Page Replacement Algorithms

นายกฤษณะ มะนุภา

รหัส 600612147 Section 801

รายงานกระบวนวิชา Operating System

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่

ภาคการศึกษาที่ 2/2562

First In First Out (FIFO)

แนวคิด

หลักการของ Algorithm นี้คือการวนใช้ Index ใน Frame ตามลำดับ จะเริ่มใหม่ก็ต่อเมื่อวนไปจนถึง Index ท้ายสุด จึงได้มีการกำหนดตัวแปร framePointer มาทำหน้าที่ระบุ Index ที่จะให้ Reference strings มาเพิ่มหรือแทนที่ ณ Index นั้น ๆ ในส่วนต่อไปจำเป็นที่จะต้องมีตัวเก็บค่านับจำนวนเพื่อแสดง Hit หรือ PageFault ได้ในแต่ละ Index คือตัวแปร pageFaultCountList โดยเก็บเป็น List และนำมาใช้ฟังก์ชัน Counter เพื่อนับรวม

CODE

```
15  def FifoReplace(refSTR, frameSize):
16
17      framePointer = 0
18
19      stringLen = len(refSTR)
20
21      pageFault = 0
22      Hit = 0
23
24      frame = [None]*frameSize
25      pageFaultCountList = []
26
27      for i in range(stringLen):
28
29          if framePointer >= frameSize:
30              framePointer = 0
31
32          if refSTR[i] not in frame:
33
34              frame[framePointer] = refSTR[i]
35              pageFaultCountList.append(framePointer+1)
36              framePointer += 1
37              pageFault += 1
38
39          else:
40              Hit += 1
41
42      FaultResult = Counter(pageFaultCountList)
43
44      print(frame, ' Frame size is ', frameSize)
45      print('Hit Count is ', Hit)
46      print('PageFault count is ', pageFault)
47      print(sorted(FaultResult.items()))
48
49      print("\n")
50
```

Optimal

แนวคิด

เนื่องจาก Optimal Algorithm นี้จะเกี่ยวเนื่องกับการจดจำ Index ภายใน Frame จึงได้สร้างตัวแปร indexCheck ตามจำนวน frameSize ขึ้นมา ภายใน Function ของ Algorithm นี้ภายในจะแบ่งเป็น 2 ส่วนใหญ่ ๆ ในส่วนแรกหมายถึงหาก Frame ยังไม่เต็ม ในส่วนที่สองหมายถึง เมื่อ Frame เต็มแล้ว ซึ่งในสองส่วนนี้ จะมีเงื่อนไขย่อยอีก คือ ยังไม่เคยมี String นั้น ๆ ใน Frame หรือมี String นั้น ๆ อยู่ใน Frame อยู่แล้ว ในการทำงาน indexCheck จะเป็น List ซึ่งเก็บตัวเลขจำนวนเต็มอยู่ตรงตาม Index ของ Frame แต่ละรอบทุก ๆ ค่าใน indexCheck จะเพิ่มขึ้น 1 ยกเว้น ณ Index ที่มีการเพิ่มเข้ามาใน Frame ล่าสุด หรือมีการ Hit จะถูกกำหนดค่าใน indexCheck ให้เป็น 0 หาก Frame เต็ม จะทำการเปรียบเทียบจาก indexCheck จาก Index ไหนมากที่สุด หมายความว่า Index นั้น ไม่ได้ถูกใช้งานนานแล้ว ส่งผลให้จะถูกแทนที่ด้วย String ชุดที่เข้ามาใหม่

CODE

```python
def OptReplace(refSTR, frameSize):

    framePointer = 0

    stringLen = len(refSTR)

    pageFault = 0
    Hit = 0
    mostTime = 0
    mostTimeIDX = 0
    returnMatchIDX = 0

    frame = [None]*frameSize
    pageFaultCountList = []
    indexCheck = [0]*frameSize

    for i in range(stringLen):

        if framePointer < frameSize:
            if refSTR[i] not in frame:
                pageFault += 1
                frame[framePointer] = refSTR[i]
                pageFaultCountList.append(framePointer+1)
                indexCheck[framePointer] = 0
                framePointer += 1

                for idx, value in enumerate(indexCheck):
                    if idx != framePointer:
                        indexCheck[idx] += 1

            else:
                Hit += 1
                indexCheck[framePointer] = 0
                framePointer += 1
```

```python
                for idx, value in enumerate(indexCheck):
                    if idx != framePointer:
                        indexCheck[idx] += 1
        else:
            # Find Index not use most time before append to the list
            mostTime = 0
            for l, value in enumerate(indexCheck):
                if value > mostTime:
                    mostTime = value
                    mostTimeIDX = l

            if refSTR[i] not in frame:

                frame[mostTimeIDX] = refSTR[i]
                pageFaultCountList.append(mostTimeIDX+1)
                pageFault += 1
                framePointer += 1
                indexCheck[mostTimeIDX] = 0

                for idx, value in enumerate(indexCheck):
                    if idx != mostTimeIDX:
                        indexCheck[idx] += 1
            else:
                Hit += 1

                for k, value in enumerate(frame):
                    if value == refSTR[i]:
                        returnMatchIDX = k

                frame[returnMatchIDX] = refSTR[i]
                indexCheck[returnMatchIDX] = 0
                for idx, value in enumerate(indexCheck):
                    if idx != returnMatchIDX:
                        indexCheck[idx] += 1
```

Least recently used (LRU)

แนวคิด

เนื่องจาก LRU Algorithm เป็นการโฟกัสไปที่ Queue ของ String ใน Frame จึงได้กำหนดตัวแปร tmpQueue ขึ้นมาเป็น Type List สาเหตุที่ไม่ใช้ Type Queue โดยตรง เพราะ Queue ไม่สามารถนำค่าที่อยู่ระหว่าง Queue ออกมาได้ เพื่อสะดวกต่อการใช้งานจึงเลือก List เป็นเสมือน Queue แทน ฟังก์ชันนี้ภายใน Loop มีสองส่วนใหญ่ ๆ ส่วนแรกคือ เมื่อ Frame ยังไม่เต็ม ส่วนที่สอง เมื่อ Frame เต็มแล้ว และส่วนย่อยในสองส่วนหลัก ๆ นี้ ประกอบไปด้วย เงื่อนไขหากไม่มี String นั้น ๆ อยู่ใน Frame และ เงื่อนไขที่เมื่อมี String นั้น ๆ คงอยู่ใน Frame อยู่แล้ว

ในการทำงานเมื่อ Frame ยังไม่เต็ม และไม่เคยมี String อยู่ใน Frame จะส่งค่า String ไปเก็บไว้ทั้งใน Frame และ tmpQueue หาก String ไหนมีอยู่ใน Frame ไม่ว่าจะอยู่ Index ไหนก็ตาม จะทำการกำหนดให้ ค่าของ String นั้น ๆ เป็น None และใช้ฟังก์ชัน filter ช่วยในการลบ None ออกจากใน Queue ให้เรียงตามลำดับติดกันไม่ถูกแทรกด้วย None (ขยับ Queue มาติดกัน) และ ทำการเพิ่มค่า String นั้น ๆ มาต่อท้ายใหม่ เสมือนการ Reset สถานะจดจำว่า String ค่านี้พึ่งมีการใช้งานใน Frame ล่าสุด

ในการนำไปใช้กำหนดค่าใน Frame นั้นได้จากการนำ String ตัวแรกของ tmpQueue มาตรวจเงื่อนไขใน Frame ให้มี Value ตรงกัน เพื่อที่จะนำ Index ของ String Value ใน Frame นั้น ๆ ถูกแทนที่ด้วย String ใหม่ ณ ตำแหน่ง Index ตามเงื่อนไข

CODE

```python
def LRUReplace(refSTR, frameSize):

    framePointer = 0

    stringLen = len(refSTR)

    pageFault = 0
    Hit = 0
    LeastuseIDX = 0

    frame = [None]*frameSize
    pageFaultCountList = []
    tmpQueue = []

    for i in range(stringLen):
        if framePointer < frameSize:
            if refSTR[i] not in frame:
                pageFault += 1

                frame[framePointer] = refSTR[i]  # เพิ่มปกติ
                pageFaultCountList.append(framePointer+1)
                tmpQueue.append(refSTR[i])  # เพิ่มใน Queue ปกติไม่มีการลบ
                framePointer += 1

            else:
                Hit += 1
                # สำหรับเอาตัวแรกออกจาก Queue และเติม refSTR ท้าย Queue
                for o, value in enumerate(frame):
                    if value == tmpQueue[0]:
                        LeastuseIDX = o
                        tmpQueue[0] = None

                        break
                # เพิ่มเข้าไปใน Queue
                tmpQueue = list(filter(None, tmpQueue))
                tmpQueue.append(refSTR[i])
```

```python

        # When framsize is full
        else:
            tmpQueue = list(filter(None, tmpQueue))
            if refSTR[i] not in frame:

                pageFault += 1
                # สำหรับเอาตัวแรกออกจาก Queue และเติม refSTR ท้าย Queue
                for o, value in enumerate(frame):
                    if value == tmpQueue[0]:
                        LeastuseIDX = o
                        tmpQueue[0] = None

                        break
                pageFaultCountList.append(LeastuseIDX+1)
                frame[LeastuseIDX] = refSTR[i]
                tmpQueue = list(filter(None, tmpQueue))
                tmpQueue.append(refSTR[i])
                framePointer += 1

            else:
                Hit += 1
            # สำหรับลบออกจาก Queue list
                for o, value in enumerate(tmpQueue):
                    if value == refSTR[i]:
                        tmpQueue[o] = None
                        break

            # เพิ่มตัวล่าสุดไปใน Queue
                tmpQueue = list(filter(None, tmpQueue))
                tmpQueue.append(refSTR[i])

    FaultResult = Counter(pageFaultCountList)

    print(frame, ' Frame size is ', frameSize)
    print('Hit Count is ', Hit)
    print('PageFault count is ', pageFault)
    print(sorted(FaultResult.items()))
```

การทดลอง

```
214
215   FifoReplace(refSTR, 3)
216   FifoReplace(refSTR, 5)
217   FifoReplace(refSTR, 8)
218
219   print("------------------O
220
221   print('Random (', len(refS
222   print('Reference strings :
223
224   OptReplace(refSTR, 3)
225   OptReplace(refSTR, 5)
226   OptReplace(refSTR, 8)
227
228   print("------------------L
229
230   print('Random (', len(refS
231   print('Reference strings :
232
233   LRUReplace(refSTR, 3)
234   LRUReplace(refSTR, 5)
235   LRUReplace(refSTR, 8)
236
```

ได้ทำการเรียกใช้ฟังก์ชันของ 3 Algorithm โดยแต่ละ Algorithm มีการทดลองใช้จำนวน Frame ที่แตกต่างกันคือ 3 , 5 และ 8 ตามลำดับ โดยมี Reference string ชุดเดียวกันจากฟังก์ชัน genString ส่งเข้าไปในทุก ๆ Algorithms

สมมุติฐาน

กำหนดสมมุติฐานในแต่ละ Algorithm คือ เนื่องจากในการทดลอง Reference strings ได้มีการกำหนดเกณฑ์ในการสุ่มไว้ สมมุติฐานว่า FIFO Algorithm เกิด Page fault มากที่สุด เนื่องจากการนำ reference strings เข้ามามีโอกาสที่จะเกิดเป็นหมายเลข Frame เดียวกันหลาย ๆ ตัวได้น้อย และจำนวนการเกิด Page Fault ที่ใกล้เคียงกันคือ Optimized และ LRU เนื่องจาก Algorithms ทั้งสองนี้ขึ้นอยู่กับ Reference String จึงแล้วแต่โอกาสที่ Reference String นั้นเข้ามาใน Frame ด้วย แต่ทั้งคู่ทำงานได้ไวกว่า FIFO Algorithm

Reference string

```
refSTR = GenString(120, 150, 1, 10)
print('Random (', len(refSTR), ') strings')
print('Reference strings : ', refSTR, '\n')
```

การ Generate reference strings จากฟังก์ชันนี้มี Parameters 4 ตัว ตัวแรกและตัวที่สอง หมายถึง ช่วงสุ่มของจำนวน strings ที่เริ่มต้นจาก 120 ถึง 150 strings Parameters ตัวที่สามและสี่ หมายถึง ค่าแต่ละ String มีโอกาสสุ่มได้ค่าช่วง 1 ถึง 10 เหตุผลที่สุ่มจำนวนระหว่างในช่วง 120 – 150 strings เพราะต้องการให้จำนวน strings มีมากกว่า ค่าของ string มาก ๆ เพื่อให้เกิดค่าของ string ที่ซ้ำกันบ่อย ๆ เพื่อทดสอบการทำงานของ Algorithms

ผลลัพธ์ของฟังก์ชันหลังจาก return reference strings

```
Random ( 147 ) strings
Reference strings :  [1, 2, 6, 10, 10, 4, 2, 3, 5, 5, 8, 1, 9, 9, 1, 5, 7, 8, 1, 9, 7, 2, 9, 10, 1, 2, 8, 7, 3, 2, 2, 2, 10, 3, 3, 8, 9, 8, 8, 5, 10, 3, 6, 2, 3, 6, 5
, 7, 2, 10, 4, 2, 5, 10, 8, 5, 8, 4, 6, 3, 6, 5, 9, 6, 4, 9, 8, 5, 5, 7, 5, 1, 2, 1, 9, 3, 7, 4, 1, 9, 2, 5, 3, 3, 2, 8, 10, 7, 4, 8, 10, 1, 1, 4, 8, 9, 4, 6, 7, 4, 4
, 2, 5, 8, 1, 7, 4, 5, 7, 1, 9, 10, 4, 5, 7, 9, 3, 1, 2, 8, 1, 8, 8, 1, 6, 1, 4, 8, 10, 10, 5, 8, 1, 1, 6, 5, 7, 2, 10, 6, 9, 3, 8, 2, 9, 1, 6]
```

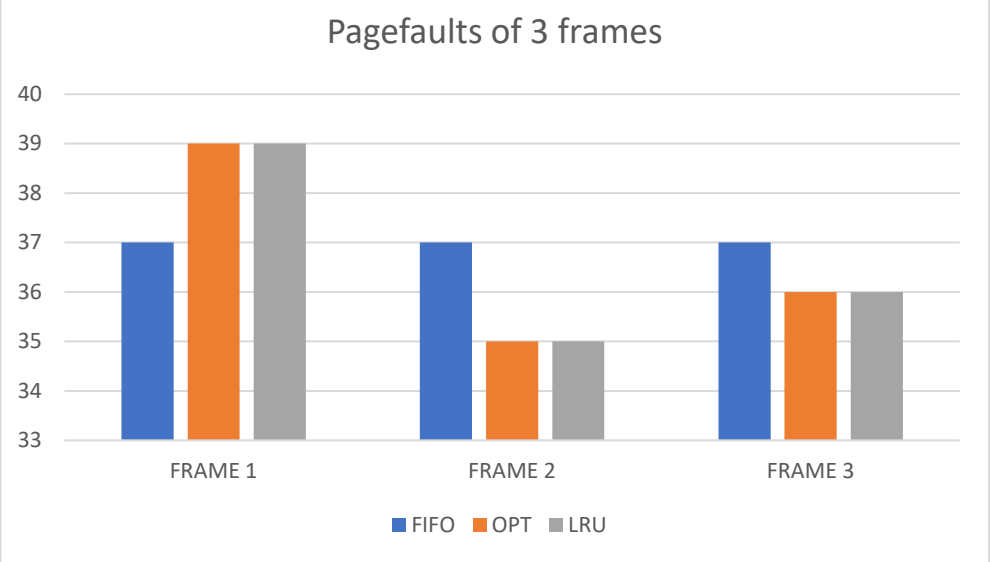
ประเมินผลการทดลอง

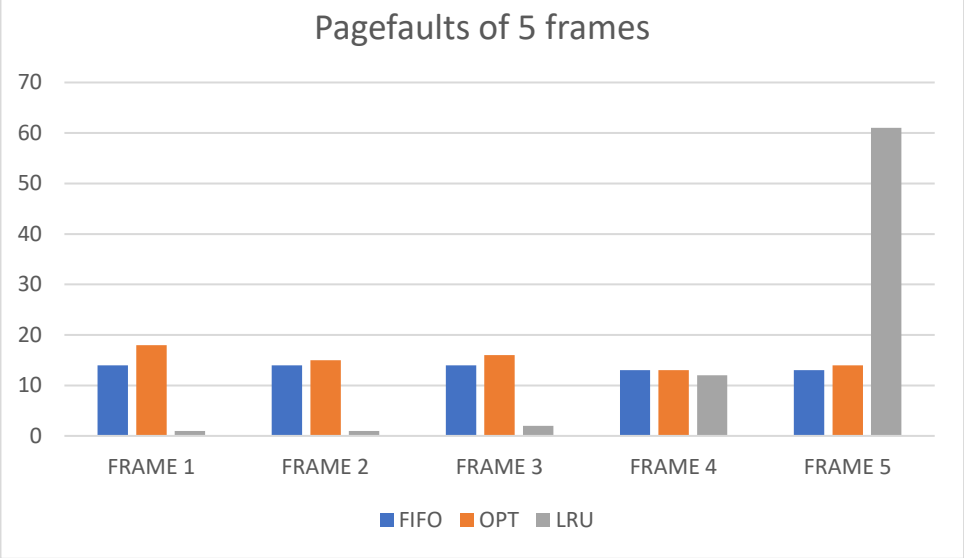
```
ENT\ReplacementALG.py"
------------------FIFO Function has been called------------------

Random ( 147 ) strings
Reference strings :  [1, 2, 6, 10, 10, 4, 2, 3, 5, 5, 8, 1, 9, 9, 1, 5, 7, 8, 1, 9, 7, 2, 9, 10, 1, 2, 8, 7, 3, 2, 2, 2, 10, 3, 3, 8, 9, 8, 8, 5, 10, 3, 6, 2, 3, 6, 5
, 7, 2, 10, 4, 2, 5, 10, 8, 5, 8, 4, 6, 3, 6, 5, 9, 6, 4, 9, 8, 5, 5, 7, 5, 1, 2, 1, 9, 3, 7, 4, 1, 9, 2, 5, 3, 3, 2, 8, 10, 7, 4, 8, 10, 1, 1, 4, 8, 9, 4, 6, 7, 4, 4
, 2, 5, 8, 1, 7, 4, 5, 7, 1, 9, 10, 4, 5, 7, 9, 3, 1, 2, 8, 1, 8, 8, 1, 6, 1, 4, 8, 10, 10, 5, 8, 1, 1, 6, 5, 7, 2, 10, 6, 9, 3, 8, 2, 9, 1, 6]

[9, 1, 6]  Frame size is  3
Hit Count is  36
PageFault count is  111
[(1, 37), (2, 37), (3, 37)]


[3, 8, 1, 6, 9]  Frame size is  5
Hit Count is  79
PageFault count is  68
[(1, 14), (2, 14), (3, 14), (4, 13), (5, 13)]


[6, 4, 3, 8, 7, 9, 2, 1]  Frame size is  8
Hit Count is  114
PageFault count is  33
[(1, 5), (2, 4), (3, 4), (4, 4), (5, 4), (6, 4), (7, 4), (8, 4)]
```

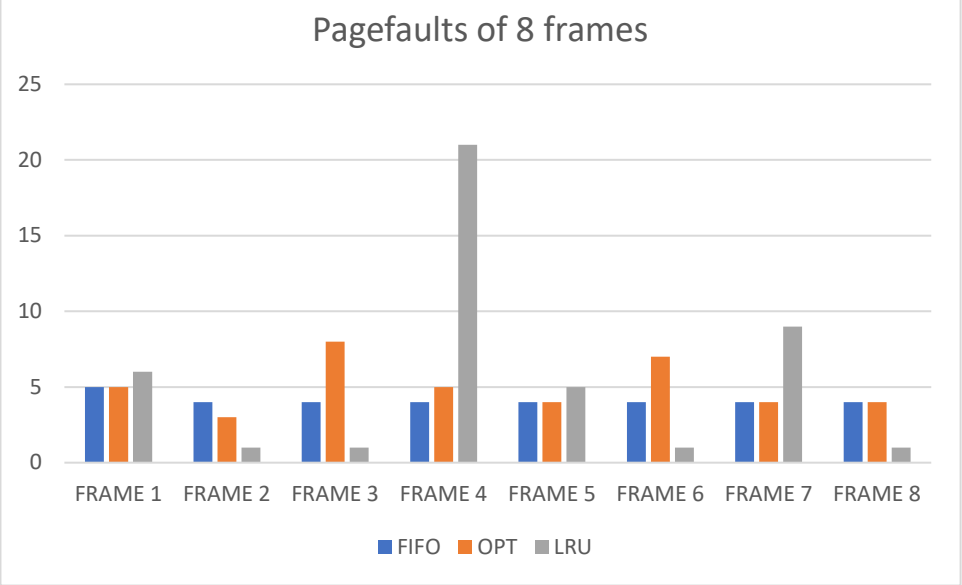
```
------------------Optimize Function has been called------------------

Random ( 147 ) strings
Reference strings :  [1, 2, 6, 10, 10, 4, 2, 3, 5, 5, 8, 1, 9, 9, 1, 5, 7, 8, 1, 9, 7, 2, 9, 10, 1, 2, 8, 7, 3, 2, 2, 2, 10, 3, 3, 8, 9, 8, 8, 5, 10, 3, 6, 2, 3, 6, 5
, 7, 2, 10, 4, 2, 5, 10, 8, 5, 8, 4, 6, 3, 6, 5, 9, 6, 4, 9, 8, 5, 5, 7, 5, 1, 2, 1, 9, 3, 7, 4, 1, 9, 2, 5, 3, 3, 2, 8, 10, 7, 4, 8, 10, 1, 1, 4, 8, 9, 4, 6, 7, 4, 4
, 2, 5, 8, 1, 7, 4, 5, 7, 1, 9, 10, 4, 5, 7, 9, 3, 1, 2, 8, 1, 8, 8, 1, 6, 1, 4, 8, 10, 10, 5, 8, 1, 1, 6, 5, 7, 2, 10, 6, 9, 3, 8, 2, 9, 1, 6]

[6, 1, 9]  Frame size is  3
Hit Count is  37
PageFault count is  110
[(1, 39), (2, 35), (3, 36)]


[6, 1, 8, 2, 9]  Frame size is  5
Hit Count is  71
PageFault count is  76
[(1, 18), (2, 15), (3, 16), (4, 13), (5, 14)]


[1, 2, 3, 10, 8, 9, 6, 7]  Frame size is  8
Hit Count is  107
PageFault count is  40
[(1, 5), (2, 3), (3, 8), (4, 5), (5, 4), (6, 7), (7, 4), (8, 4)]
```

```
------------------Least Recently Use Function has been called------------------

Random ( 147 ) strings
Reference strings :  [1, 2, 6, 10, 10, 4, 2, 3, 5, 5, 8, 1, 9, 9, 1, 5, 7, 8, 1, 9, 7, 2, 9, 10, 1, 2, 8, 7, 3, 2, 2, 2, 10, 3, 3, 8, 9, 8, 8, 5, 10, 3, 6, 2, 3, 6, 5
, 7, 2, 10, 4, 2, 5, 10, 8, 5, 8, 4, 6, 3, 6, 5, 9, 6, 4, 9, 8, 5, 5, 7, 5, 1, 2, 1, 9, 3, 7, 4, 1, 9, 2, 5, 3, 3, 2, 8, 10, 7, 4, 8, 10, 1, 1, 4, 8, 9, 4, 6, 7, 4, 4
, 2, 5, 8, 1, 7, 4, 5, 7, 1, 9, 10, 4, 5, 7, 9, 3, 1, 2, 8, 1, 8, 8, 1, 6, 1, 4, 8, 10, 10, 5, 8, 1, 1, 6, 5, 7, 2, 10, 6, 9, 3, 8, 2, 9, 1, 6]

[6, 1, 9]  Frame size is  3
Hit Count is  37
PageFault count is  110
[(1, 39), (2, 35), (3, 36)]


[1, 2, 3, 8, 6]  Frame size is  5
Hit Count is  70
PageFault count is  77
[(1, 1), (2, 1), (3, 2), (4, 12), (5, 61)]


[1, 2, 6, 9, 7, 3, 4, 8]  Frame size is  8
Hit Count is  102
PageFault count is  45
[(1, 6), (2, 1), (3, 1), (4, 21), (5, 5), (6, 1), (7, 9), (8, 1)]
```

กราฟแสดงหมายเลขจำนวน Page fault ที่เกิดขึ้นในแต่ละหมายเลข Frame






Pagefaults of 8 frames
25
20
15
10
5
0
FRAME 1
FRAME 2
FRAME 3
FRAME 4
FRAME 5
FRAME 6
FRAME 7
FRAME 8
FIFO
OPT
LRU

สรุปผลการทดลอง

จากการทดลองแต่ละ Algorithms ที่มีจำนวน Frame แตกต่างกันได้ผลการทดลองดังนี้

3 Frame ทดลองกับ FIFO , Optimized และ Least recently used Algorithms ตามลำดับ เกิด Page Fault ภายใน Frame คือ 111, 110 และ 110 ตามลำดับ

5 Frame ทดลองกับ FIFO , Optimized และ Least recently used Algorithms ตามลำดับ เกิด Page Fault ภายใน Frame คือ 68, 76 และ 77 ตามลำดับ

8 Frame ทดลองกับ FIFO , Optimized และ Least recently used Algorithms ตามลำดับ เกิด Page Fault ภายใน Frame คือ 33, 40 และ 45 ตามลำดับ

จากกราฟพบว่า 3 Frame FIFO Algorithm มีโอกาสเกิด Page fault ที่คงที่ในทุก ๆ หมายเลข Frame แต่ Optimized และ Least recently used จะเกิดใน Frame หมายเลข 1 บ่อยกว่า Frame อื่น ๆ และลดทอนตามลำดับจนถึง Frame หมายเลข 3

จากกราฟ 5 Frame พบว่า FIFO และ Optimized Algorithm มีโอกาสเกิดในทุก ๆ หมายเลข Frame ที่โอกาสเกิด Page fault เท่า ๆ กัน แต่ Least recently used เกิด hit ได้บ่อยในหมายเลข Frame แรก ๆ จนถึงช่วง หมายเลข Frame หลัง ๆ ที่เริ่มเกิด Page Fault บ่อยมากขึ้นหลายเท่าตัว

จากกราฟ 8 Frame พบว่า FIFO ยังคงมีโอกาสเกิด Page fault ทุก ๆ หมายเลข Frame ที่เท่ากัน ส่วน Optimized และ Least recently used algorithm นั้นมีบางหมายเลข Frame ที่ hit บ่อยครั้ง และบางหมายเลข Frame ที่ Page fault บ่อยครั้ง ในอัตราส่วนระดับเดียวกัน

จึงสรุปได้ว่า หากข้อมูลมีกฎเกณฑ์หรือในลักษณะที่ทราบค่า Reference strings แต่ละค่าเกิดในโอกาสใกล้เคียงหรือพอ ๆ กัน FIFO algorithm เหมาะกับข้อมูลลักษณะนี้แต่หาก Reference strings ใด ๆ ที่มีอัตรา strings ที่โอกาสเกิดมากไปทางค่าหนึ่ง Optimized กับ Least recently used algorithms จะเหมาะกับข้อมูลลักษณะนี้ ซึ่งการเลือกใช้ทั้งสอง Algorithms นี้ ขึ้นอยู่กับลักษณะของ Reference strings นั้น ๆ จากการใช้งานจริง

ภาคผนวก

Library

```python
from collections import Counter
import random
```

Counter ใช้สำหรับการนับรวมแต่ละ Index ใน Frame ที่เกิด Page fault

Random ใช้สำหรับการ Generate ค่าต่าง ๆ เพื่อสร้าง Reference String

Code Generating String

```python
def GenString(StartCount, EndCount, FirstNum, LastNum):
    strRandlen = random.randint(StartCount, EndCount)
    tmp_refSTR = [None]*strRandlen

    for i in range(strRandlen):
        tmp_refSTR[i] = random.randint(FirstNum, LastNum)

    return tmp_refSTR
```

Tool

Python 3.7.4

GITHUB

https://github.com/guylaxy31/Replacement3ALG