FinalReport

2019년 12월 7일

# 1 2019 컴퓨터구조 팀 프로젝트 과제 2 보고서

**캐시 교체 알고리즘 구현 및 성능 비교 분석**

**2조** > 201811224 조혜진
201714152 박종현
201512265 박인우
201810754 조윤직

제출일 : 2019. 12.

목차

### 1.1 문제 정의

수업 시간에 배운 캐시 교체 알고리즘의 원리를 이용하여 캐시 교체 알고리즘을 구현한다. 이 때 새로 제안한 기법이 존재할 경우, 그 기법을 사용한 알고리즘도 추가한다. 또한 적합하다고 생각하는 데이터의 개수와 패턴, 슬롯의 개수를 기준으로 삼아 각 알고리즘의 성능을 비교/분석한다.

### 1.2 주요 함수 및 변수

**함수 공용 Parameter 설명** - inputdata : 캐쉬안에 넣을 데이터 list - slotcount : 슬롯 카운트

**LRU**

**알고리즘 설명** 사용자에게 캐시에 들어갈 데이터를 정수의 배열로서 받고, 슬롯의 개수를 정수로 받는다. 그리고 해당 데이터를 반복문을 돌면서 캐시에 넣어준다. 조건문을 통해

- **1) 데이터가 이미 캐시에 있을 경우**
- **2) 슬롯의 개수보다 채워진 데이터의 개수가 아직 적은 경우**
- **3) 데이터가 이미 캐시에 있지도 않고, 슬롯의 개수보다 데이터의 개수가 아직 적지도 않은 경우**
- '1)'의 경우, 데이터가 캐시에 적중되었다는 의미이므로, 적중 메시지를 출력하고 현재 데이터의 카운트값을 -1로 저장한다. 그리고 현재 캐시에 존재하는 데이터의 모든 카운터값을 1씩 증가한다. (따라서 방금 저장한 값은 카운트값이 0이되고, 나머지의 카운트값은 1씩 증가하게 된다.)
- '2)'의 경우 값이 적중하지 않은 이상 그냥 빈 슬롯을 채우기만 하면 되므로, 그냥 데이터를 저장하고 카운트값을 '1)'의 경우와 마찬가지로 조정해준다.
- '3)'의 경우 현재 데이터중 카운트가 가장 큰 값을 찾아야하므로 해당값을 찾아주고, 해당하는 데이터를 삭제하고 새 데이터를 넣는다. 그리고 카운트값을 조정해준다.

**변수**

- n : 들어온 데이터의 길이로서, len(inputdata)로 계산된다. 이는 추후 반복문을 돌릴 때 활용된다.
- hit : 적중할때마다 하나씩 늘린다. 최종적인 적중률 계산에 활용된다.
- datas : 캐시에 들어간 데이터가 들어갈 list
- counts : 캐시에 들어간 데이터의 count값을 가질 list
- next_data : 이번 차례에 들어갈 데이터값
- max_count : 크기가 최대인 값
- max_count_index : 크기가 최대한 값의 인덱스 넘버

**함수**

- len(inputdata) : inputdata 배열의 길이값 리턴
- datas.append(next_data) : 현재 데이터값 캐시에 추가
- list(zip(datas,counts))) : datas와 counts의 값들을 묶어 리스트화

### 구현 코드

```
def LRU_implement(inputdata,slotcount):

    print("\nLRU : ")

    n = len(inputdata)
    hit = 0
    datas = []
    counts = []

    for i in range(n):
      next_data = inputdata[i]

      if next_data in datas:
        hit += 1
        counts[datas.index(next_data)] = -1
        for index in range(len(counts)):
          counts[index] += 1
        print("Round #{0} -> [{1}] 현재 캐쉬 상태 : {2}, HIT!!".format(i + 1, next_data, list(zip(d
        continue

      if len(datas) < slotcount:
        datas.append(next_data)
        counts.append(-1)
        for index in range(len(counts)):
          counts[index] += 1

      else:
        max_count = 0
        max_count_index = 0
        for index in range(len(counts)):
          if counts[index] > max_count:
            max_count = counts[index]
            max_count_index = index
        datas[max_count_index] = next_data
```

```
            counts[max_count_index] = -1
            for index in range(len(counts)):
                counts[index] += 1

        print("Round #{0} -> [{1}] 현재 캐쉬 상태 : {2}".format(i + 1, next_data, list(zip(datas,coun

    print("H = {} / {} = {}".format(hit, n, hit/n))
    return hit / n
```

### FIFO

### 알고리즘 설명

1. 캐시 슬롯은 파이썬 리스트로 구성한다.
2. inputdata를 순차적으로 읽는다.
3. 다음에 사용할 값이 캐시슬롯 안에 있는지 판단 캐시 적중, 미스를 구별
4. 캐시 미스가 일어났을 때
    1. 캐시 슬롯이 가득 차있지 않으면 캐시 슬롯 맨 뒤에 추가
    2. 캐시 슬롯이 가득 차있으면 순환 큐를 이용하여 가장 오래된 값이 담긴 슬롯에 적재

### 변수

- inputdata : 입력 데이터 리스트, slotcount :캐시 슬롯수
- n : 배열로 주어진 inputdata의 길이
- hit : 캐시 적중 횟수
- a: 캐시 슬롯 (리스트)
- idx 순환 큐를 위한 변수

### 구현 코드

```
def FIFO_implement(inputdata,slotcount):
    print("\nFIFO : ")
    # n 입력받는 데이터의 갯수
    n = len(inputdata)
    hit = 0  # hit 갯수
    a = [] # 현재 캐쉬 상태 리스트
    idx = 0 # 순환 큐 형태를 위한 변수- 가장 오래된 캐쉬 슬롯 지정
```

```
    for i in range(n): # 데이터의 갯수 만큼 입력 수행
        next_data = inputdata[i]

        if next_data in a: # 캐시 안에 있을 시에는
            hit += 1
            print("Round #{0} -> [{1}] 현재 캐쉬 상태 : {2}, HIT!!".format(i + 1, next_data, a))
            continue
        # 캐시 안에 없을 때
        if len(a) < slotcount: # 캐시 슬롯의 빈자리 존재
            a.append(next_data) # 캐시 삽입

        else: # 빈 자리가 없을 때
            a[idx % slotcount] = next_data # 가장 오래된 캐시슬롯에 캐시 삽입
            idx += 1

        #현재 상태 출력
        print("Round #{0} -> [{1}] 현재 캐쉬 상태 : {2}".format(i + 1, next_data, a))

    print("H = {} / {} = {}".format(hit, n, hit / n))
    return hit / n
```

**LFU**

**알고리즘 설명** 1. 캐시 슬롯은 최소 힙으로 구성한다. 2. inputdata를 순차적으로 읽는다. 3. 다음에 검사할 데이터(inputdata[i])가 슬롯에 있는지 판단하여 캐시 적중과 미스를 결정한다. 4. 캐시 미스가 일어났을 때, 현재 슬롯이 가득 차있지 않으면 현재 읽어들인 inputdata를 힙(슬롯)에 push한다. 이 때 count는 1로 초기화한다.
5. 캐시 적중이 일어났을 때, 해당 슬롯의 count를 1 증가시킨 후 다음 데이터를 검사한다.
6. 캐시 미스가 일어났을 때, 현재 슬롯이 다 차있다면 root를 pop한 후 현재 읽어들인 inputdata를 힙(슬롯)에 push한다. 이 때 count는 1로 초기화한다.

**변수** - n : 배열로 주어진 inputdata의 길이 - hit : 캐시 적중 횟수 - datas : 캐시 슬롯 (힙) - next_data : 현재 검사할 데이터 - datas을 이루는 튜플 (원소1, 원소2, 원소3) : (빈도 카운트, uniqueNum, 실제 데이터 값) - uniqueNum : heapq 모듈이 두번 이상의 value도 같이 정렬시켜버리므로 그것을 방지하기 위해 넣음 - current : next_data를 튜플의 세 번재 값(데이터값)으로 갖는 특정 튜플(슬롯)의 0번째 값 - oldIdx : current의 인덱스

**함수** - len(array) : 인자로 들어간 array의 길이 반환 - idx(array[i]) : 인자로 들어간 array[i]의 인덱스

반환 - heapq._siftup(힙, 인덱스) : 힙을 갱신(해당 인덱스의 원소를 밑으로 내림) - heapq.heappop(힙) : 힙의 root를 pop - heapq.heappush(힙, 추가할 데이터) : 힙에 새로운 데이터를 push

**구현 코드**

```
import heapq

def LFU_implement(inputdata,slotcount):
    print("\nLFU : ")

    n = len(inputdata)
    hit = 0
    datas = []  # 슬롯 역할을 할 힙

    for i in range(n):

        next_data = inputdata[i]
        uniqueNum = i   # uniqueNum : heapq 모듈이 두번 이상의 value도 같이 정렬시켜버리므로 그것을 방
        if next_data in [item[2] for item in datas]:  # 캐시 적중
            current = [item for item in datas if item[2] == next_data][0]   # next_data를 튜플의
            oldIdx = datas.index(current)
            datas[oldIdx]=(current[0]+1, current[1], current[2])    # 해당 슬롯의 카운터를 1 증가
            heapq._siftup(datas, oldIdx)    # 힙 갱신
            hit += 1
            print("Round#{} -> [{}] 현재 캐쉬 상태 : {}, HIT!!".format(i + 1, next_data, [(item[2]
        else: # 캐시 미스
            if len(datas) < slotcount:  # 슬롯이 다 차지 않았을 때
                heapq.heappush(datas, (1, uniqueNum, next_data))   # 슬롯에 데이터 추가. 카운터는 1
            else:   # 슬롯이 다 찼을 때
                heapq.heappop(datas)
                heapq.heappush(datas, (1, uniqueNum, next_data))   # 슬롯에 데이터 교체. 카운터는 1
            print("Round#{} -> [{}] 현재 캐쉬 상태 : {}".format(i + 1, next_data, [(item[2], item[

    print("H = {} / {} = {}".format(hit, n, hit/n))
    return hit / n
```

### 임의 교체 방식

**알고리즘 설명** 1. inputdata를 순차적으로 읽는다. 2. 만약 다음에 검사할 데이터(inputdata[i])가 슬롯에 있으면 HIT으로 간주하고 다음 데이터를 검사한다.
아니라면 현재 슬롯리스트의 길이를 검사하며 만약 슬롯 개수보다 작으면 슬롯리스트에 추가하고
그 조건도 만족하지 않으면 0 ~ 슬롯리스트-1 만큼 난수를 발생시켜 교체한다

```
def Random_implement(inputdata,slotcount):
    print("\nRandom : ")

    n = len(inputdata)
    hit = 0
    a = []

    for i in range(n):
        next_data = inputdata[i]

        if next_data in a:
            hit += 1
            print("Round #{0} -> [{1}] 현재 캐쉬 상태 : {2}, HIT!!".format(i+1, next_data, a))
            continue

        if len(a) < slotcount:
            a.append(next_data)

        else:
            random_number = randint(0, slotcount-1)
            a[random_number] = next_data

        print("Round #{0} -> [{1}] 현재 캐쉬 상태 : {2}".format(i+1, next_data, a))

    print("H = {} / {} = {}".format(hit, n, hit/n))
    return hit / n
```

### 새로 제안하는 교체 방식

**알고리즘 설명** 기본 구조는 원래의 LRU와 유사하나, 3) **데이터가 이미 캐시에 있지도 않고, 슬롯의 개수보다 데이터의 개수가 아직 적지도 않은 경우**, 가중치를 계산하여 캐시를 교체하도록 한다. 즉,

무조건 count값이 높은 값을 교체대상으로 삼는 것이 아니라, 가중치가 높은 캐시가 교체될 확률이 높도록 설정하는 것이다. random.choices(datas, weights=counts, k=1)[0] 를 활용하여, 가중치에 따라 값을 값을 뽑아내어 활용한다. 이 후 진행과정은 기존 LRU와 같다.

기존 LRU와 달리, 오래된 값이라고 무조건 삭제되는 것이 아니므로, 조금의 개선을 기대해볼 수 있다.

### 변수

- choice_data = 가중치에 따라 랜덤하게 선택된 값

### 함수

- random.choices(datas, weights=counts, k=1) : data함수에서 각 인덱스의 가중치를 counts만큼 주어 랜덤하게 하나를 뽑아낸다. 그 다음 그 성분으로 리스트를 만들어 리턴한다.

### 구현 코드

```
import random


def NEW_implement(inputdata,slotcount):
    print("\nLRU : ")

    n = len(inputdata)
    hit = 0
    datas = []
    counts = []

    for i in range(n):
        next_data = inputdata[i]

        if next_data in datas:
            hit += 1
            counts[datas.index(next_data)] = -1
            for index in range(len(counts)):
                counts[index] += 1
            print("Round #{0} -> [{1}] 현재 캐쉬 상태 : {2}, HIT!!".format(i + 1, next_data, list(
            continue
```

```
        if len(datas) < slotcount:
            datas.append(next_data)
            counts.append(-1)
            for index in range(len(counts)):
                counts[index] += 1

        else:
            choice_data = random.choices(datas, weights=counts, k=1)[0]
            datas[datas.index(choice_data)] = next_data
            counts[datas.index(next_data)] = -1
            for index in range(len(counts)):
                counts[index] += 1

        print("Round #{0} -> [{1}] 현재 캐쉬 상태 : {2}".format(i + 1, next_data, list(zip(datas,c

    print("H = {} / {} = {}".format(hit, n, hit/n))
    return hit / n
```

### 1.3 교체 알고리즘 별 성능 비교 분석

다양한 입력 형태에 따른, 교체 알고리즘의 성능 분석 결과를 도표로 작성

**비교 기준** > - 가능한 데이터 개수 > - 입력 패턴 > - 슬롯 개수

을 기준으로 모든 case에 대한 Hit Ratio & 실행시간의 테이블 및 그래프로 비교

**Intro**

- **필요한 라이브러리**

```
In [7]: import LRU, FIFO, LFU, RND, NEW
        from random import *
        import matplotlib.pyplot as plt
        import matplotlib.gridspec as gridspec
        import numpy as np
        import pandas as pd
        from datetime import datetime
```

- **적중률 데이터 가져오는 함수 + 수행시간 가져오는 함수**
  - **Parameter 설명** > - input_size : 캐쉬에 넣을 데이터 개수 > - input_max_value : 입력 패턴 0 ~ (input_max_value-1)사이의 난수 > - slot_size : 슬롯 개수

– **함수 설명** > - input_size, input_max_value, slot_size가 주어지면 모든 경우의 수를 10번씩 교체 알고리즘들의
Hit Ratio 및 실행 시간을 평균을 내어 각 리스트에 저장

```
In [8]: def get_hit_result(input_size, input_max_size, slot_max_size):
            LRU_list = []
            LFU_list = []
            RND_list = []
            FIFO_list = []
            NEW_list = []

            for slot_size in range(3, slot_max_size):

                tmp_LRU = 0
                tmp_LFU = 0
                tmp_RND = 0
                tmp_FIFO = 0
                tmp_NEW = 0

                for i in range(10):

                    input_list = []

                    for j in range(input_size):
                        input_list.append(randint(0, input_max_size))

                    tmp_LRU += LRU.LRU_implement(input_list, slot_size)
                    tmp_LFU += LFU.LFU_implement(input_list, slot_size)
                    tmp_RND += RND.Random_implement(input_list, slot_size)
                    tmp_FIFO += FIFO.FIFO_implement(input_list, slot_size)
                    tmp_NEW += NEW.NEW_implement(input_list, slot_size)

                LRU_list.append(tmp_LRU / 10)
                LFU_list.append(tmp_LFU / 10)
                RND_list.append(tmp_RND / 10)
                FIFO_list.append(tmp_FIFO / 10)
```

```
            NEW_list.append(tmp_NEW / 10)
        #index = [i for i in range(3,slot_max_size)]
        #index = np.array(index)

        return LRU_list, LFU_list, RND_list, FIFO_list, NEW_list
```

```
In [10]: def get_time_result(input_size, input_max_size, slot_max_size):
             LRU_list = []
             LFU_list = []
             RND_list = []
             FIFO_list = []
             NEW_list = []

             for slot_size in range(3, slot_max_size):

                 time_LRU = 0
                 time_LFU = 0
                 time_RND = 0
                 time_FIFO = 0
                 time_NEW = 0

                 for i in range(10):

                     input_list = []

                     for j in range(input_size):
                         input_list.append(randint(0, input_max_size))

                     starttime = datetime.now()
                     LRU.LRU_implement(input_list, slot_size)
                     tmpTime_LRU = (datetime.now()-starttime).microseconds

                     starttime = datetime.now()
                     LFU.LFU_implement(input_list, slot_size)
                     tmpTime_LFU = (datetime.now()-starttime).microseconds
```

```
            starttime = datetime.now()
            RND.Random_implement(input_list, slot_size)
            tmpTime_FIFO = (datetime.now()-starttime).microseconds

            starttime = datetime.now()
            FIFO.FIFO_implement(input_list, slot_size)
            tmpTime_RND = (datetime.now()-starttime).microseconds

            starttime = datetime.now()
            NEW.NEW_implement(input_list, slot_size)
            tmpTime_NEW = (datetime.now()-starttime).microseconds

            time_LRU += tmpTime_LRU
            time_LFU += tmpTime_LFU
            time_RND += tmpTime_RND
            time_FIFO += tmpTime_FIFO
            time_NEW += tmpTime_NEW

        LRU_list.append(time_LRU / 10)
        LFU_list.append(time_LFU / 10)
        RND_list.append(time_RND / 10)
        FIFO_list.append(time_FIFO / 10)
        NEW_list.append(time_NEW / 10)

    return LRU_list, LFU_list, RND_list, FIFO_list, NEW_list
```

- **도표 작성하는 함수**
  - **Parameter 설명** > - input_size : 캐쉬에 넣을 데이터 개수 > - input_max_value : 입력 패턴 0 ~ (input_max_value-1)사이의 난수
  - **함수 설명** > - matplotlib와 pandas 라이브러리를 사용하여 모든 case에 대한 결과 visualize

```
In [4]: def visualize(input_size, input_max_size):

        hit_avg, time_avg = [], []

        fig = plt.figure(figsize = (11,20))
```

```
gs = gridspec.GridSpec(4,2)

#slot_list = [25,50,75,100]
slot_list = [5, 10, 15, 20]
for i,slot_size in zip(range(4), slot_list):

    LRU_list, LFU_list, RND_list, FIFO_list, NEW_list = get_hit_result(input_size,
    LRU_tlist, LFU_tlist, RND_tlist, FIFO_tlist, NEW_tlist = get_time_result(input_

    def mean(list):
        return round(sum(list) / len(list), 2)

    hit_list = [LRU_list, LFU_list, RND_list, FIFO_list, NEW_list]
    time_list = [LRU_tlist, LFU_tlist, RND_tlist, FIFO_tlist, NEW_tlist]

    for hit, time in zip(hit_list, time_list):
        hit_avg.append(mean(hit))
        time_avg.append(mean(time))

    index = ['LRU', 'LFU', 'RND', 'FIFO', 'NEW']
    columns = [i for i in range(3,slot_size)]

    ax1 = fig.add_subplot(gs[i,0])
    plt.plot(columns, LRU_list, label = 'LRU')
    plt.plot(columns, LFU_list, label = 'LFU')
    plt.plot(columns, FIFO_list, label = 'FIFO')
    plt.plot(columns, RND_list, label = 'RND')
    plt.plot(columns, NEW_list, label = 'NEW')
    plt.scatter(columns, LRU_list)
    plt.scatter(columns, LFU_list)
    plt.scatter(columns, RND_list)
    plt.scatter(columns, FIFO_list)
    plt.scatter(columns, NEW_list)
    plt.title("Slot Size : " + str(slot_size))
    plt.ylabel("HIT Ratio")
    plt.xlabel("Slot Size")
```

```
        plt.legend(loc='best')

        ax2 = fig.add_subplot(gs[i,1])
        plt.plot(columns, LRU_tlist, label = 'LRU')
        plt.plot(columns, LFU_tlist, label = 'LFU')
        plt.plot(columns, FIFO_tlist, label = 'FIFO')
        plt.plot(columns, RND_tlist, label = 'RND')
        plt.plot(columns, NEW_tlist, label = 'NEW')
        plt.scatter(columns, LRU_tlist)
        plt.scatter(columns, LFU_tlist)
        plt.scatter(columns, FIFO_tlist)
        plt.scatter(columns, RND_tlist)
        plt.scatter(columns, NEW_tlist)

        plt.title("Slot Size : " + str(slot_size))
        plt.ylabel("Excution Time")
        plt.xlabel("Slot Size")
        plt.legend(loc='best')

    index = [
        'LRU / 25', 'LFU / 25', 'RND / 25', 'FIFO / 25', 'NEW / 25',
        'LRU / 50', 'LFU / 50', 'RND / 50', 'FIFO / 50', 'NEW / 50',
        'LRU / 75', 'LFU / 75', 'RND / 75', 'FIFO / 75', 'NEW / 75',
        'LRU / 100', 'LFU / 100', 'RND / 100', 'FIFO / 100', 'NEW / 100',
            ]
    print(len(NEW_list), len(FIFO_list))
    print("가능한 입력 수가 %d이며 난수의 최대 수가 %d일 때" % (input_size, input_max_size))
    return pd.DataFrame([hit_avg, time_avg], columns = index, index = ['HIT', 'TIME'])
    plt.show()
```

**가능한 입력 수가 10일 때**

**난수의 최대 수가 25일 때** 슬롯 사이즈를 5, 10, 15, 20으로 주어 적중률과 수행시간 비교

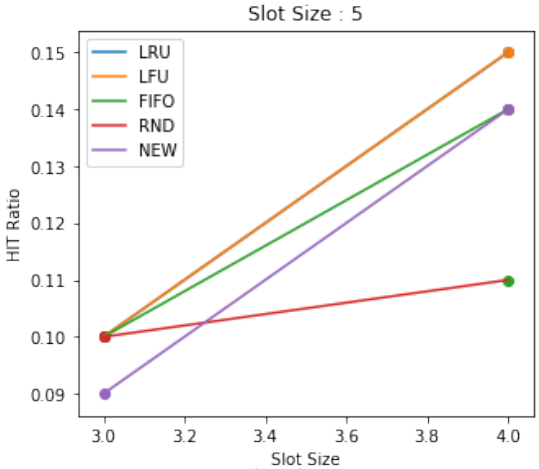
```
In [15]: visualize(input_size = 10, input_max_size = 25)
```

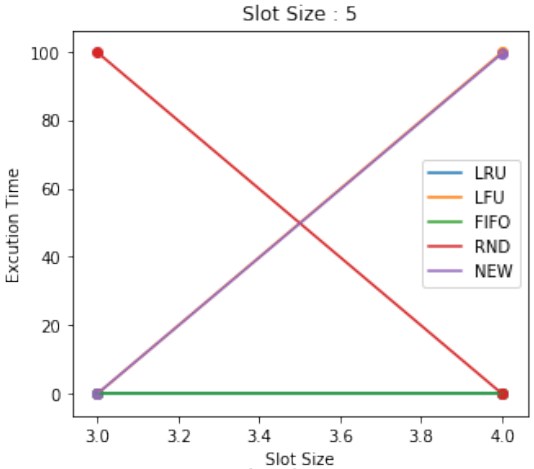
17 17

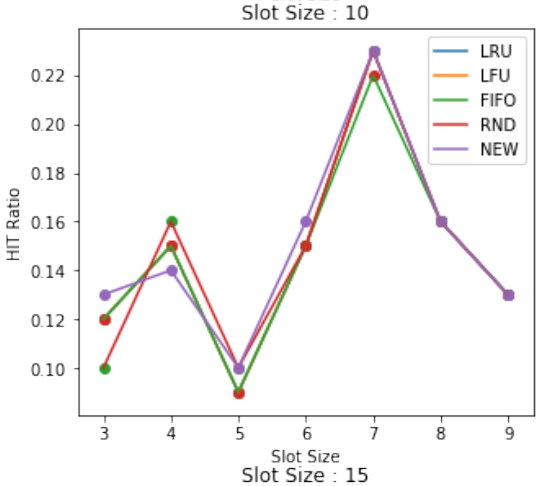
가능한 입력 수가 10이며 난수의 최대 수가 25일 때

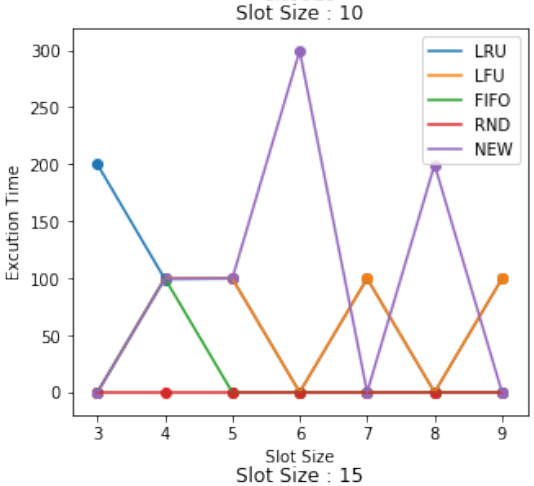
Out[15]:

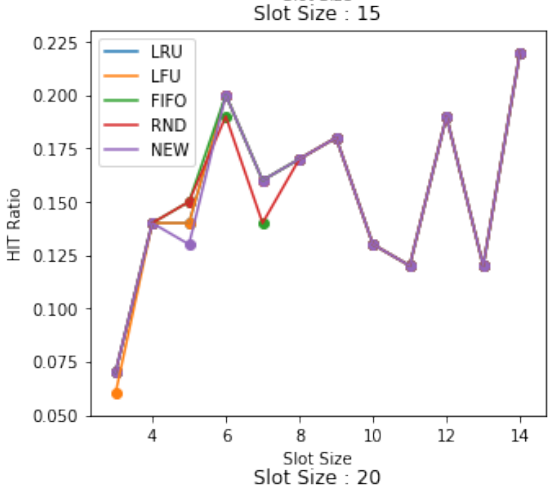
| | HIT | TIME |
|---|---|---|
| LRU / 25 | 0.12 | 0.00 |
| LFU / 25 | 0.12 | 49.90 |
| RND / 25 | 0.11 | 49.90 |
| FIFO / 25 | 0.12 | 0.00 |
| NEW / 25 | 0.11 | 49.75 |
| LRU / 50 | 0.15 | 85.43 |
| LFU / 50 | 0.15 | 57.03 |
| RND / 50 | 0.15 | 0.00 |
| FIFO / 50 | 0.15 | 14.27 |
| NEW / 50 | 0.15 | 99.76 |
| LRU / 75 | 0.15 | 8.31 |
| LFU / 75 | 0.15 | 33.26 |
| RND / 75 | 0.15 | 8.30 |
| FIFO / 75 | 0.15 | 41.58 |
| NEW / 75 | 0.15 | 33.25 |
| LRU / 100 | 0.14 | 11.75 |
| LFU / 100 | 0.14 | 11.75 |
| RND / 100 | 0.14 | 11.54 |
| FIFO / 100 | 0.14 | 12.08 |
| NEW / 100 | 0.14 | 11.57 |

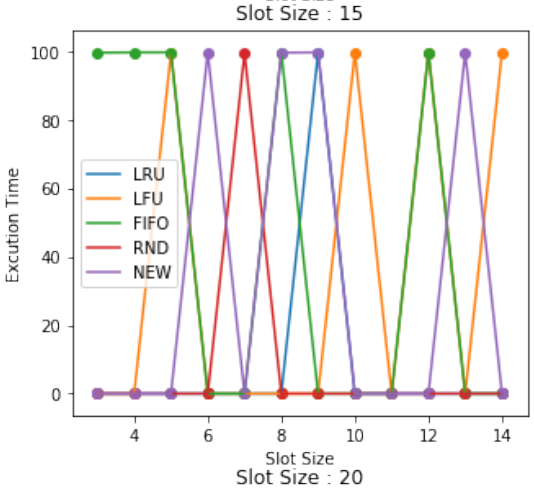

Slot Size : 5
HIT Ratio
Slot Size
LRU
LFU
FIFO
RND
NEW

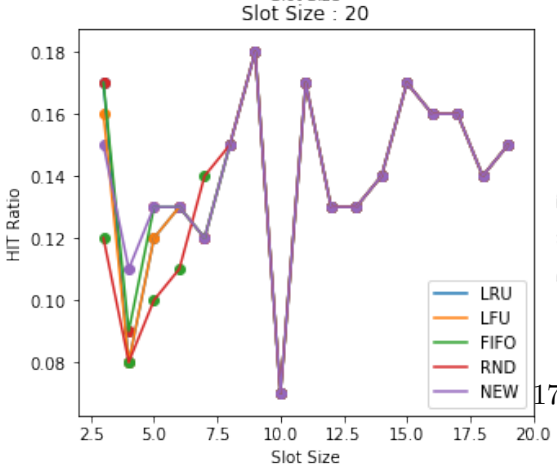

Slot Size : 5
Excution Time
Slot Size
LRU
LFU
FIFO
RND
NEW

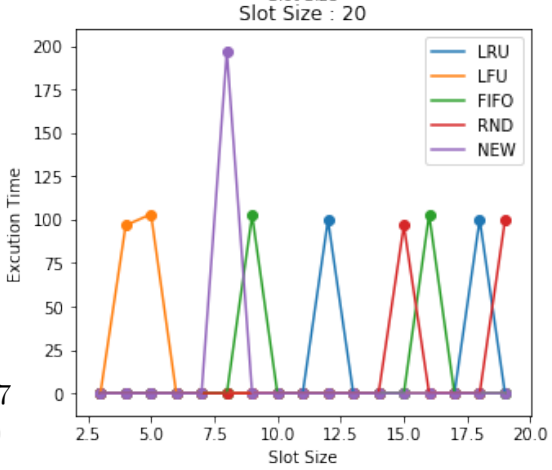

Slot Size : 10
HIT Ratio
Slot Size
LRU
LFU
FIFO
RND
NEW


Slot Size : 10
Excution Time
Slot Size
LRU
LFU
FIFO
RND
NEW


Slot Size : 15
HIT Ratio
Slot Size
LRU
LFU
FIFO
RND
NEW


Slot Size : 15
Excution Time
Slot Size
LRU
LFU
FIFO
RND
NEW


Slot Size : 20
HIT Ratio
Slot Size
LRU
LFU
FIFO
RND
NEW


Slot Size : 20
Excution Time
Slot Size
LRU
LFU
FIFO
RND
NEW

**난수의 최대 수가 50일 때** 슬롯 사이즈를 25,50,75,100으로 주어 적중률과 수행시간 비교

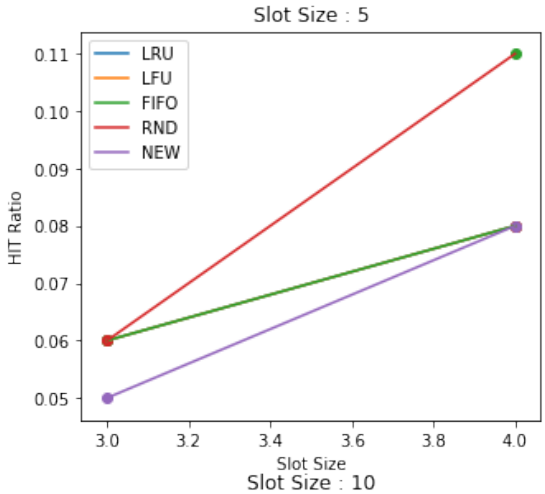
```
In [19]: visualize(input_size = 10, input_max_size = 50)
```

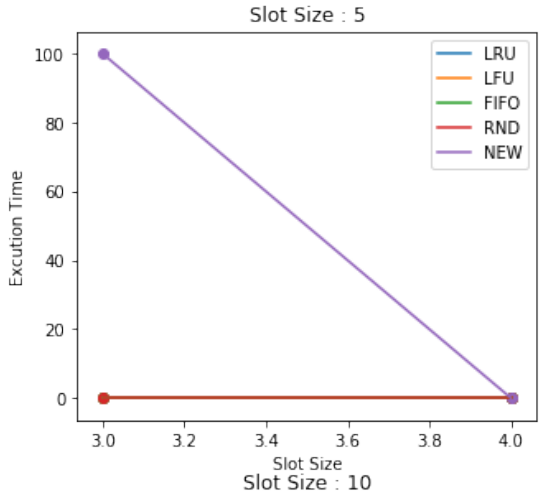
```
17 17
```

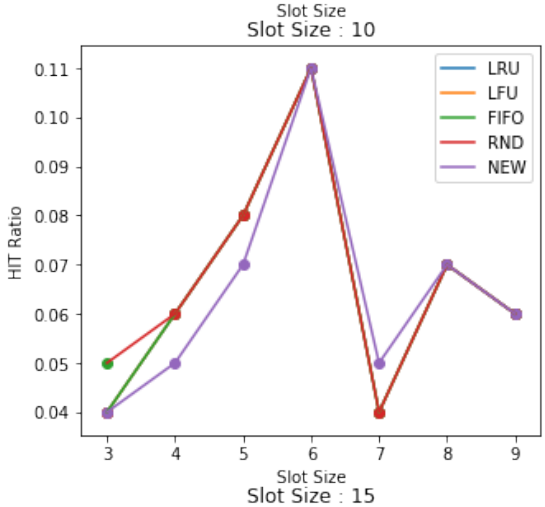
가능한 입력 수가 10이며 난수의 최대 수가 50일 때

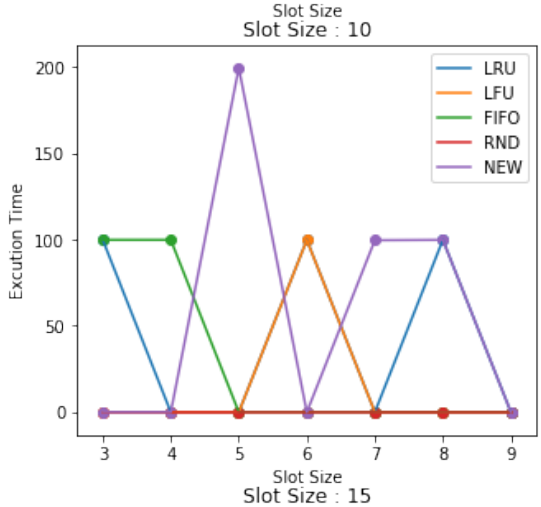
Out[19]:

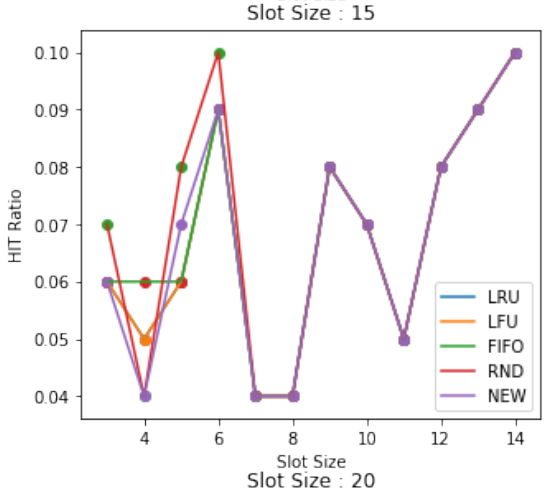
| | HIT | TIME |
|---|---|---|
| LRU / 25 | 0.07 | 0.00 |
| LFU / 25 | 0.07 | 0.00 |
| RND / 25 | 0.09 | 0.00 |
| FIFO / 25 | 0.07 | 0.00 |
| NEW / 25 | 0.07 | 49.95 |
| LRU / 50 | 0.07 | 42.77 |
| LFU / 50 | 0.07 | 14.24 |
| RND / 50 | 0.07 | 0.00 |
| FIFO / 50 | 0.07 | 28.51 |
| NEW / 50 | 0.06 | 56.97 |
| LRU / 75 | 0.07 | 8.32 |
| LFU / 75 | 0.07 | 33.24 |
| RND / 75 | 0.07 | 16.63 |
| FIFO / 75 | 0.07 | 16.62 |
| NEW / 75 | 0.07 | 16.62 |
| LRU / 100 | 0.08 | 23.47 |
| LFU / 100 | 0.07 | 17.59 |
| RND / 100 | 0.08 | 0.00 |
| FIFO / 100 | 0.08 | 5.86 |
| NEW / 100 | 0.07 | 17.61 |

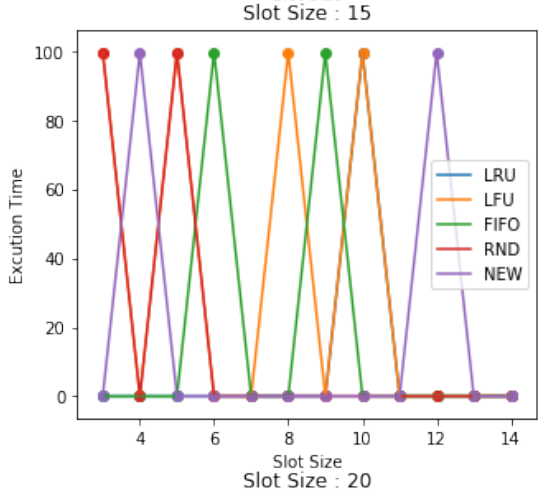

Slot Size : 5
HIT Ratio
Slot Size
LRU
LFU
FIFO
RND
NEW

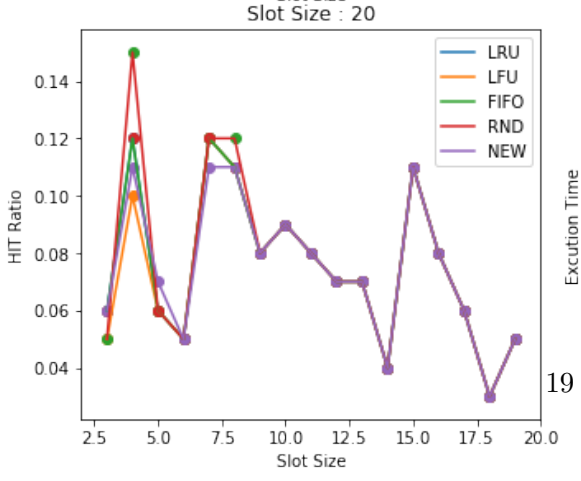

Slot Size : 5
Excution Time
Slot Size
LRU
LFU
FIFO
RND
NEW

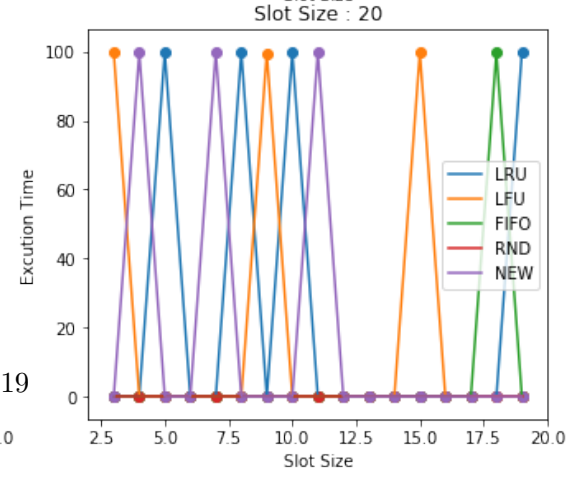

Slot Size : 10
HIT Ratio
Slot Size
LRU
LFU
FIFO
RND
NEW


Slot Size : 10
Excution Time
Slot Size
LRU
LFU
FIFO
RND
NEW


Slot Size : 15
HIT Ratio
Slot Size
LRU
LFU
FIFO
RND
NEW


Slot Size : 15
Excution Time
Slot Size
LRU
LFU
FIFO
RND
NEW


Slot Size : 20
HIT Ratio
Slot Size
LRU
LFU
FIFO
RND
NEW


Slot Size : 20
Excution Time
Slot Size
LRU
LFU
FIFO
RND
NEW

**난수의 최대 수가 75일 때** 슬롯 사이즈를 25,50,75,100으로 주어 적중률과 수행시간 비교

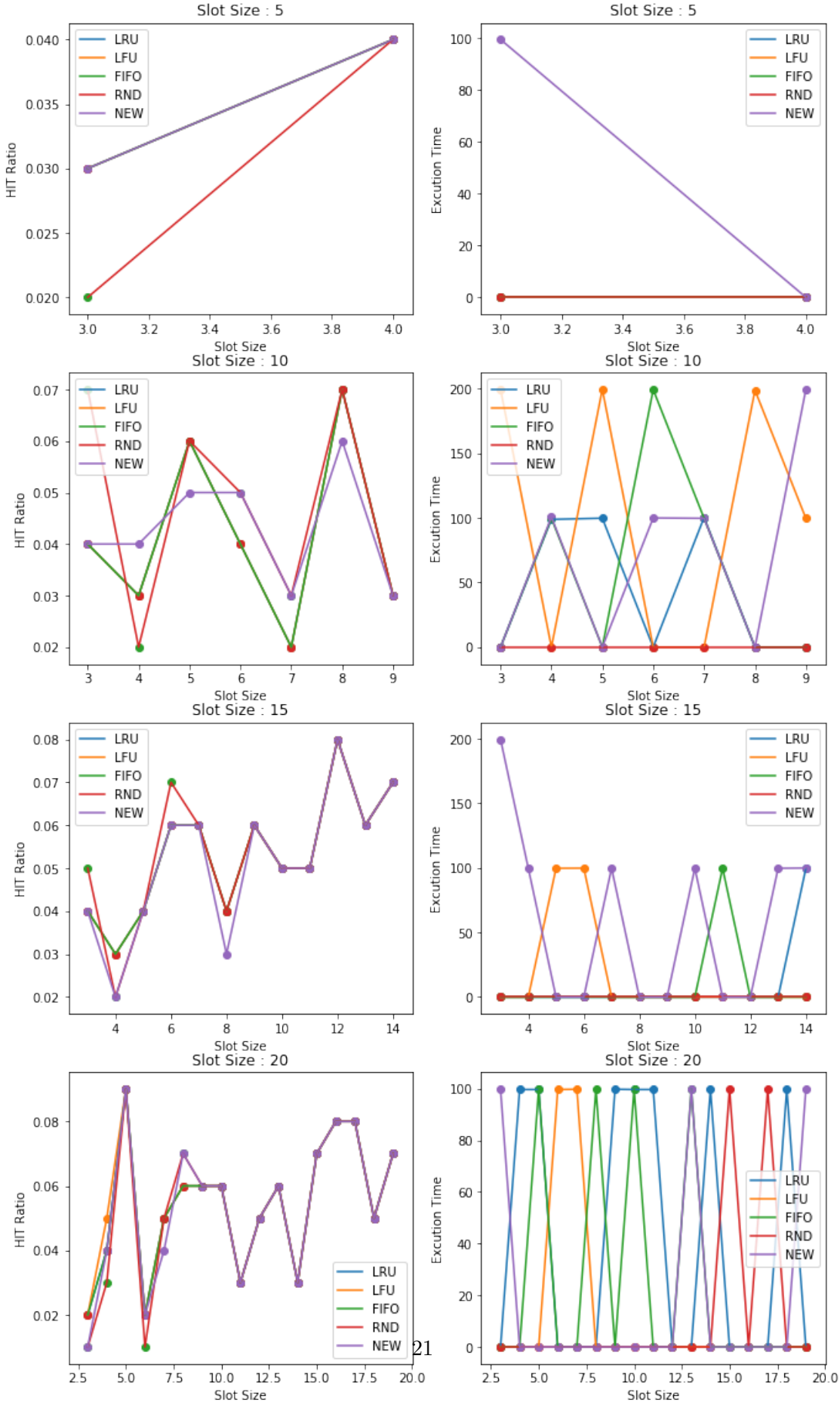
```
In [20]: visualize(input_size = 10, input_max_size = 75)
```

```
17 17
```

가능한 입력 수가 10이며 난수의 최대 수가 75일 때

Out[20]:

| | HIT | TIME |
|---|---|---|
| LRU / 25 | 0.04 | 0.00 |
| LFU / 25 | 0.04 | 0.00 |
| RND / 25 | 0.03 | 0.00 |
| FIFO / 25 | 0.04 | 0.00 |
| NEW / 25 | 0.04 | 49.75 |
| LRU / 50 | 0.04 | 42.73 |
| LFU / 50 | 0.04 | 99.64 |
| RND / 50 | 0.05 | 0.00 |
| FIFO / 50 | 0.04 | 56.96 |
| NEW / 50 | 0.04 | 71.40 |
| LRU / 75 | 0.05 | 8.30 |
| LFU / 75 | 0.05 | 16.62 |
| RND / 75 | 0.05 | 0.00 |
| FIFO / 75 | 0.05 | 8.31 |
| NEW / 75 | 0.05 | 58.18 |
| LRU / 100 | 0.05 | 41.06 |
| LFU / 100 | 0.05 | 11.74 |
| RND / 100 | 0.05 | 11.74 |
| FIFO / 100 | 0.05 | 23.47 |
| NEW / 100 | 0.05 | 17.60 |


Slot Size : 5
HIT Ratio
Slot Size
LRU
LFU
FIFO
RND
NEW
Slot Size : 5
Excution Time
Slot Size
Slot Size : 10
HIT Ratio
Slot Size
Slot Size : 10
Excution Time
Slot Size
Slot Size : 15
HIT Ratio
Slot Size
Slot Size : 15
Excution Time
Slot Size
Slot Size : 20
HIT Ratio
Slot Size
Slot Size : 20
Excution Time
Slot Size

**난수의 최대 수가 100일 때** 슬롯 사이즈를 25,50,75,100으로 주어 적중률과 수행시간 비교

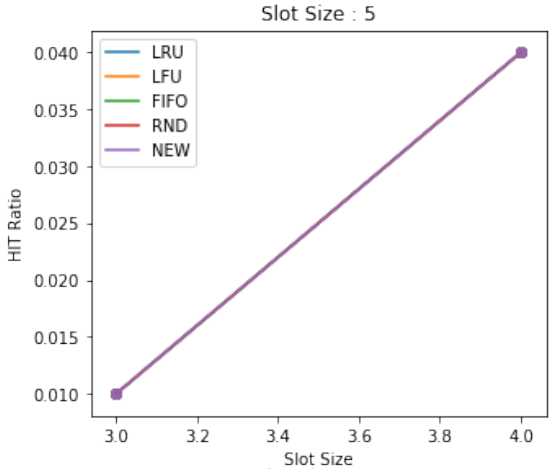
```
In [21]: visualize(input_size = 10, input_max_size = 100)
```

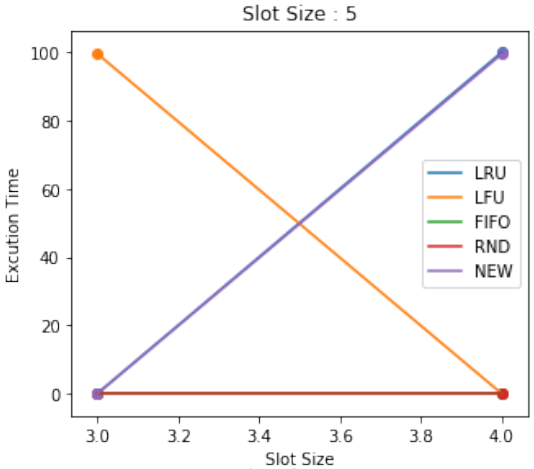
```
17 17
```

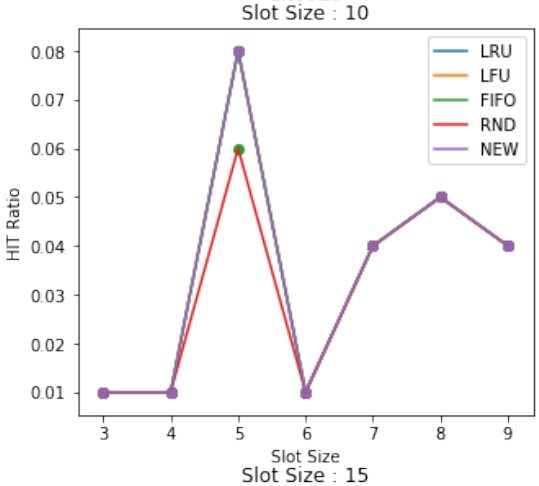
가능한 입력 수가 10이며 난수의 최대 수가 100일 때

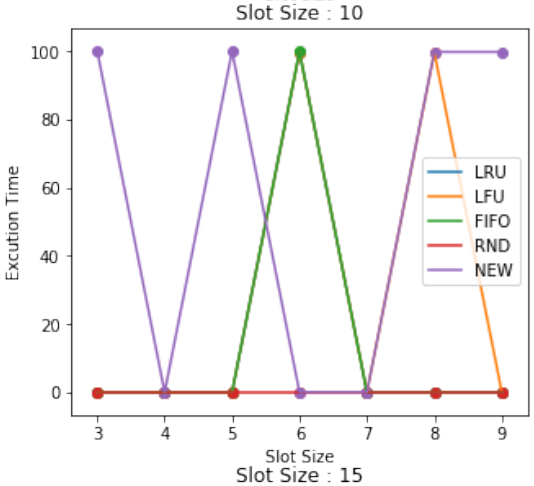
Out[21]:

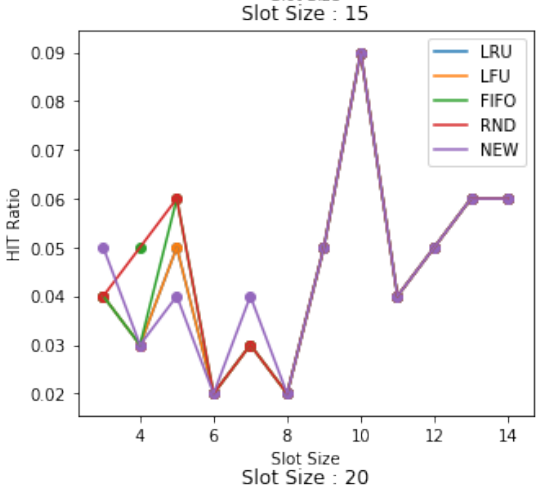
| | HIT | TIME |
|---|---|---|
| LRU / 25 | 0.03 | 50.05 |
| LFU / 25 | 0.03 | 49.85 |
| RND / 25 | 0.03 | 0.00 |
| FIFO / 25 | 0.03 | 0.00 |
| NEW / 25 | 0.03 | 49.80 |
| LRU / 50 | 0.03 | 14.24 |
| LFU / 50 | 0.03 | 28.46 |
| RND / 50 | 0.03 | 0.00 |
| FIFO / 50 | 0.03 | 14.26 |
| NEW / 50 | 0.03 | 57.04 |
| LRU / 75 | 0.04 | 41.53 |
| LFU / 75 | 0.04 | 24.96 |
| RND / 75 | 0.05 | 0.00 |
| FIFO / 75 | 0.05 | 8.33 |
| NEW / 75 | 0.05 | 24.92 |
| LRU / 100 | 0.04 | 29.26 |
| LFU / 100 | 0.04 | 11.74 |
| RND / 100 | 0.04 | 5.86 |
| FIFO / 100 | 0.04 | 17.68 |
| NEW / 100 | 0.04 | 11.74 |

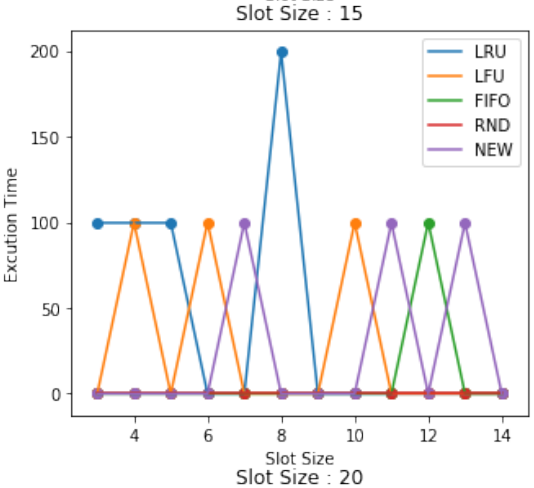

Slot Size : 5
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size

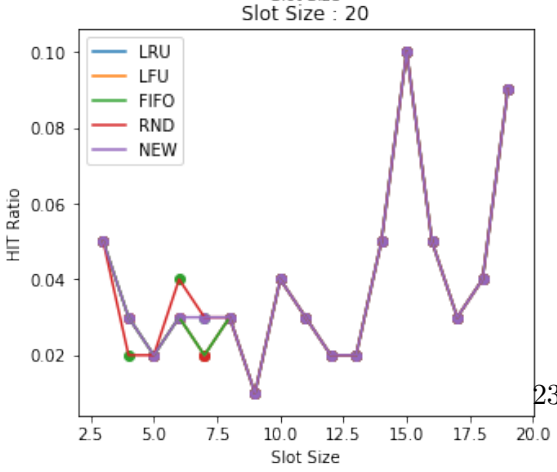

Slot Size : 5
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size

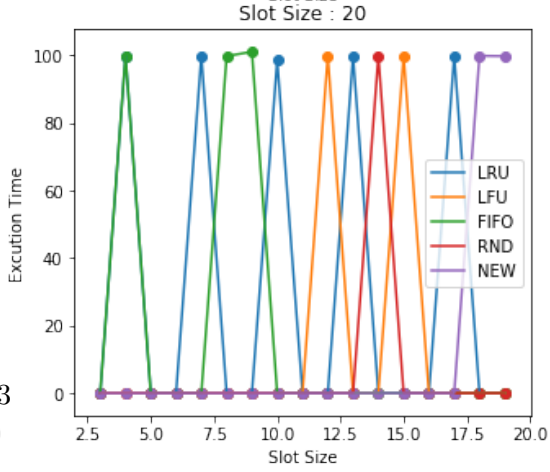

Slot Size : 10
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 10
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size


Slot Size : 15
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 15
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size


Slot Size : 20
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 20
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size

**가능한 입력 수가 100일 때**

**난수의 최대 수가 25일 때** 슬롯 사이즈를 25,50,75,100으로 주어 적중률과 수행시간 비교

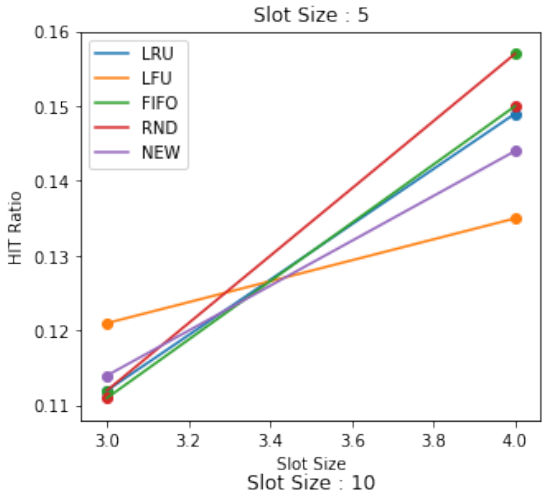
```
In [16]: visualize(input_size = 100, input_max_size = 25)
```

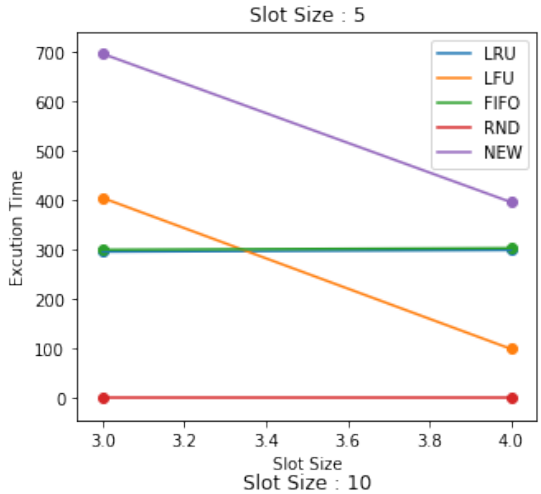
17 17

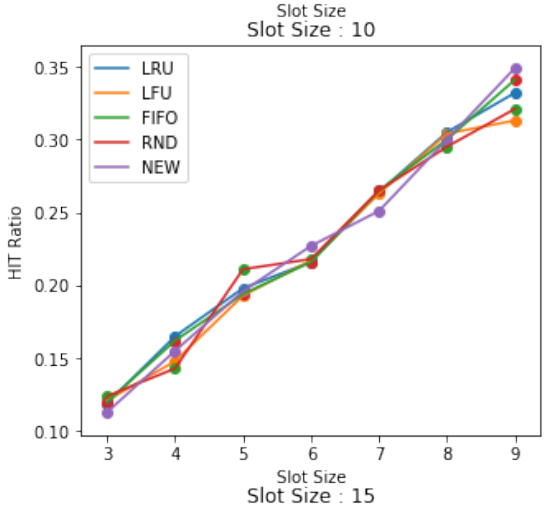
가능한 입력 수가 100이며 난수의 최대 수가 25일 때

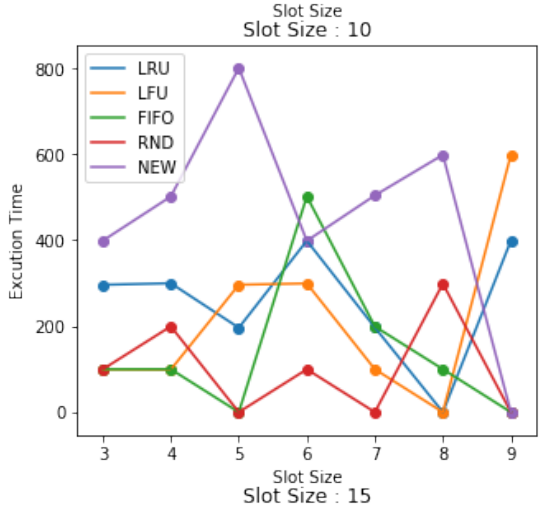
Out[16]:

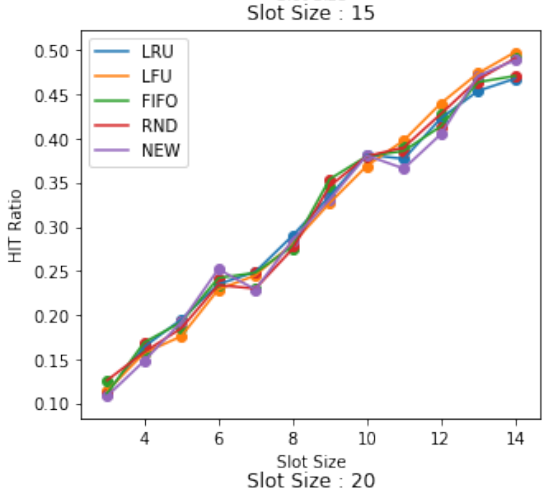
| | HIT | TIME |
|---|---|---|
| LRU / 25 | 0.13 | 297.40 |
| LFU / 25 | 0.13 | 251.85 |
| RND / 25 | 0.13 | 0.00 |
| FIFO / 25 | 0.13 | 301.35 |
| NEW / 25 | 0.13 | 545.90 |
| LRU / 50 | 0.23 | 255.30 |
| LFU / 50 | 0.22 | 213.26 |
| RND / 50 | 0.23 | 99.70 |
| FIFO / 50 | 0.23 | 142.96 |
| NEW / 50 | 0.23 | 457.46 |
| LRU / 75 | 0.31 | 91.03 |
| LFU / 75 | 0.31 | 340.51 |
| RND / 75 | 0.31 | 58.39 |
| FIFO / 75 | 0.31 | 199.47 |
| NEW / 75 | 0.30 | 416.18 |
| LRU / 100 | 0.39 | 311.16 |
| LFU / 100 | 0.38 | 334.79 |
| RND / 100 | 0.39 | 58.55 |
| FIFO / 100 | 0.39 | 93.82 |
| NEW / 100 | 0.39 | 381.15 |

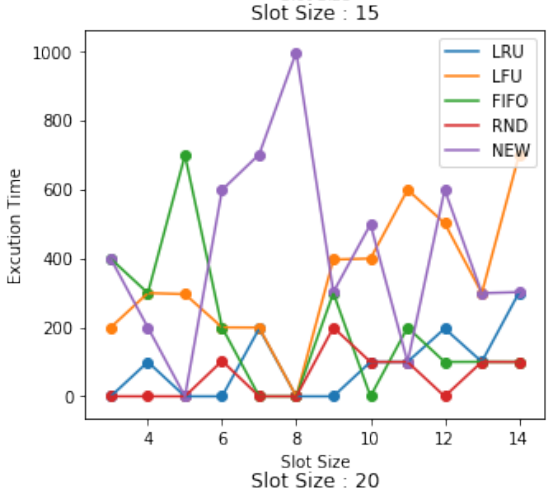

Slot Size : 5
HIT Ratio
Slot Size
LRU
LFU
FIFO
RND
NEW

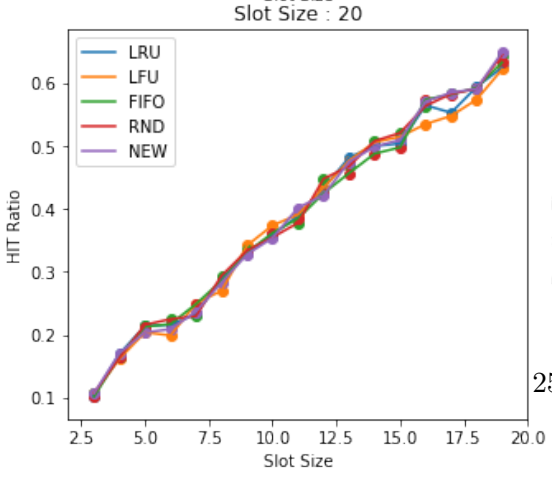

Slot Size : 5
Excution Time
Slot Size
LRU
LFU
FIFO
RND
NEW

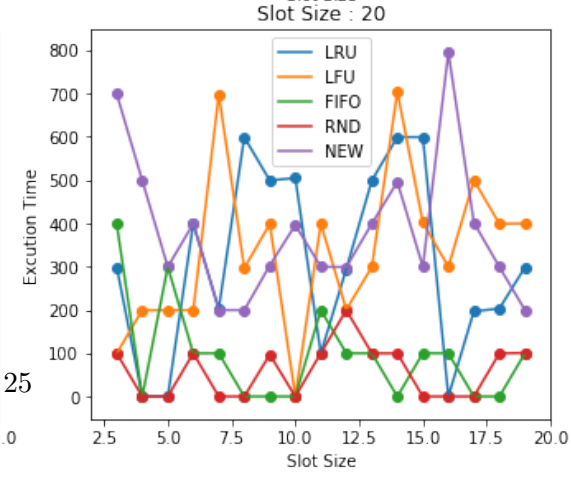

Slot Size : 10
HIT Ratio
Slot Size
LRU
LFU
FIFO
RND
NEW


Slot Size : 10
Excution Time
Slot Size
LRU
LFU
FIFO
RND
NEW


Slot Size : 15
HIT Ratio
Slot Size
LRU
LFU
FIFO
RND
NEW


Slot Size : 15
Excution Time
Slot Size
LRU
LFU
FIFO
RND
NEW


Slot Size : 20
HIT Ratio
Slot Size
LRU
LFU
FIFO
RND
NEW


Slot Size : 20
Excution Time
Slot Size
LRU
LFU
FIFO
RND
NEW

**난수의 최대 수가 50일 때** 슬롯 사이즈를 25,50,75,100으로 주어 적중률과 수행시간 비교

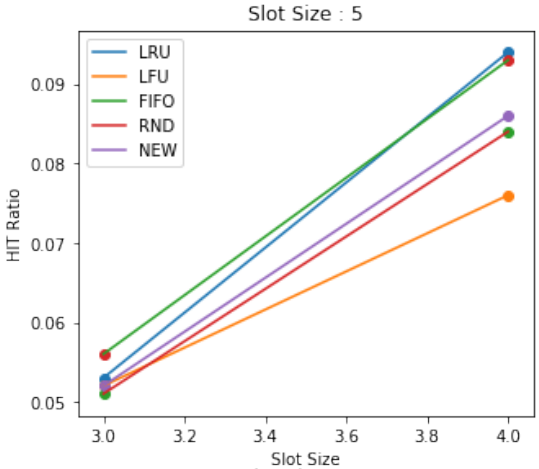
```
In [22]: visualize(input_size = 100, input_max_size = 50)
```

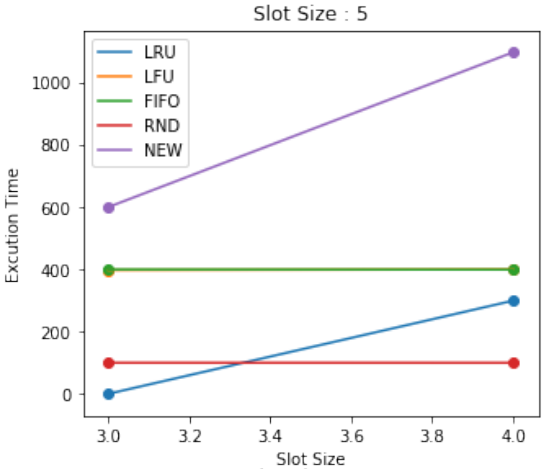
```
17 17
```

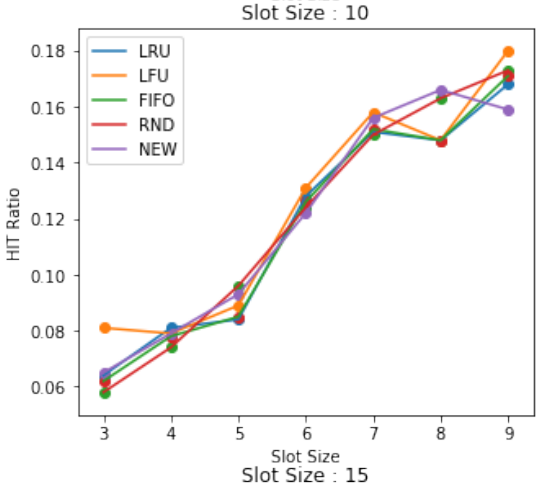
가능한 입력 수가 100이며 난수의 최대 수가 50일 때

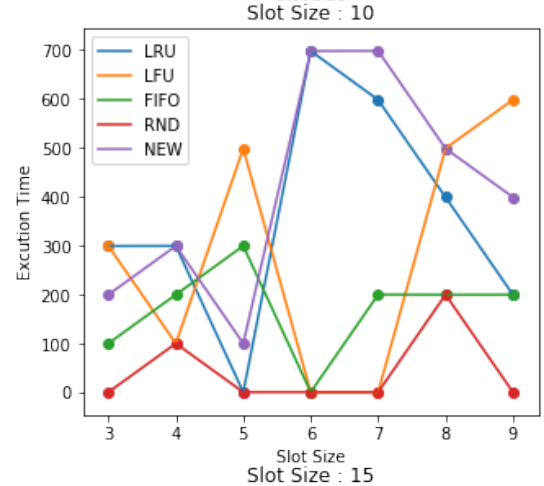
Out[22]:

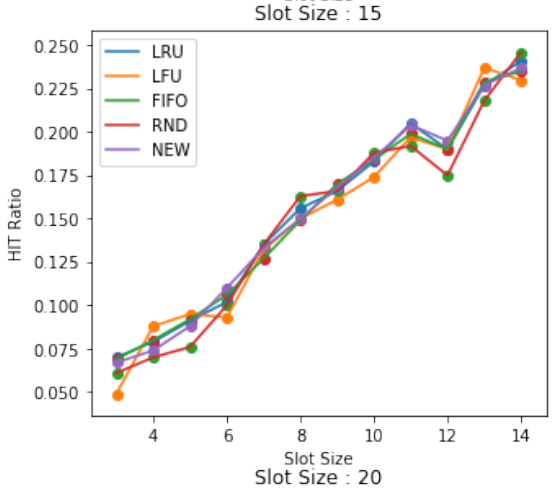
| | HIT | TIME |
|---|---|---|
| LRU / 25 | 0.07 | 149.55 |
| LFU / 25 | 0.06 | 398.40 |
| RND / 25 | 0.07 | 99.75 |
| FIFO / 25 | 0.07 | 399.10 |
| NEW / 25 | 0.07 | 848.15 |
| LRU / 50 | 0.12 | 356.13 |
| LFU / 50 | 0.12 | 284.96 |
| RND / 50 | 0.12 | 42.76 |
| FIFO / 50 | 0.12 | 170.99 |
| NEW / 50 | 0.12 | 413.21 |
| LRU / 75 | 0.15 | 365.79 |
| LFU / 75 | 0.15 | 207.92 |
| RND / 75 | 0.15 | 95.72 |
| FIFO / 75 | 0.15 | 249.06 |
| NEW / 75 | 0.15 | 532.21 |
| LRU / 100 | 0.20 | 381.36 |
| LFU / 100 | 0.20 | 263.95 |
| RND / 100 | 0.20 | 82.06 |
| FIFO / 100 | 0.20 | 205.25 |
| NEW / 100 | 0.20 | 616.14 |

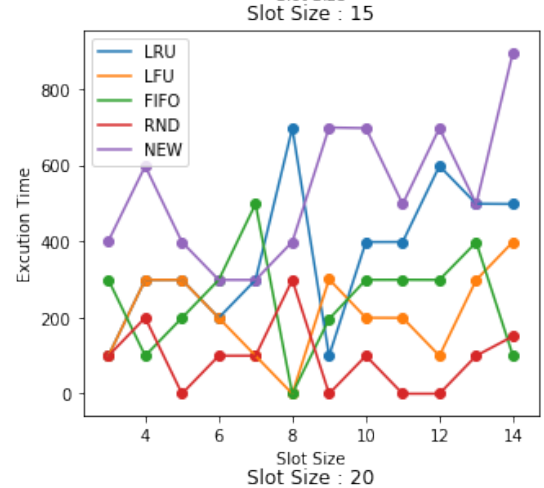

Slot Size : 5
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size

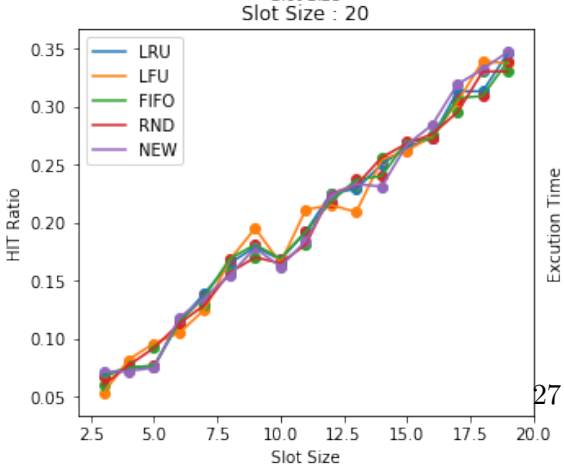

Slot Size : 5
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size

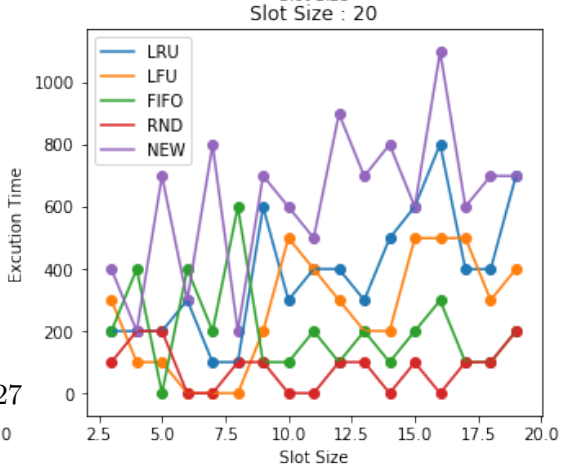

Slot Size : 10
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 10
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size


Slot Size : 15
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 15
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size


Slot Size : 20
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 20
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size

**난수의 최대 수가 75일 때** 슬롯 사이즈를 25,50,75,100으로 주어 적중률과 수행시간 비교

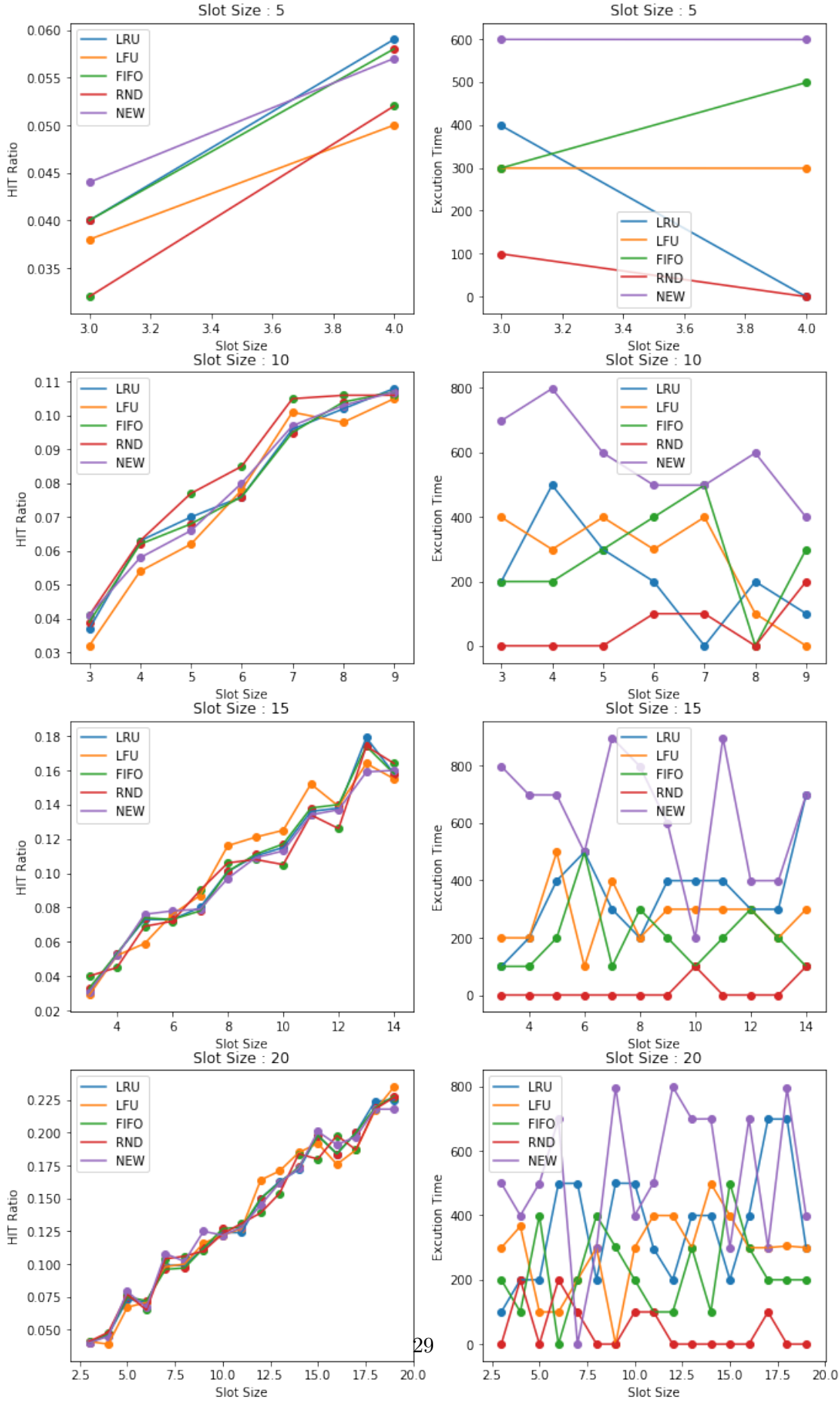
```
In [23]: visualize(input_size = 100, input_max_size = 75)
```

```
17 17
```

가능한 입력 수가 100이며 난수의 최대 수가 75일 때

Out[23]:

| | HIT | TIME |
|---|---|---|
| LRU / 25 | 0.05 | 199.05 |
| LFU / 25 | 0.04 | 299.10 |
| RND / 25 | 0.04 | 49.80 |
| FIFO / 25 | 0.05 | 399.10 |
| NEW / 25 | 0.05 | 598.85 |
| LRU / 50 | 0.08 | 213.71 |
| LFU / 50 | 0.08 | 270.70 |
| RND / 50 | 0.08 | 56.99 |
| FIFO / 50 | 0.08 | 270.76 |
| NEW / 50 | 0.08 | 584.16 |
| LRU / 75 | 0.10 | 349.12 |
| LFU / 75 | 0.11 | 274.24 |
| RND / 75 | 0.10 | 16.62 |
| FIFO / 75 | 0.10 | 199.46 |
| NEW / 75 | 0.10 | 631.63 |
| LRU / 100 | 0.14 | 369.42 |
| LFU / 100 | 0.14 | 285.78 |
| RND / 100 | 0.13 | 46.94 |
| FIFO / 100 | 0.14 | 223.04 |
| NEW / 100 | 0.14 | 516.10 |


Slot Size : 5
HIT Ratio
Slot Size
LRU
LFU
FIFO
RND
NEW
Slot Size : 5
Excution Time
Slot Size
Slot Size : 10
HIT Ratio
Slot Size
Slot Size : 10
Excution Time
Slot Size
Slot Size : 15
HIT Ratio
Slot Size
Slot Size : 15
Excution Time
Slot Size
Slot Size : 20
HIT Ratio
Slot Size
Slot Size : 20
Excution Time
Slot Size

**난수의 최대 수가 100일 때** 슬롯 사이즈를 25,50,75,100으로 주어 적중률과 수행시간 비교

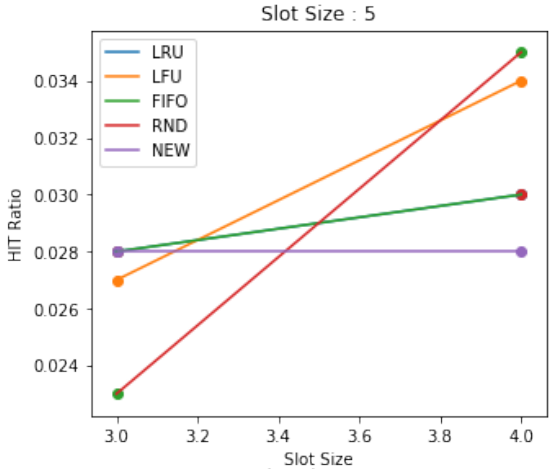
```
In [24]: visualize(input_size = 100, input_max_size = 100)
```

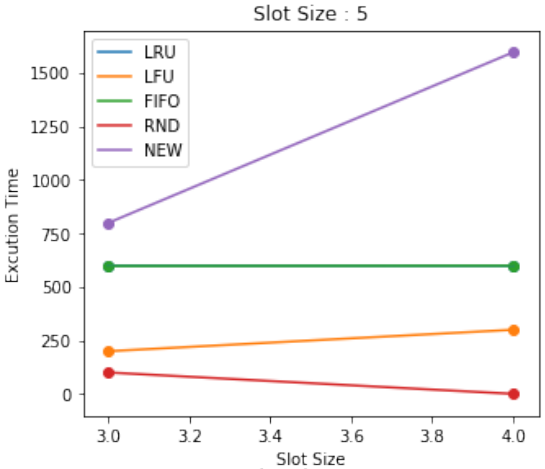
```
17 17
```

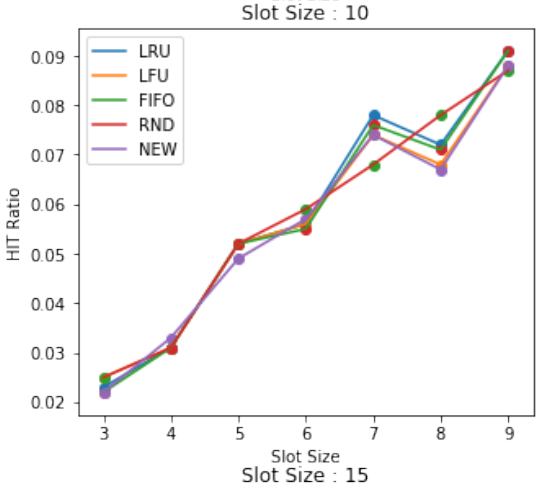
가능한 입력 수가 100이며 난수의 최대 수가 100일 때

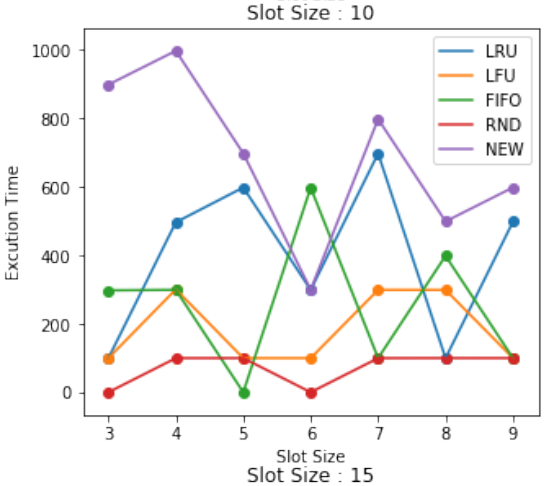
Out[24]:

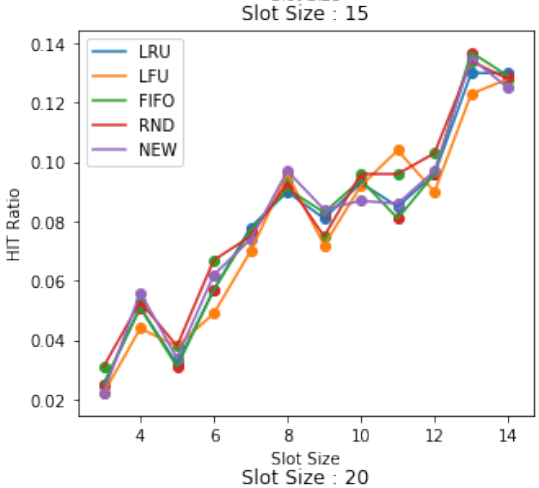
| | HIT | TIME |
|---|---|---|
| LRU / 25 | 0.03 | 598.80 |
| LFU / 25 | 0.03 | 249.25 |
| RND / 25 | 0.03 | 49.90 |
| FIFO / 25 | 0.03 | 598.10 |
| NEW / 25 | 0.03 | 1197.25 |
| LRU / 50 | 0.06 | 398.56 |
| LFU / 50 | 0.06 | 185.39 |
| RND / 50 | 0.06 | 71.21 |
| FIFO / 50 | 0.06 | 256.16 |
| NEW / 50 | 0.06 | 684.40 |
| LRU / 75 | 0.08 | 440.38 |
| LFU / 75 | 0.08 | 241.03 |
| RND / 75 | 0.08 | 58.17 |
| FIFO / 75 | 0.08 | 224.60 |
| NEW / 75 | 0.08 | 606.68 |
| LRU / 100 | 0.10 | 492.78 |
| LFU / 100 | 0.10 | 357.83 |
| RND / 100 | 0.10 | 29.32 |
| FIFO / 100 | 0.10 | 252.35 |
| NEW / 100 | 0.10 | 692.31 |

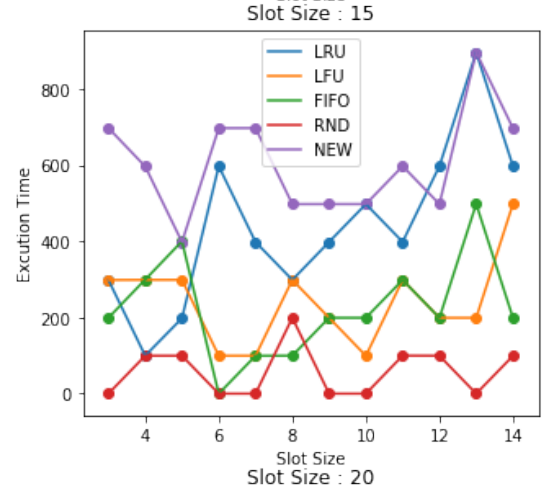

Slot Size : 5
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size

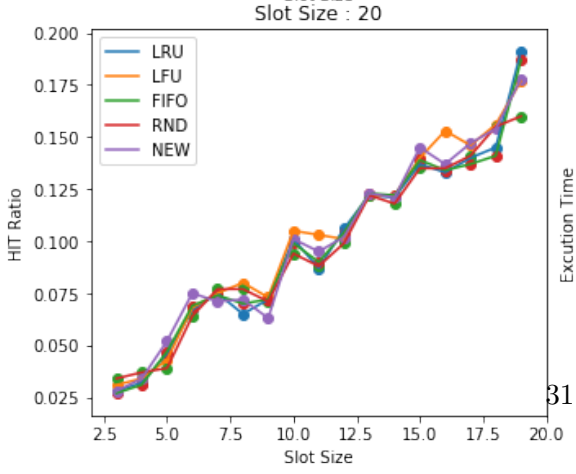

Slot Size : 5
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size

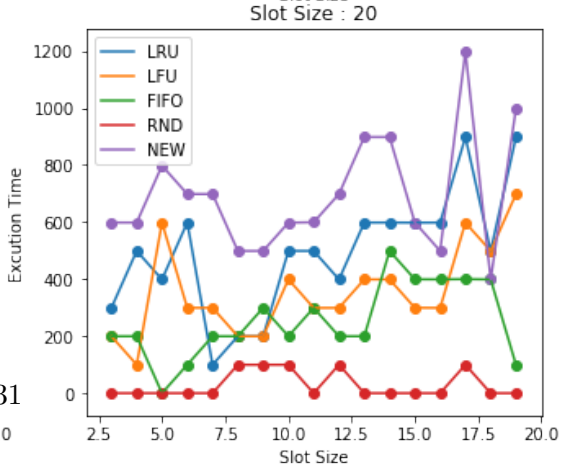

Slot Size : 10
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 10
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size


Slot Size : 15
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 15
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size


Slot Size : 20
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 20
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size

**가능한 입력 수가 1000일 때**

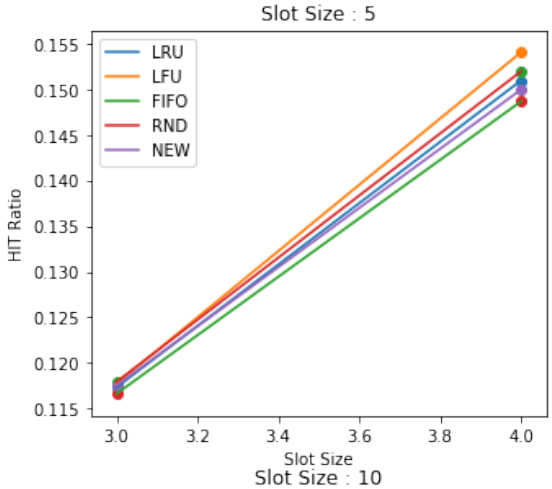
**난수의 최대 수가 25일 때** 슬롯 사이즈를 25,50,75,100으로 주어 적중률과 수행시간 비교

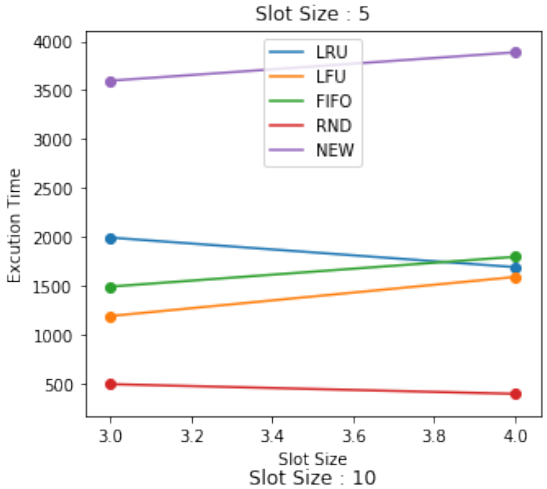
```
In [17]: visualize(input_size = 1000, input_max_size = 25)
```

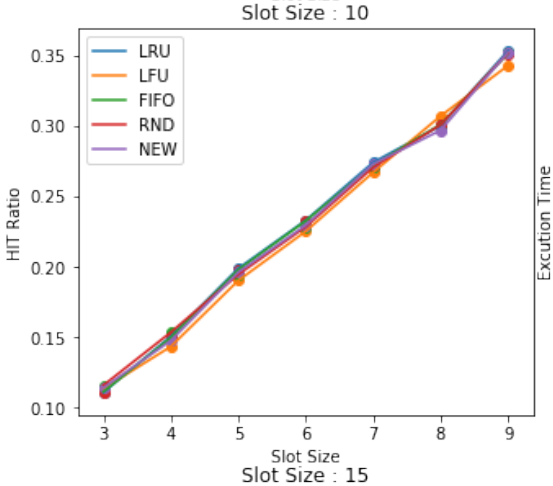
17 17
가능한 입력 수가 1000이며 난수의 최대 수가 25일 때

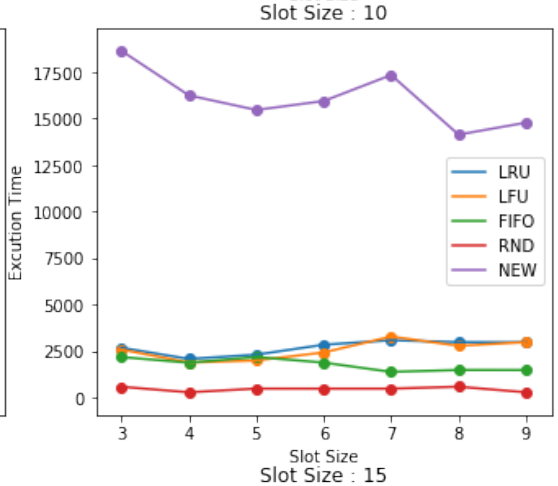
Out[17]:

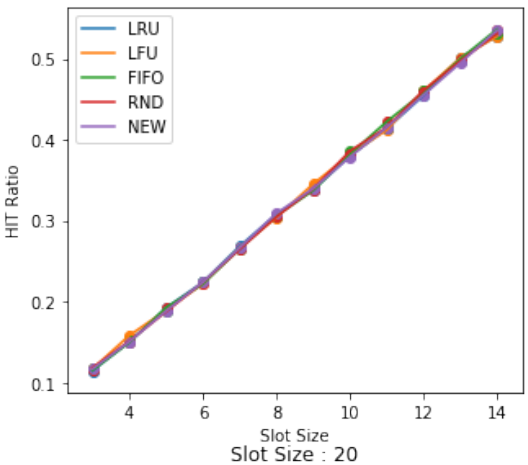
| | HIT | TIME |
|---|---|---|
| LRU / 25 | 0.13 | 1843.40 |
| LFU / 25 | 0.14 | 1392.75 |
| RND / 25 | 0.13 | 448.05 |
| FIFO / 25 | 0.13 | 1646.00 |
| NEW / 25 | 0.13 | 3741.75 |
| LRU / 50 | 0.23 | 2720.33 |
| LFU / 50 | 0.23 | 2576.50 |
| RND / 50 | 0.23 | 469.51 |
| FIFO / 50 | 0.23 | 1794.04 |
| NEW / 50 | 0.23 | 16085.60 |
| LRU / 75 | 0.32 | 3083.71 |
| LFU / 75 | 0.32 | 3094.03 |
| RND / 75 | 0.32 | 486.71 |
| FIFO / 75 | 0.32 | 1686.39 |
| NEW / 75 | 0.32 | 4996.22 |
| LRU / 100 | 0.42 | 3569.51 |
| LFU / 100 | 0.42 | 3841.68 |
| RND / 100 | 0.42 | 473.80 |
| FIFO / 100 | 0.42 | 1562.06 |
| NEW / 100 | 0.42 | 5183.90 |

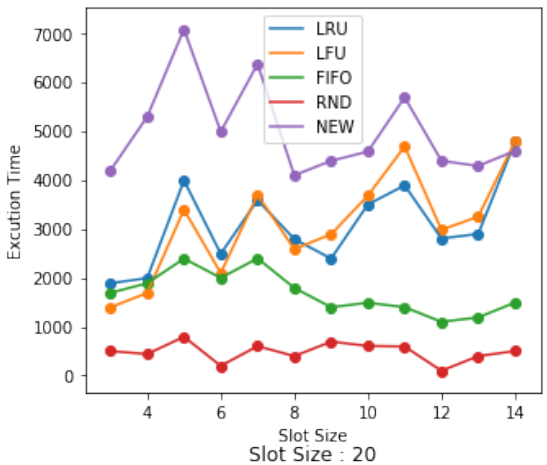

Slot Size : 5
HIT Ratio
Slot Size
LRU
LFU
FIFO
RND
NEW

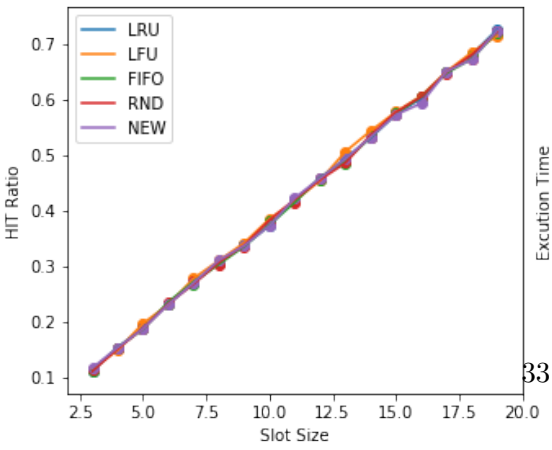

Slot Size : 5
Excution Time
Slot Size
LRU
LFU
FIFO
RND
NEW

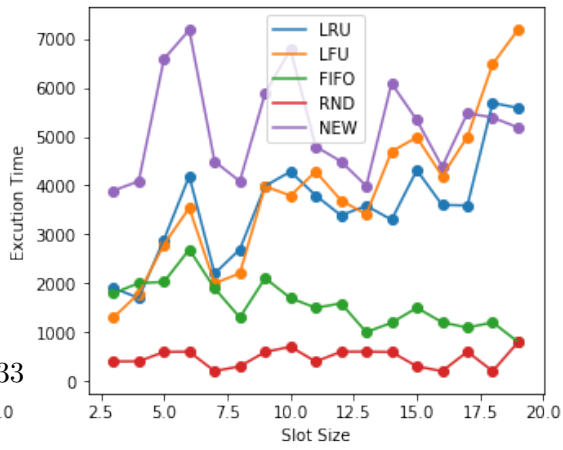

Slot Size : 10
HIT Ratio
Slot Size
LRU
LFU
FIFO
RND
NEW


Slot Size : 10
Excution Time
Slot Size
LRU
LFU
FIFO
RND
NEW


Slot Size : 15
HIT Ratio
Slot Size
LRU
LFU
FIFO
RND
NEW


Slot Size : 15
Excution Time
Slot Size
LRU
LFU
FIFO
RND
NEW


Slot Size : 20
HIT Ratio
Slot Size
LRU
LFU
FIFO
RND
NEW


Slot Size : 20
Excution Time
Slot Size
LRU
LFU
FIFO
RND
NEW

**난수의 최대 수가 50일 때** 슬롯 사이즈를 25,50,75,100으로 주어 적중률과 수행시간 비교

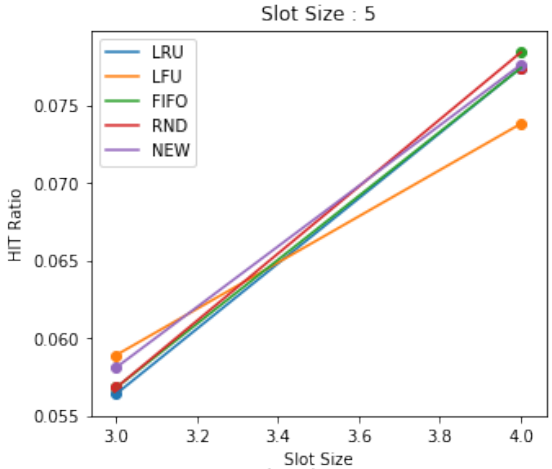
```
In [25]: visualize(input_size = 1000, input_max_size = 50)
```

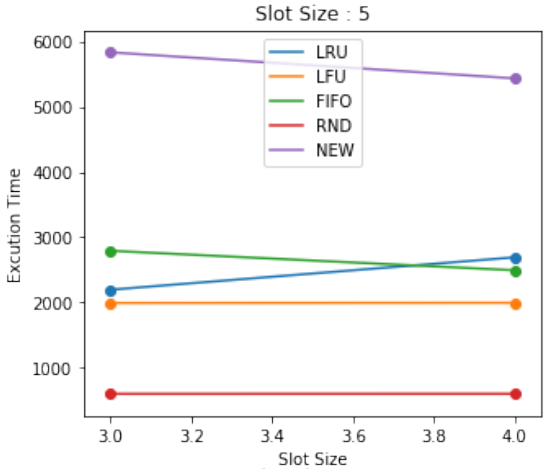
```
17 17
```

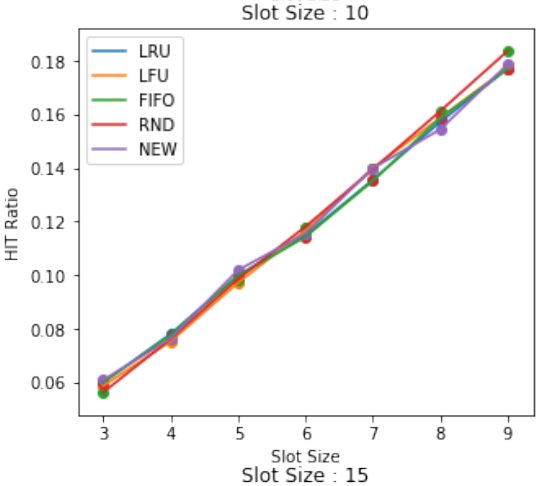
가능한 입력 수가 1000이며 난수의 최대 수가 50일 때

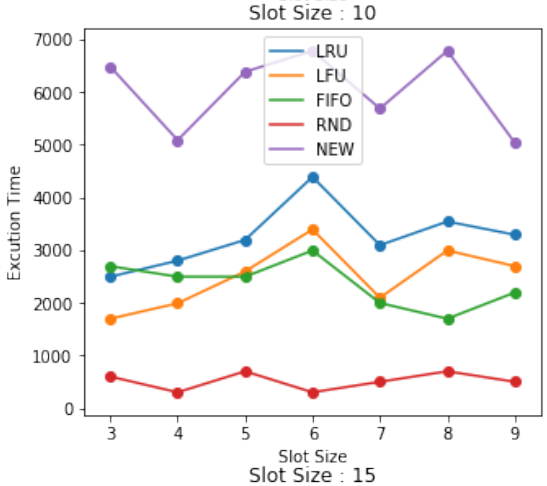
Out[25]:

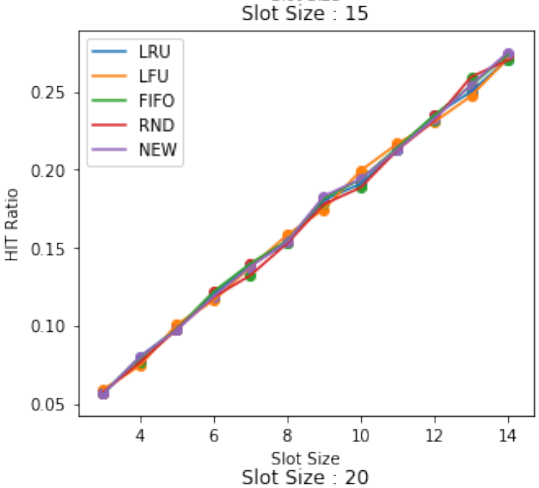
| | HIT | TIME |
|---|---|---|
| LRU / 25 | 0.07 | 2443.50 |
| LFU / 25 | 0.07 | 1993.25 |
| RND / 25 | 0.07 | 598.55 |
| FIFO / 25 | 0.07 | 2642.95 |
| NEW / 25 | 0.07 | 5636.40 |
| LRU / 50 | 0.12 | 3256.36 |
| LFU / 50 | 0.12 | 2492.63 |
| RND / 50 | 0.12 | 512.83 |
| FIFO / 50 | 0.12 | 2365.10 |
| NEW / 50 | 0.12 | 6033.97 |
| LRU / 75 | 0.17 | 4421.25 |
| LFU / 75 | 0.17 | 3453.82 |
| RND / 75 | 0.16 | 573.74 |
| FIFO / 75 | 0.17 | 2698.22 |
| NEW / 75 | 0.17 | 7342.53 |
| LRU / 100 | 0.21 | 4141.69 |
| LFU / 100 | 0.21 | 3470.05 |
| RND / 100 | 0.21 | 564.59 |
| FIFO / 100 | 0.21 | 2214.19 |
| NEW / 100 | 0.21 | 6332.79 |

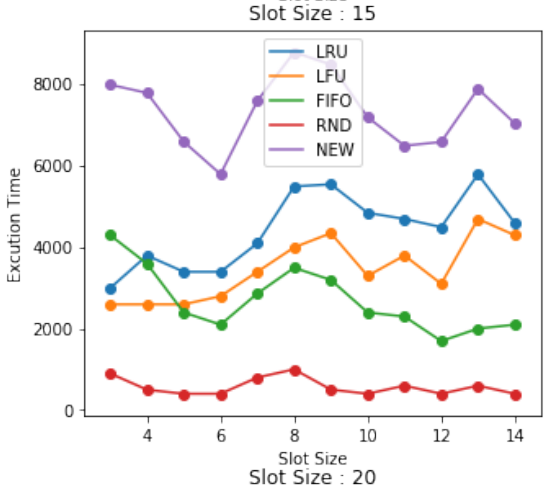

Slot Size : 5
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size

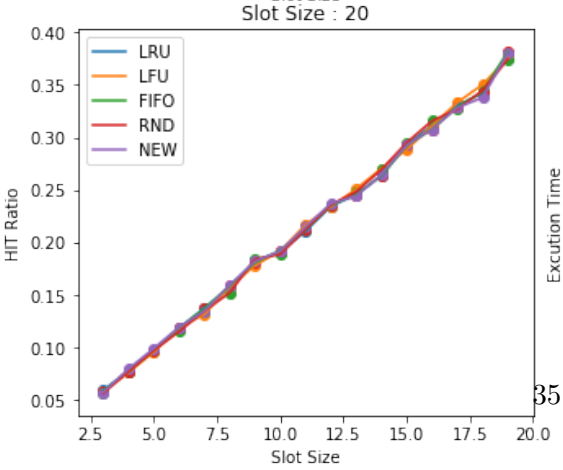

Slot Size : 5
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size

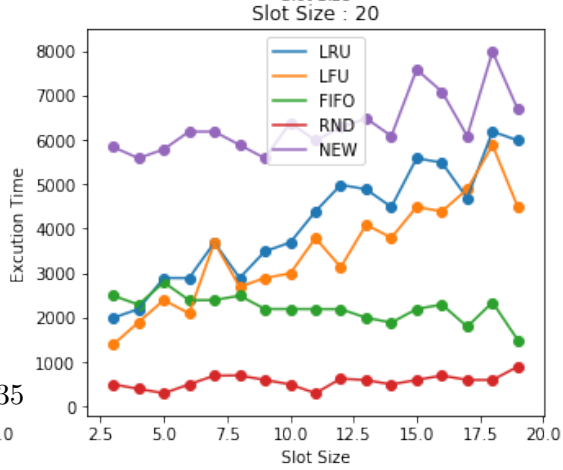

Slot Size : 10
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 10
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size


Slot Size : 15
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 15
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size


Slot Size : 20
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 20
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size

**난수의 최대 수가 75일 때** 슬롯 사이즈를 25,50,75,100으로 주어 적중률과 수행시간 비교

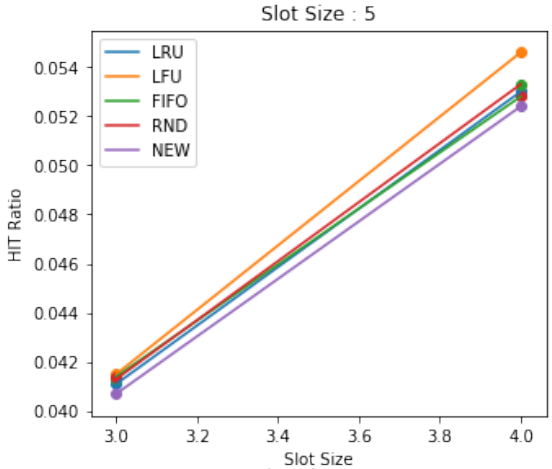
```
In [26]: visualize(input_size = 1000, input_max_size = 75)
```

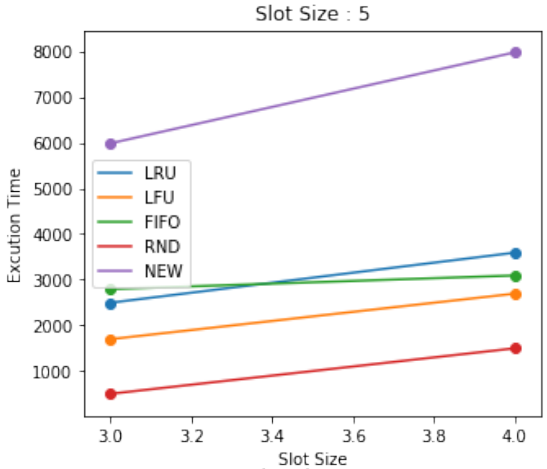
```
17 17
```

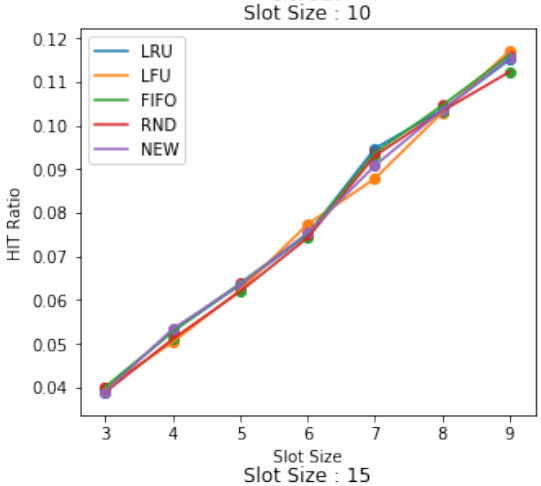
가능한 입력 수가 1000이며 난수의 최대 수가 75일 때

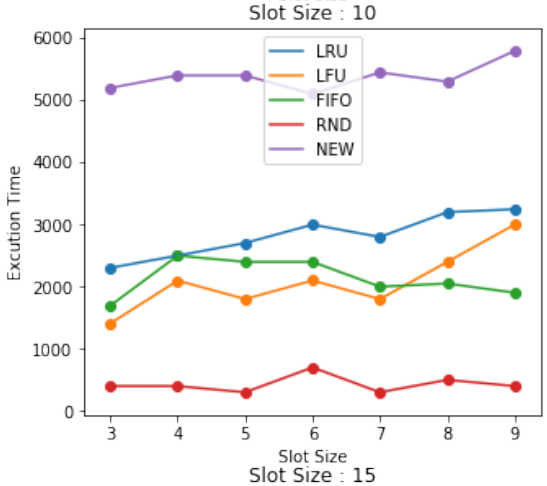
Out[26]:

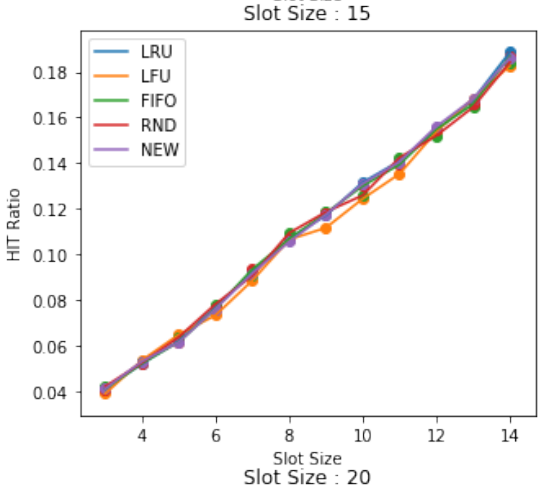
| | HIT | TIME |
|---|---|---|
| LRU / 25 | 0.05 | 3042.05 |
| LFU / 25 | 0.05 | 2194.35 |
| RND / 25 | 0.05 | 997.50 |
| FIFO / 25 | 0.05 | 2941.80 |
| NEW / 25 | 0.05 | 6981.25 |
| LRU / 50 | 0.08 | 2814.13 |
| LFU / 50 | 0.08 | 2080.99 |
| RND / 50 | 0.08 | 427.51 |
| FIFO / 50 | 0.08 | 2129.44 |
| NEW / 50 | 0.08 | 5363.77 |
| LRU / 75 | 0.11 | 3940.94 |
| LFU / 75 | 0.11 | 2909.17 |
| RND / 75 | 0.11 | 634.95 |
| FIFO / 75 | 0.11 | 2465.07 |
| NEW / 75 | 0.11 | 6673.43 |
| LRU / 100 | 0.14 | 5300.12 |
| LFU / 100 | 0.14 | 3625.64 |
| RND / 100 | 0.14 | 633.24 |
| FIFO / 100 | 0.14 | 2652.66 |
| NEW / 100 | 0.14 | 7883.99 |

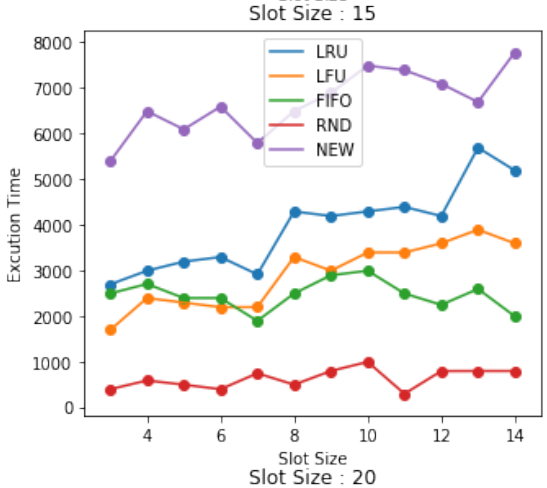

Slot Size : 5
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size

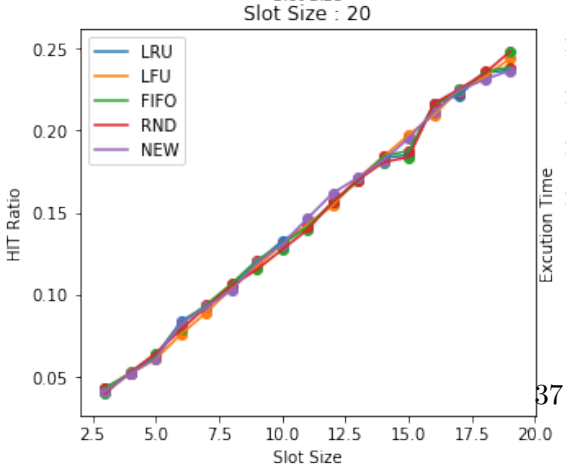

Slot Size : 5
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size

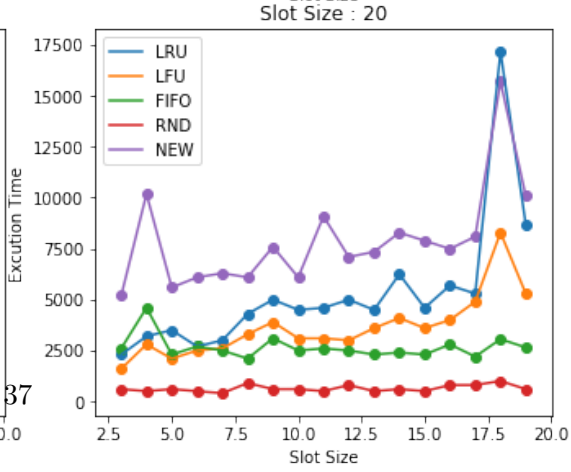

Slot Size : 10
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 10
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size


Slot Size : 15
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 15
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size


Slot Size : 20
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 20
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size

**난수의 최대 수가 100일 때** 슬롯 사이즈를 25,50,75,100으로 주어 적중률과 수행시간 비교

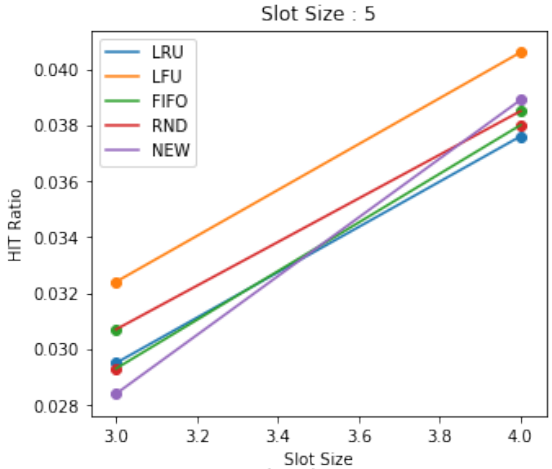
```
In [27]: visualize(input_size = 1000, input_max_size = 100)
```

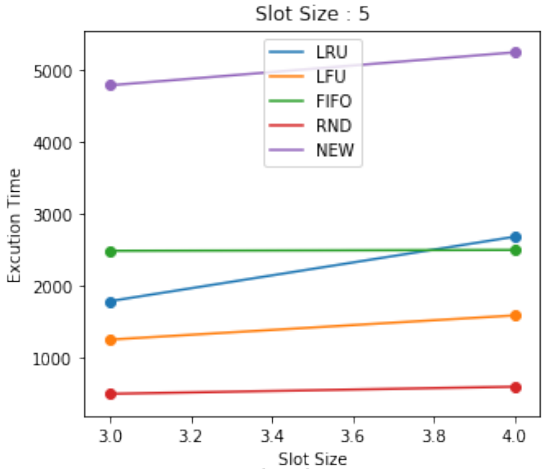
```
17 17
```

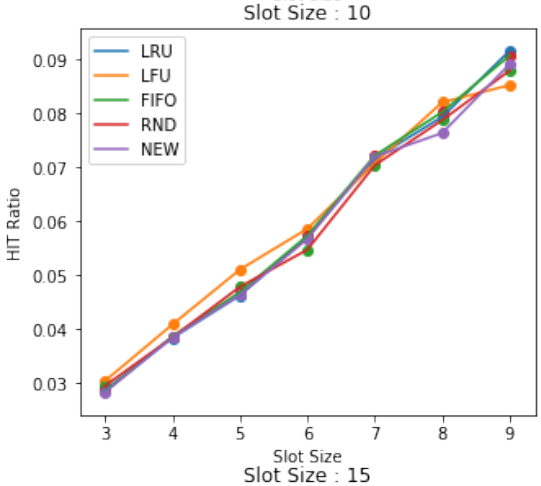
가능한 입력 수가 1000이며 난수의 최대 수가 100일 때

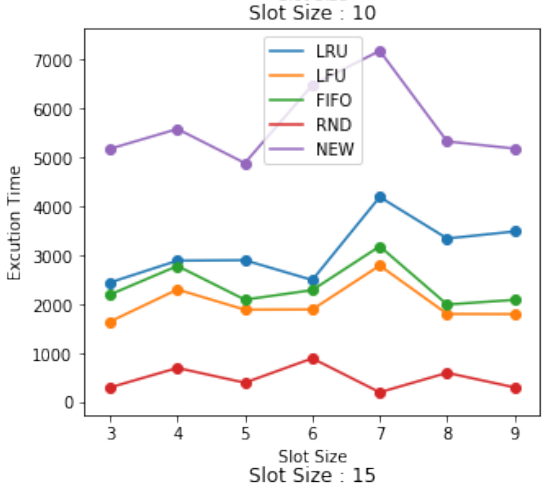
Out[27]:

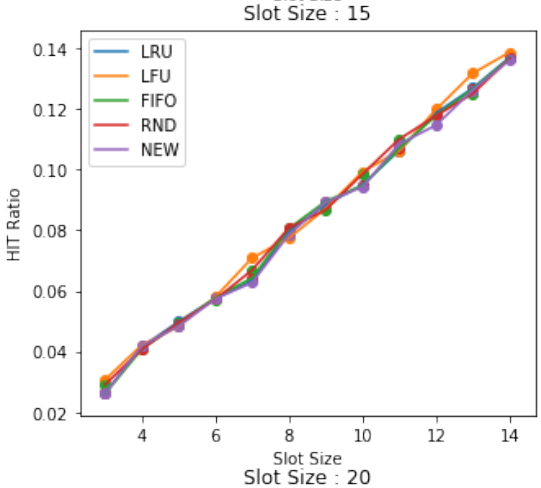
| | HIT | TIME |
|---|---|---|
| LRU / 25 | 0.03 | 2235.20 |
| LFU / 25 | 0.04 | 1422.40 |
| RND / 25 | 0.03 | 551.80 |
| FIFO / 25 | 0.03 | 2492.80 |
| NEW / 25 | 0.03 | 5012.25 |
| LRU / 50 | 0.06 | 3109.03 |
| LFU / 50 | 0.06 | 2017.50 |
| RND / 50 | 0.06 | 483.49 |
| FIFO / 50 | 0.06 | 2376.81 |
| NEW / 50 | 0.06 | 5690.40 |
| LRU / 75 | 0.08 | 3474.45 |
| LFU / 75 | 0.08 | 2311.11 |
| RND / 75 | 0.08 | 522.94 |
| FIFO / 75 | 0.08 | 2101.06 |
| NEW / 75 | 0.08 | 5794.70 |
| LRU / 100 | 0.11 | 5111.49 |
| LFU / 100 | 0.11 | 3717.11 |
| RND / 100 | 0.11 | 782.87 |
| FIFO / 100 | 0.11 | 3158.34 |
| NEW / 100 | 0.11 | 8501.11 |

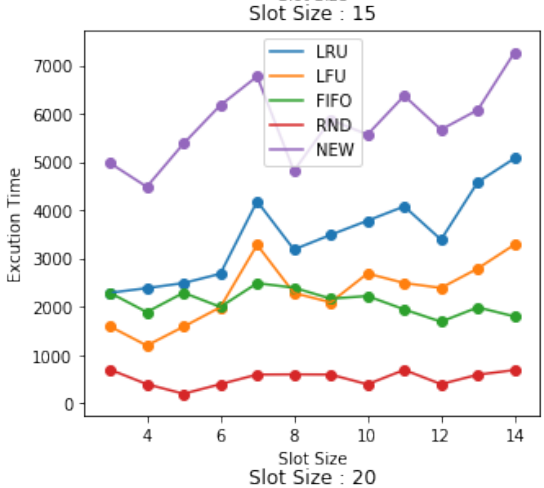

Slot Size : 5
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size

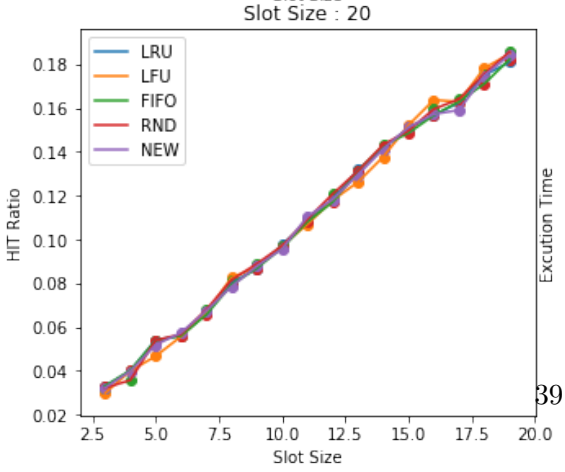

Slot Size : 5
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size

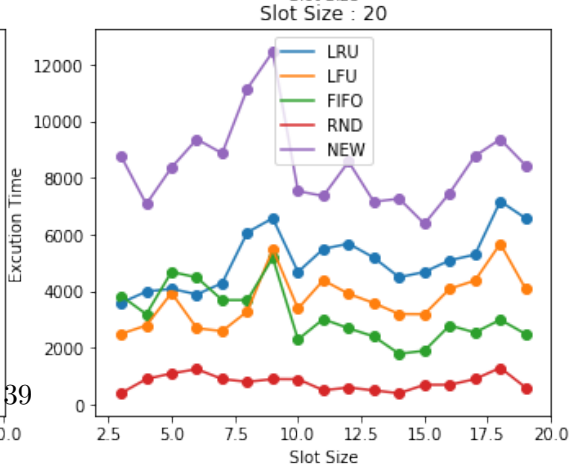

Slot Size : 10
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 10
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size


Slot Size : 15
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 15
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size


Slot Size : 20
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 20
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size

**가능한 입력 수가 10000일 때**

**난수의 최대 수가 25일 때** 슬롯 사이즈를 25,50,75,100으로 주어 적중률과 수행시간 비교

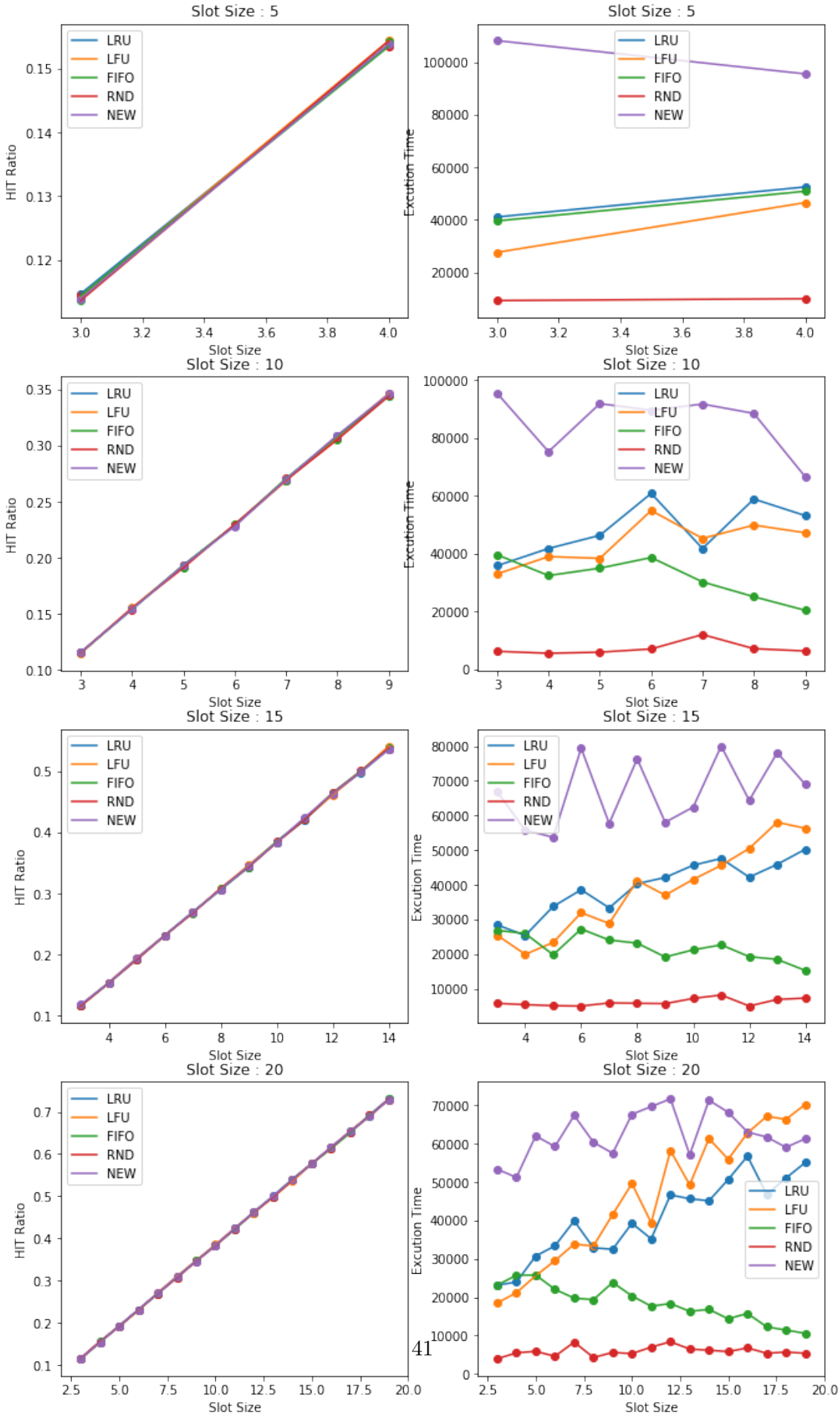
```
In [28]: visualize(input_size = 10000, input_max_size = 25)
```

17 17

가능한 입력 수가 10000이며 난수의 최대 수가 25일 때

Out[28]:

| | HIT | TIME |
|---|---|---|
| LRU / 25 | 0.13 | 46799.25 |
| LFU / 25 | 0.13 | 37084.60 |
| RND / 25 | 0.13 | 9623.55 |
| FIFO / 25 | 0.13 | 45231.25 |
| NEW / 25 | 0.13 | 101878.95 |
| LRU / 50 | 0.23 | 48282.11 |
| LFU / 50 | 0.23 | 43858.07 |
| RND / 50 | 0.23 | 7121.43 |
| FIFO / 50 | 0.23 | 31553.24 |
| NEW / 50 | 0.23 | 85440.67 |
| LRU / 75 | 0.33 | 39433.30 |
| LFU / 75 | 0.33 | 38293.50 |
| RND / 75 | 0.33 | 6093.47 |
| FIFO / 75 | 0.33 | 21908.07 |
| NEW / 75 | 0.33 | 66796.74 |
| LRU / 100 | 0.42 | 40502.96 |
| LFU / 100 | 0.42 | 46100.21 |
| RND / 100 | 0.42 | 5906.22 |
| FIFO / 100 | 0.42 | 18433.83 |
| NEW / 100 | 0.42 | 62456.35 |


Slot Size : 5
HIT Ratio
Slot Size
LRU
LFU
FIFO
RND
NEW
Slot Size : 5
Excution Time
Slot Size
Slot Size : 10
HIT Ratio
Slot Size
Slot Size : 10
Excution Time
Slot Size
Slot Size : 15
HIT Ratio
Slot Size
Slot Size : 15
Excution Time
Slot Size
Slot Size : 20
HIT Ratio
Slot Size
Slot Size : 20
Excution Time
Slot Size

**난수의 최대 수가 50일 때** 슬롯 사이즈를 25,50,75,100으로 주어 적중률과 수행시간 비교

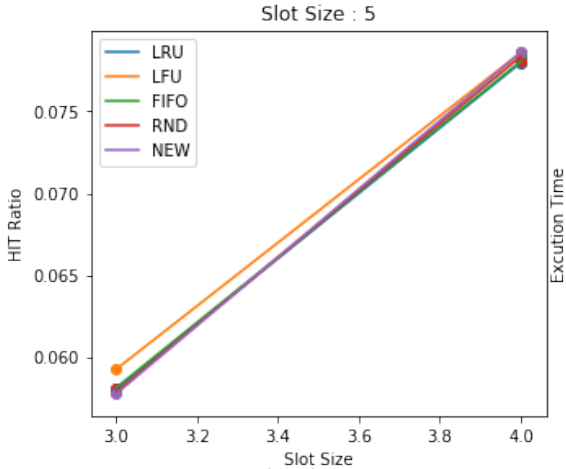
```
In [29]: visualize(input_size = 10000, input_max_size = 50)
```

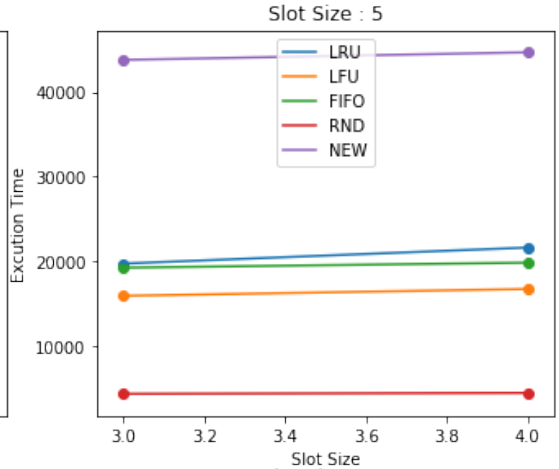
```
17 17
```

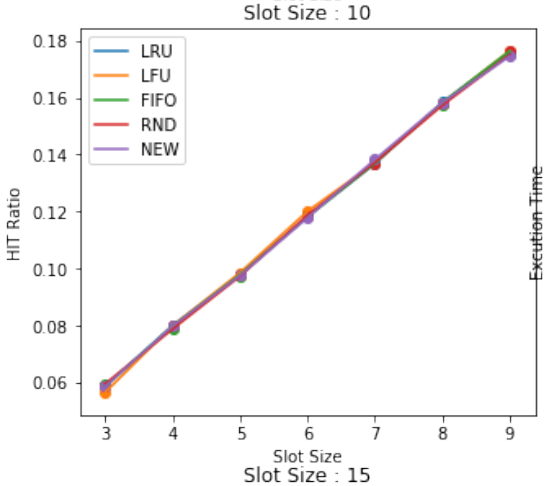
가능한 입력 수가 10000이며 난수의 최대 수가 50일 때

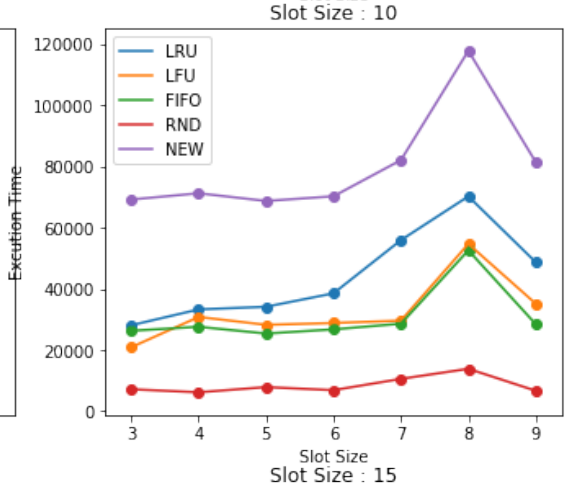
Out[29]:

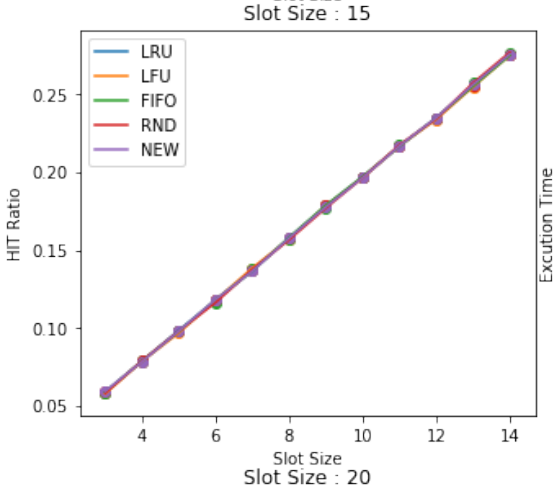
| | HIT | TIME |
|---|---|---|
| LRU / 25 | 0.07 | 20696.00 |
| LFU / 25 | 0.07 | 16348.70 |
| RND / 25 | 0.07 | 4430.40 |
| FIFO / 25 | 0.07 | 19561.40 |
| NEW / 25 | 0.07 | 44234.90 |
| LRU / 50 | 0.12 | 44105.26 |
| LFU / 50 | 0.12 | 32586.43 |
| RND / 50 | 0.12 | 8463.23 |
| FIFO / 50 | 0.12 | 30795.00 |
| NEW / 50 | 0.12 | 80058.09 |
| LRU / 75 | 0.17 | 35908.21 |
| LFU / 75 | 0.17 | 27900.55 |
| RND / 75 | 0.17 | 5657.35 |
| FIFO / 75 | 0.17 | 21876.16 |
| NEW / 75 | 0.17 | 59974.82 |
| LRU / 100 | 0.22 | 39489.58 |
| LFU / 100 | 0.22 | 31992.38 |
| RND / 100 | 0.22 | 5739.72 |
| FIFO / 100 | 0.22 | 20502.69 |
| NEW / 100 | 0.22 | 63722.76 |

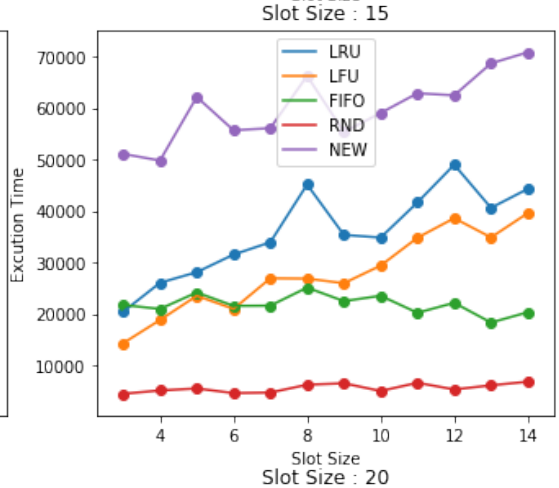

Slot Size : 5
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size

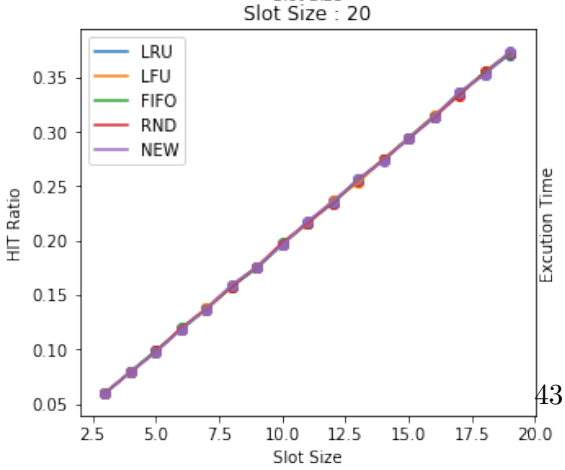

Slot Size : 5
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size

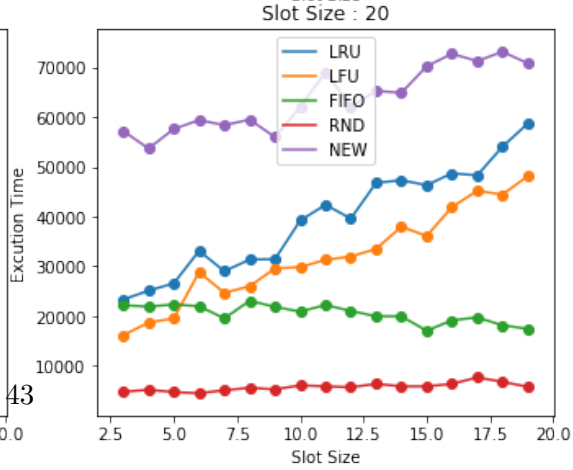

Slot Size : 10
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 10
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size


Slot Size : 15
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 15
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size


Slot Size : 20
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 20
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size

**난수의 최대 수가 75일 때** 슬롯 사이즈를 25,50,75,100으로 주어 적중률과 수행시간 비교

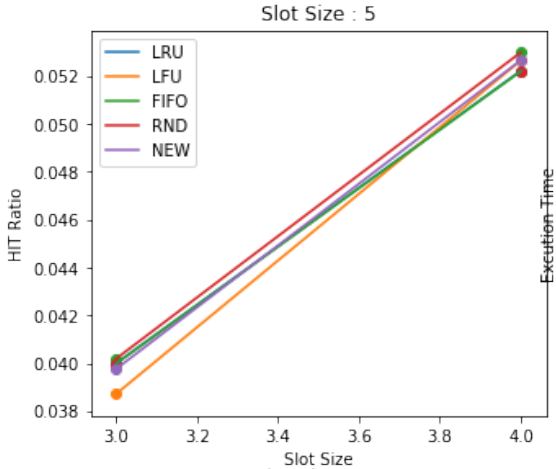
```
In [30]: visualize(input_size = 10000, input_max_size = 75)
```

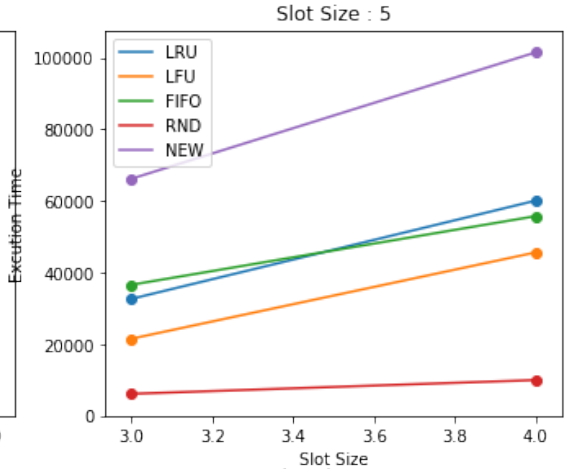
```
17 17
```

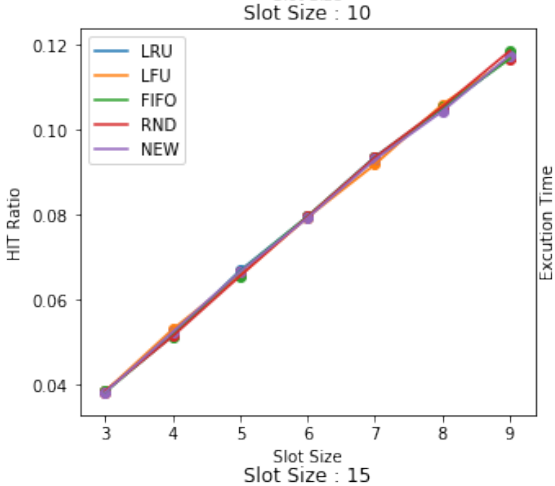
가능한 입력 수가 10000이며 난수의 최대 수가 75일 때

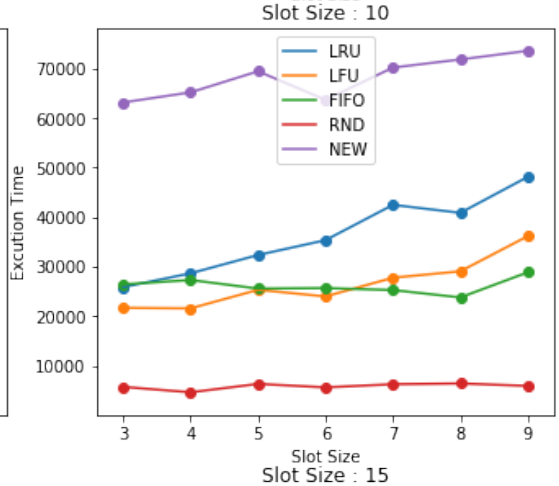
Out[30]:

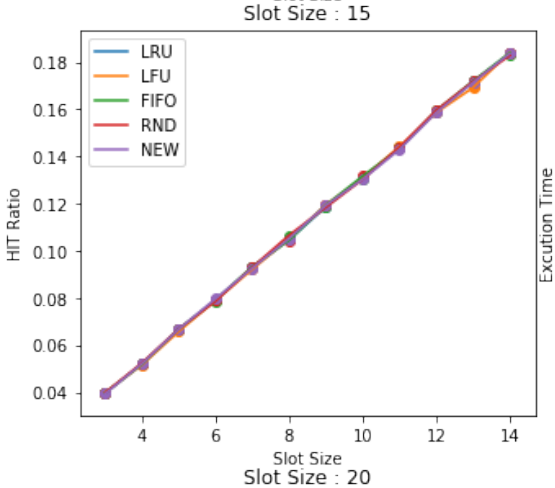
| | HIT | TIME |
|---|---|---|
| LRU / 25 | 0.05 | 46385.55 |
| LFU / 25 | 0.05 | 33598.10 |
| RND / 25 | 0.05 | 8103.80 |
| FIFO / 25 | 0.05 | 46188.00 |
| NEW / 25 | 0.05 | 83824.80 |
| LRU / 50 | 0.08 | 36203.97 |
| LFU / 50 | 0.08 | 26469.16 |
| RND / 50 | 0.08 | 5802.74 |
| FIFO / 50 | 0.08 | 26087.46 |
| NEW / 50 | 0.08 | 68120.47 |
| LRU / 75 | 0.11 | 42326.53 |
| LFU / 75 | 0.11 | 31004.48 |
| RND / 75 | 0.11 | 6477.01 |
| FIFO / 75 | 0.11 | 26827.40 |
| NEW / 75 | 0.11 | 72107.63 |
| LRU / 100 | 0.14 | 48093.12 |
| LFU / 100 | 0.14 | 35800.62 |
| RND / 100 | 0.14 | 6805.66 |
| FIFO / 100 | 0.14 | 26147.63 |
| NEW / 100 | 0.14 | 78203.34 |

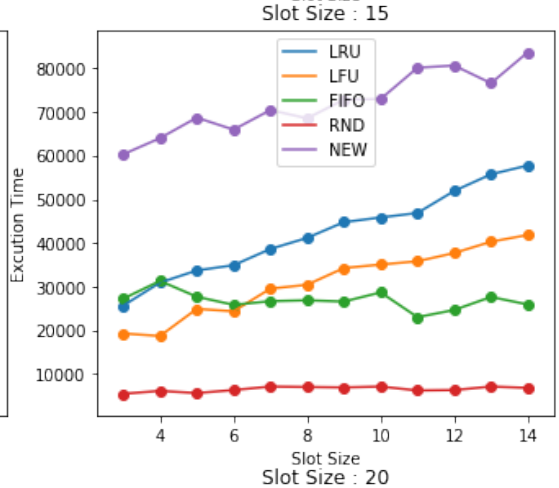

Slot Size : 5
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size

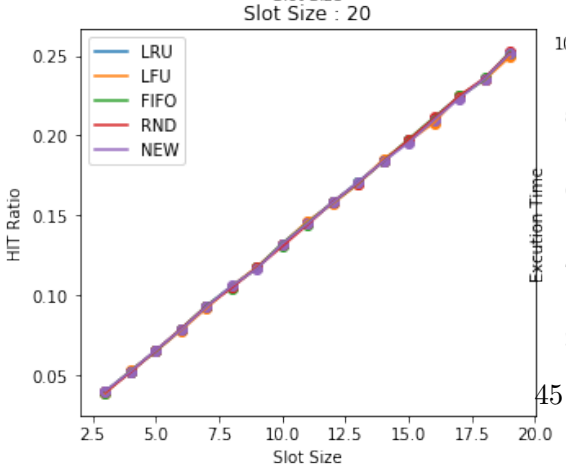

Slot Size : 5
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size

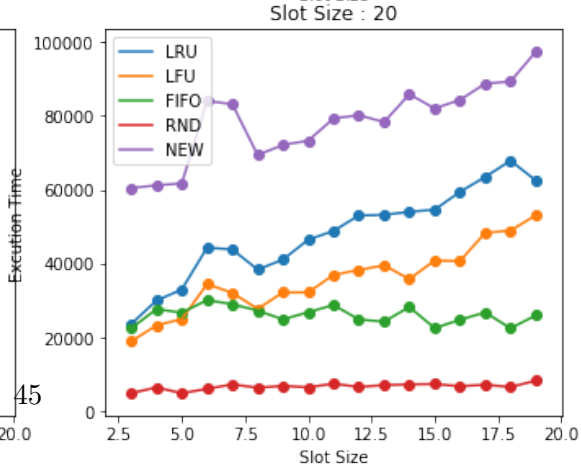

Slot Size : 10
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 10
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size


Slot Size : 15
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 15
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size


Slot Size : 20
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 20
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size

**난수의 최대 수가 100일 때** 슬롯 사이즈를 25,50,75,100으로 주어 적중률과 수행시간 비교

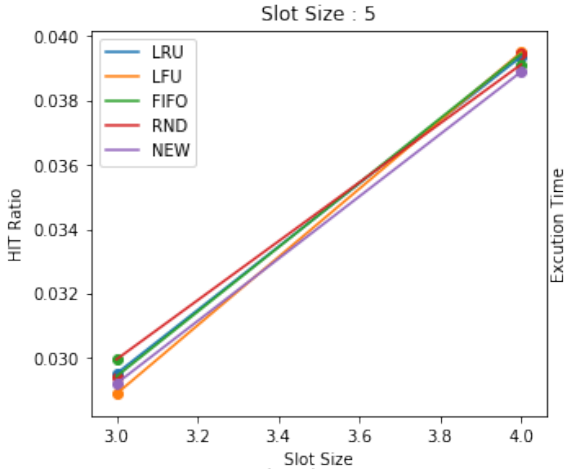
```
In [31]: visualize(input_size = 10000, input_max_size = 100)
```

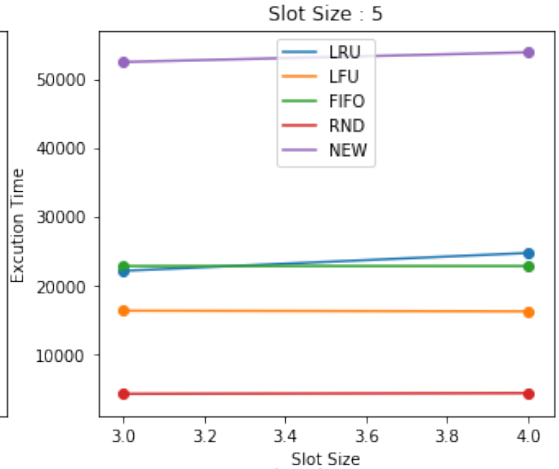
```
17 17
```

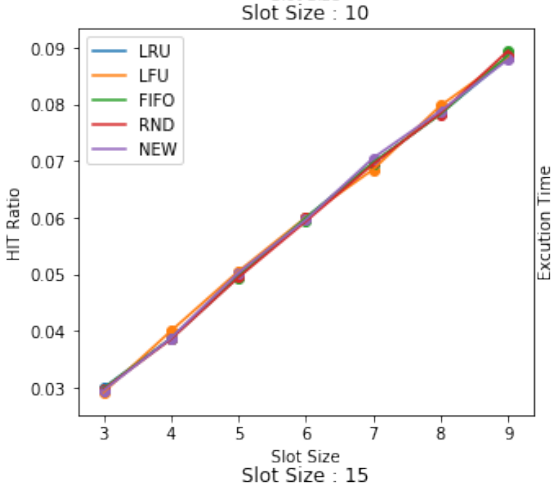
가능한 입력 수가 10000이며 난수의 최대 수가 100일 때

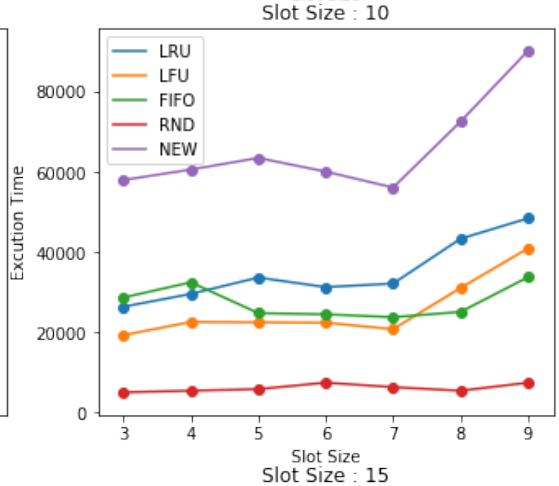
Out[31]:

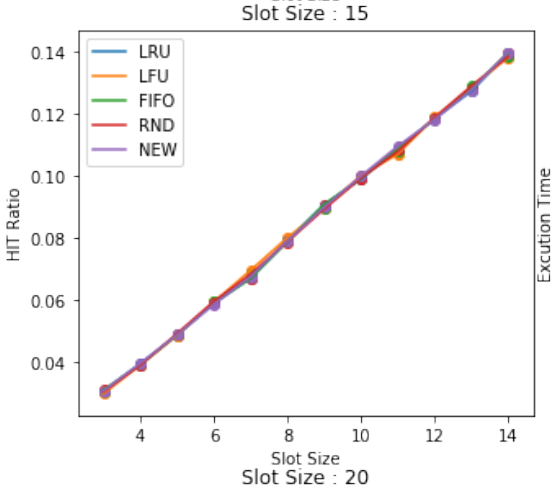
| | HIT | TIME |
|---|---|---|
| LRU / 25 | 0.03 | 23438.75 |
| LFU / 25 | 0.03 | 16306.95 |
| RND / 25 | 0.03 | 4336.90 |
| FIFO / 25 | 0.03 | 22839.65 |
| NEW / 25 | 0.03 | 53156.30 |
| LRU / 50 | 0.06 | 34921.86 |
| LFU / 50 | 0.06 | 25603.41 |
| RND / 50 | 0.06 | 6085.33 |
| FIFO / 50 | 0.06 | 27525.63 |
| NEW / 50 | 0.06 | 65808.43 |
| LRU / 75 | 0.08 | 42897.83 |
| LFU / 75 | 0.08 | 30243.13 |
| RND / 75 | 0.08 | 7039.27 |
| FIFO / 75 | 0.08 | 27435.57 |
| NEW / 75 | 0.08 | 75025.19 |
| LRU / 100 | 0.11 | 47465.28 |
| LFU / 100 | 0.11 | 34656.48 |
| RND / 100 | 0.11 | 7015.52 |
| FIFO / 100 | 0.11 | 26693.92 |
| NEW / 100 | 0.11 | 78828.87 |

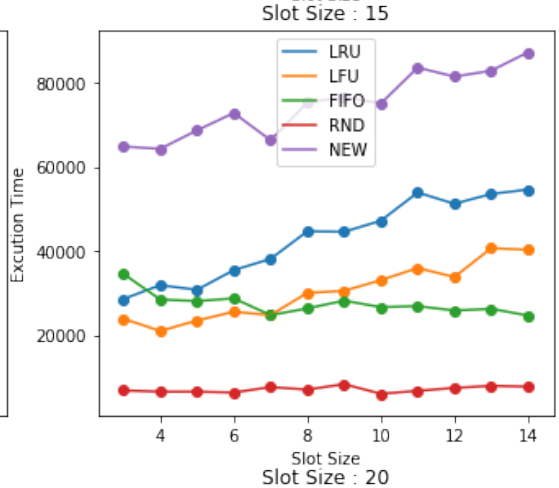

Slot Size : 5
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size

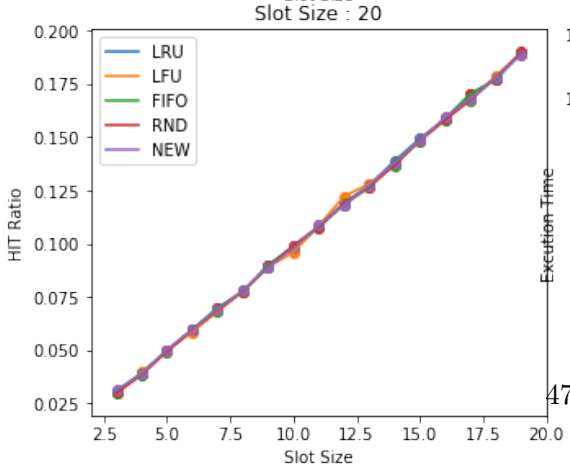

Slot Size : 5
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size

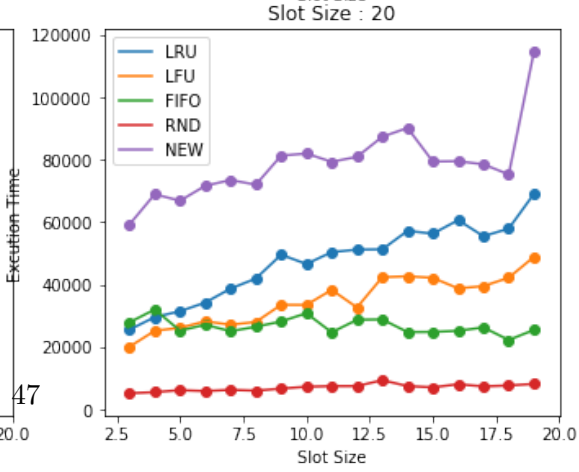

Slot Size : 10
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 10
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size


Slot Size : 15
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 15
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size


Slot Size : 20
LRU
LFU
FIFO
RND
NEW
HIT Ratio
Slot Size


Slot Size : 20
LRU
LFU
FIFO
RND
NEW
Excution Time
Slot Size

### 가능한 입력 수가 100000일 때

난수의 최대 수를 100, 슬롯 사이즈를 25,50,75,100으로 주어 적중률과 수행시간 비교

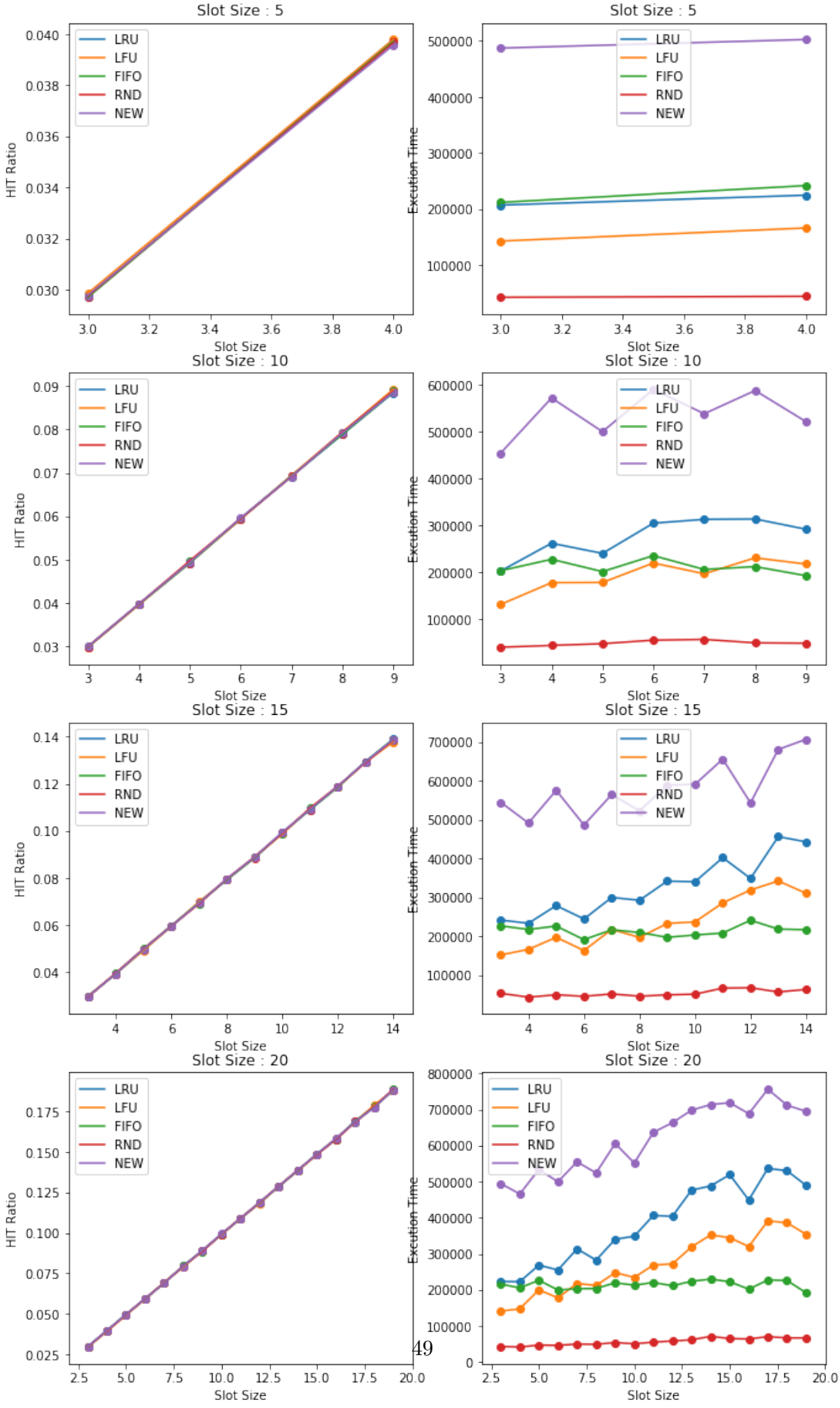
```
In [34]: visualize(input_size = 100000, input_max_size = 100)
```

17 17

가능한 입력 수가 100000이며 난수의 최대 수가 100일 때

Out[34]:

| | HIT | TIME |
|---|---|---|
| LRU / 25 | 0.03 | 216208.00 |
| LFU / 25 | 0.03 | 154901.50 |
| RND / 25 | 0.03 | 43843.30 |
| FIFO / 25 | 0.03 | 227155.90 |
| NEW / 25 | 0.03 | 494946.00 |
| LRU / 50 | 0.06 | 275809.00 |
| LFU / 50 | 0.06 | 193712.74 |
| RND / 50 | 0.06 | 49562.94 |
| FIFO / 50 | 0.06 | 211718.67 |
| NEW / 50 | 0.06 | 537585.57 |
| LRU / 75 | 0.08 | 326912.35 |
| LFU / 75 | 0.08 | 235124.07 |
| RND / 75 | 0.08 | 53684.49 |
| FIFO / 75 | 0.08 | 214471.74 |
| NEW / 75 | 0.08 | 579135.93 |
| LRU / 100 | 0.11 | 385702.09 |
| LFU / 100 | 0.11 | 270046.23 |
| RND / 100 | 0.11 | 56888.62 |
| FIFO / 100 | 0.11 | 214562.88 |
| NEW / 100 | 0.11 | 618296.96 |


Slot Size : 5
HIT Ratio
Slot Size
Slot Size : 5
Execution Time
Slot Size
Slot Size : 10
HIT Ratio
Slot Size
Slot Size : 10
Execution Time
Slot Size
Slot Size : 15
HIT Ratio
Slot Size
Slot Size : 15
Execution Time
Slot Size
Slot Size : 20
HIT Ratio
Slot Size
Slot Size : 20
Execution Time
Slot Size
LRU
LFU
FIFO
RND
NEW

**비교 분석**

**데이터 개수에 따른 적중률과 실행 시간 비교 분석**

- **데이터 개수는 입력되는 수가 10,100,1000,10000,100000 인 경우에 대해 분석**
- **적중률 분석**
  - 슬롯 개수에 따라 각 알고리즘의 적중률 차이가 슬롯 개수 에 따라 편차가 있었지만, 데이터 입력수가 커질수록 각 알고리즘 사이의 적중률이 슬롯 사이즈에 따른 일정한 값으로 수렴하는 양상을 보인다. -적중률 알고리즘에 상관없이 거의 비슷.
  - 실제 컴퓨터에서는 데이터 갯수가 거의 무한에 가까울 것이므로 실제 컴퓨터에서도 컴퓨터가 계속 작동할 수록 교체 알고리즘의 따른 적중률의 차이가 크지 않을 것으로 예상되어진다.
- **실행 시간 분석**
  - 일단 실행 시간에 캐시 교체 알고리즘에 따른 연산, 알고리즘에 따른 입력해야하는 가짓수 등의 캐시 교체 알고리즘 자체요인의 시간도 있겠지만 교체 알고리즘 따라 자료구조, 탐색 방식, 등에 따른 구현상의 차이에 따른 시간도 있어 실제 캐시 작동시간과는 차이가 있을 것이다.
  - 데이터 개수가 늘어남에따라 알고리즘에 따른 차이가 명확해 진다. RND - FIFO - LFU - LRU - NEW의 순서로 시간이 덜 걸린다.

**입력 패턴에 따른 적중률과 실행 시간 비교 분석**

- 입력 패턴은 난수의 범위를 다르게 함으로써 다양화하였음

  **적중률 관련 분석**

- 난수의 최댓값이 높을수록(난수의 범위가 넓을수록) 적중률이 낮아짐. 난수의 최댓값과 적중률은 반비례한다고 봐도 될 정도. 이 이유는 데이터 값의 범위가 넓어져서, 지역성을 만족할 확률도 낮아지기 때문으로 보임

- 난수의 최댓값에 상관없이 알고리즘들의 적중률 차이는 동일, 하지만 그렇게 크지 않음. (거의 적중률이 동일하다고 봐도 될 정도)

- 미세하게 적중률이 차이나는 부분을 보고 알고리즘별 순위를 매기려고 시도했으나 case마다 순위가 제각각이였음. 입력 패턴에 따라 적중률에서 뚜렷하게 강세를 보이는 알고리즘은 딱히 없어 보임

  **실행 시간 관련 분석**

- 실행 시간은 난수의 최댓값(난수의 범위)와 관련 없이 제각각임. 그 이유는 데이터에 들어갈 난수를 호출하는 것은 알고리즘의 구동 시간과 별개이기 때문인것으로 보임
- 각 알고리즘의 실행 시간 순위는 난수의 최댓값에 관계 없이 일정. (RND - FIFO - LFU - LRU - NEW 순) 난수의 최댓값보다는, 알고리즘 자체 실행 시간만이 영향을 준 것으로 보임

**슬롯 개수에 따른 적중률과 실행 시간 비교 분석**

- **적중률 관련 분석**
  - 슬롯의 개수가 증가함에 따라 적중률이 조금씩 상승함을 알 수 있다. 슬롯의 개수가 많아지면 적중률이 올라감은 지극히 당연한 현상이다. 슬롯의 개수가 늘어난다는 것은 적중할 수 있는 후보가 많아진다는 뜻이기 때문이다.
- **실행 시간 관련 분석**
  - 실행시간은 슬롯의 개수가 많아짐에 따라 증가하는 추세를 보였다. 알고리즘 상에서 적절한 캐시를 찾을 때의 작업의 양이 증가하여 생기는 현상으로 파악된다.

### 1.4 토의 사항

1. 이 프로젝트에서 주어진 데이터는 정수값밖에 없어 데이터의 크기가 일정했지만, 실제로는 데이터의 사이즈가 제각각일 가능성이 높다. 캐시의 용량은 제한적이므로, 캐시는 여러개의 작은 사이즈를 가진 데이터들을 저장하기 위해 하나의 큰 사이즈를 가진 데이터를 버리는 것이 더 효율적일 수도 있다. 실제로 사용할 알고리즘에선 이 점을 고려해야 할 것이다.
2. 웹 브라우저 캐시와 같이 일정 시간이 지나면 쓸모 없게 되는 데이터는, 교체할 때 유효 시간도 반영을 해 주어야 한다.
3. 이 프로젝트에서 주어진 데이터는 정수값밖에 없어 코스트가 일정했지만, 실제로는 여러 종류의 데이터가 주어질 가능성이 높다. 개중에는 코스트가 커서 받아들이기에 많은 시간이 걸리는 데이터도 존재할 것이다. 이럴 경우에는 한번 캐시에 들어왔을 경우, 나중에 캐시 미스가 나더라도 교체하지 않는 것이 더 이득일 수 있다. 실제로 사용할 알고리즘에선 이 점을 고려해야 할 것이다.
4. LinkedList와 Array
   - 원소 값들의 위치가 연속이 되어 있지 않은 LinkedList의 경우 원소 값들의 위치가 연속 되어 있는 Array에 비해 연결되어 있는 다음 원소를 접근 할때 캐시 미스가 날 확률이 더 높을 것이다.
5. 데이터입력 개수, 데이터범위, 슬롯 갯수가 어느정도에 이르면, 5가지의 알고리즘 모두 적중률은 비슷해 지지만 실행 속도가 차이나는 이유는 무엇일까
   - RND의 경우 교체, 적재시에 데이터 값을 제외하고 추가로 입력해야하는 값이 없고 , 슬롯마다 추가 공간이 필요하지도, 추가적인 변수가 필요하지도 않아 가장 빨리 걸린 것으로 예상

- FIFO의 경우 적재시에 데이터 값을 제외하고 추가적으로 입력되는 값이 없고 , 가장 오래된 캐시 슬롯의 구별을 위해 변수가 필요하긴 하지만, 슬롯마다 추가공간이 필요하지 않아 2번째 인것으로 보인다.
- LFU의 경우
- LRU의 경우
- NEW의 경우

### 1.5 기여도

- 조혜진 : [25%] LFU 구현, 보고서 문제 정의 작성, 함수 및 변수 LFU 부분 작성, 입력 패턴에 따른 적중률&실행 시간 비교 분석 작성
- 박종현 : [25%] 임의 교체 방식 구현, 임의 교체 방식 알고리즘 설명 부분 보고서 작성, 알고리즘별 적중률&실행 시간 비교 그래프 출력 구현, 실행 결과 보고서(3번) 작성
- 박인우 : [25%] LRU 함수 작성, 새로운 알고리즘 제안 및 NEW 함수 작성, LRU 및 NEW 함수 설명 부분 보고서 작성, 슬롯개수에 따른 적중률&실행시간 비교 분석 작성
- 조윤직 : [25%] FIFO 함수 작성, FIFO 함수 설명 부분 보고서 작성, 데이터 갯수 차이에 따른 적중률&실행시간 분석 작성

In [ ]:

In [ ]: