W ramach listy 7 i zagadnień związanych z programowaniem równoległym przeprowadziłam testy na kilku problemach. Problemy rozwiązane zostały zarówno sekwencyjnie jak i za pomocą zrównoleglenia.

Testy przeprowadzono na problemach o różnych rozmiarach. W celu uwiarygodnienia wyników, każdy test składał się z 5 prób czasowych, z których uzyskano średni czas. Zbadano również pływ aktywnych procesów w tle, niezwiązanych z testami – przeprowadzono testy przy działających w tle aplikacjach oraz przy wyłączonych. W drugim przypadku uzyskano nieznacznie lepsze wyniki, jednak nie można stwierdzić, wpływ jednak był jednakowy na wszystkie testy.

1. Wyznaczanie liczy PI metodą Monte Carlo

	Czas w milisekundach					
Liczba prób	sekwencyjnie	2 futures	4 futures	10 futures	100 futures	2000 futures
1 000	3	18	0	0	0	12
100 000	3	3	0	0	3	0
10 000 000	359	187	96	53	50	50
1 000 000 000	35 813	18 125	9 428	5 175	4 825	4 603

Uzyskano wyniki zgodnie z oczekiwaniami, wraz z zwiększaniem się problemu (większa liczba prób daje precyzyjniejszy wynik wyznaczenia liczby PI) wzrasta użyteczność zrównoleglania problemu. Dla 1 000 000 000 prób precyzji możemy skrócić czas prawie 8-krotnie. Najwięcej czasu zajmuje losowanie liczb z zakresu [0, 1).

Znaczący wpływ na wynik testów miało jednokrotne stworzenie obiektu new Random() i następnie używanie metody nextDouble() zamiast po prostu wywoływanie metody Math.random() (w drugim przypadku otrzymywano wyniki ponad 7-krotnie większe dla zrównoleglonego rozwiązania)

2. Działania na drzewach

Generowanie drzewa, Pogłębianie drzewa, sumowanie elementów drzewa:

	Czas w milisekundach						
Głębokość drzewa	sekwencyjnie	Parallel 1	parallel 2	Parallel 3	Parallel 6	Parallel 10	Parallel 15
5	0	3	0	0	-	-	-
15	0	0	0	0	3	24	2
25	194	94	99	90	105	127	609

Testów nie przeprowadzono na większych drzewach, ponieważ niemożliwym było stworzenie większego problemu (outOfMemoryError dla głębokości większych niż 25).

Różnice widać w testach dopiero na drzewie o głębokości 25. Udaje się dwukrotnie skrócić czas obliczeń. Jest jednak próg, gdzie dalszy podział na podproblemy jest kosztowny i czas rośnie nawet 3-krotnie.

3. Działania na listach – obliczanie wydatków firmy na pensje dla pracowników

		Czas w milisekundach		
L. pracowników	L. dni pracy	sekwencyjnie	future 2	future 4
1 000	30	1	13	9
1 000	3 000	0	0	0
100 000	30	8	20	22
100 000	3 000	420	247	174

W zadaniu nr 3 dla małych danych zrównoleglenie nie przynosi zysków. Dzielenie listy jest operacją kosztowną i przewyższa ona zalety używania Future. Różnice widać w przypadku większej liczby pracowników, czas wykonania zmniejsza się.

Sekwencyjne rozwiązanie problemu opiera się na rekurencji ogonowej, co przyspiesza czas obliczeń. Bez rekurencji ogonowej nie można rozwiązać problemu dla takich rozmiarów.

Testów nie przeprowadzono na większych danych, ponieważ niemożliwym było stworzenie większego problemu (przepełnienie stosu).

Przykład ten może odwzorowywać badania statystyczne na dużych większych zbiorach danych, jednak należy wtedy zoptymalizować zużycie problemu pamięci.