SYSTEMY OPERACYJNE

Maciej Siedlak 273161

Sprawozdanie z Projektu:   
Symulacja Algorytmów Planowania Procesora i Zastępowania Stron

# Wstęp:

Celem tego projektu jest zrozumienie i porównanie różnych algorytmów planowania procesora oraz zastępowania stron. Poprzez symulację różnych strategii zarządzania procesami i pamięcią, projekt ma na celu identyfikację efektywności i ograniczeń każdego z algorytmów w różnych warunkach operacyjnych.  
W moim projekcie zdecydowałem się na zastosowanie algorytmów FCFS, LCFS, FIFO, LRU.

# Opis Algorytmów

## Algorytmy Planowania Procesora

**First Come First Serve (FCFS)**

FCFS to prosty algorytm planowania, który przydziela procesor procesom w kolejności ich przybycia. Jest to podejście nie wywłaszczające, co oznacza, że raz rozpoczęty proces kontynuuje swoje wykonanie aż do zakończenia.

**Last Come First Serve (LCFS)**

LCFS, będący przeciwieństwem FCFS, przydziela procesor najnowszemu procesowi w kolejce. Ten algorytm również nie stosuje wywłaszczania.

## Algorytmy Zastępowania Stron

**First In First Out (FIFO)**

W algorytmie FIFO, kiedy potrzebne jest załadowanie nowej strony, usuwana jest ta, która znajduje się najdłużej w pamięci. Jest to prosta, ale czasami nieefektywna metoda zarządzania pamięcią.

**Least Recently Used (LRU)**

LRU polega na usunięciu strony, która była używana najdawniej. Algorytm ten próbuje przewidzieć strony, które będą najmniej prawdopodobne do ponownego użycia w najbliższej przyszłości.

# Procedura Testowania

Testy zostały przeprowadzone przez symulację serii procesów z różnymi czasami przybycia i wykonania. Dla każdego algorytmu planowania procesora i zastępowania stron zmieniano różne parametry, takie jak liczba procesów, rozmiar pamięci czy intensywność dostępu do stron, aby zaobserwować wpływ na wydajność systemu.

Procedura testowania algorytmów w projekcie rozpoczyna się od pliku main.py, który pełni funkcję koordynatora, wywołując różne moduły i funkcje zaimplementowane w projekcie. Poniżej znajduje się szczegółowy opis procesu testowania oraz roli poszczególnych plików i modułów:

* Plik: main.py
* Funkcja get\_simulation\_parameters: Pobiera od użytkownika parametry potrzebne do symulacji, takie jak liczba procesów, liczba żądań stron i liczba ramek w pamięci.
* Funkcja print\_process\_details: Wyświetla szczegółowe informacje o procesach po wykonaniu algorytmu planowania procesora. Pokazuje ID procesu, czas przybycia, startu, zakończenia i czas oczekiwania.
* Funkcja main: Główna funkcja programu, która pozwala użytkownikowi wybrać między symulacją a generowaniem wykresów. Symulacja zawiera uruchomienie algorytmów planowania procesora (FCFS, LCFS) i zastępowania stron (FIFO, LRU), a następnie zapisuje wyniki do pliku CSV i generuje wykresy.
* Plik: data\_generator.py
* Funkcja generate\_processes: Generuje listę procesów z losowymi czasami przybycia i wykonania. Dla algorytmów FCFS i LCFS generuje różne charakterystyki czasów wykonania.
* Funkcja generate\_page\_requests\_fifo i generate\_page\_requests\_lru: Generują listy żądań stron zoptymalizowane pod kątem algorytmów FIFO i LRU.
* Plik: experiments.py
* Funkcja run\_fcfs i run\_lcfs: Uruchamiają symulacje dla algorytmów FCFS i LCFS. Wykorzystują odpowiednie listy procesów wygenerowane przez data\_generator.
* Funkcja run\_fifo i run\_lru: Uruchamiają symulacje dla algorytmów zastępowania stron FIFO i LRU.
* Funkcja analyze\_results: Analizuje wyniki procesów po symulacji, obliczając średni czas oczekiwania i średni czas cyklu.
* Plik: page\_replacement.py
* Funkcja fifo i lru: Implementacje algorytmów zastępowania stron FIFO i LRU. Obliczają liczbę błędów stron (page faults) na podstawie dostarczonej listy żądań stron.
* Plik: reporting.py
* Funkcja save\_results\_to\_csv: Zapisuje wyniki symulacji do pliku CSV, obejmujące wyniki dla algorytmów planowania procesora i zastępowania stron.
* Funkcja generate\_charts\_from\_csv: Generuje wykresy na podstawie danych z pliku CSV. Wykresy porównują średnie czasy oczekiwania i średnie czasy cyklu dla algorytmów FCFS i LCFS, a także liczbę błędów stron dla algorytmów FIFO i LRU.
* Plik: schedulers.py
* Klasa Process: Reprezentuje proces z unikalnym ID, czasem przybycia i czasem wykonania. Zawiera również zmienne do śledzenia czasu startu i zakończenia.
* Funkcje fcfs i lcfs: Implementują algorytmy planowania procesora FCFS (First Come First Serve) i LCFS (Last Come First Serve), obliczając czasy startu i zakończenia dla każdego procesu.
* Każda funkcja w projekcie ma swoje specyficzne zadanie i współpracuje z innymi, aby umożliwić symulację i analizę różnych strategii zarządzania procesami i pamięcią w systemie operacyjnym.

Każdy z tych modułów odgrywa kluczową rolę w procesie testowania, począwszy od generowania danych, przez symulację działania algorytmów, aż po analizę i prezentację wyników. Proces testowania obejmuje uruchamianie symulacji z różnymi parametrami wejściowymi, zbieranie danych wyjściowych i wyciąganie wniosków na podstawie zebranych informacji.

# Wyniki Eksperymentów

## Algorytmy zastępowania stron (FIFO i LRU):

Na pierwszym wykresie widzimy, że liczba błędów strony (page faults) dla algorytmu FIFO jest wyższa w porównaniu z algorytmem LRU. To sugeruje, że algorytm LRU może być bardziej efektywny w wykorzystaniu dostępnych ramek pamięci, ponieważ lepiej przewiduje, które strony będą potrzebne w najbliższej przyszłości poprzez analizę ostatnio używanych stron.

## Algorytmy planowania procesora (FCFS i LCFS):

Na drugim wykresie widać, że średni czas oczekiwania (Average Waiting Time) oraz średni czas cyklu (Average Turnaround Time) dla algorytmu LCFS jest znacznie wyższy niż dla FCFS. Wskazuje to na to, że FCFS jest bardziej wydajny pod kątem czasu oczekiwania procesów oraz całkowitego czasu potrzebnego na ich zakończenie. Algorytm LCFS, choć prosty, nie jest optymalny w środowisku, gdzie czas reakcji jest kluczowym czynnikiem.

## Porównanie pomiędzy dwoma kategoriami algorytmów:

Porównując algorytmy planowania procesora z algorytmami zastępowania stron, możemy zauważyć, że metryki dla algorytmów planowania procesora (czas oczekiwania i cyklu) są reprezentowane w większej skali niż błędy strony dla algorytmów zastępowania stron. To pokazuje, że algorytmy planowania mają bezpośredni wpływ na percepcję wydajności systemu przez użytkownika, ponieważ są bezpośrednio związane z czasem, jaki procesy spędzają czekając na wykonanie.

* 1. Podsumowanie

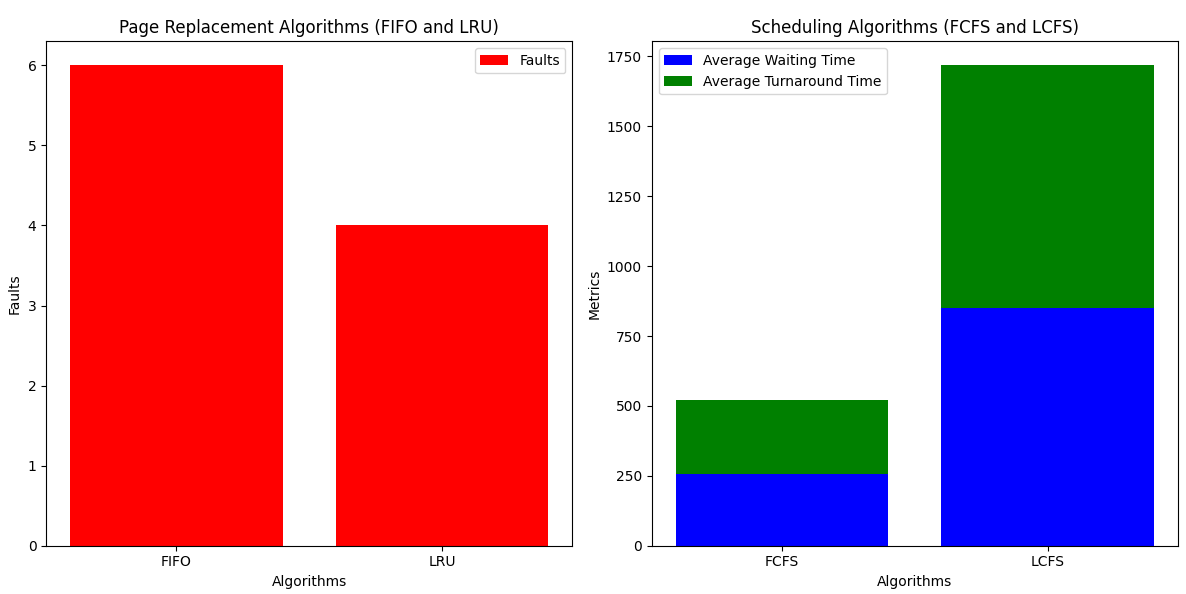
FCFS wydaje się być bardziej efektywny niż LCFS pod względem zarządzania czasem procesów, natomiast LRU wydaje się być lepszym wyborem niż FIFO w zarządzaniu pamięcią stronicowaną. Dla szczegółowych wniosków i rekomendacji warto dokonać dodatkowej analizy, biorąc pod uwagę specyfikę środowiska wykonawczego oraz charakterystykę obciążenia procesora i pamięci.

# Wnioski

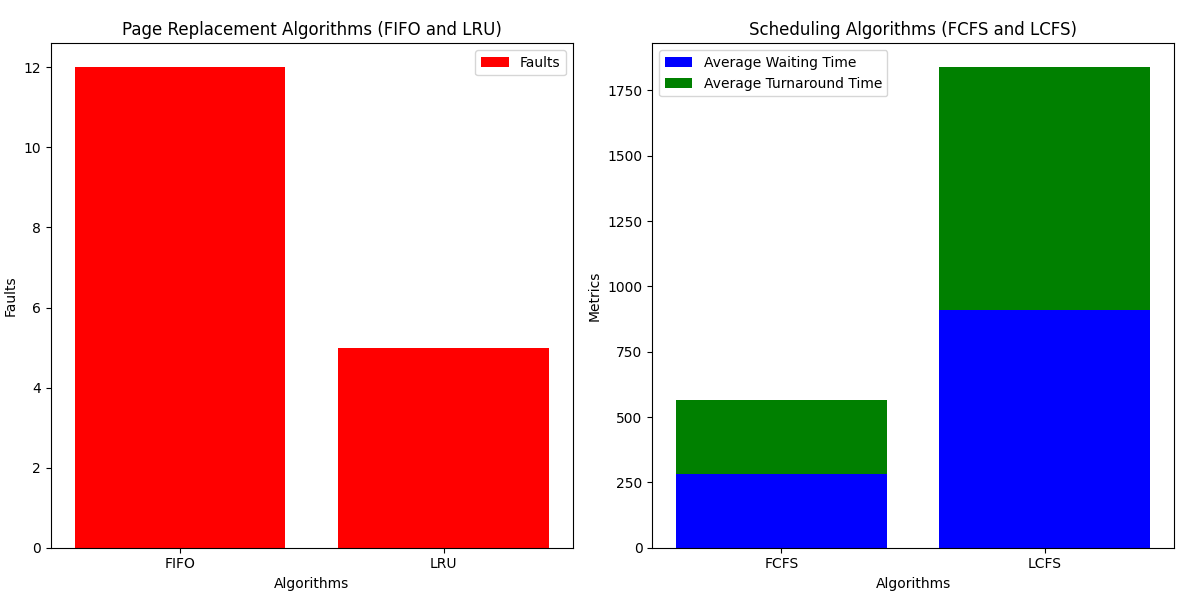
* FCFS: Efektywny dla jednorodnych zadań, ale może opóźniać krótkie procesy za dłuższymi.
* LCFS: Może generować duże opóźnienia; użyteczny, gdy ostatnio przybyłe procesy mają priorytet.
* FIFO: Prosty, ale może często wymieniać używane strony, co nie jest efektywne.
* LRU: Skuteczniejszy niż FIFO w zarządzaniu pamięcią, redukuje niepotrzebne zastępowanie stron.

W oparciu o te wyniki, wybór algorytmu powinien być dostosowany do specyficznych wymagań i charakterystyk systemu operacyjnego.

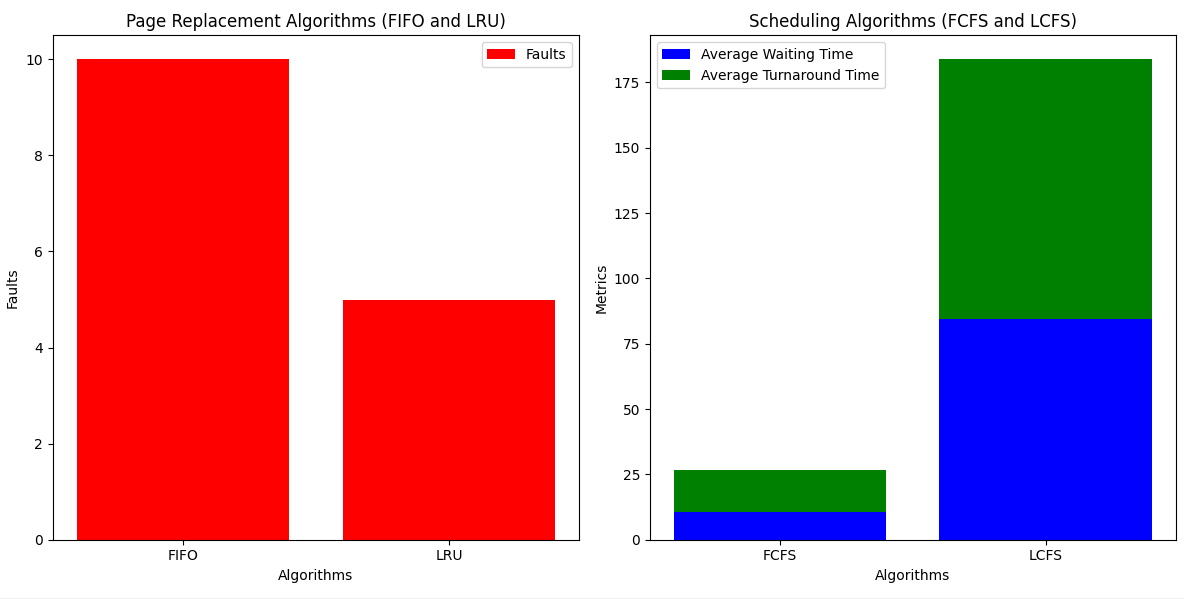
# Wykresy wygenerowane po 3 przykładowych symulacjach



Wykres 1



Wykres 2



Wykres 3