[REPORT]

■과 목 명: 운영체제 (SWE3004-42)

■담 당 교 수: 신동군 교수님

■제 출 일: 2022년 4월 8일

■학　　과: 수학과

■학　　번: 2016314786

■성　　명: 김호진

Chap 22. Paging Policy

수학과 김호진 (2016314786)

1. Generate random addresses with the following arguments: -s 0 -n 10**,** -s 1 -n 10**, and** -s 2 -n 10**. Change the policy from FIFO, to LRU, to OPT. Compute whether each access in said address traces are hits or misses.**

(1) with FIFO

arguments: -s 0 -n 10을 실행한 결과, hit rate는 10%를 기록하였다.

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 0 -n 10 -c --policy=FIFO
ARG addresses -1
ARG addressfile
ARG numaddrs 10
ARG policy FIFO
ARG clockbits 2
ARG cachesize 3
ARG maxpage 10
ARG seed 0
ARG notrace False

Solving...

Access: 8  MISS FirstIn ->         [8] <- Lastin  Replaced:- [Hits:0 Misses:1]
Access: 7  MISS FirstIn ->      [8, 7] <- Lastin  Replaced:- [Hits:0 Misses:2]
Access: 4  MISS FirstIn ->   [8, 7, 4] <- Lastin  Replaced:- [Hits:0 Misses:3]
Access: 2  MISS FirstIn ->   [7, 4, 2] <- Lastin  Replaced:8 [Hits:0 Misses:4]
Access: 5  MISS FirstIn ->   [4, 2, 5] <- Lastin  Replaced:7 [Hits:0 Misses:5]
Access: 4  HIT  FirstIn ->   [4, 2, 5] <- Lastin  Replaced:- [Hits:1 Misses:5]
Access: 7  MISS FirstIn ->   [2, 5, 7] <- Lastin  Replaced:4 [Hits:1 Misses:6]
Access: 3  MISS FirstIn ->   [5, 7, 3] <- Lastin  Replaced:2 [Hits:1 Misses:7]
Access: 4  MISS FirstIn ->   [7, 3, 4] <- Lastin  Replaced:5 [Hits:1 Misses:8]
Access: 5  MISS FirstIn ->   [3, 4, 5] <- Lastin  Replaced:7 [Hits:1 Misses:9]

FINALSTATS hits 1   misses 9   hitrate 10.00
```

arguments: -s 1 -n 10을 실행한 결과, hit rate는 20%를 기록하였다.

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 1 -n 10 -c --policy=FIFO
ARG addresses -1
ARG addressfile
ARG numaddrs 10
ARG policy FIFO
ARG clockbits 2
ARG cachesize 3
ARG maxpage 10
ARG seed 1
ARG notrace False

Solving...

Access: 1  MISS FirstIn ->         [1] <- Lastin  Replaced:- [Hits:0 Misses:1]
Access: 8  MISS FirstIn ->      [1, 8] <- Lastin  Replaced:- [Hits:0 Misses:2]
Access: 7  MISS FirstIn ->   [1, 8, 7] <- Lastin  Replaced:- [Hits:0 Misses:3]
Access: 2  MISS FirstIn ->   [8, 7, 2] <- Lastin  Replaced:1 [Hits:0 Misses:4]
Access: 4  MISS FirstIn ->   [7, 2, 4] <- Lastin  Replaced:8 [Hits:0 Misses:5]
Access: 4  HIT  FirstIn ->   [7, 2, 4] <- Lastin  Replaced:- [Hits:1 Misses:5]
Access: 6  MISS FirstIn ->   [2, 4, 6] <- Lastin  Replaced:7 [Hits:1 Misses:6]
Access: 7  MISS FirstIn ->   [4, 6, 7] <- Lastin  Replaced:2 [Hits:1 Misses:7]
Access: 0  MISS FirstIn ->   [6, 7, 0] <- Lastin  Replaced:4 [Hits:1 Misses:8]
Access: 0  HIT  FirstIn ->   [6, 7, 0] <- Lastin  Replaced:- [Hits:2 Misses:8]

FINALSTATS hits 2   misses 8   hitrate 20.00
```

arguments: -s 2 -n 10을 실행한 결과, hit rate는 40%를 기록하였다.

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 2 -n 10 -c --policy=FIFO
ARG addresses -1
ARG addressfile
ARG numaddrs 10
ARG policy FIFO
ARG clockbits 2
ARG cachesize 3
ARG maxpage 10
ARG seed 2
ARG notrace False

Solving...

Access: 9  MISS FirstIn ->         [9] <- Lastin  Replaced:- [Hits:0 Misses:1]
Access: 9  HIT  FirstIn ->         [9] <- Lastin  Replaced:- [Hits:1 Misses:1]
Access: 0  MISS FirstIn ->      [9, 0] <- Lastin  Replaced:- [Hits:1 Misses:2]
Access: 0  HIT  FirstIn ->      [9, 0] <- Lastin  Replaced:- [Hits:2 Misses:2]
Access: 8  MISS FirstIn ->   [9, 0, 8] <- Lastin  Replaced:- [Hits:2 Misses:3]
Access: 7  MISS FirstIn ->   [0, 8, 7] <- Lastin  Replaced:9 [Hits:2 Misses:4]
Access: 6  MISS FirstIn ->   [8, 7, 6] <- Lastin  Replaced:0 [Hits:2 Misses:5]
Access: 3  MISS FirstIn ->   [7, 6, 3] <- Lastin  Replaced:8 [Hits:2 Misses:6]
Access: 6  HIT  FirstIn ->   [7, 6, 3] <- Lastin  Replaced:- [Hits:3 Misses:6]
Access: 6  HIT  FirstIn ->   [7, 6, 3] <- Lastin  Replaced:- [Hits:4 Misses:6]

FINALSTATS hits 4   misses 6   hitrate 40.00
```

(2) with LRU

arguments: -s 0 -n 10을 실행한 결과, hit rate는 20%를 기록하였다.

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 0 -n 10 -c --policy=LRU
ARG addresses -1
ARG addressfile
ARG numaddrs 10
ARG policy LRU
ARG clockbits 2
ARG cachesize 3
ARG maxpage 10
ARG seed 0
ARG notrace False

Solving...

Access: 8  MISS LRU ->          [8] <- MRU Replaced:- [Hits:0 Misses:1]
Access: 7  MISS LRU ->       [8, 7] <- MRU Replaced:- [Hits:0 Misses:2]
Access: 4  MISS LRU ->    [8, 7, 4] <- MRU Replaced:- [Hits:0 Misses:3]
Access: 2  MISS LRU ->    [7, 4, 2] <- MRU Replaced:8 [Hits:0 Misses:4]
Access: 5  MISS LRU ->    [4, 2, 5] <- MRU Replaced:7 [Hits:0 Misses:5]
Access: 4  HIT  LRU ->    [2, 5, 4] <- MRU Replaced:- [Hits:1 Misses:5]
Access: 7  MISS LRU ->    [5, 4, 7] <- MRU Replaced:2 [Hits:1 Misses:6]
Access: 3  MISS LRU ->    [4, 7, 3] <- MRU Replaced:5 [Hits:1 Misses:7]
Access: 4  HIT  LRU ->    [7, 3, 4] <- MRU Replaced:- [Hits:2 Misses:7]
Access: 5  MISS LRU ->    [3, 4, 5] <- MRU Replaced:7 [Hits:2 Misses:8]

FINALSTATS hits 2   misses 8   hitrate 20.00
```

arguments: -s 1 -n 10을 실행한 결과, hit rate는 20%를 기록하였다.

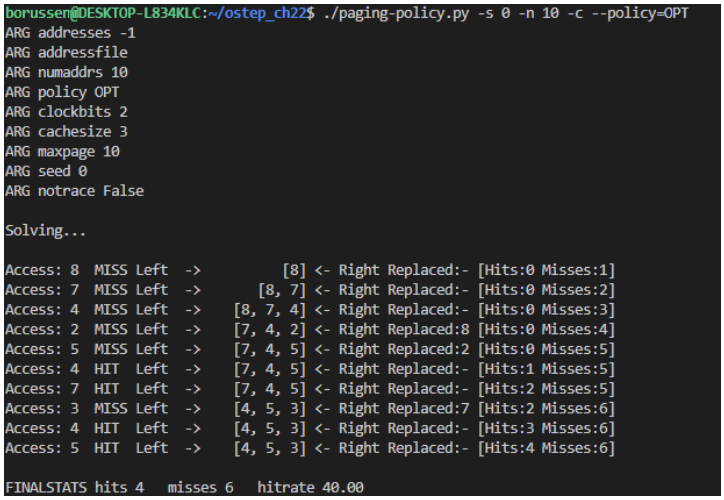
```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 1 -n 10 -c --policy=LRU
ARG addresses -1
ARG addressfile
ARG numaddrs 10
ARG policy LRU
ARG clockbits 2
ARG cachesize 3
ARG maxpage 10
ARG seed 1
ARG notrace False

Solving...

Access: 1  MISS LRU ->          [1] <- MRU Replaced:- [Hits:0 Misses:1]
Access: 8  MISS LRU ->       [1, 8] <- MRU Replaced:- [Hits:0 Misses:2]
Access: 7  MISS LRU ->    [1, 8, 7] <- MRU Replaced:- [Hits:0 Misses:3]
Access: 2  MISS LRU ->    [8, 7, 2] <- MRU Replaced:1 [Hits:0 Misses:4]
Access: 4  MISS LRU ->    [7, 2, 4] <- MRU Replaced:8 [Hits:0 Misses:5]
Access: 4  HIT  LRU ->    [7, 2, 4] <- MRU Replaced:- [Hits:1 Misses:5]
Access: 6  MISS LRU ->    [2, 4, 6] <- MRU Replaced:7 [Hits:1 Misses:6]
Access: 7  MISS LRU ->    [4, 6, 7] <- MRU Replaced:2 [Hits:1 Misses:7]
Access: 0  MISS LRU ->    [6, 7, 0] <- MRU Replaced:4 [Hits:1 Misses:8]
Access: 0  HIT  LRU ->    [6, 7, 0] <- MRU Replaced:- [Hits:2 Misses:8]

FINALSTATS hits 2   misses 8   hitrate 20.00
```

arguments: -s 2 -n 10을 실행한 결과, hit rate는 40%를 기록하였다.

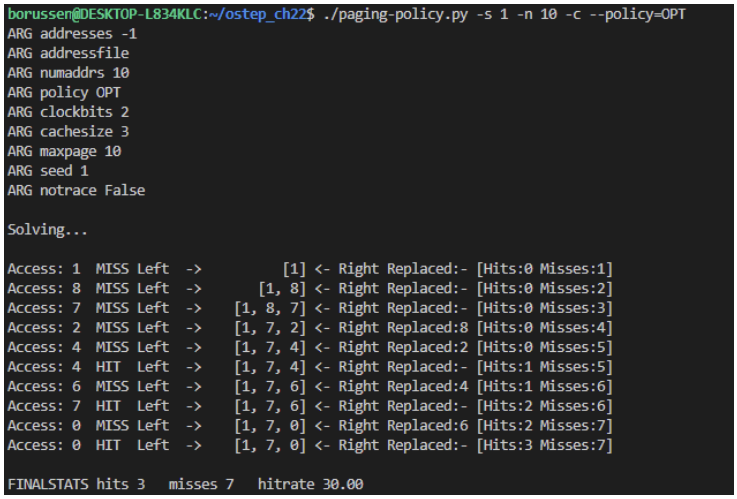
```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 2 -n 10 -c --policy=LRU
ARG addresses -1
ARG addressfile
ARG numaddrs 10
ARG policy LRU
ARG clockbits 2
ARG cachesize 3
ARG maxpage 10
ARG seed 2
ARG notrace False

Solving...

Access: 9  MISS LRU ->          [9] <- MRU Replaced:- [Hits:0 Misses:1]
Access: 9  HIT  LRU ->          [9] <- MRU Replaced:- [Hits:1 Misses:1]
Access: 0  MISS LRU ->       [9, 0] <- MRU Replaced:- [Hits:1 Misses:2]
Access: 0  HIT  LRU ->       [9, 0] <- MRU Replaced:- [Hits:2 Misses:2]
Access: 8  MISS LRU ->    [9, 0, 8] <- MRU Replaced:- [Hits:2 Misses:3]
Access: 7  MISS LRU ->    [0, 8, 7] <- MRU Replaced:9 [Hits:2 Misses:4]
Access: 6  MISS LRU ->    [8, 7, 6] <- MRU Replaced:0 [Hits:2 Misses:5]
Access: 3  MISS LRU ->    [7, 6, 3] <- MRU Replaced:8 [Hits:2 Misses:6]
Access: 6  HIT  LRU ->    [7, 3, 6] <- MRU Replaced:- [Hits:3 Misses:6]
Access: 6  HIT  LRU ->    [7, 3, 6] <- MRU Replaced:- [Hits:4 Misses:6]

FINALSTATS hits 4   misses 6   hitrate 40.00
```

(3) with OPT

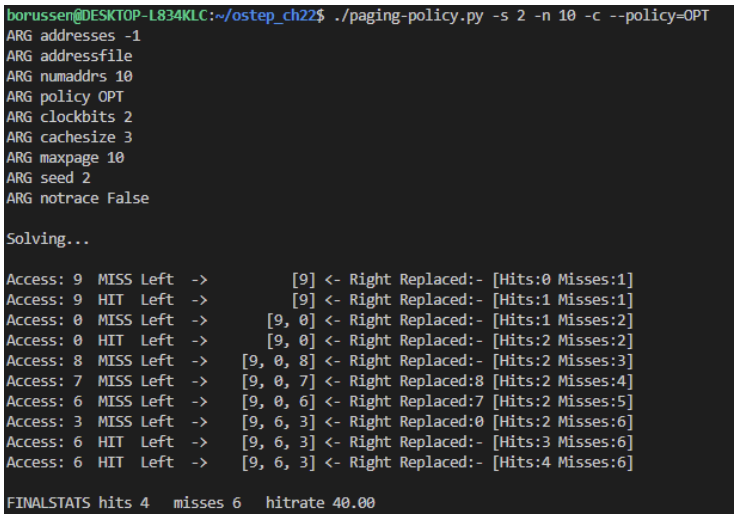
arguments: -s 0 -n 10을 실행한 결과, hit rate는 40%를 기록하였다.



arguments: -s 1 -n 10을 실행한 결과, hit rate는 30%를 기록하였다.



arguments: -s 2 -n 10을 실행한 결과, hit rate는 40%를 기록하였다.



2. For a cache of size 5, generate worst-case address reference streams for each of the following policies: FIFO, LRU, and MRU (worst-case reference streams cause the most misses possible. For the worst case reference streams, how much bigger of a cache is needed to improve performance dramatically and approach OPT?

(1) with FIFO

크기가 5인 캐시에 대해 address reference streams: [0, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 5, ...]를 받는 경우, FIFO는 Hit rate가 0%로 최악의 결과를 보인다.

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py --addresses=0,1,2,3,4,5,0,1,2,3,4,5 --policy=FIFO --cachesize=5 -c
ARG addresses 0,1,2,3,4,5,0,1,2,3,4,5
ARG addressfile
ARG numaddrs 10
ARG policy FIFO
ARG clockbits 2
ARG cachesize 5
ARG maxpage 10
ARG seed 0
ARG notrace False

Solving...

Access: 0  MISS FirstIn ->          [0] <- Lastin  Replaced:- [Hits:0 Misses:1]
Access: 1  MISS FirstIn ->       [0, 1] <- Lastin  Replaced:- [Hits:0 Misses:2]
Access: 2  MISS FirstIn ->    [0, 1, 2] <- Lastin  Replaced:- [Hits:0 Misses:3]
Access: 3  MISS FirstIn -> [0, 1, 2, 3] <- Lastin  Replaced:- [Hits:0 Misses:4]
Access: 4  MISS FirstIn -> [0, 1, 2, 3, 4] <- Lastin  Replaced:- [Hits:0 Misses:5]
Access: 5  MISS FirstIn -> [1, 2, 3, 4, 5] <- Lastin  Replaced:0 [Hits:0 Misses:6]
Access: 0  MISS FirstIn -> [2, 3, 4, 5, 0] <- Lastin  Replaced:1 [Hits:0 Misses:7]
Access: 1  MISS FirstIn -> [3, 4, 5, 0, 1] <- Lastin  Replaced:2 [Hits:0 Misses:8]
Access: 2  MISS FirstIn -> [4, 5, 0, 1, 2] <- Lastin  Replaced:3 [Hits:0 Misses:9]
Access: 3  MISS FirstIn -> [5, 0, 1, 2, 3] <- Lastin  Replaced:4 [Hits:0 Misses:10]
Access: 4  MISS FirstIn -> [0, 1, 2, 3, 4] <- Lastin  Replaced:5 [Hits:0 Misses:11]
Access: 5  MISS FirstIn -> [1, 2, 3, 4, 5] <- Lastin  Replaced:0 [Hits:0 Misses:12]

FINALSTATS hits 0   misses 12   hitrate 0.00
```

이때 캐시의 크기를 하나 더 늘려주면, 첫 입력 루프를 제외한 모든 경우에서 Cache hit가 일어나므로 성능이 크게 향상되고 OPT와 비슷한 수준의 Hit rate를 얻을 수 있다.

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py --addresses=0,1,2,3,4,5,0,1,2,3,4,5 --policy=FIFO --cachesize=6 -c
ARG addresses 0,1,2,3,4,5,0,1,2,3,4,5
ARG addressfile
ARG numaddrs 10
ARG policy FIFO
ARG clockbits 2
ARG cachesize 6
ARG maxpage 10
ARG seed 0
ARG notrace False

Solving...

Access: 0  MISS FirstIn ->          [0] <- Lastin  Replaced:- [Hits:0 Misses:1]
Access: 1  MISS FirstIn ->       [0, 1] <- Lastin  Replaced:- [Hits:0 Misses:2]
Access: 2  MISS FirstIn ->    [0, 1, 2] <- Lastin  Replaced:- [Hits:0 Misses:3]
Access: 3  MISS FirstIn -> [0, 1, 2, 3] <- Lastin  Replaced:- [Hits:0 Misses:4]
Access: 4  MISS FirstIn -> [0, 1, 2, 3, 4] <- Lastin  Replaced:- [Hits:0 Misses:5]
Access: 5  MISS FirstIn -> [0, 1, 2, 3, 4, 5] <- Lastin  Replaced:- [Hits:0 Misses:6]
Access: 0  HIT  FirstIn -> [0, 1, 2, 3, 4, 5] <- Lastin  Replaced:- [Hits:1 Misses:6]
Access: 1  HIT  FirstIn -> [0, 1, 2, 3, 4, 5] <- Lastin  Replaced:- [Hits:2 Misses:6]
Access: 2  HIT  FirstIn -> [0, 1, 2, 3, 4, 5] <- Lastin  Replaced:- [Hits:3 Misses:6]
Access: 3  HIT  FirstIn -> [0, 1, 2, 3, 4, 5] <- Lastin  Replaced:- [Hits:4 Misses:6]
Access: 4  HIT  FirstIn -> [0, 1, 2, 3, 4, 5] <- Lastin  Replaced:- [Hits:5 Misses:6]
Access: 5  HIT  FirstIn -> [0, 1, 2, 3, 4, 5] <- Lastin  Replaced:- [Hits:6 Misses:6]

FINALSTATS hits 6   misses 6   hitrate 50.00
```

(2) with LRU

크기가 5인 캐시에 대해 address reference streams: [0, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 5, ...]를 받는 경우, LRU는 Hit rate가 0%로 최악의 결과를 보인다.

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py --addresses=0,1,2,3,4,5,0,1,2,3,4,5 --policy=LRU --cachesize=5 -c
ARG addresses 0,1,2,3,4,5,0,1,2,3,4,5
ARG addressfile
ARG numaddrs 10
ARG policy LRU
ARG clockbits 2
ARG cachesize 5
ARG maxpage 10
ARG seed 0
ARG notrace False

Solving...

Access: 0  MISS LRU ->          [0] <- MRU Replaced:- [Hits:0 Misses:1]
Access: 1  MISS LRU ->       [0, 1] <- MRU Replaced:- [Hits:0 Misses:2]
Access: 2  MISS LRU ->    [0, 1, 2] <- MRU Replaced:- [Hits:0 Misses:3]
Access: 3  MISS LRU -> [0, 1, 2, 3] <- MRU Replaced:- [Hits:0 Misses:4]
Access: 4  MISS LRU -> [0, 1, 2, 3, 4] <- MRU Replaced:- [Hits:0 Misses:5]
Access: 5  MISS LRU -> [1, 2, 3, 4, 5] <- MRU Replaced:0 [Hits:0 Misses:6]
Access: 0  MISS LRU -> [2, 3, 4, 5, 0] <- MRU Replaced:1 [Hits:0 Misses:7]
Access: 1  MISS LRU -> [3, 4, 5, 0, 1] <- MRU Replaced:2 [Hits:0 Misses:8]
Access: 2  MISS LRU -> [4, 5, 0, 1, 2] <- MRU Replaced:3 [Hits:0 Misses:9]
Access: 3  MISS LRU -> [5, 0, 1, 2, 3] <- MRU Replaced:4 [Hits:0 Misses:10]
Access: 4  MISS LRU -> [0, 1, 2, 3, 4] <- MRU Replaced:5 [Hits:0 Misses:11]
Access: 5  MISS LRU -> [1, 2, 3, 4, 5] <- MRU Replaced:0 [Hits:0 Misses:12]

FINALSTATS hits 0   misses 12   hitrate 0.00
```

이때 캐시의 크기를 하나 더 늘려주면, 첫 입력 루프를 제외한 모든 경우에서 Cache hit가 일어나므로 성능이 크게 향상되고 OPT와 비슷한 수준의 Hit rate를 얻을 수 있다.

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py --addresses=0,1,2,3,4,5,0,1,2,3,4,5 --policy=LRU --cachesize=6 -c
ARG addresses 0,1,2,3,4,5,0,1,2,3,4,5
ARG addressfile
ARG numaddrs 10
ARG policy LRU
ARG clockbits 2
ARG cachesize 6
ARG maxpage 10
ARG seed 0
ARG notrace False

Solving...

Access: 0  MISS LRU ->          [0] <- MRU Replaced:- [Hits:0 Misses:1]
Access: 1  MISS LRU ->       [0, 1] <- MRU Replaced:- [Hits:0 Misses:2]
Access: 2  MISS LRU ->    [0, 1, 2] <- MRU Replaced:- [Hits:0 Misses:3]
Access: 3  MISS LRU -> [0, 1, 2, 3] <- MRU Replaced:- [Hits:0 Misses:4]
Access: 4  MISS LRU -> [0, 1, 2, 3, 4] <- MRU Replaced:- [Hits:0 Misses:5]
Access: 5  MISS LRU -> [0, 1, 2, 3, 4, 5] <- MRU Replaced:- [Hits:0 Misses:6]
Access: 0  HIT  LRU -> [1, 2, 3, 4, 5, 0] <- MRU Replaced:- [Hits:1 Misses:6]
Access: 1  HIT  LRU -> [2, 3, 4, 5, 0, 1] <- MRU Replaced:- [Hits:2 Misses:6]
Access: 2  HIT  LRU -> [3, 4, 5, 0, 1, 2] <- MRU Replaced:- [Hits:3 Misses:6]
Access: 3  HIT  LRU -> [4, 5, 0, 1, 2, 3] <- MRU Replaced:- [Hits:4 Misses:6]
Access: 4  HIT  LRU -> [5, 0, 1, 2, 3, 4] <- MRU Replaced:- [Hits:5 Misses:6]
Access: 5  HIT  LRU -> [0, 1, 2, 3, 4, 5] <- MRU Replaced:- [Hits:6 Misses:6]

FINALSTATS hits 6   misses 6   hitrate 50.00
```

(3) with MRU

크기가 5인 캐시에 대해 address reference streams: [0, 1, 2, 3, 4, 5, 4, 5, 4, 5, 4, 5, ...]를 받는 경우, MRU는 Hit rate가 0%로 최악의 결과를 보인다.

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py --addresses=0,1,2,3,4,5,4,5,4,5,4,5 --policy=MRU --cachesize=5 -c
ARG addresses 0,1,2,3,4,5,4,5,4,5,4,5
ARG addressfile
ARG numaddrs 10
ARG policy MRU
ARG clockbits 2
ARG cachesize 5
ARG maxpage 10
ARG seed 0
ARG notrace False

Solving...

Access: 0  MISS LRU ->          [0] <- MRU Replaced:- [Hits:0 Misses:1]
Access: 1  MISS LRU ->       [0, 1] <- MRU Replaced:- [Hits:0 Misses:2]
Access: 2  MISS LRU ->    [0, 1, 2] <- MRU Replaced:- [Hits:0 Misses:3]
Access: 3  MISS LRU -> [0, 1, 2, 3] <- MRU Replaced:- [Hits:0 Misses:4]
Access: 4  MISS LRU -> [0, 1, 2, 3, 4] <- MRU Replaced:- [Hits:0 Misses:5]
Access: 5  MISS LRU -> [0, 1, 2, 3, 5] <- MRU Replaced:4 [Hits:0 Misses:6]
Access: 4  MISS LRU -> [0, 1, 2, 3, 4] <- MRU Replaced:5 [Hits:0 Misses:7]
Access: 5  MISS LRU -> [0, 1, 2, 3, 5] <- MRU Replaced:4 [Hits:0 Misses:8]
Access: 4  MISS LRU -> [0, 1, 2, 3, 4] <- MRU Replaced:5 [Hits:0 Misses:9]
Access: 5  MISS LRU -> [0, 1, 2, 3, 5] <- MRU Replaced:4 [Hits:0 Misses:10]
Access: 4  MISS LRU -> [0, 1, 2, 3, 4] <- MRU Replaced:5 [Hits:0 Misses:11]
Access: 5  MISS LRU -> [0, 1, 2, 3, 5] <- MRU Replaced:4 [Hits:0 Misses:12]

FINALSTATS hits 0   misses 12   hitrate 0.00
```

이때 캐시의 크기를 하나 더 늘려주면, 첫 입력을 제외한 모든 경우에서 Cache hit가 일어나므로 성능이 크게 향상되고 OPT와 비슷한 수준의 Hit rate를 얻을 수 있다.

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py --addresses=0,1,2,3,4,5,4,5,4,5,4,5 --policy=MRU --cachesize=6 -c
ARG addresses 0,1,2,3,4,5,4,5,4,5,4,5
ARG addressfile
ARG numaddrs 10
ARG policy MRU
ARG clockbits 2
ARG cachesize 6
ARG maxpage 10
ARG seed 0
ARG notrace False

Solving...

Access: 0  MISS LRU ->          [0] <- MRU Replaced:- [Hits:0 Misses:1]
Access: 1  MISS LRU ->       [0, 1] <- MRU Replaced:- [Hits:0 Misses:2]
Access: 2  MISS LRU ->    [0, 1, 2] <- MRU Replaced:- [Hits:0 Misses:3]
Access: 3  MISS LRU -> [0, 1, 2, 3] <- MRU Replaced:- [Hits:0 Misses:4]
Access: 4  MISS LRU -> [0, 1, 2, 3, 4] <- MRU Replaced:- [Hits:0 Misses:5]
Access: 5  MISS LRU -> [0, 1, 2, 3, 4, 5] <- MRU Replaced:- [Hits:0 Misses:6]
Access: 4  HIT  LRU -> [0, 1, 2, 3, 5, 4] <- MRU Replaced:- [Hits:1 Misses:6]
Access: 5  HIT  LRU -> [0, 1, 2, 3, 4, 5] <- MRU Replaced:- [Hits:2 Misses:6]
Access: 4  HIT  LRU -> [0, 1, 2, 3, 5, 4] <- MRU Replaced:- [Hits:3 Misses:6]
Access: 5  HIT  LRU -> [0, 1, 2, 3, 4, 5] <- MRU Replaced:- [Hits:4 Misses:6]
Access: 4  HIT  LRU -> [0, 1, 2, 3, 5, 4] <- MRU Replaced:- [Hits:5 Misses:6]
Access: 5  HIT  LRU -> [0, 1, 2, 3, 4, 5] <- MRU Replaced:- [Hits:6 Misses:6]

FINALSTATS hits 6   misses 6   hitrate 50.00
```

3. Generate a random trace (use python or perl). How would you expect the different policies to perform on such a trace?

Random trace에는 이용할 수 있는 Locality가 없기 때문에 모든 policy에서 대체적으로 낮은 성능을 보인다. 그러므로 OPT를 제외한 policy 간의 성능은 비교하기 어렵다.

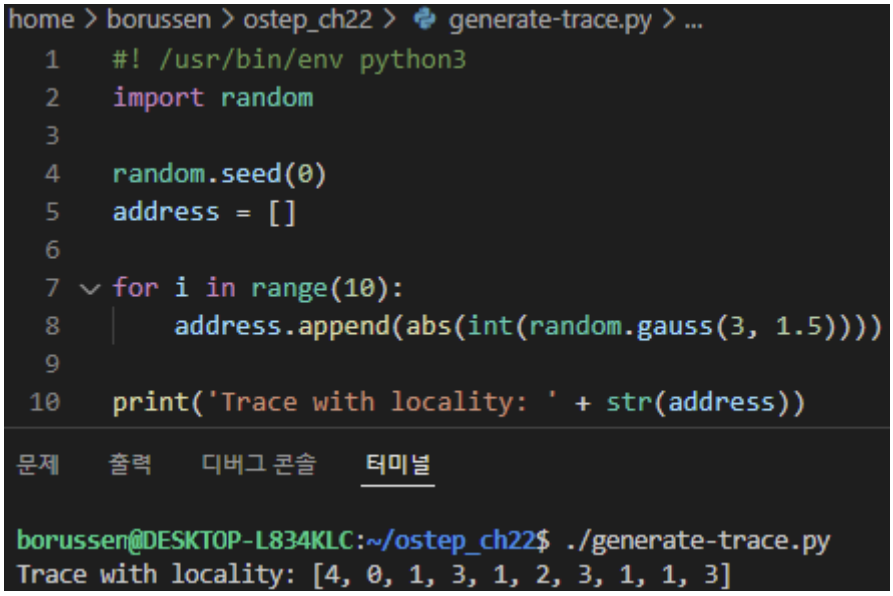
```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 0 -n 10 -c --policy=OPT
FINALSTATS hits 4   misses 6   hitrate 40.00

borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 0 -n 10 -c --policy=FIFO
FINALSTATS hits 1   misses 9   hitrate 10.00

borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 0 -n 10 -c --policy=LRU
FINALSTATS hits 2   misses 8   hitrate 20.00

borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 0 -n 10 -c --policy=MRU
FINALSTATS hits 2   misses 8   hitrate 20.00

borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 0 -n 10 -c --policy=RAND
FINALSTATS hits 0   misses 10   hitrate 0.00

borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 0 -n 10 -c --policy=UNOPT
FINALSTATS hits 0   misses 10   hitrate 0.00

borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 0 -n 10 -c --policy=CLOCK
FINALSTATS hits 1   misses 9   hitrate 10.00
```

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 1 -n 10 -c --policy=OPT
FINALSTATS hits 3   misses 7   hitrate 30.00

borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 1 -n 10 -c --policy=FIFO
FINALSTATS hits 2   misses 8   hitrate 20.00

borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 1 -n 10 -c --policy=LRU
FINALSTATS hits 2   misses 8   hitrate 20.00

borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 1 -n 10 -c --policy=MRU
FINALSTATS hits 2   misses 8   hitrate 20.00

borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 1 -n 10 -c --policy=RAND
FINALSTATS hits 2   misses 8   hitrate 20.00

borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 1 -n 10 -c --policy=UNOPT
FINALSTATS hits 2   misses 8   hitrate 20.00

borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 1 -n 10 -c --policy=CLOCK
FINALSTATS hits 2   misses 8   hitrate 20.00
```

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 2 -n 10 -c --policy=OPT
FINALSTATS hits 4   misses 6   hitrate 40.00

borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 2 -n 10 -c --policy=FIFO
FINALSTATS hits 4   misses 6   hitrate 40.00

borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 2 -n 10 -c --policy=LRU
FINALSTATS hits 4   misses 6   hitrate 40.00

borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 2 -n 10 -c --policy=MRU
FINALSTATS hits 3   misses 7   hitrate 30.00

borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 2 -n 10 -c --policy=RAND
FINALSTATS hits 4   misses 6   hitrate 40.00

borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 2 -n 10 -c --policy=UNOPT
FINALSTATS hits 3   misses 7   hitrate 30.00

borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -s 2 -n 10 -c --policy=CLOCK
FINALSTATS hits 4   misses 6   hitrate 40.00
```

4. Now generate a trace with some locality. How can you generate such a trace? How does LRU perform on it? How much better than RAND is LRU? How does CLOCK do? How about CLOCK with different numbers of clock bits?

*mu*를 평균으로, *sigma*를 표준 편차로 가지는 Gaussian Distribution (Normal distribution)을 통해 난수를 생성하는 random.gauss(*mu*, *sigma*)를 이용하여 trace를 생성하였다. 이렇게 생성된 trace: [4, 0, 1, 3, 1, 2, 3, 1, 1, 3]의 경우, 표준 편차를 비교적 작은 값으로 설정했기 때문에 최근에 나온 값이 높은 확률로 재등장하고 따라서 temporal locality를 가진다.



LRU는 temporal locality를 가지는 trace에 대해 다음과 같이 작동한다.

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py --addresses=4,0,1,3,1,2,3,1,1,3 --policy=LRU -c
Access: 4  MISS LRU ->          [4] <- MRU Replaced:- [Hits:0 Misses:1]
Access: 0  MISS LRU ->       [4, 0] <- MRU Replaced:- [Hits:0 Misses:2]
Access: 1  MISS LRU ->    [4, 0, 1] <- MRU Replaced:- [Hits:0 Misses:3]
Access: 3  MISS LRU ->    [0, 1, 3] <- MRU Replaced:4 [Hits:0 Misses:4]
Access: 1  HIT  LRU ->    [0, 3, 1] <- MRU Replaced:- [Hits:1 Misses:4]
Access: 2  MISS LRU ->    [3, 1, 2] <- MRU Replaced:0 [Hits:1 Misses:5]
Access: 3  HIT  LRU ->    [1, 2, 3] <- MRU Replaced:- [Hits:2 Misses:5]
Access: 1  HIT  LRU ->    [2, 3, 1] <- MRU Replaced:- [Hits:3 Misses:5]
Access: 1  HIT  LRU ->    [2, 3, 1] <- MRU Replaced:- [Hits:4 Misses:5]
Access: 3  HIT  LRU ->    [2, 1, 3] <- MRU Replaced:- [Hits:5 Misses:5]

FINALSTATS hits 5   misses 5   hitrate 50.00
```

Trace: [4, 0, 1, 3, 1, 2, 3, 1, 1, 3]에 대해 LRU는 50%의 hit rate을 보이고, RAND는 30%의 hit rate을 보이므로 LRU가 RAND에 비해서 더 좋은 성능을 가진다.

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py --addresses=4,0,1,3,1,2,3,1,1,3 --policy=LRU -c
FINALSTATS hits 5   misses 5   hitrate 50.00

borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py --addresses=4,0,1,3,1,2,3,1,1,3 --policy=RAND -c
FINALSTATS hits 3   misses 7   hitrate 30.00
```

Locality를 가지는 trace에 대해서 서로 다른 Clock bits 수(0, 1, 2, 3)를 가지는 CLOCK은 작동은 다음과 같이 일어난다. 해당 경우에서, CLOCK은 Clock bits가 많을수록 높은 hit rate를 보인다.

(1) The number of clock bits = 0

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py --addresses=4,0,1,1,1,2,3,1,1,3 --policy=CLOCK -c -b 0
Access: 4  MISS Left  ->         [4] <- Right Replaced:- [Hits:0 Misses:1]
Access: 0  MISS Left  ->      [4, 0] <- Right Replaced:- [Hits:0 Misses:2]
Access: 1  MISS Left  ->   [4, 0, 1] <- Right Replaced:- [Hits:0 Misses:3]
Access: 1  HIT  Left  ->   [4, 0, 1] <- Right Replaced:- [Hits:1 Misses:3]
Access: 1  HIT  Left  ->   [4, 0, 1] <- Right Replaced:- [Hits:2 Misses:3]
Access: 2  MISS Left  ->   [4, 0, 2] <- Right Replaced:1 [Hits:2 Misses:4]
Access: 3  MISS Left  ->   [4, 2, 3] <- Right Replaced:0 [Hits:2 Misses:5]
Access: 1  MISS Left  ->   [4, 3, 1] <- Right Replaced:2 [Hits:2 Misses:6]
Access: 1  HIT  Left  ->   [4, 3, 1] <- Right Replaced:- [Hits:3 Misses:6]
Access: 3  HIT  Left  ->   [4, 3, 1] <- Right Replaced:- [Hits:4 Misses:6]

FINALSTATS hits 4   misses 6   hitrate 40.00
```

(2) The number of clock bits = 1

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py --addresses=4,0,1,1,1,2,3,1,1,3 --policy=CLOCK -c -b 1
Access: 4  MISS Left  ->         [4] <- Right Replaced:- [Hits:0 Misses:1]
Access: 0  MISS Left  ->      [4, 0] <- Right Replaced:- [Hits:0 Misses:2]
Access: 1  MISS Left  ->   [4, 0, 1] <- Right Replaced:- [Hits:0 Misses:3]
Access: 1  HIT  Left  ->   [4, 0, 1] <- Right Replaced:- [Hits:1 Misses:3]
Access: 1  HIT  Left  ->   [4, 0, 1] <- Right Replaced:- [Hits:2 Misses:3]
Access: 2  MISS Left  ->   [4, 0, 2] <- Right Replaced:1 [Hits:2 Misses:4]
Access: 3  MISS Left  ->   [4, 2, 3] <- Right Replaced:0 [Hits:2 Misses:5]
Access: 1  MISS Left  ->   [2, 3, 1] <- Right Replaced:4 [Hits:2 Misses:6]
Access: 1  HIT  Left  ->   [2, 3, 1] <- Right Replaced:- [Hits:3 Misses:6]
Access: 3  HIT  Left  ->   [2, 3, 1] <- Right Replaced:- [Hits:4 Misses:6]

FINALSTATS hits 4   misses 6   hitrate 40.00
```

(3) The number of clock bits = 2

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py --addresses=4,0,1,1,1,2,3,1,1,3 --policy=CLOCK -c -b 2
Access: 4  MISS Left  ->         [4] <- Right Replaced:- [Hits:0 Misses:1]
Access: 0  MISS Left  ->      [4, 0] <- Right Replaced:- [Hits:0 Misses:2]
Access: 1  MISS Left  ->   [4, 0, 1] <- Right Replaced:- [Hits:0 Misses:3]
Access: 1  HIT  Left  ->   [4, 0, 1] <- Right Replaced:- [Hits:1 Misses:3]
Access: 1  HIT  Left  ->   [4, 0, 1] <- Right Replaced:- [Hits:2 Misses:3]
Access: 2  MISS Left  ->   [4, 1, 2] <- Right Replaced:0 [Hits:2 Misses:4]
Access: 3  MISS Left  ->   [4, 2, 3] <- Right Replaced:1 [Hits:2 Misses:5]
Access: 1  MISS Left  ->   [2, 3, 1] <- Right Replaced:4 [Hits:2 Misses:6]
Access: 1  HIT  Left  ->   [2, 3, 1] <- Right Replaced:- [Hits:3 Misses:6]
Access: 3  HIT  Left  ->   [2, 3, 1] <- Right Replaced:- [Hits:4 Misses:6]

FINALSTATS hits 4   misses 6   hitrate 40.00
```

(4) The number of clock bits = 3

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py --addresses=4,0,1,1,1,2,3,1,1,3 --policy=CLOCK -c -b 3
Access: 4  MISS Left  ->         [4] <- Right Replaced:- [Hits:0 Misses:1]
Access: 0  MISS Left  ->      [4, 0] <- Right Replaced:- [Hits:0 Misses:2]
Access: 1  MISS Left  ->   [4, 0, 1] <- Right Replaced:- [Hits:0 Misses:3]
Access: 1  HIT  Left  ->   [4, 0, 1] <- Right Replaced:- [Hits:1 Misses:3]
Access: 1  HIT  Left  ->   [4, 0, 1] <- Right Replaced:- [Hits:2 Misses:3]
Access: 2  MISS Left  ->   [4, 1, 2] <- Right Replaced:0 [Hits:2 Misses:4]
Access: 3  MISS Left  ->   [1, 2, 3] <- Right Replaced:4 [Hits:2 Misses:5]
Access: 1  HIT  Left  ->   [1, 2, 3] <- Right Replaced:- [Hits:3 Misses:5]
Access: 1  HIT  Left  ->   [1, 2, 3] <- Right Replaced:- [Hits:4 Misses:5]
Access: 3  HIT  Left  ->   [1, 2, 3] <- Right Replaced:- [Hits:5 Misses:5]

FINALSTATS hits 5   misses 5   hitrate 50.00
```

5. Use a program like valgrind **to instrument a real application and generate a virtual page reference stream. For example, running** valgrind --tool=lackey --trace-mem=yes ls **will output a nearly-complete reference trace of every instruction and data reference made by the program** ls**. To make this useful for the simulator above, you'll have to first transform each virtual memory reference into a virtual page-number reference (done by masking off the offset and shifting the resulting bits downward). How big of a cache is needed for your application trace in order to satisfy a large fraction of requests? Plot a graph of its working set as the size of the cache increases.**

'valgrind --tool=lackey --trace-mem=yes ls' 명령어를 이용하여 프로그램 ls에 의해 만들어진 모든 명령 및 데이터 참조에 대한 reference trace를 찾았고 이를 'trace.txt' 파일에 저장했다.

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ valgrind --tool=lackey --trace-mem=yes ls &> trace.txt
```

```
==7155== Lackey, an example Valgrind tool
==7155== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote.
==7155== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==7155== Command: ls
==7155==
I  04001100,3
I  04001103,5
 S 1ffefffdf8,8
I  04001df0,4
I  04001df4,1
 S 1ffefffdf0,8
I  04001df5,3
I  04001df8,2
 S 1ffefffde8,8
```

Virtual memory reference trace를 virtual page-number reference trace로 변환시키기 위해서 offset을 masking하고 결과값에 대해 비트 시프트 연산을 하는 프로그램 transform.py를 작성했다.

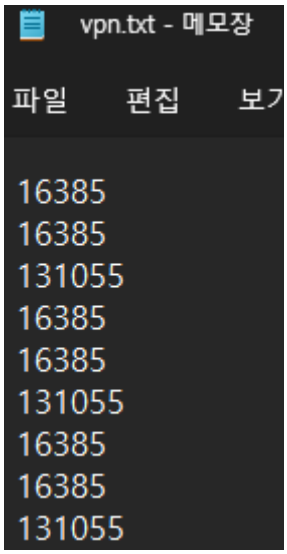
home > borussen > ostep_ch22 > transform.py > ...

```
1   #! /usr/bin/env python3
2
3   traceFile = open('./trace.txt', 'r')
4   vpnFile = open('./vpn.txt', 'w')
5
6   for line in traceFile:
7       if not line.startswith("="):
8           vpnFile.write(str((int("0x" + line[3:11], 16) & 0xFFFFF000) >> 12) + "\n")
9
10  traceFile.close()
11  vpnFile.close()
```

문제 출력 디버그 콘솔 터미널

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./transform.py
```

프로그램 실행 결과, 다음과 같이 virtual-page number reference trace가 생성된 것을 확인하였다.



생성된 virtual-page number reference trace 중에서 대략 만 개정도의 연속된 trace를 골라 각 policy별로 cache의 크기에 따라 hit rate가 어떻게 달라지는지 알아보았다.

(1) with FIFO

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "FIFO" -C 1
FINALSTATS hits 6404   misses 3610   hitrate 63.95
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "FIFO" -C 2
FINALSTATS hits 8879   misses 1135   hitrate 88.67
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "FIFO" -C 3
FINALSTATS hits 9407   misses 607   hitrate 93.94
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "FIFO" -C 4
FINALSTATS hits 9908   misses 106   hitrate 98.94
```

(2) with LRU

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "LRU" -C 1
FINALSTATS hits 6404   misses 3610   hitrate 63.95
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "LRU" -C 2
FINALSTATS hits 9252   misses 762   hitrate 92.39
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "LRU" -C 3
FINALSTATS hits 9559   misses 455   hitrate 95.46
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "LRU" -C 4
FINALSTATS hits 9945   misses 69   hitrate 99.31
```

(3) with MRU

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "MRU" -C 1
FINALSTATS hits 6404   misses 3610   hitrate 63.95
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "MRU" -C 2
FINALSTATS hits 6420   misses 3594   hitrate 64.11
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "MRU" -C 3
FINALSTATS hits 8187   misses 1827   hitrate 81.76
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "MRU" -C 4
FINALSTATS hits 8326   misses 1688   hitrate 83.14
```

(4) with OPT

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "OPT" -C 1
FINALSTATS hits 6404   misses 3610   hitrate 63.95
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "OPT" -C 2
FINALSTATS hits 9254   misses 760   hitrate 92.41
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "OPT" -C 3
FINALSTATS hits 9756   misses 258   hitrate 97.42
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "OPT" -C 4
FINALSTATS hits 9967   misses 47   hitrate 99.53
```

(5) with RAND

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "RAND" -C 1
FINALSTATS hits 6404   misses 3610   hitrate 63.95
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "RAND" -C 2
FINALSTATS hits 8836   misses 1178   hitrate 88.24
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "RAND" -C 3
FINALSTATS hits 9514   misses 500   hitrate 95.01
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "RAND" -C 4
FINALSTATS hits 9860   misses 154   hitrate 98.46
```

(6) with UNOPT

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "UNOPT" -C 1
FINALSTATS hits 6404   misses 3610   hitrate 63.95
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "UNOPT" -C 2
FINALSTATS hits 6417   misses 3597   hitrate 64.08
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "UNOPT" -C 3
FINALSTATS hits 6420   misses 3594   hitrate 64.11
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "UNOPT" -C 4
FINALSTATS hits 6433   misses 3581   hitrate 64.24
```

(7) with CLOCK

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "CLOCK" -C 1
FINALSTATS hits 6404   misses 3610   hitrate 63.95
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "CLOCK" -C 2
FINALSTATS hits 9008   misses 1006   hitrate 89.95
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "CLOCK" -C 3
FINALSTATS hits 9593   misses 421   hitrate 95.80
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./paging-policy.py -c -f ./vpn.txt -p "CLOCK" -C 4
FINALSTATS hits 9906   misses 108   hitrate 98.92
```

이러한 결과들을 그래프로 표현하는 프로그램 plot.py를 작성하였다.

home > borussen > ostep_ch22 > plot.py > ...

```python
#! /usr/bin/env python3
import os
import matplotlib.pyplot as plt
import numpy as np

cache_sizes = np.arange(1, 5)
policies = ["FIFO", "LRU", "MRU", "OPT", "RAND", "UNOPT", "CLOCK"]

hit_rates = [
    # FIFO
    [63.95, 88.67, 93.94, 98.94],
    # LRU
    [63.95, 92.39, 95.46, 99.31],
    # MRU
    [63.95, 64.11, 81.76, 83.14],
    # OPT
    [63.95, 92.41, 97.42, 99.53],
    # RAND
    [63.95, 88.24, 95.01, 98.46],
    # UNOPT
    [63.95, 64.08, 64.11, 64.24],
    # CLOCK
    [63.95, 89.95, 95.80, 98.92],
]

for i in range(len(policies)):
    plt.plot(cache_sizes, hit_rates[i])

plt.legend(policies)
plt.margins(0)
plt.xticks(cache_sizes, cache_sizes)
plt.xlabel("Cache Size (Blocks)")
plt.ylabel("Hit Rate (%)")
plt.savefig("result.png", dpi=227)
```

문제 출력 디버그 콘솔 터미널

```
borussen@DESKTOP-L834KLC:~/ostep_ch22$ ./plot.py
```

plot.py를 실행하여 Cache size에 따른 Hit rate 관계 그래프를 얻었다.