Problem pieciu filozofów

Paweł Gorgolewski

3 listopada 2022

1 Zawartość

W katalogach *java-homework* i *js-homework* znajduja sie rozwiazania w odpowiednich jezykach. Aby uruchomić programy dla różnych parametrów i rozwiazań (filozofowie i iteracje o wartościach odpowiednio: 5, 10, 15 oraz 50, 100, 150) należy uruchomić klase *Main* (dla rozwiazania z użyciem Javy) lub skrypt pythonowy *run_philosophers* (dla rozwiazania z użyciem JavaScript). Oba rozwiazania generuja pliki zawierajace informacje na temat egzekucji danego rozwiazania (w tym średnie czasy oczekwiania na widelce dla każdego filozofa) oraz użytych parametrów. Do tych plików należa wykresy jak i pliki txt z wynikami.

2 Wersja Javy

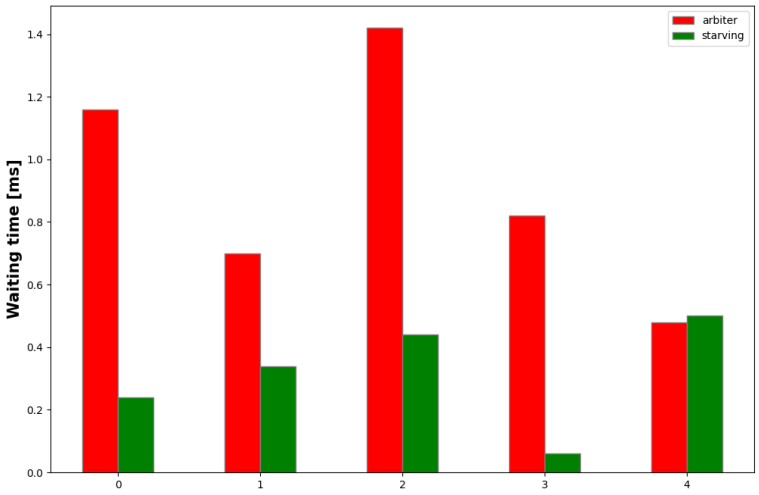
2.1 Uruchamianie programu

```
mvn compile exec:java -D exec.mainClass=org.example.Main -D exec.args=""
```

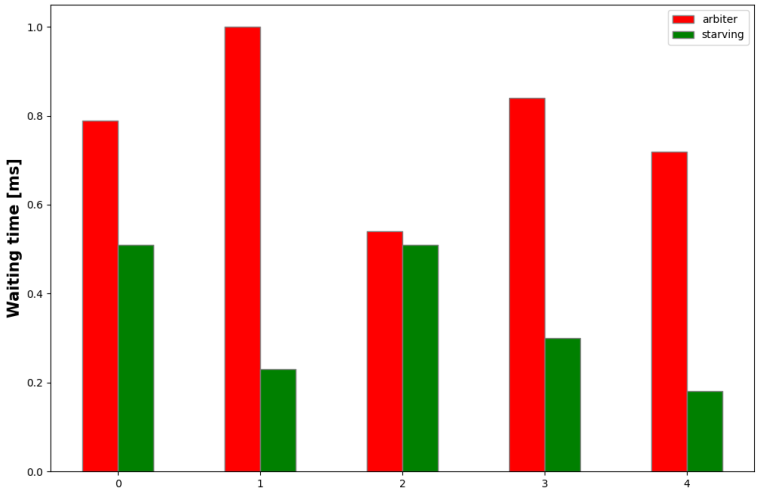
2.2 Szczegóły implementacji

- Zaimplementowane rozwiazania: zagłodzenie i arbiter
- Zagłodzenie - filozof czeka na lewy widelc i bierze prawy, jeżeli jest dostepny (zastosowanie semaphore.tryAcquire()). Jeżeli prawy widelec jest zajety to oddaje lewy i omija go ta kolejka jedzenia
- Arbtiter - arbiter zapewnia dostep do stołu N-1 filozofom, dzieki czemu zawsze przynajmniej jeden jest w stanie użyć obu widelców. Filozofowie zaimplemetowani sa na podstawie naiwnego rozwiazania - czeka na lewy i nastepnie na prawy widelec.
- Myślenie filozofów może trwać maksymalnie 100, zaś jedzenie 10 milisekund. Obie wartości sa za każdym razem losowane.

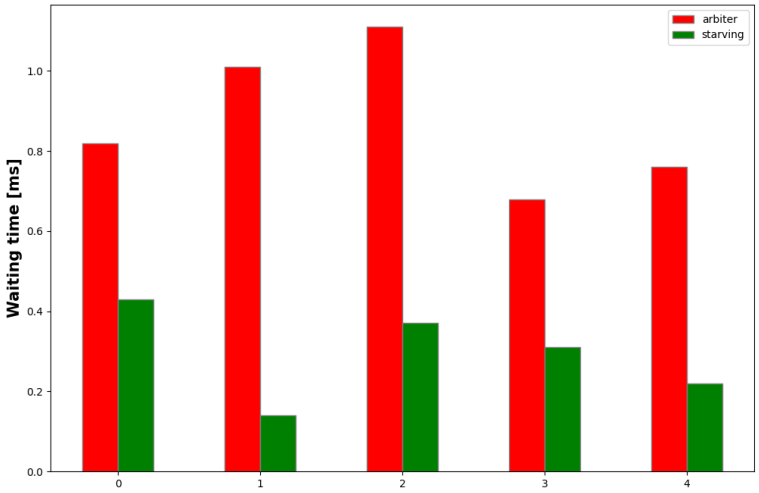
2.3 Wyniki dla 5 filozofów



Rysunek 1: 5 filozofów i 50 iteracji

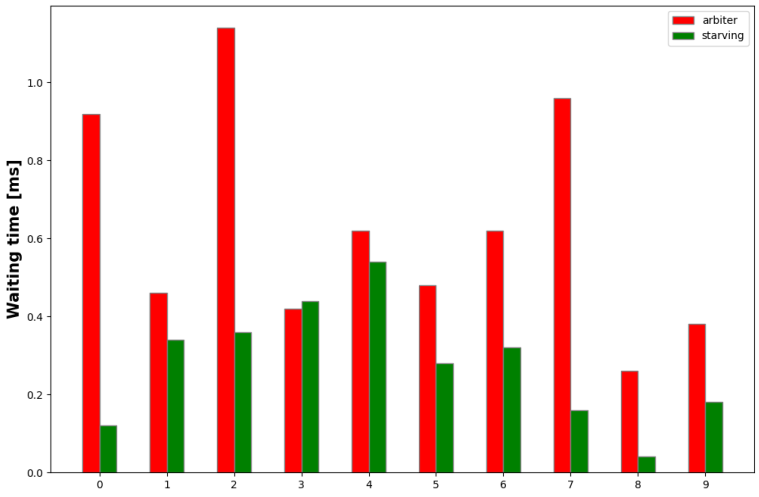


Rysunek 2: 5 filozofów i 100 iteracji

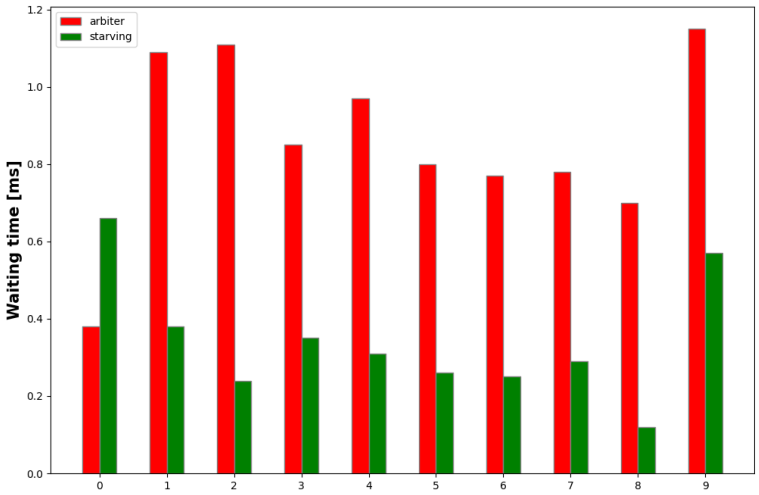


Rysunek 3: 5 filozofów i 150 iteracji

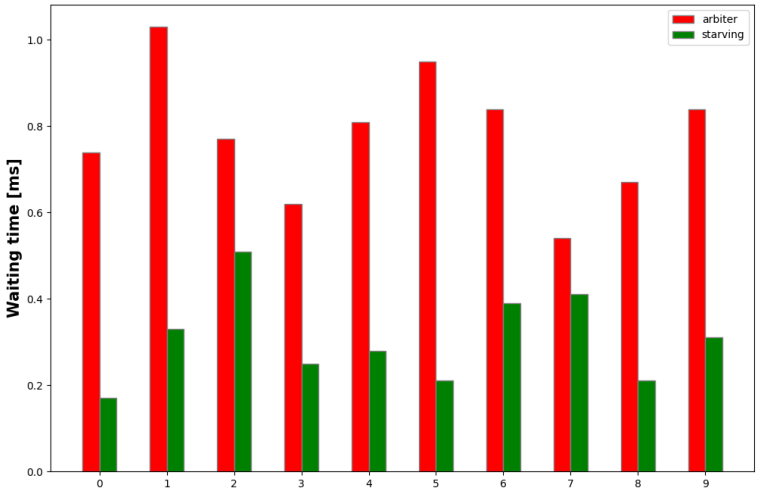
2.4 Wyniki dla 10 filozofów



Rysunek 4: 10 filozofów i 50 iteracji

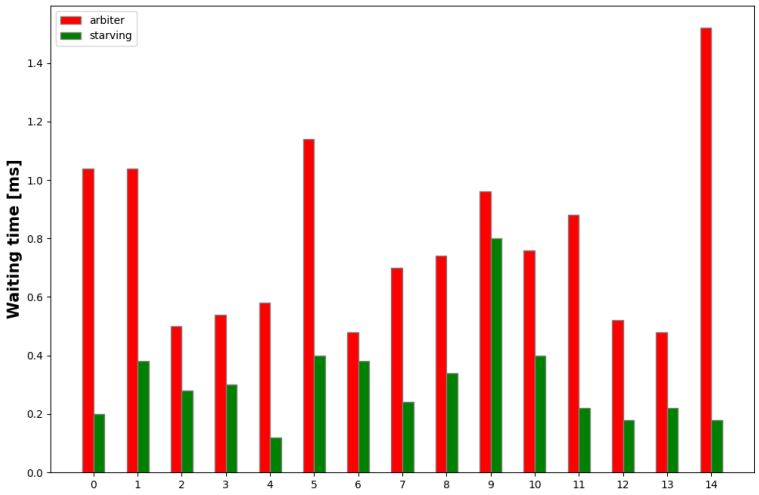


Rysunek 5: 10 filozofów i 100 iteracji

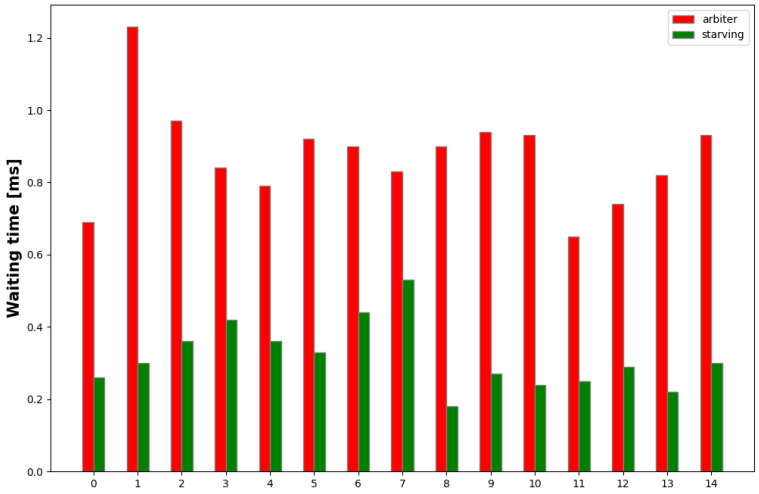


Rysunek 6: 10 filozofów i 150 iteracji

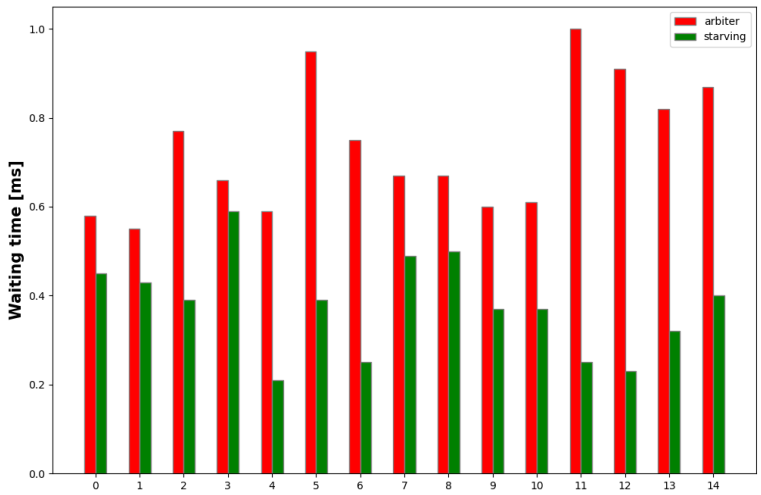
2.5 Wyniki dla 15 filozofów



Rysunek 7: 15 filozów i 50 iteracji



Rysunek 8: 15 filozów i 100 iteracji



Rysunek 9: 15 filozów i 150 iteracji

3 Interpretacja wyników

Wersja z arbitrem daje dłuższy czas oczekiwania na widelce. Zapewnia ona, że każdy filozof zje za każdym razem gdy zacznie czekać na widelce co nie jest zagwarantowane w rozwiazaniu z głodowaniem.

Patrzac na wykresy, łatwo zauważyć, że dla wiekszych ilości filozofów i iteracji, dostajemy wieksze różnice pomiedzy rozwiazaniami

4 Wersja JavaScript

4.1 Uruchamianie programu

Do opalenia programu należy użyc komendy:

```
node philosophers.js <philosophersNum> <eatCount> <version>
```

- philosophersNum - liczba filozofów przy stole
- eatCount - liczba iteracji, czyli ile razy każdy filozof ma jeść
- version - wersja programu. Należy podać jedna z wartośći: ”naive”, ”asym”, ”bothForks”, ”conductor”

UWAGA!

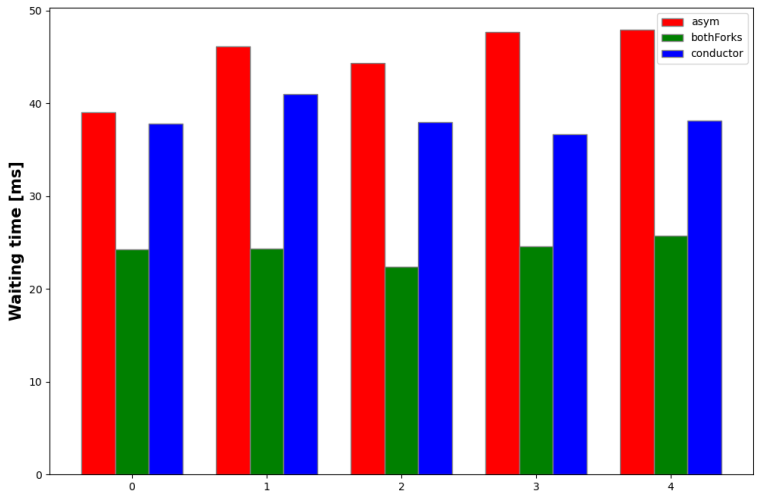
Podanie wiekszej ilości argumentów spowoduje wypisanie tak zwanych *debug printów* - informuja one o podonoszeniu i opuszczaniu widelców przez filozofów. Nie sa one użyte w programie, sprawdzana jest tylko ilośc podanych argumentów

4.2 Szczegóły implementacji

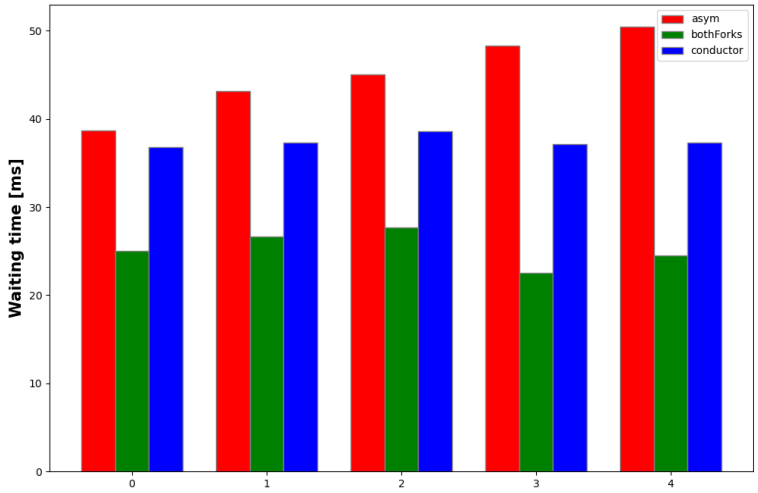
- Zaimplementowane rozwiazania: naiwne, asymetryczne, oba widlece na raz oraz arbiter
- Naiwne - filozof czeka na lewy widelec a nastepnie na prawy. Rozwiazanie blokuje sie, zatem nie zostało uwzglednione w skrypcie pythonowym.
- Asymetryczne - filozofowie o parzystym *id* jako pierwszy biora lewy widelec a reszta prawy.
- Arbtiter - arbiter zapewnia dostep do stołu N-1 filozofom, dzieki czemu zawsze przynajmniej jeden jest w stanie użyć obu widelców. Filozofowie zaimplemetowani sa na podstawie rozwiazania z podnoszeniem obu widelcy na raz. Arbiter oparty jest o algorytm BEB.

- Oba widelce - filozofowie czekaja aż oba widelce beda dostepne.
- Alogrytm BEB stworzony na podstawie funkcji *setTimeout()*. Maksymalny czas to 2^{12} milisekund (czyli około 4s). Po osiagniecu tej wartości filozof nie bedzie zwiekszał czasu odczekiwania tylko za każdym razem bedzie odczekiwał ta wartość
- Myślenie filozofów może trwać maksymalnie 100, zaś jedzenie 10 milisekund. Obie wartości sa za każdym razem losowane.

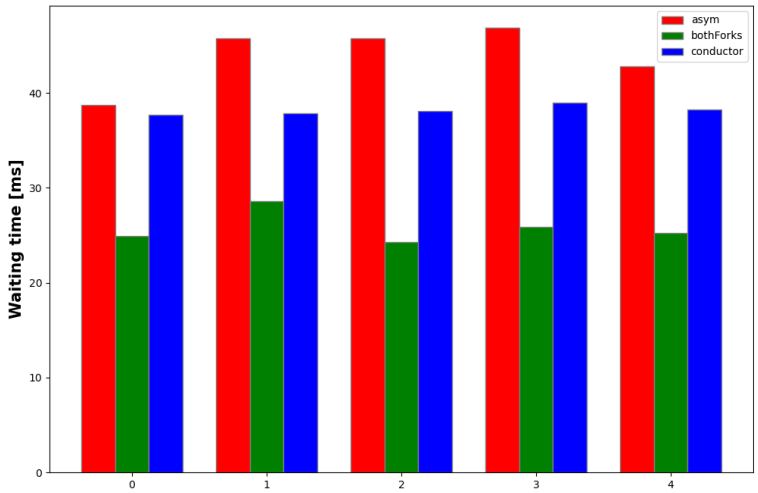
4.3 Wyniki dla 5 filozofów



Rysunek 10: 5 filozofów i 50 iteracji

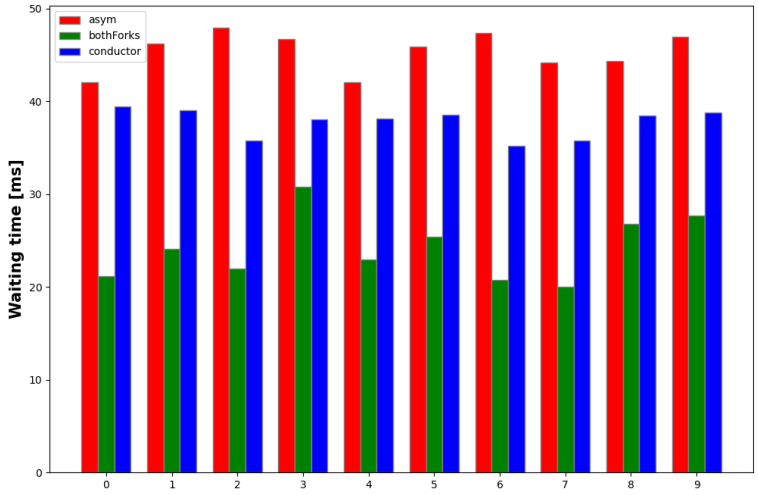


Rysunek 11: 5 filozofów i 100 iteracji

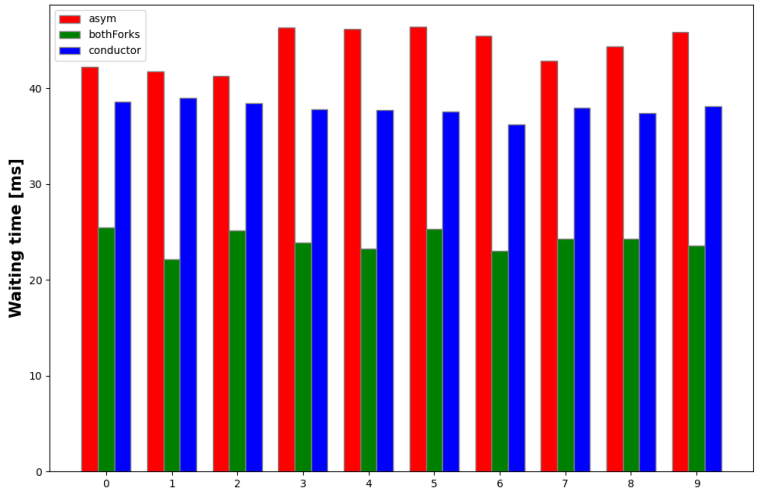


Rysunek 12: 5 filozofów i 150 iteracji

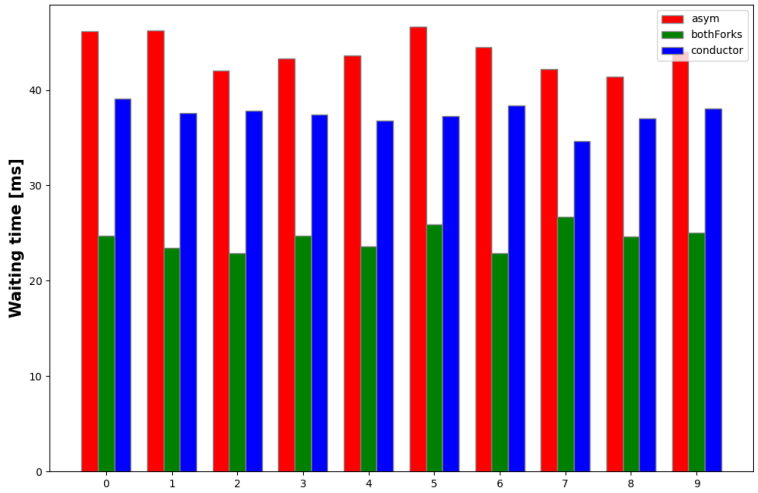
4.4 Wyniki dla 10 filozofów



Rysunek 13: 10 filozofów i 50 iteracji

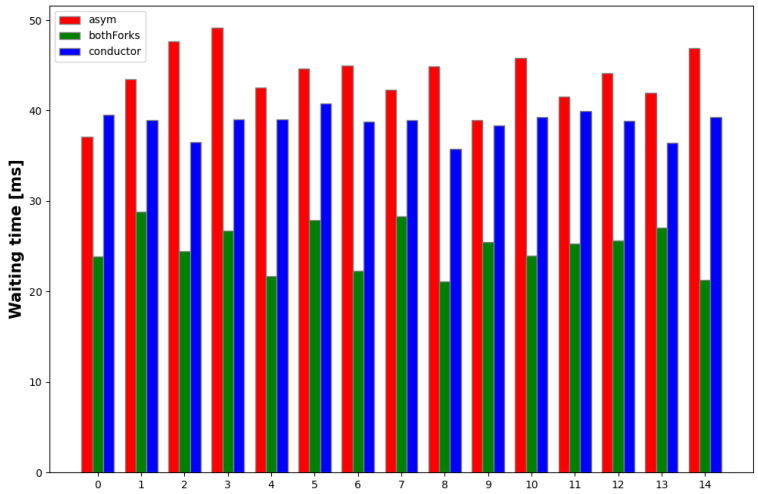


Rysunek 14: 10 filozofów i 100 iteracji

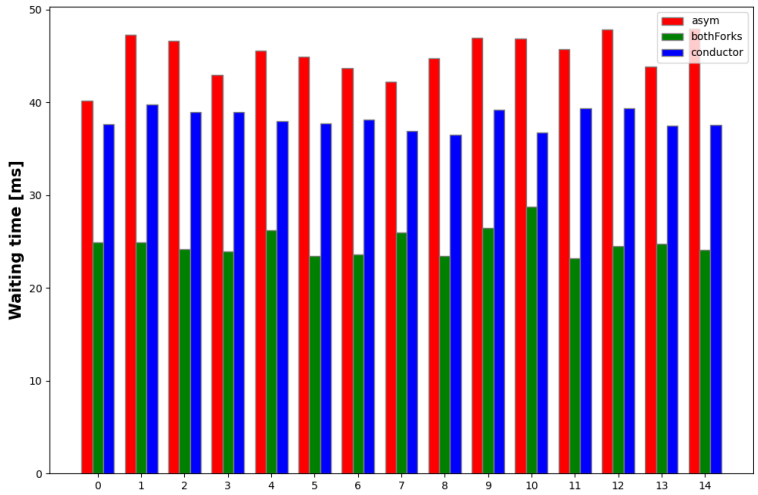


Rysunek 15: 10 filozofów i 150 iteracji

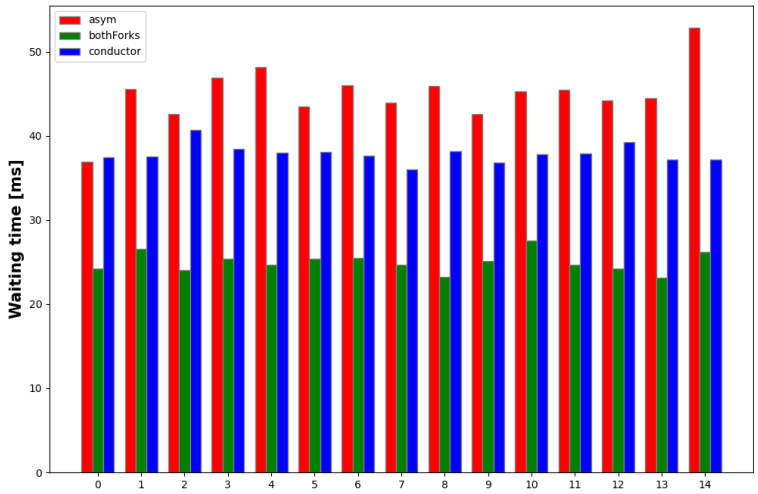
4.5 Wyniki dla 15 filozofów



Rysunek 16: 15 filozów i 50 iteracji



Rysunek 17: 15 filozów i 100 iteracji



Rysunek 18: 15 filozów i 150 iteracji

5 Interpretacja wyników

Rozwiazanie asymetryczne okazało sie tym, o najdłuższym czasie czekania na widelce. Niewatpliwie jest to spowodowane uzyciem metody dwóch widelcy na raz w rozwiazaniu z kelnerem, ponieważ intuicyjnie powinno mieć dłuższy czas przez synchronizacje dostepu do stołu. Ilośc iteracji jak i liczba filozofów wydaje sie nie mieć wiekszego znaczenia.

Dodatkowo, wykresy te mogłyby wygladać zupełnie inaczej, gdybyśmy nie uwzgledniali przerwy na myślenie u filozofów. Filozofowie na samym poczatku blokowali by sie, ponieważ wszyscy chcieliby mieć dostep do widelcy. Wraz z losowym czasem myślenia, niwelujemy ten ruch na samym poczatku programu.