Sprawozdanie

Systemy Operacyjne

Projekt programu symulacyjnego implementującego algorytmy planowania czasu procesora i algorytmy zastępowania stron.

Autor:	Oliver Pirko
Nr indeksu:	281 366
Wybrane algorytmy:	FCFS, LCFS, FIFO, LRU
Wybrany język programowania:	Python
Link do repozytorium GitHub:	https://github.com/olip141/SO-Program-Symulacyjny

1. Wstęp.

Celem sprawozdania jest porównanie dwóch algorytmów planowania czasu procesora: algorytm FCFS (First Come First Served) i LCFS (Last Come First Served) oraz dwóch algorytmów zastępowania stron: algorytm FIFO (First In First Out) i LRU (Least Recently Used) na zadanych danych symulacyjnych, analiza wyników eksperymentów i wyciągnięcie na ich podstawie wniosków.

Implementacja algorytmów znajduje się w plikach:

- FCFS – fcfs.py
- LCFS – lcfs.py
- FIFO – fifo.py
- LRU – lru.py

Dane testowe znajdują się w plikach:

- Dla FCFS i LCFS w folderze:
  - DaneTestoweFCFSLCFS w formacie:
    - test_data_{nr eksperymentu}.txt
- Dla FIFO i LRU w folderze:
  - DaneTestoweFIFOLRU w formacie:
    - test_data_{nr eksperymentu}.txt

Surowe wyniki eksperymentów znajdują się w plikach:

- Dla FCFS i LCFS w folderze:
  - WynikiEksperymentówFCFSLCFS w formacie:
    - eksperyment{nr eksperymentu}_{rodzaj algorytmu}.txt
- Dla FIFO i LRU w folderze:
  - WynikiEksperymentówFIFOLRU w formacie:
    - eksperyment{nr eksperymentu}_{rodzaj algorytmu}.txt
    -

2. Opis algorytmów planowania czasu procesora.

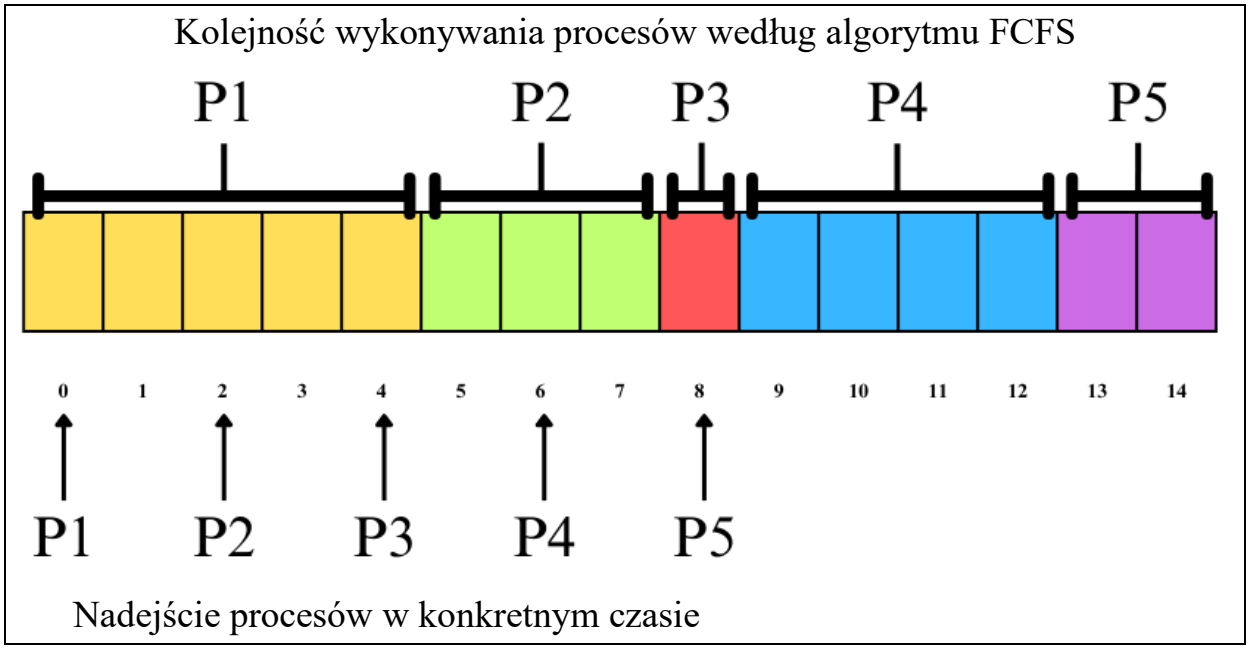
Algorytm FCFS (First Come First Served):

Algorytm ten obsługuje zadania w kolejności, w jakiej zgłosiły się do systemu. Pierwsze zadanie, które przybyło, jest przetwarzane jako pierwsze, niezależnie od jego czasu wykonania.

Przykładowe dane dla algorytmu FCFS:

ID Procesu	Czas nadejścia	Czas wykonania
1	0 ms	5 ms
2	2 ms	3 ms
3	4 ms	1 ms
4	6 ms	4 ms
5	8 ms	2 ms

Graficzne przedstawienie działania algorytmu FCFS na przykładowych danych:



Wyniki dla przykładowych danych:

ID Procesu	Czas oczekiwania	Średni czas oczekiwania
1	0 ms	3 ms
2	3 ms	
3	4 ms	
4	3 ms	
5	5 ms	

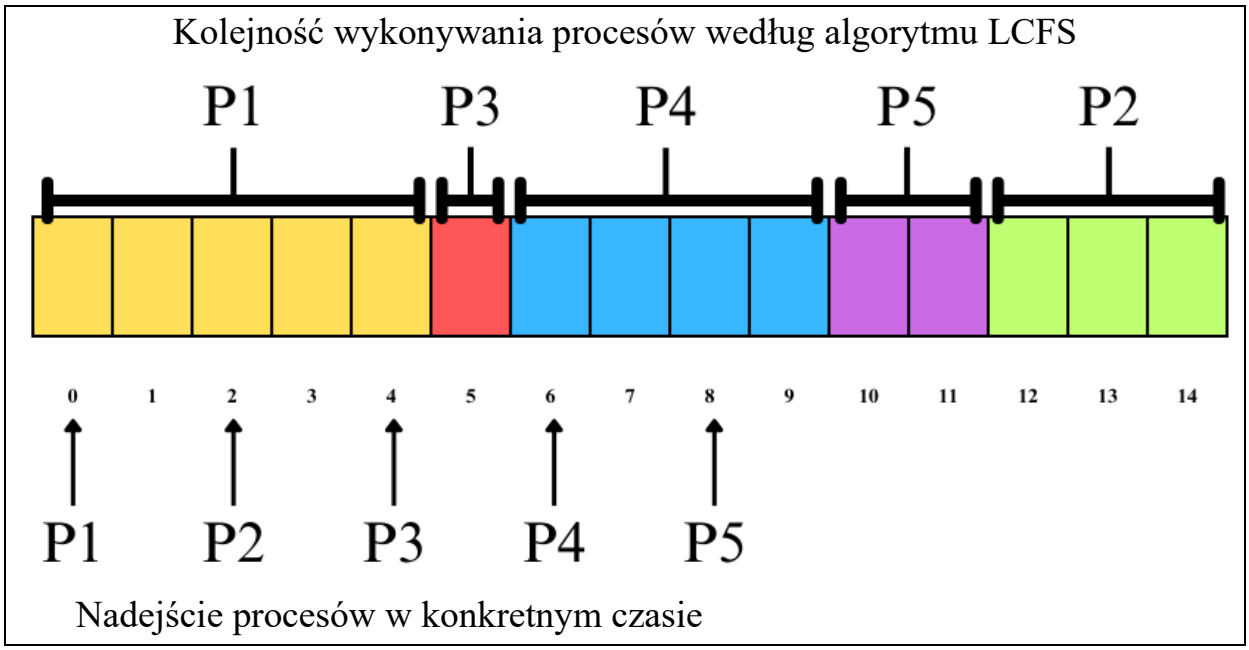
Algorytm LCFS (Last Come First Served):

W tym algorytmie ostatnie zgłoszone zadanie jest obsługiwane jako pierwsze. W przeciwieństwie do algorytmu FCFS w algorytmie LCFS, ostatnio zgłoszone zadania mają priorytet. Tak samo ja w przypadku algorytmu FCFS, w algorytmie LCFS czas wykonania zadania nie ma wpływu na jego działanie.

Przykładowe dane dla algorytmu LCFS:

ID Procesu	Czas nadejścia	Czas wykonania
1	0 ms	5 ms
2	2 ms	3 ms
3	4 ms	1 ms
4	6 ms	4 ms
5	8 ms	2 ms

Graficzne przedstawienie działania algorytmu LCFS na przykładowych danych:



Wyniki dla przykładowych danych:

ID Procesu	Czas oczekiwania	Średni czas oczekiwania
1	0 ms	2,6 ms
2	10 ms	
3	1 ms	
4	0 ms	
5	2 ms	

3. Opis algorytmów zastępowania stron.

Algorytm FIFO (First In First Out):

Działa na zasadzie kolejkowania, gdzie pierwszy element, który trafił do kolejki, jest pierwszym, który zostanie zastąpiony. Algorytm FIFO nie uwzględnia żadnych priorytetów.

Przykładowa sekwencja stron:

7	0	1	2	0	3	0	1	0

Graficzne przedstawienie działania algorytmu FIFO na przykładowej sekwencji stron i buforu o pojemności 2 ramek:

Kroki								
1	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***
7	7	1	1	0	0	0	1	1
	0	0	2	2	3	3	3	0

Wyniki dla przykładowej sekwencji:

Liczba błędów braku strony	8
Liczba trafień	1

Algorytm LRU (Least Recently Used):

Algorytm wymiany stron pamięci, który usuwa tę stronę, która była najdłużej nieużywana. Działanie algorytmu zależy od czasu ostatniego dostępu. W przypadku konieczności zastąpienia strony, to wybierana jest ta która ma najstarszy czas dostępu.

Przykładowa sekwencja stron:

7	0	1	2	0	3	0	1	0

Graficzne przedstawienie działania algorytmu LRU na przykładowej sekwencji stron i buforu o pojemności 2 ramek:

Kroki								
1	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***
7	7	1	1	0	0	0	0	0
	0	0	2	2	3	3	1	1

Wyniki dla przykładowej sekwencji:

Liczba błędów braku strony	7
Liczba trafień	2

4. Przeprowadzone eksperymenty dla algorytmów planowania czasu procesora.

Eksperymenty od 1 do 3:

Różna ilość procesów (25, 75, 125), losowe czasy nadejścia w przedziale od 0 do 100 i stałym czasie wykonania = 5.

Eksperyment nr 1:

25 procesów o losowych czasach nadejścia w przedziale od 0 do 100 i czasie wykonania = 5.

Przykładowe dane testowe:

processes = [Process(name = 19, arrival_time = 13, burst_time = 5),
Process(name = 18, arrival_time = 16, burst_time = 5),
Process(name = 4, arrival_time = 25, burst_time = 5),
Process(name = 6, arrival_time = 26, burst_time = 5),
Process(name = 1, arrival_time = 28, burst_time = 5),
Process(name = 7, arrival_time = 33, burst_time = 5),
Process(name = 16, arrival_time = 33, burst_time = 5),
Process(name = 22, arrival_time = 37, burst_time = 5),
Process(name = 23, arrival_time = 40, burst_time = 5),
Process(name = 8, arrival_time = 41, burst_time = 5),
Process(name = 21, arrival_time = 45, burst_time = 5),
Process(name = 3, arrival_time = 47, burst_time = 5),
Process(name = 25, arrival_time = 49, burst_time = 5),
Process(name = 2, arrival_time = 52, burst_time = 5),
Process(name = 15, arrival_time = 52, burst_time = 5),
Process(name = 14, arrival_time = 55, burst_time = 5),
Process(name = 17, arrival_time = 55, burst_time = 5),
Process(name = 24, arrival_time = 57, burst_time = 5),
Process(name = 5, arrival_time = 64, burst_time = 5),
Process(name = 12, arrival_time = 64, burst_time = 5),
Process(name = 13, arrival_time = 66, burst_time = 5),
Process(name = 9, arrival_time = 72, burst_time = 5),
Process(name = 20, arrival_time = 73, burst_time = 5),
Process(name = 10, arrival_time = 74, burst_time = 5),
Process(name = 11, arrival_time = 77, burst_time = 5)]

Wyniki eksperymentu nr 1:

Średni czas oczekiwania dla FCFS = 27,08

Średni czas oczekiwania dla LCFS = 27,08

Eksperyment nr 2:

75 procesów o losowych czasach nadejścia w przedziale od 0 do 100 i czasie wykonania =5

Dane testowe:

W pliku test_data_2.txt

Wyniki eksperymentu nr 2:

Średni czas oczekiwania dla FCFS = 144.16

Średni czas oczekiwania dla LCFS = 144.16

Eksperyment nr 3:

125 procesów o losowych czasach nadejścia w przedziale od 0 do 100 i czasie wykonania =5

Dane testowe:

W pliku test_data_3.txt

Wyniki eksperymentu nr 1:

Średni czas oczekiwania dla FCFS = 260.008 ≈ 260.01

Średni czas oczekiwania dla LCFS = 260.008 ≈ 260.01

Eksperymenty od 4 do 6:

Różna ilość procesów (25, 75, 125), stały czas nadejścia = 0 i czas wykonania w przedziale od 1 do 10.

Eksperyment nr 4:

25 procesów o czasie nadejścia = 0 i czasie wykonania w przedziale od 1 do 10.

Dane testowe:

W pliku test_data_4.txt

Wyniki eksperymentu nr 4:

Średni czas oczekiwania dla FCFS = 68.52

Średni czas oczekiwania dla LCFS = 59.16

Eksperyment nr 5:

75 procesów o czasie nadejścia = 0 i czasie wykonania w przedziale od 1 do 10.

Dane testowe:

W pliku test_data_5.txt

Wyniki eksperymentu nr 5:

Średni czas oczekiwania dla FCFS = 189.72

Średni czas oczekiwania dla LCFS = 179.29333333333332 ≈ 179.29

Eksperyment nr 6:

125 procesów o czasie nadejścia = 0 i czasie wykonania w przedziale od 1 do 10.

Dane testowe:

W pliku test_data_6.txt

Wyniki eksperymentu nr 6:

Średni czas oczekiwania dla FCFS = 301.272 ≈ 301.27

Średni czas oczekiwania dla LCFS = 280.04

Eksperymenty od 7 do 9:

Różna ilość procesów (25, 75, 125), czasie nadejścia i czasie wykonania = (10, 20, 30) z odchylenie standardowym = 5.

Eksperyment nr 7:

25 procesów o czasie nadejścia i czasie wykonania = 10 z odchyleniem standardowym = 5.

Dane testowe:

W pliku test_data_7.txt

Wyniki eksperymentu nr 7:

Średni czas oczekiwania dla FCFS = 80.08

Średni czas oczekiwania dla LCFS = 95.88

Eksperyment nr 8:

75 procesów o czasie nadejścia i czasie wykonania = 20 z odchyleniem standardowym = 5.

Dane testowe:

W pliku test_data_8.txt

Wyniki eksperymentu nr 8:

Średni czas oczekiwania dla FCFS = 738.4933333333333… ≈ 738.49

Średni czas oczekiwania dla LCFS = 751.1466666666666… ≈ 751.15

Eksperyment nr 9:

125 procesów o czasie nadejścia i czasie wykonania = 30 z odchyleniem standardowym = 5.

Dane testowe:

W pliku test_data_9.txt

Wyniki eksperymentu nr 9:

Średni czas oczekiwania dla FCFS = 1812.24

Średni czas oczekiwania dla LCFS = 1778.832 ≈ 1778.83

Eksperyment nr 10:

200 procesów o losowych czasach nadejścia z przedziału od 0 do 500 i losowym czasie wykonania w przedziale od 1 do 10.

Dane testowe:

W pliku test_data_10.txt

Wyniki eksperymentu nr 10:

Średni czas oczekiwania dla FCFS = 891.0

Średni czas oczekiwania dla LCFS = 966.135≈ 966.14

Eksperyment nr 11:

200 procesów o losowych czasach nadejścia z przedziału od 0 do 500 i wykonania z przedziału 1 do 20, gdzie czas wykonania jest dynamiczny.

Dane testowe:

W pliku test_data_11.txt

Wyniki eksperymentu nr 11:

Średni czas oczekiwania dla FCFS = 1276.035 ≈ 1276.04

Średni czas oczekiwania dla LCFS = 1368.865 ≈ 1368.87

Eksperyment nr 12:

250 procesów o losowych czasach nadejścia z przedziału od 0 do 1000 i wykonania z przedziału od 1 do 50, z dużą zmiennością (odchyleniem standardowym równym odpowiednio 50 i 20).

Dane testowe:

W pliku test_data_12.txt

Wyniki eksperymentu nr 12:

Średni czas oczekiwania dla FCFS = 2950.156 ≈ 2950.16

Średni czas oczekiwania dla LCFS = 3028.512 ≈ 3028.51

5. Przeprowadzone eksperymenty dla algorytmów zastępowania stron.

Eksperyment nr 1 i 2:

Eksperyment bada wpływ liczby ramek na taką samą sekwencję stron.

Eksperyment nr 1:

Mała liczba ramek.

Przykładowe dane testowe:

page_sequence = [1, 7, 5, 1, 2, 5, 7, 7, 2, 1]

page_frames = 3

Wyniki eksperymentu nr 1:

FIFO Błędy braku strony: 5

LRU Błędy braku strony: 6

Eksperyment nr 2:

Duża liczba ramek.

Dane testowe:

W pliku test_data_2.txt

Wyniki eksperymentu nr 2:

FIFO Błędy braku strony: 4

LRU Błędy braku strony: 4

Eksperyment nr 3.1:

Eksperyment bada anomalię Belady'ego, która powoduje zwiększenie się ilości błędów mimo zwiększania ilości ramek. Liczba ramek = 3.

Dane testowe:

W pliku test_data_3_3_Belady.txt

Wyniki eksperymentu nr 3.1:

FIFO Błędy braku strony: 9

LRU Błędy braku strony: 10

Eksperyment nr 3.2:

Eksperyment bada anomalię Belady'ego, która powoduje zwiększenie się ilości błędów mimo zwiększania ilości ramek. Liczba ramek = 4

Dane testowe:

W pliku test_data_3_4_Belady.txt

Wyniki eksperymentu nr 3.2:

FIFO Błędy braku strony: 10

LRU Błędy braku strony: 8

Eksperyment nr 3.3:

Eksperyment bada anomalię Belady'ego, która powoduje zwiększenie się ilości błędów mimo zwiększania ilości ramek. Liczba ramek = 5.

Dane testowe:

W pliku test_data_3_5_Belady.txt

Wyniki eksperymentu nr 3.3:

FIFO Błędy braku strony: 5

LRU Błędy braku strony: 5

Eksperyment nr 4:

Eksperyment bada jak lokalność odniesienia (grupy stron są używane w krótkim odstępie czasu) wpływa na liczbę błędów.

Dane testowe:

W pliku test_data_4.txt

Wyniki eksperymentu nr 4:

FIFO Błędy braku strony: 9

LRU Błędy braku strony: 8

Eksperyment nr 5:

Eksperyment bada jak losowa sekwencja stron wpływa na liczbę błędów.

Dane testowe:

W pliku test_data_5.txt

Wyniki eksperymentu nr 5:

FIFO Błędy braku strony: 10

LRU Błędy braku strony: 10

Eksperyment nr 6:

Eksperyment bada jak monotoniczny wzrost sekwencji stron wpływa na liczbę błędów.

Dane testowe:

W pliku test_data_6.txt

Wyniki eksperymentu nr 6:

FIFO Błędy braku strony: 10

LRU Błędy braku strony: 10

Eksperyment nr 7:

Eksperyment bada jak duża powtarzalność jednej strony w sekwencji wpływa na liczbę błędów.

Dane testowe:

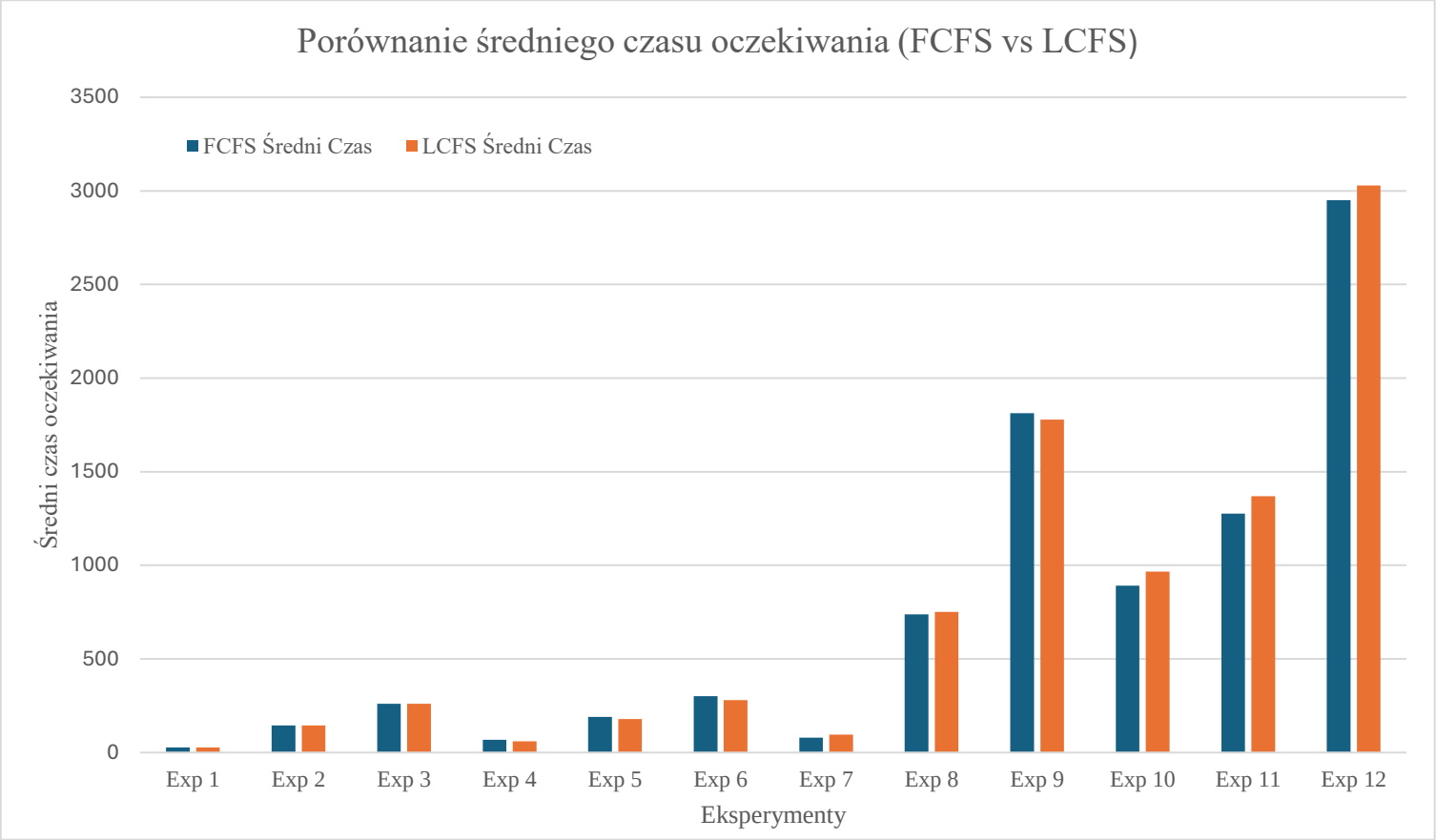
W pliku test_data_7.txt

Wyniki eksperymentu nr 7:

FIFO Błędy braku strony: 14

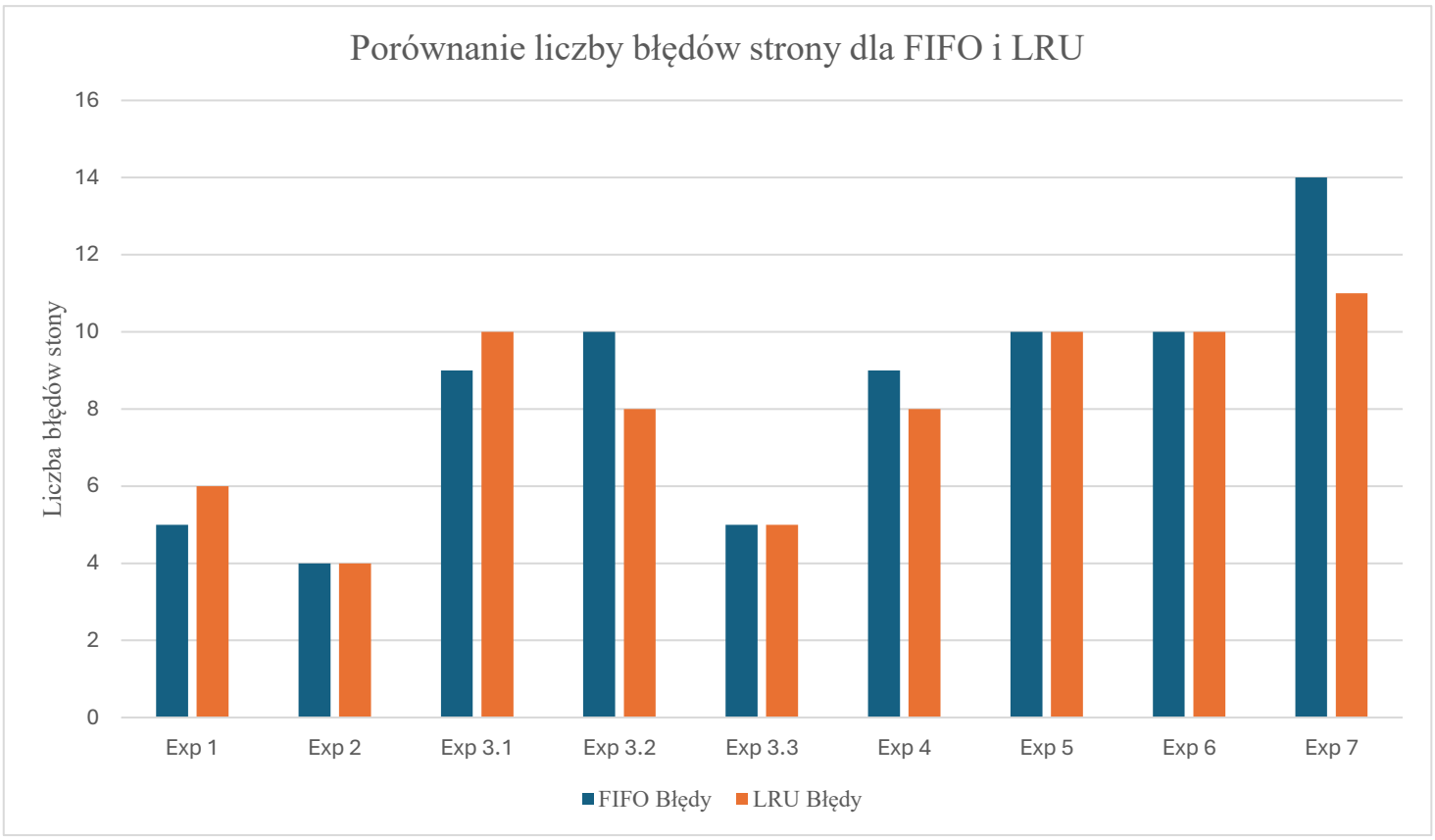
LRU Błędy braku strony: 11

6. Analiza wyników.



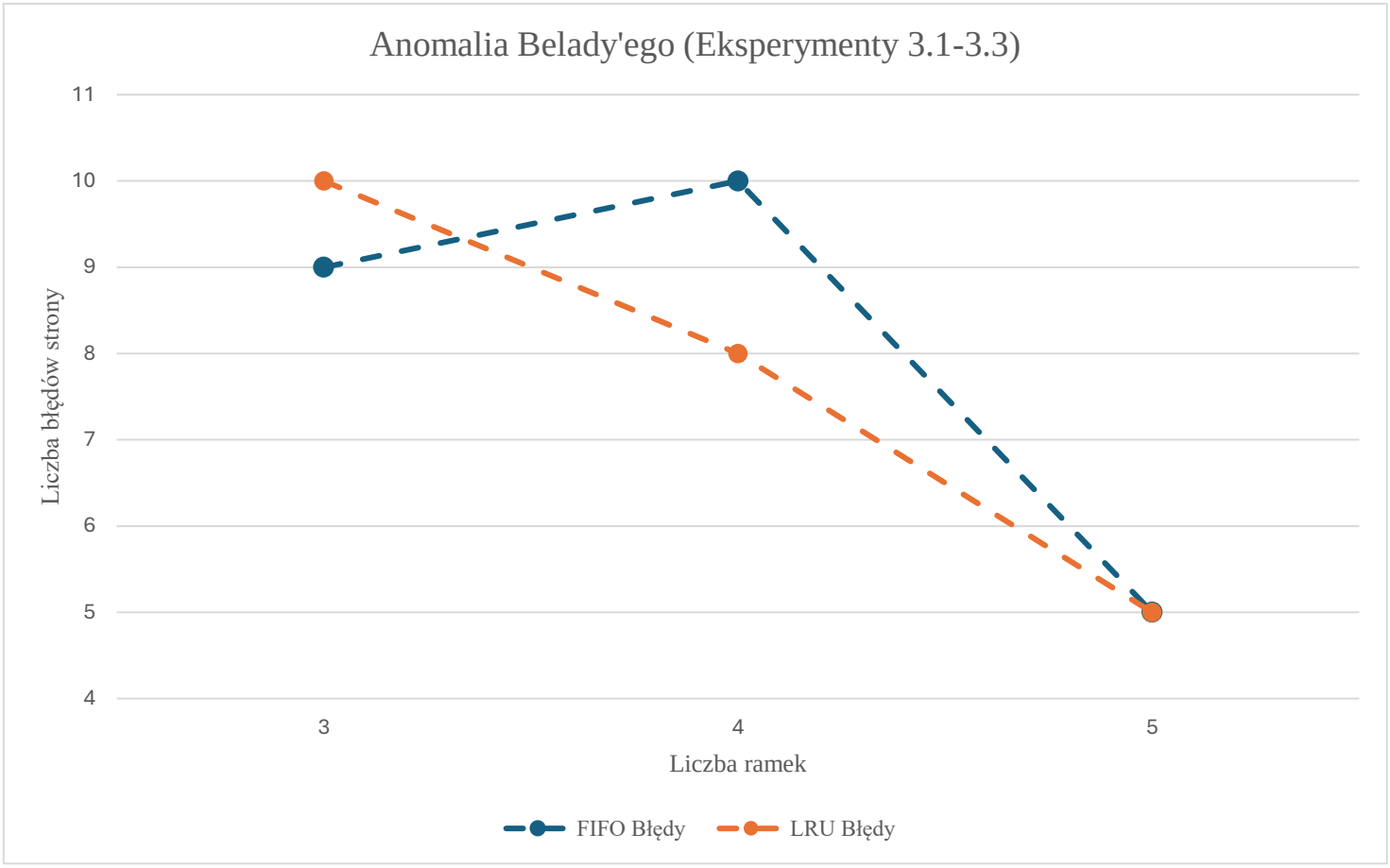
Na podstawie wykresu porównującego średni czas oczekiwania możemy zauważyć, że:

- W eksperymentach 1 do 3 w których czas wykonywania jest stały, a zmienia się jedynie liczba procesów i czas nadejścia średni czas oczekiwania jest taki sam,
- LCFS często daje krótszy średni czas oczekiwania w eksperymentach o stałym czasie nadejścia i różnym czasie wykonania (eksperymenty 4–6), co może wynikać z prioretyzowania najnowszych procesów,
- W eksperymentach z losowymi czasami nadejścia i dużą zmiennością (eksperymenty 10–12) FCFS wykazuje niższe lub zbliżone czasy oczekiwania do LCFS. To sugeruje, że dla systemów o dużym zróżnicowaniu obciążenia LCFS może nie być optymalny.
- Eksperyment 7 (czas nadejścia i wykonania z odchyleniem standardowym) pokazuje, że FCFS daje lepsze wyniki przy mniejszym odchyleniu.



Na podstawie wykresu porównującego liczbę błędów strony możemy zauważyć, że:

- Zgodnie z teorią zwiększanie ilości ramek zmniejsza liczbę błędów braku strony (eksperyment 1 i 2),
- W większości eksperymentów liczba błędów strony dla LRU jest równa lub mniejsza niż dla FIFO, co potwierdza ogólnie lepszą wydajność LRU w środowiskach o wysokiej lokalności odniesienia,
- W eksperymentach 4 i 7 różnica między FIFO a LRU jest szczególnie widoczna. W eksperymencie 7, gdzie testowano powtarzalność jednej strony, LRU osiągnęło znacząco lepszy wynik,
- W eksperymentach 5 i 6 (losowa sekwencja oraz monotoniczny wzrost stron) wyniki dla FIFO i LRU były identyczne, co sugeruje, że lokalność odniesienia nie była tu kluczowym czynnikiem.



Na podstawie wykresu porównującego liczbę błędów strony dla eksperymentów 3.1 do 3.3 można zauważyć że:

- Dla FIFO widoczna jest anomalia Belady’ego – liczba błędów strony rośnie przy zwiększaniu liczby ramek z 3 do 4, a następnie spada dla 5 ramek,
- LRU nie wykazuje anomalii Belady’ego, co potwierdza jego przewagę w takich scenariuszach.

7. Wnioski.

Wydajność algorytmów planowania czasu procesora:

- **FCFS** (First Come First Served) sprawdza się lepiej w systemach o dużej zmienności czasów nadejścia i wykonania procesów, co widoczne jest w eksperymentach 10–12. FCFS wydaje się być bardziej efektywny niż LCFS w większości przypadków testowych.
- **LCFS** (Last Come First Served) jest bardziej efektywny w eksperymentach o stałym czasie nadejścia i różnym czasie wykonania (eksperymenty 4–6), co wynika z jego zdolności do szybkiego przetwarzania nowych procesów, jednak może prowadzić do wydłużonego czasu oczekiwania w systemach z dużą zmiennością.

Wydajność algorytmów zastępowania stron:

- **LRU** (Least Recently Used) wykazuje lepsze wyniki w większości scenariuszy, szczególnie w środowiskach o wysokiej lokalności odniesienia (eksperymenty 4 i 7). Jest bardziej efektywny niż FIFO w redukcji błędów strony.
- **FIFO** (First In First Out) jest prostszy, ale bardziej podatny na anomalię Belady'ego, co pokazują eksperymenty 3.1–3.3.

Anomalia Belady'ego:

- Eksperymenty 3.1–3.3 potwierdzają, że zwiększenie liczby ramek nie zawsze prowadzi do zmniejszenia liczby błędów strony w przypadku FIFO, co czyni go mniej przewidywalnym w zastosowaniach wymagających większej liczby ramek.
- LRU, dzięki adaptacji do lokalności odniesienia, nie wykazuje anomalii Belady'ego, co czyni go bardziej niezawodnym w takich sytuacjach.