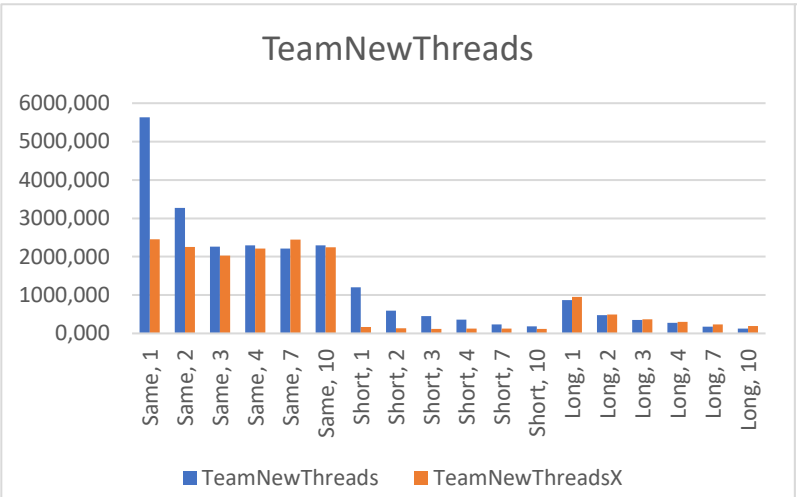
Raport

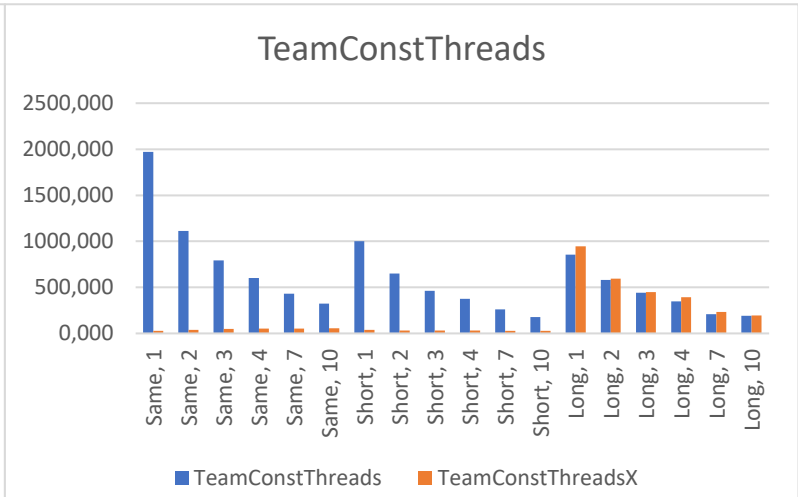
Przedstawiam wyniki trzeciego zadania zaliczeniowego „Problem Collatza”. Z powodu że na maszynie students moje rozwiązanie nie przechodziło wszystkich testów, a na moim komputerze je przechodziło, zdecydowałam się uwzględnić tylko wyniki działania programu na moim komputerze, a pominąć częściowe wyniki na studentsie.

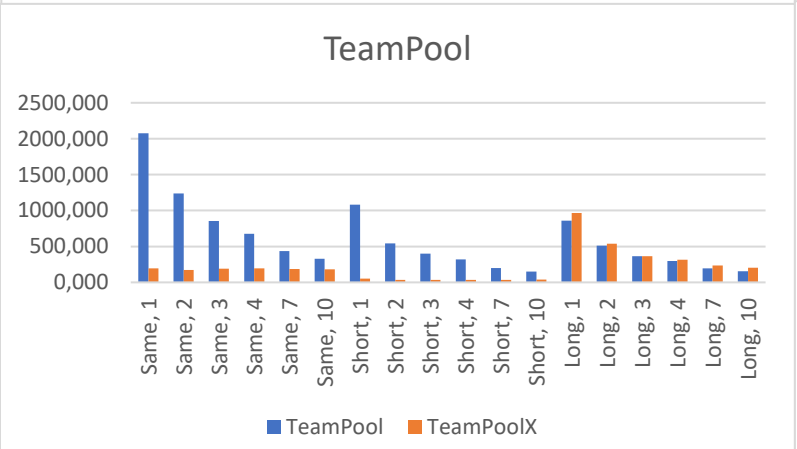
Model procesora mojego komputera: AMD Ryzen 5 5500U, 6 rdzeni

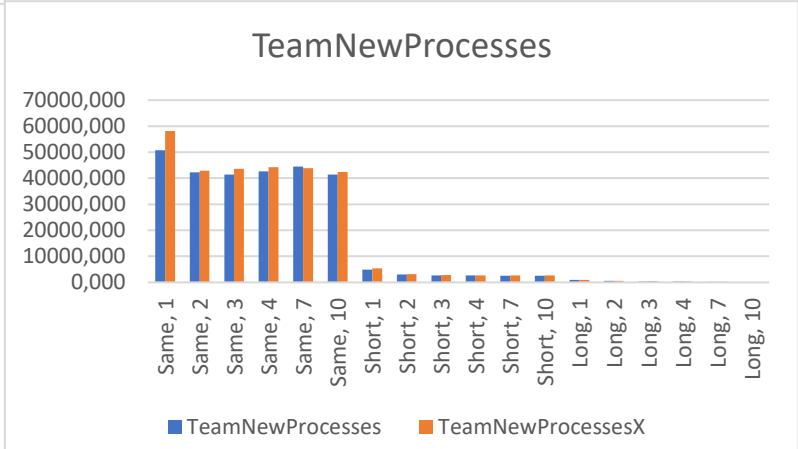
Jak wynika z wykresów, drużyny ConstThreads, Pool i ConstProcesses osiągnęły bardzo podobne wyniki. Drużyna NewThreads osiągnęła od nich gorsze wyniki w konkursie SameNumber. Drużyna Async wydaje się być najlepsza spośród drużyn niekorzystających z SharedResults, a drużyna NewProcesses zdecydowanie najgorsza.

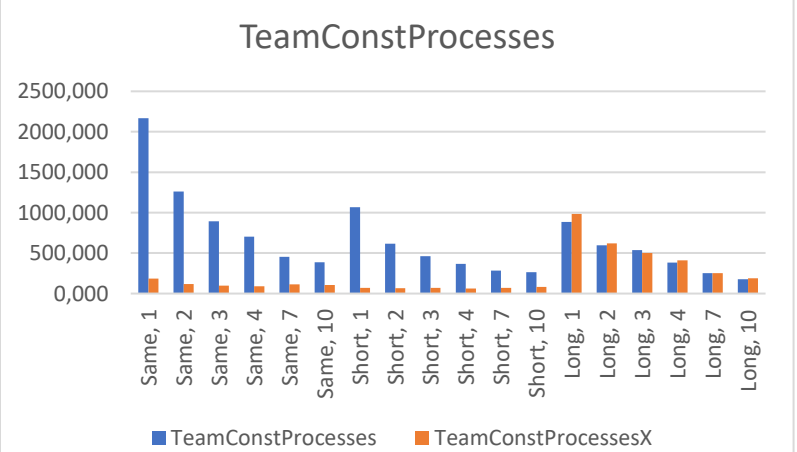
Poniżej przedstawiam zebrane dane. Czas na osi pionowej podany jest w milisekundach. Każdy słupek (poza Async) jest średnią z wyników dla wybranego konkursu i wybranej wielkości getSize().

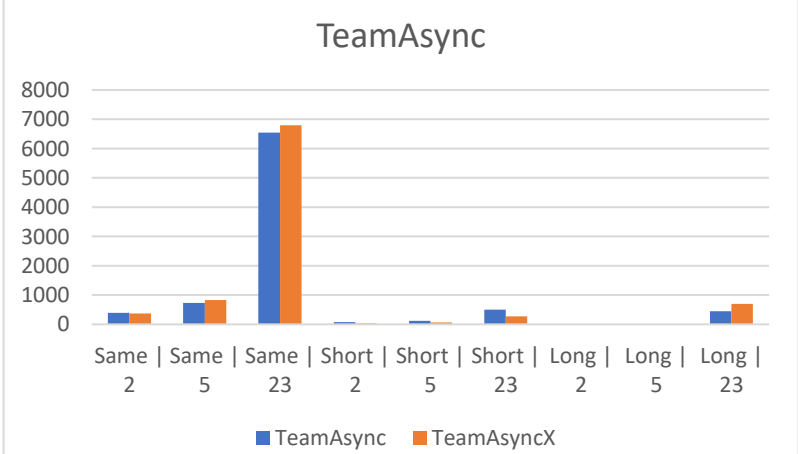












SameNumber

- Drużyny ConstThreads, Pool i ConstProcesses osiągnęły bardzo podobne wyniki z widocznym wpływem zwiększenia zmiennej getSize() na zmniejszenie czasu działania. Można się spodziewać takiej wyraźnej zależności, ponieważ wątki/procesy wykonują znaczącą i podobną (lub taką samą) ilość pracy.
- Drużyny NewThreads i NewProcesses osiągnęły gorsze wyniki (przy czym NewProcesses nieporównywalnie gorsze). Dla tych drużyn zwiększenie getSize() nie miało dużego znaczenia. Może tak być, ponieważ stworzenie nowego wątku/procesu wydaje się być kosztowniejsze od obliczenia funkcji calcCollatz() dla małych liczb, a niezależnie od getSize() trzeba stworzyć ich tyle samo. Z tego samego powodu drużyny NewThreadsX i NewProcessesX osiągnęły bardzo podobne wyniki do ich odpowiedników bez SharedResults.
- Najlepsze wyniki osiągnęła drużyna ConstThreadsX, a niewiele gorsze drużyny PoolX i ConstProcessesX. W przypadku tych drużyn zwiększenie współbieżności miało mały wpływ na ich czas działania, dlatego, że obliczenie funkcji calcCollatzX było natychmiastowe, a więc koszt stworzenia nowego wątku/procesu mógł zniwelować zysk z dodatkowej współbieżności.

ShortNumber

- Drużyny NewThreads, ConstThreads, Pool i ConstProcesses osiągnęły podobne wyniki. Tym razem (porównując z konkursem SameNumber) drużyna NewThreads osiągnęła podobny rezultat, ponieważ obliczenie funkcji calcCollatz stało się bardziej kosztowne, przez co stworzenie nowego wątku stało się względnie mniej kosztowne.
- W tym konkursie drużyny X osiągnęły (poza NewProcesses i Async) dużo lepsze wyniki, dzięki temu, że wejście było różnorodne i dość długie. Drużyny NewThreadsX i ConstProcessesX osiągnęły gorsze wyniki niż ConstThreadsX i PoolX, ponieważ dzięki użyciu SharedResults koszt obliczenia calcCollatz(X) zmalał, a więc stworzenie nowego wątku/procesu stało się względnie bardziej kosztowne.

LongNumber

- W tym konkursie koszt obliczenia calcCollatz jest duży, a wielkość wejścia mała. Powoduje to brak przydatności struktury SharedResults i gorsze wyniki drużyn X niż ich odpowiedników, ponieważ korzystanie ze struktury wiąże się z dodatkowym kosztem.
- Drużyny NewThreads, NewThreadsX, ConstThreads, ConstThreadsX, Pool, PoolX, ConstProcesses, ConstProcessesX wszystkie osiągnęły bardzo podobne wyniki, przy czym drużyny X nieco gorsze.