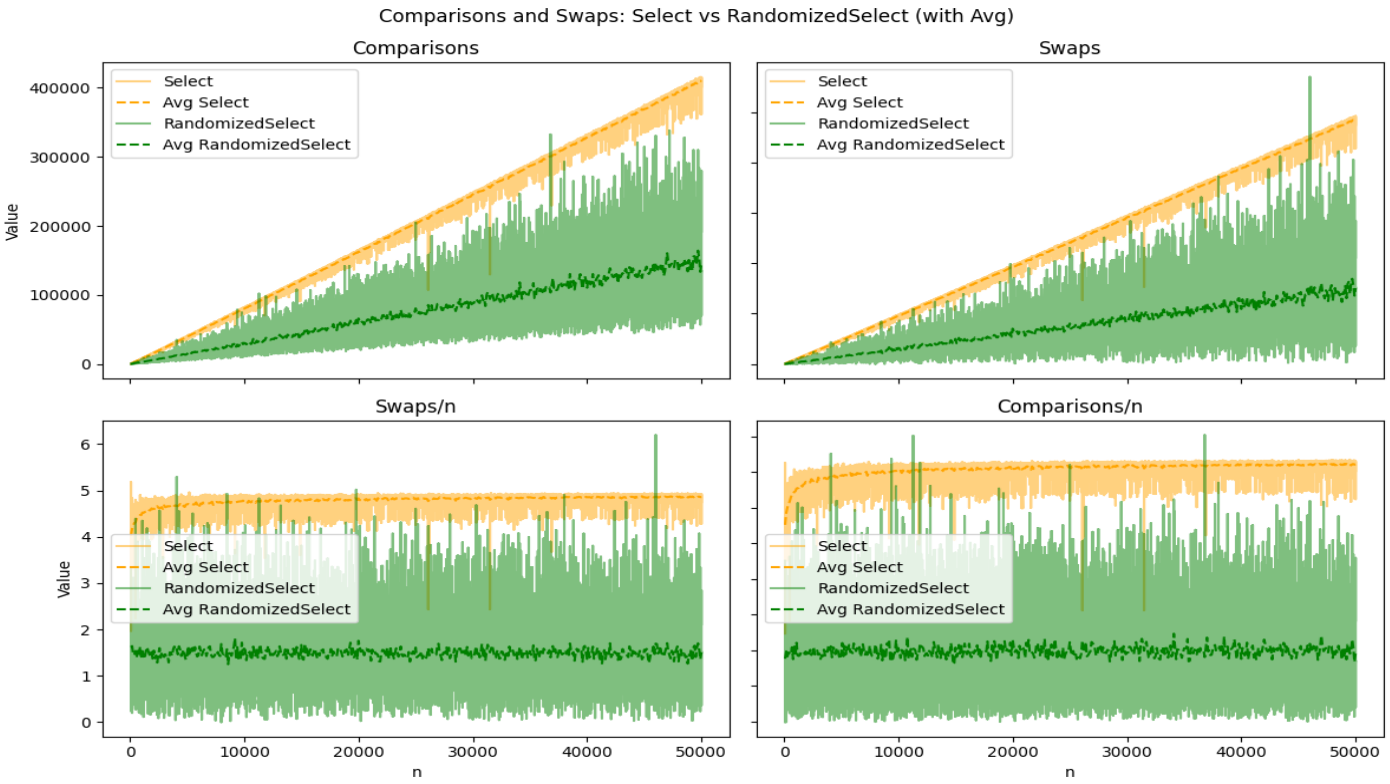
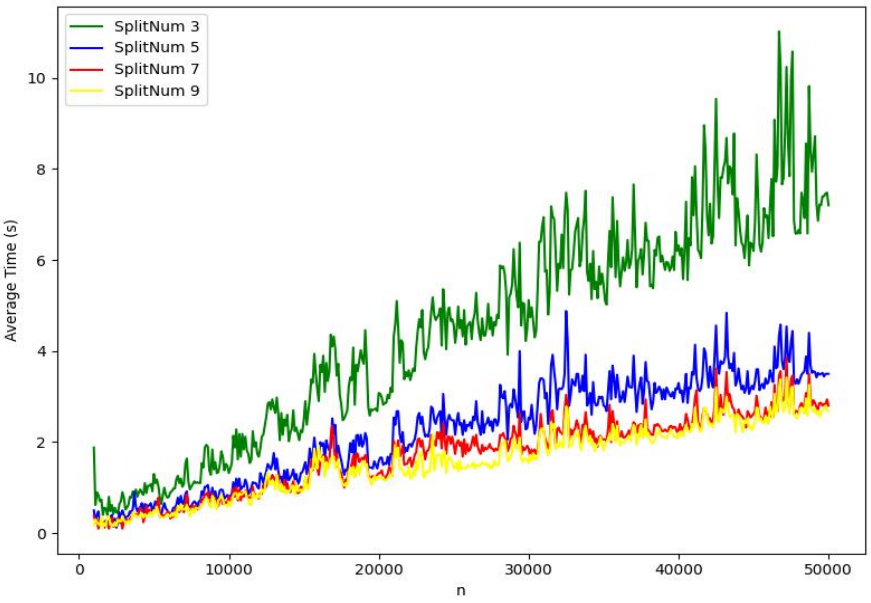
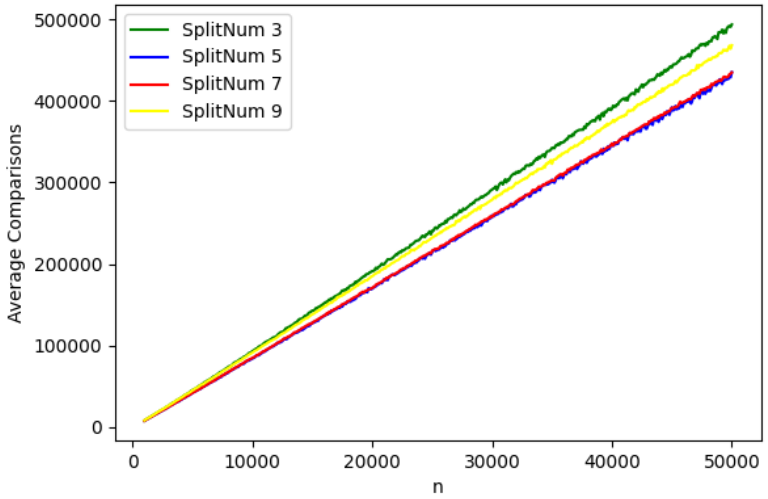
Zadanie 2:

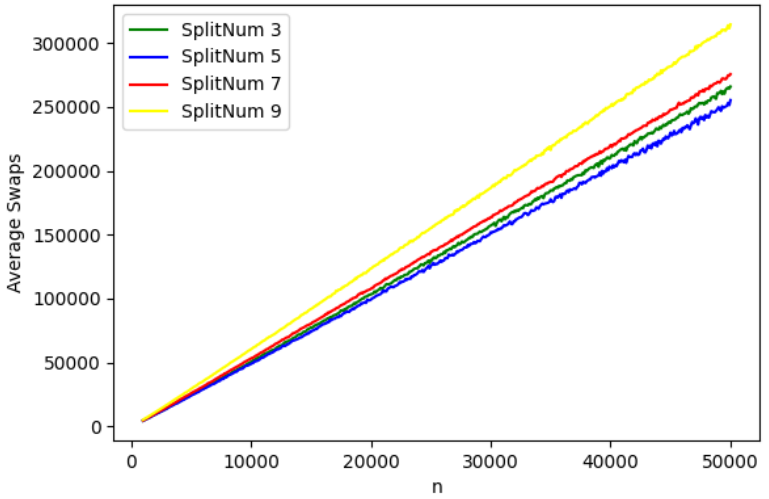


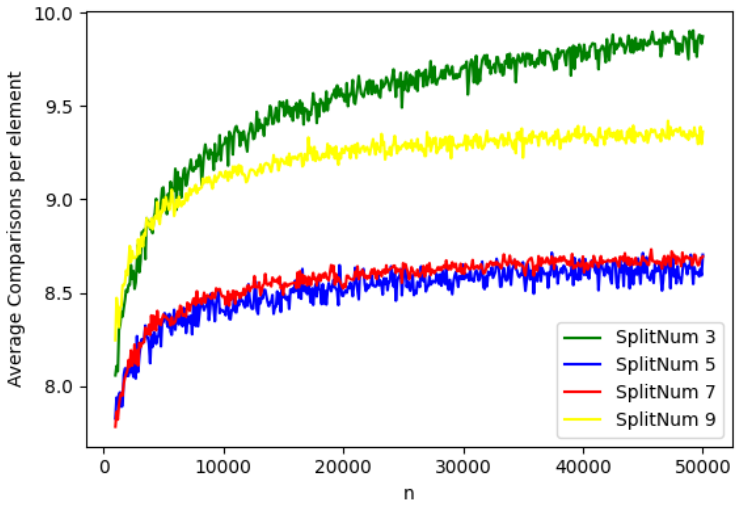
Patrząc na wykres, możemy zauważyć, że Select działa bardziej stabilnie, ponieważ wartość nie odbiega zbytnio od linii środkowej, w przypadku RandomizedSelect wartość znacznie odbiega od linii środkowej, ale Select potrzebuje więcej permutacji i porównań, ponieważ wybiera medianę median jako pivota, którego poszukiwanie kosztuje więcej. Istnieje kompromis między dwoma algorytmami. Select jest bardziej kontrolowany, ale intensywniejszy obliczeniowo pod względem porównań, podczas gdy RandomizedSelect potencjalnie oferuje szybsze średnie osiągi kosztem większej zmienności

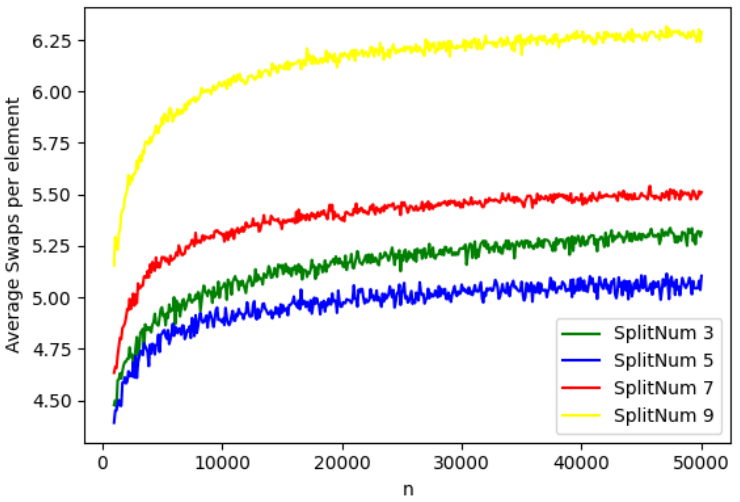
Zadanie 3:





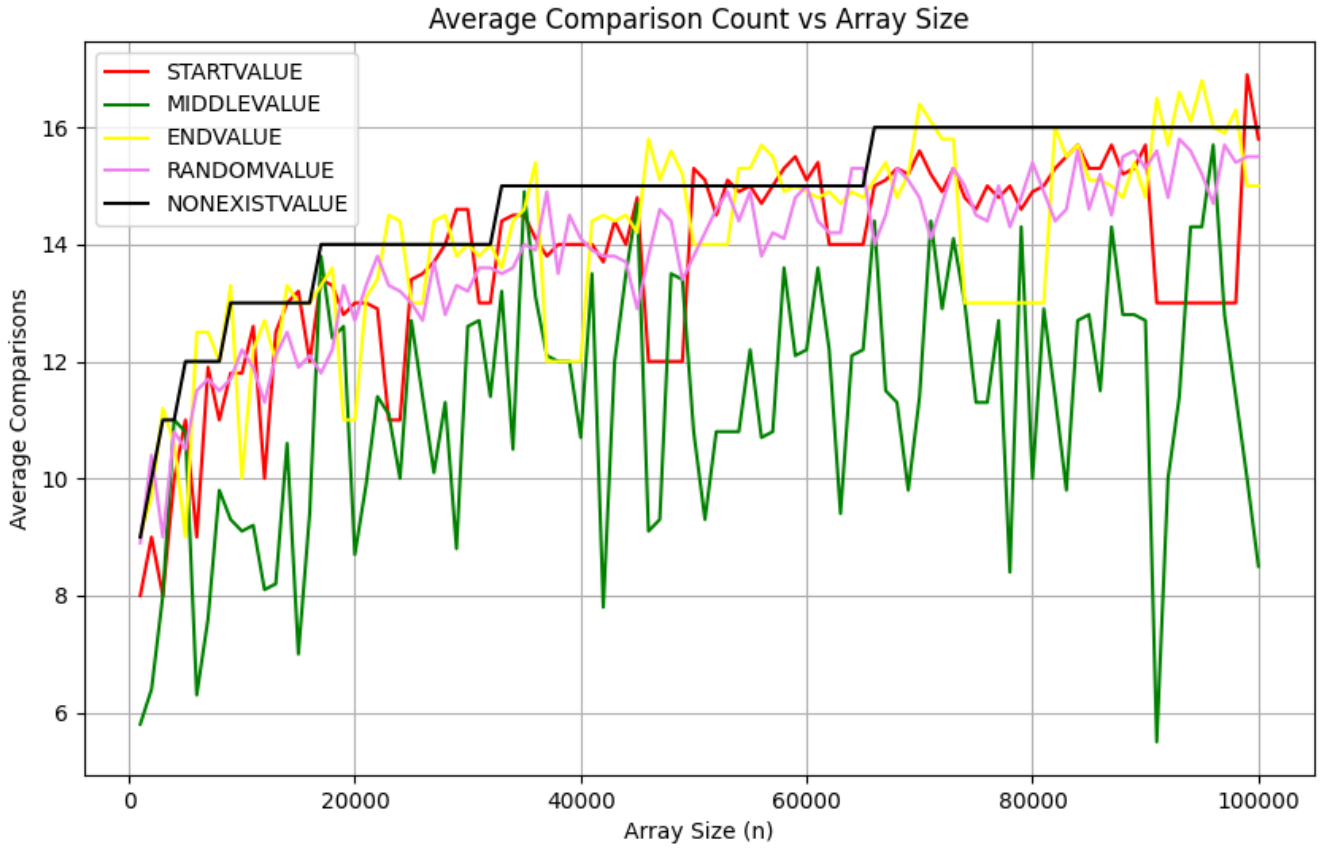


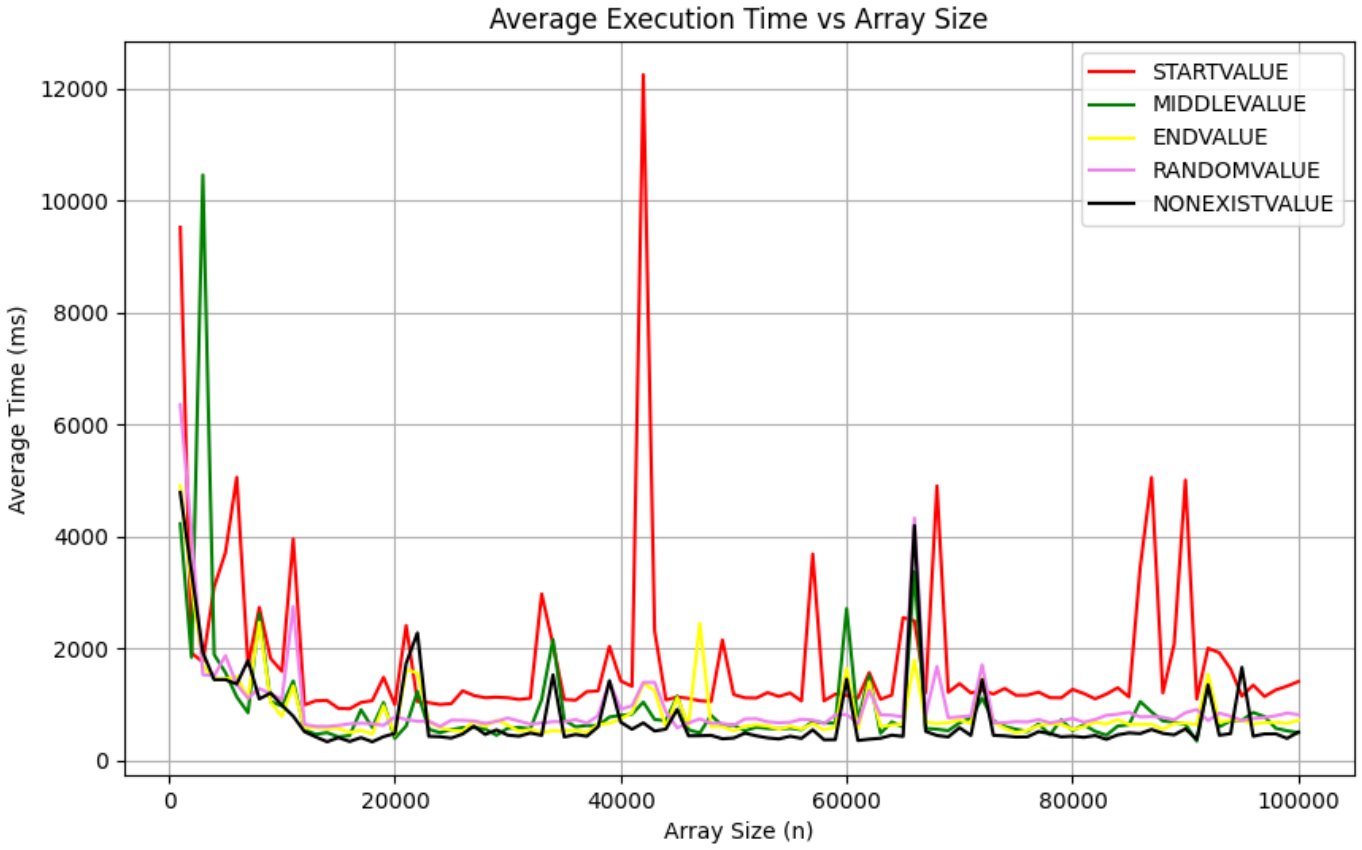




W przypadku rozmiaru 3 algorytm ma zwykle większy narzut ze względu na większą liczbę wywołań rekurencyjnych i mniej efektywną redukcję rozmiaru problemu na krok. W przypadku rozmiarów takich jak 7 lub 9 złożoność znalezienia mediany wzrasta, co może nie kompensować wystarczającej redukcji głębokości lub szerokości rekurencji w porównaniu z rozmiarem 5. Różnice w czasie działania przy różnych rozmiarach grup są wyraźne. Mniejsza grupa (SplitNum 3, zielona linia) wykazuje znacznie wyższy czas działania w porównaniu do większych grup, szczególnie przy wyższych wartościach n.Wykres pokazuje, że zwiększenie rozmiaru grupy prowadzi do bardziej wyraźnego zmniejszenia czasu działania. Trend ten można przypisać bardziej efektywnej segmentacji i mniejszej liczbie potrzebnych wywołań rekurencyjnych, gdy używane są większe grupy. Dlatego widać że 5 jest optymalnym wyborem. Podsumowując, wyniki te sugerują, że dostosowanie rozmiaru grupy w algorytmie Mediany Median może znacząco wpłynąć na ogólną wydajność, szczególnie dla większych zbiorów danych. Wybór rozmiaru grupy powinien uwzględniać zarówno wielkość danych

Zadanie 4:





Średnia Liczba Porównań:

Wykres porównań pokazuje stosunkowo stabilną średnią liczbę porównań we wszystkich kategoriach, waha się od około 10 do 16 porównań, pomimo zwiększania rozmiarów tablic. Jest to zgodne z oczekiwaną logarytmiczną złożonością wyszukiwania binarnego, O(logn).

Zauważalnie, 'NONEXISTVALUE' ma tendencję do nieco wyższej liczby porównań, co jest zgodne z najgorszym przypadkiem w wyszukiwaniu binarnym — sprawdzaniem maksymalnej liczby poziomów przed stwierdzeniem nieistnienia.

Czas Wykonania:

Czas wykonania wykazuje większą zmienność, ze szczególnymi szczytami dla 'STARTVALUE' i 'ENDVALUE'. Może to wynikać z takich czynników jak efektywność pamięci podręcznej i wzorce dostępu do pamięci, szczególnie w większych tablicach.

Stosunkowo stabilne czasy dla 'MIDDLEVALUE', 'RANDOMVALUE' i 'NONEXISTVALUE' sugerują, że średni i najgorszy czas wykonania są bliskie, prawdopodobnie ze względu na konsekwentnie niską liczbę gałęzi, które wyszukiwanie binarne musi sprawdzić przed uzyskaniem wyniku.

Szacowanie Współczynnika: O(1):

Dla Porównań:

Można wziąć średnią z liczb porównań przy wysokich wartościach n (od 80,000 do 100,000) dla każdej wartości docelowej.

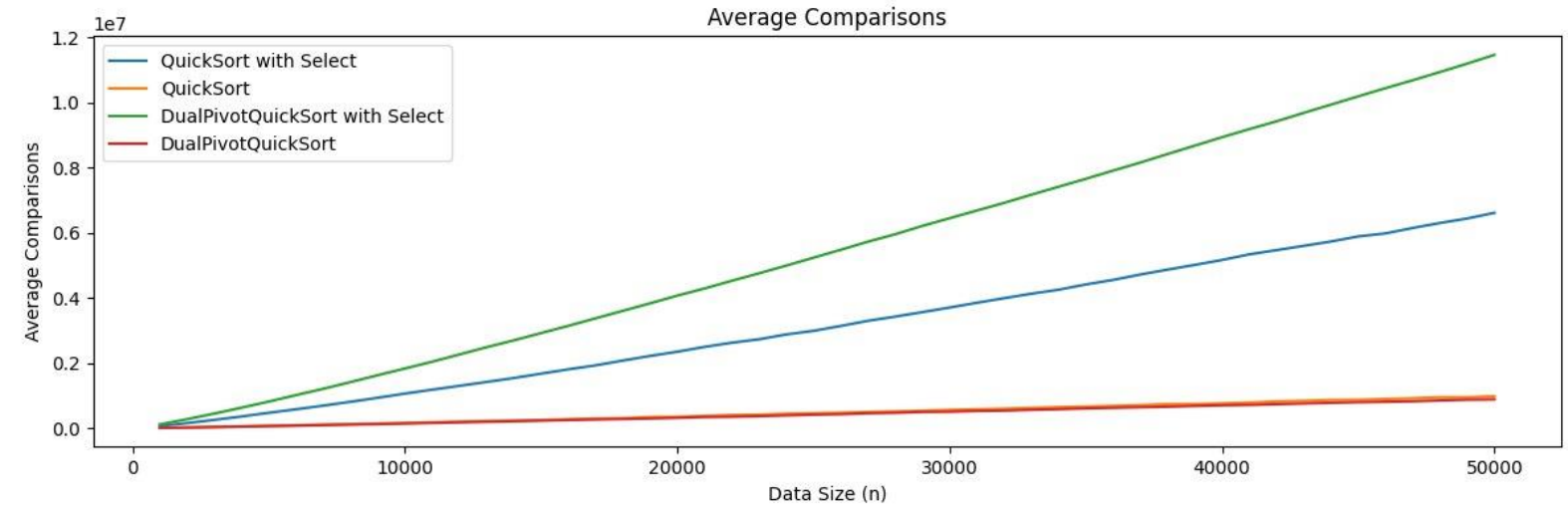
Ponieważ różnice między wartościami nie są znaczące, można oszacować współczynnik O(1) dla porównań jako medianę tych średnich, która wydaje się wynosić około 14 porównań.

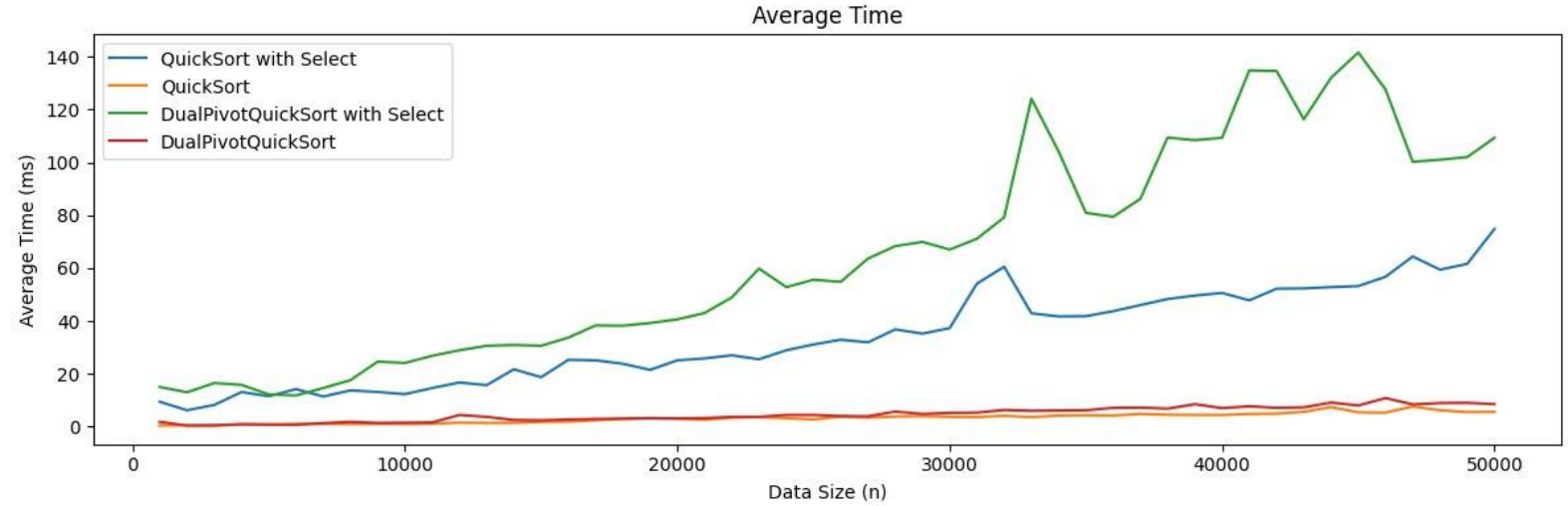
Dla Czasu Wykonania:

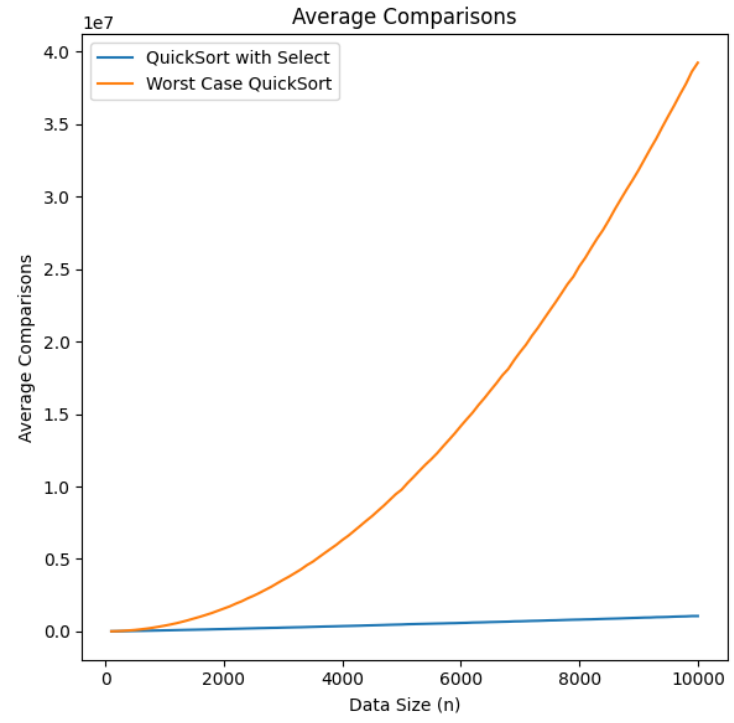
Rozważyć stabilne, niższe wartości czasu wykonania dla 'MIDDLEVALUE', 'RANDOMVALUE' i 'NONEXISTVALUE'.

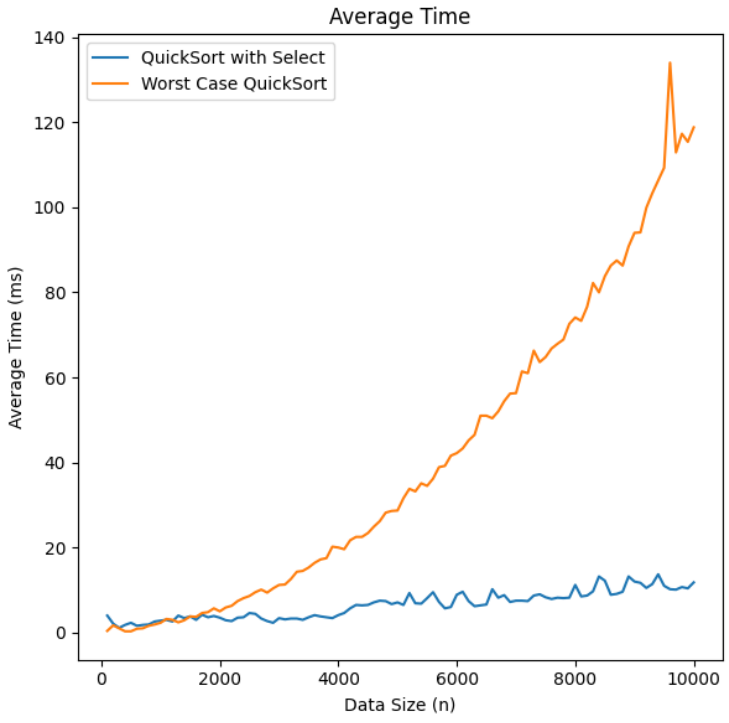
Najniższy poziom na wykresie czasu wykonania dla tych wartości można uznać za współczynnik O(1), sugerujący bazowy koszt czasowy algorytmu niezależnie od tego, czy wartość znajduje się na początku, w środku czy na końcu, który może wynosić około 2,000 ms dla dużych n.

Zadanie5:









Średnia liczba porównań i czas wykonania (pierwszy zestaw wykresów):

Pierwszy zestaw wykresów pokazuje, że DualPivotQuicksort konsekwentnie wypada lepiej niż inne warianty zarówno pod względem mniejszej liczby porównań, jak i krótszego czasu wykonania przy różnych rozmiarach danych. Utrzymuje on niemal stały, niski czas i liczbę porównań, co wskazuje na wysoką efektywność.

QuickSort i QuickSort z Select wykazują podobny wzrost liczby porównań, ale QuickSort z Select ogólnie osiąga lepszy czas wykonania. Sugeruje to pewne obciążenie związane z metodą Select, które niekoniecznie przekłada się na mniejszą liczbę porównań, ale pomaga w szybszym sortowaniu w określonych warunkach.

Wydajność w najgorszych przypadkach (drugi zestaw wykresów):

Dla danych w najgorszym przypadku, gdzie tradycyjny QuickSort miałby złożoność czasową kwadratową, wykres pokazuje wyraźną różnicę w wydajności. QuickSort z Select radzi sobie znacznie lepiej niż QuickSort w najgorszym przypadku.

Czas i liczba porównań QuickSorta w najgorszym przypadku szybko rosną w miarę zwiększania się rozmiaru danych, co podkreśla jego wrażliwość na pewne układy danych.