Programy symulacyjne

WYBRANE ALGORYTMY PRZYDZIAŁU CZASU PROCESORA I ZASTĘPOWANIA STRON

Wybrane algorytmy przydziału czasu procesora to **FCFS** (First-Come, First-Served) oraz **SJF** (First-Come, First-Served scheduling). Wybrane algorytmy zastępowania stron to **FIFO** (First-In, First-Out) oraz **LRU** (Least Recently Used). Wszystkie algorytmy zostały napisane w języku programowania C++. Każdy z algorytmów ma swój dedykowany plik (.cpp), jednak aby ułatwić testowanie został stworzony dodatkowy plik zawierający wszystkie algorytmy. Dodatkowo został stworzony program zawierający tylko algorytmy przydziału czasu procesora oraz program zawierający tylko algorytmy zastępowania stron.

Spis przygotowanych programów:

- FCFS.cpp
- SJF.cpp
- FIFO.cpp
- LRU.cpp
- ALGORYTMY.cpp
- ALGORYTMY_FCFS_SJF.cpp
- ALGORYTMY_FIFO_LRU.cpp

Każdy algorytm zawiera swój własny plik tekstowy (.txt), który wypełniony jest losowymi liczbami używanymi do obliczeń w konkretnym algorytmie.

Spis przygotowanych plików tekstowych:

- dane.txt
- dane_STRONY.txt

Poniżej przedstawiono struktury plików tekstowych, z których pobierane są dane do działania poszczególnych algorytmów.

Dla algorytmu przydziału czasu procesora zostało przygotowanych 20 000 losowych liczb, oddzielonych „Enter'em" co 200, dodatkowo każda liczba oddzielona jest „spacją".

5 13 14 11 17 7 12 3 7 16 9 1 13 5 10 3 4 8 2 15 5 9 1 19 2 14 9 4 3 17 19 20 13 1 19 16 16 5 1 3 8 13 3 17 6 15 13 5 4 2 7 20 9 14 15 4 12 17 11 4 10 1 14 19 17 18 9 4 13 10 14 13 18 1 12 5 14 3 11 16 15 13 10 12 8 3 2 16 8 7 2 19 9 13 6 16 12 15 9 14 9 18 7 8 18 2 4 7 6 15 12 17 18 2 20 14 2 8 6 3 20 15 13 11 17 17 19 15 4 2 2 19 5 7 11 18 14 12 13 10 4 20 9 19 12 11 5 14 13 7 6 8 3 9 9 1 7 5 8 9 14 3 18 2 5 15 5 12 20 13 17 10 5 17 7 10 17 11 20 13 14 14 5 19 17 11 15 12 4 14 12 10 17 19 13 11 18 5 13 9

Przykład losowo wybranych 200 liczb

Pierwsze 100 liczb oznacza czas przybycia procesu (AT - Arrival Time), drugie 100 liczb oznacza czas obliczeń (BT – Burst Time). Każde 200 liczb należy do jednego ciągu, który zawiera 100 procesów. Takich ciągów jest 100 czyli 200*100=20 000.

Dla algorytmu zastępowania stron zostało przygotowanych 30 300 losowych liczb, oddzielonych „Enter'em" co 101, dodatkowo każda liczba oddzielona jest „spacją".

3 5 13 14 11 17 7 12 3 7 16 9 1 13 5 10 3 4 8 2 15 5 9 1 19 2 14 9 4 3 17 19 20 13 1 19 16 16 5 1 3 8 13 3 17 6 15 13 5 4 2 7 20 9 14 15 4 12 17 11 4 10 1 14 19 17 18 9 4 13 10 14 13 18 1 12 5 14 3 11 16 15 13 10 12 8 3 2 16 8 7 2 19 9 13 6 16 12 15 9 14

Przykład losowo wybranych 101 liczb

Pierwsza liczba oznacza liczbę ramek pamięci, według polecenia należało użyć trzech różnych wartości tego parametru. Dlatego wybrane zostały 3, 5 oraz 7 dla każdych 100 ciągów. Kolejne 100 liczb to numery stron (w przedziale 1-20) które należą do jednego ciągu. Zatem jeden ciąg zawiera 101 liczb i takich ciągów jest 100 dla jednej ramki, a my mamy 3 różne ilości ramek. Zatem 101*100*3=30300.

Każdy z przygotowanych algorytmów po wykonaniu obliczeń przesyła dane do plików tekstowych.

Spis plików tekstowych do zapisu:

- zapis_dane_test_FCFS.txt
- zapis_dane_test_SJF.txt
- zapis_dane_test_FIFO.txt
- zapis_dane_test_LRU.txt

Uwaga przed uruchomieniem programu należy upewnić się, że pliki do zapisu zostały usunięte lub wyczyszczone.

Procedura testowania

FCFS

Dane wczytane za pomocą funkcji „ifstream” zostają wczytane w sposób opisany powyżej. Pętla „while” jest odpowiedzialna za sterowanie całym algorytmem kiedy w pliku napotkamy na znak końca pliku algorytm/program się zakończy. Następnie za pomocą funkcji „stable_sort()” sortujemy całą tablice struktur względem zmiennej AT. Kolejnym krokiem jest przeprowadzenie obliczeń dla jednego ciągu. Po przekierowaniu wszystkiego do pliku lub na ekran, zerujemy odpowiednie zmienne używane do obliczeń i powtarzamy cały proces aż wykonamy 100 ciągów.

SJF

Dane wczytane za pomocą funkcji „ifstream” zostają wczytane w sposób opisany powyżej. Pętla „while” jest odpowiedzialna za sterowanie całym algorytmem kiedy w pliku napotkamy na znak końca pliku algorytm/program się zakończy. Kolejnym krokiem jest przeprowadzenie obliczeń dla jednego ciągu, które zamknięte zostają w kolejnej pętli „while”. To wewnątrz niej wyszukiwany jest najkrócej trwający proces, biorąc pod uwagę czas przybycia procesu. Po przekierowaniu wszystkiego do pliku lub na ekran, zerujemy odpowiednie zmienne używane do obliczeń i powtarzamy cały proces aż wykonamy 100 ciągów.

FIFO

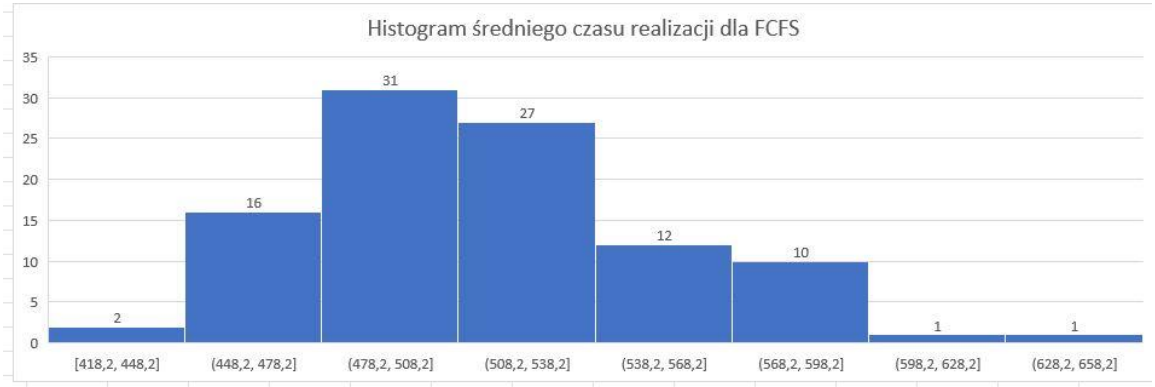
Dane wczytane za pomocą funkcji „ifstream” zostają wczytane w sposób opisany powyżej. Pętla „while” jest odpowiedzialna za sterowanie całym algorytmem kiedy w pliku napotkamy na znak końca pliku algorytm/program się zakończy. Następnie wypisujemy nagłówki rubryk w naszej tabeli, robimy to teraz ponieważ obliczenia nie są zapisywane na stałe i będą się zmieniać. Kolejnym krokiem jest przeprowadzenie obliczeń dla jednego ciągu, które zamknięte zostają w kolejnej pętli „while”.To wewnątrz niej wyszukiwana jest strona która przybyła najpóźniej i zastępowana nową stroną. Po przekierowaniu wszystkiego do pliku lub na ekran, zerujemy odpowiednie zmienne używane do obliczeń i powtarzamy aż wykonamy 100 ciągów.

LRU

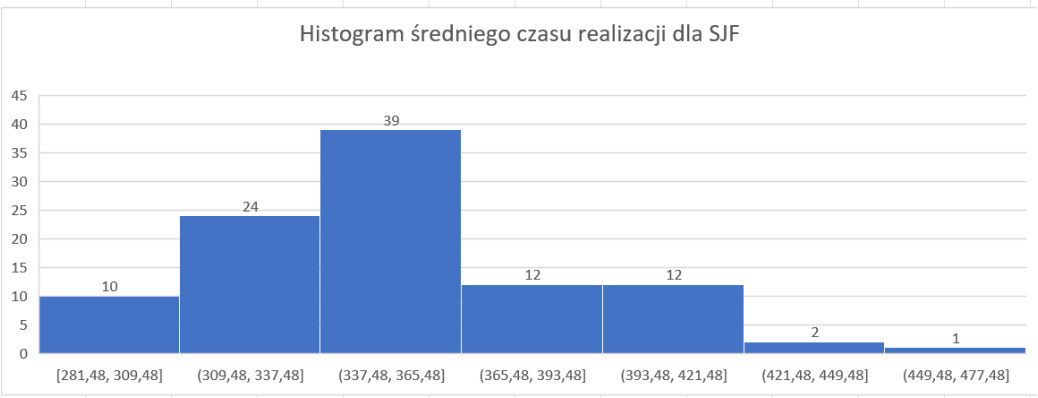
Dane wczytane za pomocą funkcji „ifstream” zostają wczytane w sposób opisany powyżej. Pętla „while” jest odpowiedzialna za sterowanie całym algorytmem kiedy w pliku napotkamy na znak końca pliku algorytm/program się zakończy. Następnie wypisujemy nagłówki rubryk w naszej tabeli, robimy to teraz ponieważ obliczenia nie są zapisywane na stałe i będą się zmieniać. Kolejnym krokiem jest przeprowadzenie obliczeń dla jednego ciągu, które zamknięte zostają w kolejnej pętli „while”. To wewnątrz niej wyszukiwana jest strona która była najdawniej używana i zastępowana nową stroną. Po przekierowaniu wszystkiego do pliku lub na ekran, zerujemy odpowiednie zmienne używane do obliczeń i powtarzamy aż wykonamy 100 ciągów.

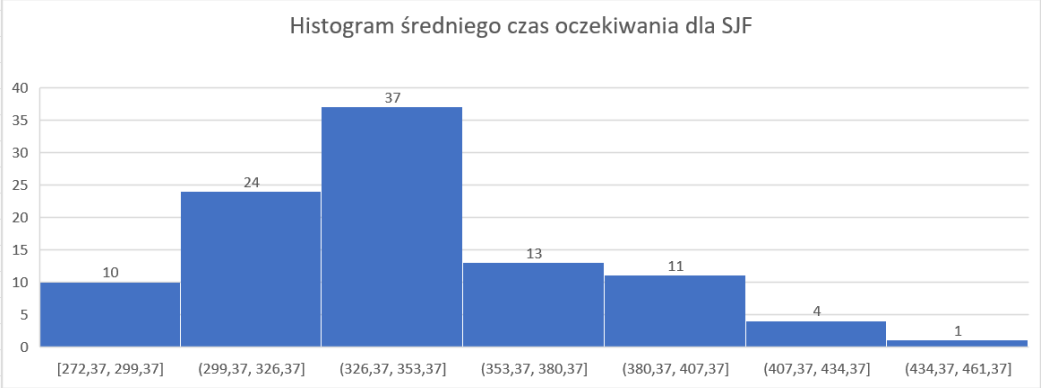
Opracowane wyniki eksperymentów

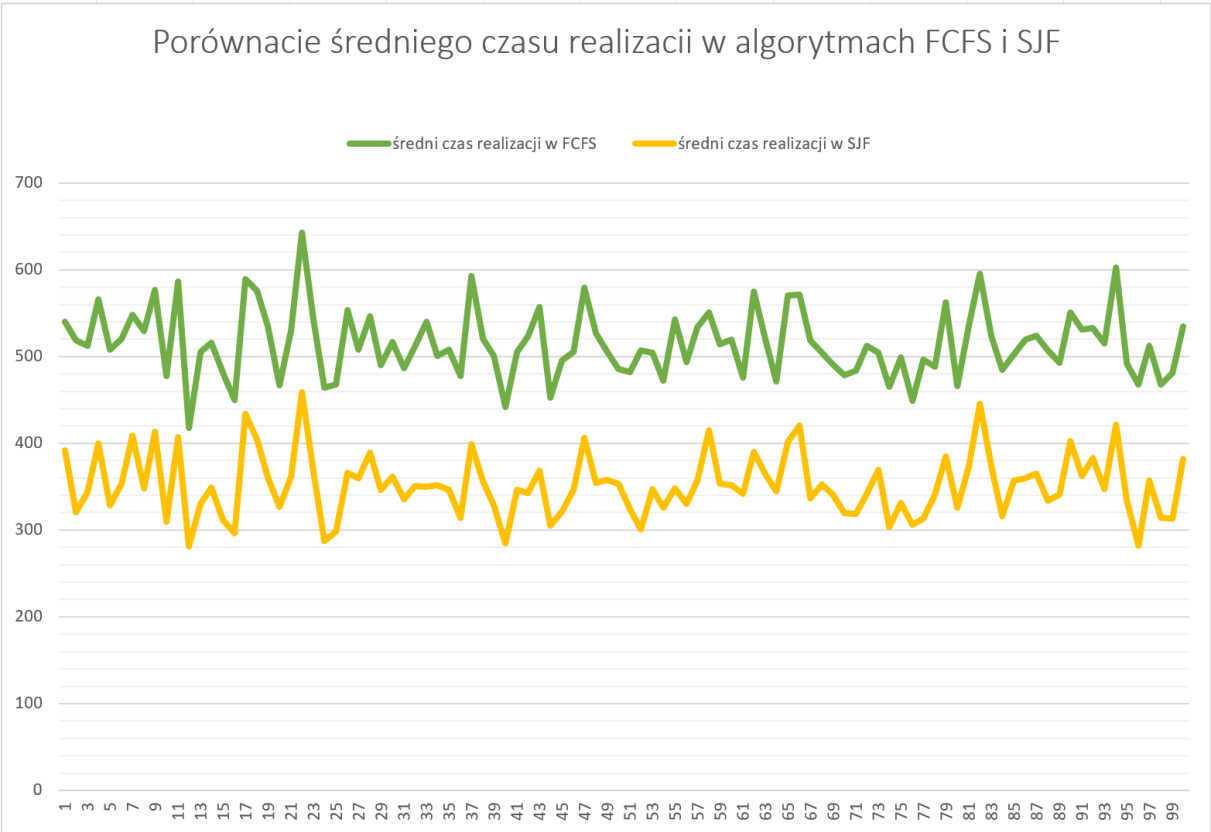
FCFS



SJF







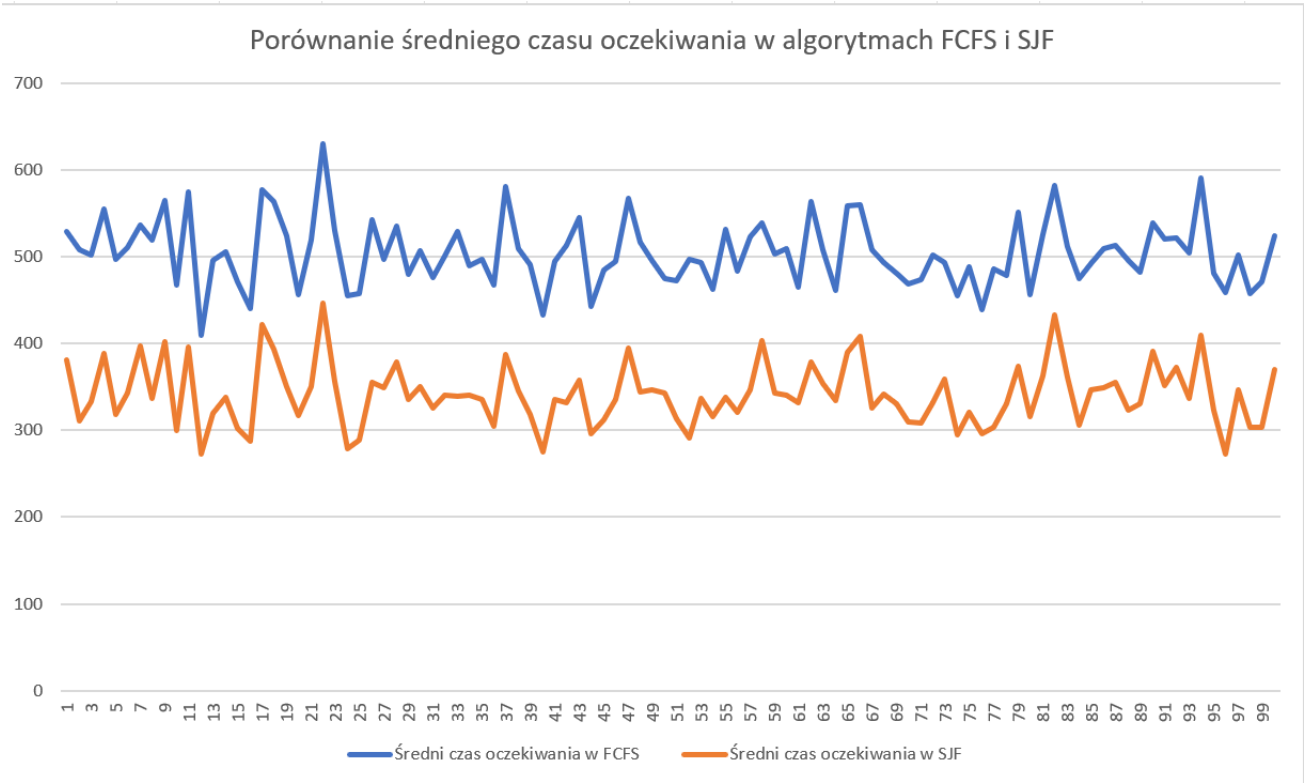
średni czas realizacji 100 ciągów dla FCFS = 514,8821

Odchylenie standardowe średniego czasu realizacji dla FCFS = 39,12

średni czas realizacji 100 ciągów dla SJF = 352,0095

Odchylenie standardowe średni czas realizacji dla SJF = 36,31

Różnica =37,58%



średni czas oczekiwania 100 ciągów dla FCFS = 504,4912

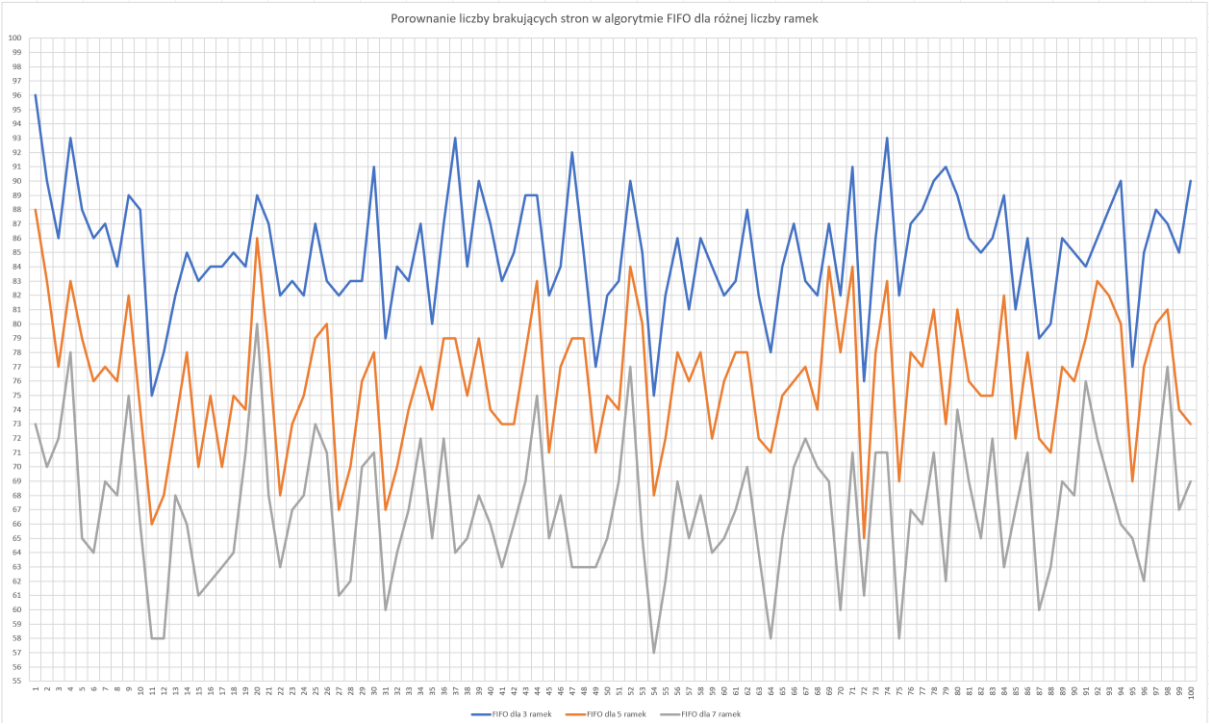
Odchylenie standardowe średniego czasu oczekiwania dla FCFS = 38,49

średni czas oczekiwania 100 ciągów dla SJF = 341,6186

Odchylenie standardowe średniego czasu oczekiwania dla SJF= 35,64

Różnica =38,5%

FIFO



Średnia liczba brakujących stron dla 3 ramek = 85,07

Odchylenie standardowe liczby brakujących stron dla 3 ramek w FIFO= 4,05

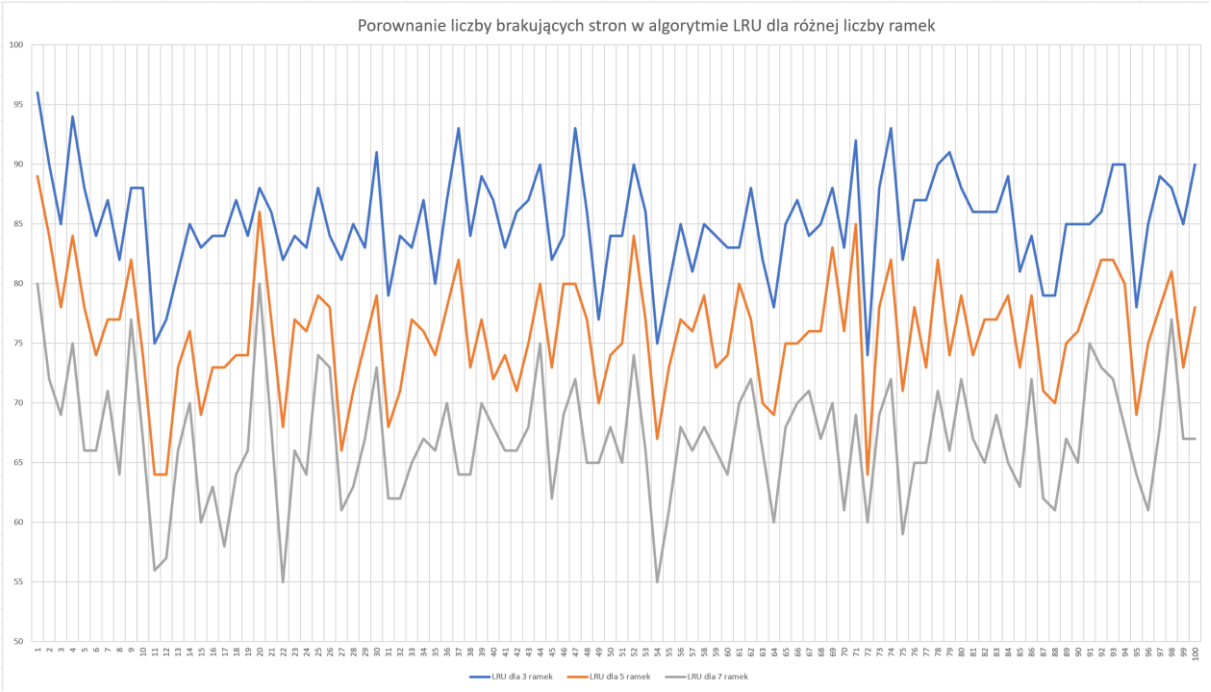
Średnia liczba brakujących stron dla 5 ramek = 76,03

Odchylenie standardowe liczby brakujących stron dla 5 ramek w FIFO= 4,63

Średnia liczba brakujących stron dla 7 ramek = 67,06

Odchylenie standardowe liczby brakujących stron dla 7 ramek w FIFO= 4,72

LRU



Średnia liczba brakujących stron dla 3 ramek = 85,17

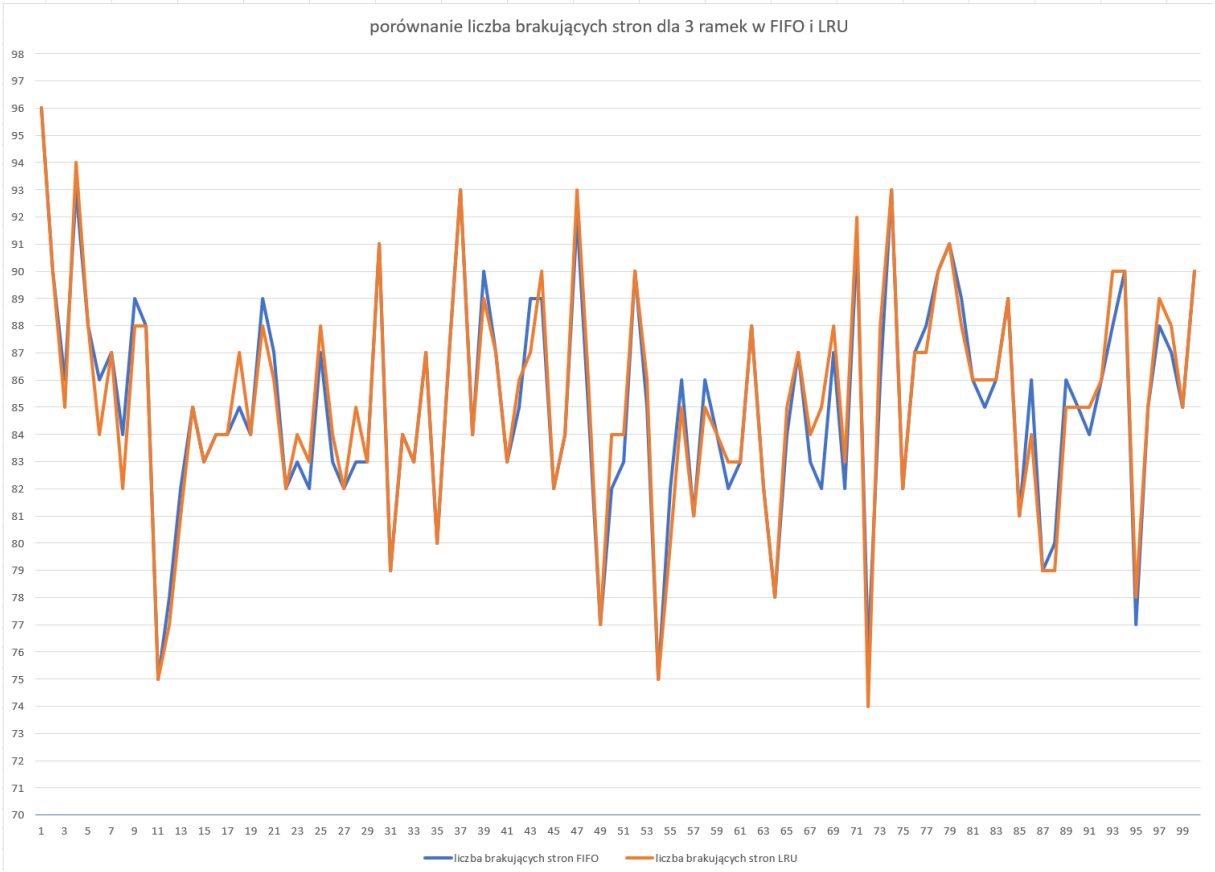
Odchylenie standardowe liczby brakujących stron dla 3 ramek w LRU = 4,15

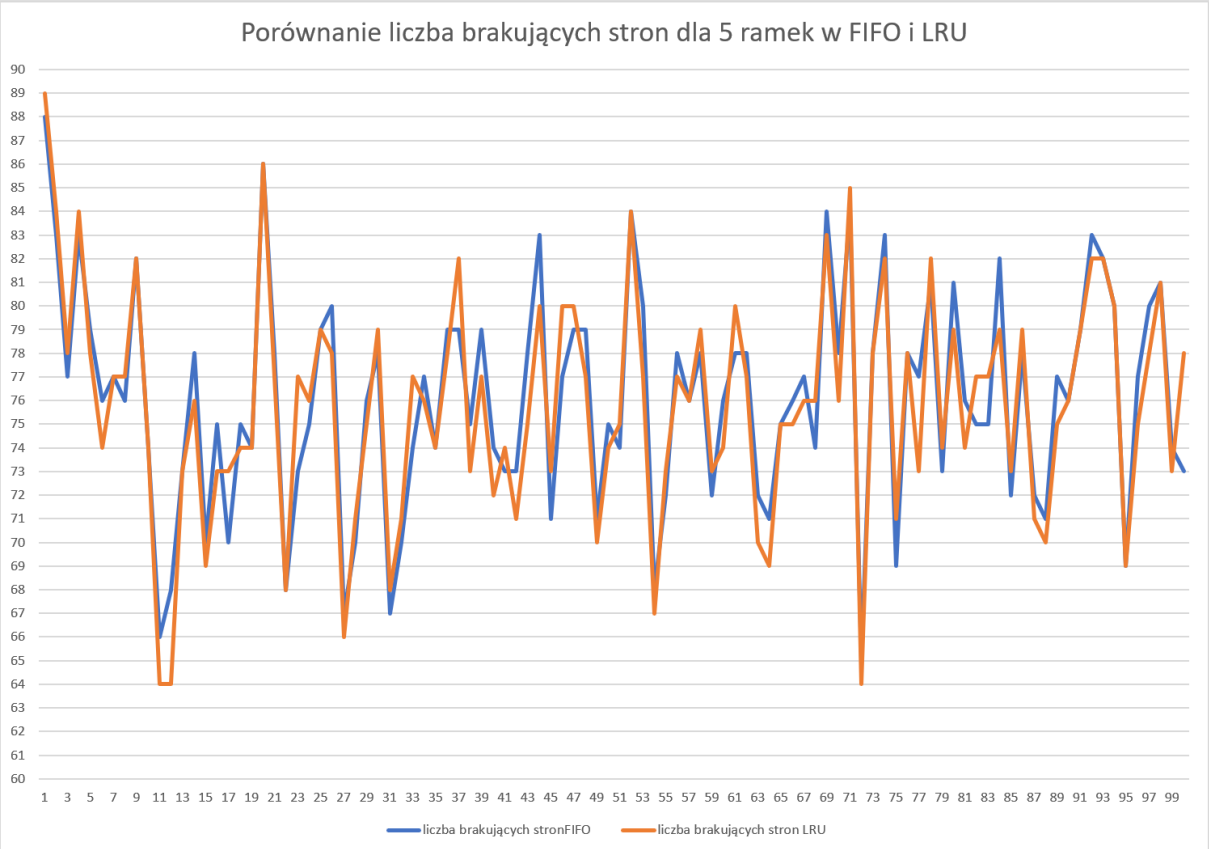
Średnia liczba brakujących stron dla 5 ramek = 75,77

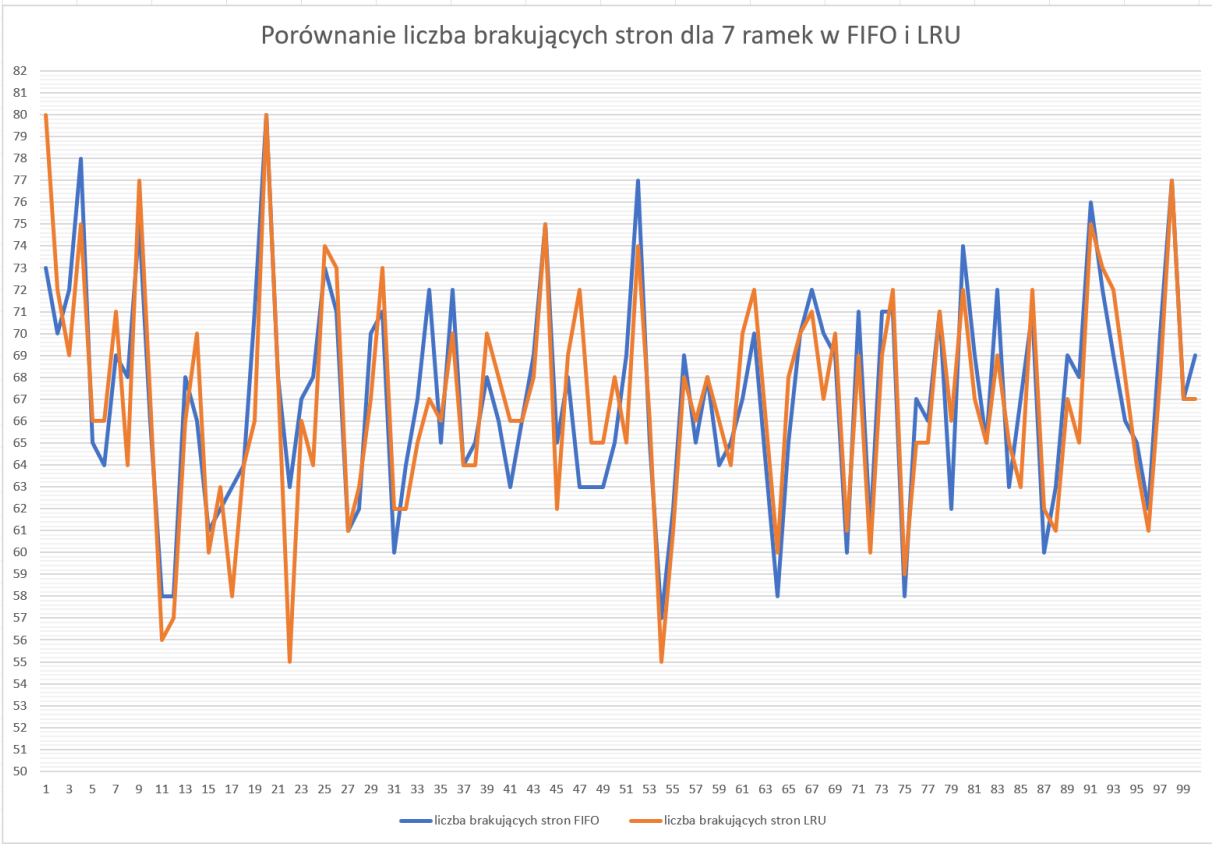
Odchylenie standardowe liczby brakujących stron dla 5 ramek w LRU = 4,78

Średnia liczba brakujących stron dla 7 ramek = 66,89

Odchylenie standardowe liczby brakujących stron dla 7 ramek w LRU = 4,98


porównanie liczba brakujących stron dla 3 ramek w FIFO i LRU
98
97
96
95
94
93
92
91
90
89
88
87
86
85
84
83
82
81
80
79
78
77
76
75
74
73
72
71
70
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
liczba brakujących stron FIFO
liczba brakujących stron LRU


Porównanie liczba brakujących stron dla 5 ramek w FIFO i LRU
90
89
88
87
86
85
84
83
82
81
80
79
78
77
76
75
74
73
72
71
70
69
68
67
66
65
64
63
62
61
60
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
liczba brakujących stronFIFO
liczba brakujących stron LRU


Porównanie liczba brakujących stron dla 7 ramek w FIFO i LRU
liczba brakujących stron FIFO
liczba brakujących stron LRU

Wnioski

FCFS kontra SJF

- Porównując średni czas realizacji w obu algorytmach możemy zauważyć, że jest on niższy w algorytmie SJF,
- Analogicznie po porównaniu średniego czasu oczekiwania zauważamy, że jest on niższy w algorytmie SJF,
- Dodatkowo, na wykresach zestawiających średnie czasy realizacji (100 ciągów) obu algorytmów, także możemy zauważyć korzyść algorytmu SJF,
- Analogiczną sytuację możemy zauważyć dla wykresu średniego czasu realizacji, gdzie również widoczna przewaga SJF,
- Z przeprowadzonych obliczeń wynika, że różnica średniego czasu realizacji oraz średniego czasu oczekiwania w obu przypadkach **wynosi prawie 40%,**
- Warto zwrócić uwagę na odchylenia standardowe, które pokazują przewagę algorytmu SJF

Biorąc pod uwagę powyższe argumenty oraz przedstawione wykresy i wyliczenia możemy jednogłośnie wybrać lepszy algorytm, którym jest SJF. Argumenty takie jak większa efektywność, krótszy czas pracy, ekonomiczność wzmacniają przekonanie, że to właśnie SJF jest lepszym algorytmem.

FIFO kontra LRU

- Patrząc na poszczególne średnie brakujących stron dla konkretnej liczby ramek, możemy zauważyć, że wraz ze zwiększaniem liczby ramek spada średnia brakujących stron niezależnie od wybranego algorytmu,
- Początkowo różnice pomiędzy algorytmami nie są zauważalne. Kiedy używamy 3 ramek wykresy FIFO i LRU pokrywają się w dużym stopniu. Dla 5 ramek możemy zauważyć pewne różnice, z kolei dla 7 ramek różnice są już mocno zauważalne. Z tego powodu możemy uważać, że dla mniejszej ilości ramek algorytmy działają podobnie, jednak dla większej ilości ramek mogą się znacznie różnić,
- Wraz ze wzrostem ramek zmniejsza się największa liczba brakujących stron, a różnica pomiędzy największą a najmniejszą liczbą brakujących stron jest mniejsza w algorytmie FIFO,
- Warto powiedzieć, że LRU nie jest podatny na anomalię Belady'ego, podczas gdy FIFO jest,
- W algorytmie LRU odchylenie standardowe jest większe względem odchylenia dla algorytmu w FIFO

Biorąc pod uwagę powyższe argumenty oraz przedstawione wykresy i wyliczenia nie możemy jednogłośnie wybrać lepszego algorytmu. W zależności od wprowadzonych danych FIFO i LRU mogą pokonywać się wzajemnie w każdym ciągu. Podsumowując, nie możemy wskazać wyraźnej przewagi jednego algorytmu nad drugim.