Page and Frame Replacement Algorithms Program

โดย
ชื่อ วุฒิภัทร แสนไชย
รหัส 640610668

Computer Engineering
Chiang Mai University

คำแนะนำ: พัฒนาโปรแกรมของนักศึกษาเองเพื่อประเมินอัลกอริธึม Page and Frame Replacement แบบต่างๆ ให้ระบุเกณฑ์ที่ต้องการใช้เปรียบเทียบประสิทธิภาพของแต่ละอัลกอริทึม

แสดงโค้ดการเขียนโปรแกรมทั้งหมด ตามด้วยคำอธิบายสั้นๆ ของอัลกอริธึมทั้งสาม

The code and description of the First-In-First-Out (FIFO) algorithm (20 points)

ใช้ queue ในการเก็บข้อมูล แล้วใช้การ append ในการใส่ข้อมูลเพิ่มกับ pop สำหรับดึงข้อมูลที่เก่าที่สุดออก ทำให้สามารถดึงข้อมูลที่มาก่อนออกเพื่อสลับข้อมูลใหม่เข้าไป

```
def fifo(arr,maxf):
  t=[]
  fault=0
  for i in arr:
    if i not in t:
      if len(t)<maxf:
        t.append(i)
        print(i,"table:",t)
        fault=fault+1
      else:
        t.pop(0)
        t.append(i)
        print(i,"table:",t)
        fault=fault+1
    else:
        print(i)
  return t, fault
```

The code and description of the Optimal algorithm (20 points)

ต่อยอดมาจาก FIFO แต่แก้ไขขั้นตอนการ pop โดยใน Optimal Algorithm นี้จะทำการสร้างฟังก์ชันเพิ่มขึ้นมาเพื่อทำการเลือก pop สมาชิกที่ในอนาคตจะออกมาช้าที่สุด (Longest not used)

```
def getLongestNotUse(pTable,strArr):
  maxCount=0
  victim=0
  for i in pTable:
    ctn=0
    for j in strArr:
      if j!=i:
        ctn=ctn+1
      else:
        break
    if ctn>=maxCount:
      maxCount=ctn
```

```
      victim=i
  return victim

def opt(arr,maxf):
  t=[]
  fault=0
  for i in range(len(arr)):
    if arr[i] not in t:
      if len(t)<maxf:
        t.append(arr[i])
        print(arr[i],"table:",t)
        fault=fault+1
      else:
        currStr=arr[i:]
        vIdx=t.index(getLongestNotUse(t,currStr))
        t.pop(vIdx)
        t.append(arr[i])
        print(arr[i],"table:",t)
        fault=fault+1
    else:
      print(arr[i])
  return t, fault
```

The code and description of the Least Recently Used (LRU) algorithm (20 points)

ต่อยอดมาจาก Optimal Algorithm แต่แก้ไขขั้นตอนการ pop โดยใน Least Recently Used นี้จะทำการแก้ไขฟังก์ชัน `getLongestNotUse` เพื่อทำการเลือก pop สมาชิกที่ในที่ผ่านมาว่าตัวไหนมรอายุที่เยอะที่สุด(Most Aged)

```
def getMostNotUse(pTable,strArr):
  maxCount=0
  victim=0
  for i in pTable:
    ctn=0
    for j in strArr[::-1]:
      if j!=i:
        ctn=ctn+1
      else:
        break
    if ctn>=maxCount:
      maxCount=ctn
      victim=i
  return victim
```

```
def lru(arr,maxf):
  t=[]
  fault=0
  for i in range(len(arr)):
    if arr[i] not in t:
      if len(t)<maxf:
        t.append(arr[i])
        print(arr[i],"table:",t)
        fault=fault+1
      else:
        currStr=arr[:i+1]
        vIdx=t.index(getMostNotUse(t,currStr))
        t.pop(vIdx)
        t.append(arr[i])
        print(arr[i],"table:",t)
        fault=fault+1
    else:
      print(arr[i])
  return t, fault
```

Experiments

1. **ชุดข้อมูลแรก first reference string dataset (Original)** (5 points)

สิ่งที่แสดงด้านล่างนี้คือการสร้าง reference string ชุดแรกโดยการสุ่ม มีความยาวอย่างน้อย 30 pages

```
[5, 0, 4, 6, 1, 0, 0, 0, 5, 3, 6, 7, 1, 3, 1, 1, 1, 0, 2, 3, 7, 1, 7, 4,
7, 5, 1, 1, 4, 7]
```

อธิบายสมมติฐานที่นักศึกษาใช้ในการสร้าง reference string เช่นความหลากหลาย การถูกอ้างซ้ำเป็นต้น (5 points)

```
Randomly generated array:
Length: 30
Unique values: {0, 1, 2, 3, 4, 5, 6, 7}
Frequency of each value:
  - 0: 5 occurrences
  - 1: 8 occurrences
  - 2: 1 occurrences
  - 3: 3 occurrences
  - 4: 3 occurrences
  - 5: 3 occurrences
  - 6: 2 occurrences
  - 7: 5 occurrences
```

แสดงผลการทดลองที่ได้จากโปรแกรมที่พัฒนาขึ้นโดยใช้ First-In-First-Out (FIFO) algorithm กับชุดข้อมูลแรก (5 points)

```
5 table: [5]
0 table: [5, 0]
4 table: [5, 0, 4]
6 table: [0, 4, 6]
1 table: [4, 6, 1]
0 table: [6, 1, 0]
0
0
5 table: [1, 0, 5]
3 table: [0, 5, 3]
6 table: [5, 3, 6]
7 table: [3, 6, 7]
1 table: [6, 7, 1]
3 table: [7, 1, 3]
1
1
0 table: [1, 3, 0]
2 table: [3, 0, 2]
3
7 table: [0, 2, 7]
1 table: [2, 7, 1]
7
4 table: [7, 1, 4]
7
5 table: [1, 4, 5]
1
1
4
7 table: [4, 5, 7]
```

แสดงผลการทดลองที่ได้จากโปรแกรมที่พัฒนาขึ้นโดยใช้ Optimal algorithm กับชุดข้อมูลแรก (5 points)

```
5 table: [5]
0 table: [5, 0]
4 table: [5, 0, 4]
6 table: [5, 0, 6]
1 table: [5, 0, 1]
0
0
0
5
3 table: [0, 1, 3]
6 table: [1, 3, 6]
7 table: [1, 3, 7]
1
3
1
1
1
0 table: [3, 7, 0]
2 table: [3, 7, 2]
3
7
1 table: [3, 7, 1]
7
4 table: [7, 1, 4]
7
5 table: [1, 4, 5]
1
1
4
7 table: [1, 4, 7]
```

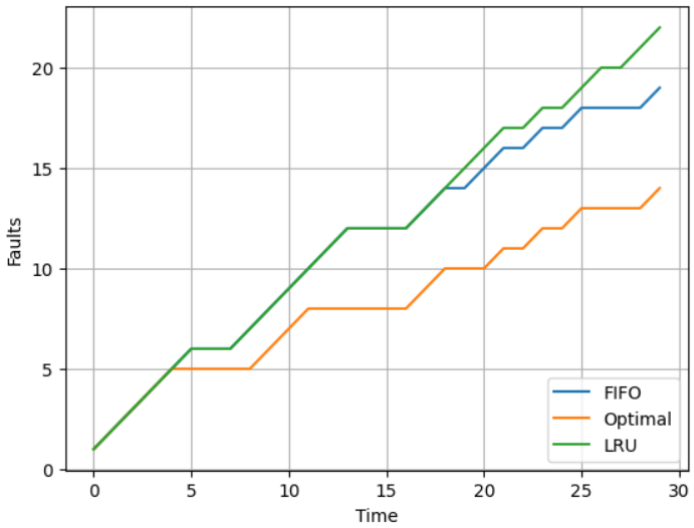
แสดงผลการทดลองที่ได้จากโปรแกรมที่พัฒนาขึ้นโดยใช้ Least Recently Used (LRU) algorithm กับชุดข้อมูลแรก (5 points)

```
5 table: [5]
0 table: [5, 0]
4 table: [5, 0, 4]
6 table: [0, 4, 6]
1 table: [4, 6, 1]
0 table: [6, 1, 0]
0
0
5 table: [1, 0, 5]
3 table: [0, 5, 3]
6 table: [5, 3, 6]
7 table: [3, 6, 7]
1 table: [6, 7, 1]
3 table: [7, 1, 3]
1
1
0 table: [1, 3, 0]
2 table: [1, 0, 2]
3 table: [0, 2, 3]
7 table: [2, 3, 7]
1 table: [3, 7, 1]
7
4 table: [7, 1, 4]
7
5 table: [7, 4, 5]
1 table: [7, 5, 1]
1
4 table: [5, 1, 4]
7 table: [1, 4, 7]
```

ตารางสรุป ผลการทดลอง

	FIFO	Optimal Alogorithm	LRU
Faults	19	14	22

Graph สรุปผลการทดลอง



2. **ชุดข้อมูลแรก second reference string dataset (Original)** (5 points)

สิ่งที่แสดงด้านล่างนี้คือการสร้าง reference string ชุดแรกโดยการสุ่ม มีความยาวอย่างน้อย 50 pages

```
[2, 0, 1, 2, 5, 3, 1, 2, 1, 3, 7, 0, 4, 4, 3, 3, 4, 7, 7, 2, 2, 2, 4, 4,
0, 2, 0, 5, 5, 1, 2, 7, 0, 0, 1, 7, 0, 1, 5, 3, 6, 7, 1, 6, 7, 4, 4, 2, 5,
1]
```

อธิบายสมมติฐานที่นักศึกษาใช้ในการสร้าง reference string เช่นความหลากหลาย การถูกอ้างซ้ำเป็นต้น (5 points)

```
Randomly generated array:
Length: 50
Unique values: {0, 1, 2, 3, 4, 5, 6, 7}
Frequency of each value:
  - 0: 7 occurrences
  - 1: 8 occurrences
  - 2: 9 occurrences
  - 3: 5 occurrences
  - 4: 7 occurrences
  - 5: 5 occurrences
  - 6: 2 occurrences
  - 7: 7 occurrences
```

แสดงผลการทดลองที่ได้จากโปรแกรมที่พัฒนาขึ้นโดยใช้ First-In-First-Out (FIFO) algorithm กับชุดข้อมูลที่ 2 (5 points)

```
2 table: [2]              1                        4 table: [7, 0, 4]
0 table: [2, 0]           2 table: [5, 3, 2]       4
1 table: [2, 0, 1]        1 table: [3, 2, 1]       3 table: [0, 4, 3]
2                         3                        3
5 table: [0, 1, 5]        7 table: [2, 1, 7]       4
3 table: [1, 5, 3]        0 table: [1, 7, 0]       7 table: [4, 3, 7]
```

```
7                       1 table: [0, 5, 1]      6 table: [5, 3, 6]
2 table: [3, 7, 2]      2 table: [5, 1, 2]      7 table: [3, 6, 7]
2                       7 table: [1, 2, 7]      1 table: [6, 7, 1]
2                       0 table: [2, 7, 0]      6
4 table: [7, 2, 4]      0                       7
4                       1 table: [7, 0, 1]      4 table: [7, 1, 4]
0 table: [2, 4, 0]      7                       4
2                       0                       2 table: [1, 4, 2]
0                       1                       5 table: [4, 2, 5]
5 table: [4, 0, 5]      5 table: [0, 1, 5]      1 table: [2, 5, 1]
5                       3 table: [1, 5, 3]
```

แสดงผลการทดลองที่ได้จากโปรแกรมที่พัฒนาขึ้นโดยใช้ Optimal algorithm กับชุดข้อมูลที่ 2 (5 points)

```
2 table: [2]            7                       1
0 table: [2, 0]         7                       7
1 table: [2, 0, 1]      2 table: [7, 4, 2]      0
2                       2                       1
5 table: [2, 1, 5]      2                       5 table: [7, 1, 5]
3 table: [2, 1, 3]      4                       3 table: [7, 1, 3]
1                       4                       6 table: [7, 1, 6]
2                       0 table: [7, 2, 0]      7
1                       2                       1
3                       0                       6
7 table: [2, 3, 7]      5 table: [7, 2, 5]      7
0 table: [3, 7, 0]      5                       4 table: [7, 1, 4]
4 table: [3, 7, 4]      1 table: [7, 2, 1]      4
4                       2                       2 table: [7, 1, 2]
3                       7                       5 table: [7, 1, 5]
3                       0 table: [7, 1, 0]      1
4                       0
```

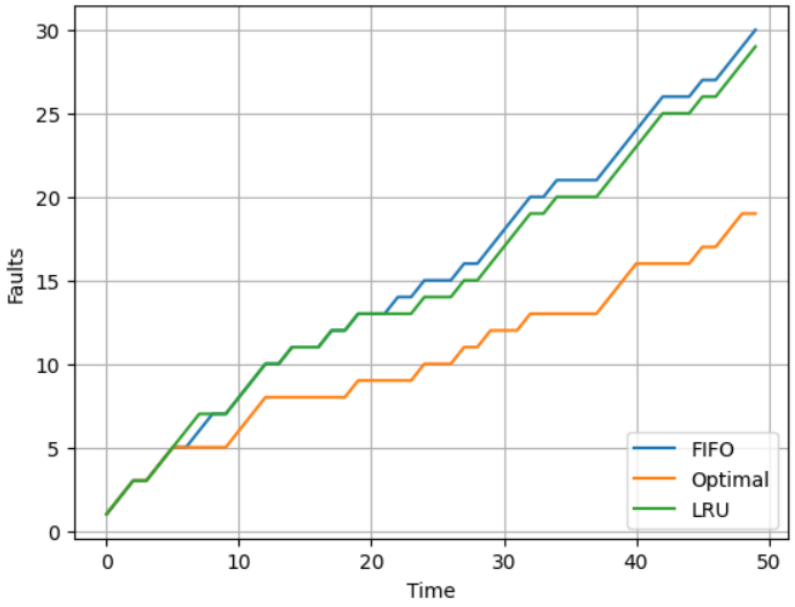
แสดงผลการทดลองที่ได้จากโปรแกรมที่พัฒนาขึ้นโดยใช้ Least Recently Used (LRU) algorithm กับชุดข้อมูลที่ 2 (5 points)

```
2 table: [2]            7 table: [4, 3, 7]      1 table: [7, 0, 1]
0 table: [2, 0]         7                       7
1 table: [2, 0, 1]      2 table: [4, 7, 2]      0
2                       2                       1
5 table: [2, 1, 5]      2                       5 table: [0, 1, 5]
3 table: [2, 5, 3]      4                       3 table: [1, 5, 3]
1 table: [5, 3, 1]      4                       6 table: [5, 3, 6]
2 table: [3, 1, 2]      0 table: [4, 2, 0]      7 table: [3, 6, 7]
1                       2                       1 table: [6, 7, 1]
3                       0                       6
7 table: [3, 1, 7]      5 table: [2, 0, 5]      7
0 table: [3, 7, 0]      5                       4 table: [6, 7, 4]
4 table: [7, 0, 4]      1 table: [0, 5, 1]      4
4                       2 table: [5, 1, 2]      2 table: [7, 4, 2]
3 table: [0, 4, 3]      7 table: [1, 2, 7]      5 table: [4, 2, 5]
3                       0 table: [2, 7, 0]      1 table: [2, 5, 1]
4                       0
```

ตารางสรุป ผลการทดลอง

	FIFO	Optimal Alogorithm	LRU
Faults	30	19	29

Graph สรุปผลการทดลอง



3. **ข้อมูลแรก third reference string dataset (Original)** (5 points)

สิ่งที่แสดงด้านล่างนี้คือการสร้าง reference string ชุดแรกโดยการสุ่ม มีความยาวอย่างน้อย 100 pages

```
[2, 7, 3, 5, 4, 2, 4, 1, 4, 5, 1, 6, 7, 6, 4, 3, 6, 7, 1, 7, 5, 5, 3, 0,
4, 4, 2, 4, 0, 5, 1, 6, 2, 0, 6, 7, 3, 3, 5, 2, 1, 3, 3, 2, 7, 2, 4, 2, 0,
7, 2, 5, 4, 1, 5, 5, 7, 2, 5, 4, 4, 3, 5, 2, 3, 1, 0, 5, 2, 3, 4, 3, 1, 7,
3, 3, 4, 4, 0, 2, 5, 4, 6, 6, 3, 6, 6, 6, 4, 6, 5, 3, 5, 4, 3, 6, 3, 2, 6,
6]
```

อธิบายสมมติฐานที่นักศึกษาใช้ในการสร้าง reference string เช่นความหลากหลาย การถูกอ้างซ้ำเป็นต้น (5 points)

```
Randomly generated array:
Length: 100
Unique values: {0, 1, 2, 3, 4, 5, 6, 7}
Frequency of each value:
  - 0: 6 occurrences
  - 1: 8 occurrences
  - 2: 14 occurrences
```

```
  - 3: 17 occurrences
  - 4: 17 occurrences
  - 5: 15 occurrences
  - 6: 14 occurrences
  - 7: 9 occurrences
```

แสดงผลการทดลองที่ได้จากโปรแกรมที่พัฒนาขึ้นโดยใช้ First-In-First-Out (FIFO) algorithm กับชุดข้อมูลที่ 3 (5 points)

```
2 table: [2]
7 table: [2, 7]
3 table: [2, 7, 3]
5 table: [7, 3, 5]
4 table: [3, 5, 4]
2 table: [5, 4, 2]
4
1 table: [4, 2, 1]
4
5 table: [2, 1, 5]
1
6 table: [1, 5, 6]
7 table: [5, 6, 7]
6
4 table: [6, 7, 4]
3 table: [7, 4, 3]
6 table: [4, 3, 6]
7 table: [3, 6, 7]
1 table: [6, 7, 1]
7
5 table: [7, 1, 5]
5
3 table: [1, 5, 3]
0 table: [5, 3, 0]
4 table: [3, 0, 4]
4
2 table: [0, 4, 2]
4
0
5 table: [4, 2, 5]
1 table: [2, 5, 1]
6 table: [5, 1, 6]
2 table: [1, 6, 2]
0 table: [6, 2, 0]
```

```
6
7 table: [2, 0, 7]
3 table: [0, 7, 3]
3
5 table: [7, 3, 5]
2 table: [3, 5, 2]
1 table: [5, 2, 1]
3 table: [2, 1, 3]
3
2
7 table: [1, 3, 7]
2 table: [3, 7, 2]
4 table: [7, 2, 4]
2
0 table: [2, 4, 0]
7 table: [4, 0, 7]
2 table: [0, 7, 2]
5 table: [7, 2, 5]
4 table: [2, 5, 4]
1 table: [5, 4, 1]
5
5
7 table: [4, 1, 7]
2 table: [1, 7, 2]
5 table: [7, 2, 5]
4 table: [2, 5, 4]
4
3 table: [5, 4, 3]
5
2 table: [4, 3, 2]
3
1 table: [3, 2, 1]
0 table: [2, 1, 0]
5 table: [1, 0, 5]
```

```
2 table: [0, 5, 2]
3 table: [5, 2, 3]
4 table: [2, 3, 4]
3
1 table: [3, 4, 1]
7 table: [4, 1, 7]
3 table: [1, 7, 3]
3
4 table: [7, 3, 4]
4
0 table: [3, 4, 0]
2 table: [4, 0, 2]
5 table: [0, 2, 5]
4 table: [2, 5, 4]
6 table: [5, 4, 6]
6
3 table: [4, 6, 3]
6
6
6
4
6
5 table: [6, 3, 5]
3
5
4 table: [3, 5, 4]
3
6 table: [5, 4, 6]
3 table: [4, 6, 3]
2 table: [6, 3, 2]
6
6
```

แสดงผลการทดลองที่ได้จากโปรแกรมที่พัฒนาขึ้นโดยใช้ Optimal algorithm กับชุดข้อมูลที่ 3 (5 points)

```
2 table: [2]
7 table: [2, 7]
3 table: [2, 7, 3]
5 table: [2, 7, 5]
4 table: [2, 5, 4]
2
4
```

```
1 table: [5, 4, 1]
4
5
1
6 table: [4, 1, 6]
7 table: [4, 6, 7]
6
```

```
4
3 table: [6, 7, 3]
6
7
1 table: [7, 3, 1]
7
5 table: [3, 1, 5]
```

```
5
3
0 table: [1, 5, 0]
4 table: [5, 0, 4]
4
2 table: [0, 4, 2]
4
0
5 table: [0, 2, 5]
1 table: [0, 2, 1]
6 table: [0, 2, 6]
2
0
6
7 table: [0, 2, 7]
3 table: [2, 7, 3]
3
5 table: [2, 3, 5]
2
1 table: [2, 3, 1]
3
3
2
7 table: [2, 1, 7]
2
4 table: [2, 7, 4]
2
```

```
0 table: [2, 7, 0]
7
2
5 table: [2, 7, 5]
4 table: [7, 5, 4]
1 table: [7, 5, 1]
5
5
7
2 table: [5, 1, 2]
5
4 table: [5, 2, 4]
4
3 table: [5, 2, 3]
5
2
3
1 table: [5, 2, 1]
0 table: [5, 2, 0]
5
2
3 table: [2, 0, 3]
4 table: [0, 3, 4]
3
1 table: [3, 4, 1]
7 table: [3, 4, 7]
3
```

```
3
4
4
0 table: [3, 4, 0]
2 table: [3, 4, 2]
5 table: [3, 4, 5]
4
6 table: [3, 4, 6]
6
3
6
6
6
4
6
5 table: [3, 4, 5]
3
5
4
3
6 table: [3, 4, 6]
3
2 table: [3, 6, 2]
6
6
```

แสดงผลการทดลองที่ได้จากโปรแกรมที่พัฒนาขึ้นโดยใช้ Least Recently Used (LRU) algorithm กับชุดข้อมูลที่ 3 (5 points)

```
2 table: [2]
7 table: [2, 7]
3 table: [2, 7, 3]
5 table: [7, 3, 5]
4 table: [3, 5, 4]
2 table: [5, 4, 2]
4
1 table: [4, 2, 1]
4
5 table: [4, 1, 5]
1
6 table: [1, 5, 6]
7 table: [1, 6, 7]
6
4 table: [6, 7, 4]
3 table: [6, 4, 3]
6
7 table: [6, 3, 7]
1 table: [6, 7, 1]
7
5 table: [7, 1, 5]
5
```

```
3 table: [7, 5, 3]
0 table: [5, 3, 0]
4 table: [3, 0, 4]
4
2 table: [0, 4, 2]
4
0
5 table: [0, 4, 5]
1 table: [0, 5, 1]
6 table: [5, 1, 6]
2 table: [1, 6, 2]
0 table: [6, 2, 0]
6
7 table: [6, 0, 7]
3 table: [6, 7, 3]
3
5 table: [7, 3, 5]
2 table: [3, 5, 2]
1 table: [5, 2, 1]
3 table: [2, 1, 3]
3
2
```

```
7 table: [2, 3, 7]
2
4 table: [2, 7, 4]
2
0 table: [2, 4, 0]
7 table: [2, 0, 7]
2
5 table: [2, 7, 5]
4 table: [2, 5, 4]
1 table: [5, 4, 1]
5
5
7 table: [5, 1, 7]
2 table: [5, 7, 2]
5
4 table: [5, 2, 4]
4
3 table: [5, 4, 3]
5
2 table: [5, 3, 2]
3
1 table: [3, 2, 1]
```

```
0 table: [3, 1, 0]
5 table: [1, 0, 5]
2 table: [0, 5, 2]
3 table: [5, 2, 3]
4 table: [2, 3, 4]
3
1 table: [3, 4, 1]
7 table: [3, 1, 7]
3
3
4 table: [3, 7, 4]
4
```

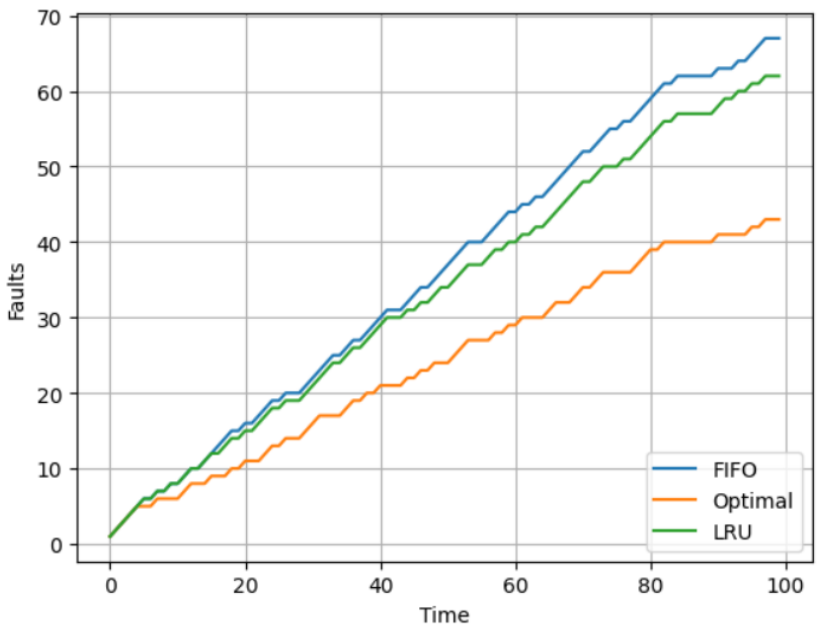
```
0 table: [3, 4, 0]
2 table: [4, 0, 2]
5 table: [0, 2, 5]
4 table: [2, 5, 4]
6 table: [5, 4, 6]
6
3 table: [4, 6, 3]
6
6
6
4
6
```

```
5 table: [4, 6, 5]
3 table: [6, 5, 3]
5
4 table: [5, 3, 4]
3
6 table: [3, 4, 6]
3
2 table: [3, 6, 2]
6
6
```

ตารางสรุป ผลการทดลอง

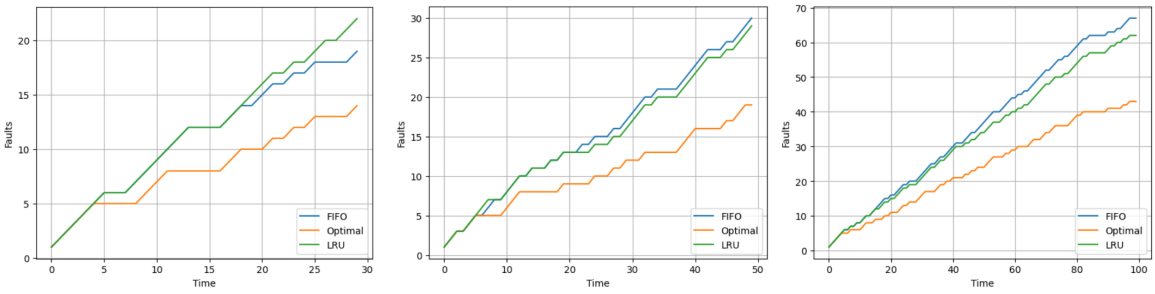
	FIFO	Optimal Alogorithm	LRU
Faults	67	43	62

Graph สรุปผลการทดลอง



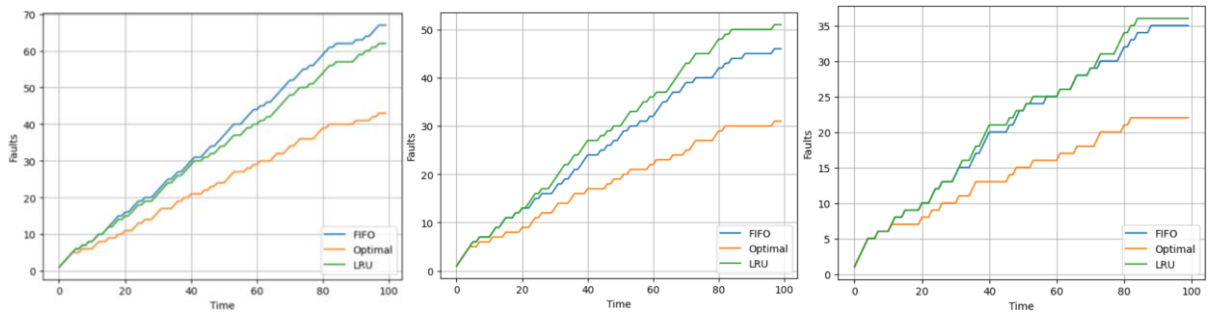
สรุปผลการทดลอง (20 points)

อธิบายการวิเคราะห์ ประสิทธิภาพ การทำงาน ของ Algorithm ทั้ง 3 โดยเปรียบเทียบ ผลการทดลองทั้งหมด ที่ได้จากการเขียนโปรแกรม



จากกราฟทั้ง 3 เมื่อวัดจากจำนวน Page fault แสดงให้เห็นว่า LRU กับ FIFO นั้นมีประสิทธิภาพทำงานที่ใกล้เคียงกัน โดยถ้าสังเกตจะเห็นได้ว่าช่วงที่จำนวนเพจประมาณ 0 – 20 นั้น LRU และ FIFO นั้นทำให้เกิด Page faultsที่ไม่ต่างกันมาก แต่เมื่อจำนวนเพจมากขึ้นก็จะยิ่งเห็นความแตกต่างในเรื่องของจำนวน Page faults โดย LRU นั้นจะก่อให้เกิด Page fault ที่น้อยกว่า FIFO แต่ถึงอย่างนั้นทั้ง 2 อัลกอริทึมก็ยังไม่ใกล้เคียงกับ Optimal Algorithm

<u>เปรียบเทียบจำนวน Page faults ที่จำนวน Page table ต่างๆ</u>



Page table=3 Page table=4 Page Page table=5