Politechnika Wrocławska
Wydział Informatyki i Telekomunikacji
Cyberbezpieczeństwo
Rok akademicki 2021/22
Semestr zimowy

Systemy Operacyjne

Sprawozdanie z projektu

Symulacje oraz porównanie wybranych algorytmów planowania czasu procesora oraz zastępowania stron

Prowadzący:

Mgr inż. Jakub Klikowski

Wykonał:

Patryk Fidler 259468

Narzędzia użyte w przygotowaniu projektu:

- Python 3.10.0 64-bit
- Biblioteki: Numpy, Scipy
- Microsoft Visual Studio Code
- Microsoft Word

Symulator

Opis działania:

Symulator jest napisany w języku Python, dlatego aby uruchomić symulator konieczne jest zainstalowanie Pythona w wersji przynajmniej 3.6. Dodatkowo z racji użycia bibliotek Numpy i Scipy je też należy mieć zainstalowane.

Program jest wstanie symulować algorytmy planowania pracy procesora: FCFS, SJF oraz wymiany stron: FIFO, LRU. Implementacja LFU zawiera błędy, dlatego odradza się jej wybieranie.

Program można uruchomić wywołując go z konsoli nazwą main.py, będąc w odpowiednim folderze. Przy wywołaniu należy podać interesujące nas parametry, które można obejrzeć komendą main.py –help.

```
usage: main.py [-h] [-p PATH] [-m MEAN] [-s SCALE] [-f FRAMES] [--random] [--upp UPP] [--zeroes] [--FCFS] [--SJF]
               [--FIFO] [--LRU] [--LFU]

Symulator algorytmów planowania czasu procesora or zastępowania stron. Gdy nie ma wybranej żadnej opcji symuluje FCFS

options:
  -h, --help            show this help message and exit
  -p PATH, --path PATH  Podaj ścieżkę pliku źródłowego lub liczbę procesów do wygenerowania (domyślnie = 20)
  -m MEAN, --mean MEAN  Przekaż do generatora średni czas wykonania (domyślnie = 20)
  -s SCALE, --scale SCALE
                        Przekaż do generatora odchylenie standardowe (domyślnie = 5)
  -f FRAMES, --frames FRAMES
                        Podaj ilość ramek dla algorytmów zastępowania stron (domyślnie = 5)
  --random              Generuj procesy losowo, domyślnie generuje procesy w rozkładzie normalnym
  --upp UPP             Podaj górną granicę generowanych danych (domyślnie taka jak liczba procesów do generowania)
  --zeroes              Generuj procesy o czasie przyjścia równym 0
  --FCFS                Symuluj algorytm First Come First Serve
  --SJF                 Symuluj algorytm Shortest Job First
  --FIFO                Symuluj algorytm First In First Out
  --LRU                 Symuluj algorytm Least Recently Used
  --LFU                 Symuluj algorytm Least Frequently Used --- ZBUGOWANE, UŻYWAĆ NA WŁASNĄ ODPOWIEDZIALNOŚĆ!!!
```

Poprzez przekazywane parametry można wybrać wykonywany algorytm, podać ścieżkę do pliku z danymi procesów, lub wybrać parametry generowania procesów.

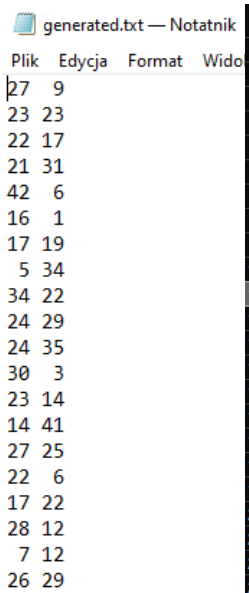
Przykładowe działanie dla domyślnych wartości:

```
C:\Users\fidle\OneDrive\Dokumenty\PWR\Systemy operacyjne\projekt\simulation_project>main.py
Czas nadejścia    Czas wykonywania  Czas zakończenia  Czas realizacji   Czas oczekiwania
1                 13                14                13                0
3                 44                58                55                11
8                 12                70                62                50
9                 23                93                84                61
11                9                 102               91                82
11                29                131               120               91
11                25                156               145               120
14                8                 164               150               142
15                13                177               162               149
16                8                 185               169               161
16                20                205               189               169
18                14                219               201               187
20                9                 228               208               199
21                46                274               253               207
21                25                299               278               253
22                30                329               307               277
26                20                349               323               303
32                5                 354               322               317
33                31                385               352               321
39                7                 392               353               346
Średni czas realizacji: 191.85
Średni czas oczekiwania: 172.3
```

Przykładowe działanie z podanymi parametrami i wybranym algorytmem wymiany stron:

```
C:\Users\fidle\OneDrive\Dokumenty\PWR\Systemy operacyjne\projekt\simulation_project>main.py --random --FIFO -p 20
Lista odwołań: [28, 4, 23, 7, 31, 23, 37, 24, 24, 12, 34, 19, 22, 17, 22, 22, 15, 13, 29, 27]
Szukam: 28 --> [None, None, None, None, 28]
Szukam: 4 --> [None, None, None, 28, 4]
Szukam: 23 --> [None, None, 28, 4, 23]
Szukam: 7 --> [None, 28, 4, 23, 7]
Szukam: 31 --> [28, 4, 23, 7, 31]
Szukam: 23 --> [28, 4, 23, 7, 31]
Szukam: 37 --> [4, 23, 7, 31, 37]
Szukam: 24 --> [23, 7, 31, 37, 24]
Szukam: 24 --> [23, 7, 31, 37, 24]
Szukam: 12 --> [7, 31, 37, 24, 12]
Szukam: 34 --> [31, 37, 24, 12, 34]
Szukam: 19 --> [37, 24, 12, 34, 19]
Szukam: 22 --> [24, 12, 34, 19, 22]
Szukam: 17 --> [12, 34, 19, 22, 17]
Szukam: 22 --> [12, 34, 19, 22, 17]
Szukam: 22 --> [12, 34, 19, 22, 17]
Szukam: 15 --> [34, 19, 22, 17, 15]
Szukam: 13 --> [19, 22, 17, 15, 13]
Szukam: 29 --> [22, 17, 15, 13, 29]
Szukam: 27 --> [17, 15, 13, 29, 27]
Liczba wymian strony: 16
```

Symulator automatycznie generuje pliki z wygenerowanymi danymi wejściowymi oraz rezultatami symulacji. Przykładowe zawartości plików:



generated.txt — Notatnik

Plik Edycja Format Widok

```
27  9
23 23
22 17
21 31
42  6
16  1
17 19
 5 34
34 22
24 29
24 35
30  3
23 14
14 41
27 25
22  6
17 22
28 12
 7 12
26 29
```

	A	B	C	D	E
1	Czas nadejścia	Czas wykonywania	Czas zakończenia	Czas realizacji	Czas oczekiwania
2	0	10	10	10	0
3	2	18	28	26	8
4	4	15	43	39	24
5	4	9	52	48	39
6	4	30	82	78	48
7	4	16	98	94	78
8	5	27	125	120	93
9	7	16	141	134	118
10	11	20	161	150	130
11	12	24	185	173	149
12	13	24	209	196	172
13	13	15	224	211	196
14	13	23	247	234	211
15	13	10	257	244	234
16	15	23	280	265	242
17	16	23	303	287	264
18	17	21	324	307	286
19	17	20	344	327	307
20	18	22	366	348	326
21	19	14	380	361	347
22	Średni czas realizacji:	182.6			
23	Średni czas oczekiwania:	163.6			
24					

Eksperymenty

Algorytmy planowania czasu procesora

First Come First Serve(FCFS)

First Come First Serve jest najprostszym algorytmem planowania czasu procesora. Jak sama nazwa wskazuje polega on po prostu na kolejkowaniu procesów zgodnie z ich czasem przyjścia. Jest oparty na zasadzie First In First Out(FIFO) i tej zasady użyłem też w implementacji w formie kolejki. Działanie może zostać znacznie spowolnione gdy procesy które przyszły pierwsze mają długi czas wykonywania.

Przykładowe działanie FCFS:

```
C:\Users\fidle\OneDrive\Dokumenty\PWR\Systemy operacyjne\projekt\simulation_project>main.py --FCFS -p 20
Czas nadejścia    Czas wykonywania  Czas zakończenia  Czas realizacji   Czas oczekiwania
0                 10                10                10                0
5                 26                36                31                5
5                 39                75                70                31
10                26                101               91                65
10                10                111               101               91
14                25                136               122               97
17                10                146               129               119
18                13                159               141               128
20                17                176               156               139
23                7                 183               160               153
24                10                193               169               159
25                21                214               189               168
26                43                257               231               188
29                21                278               249               228
30                28                306               276               248
30                36                342               312               276
32                8                 350               318               310
38                18                368               330               312
41                15                383               342               327
41                5                 388               347               342
Średni czas realizacji: 188.7
Średni czas oczekiwania: 169.3
```

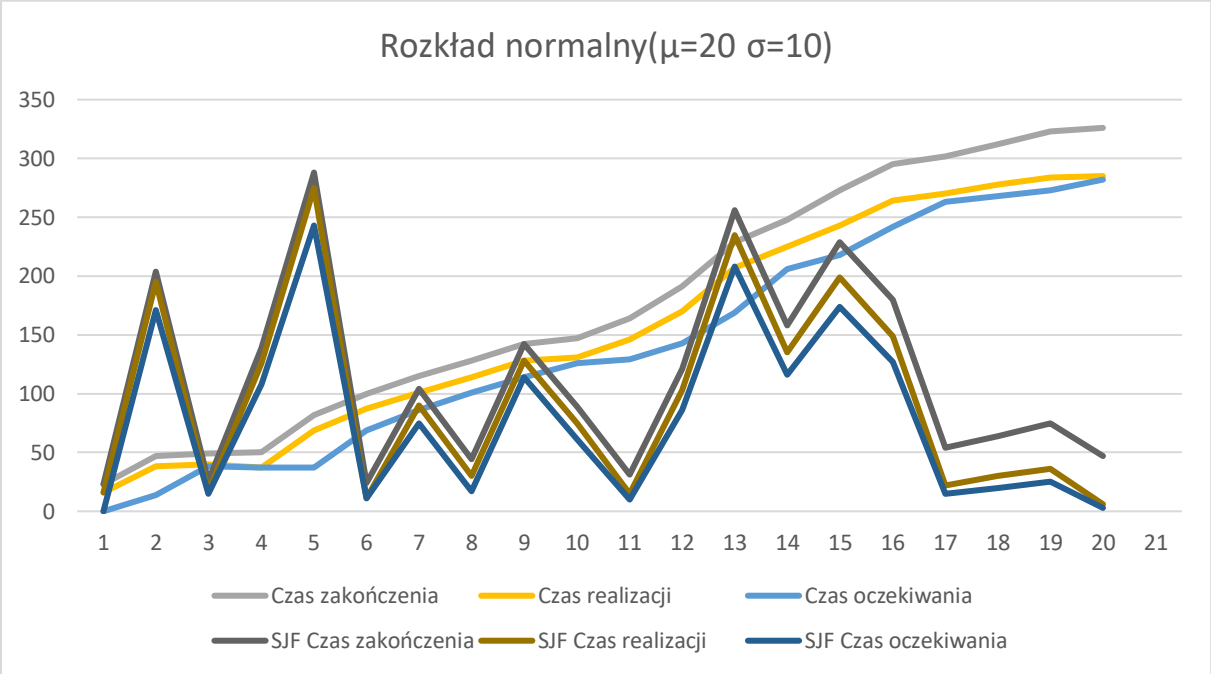
Shortest Job First(SJF)

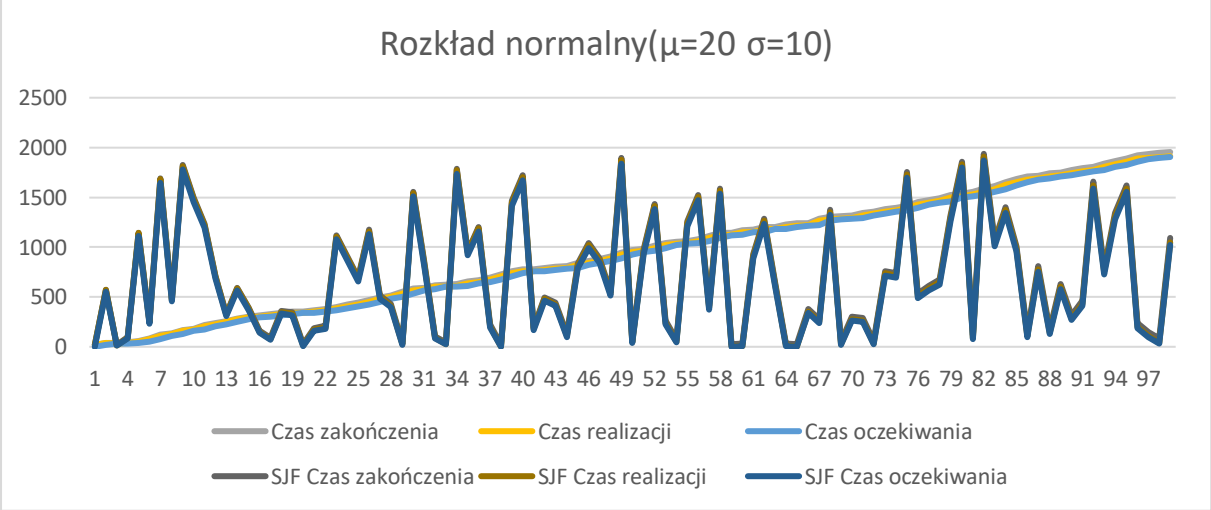
Jest algorytmem w mojej implementacji nie wywłaszczeniowym. Jego największą zaletą jest maksymalne optymalizowanie czasu oczekiwania. Procesy są sortowane w kolejce zgodnie z czasem ich wykonywania.

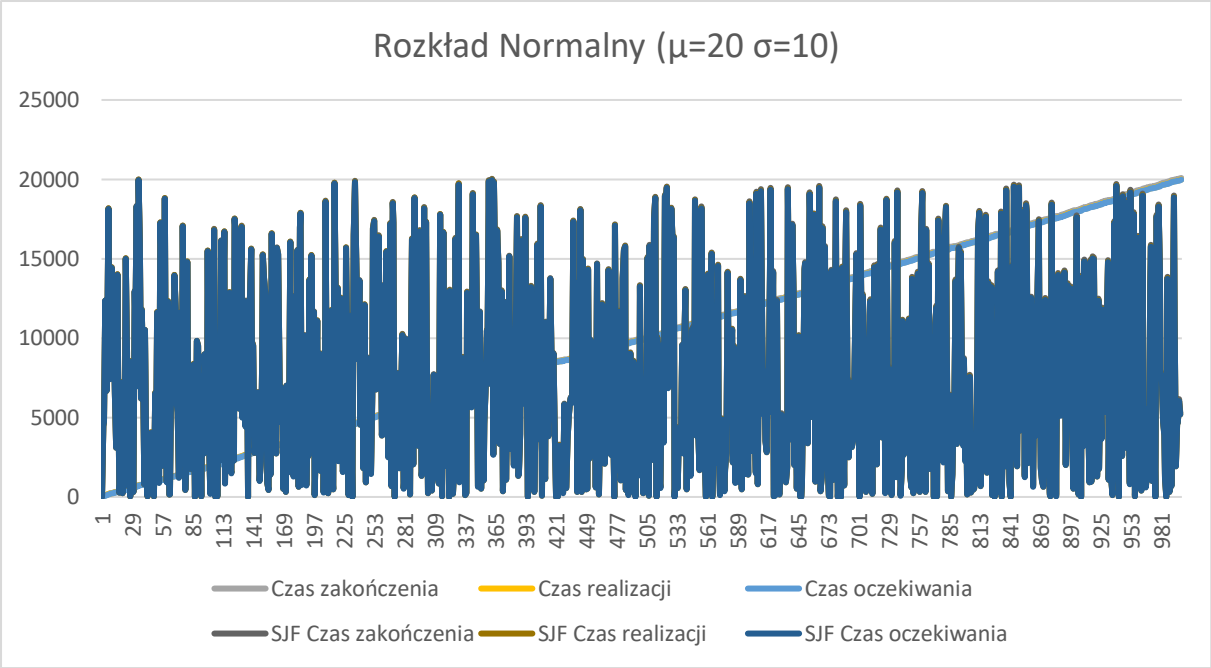
Przykładowe działanie SJF:

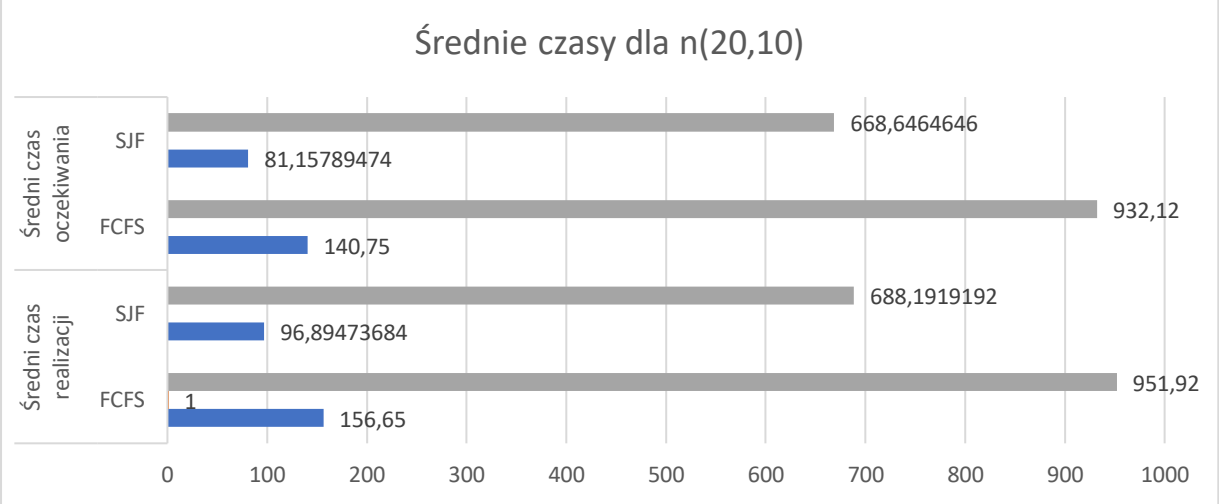
```
C:\Users\fidle\OneDrive\Dokumenty\PWR\Systemy operacyjne\projekt\simulation_project>main.py --SJF
    Czas nadejścia  Czas wykonywania  Czas zakończenia   Czas realizacji  Czas oczekiwania
                 7                45                52                45                 0
                 8                 4                56                48                44
                 9                40               455               446               406
                11                33               415               404               371
                12                18               118               106                88
                14                28               265               251               223
                17                 6                62                45                39
                19                 8                70                51                43
                19                29               294               275               246
                21                25               211               190               165
                22                23               186               164               141
                23                22               140               117                95
                23                26               237               214               188
                24                16               100                76                60
                26                29               323               297               268
                26                30               382               356               326
                27                23               163               136               113
                30                29               352               322               293
                35                14                84                49                35
Średni czas realizacji: 189.05263157894737
Średni czas oczekiwania: 165.47368421052633
```

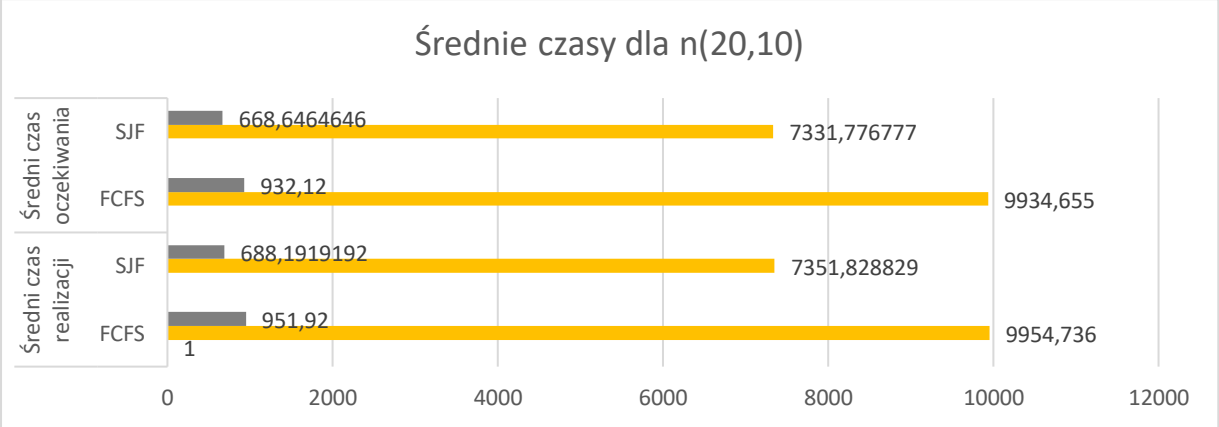
Porównanie dla różnych danych



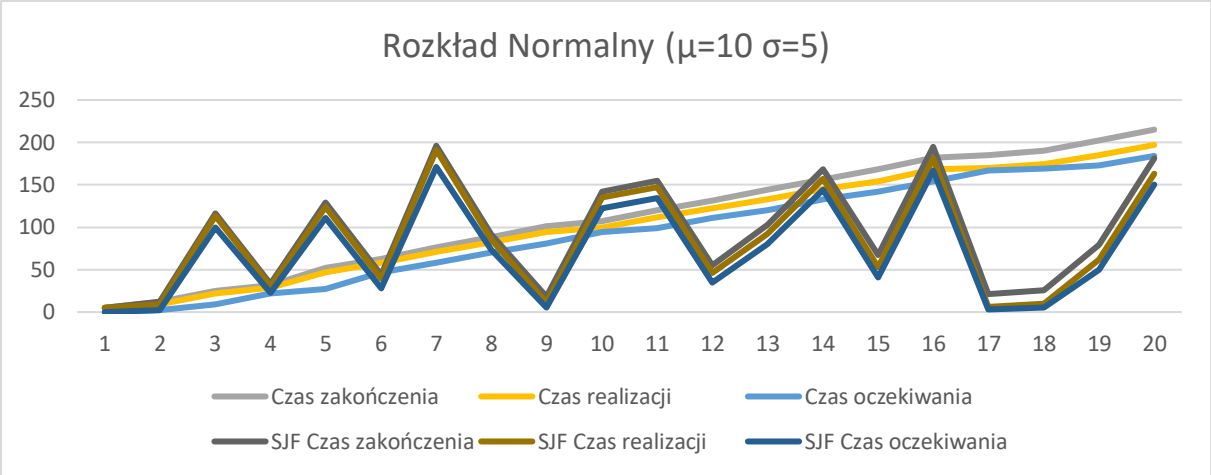


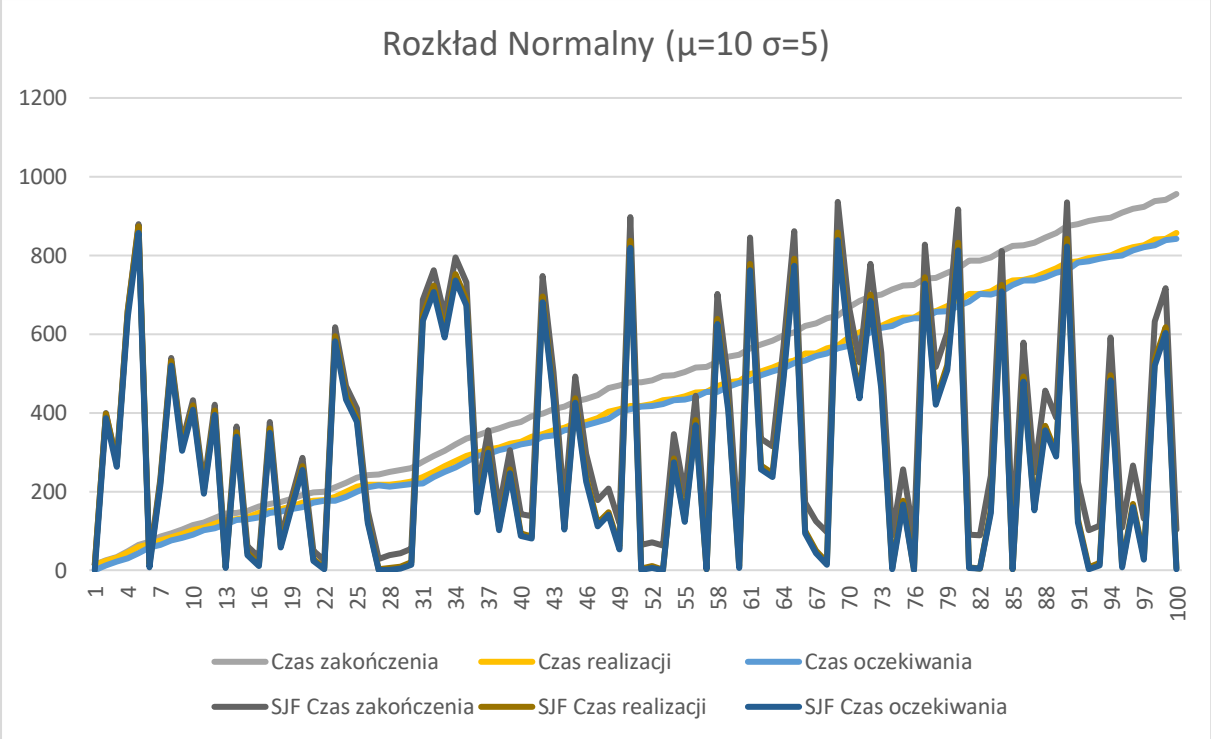


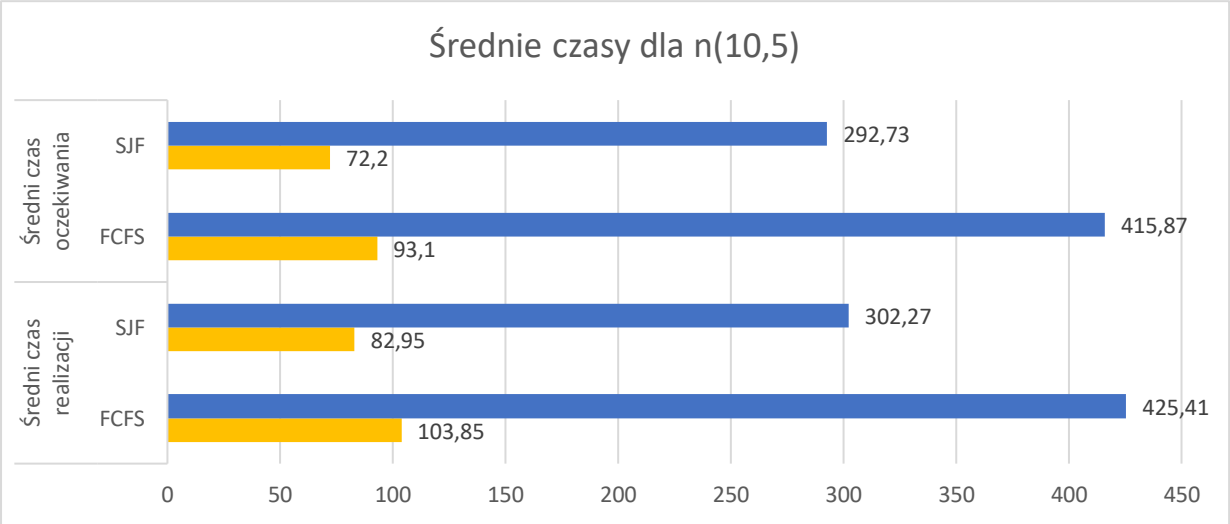




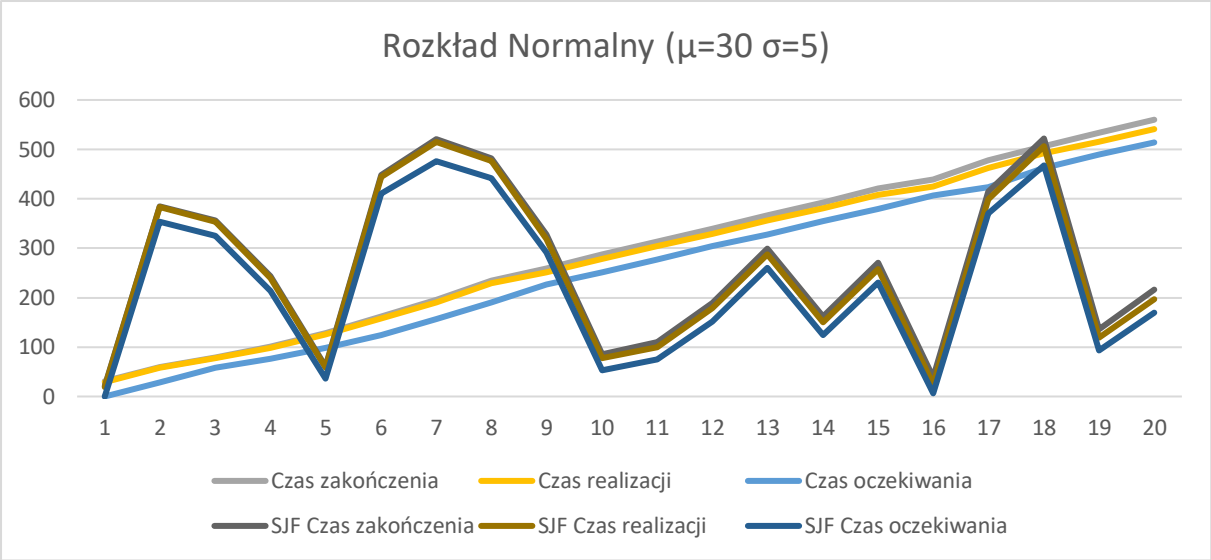
Efektywność SJF jest ok. 27% lepsza!

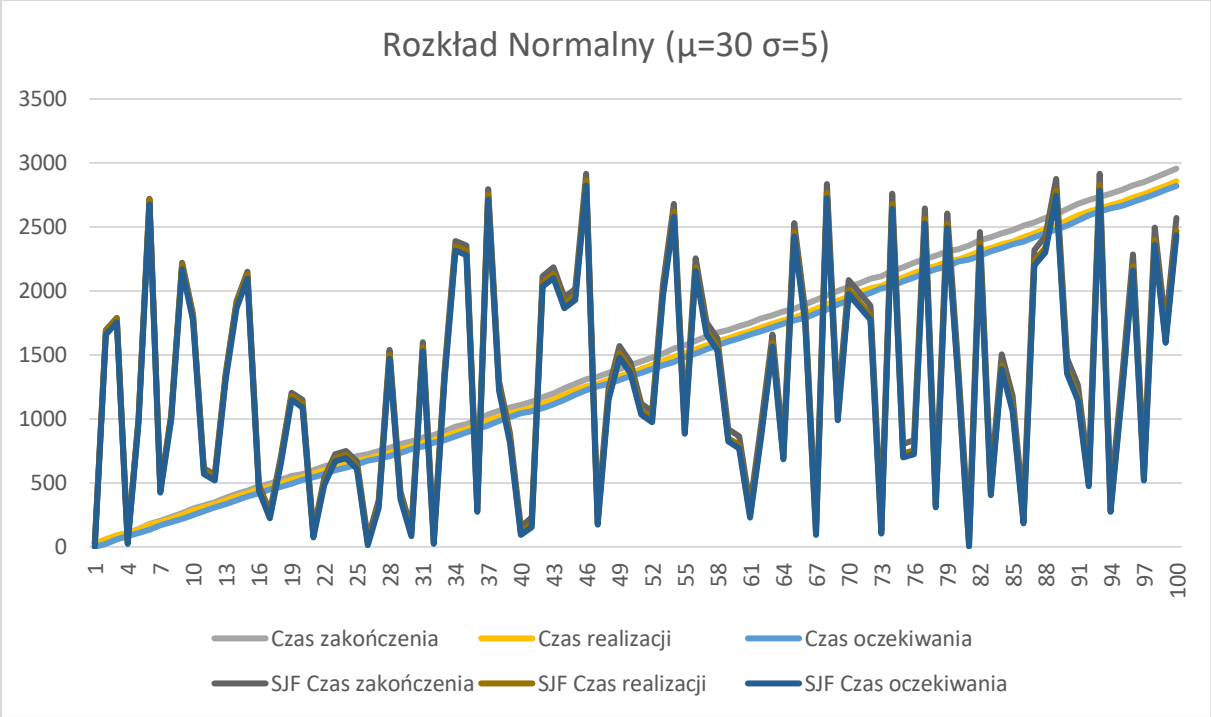


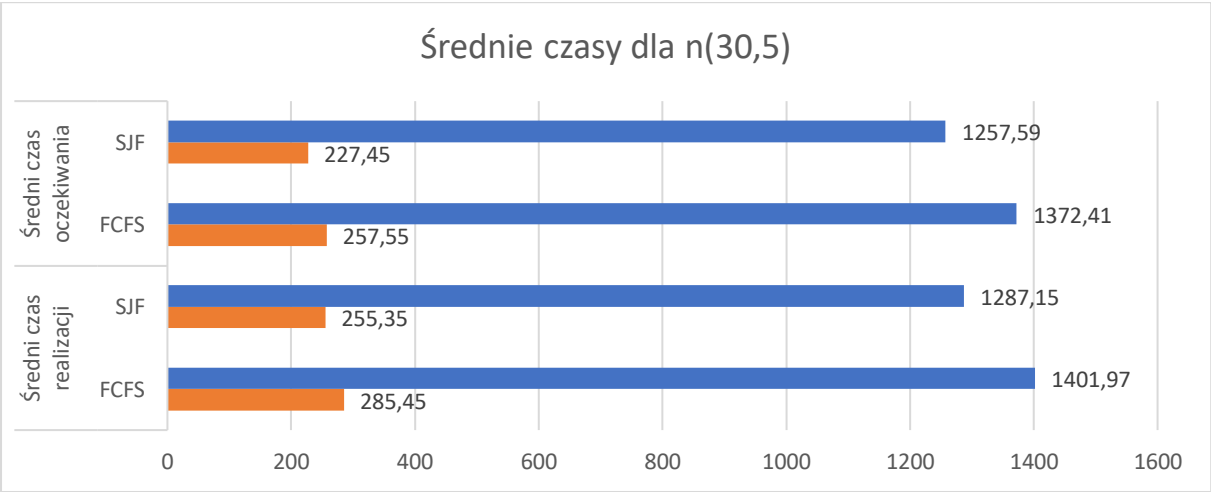




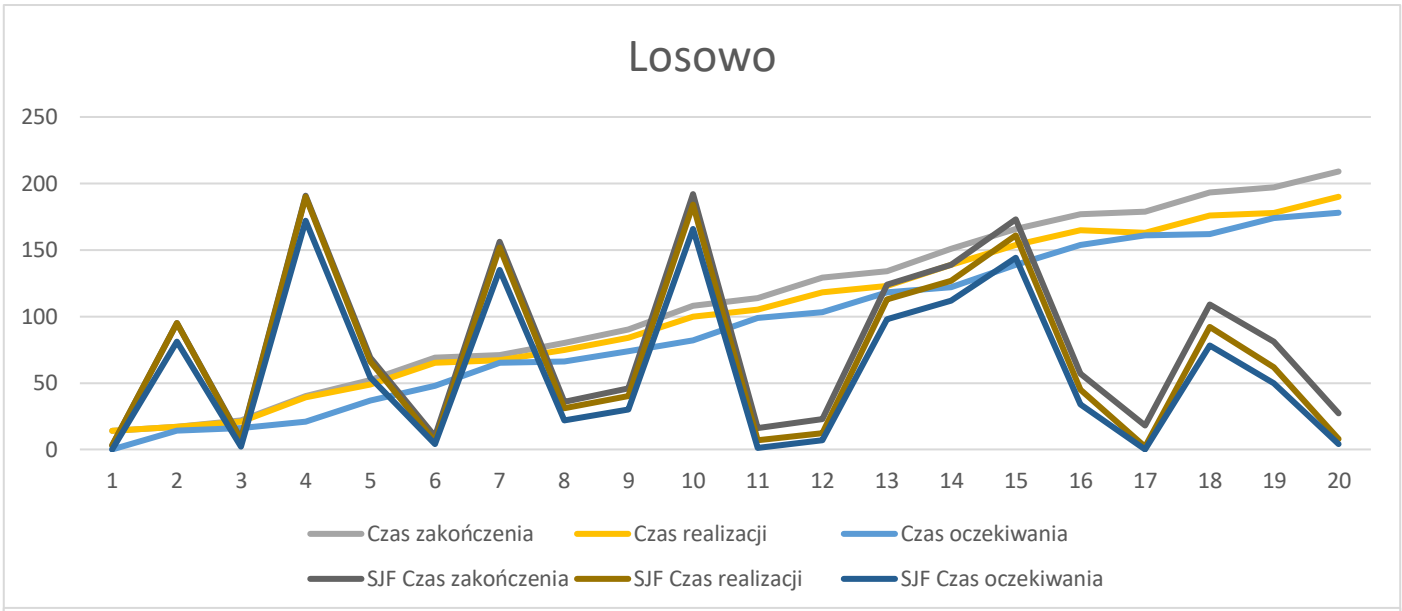
Efektywność SJF jest ok. 29% lepsza!

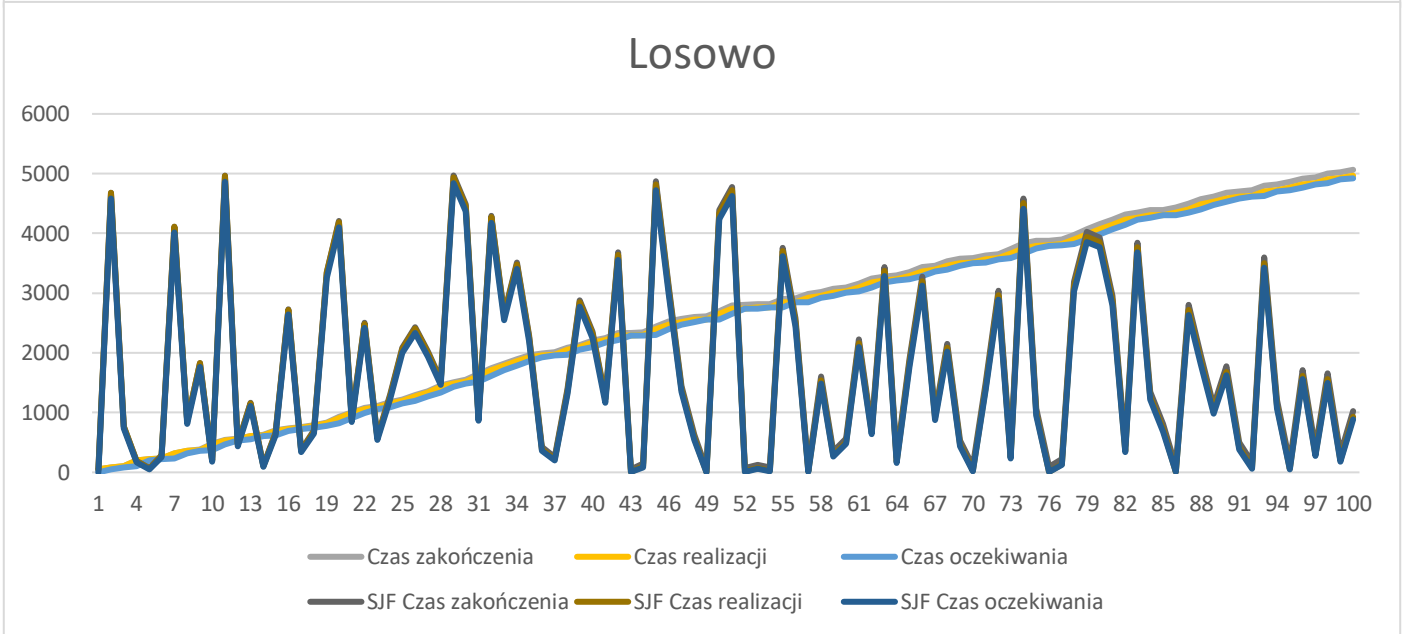




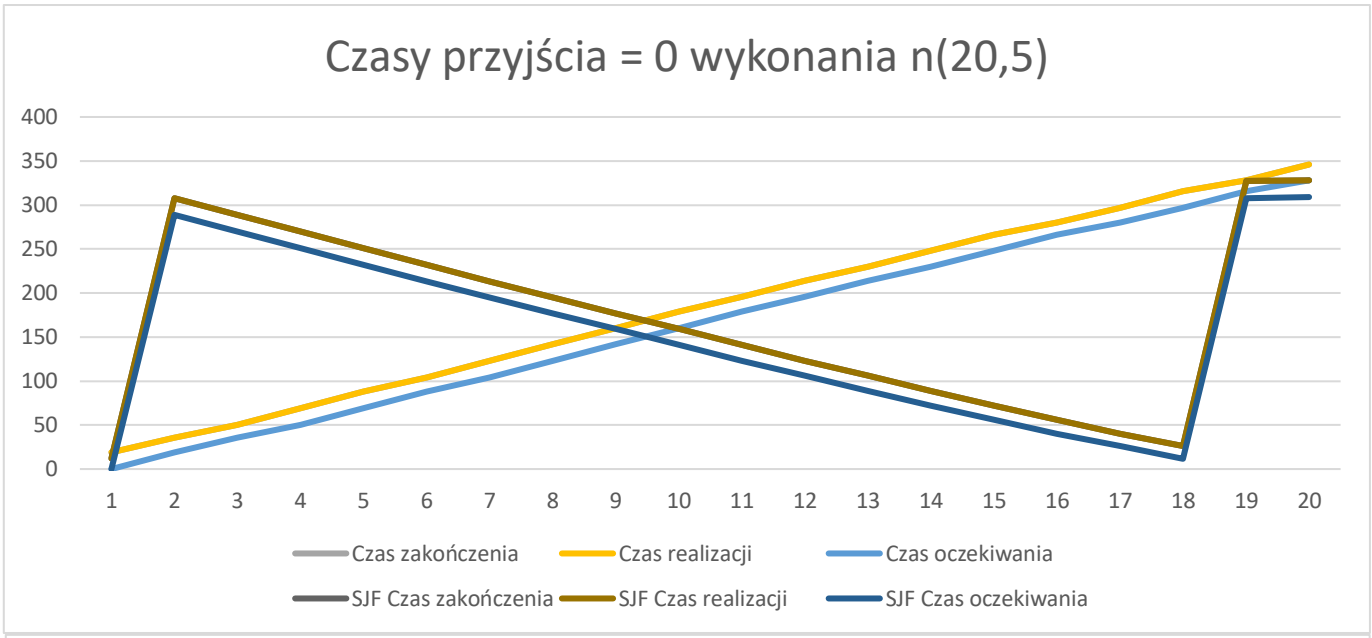


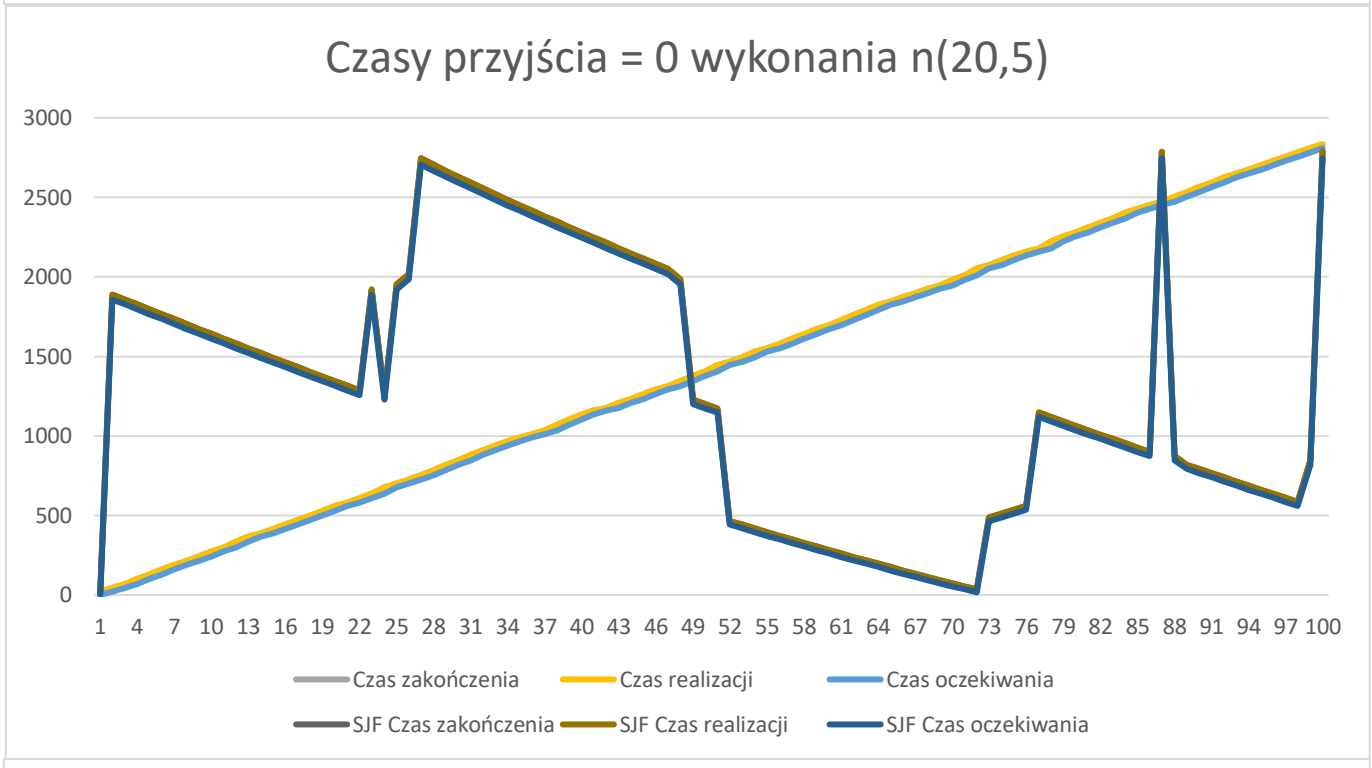
Efektywność SJF jest ok. 9% lepsza.

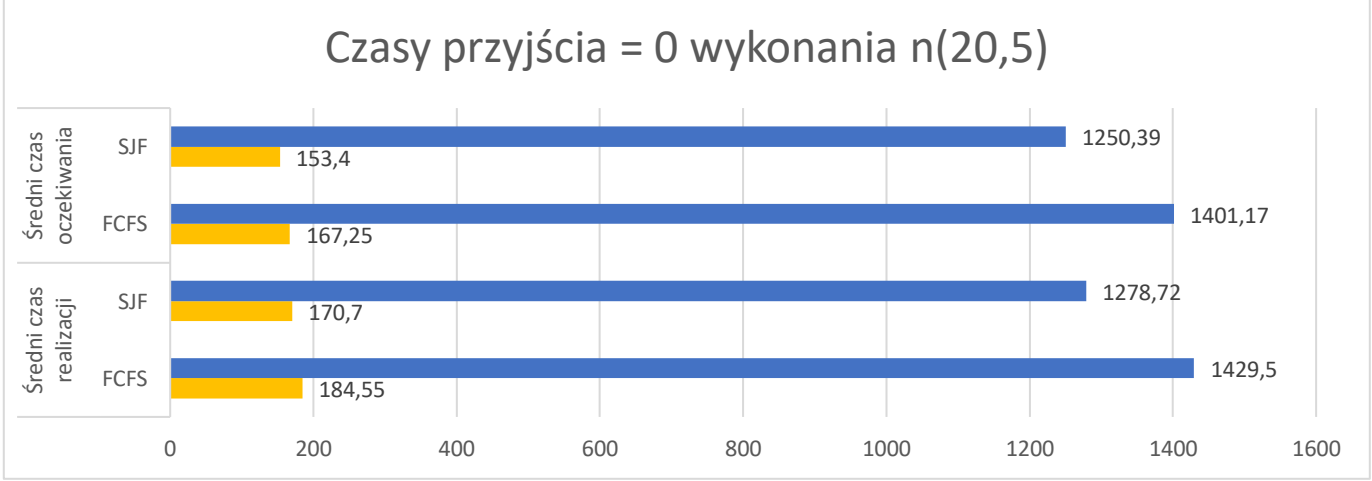




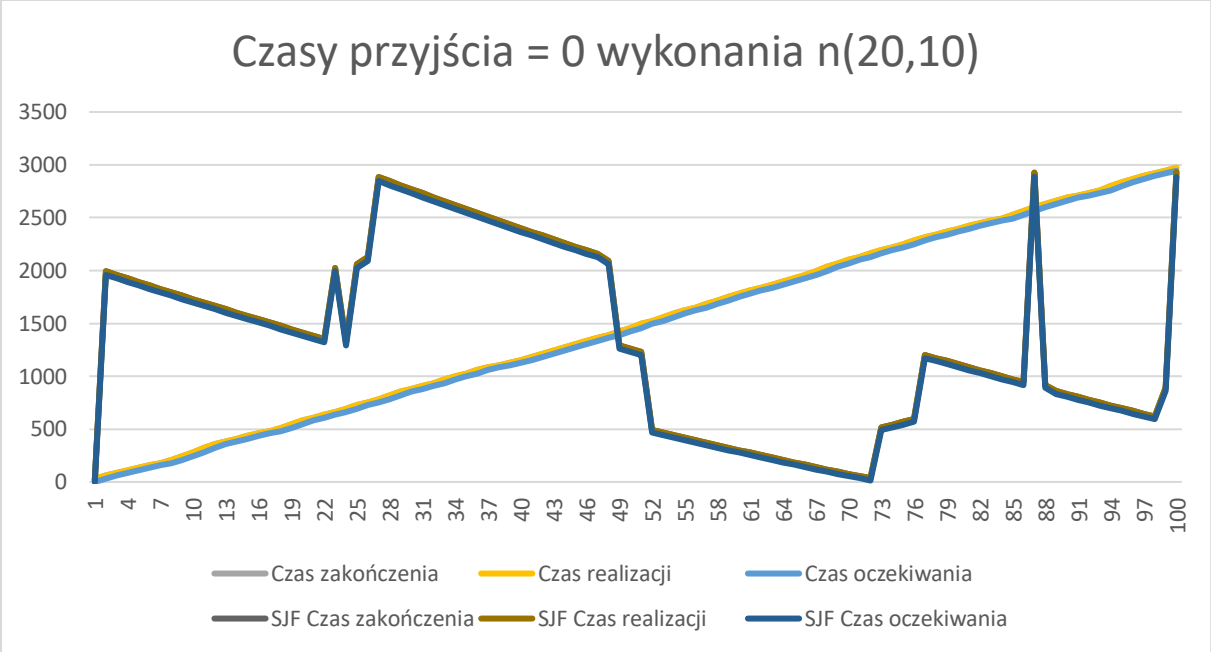
Efektywność SJF jest ok. 34% lepsza!!!

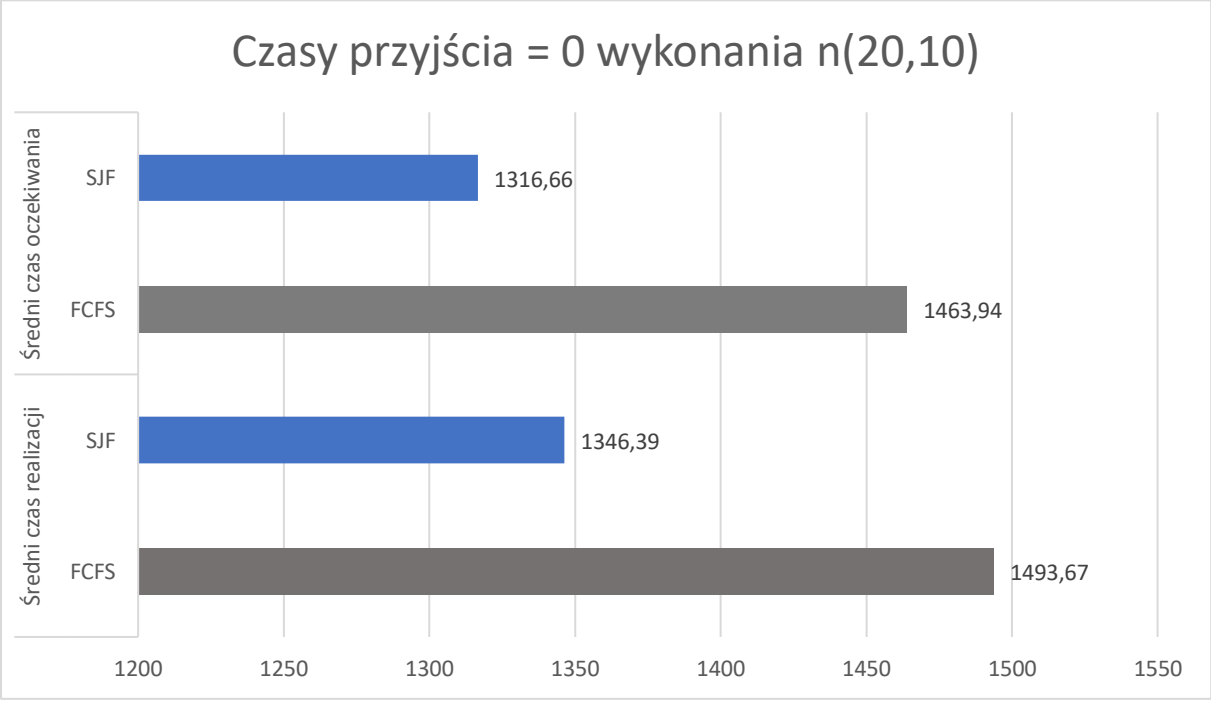






Efektywność SJF jest ok. 11% lepsza





Efektywność SJF jest ok. 10% lepsza.

Efekt Konwoju

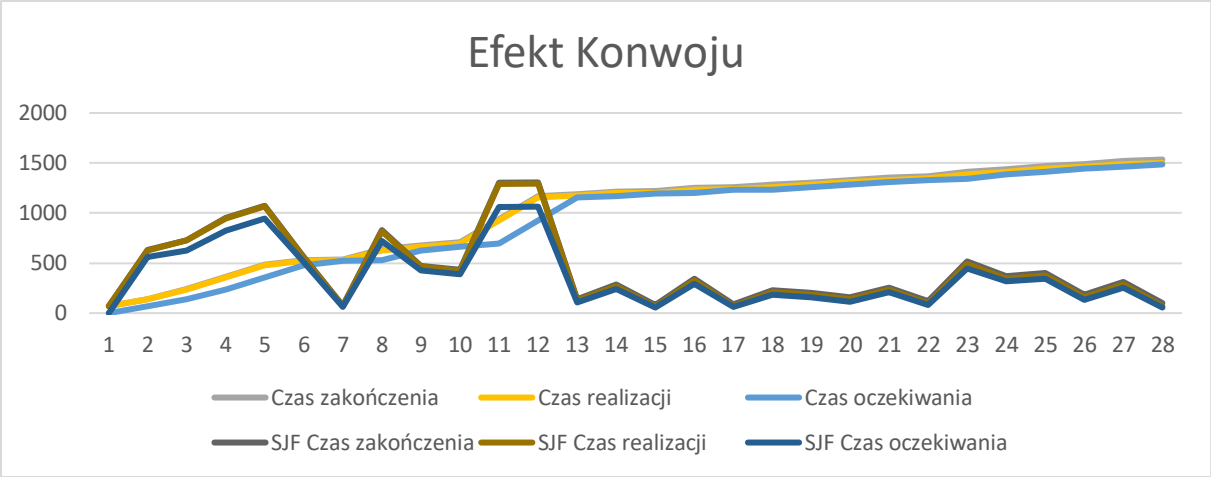
FCFS nie radzi sobie dobrze gdy czas wykonywania pierwszych paru procesów jest znacznie dłuższy od reszty. Wynika to z tego że procesor nie może obsłużyć żadnych procesów gdy jest zajęty jednym długim.

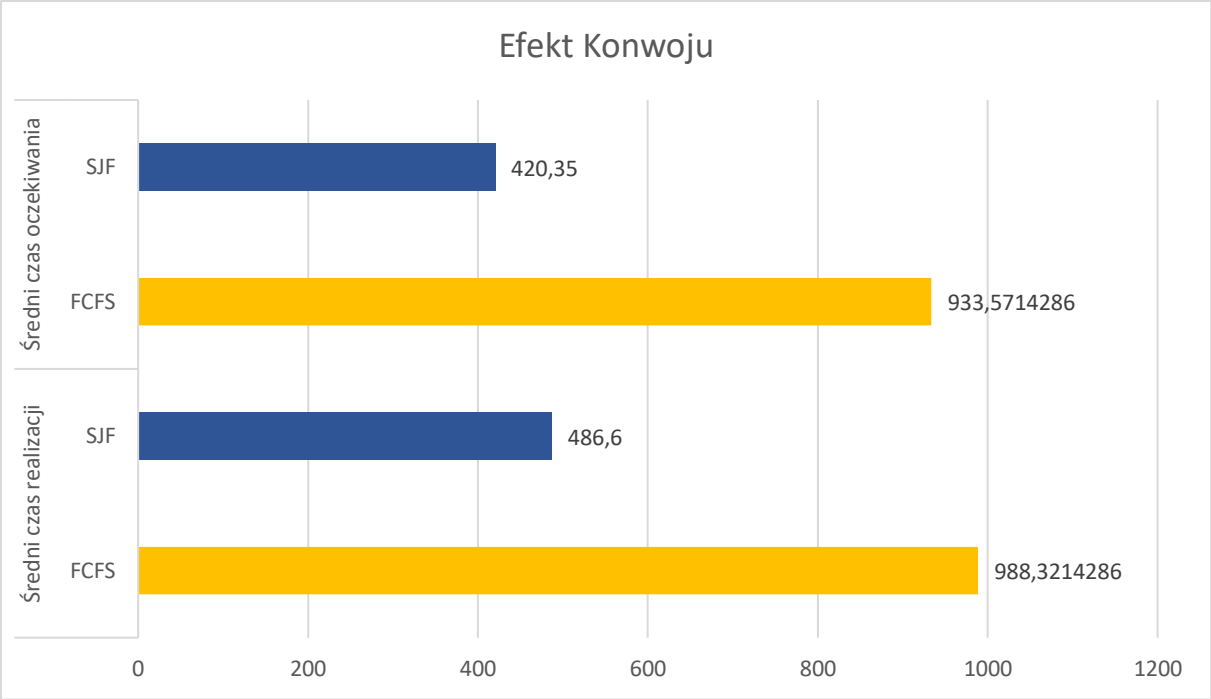
Prezentacja efektu konwoju:

```
C:\Users\fidle\OneDrive\Dokumenty\PWR\Systemy operacyjne\projekt\simulation_project>main.py --FCFS -p processes.txt
Czas nadejścia    Czas wykonywania  Czas zakończenia  Czas realizacji   Czas oczekiwania
2                 69                71                69                0
2                 69                140               138               69
4                 100               240               236               136
5                 123               363               358               235
6                 123               486               480               357
7                 45                531               524               479
8                 4                 535               527               523
9                 100               635               626               526
9                 40                675               666               626
11                33                708               697               664
12                230               938               926               696
12                230               1168              1156              926
12                18                1186              1174              1156
14                28                1214              1200              1172
17                6                 1220              1203              1197
19                29                1249              1230              1201
19                8                 1257              1238              1230
21                25                1282              1261              1236
22                23                1305              1283              1260
23                26                1331              1308              1282
23                22                1353              1330              1308
24                16                1369              1345              1329
25                41                1410              1385              1344
26                30                1440              1414              1384
26                29                1469              1443              1414
27                23                1492              1465              1442
30                29                1521              1491              1462
35                14                1535              1500              1486
Średni czas realizacji: 988.3214285714286
Średni czas oczekiwania: 933.5714285714286
```

Jak widać średni czas oczekiwania znacznie wzrósł, a razem z nim średni czas realizacji, zwłaszcza w porównaniu z SJF:

```
C:\Users\fidle\OneDrive\Dokumenty\PWR\Systemy operacyjne\projekt\simulation_project>main.py --SJF -p processes.txt
    Czas nadejścia  Czas wykonywania  Czas zakończenia  Czas realizacji  Czas oczekiwania
                 2                69                71               69                 0
                 2                69               629              627               558
                 4               100               729              725               625
                 5               123               952              947               824
                 6               123              1075             1069               946
                 7                45               560              553               508
                 8                 4                75               67                63
                 9               100               829              820               720
                 9                40               474              465               425
                11                33               434              423               390
                12               230              1305             1293              1063
                12                18               137              125               107
                14                28               284              270               242
                17                 6                81               64                58
                19                29               342              323               294
                19                 8                89               70                62
                21                25               230              209               184
                22                23               205              183               160
                23                26               256              233               207
                23                22               159              136               114
                24                16               119               95                79
                25                41               515              490               449
                26                29               371              345               316
                26                30               401              375               345
                27                23               182              155               132
                30                29               313              283               254
                35                14               103               68                54
Średni czas realizacji: 388.22222222222223
Średni czas oczekiwania: 339.962962962963
```





Efektywność SJF jest ponad dwukrotnie większa!!!!!

Wygładzanie procesów

Ze zjawiskiem wygładzania procesów mamy do czynienia, gdy z jakiegoś powodu procesy są odkładane w kolejce przez bardzo długi czas a czasem w nieskończoność. W algorytmie SJF ma to miejsce, gdy jakiś proces ma długi czas wykonywania i przez to cały czas wybierane są krótsze procesy.

Wnioski

- Analizując wygenerowane dane łatwo zauważyć znaczącą przewagę algorytmu SJF nad FCFS. Przy odpowiednich warunkach zwiększenie prędkości wynosiło nawet 30%. Różnica jest najlepiej widoczna, gdy w przychodzących procesach znajdują się procesy o znacznie krótszym czasie wykonywania(Patrz wykresy: losowe, n(20,10))
- Na wykresach obrazujących czasy dla poszczególnych procesów świetnie widać zjawisko wygładzania procesów przez algorytm SJF. Jest to zobrazowane skokowymi wzrostami czasów realizacji niektórych procesów. Pomimo podatności SJF na to zjawisko zwiększenie prędkości jest znaczące, a problemy które może wywoływać wygładzanie nie są widoczne w symulatorze.
- Aby zaobserwować efekt konwoju dane muszą przyjąć specyficzny wygląd, ale gdy efekt wystąpi znacząco pogarsza wynik wykonania kolejkowania. Jest to według mnie mimo rzadkości występowania duży minus algorytmu FCFS, a problem jest zauważalny w symulacji.
- Niesamowite jest jak nieduża modyfikacja algorytmu (wystarczy dodać sortowanie kolejki) potrafi sprawić, że działa on znacznie efektywniej.

Algorytmy zastępowania stron

Pamięć wirtualna

Każdy system ma ograniczoną ilość pamięci operacyjnej. Przy dużym wykorzystaniu procesora i wielu procesach do przetworzenia może zdarzyć się sytuacja, że pamięci operacyjnej zabraknie. Aby uniknąć poważnych błędów, które może spowodować brak dostępnej pamięci operacyjnej stosuje się pamięć wirtualną, która w skrócie jest dodatkowym miejscem dla danych wykorzystywanych przez procesy na wolniejszym nośniku. By wykorzystać dane w pamięci wirtualnej, należy je wpierw załadować do zapełnionej pamięci operacyjnej, a żeby to zrobić konieczne jest zastąpienie zasobu, który już się w niej znajduje. System operacyjny może to robić na parę sposobów, a ja porównałem dwa z nich: FIFO i LRU. Są to właśnie algorytmy zastępowania stron.

First In First Out(FIFO)

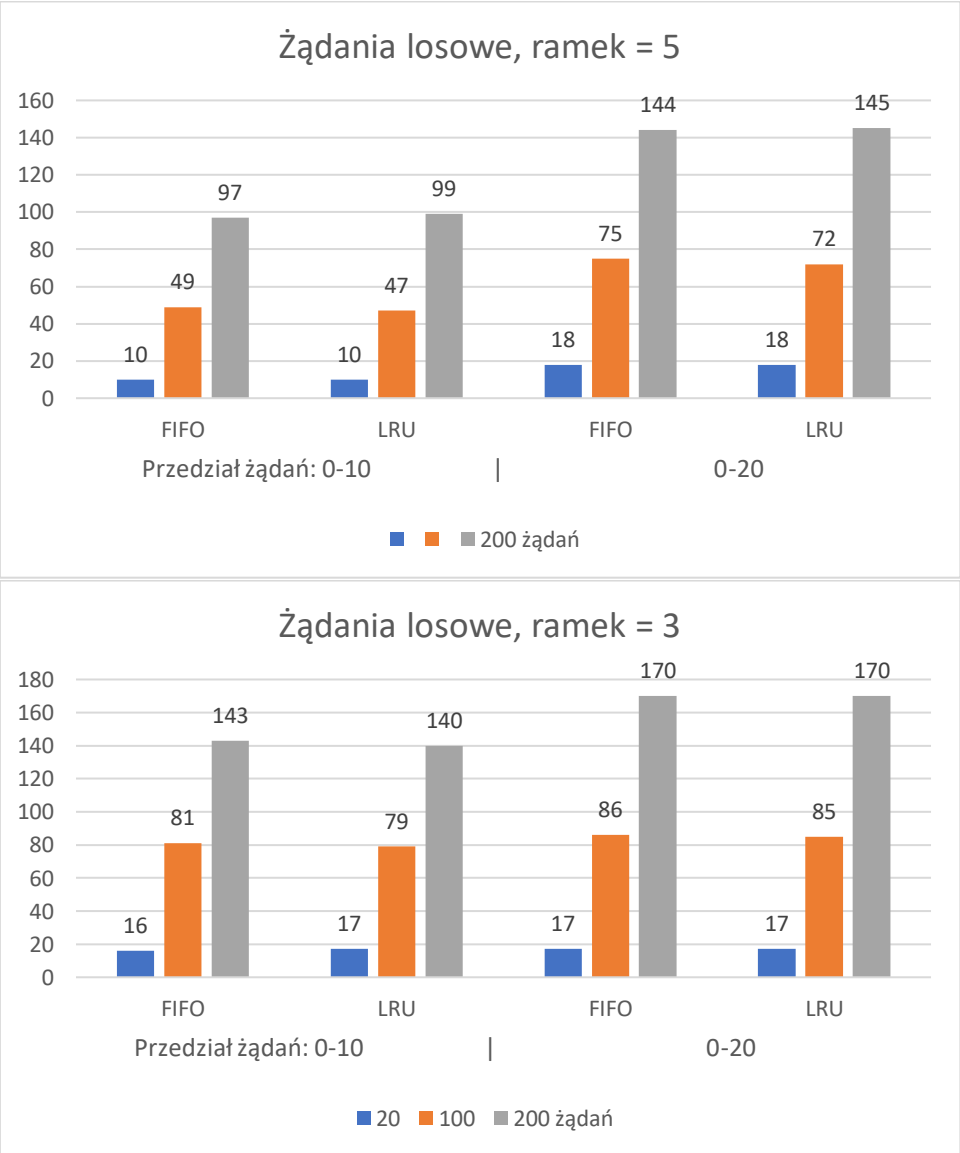
Jest prostym algorytmem, który nie zastanawia się którą stronę najlepiej wymienić i zastępuje ta która przyszła pierwsza, zgodnie z zasadą pierwsze przyszło, pierwsze wyszło.

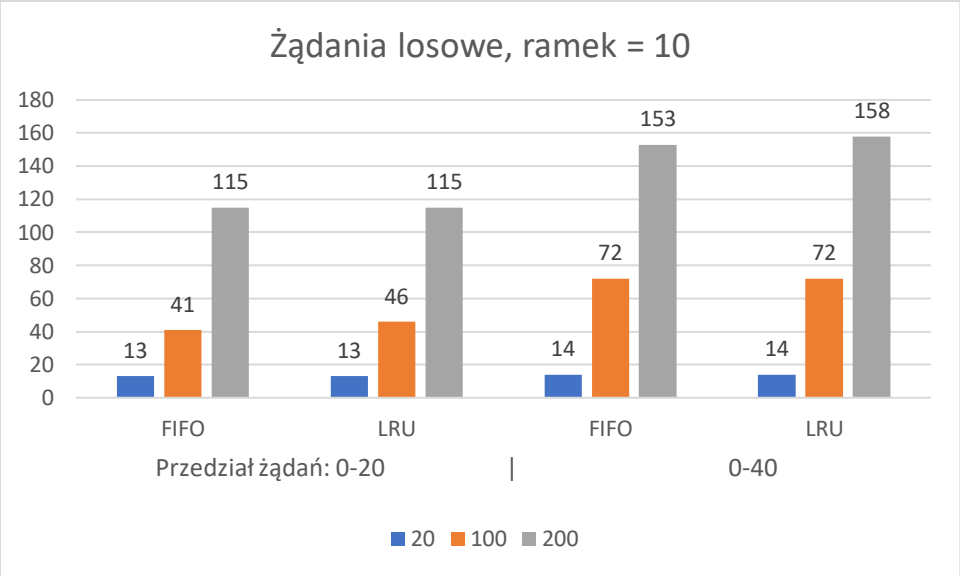
Least Recently Used(LRU)

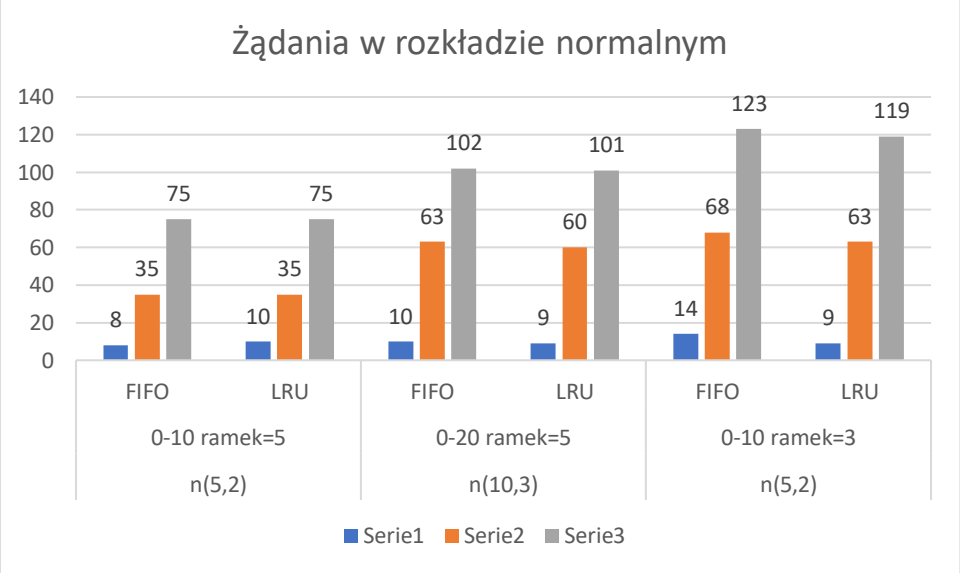
Ten algorytm już stara się przewidzieć, która strona może być mniej przydatna w przyszłości od reszty. Robi to na podstawie tego jak dawno temu dana strona była używana. Logika będąca podstawą tego algorytmu jest taka, że jeśli strona nie była dawno używana to pewnie już nie będzie w przyszłości.

Porównanie dla różnych danych

Liczba wymian strony







Wnioski

- Wyniki są zależne od tego jakie dane chcemy przetworzyć przez zadane algorytmy, jednak różnica w efektywności obu algorytmów jest nie znacząca. Symulacje nie pokazały żadnej przewagi bardziej skomplikowanego algorytmu LRU.
- Możliwe, że w zastosowaniu prawdziwym jakieś różnice zostałyby zauważone, jednak po przeprowadzeniu symulacji uważam, że stosowanie bardziej skomplikowanego algorytmu nie przynosi żądanej poprawy efektywności i w związku z tym nie warto jest go używać.