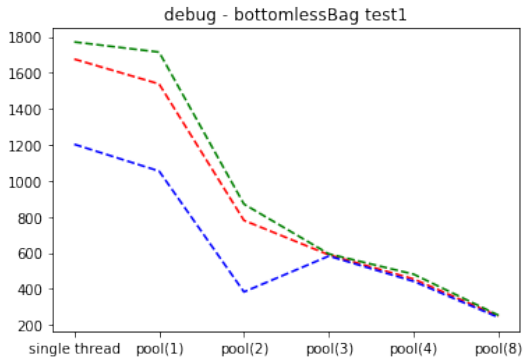
Projekt zaliczeniowy z programowania współbieżnego w C++ - raport

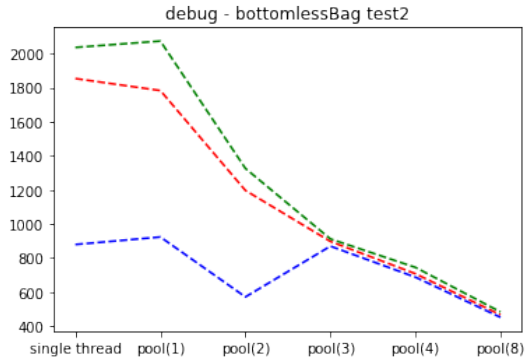
Do testowania zostały użyte dwa ostatnie większe testy dla każdego problemu dostarczone w treści zadania. Każdy test został odpalony 12 razy w wersji debug oraz release na maszynie students.

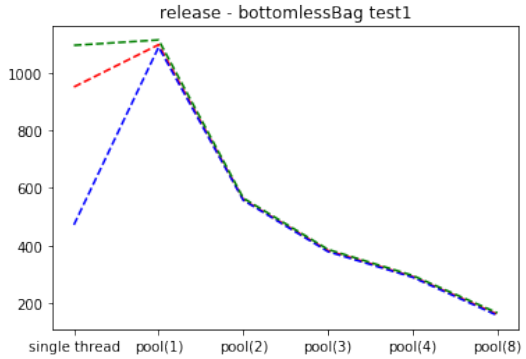
Legenda:
zielona przerywana linia - max
czerwona przerywana linia - średnia
niebieska przerywana linia - min

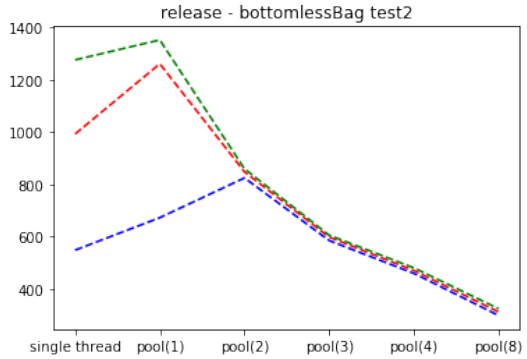
Dyskretny problem plecakowy (problem 1.)

Obliczanie kolejnego wiersza macierzy bazuje na wartościach w poprzednim wierszu, więc pracę do obliczenia danego wiersza można podzielić i każdą część wykonać w puli wątków.



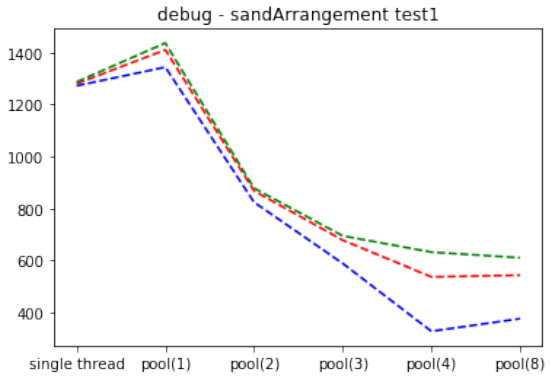


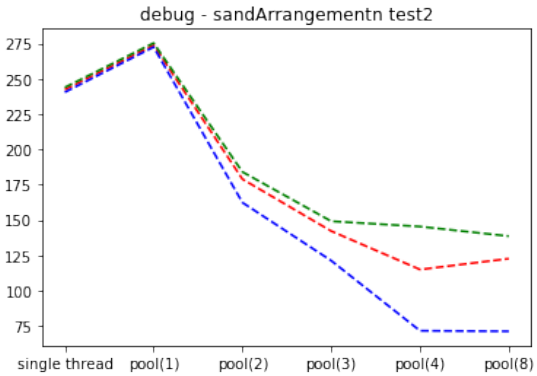


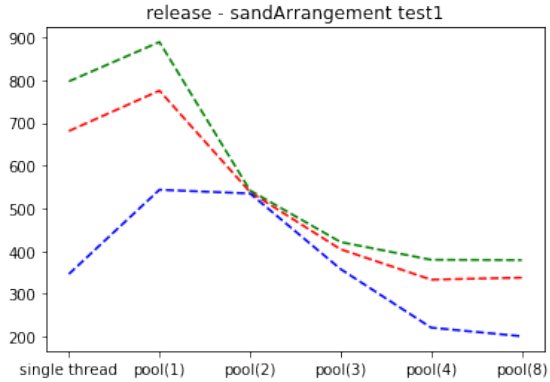


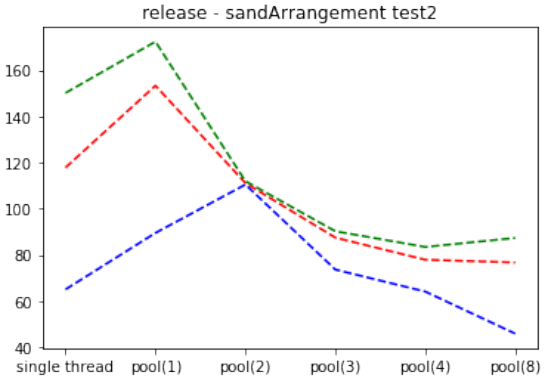
Sortowanie przez scalanie (Merge sort) lub sortowanie szybkie (Quicksort) (problem 2.)

Wybrany algorytm to iteracyjny merge sort. Początkowa część algorytmu, gdy mergujemy małe podtabele, musi zostać zaimplementowana sekwencyjnie, ponieważ zbyt duża liczba podzadań zwalniałaby algorytm tak mocno, że byłby on wolniejszy niż całkowicie sekwencyjny algorytm. Gdy mergujemy współbieżnie tylko większe podtabele można zauważyć znaczący wzrost szybkości.



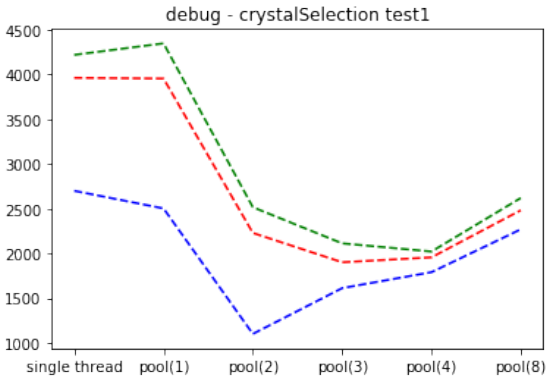


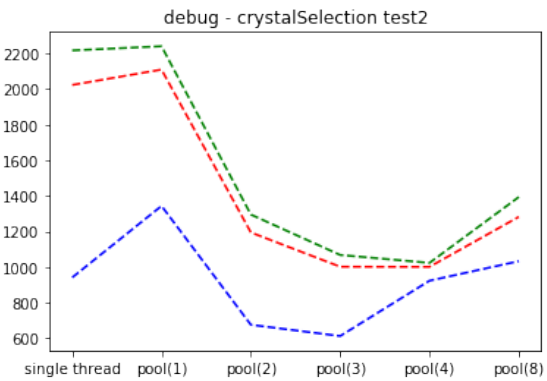


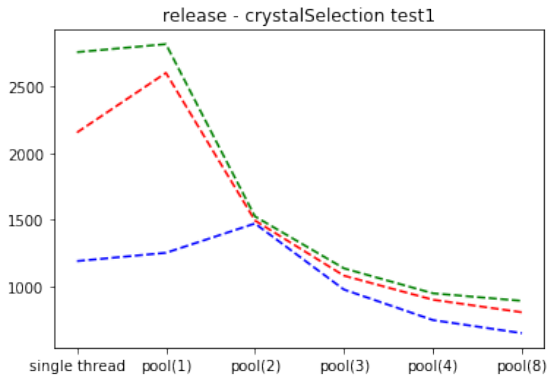


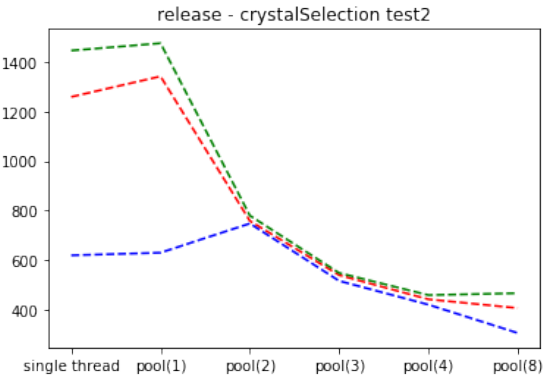
Znajdowanie elementu maksymalnego (problem 3.)

Rozwiązanie polega na podzieleniu tablicy części, obliczeniu maksimum każdej z tych części współbieżnie, a na koniec z otrzymanych wyników znaleźć maksimum całej tabeli.









Każdy algorytm uzyskuje zadawalające przyspieszenie gdy jest wykonywany współbieżnie. Zdecydowanie największe przyspieszenie jest zauważalne w problemie plecakowym nawet blisko 7-krotne. Testy wypadły lokalnie nieco szybciej, ale z gorszym przyspieszeniem.

Patryk Skrzeczkowski