# Systemy Operacyjne

Bartosz Iwanicki 273163

Styczeń 2024

## Spis treści



# 1 Wstęp

Celem projektu było stworzenie dwóch symulacji w dowolonym języku programowania, umożliwiających analizę danych oraz porównanie różnych rozwiązań problemów.W tym przypadku algorytmy zostały zaimplementowane za pomocą języka programowania Python. Pierwsza symulacja dotyczyła planowaniem czasu procesora (process scheduler), a druga symulacji zastępowania stron (page manager).

W ramach "Process scheduler"zaimplementowane zostały algorytmy FCFS (First-Come, First-Served) oraz LCFS (Last-Come, First-Served). Natomiast w drugiej symulacji "Page Manager"zastosowano algorytmy FIFO(First In First Out) i LRU(Least Recently Used).

Analiza wyników uzyskanych z obu symulacji pozwoliła na porównanie efektywności i wydajności różnych podejść do rozwiązywania problemów związanych z harmonogramowaniem procesów oraz zarządzaniem stronami pamięci.

# 2 Zasada działania algorytmów

## 2.1 Process Scheduler

### 2.1.1 First Come First Serve

Kolejność wykonywania procesów w algorytmie FCFS jest analogiczna do organizacji kolejki, gdzie procesy są obsługiwane w porządku ich przyjścia. Proces, który jako pierwszy pojawił się w systemie, jest również pierwszym, który zostanie wykonany. Algorytm ten jest charakteryzowany przez brak mechanizmu wywłaszczania, co oznacza, że procesy są obsługiwane sekwencyjnie. Gdy nowy proces przychodzi w trakcie wykonania innego, musi oczekiwać, aż ten poprzedni zostanie zakończony. Pomimo prostoty, FCFS może prowadzić do efektu "czekającego", gdzie krótkotrwałe procesy muszą oczekiwać na zakończenie dłuższych, co potencjalnie wpływa na ogólną wydajność systemu.

### 2.1.2 Last Come First Serve

W przypadku algorytmu LCFS, procesy są traktowane zgodnie ze zasadą stosu, gdzie proces, który ostatnio przybył, jest obsługiwany jako pierwszy. Podobnie jak w przypadku FCFS, LCFS nie posiada mechanizmu wywłaszczania, co oznacza, że procesy są obsługiwane sekwencyjnie. W momencie, gdy nowy proces przychodzi, procesy są obsługiwane w odwrotnej kolejności ich przyjścia. LCFS, choć prosty do zrozumienia, stosunkowo rzadko znajduje zastosowanie w praktyce z uwagi na ograniczone sytuacje, w których może być skuteczny. Algorytm ten również nie uwzględnia priorytetów czy charakterystyki czasowej procesów.

## 2.2 Page Manager

### 2.2.1 First In First Out

Algorytm FIFO jest prostym algorytmem zastępowania stron, w którym strony są obsługiwane w kolejności ich przyjścia. Strona, która najwcześniej została załadowana do pamięci, jest pierwsza do opuszczenia w przypadku konieczności zastąpienia.

### 2.2.2 Least Recenttly Used

LRU to algorytm zastępowania stron, który opiera się na zasadzie, że strony, które były używane najrzadziej, są najmniej prawdopodobne do ponownego użycia w przyszłości. W kontekście zarządzania pamięcią, LRU jest używane do decydowania, które strony należy zastąpić, gdy pamięć jest pełna.

# 3 Szukane dane

Dane które będą wykorzystne do porównania algorytmów

## 3.1 Process Scheduler

1. Czas całkowity wykonania algorytmów
2. Średni czas oczekiwania na wykonanie procesu

## 3.2 Page Manager

1. Page Fault
2. Hit - hit występuje, gdy strona, do której odwołuje się program, już znajduje się w pamięci RAM

# 4 Klasy w programie

## 4.1 Page

Klasa Page reprezentuje stronę pamięci i zawiera metody do generowania losowego strumienia stron.

### 4.2 FIFO

Klasa FIFO implementuje algorytm zastępowania stron "First-In-First-Out"(FIFO) dla menedżera stron.

### 4.3 LRU

Klasa LRU implementuje algorytm zastępowania stron "Least-Recently-Used"(LRU) dla menedżera stron.

### 4.4 ProcessGenerator

Klasa ProcessGenerator generuje losowe procesy, które są używane w symulacjach algorytmów planowania procesów.

### 4.5 Scheduler

Klasa Scheduler zawiera metody do symulowania algorytmów planowania procesów, takich jak FCFS i LCFS.

### 4.6 MetricsCalculator

Klasa MetricsCalculator oblicza różne metryki związane z wykonaniem procesów, takie jak średni czas oczekiwania i całkowity czas wykonania.

### 4.7 ResultsSaver

Klasa ResultsSaver zajmuje się zapisywaniem wyników symulacji do plików JSON.

### 4.8 Process

Klasa `Process` reprezentuje pojedynczy proces w systemie operacyjnym. Każdy proces posiada unikalny identyfikator (`process_id`), czas przyjścia do systemu (`arrival_time`) oraz czas wymagany do wykonania (`execution_time`). Dodatkowo, klasa przechowuje informacje o czasie oczekiwania na wykonanie procesu (`waiting_time`) oraz czasie zakończenia wykonania procesu (`finish_time`).

### 4.9 JsonReader

Klasa JsonReader wczytuje dane z plików JSON, które zawierają wyniki symulacji.

### 4.10 Plots

Klasa Plots odpowiada za rysowanie wykresów porównawczych na podstawie wczytanych danych.

### 4.11 Main

Klasa Main jest główną klasą programu, która obsługuje interakcję z użytkownikiem, uruchamia symulacje i zarządza innymi klasami.

## 5 Działanie Programu

Po uruchomieniu programu pojawi się menu, w którym użytkownik może wybrać interesującą go opcję:

1. Porównanie algorytmów FIFO i LRU.
2. Porównanie algorytmów FCFS i LCFS.
3. Rysowanie wykresów.
4. Odczyt plików z wynikami.

Przy wyborze algorytmów do zastępowania stron program oczekuje od użytkownika podania:

1. Ilość ramek - odnosi się do fizycznych bloków pamięci używanych do przechowywania stron znajdujących się obecnie w pamięci RAM.

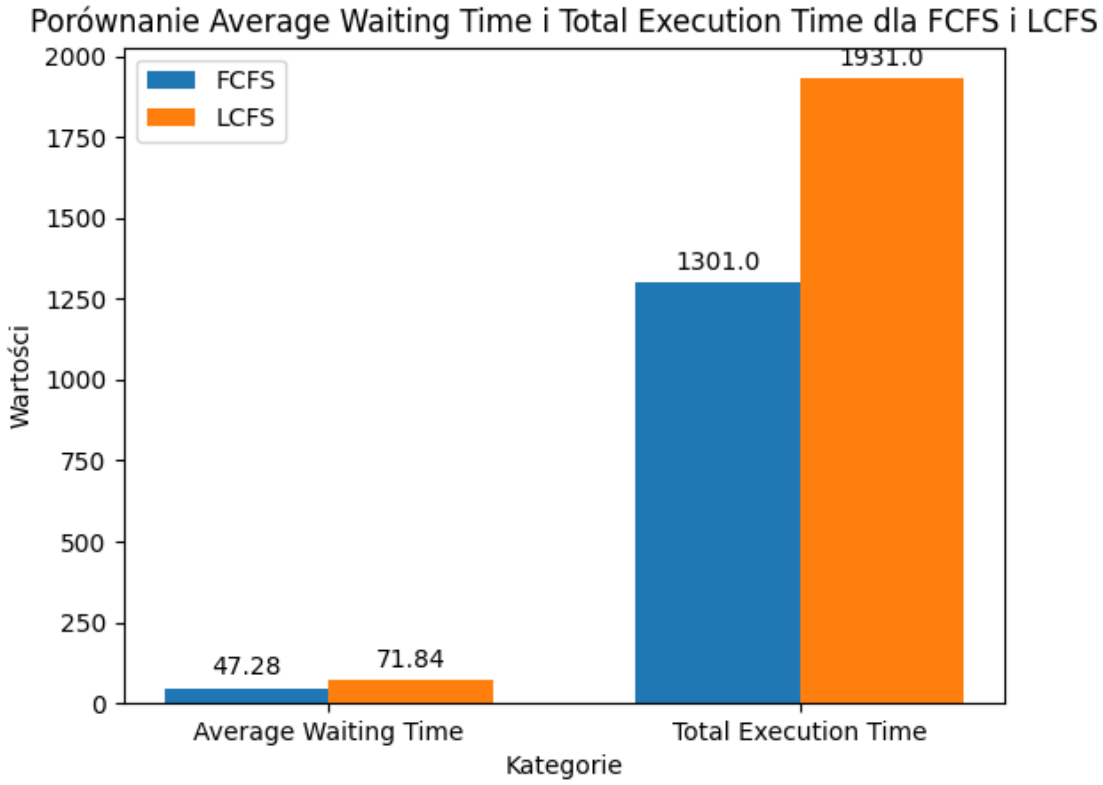
2. Ilość stron - liczba stron używanych w algorytmie (numer strony generowany losowo, domyślnie num$_p$*agestolosowaliczbacakow*

   Przy wyborze algorytmów do planowania czasu procesora, program oczekuje od użytkownika wprowadzenia liczby procesów, na podstawie których algorytmy będą działały.

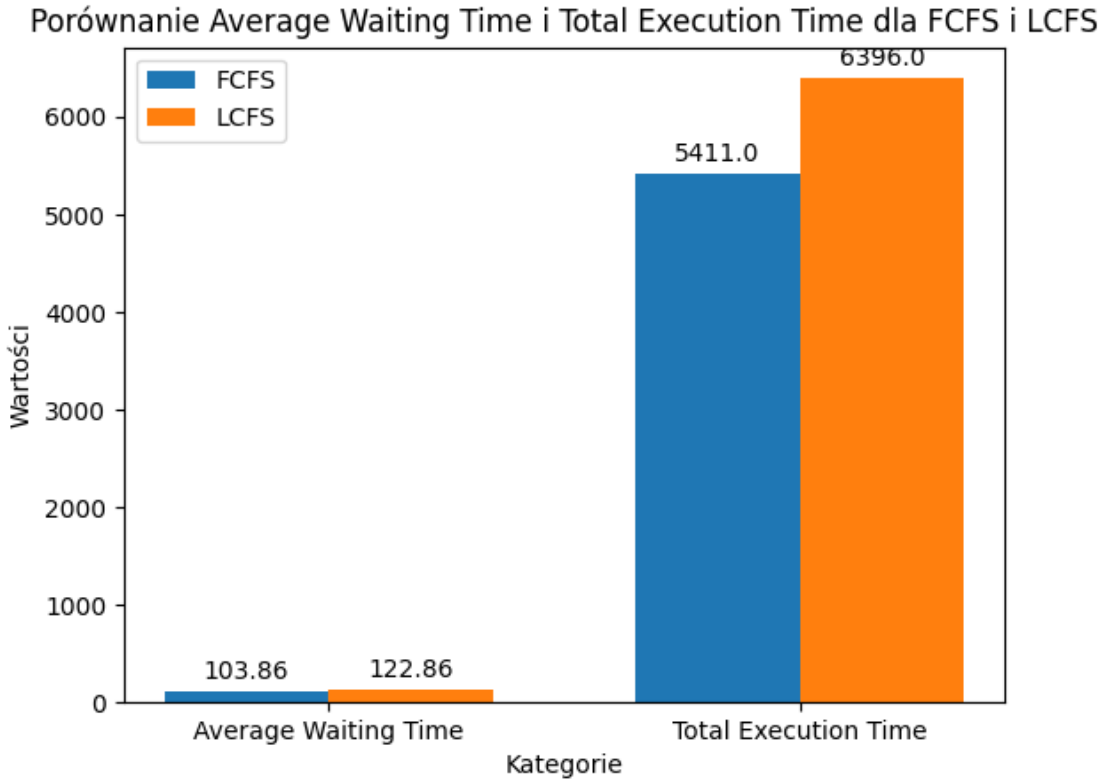
   Po przeprowadzeniu symulacji program umożliwia wyświetlenie wykresów oraz odczyt plików JSON zawierających wyniki ostatnich przeprowadzonych symulacji. Dostosuj tekst, aby nadawał się do dokumentacji.

# 6 Porównanie wyników

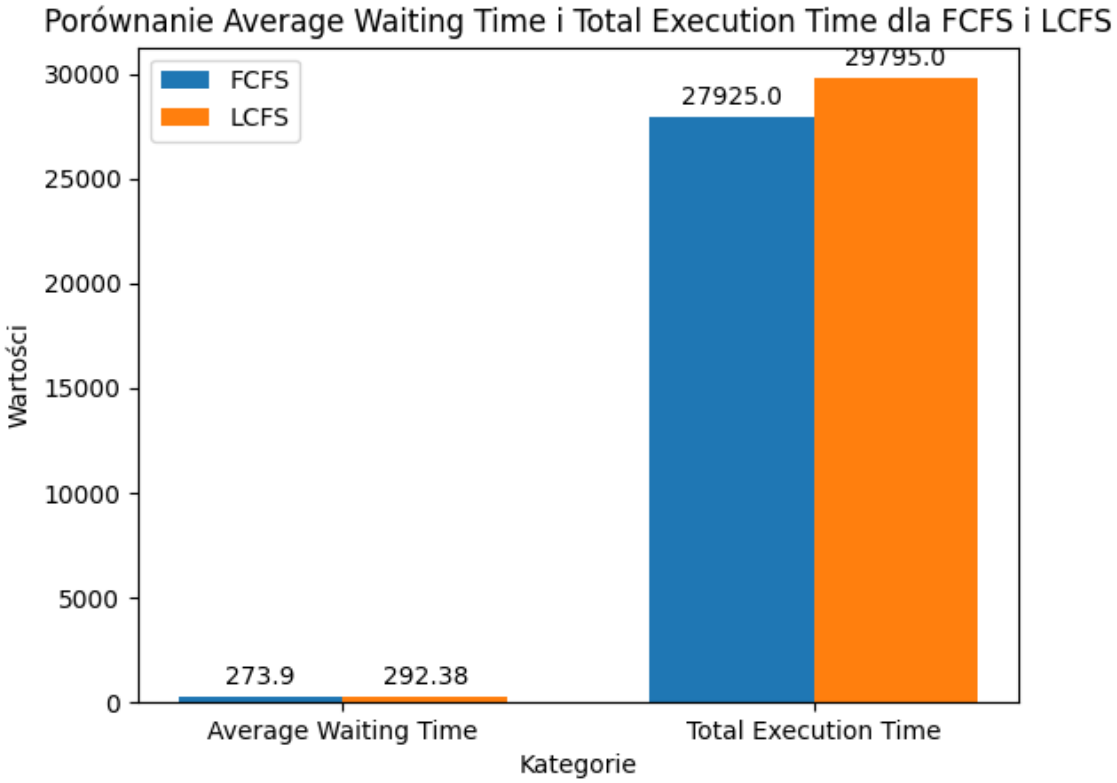
### 6.0.1 FCFS I LCFS 25 procesów



### 6.0.2 FCFS I LCFS 50 procesów



### 6.0.3 FCFS I LCFS 100 procesów

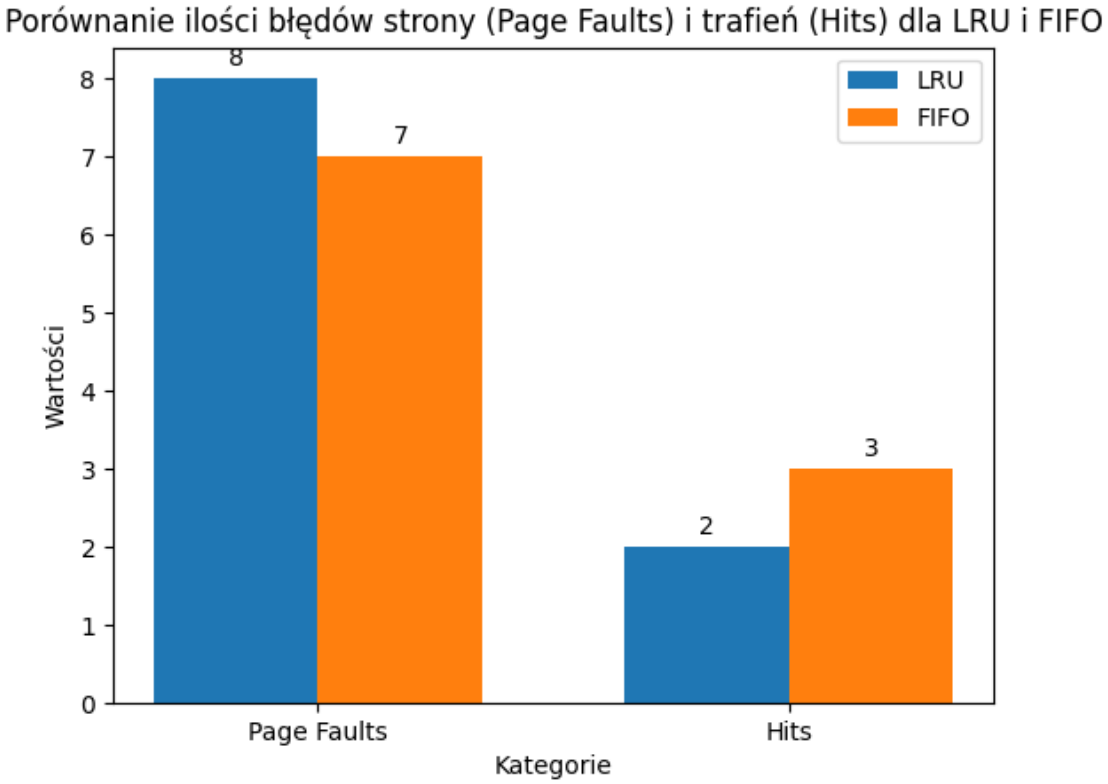



### 6.0.4 Wnioski

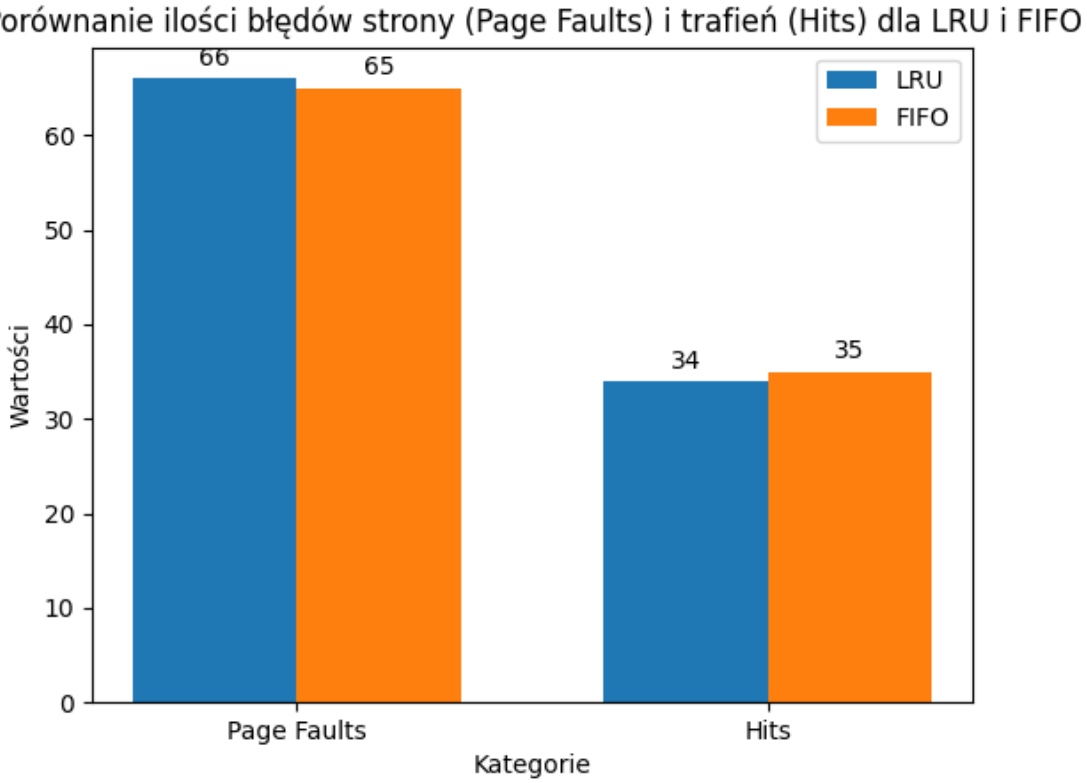
Dzięki wykresom jesteśmy w stanie zauważyć, że algorytm FCFS jest o bardziej efektywny i sprawniejszy od algorytmu LCFS. Czas średni oczekiwania na wykonanie procesu jest krótszy dla FCFS. Czas wykonania również różni sie na korzyść FCFS. Na podstawie tych danych jestesmy w stanie swierdzic że w tym przypadku algorytm FCFS jest o wiele bardziej efektywny niż LCFS.

## 6.1 FIFO I LRU

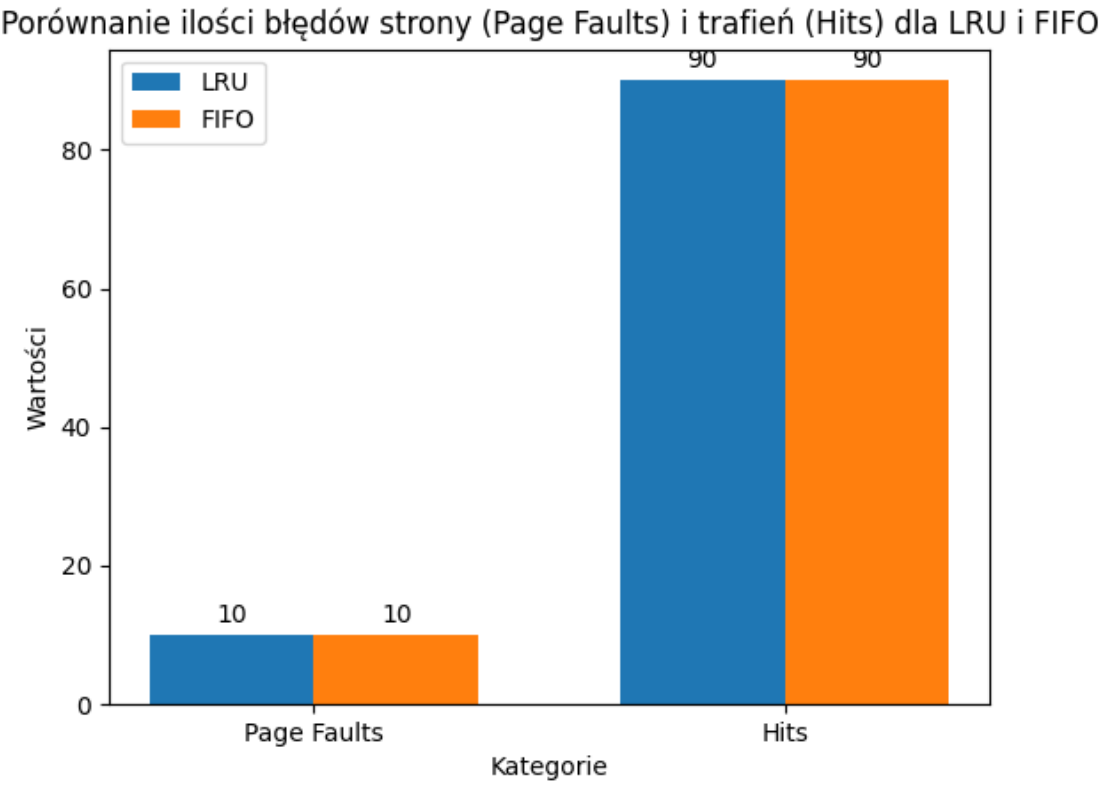
Symulacja przeprowadzona dla 3 frames i 10 stron



### 6.1.1 Symulacja przeprowadzona dla 3 frames i 100 stron



### 6.1.2 Symulacja przeprowadzona dla 25 frames i 100 stron




### 6.1.3 wnioski

Na podstawie wykresu jesteśmy w stanie stwierdzić, że algorytm LRU wykonał mniejszą ilość zamian stron niż FIFO co może oznaczać wiekszą wydajność algorytmu FIFO.Zdarzają sie przypadki, że wydajność algorytmów jest taka sama