 18 R 18 R 24 R 25 R 25 R 25 R 10 R 2 W 29 R 26 R 25 R 28 W 1 R 12 W 30 R 30
    R 3 W 20 W 14 W 24 W 23 R 3 W 3 R 18 R 13 W 2 W 24 W 10 R 17 W 9 W 4 R 7 W 10 R 1 R 19 R 3 W 18 R 13 W 14 R 21 W 14 R
    11 R 3 R 7 W 26 R 21 W 7 R 28 R 28 W 5 W 24 W 1 W 9 W 26 R 4 R 28 R 22 W 25 W 21 R 1 W 17 R 21 W 17 W 3 R 22 W 19 W
    22 W 19 W 10 R 30 R 9 R 10 R 5 R 10 R 7 R 30 W 13 R 5 W 21 W 17 R 25 W 19 R 11 W 4 W 16 W 12 W 28 W 12 R 9 R 7 R 25 W
    4 W 10 R 27 W 10 R 25 R 25 R 11 R 2 W 14 R 10 W 23 W 26 W 25 W 28 R 21 R 19 R 17 W 6 R 28 W 16 W 15 W 17 W 3 W 27 W
    25 W 29 R 12 R 19 W 24 R 7 W 10 R 7 R 9 R 8 R 27 W 20 R 26 W 26 R 13 W 3 R 13 R 3 R 15 W 30 W 20 W 11 R 15 W 12 W 29
    R 28 R 29 R 2 W 20 R 14 W 11 W 11 R 14 R 15 W 12 W 4 R 9 R 17 R 24 R 21 R 24 W 9 W 5 W 22 W 23 W 1 R 23 W 29 W 4 R 28
    W 21 W 29 R 8 W 19 W 11 R 8 R 20 R 2 W 21 R 17 W 24 R 23 R 2 W 29 R 13 W 29 W 18 W 29 R 10 R 9 W 17 W 10 W 5 R 7 R 29
    W 18 R 28 W 9 R 26 W 9 R 16 R 16 R 25 W 2 R 30 W 16 W 28 R 23 W 6 W 5 R 25 R 16 W 13 W 11 R 1 W 3 R 11 W 8 W 16 W 6 W
    25 R 21 W 6 R 30 R 12 R 2 R 18 W 29 R 3 R 26 W 29 R 29 W 30 R 17 R 20 R 27 W 18 R 4 W 24 W 17 R 17 R 21 W 14 W 3 R 6
    R 3 W 18 R 30 R 17 W 19 W 1 W 25 W 20 R 3 W 6 W 12 R 10 R 23 R 18 R 5 R 10 R 6 R 25 W 15 R 30 W 1 R 19 R 22 W 10 R 1
    R 20 R 30 W 11 W 4 R 13 W 24 W 29 R 1 W 3 W 7 W 28 R 28 W 5 W 23 W 12 R 2 R 14 R 18 R 16 W 29 R 22 W 5 W 12 W 14 W 20
    R 12 R 27 W 11 W 30 W 2 R 5 W 13 W 25 W 16 R 11 W 4 R 8 W 30 R 11 W 3 R 4 W 20 R 30 R 12 W 4 R 19 W 10 R 18 R 28 W 2
    R 26 R 27 W 23 W 16 R 26 R 25 R 11 R 5 W 20 R 2 W 16 R 14 R 14 W 27 W 21 W 28 W 12 W 13 W 29 R 10 W 22 W 11 R 6 W 7 R
    1 R 30 R

Do you want to save the output to output.txt? (Y/N)
>Y
```

다음은 표준 출력 내용 중 페이지 폴트 횟수에 관한 부분이다.

```
Total number of page fault 294
==OPTIMAL simulation end==
```

```
Total number of page fault 417
==FIFO simulation end==

Total number of page fault 421
==LIFO simulation end==

Total number of page fault 416
==LRU simulation end==

Total number of page fault 403
==LFU simulation end==

Total number of page fault 416
==SC simulation end==

Total number of page fault 426
==ESC simulation end==
```

optimal 알고리즘이 월등히 적고 나머지 알고리즘은 LFU < SC = LRU < FIFO < LIFO < ESC 순으로 관찰되었다.

좀 더 체계적인 비교 분석을 해보았다.
알고리즘만의 성능을 분석하기 위해 페이지 프레임은 5개로 고정해놓고 string의 길이를 500, 750, 1000, 1500으로 설정해 표로 나타내보았다.

	optimal	FIFO	LIFO	LRU	LFU	SC	ESC
500	298	420	425	416	418	414	421
750	435	612	623	616	624	617	636
1000	592	838	822	831	833	832	837
1500	880	1239	1255	1239	1254	1239	1268

string 500 : optimal < LRU < SC < LFU < FIFO < ESC < LIFO
string 750 : optimal < FIFO < LRU < SC < LIFO < LFU < ESC
string 1000 : optimal < LIFO < LRU < SC < LFU < ESC < FIFO
string 1500 : optimal < FIFO = LRU = SC < LFU < LIFO < ESC

정리해보자면, optimal이 역시 가장 성능이 좋다. 그 다음 LRU와 SC가 4가지 경우에서 모두 균일하게 좋은 성능을 보인다. ESC는 4가지 경우에서 균일하게 제일 안좋은 성능을 보인다. LFU는 가장 평균적인 성능을 보인다. FIFO와 LIFO는 경우에 따라 편차가 심해 보인다.

다음은 페이지 프레임 갯수에 따른 알고리즘 성능을 파악하기 위해 string의 길이를 1000으로 고정하고 페이지 프레임 수를 각각 3, 5, 7, 10으로 설정해보았다.

	optimal	FIFO	LIFO	LRU	LFU	SC	ESC
3	721	908	914	908	915	906	895
5	586	832	825	824	851	824	793
7	484	760	783	761	772	758	784
10	383	683	683	693	683	685	666

페이지 프레임 3개 : optimal < ESC < SC < FIFO = LRU < LRU < LIFO
페이지 프레임 5개 : optimal < ESC < SC = LRU < LIFO < FIFO < LFU
페이지 프레임 7개 : optimal < SC < FIFO < LRU < LFU < LIFO < ESC
페이지 프레임 10개 : optimal < ESC < FIFO = LIFO = LFU < SC < LRU

페이지 프레임 수가 10개로 증가했더니 LRU의 성능이 안좋아졌다. 앞의 실험에서는 ESC의 성능이 제일 좋지 않았으나 지금 실험에서는 좋은 성능을 보인다. 페이지 프레임의 수가 증가할수록 FIFO, LIFO의 성능이 더 좋아진다.
optimal 알고리즘과 비교해보면, 페이지 프레임 수가 증가할수록 다른 알고리즘보다 2배에 가깝게 성능이 좋아진다. 페이지 프레임이 3개일 때는 1.3배 성능이 좋았지만 페이지 프레임이 10개일 때에는 1.7배 성능이 더 좋아졌다. 페이지 프레임 수가 늘어난다는 것은 리소스를 더 할당한다는 뜻인데, 물리적인 리소스를 더 할당받으니 성능이 좋아지는 것이 올바른 결과가 도출되는 것을 확인했다.

구현한 알고리즘의 Pseudo code

1. FIFO

```
for i=0 to REFERENCE_SIZE do
        if 페이지 프레임 안에 현재 페이지가 존재하지 않으면 then
                현재 인덱스가 가르키는 위치에 현재 페이지 삽입
                modular 연산을 통해 인덱스 한 칸 증가
        endif
endfor
```

2. LIFO

```
for i=0 to REFERENCE_SIZE do
        if 페이지 프레임 안에 현재 페이지가 존재하지 않으면 then
                현재 인덱스가 가르키는 위치에 현재 페이지 삽입
                if 페이지 프레임이 비어있다면 then
                        인덱스 한 칸 증가
                endif
        endif
endfor
```

3. LRU

```
for i=0 to REFERENCE_SIZE do
        if 페이지 프레임 안에 현재 페이지가 존재하지 않으면 then
                if 페이지 프레임이 비어있다면 then
                        현재 인덱스가 가르키는 위치에 현재 페이지 삽입
                        인덱스 한 칸 증가
                else
                        1번 인덱스부터 마지막 인덱스에 있는 페이지 한 칸씩 앞으로 이동
                        맨 마지막 인덱스에 현재 페이지 삽입
                endif
        else
                히트한 페이지 뒤에 있는 페이지들을 한 칸씩 앞으로 이동
                히트한 페이지를 마지막 인덱스에 삽입
        endif
endfor
```

4. LFU

```
for i=0 to REFERENCE_SIZE do
        현재 페이지에 대한 참조횟수와 현재 시간 기록
        페이지 프레임 안에서 가장 적게 참조한 페이지 횟수 저장

        if 페이지 프레임 안에 현재 페이지가 존재하지 않으면 then
                if 페이지 프레임이 비어있다면 then
                        현재 인덱스가 가르키는 위치에 현재 페이지 삽입
                        인덱스 한 칸 증가
                else
                        페이지 프레임 내에서 가장 적게 참조되고 들어온지 오래된 페이지 찾기
                        해당 페이지로 새로운 페이지 교체
                        새로운 페이지에 대한 참조횟수 초기화
                endif
        endif
endfor
```

5. SC

```
for i=0 to REFERENCE_SIZE do
        if 페이지 프레임 안에 현재 페이지가 존재하지 않으면 then
                if 페이지 프레임이 비어있다면 then
                        현재 인덱스가 가르키는 위치에 현재 페이지 삽입
                        인덱스 한 칸 증가
```

```
                else
                        reference bit가 1이면 0으로 만들면서 reference bit가 0인 페이지 찾기
                        해당 페이지에 새로운 페이지 교체
                        modular 연산을 통해 인덱스 한 칸 증가
                endif
        else
                히트한 페이지의 reference bit를 1로 설정
        endif
endfor

6. ESC
for i=0 to REFERENCE_SIZE do
        현재 페이지 번호, R/W 여부 기록

        if 페이지 프레임 안에 현재 페이지가 존재하지 않으면 then
                if 페이지 프레임이 비어있다면 then
                        현재 인덱스가 가르키는 위치에 현재 페이지 삽입
                        인덱스 한 칸 증가
                                if 현재 페이지를 read 했으면 then
                                        reference bit를 1로 설정
                                else
                                        dirty bit를 1로 설정
                                endif
                else
                        페이지 프레임에서 가장 낮은 우선순위를 갖는 페이지 찾기
                        해당 페이지를 새로운 페이지로 교체
                                if 현재 페이지를 read 했으면 then
                                        reference bit를 1로 설정
                                else
                                        dirty bit를 1로 설정
                                endif
                        modular 연산을 통해 인덱스 한 칸 증가
                endif
        else
                if 히트한 페이지를 read 했으면 then
                        reference bit를 1로 설정
                else
                        dirty bit를 1로 설정
                endif
        endif
endfor

7. Optimal
for i=0 to REFERENCE_SIZE do
        if 페이지 프레임 안에 현재 페이지가 존재하지 않으면 then
                if 페이지 프레임이 비어있다면 then
                        현재 인덱스가 가르키는 위치에 현재 페이지 삽입
                        인덱스 한 칸 증가
                else
                        페이지 프레임 내에서 가장 나중에 참조될 예정인 페이지 찾기
                        해당 페이지를 현재 페이지로 교체
                endif
        endif
endfor
```