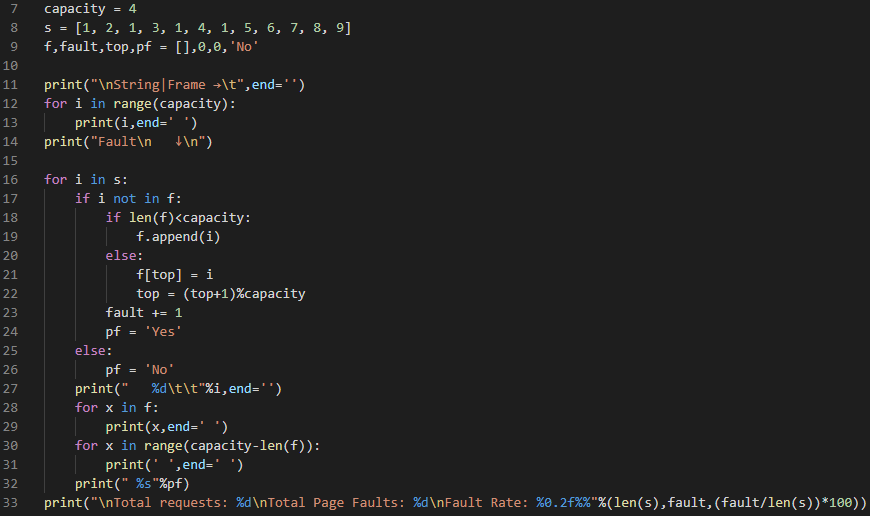
**Page Replacement Algorithms**

**แนวคิดในการพัฒนา Algorithm First In First Out (FIFO)**

มีหลักการทำงานคือเข้าก่อนออกก่อน เข้าทีหลังออกทีหลังโดยจะหลักการ append ตัวที่ต้องการเข้าไปโดยเมื่อ frame ยังมีที่ว่างอยู่จะเข้าไปได้เลย แต่เมื่อ frame เต็มแล้วจะทำการแทนที่แทน โดยการแทนที่จะมีการเก็บตำแน่งที่อยู่เก่าสุดไว้ในตัวแปร top และแทนค่าของมัน หลังจากนั้นให้เพิ่มตำแหน่งของ top เพื่อเตรียมการแทนค่าในครั้งต่อไปแล้ววนไปเรื่อยๆจนครบทุก process



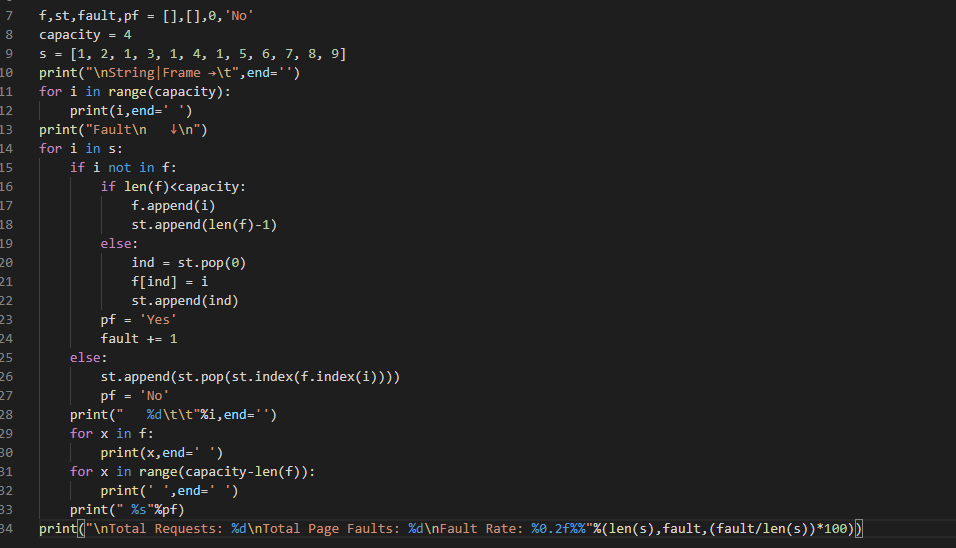
**แนวคิดในการพัฒนา Algorithm** **Optimal**

หลักการการทำงานของ optimal คือ เมื่อ append เข้าไปจน frame เต็มจะทำการเช็คแต่ละตัวที่อยู่ใน frame แต่ละตัวว่ามีการใช้งานในอนาคตอันใกล้หรือไม่ แล้วเปลี่ยนตัวที่จะได้ใช้งานในอนาคตที่ไกลที่สุดออกหรือเรียกอย่างง่ายว่า replace แล้วทำงานไปจนครบทุก process



**แนวคิดในการพัฒนา Algorithm Least Recently used(LRU)**

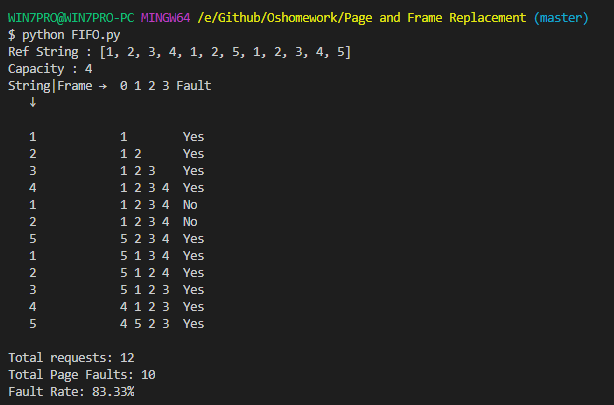
หลักการทำงานของ lru คือจะทำการเช็คย้อนหลังไปว่าในหลังจากผ่านไปหลาย process แล้วตัวไหนใน frame มีการเรียกใช้น้อยที่สุดแล้วจะทำการ replace ค่านั้นทิ้งไปซึ่งหลักการทำงานคล้ายกับ FIFO



**การทดลองที่ 1** : เปรียบเทียบจำนวน Page Fault ของทุก algorithm เมื่ออยู่ภายใต้เงื่อนไขเดียวกัน

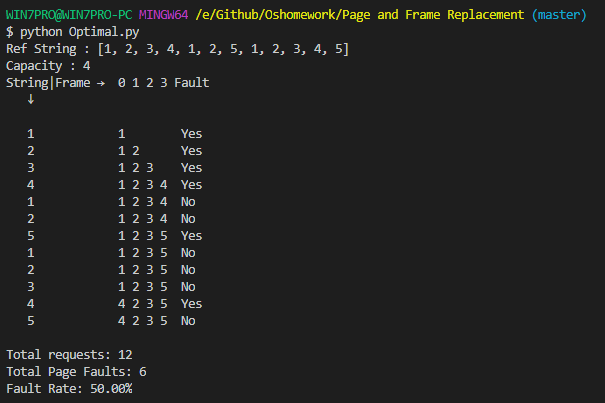
สมมติฐาน : หากมีการใช้ขนาด frame ที่เท่ากัน และ Process ที่เหมือนกันและเท่ากัน จำนวนPage Fault ควรจะเรียงจำนวนการเกิดตาม Algorithm คือ FIFO ,LRU, Optimal จากมากไปน้อยตามลำดับ

**FIFO**



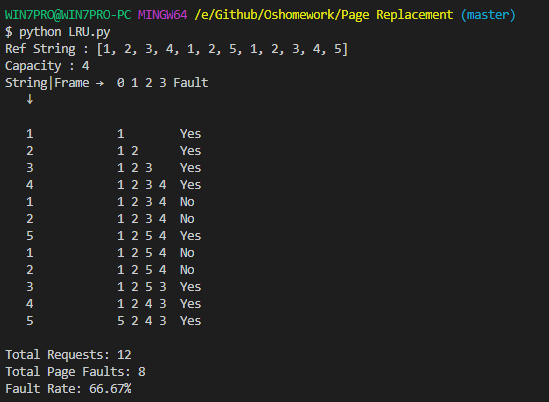
จากภาพแสดงให้เห็นถึงอัตราการเกิด Fault Rate ของ FIFO

**Optimal**



จากภาพแสดงให้เห็นถึงอัตราการเกิด Fault Rate ของ Optimal

**LRU**

****

จากภาพแสดงให้เห็นถึงอัตราการเกิด Fault Rate ของ LRU

ผลการทดลอง

Reference Processes: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

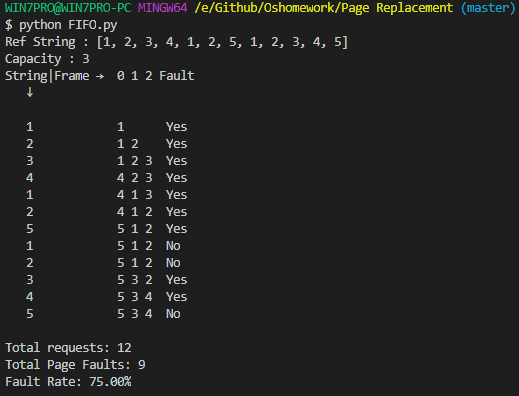
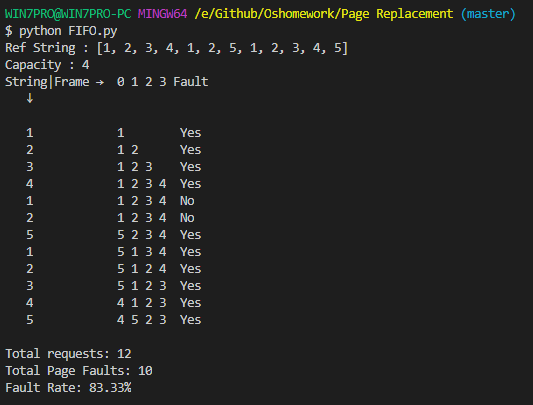
|  |  |
| --- | --- |
| Algorithm | Page fault (ครั้ง) |
| FIFO | 10 |
| Optimal | 6 |
| LRU | 8 |

สรุปผลการทดลอง

จากผลการทดลองพบว่า FIFO ,Optimal และ LRU เกิด Fault Rate จำนวน 10, 6 และ 8 ครั้งตามลำดับซึ่ง จะสามารถเรียงจากมากไปน้อยได้ FIFO LRU และ Optimal ตามลำดับซึ่งเป็นไปตามสมมติฐาน

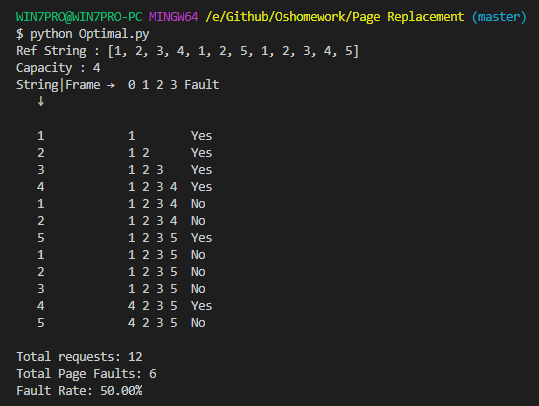
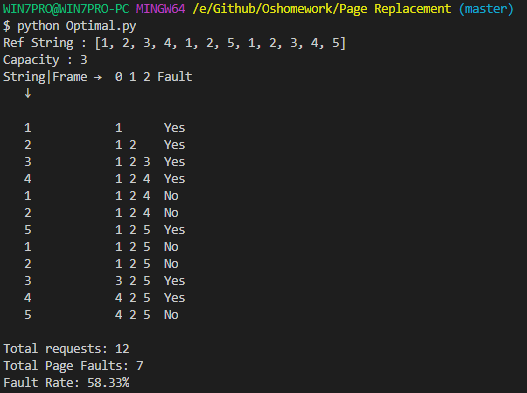
**การทดลองที่ 2 :** การทำงานภายใต้ขนาดของ Frame ที่มากขึ้น

สมมติฐาน : เมื่อมีการเพิ่มขนาดของ Frame แล้ว Page Fault จะต้องลดลง

**FIFO**

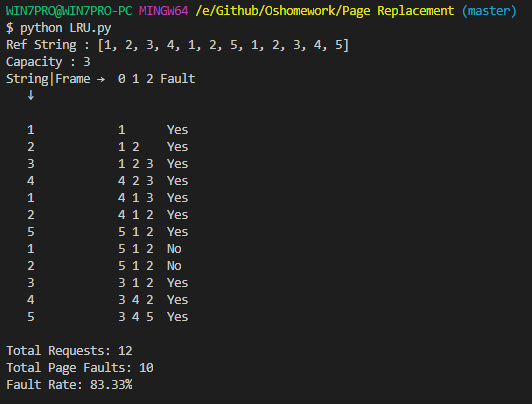
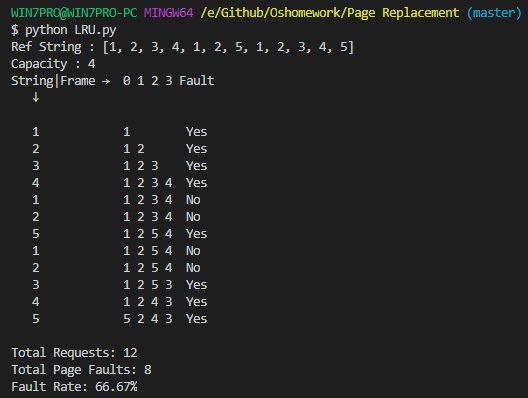
ภาพแสดงผลการทดลองของ FIFO

**Optimal**



ภาพแสดงผลการทดลองของ Optimal

**LRU**

ภาพแสดงผลการทดลองของ LRU

จากการทดลอง 3 algorithm สามารถเขียนเป็นตารางและกราฟได้ดังนี้

Reference Processes : 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm | PageFault while  Frame Size = 3 | PageFault while  Frame Size = 4 | การเปลี่ยนแปลงของ Page Fault |
| FIFO | 9 | 10 | เพิ่มขึ้น 1 ครั้ง |
| Optimal | 7 | 6 | ลดลง 1 ครั้ง |
| LRU | 10 | 8 | ลดลง 2 ครั้ง |

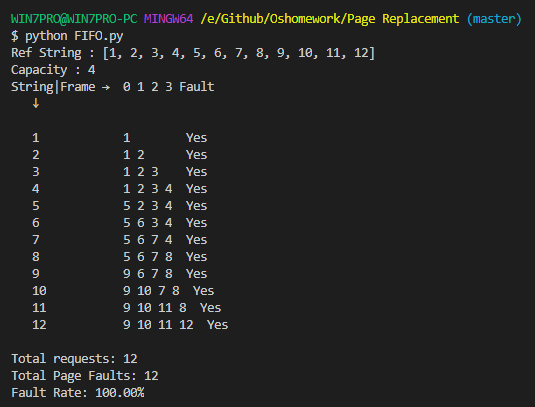
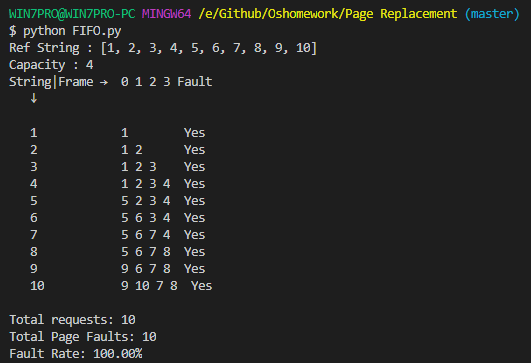
**สรุปผลการทดลอง**

จากการทดลองพบว่าทั้ง Optimal และ LRU นั้นเป็นไปตามสมมติฐานนั่นคือถ้า frame มากขึ้น Page Fault จะลดลง ซึ่งมีเพียงแต่ FIFO เท่านั้นที่เมื่อ frame มากขึ้นแล้ว Page Fault มากขึ้น แสดงว่ามีเพียง FIFO เท่านั้นที่เป็นไปตาม Belady's anomaly

**การทดลองที่ 3 :** เมื่อมี Ref Str ไม่ซ้ำกันเลย

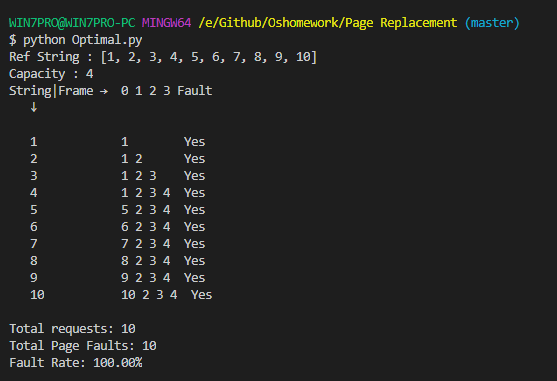
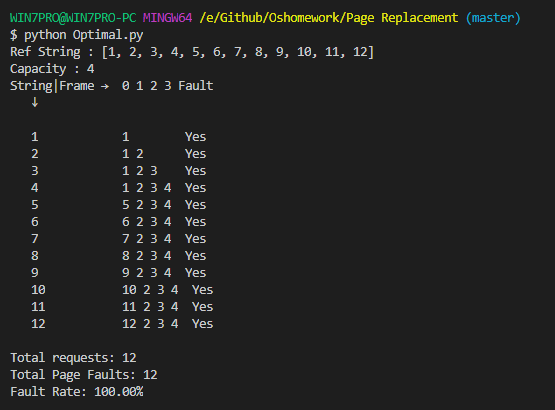
สมมุติฐาน : หากมีการใช้ Ref\_Str ที่ไม่ซ้ำกันเลย จะทำให้ทุก Algorithm มีจำนวน Page Fault เท่ากันทุก Algorithm โดยจะมีจำนวน page fault ตามจำนวน Reference Processes

**FIFO**

** **

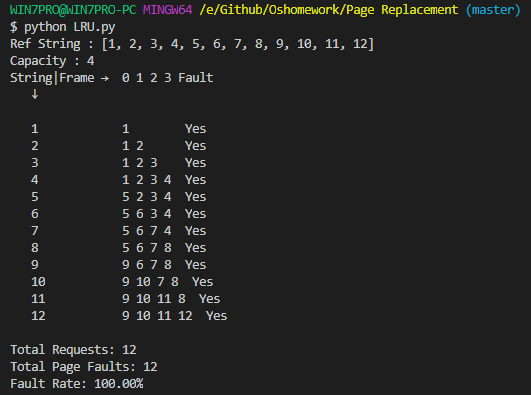
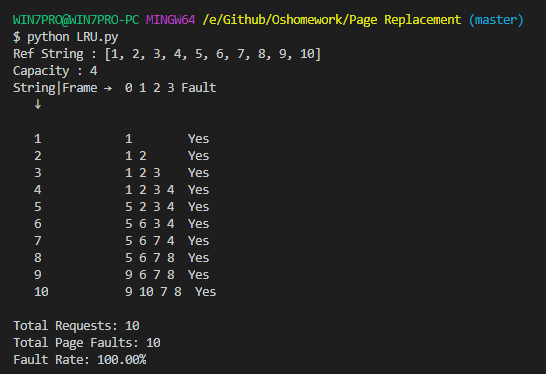
ภาพแสดงผลการทดลองของ FIFO

**Optimal**

****

ภาพแสดงผลการทดลองของ Optimal

**LRU**

ภาพแสดงผลการทดลองของ LRU

Reference Processes: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

และ Reference Processes: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

**สรุปผลการทดลอง**

จากการทดลองพบว่าหากใช้ Ref Str ที่ไม่ซ้ำกันเลยจะพบว่าทั้งสาม algorithm มี Page Fault ที่เท่ากับจำนวนของ Ref Str ที่ใช้ โดยแม้จะเพิ่มขนาดของ frame ก็จะยังมี Page Fault เท่ากับจำนวน Process ที่ใช้งาน