Algorytmy i struktury danych

Projekt 3

Projektodawca: Kamil Reczek

Michał Wolny | Patryk Tomaszewski
2020-06-06

Spis treści

1. Wprowadzenie do projektu 2
   a. Narzędzia użyte w projekcie 2
   b. Środowisko testowe 2
   c. Kod źródłowy 2
2. Dane oraz wyniki 3
   I. Część pierwsza 3
   II. Część druga 7
3. Podsumowanie 12

1. Wprowadzenie do projektu

Projekt zakłada przetestowanie algorytmów sortujących na różnych rodzajach tablic. Parametry jakie będą brane pod uwagę to jak dużo czasu algorytm potrzebuje do posortowania tablicy oraz jak zachowuje się w przypadkach gdy tablica już jest posortowana.

Algorytmy sortujące to:

- *SelectionSort* (sortowanie przez wybieranie) – $O(n^2)$
- *InsertionSort* (sortowanie przez wstawianie) – $O(n^2)$
- *CocktailSort* (sortowanie koktajlowe) – $O(n^2)$
- HeapSort (Sortowanie przez kopcowanie, sortowanie stogowe) - $O(n \cdot log(n))$

a. Narzędzia użyte w projekcie

Do przeprowadzenia badania dwóch algorytmów wyszukiwania zostały użyte następujące narzędzia:

- Program Microsoft Visual Studio 16.5.2
- Konsola CMD
- Pakiet Biurowy Microsoft Office (wykresy) oraz dokumentacja

b. Środowisko testowe

Testy zostały przeprowadzone na systemie operacyjnym Windows 10 Pro w wersji 1909. Za to za platformę testową posłużył komputer stacjonarny o następującej specyfikacji:

- CPU – AMD Ryzen 3700X
- RAM – 2x8 GB 3200MHz
- MB – MSI B350
- Dysk – M.2 PCIe 3.0 256GB

c. Kod źródłowy

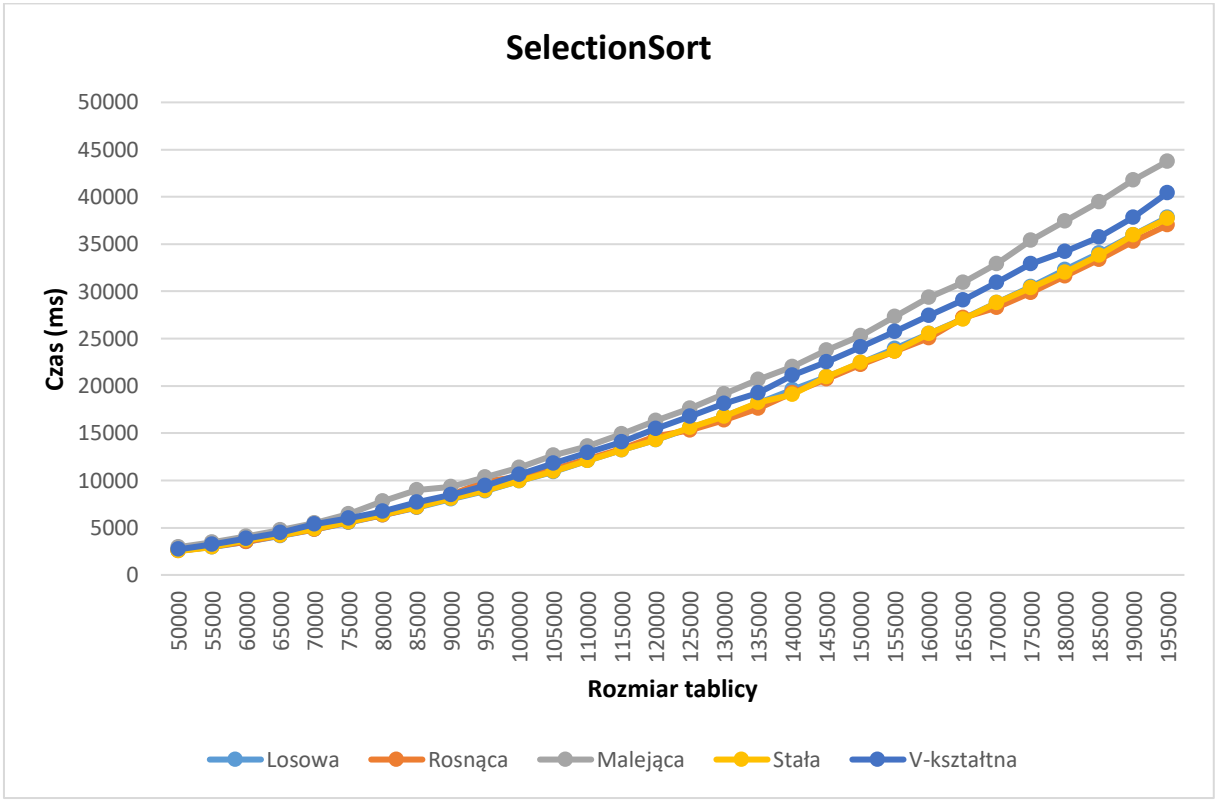
Kod źródłowy można znaleźć na repozytorium GitHub pod tym: https://github.com/MichalWolnyDev/Projekt_3

2. Dane oraz wyniki

I. Część pierwsza

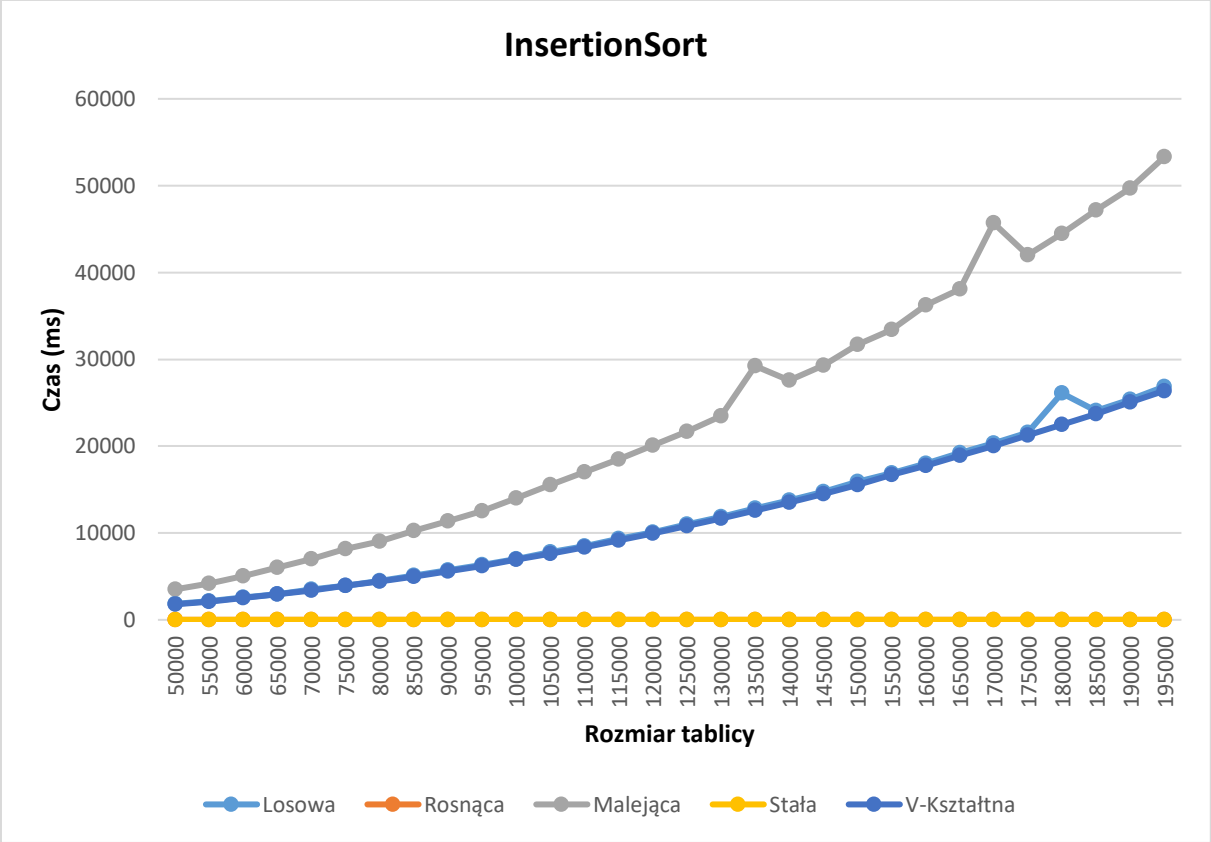
Część pierwsza projektu polega na pomiarze czasu jaki jest potrzebny danemu algorytmowi do posortowania różnych tablic.

a) SelectionSort



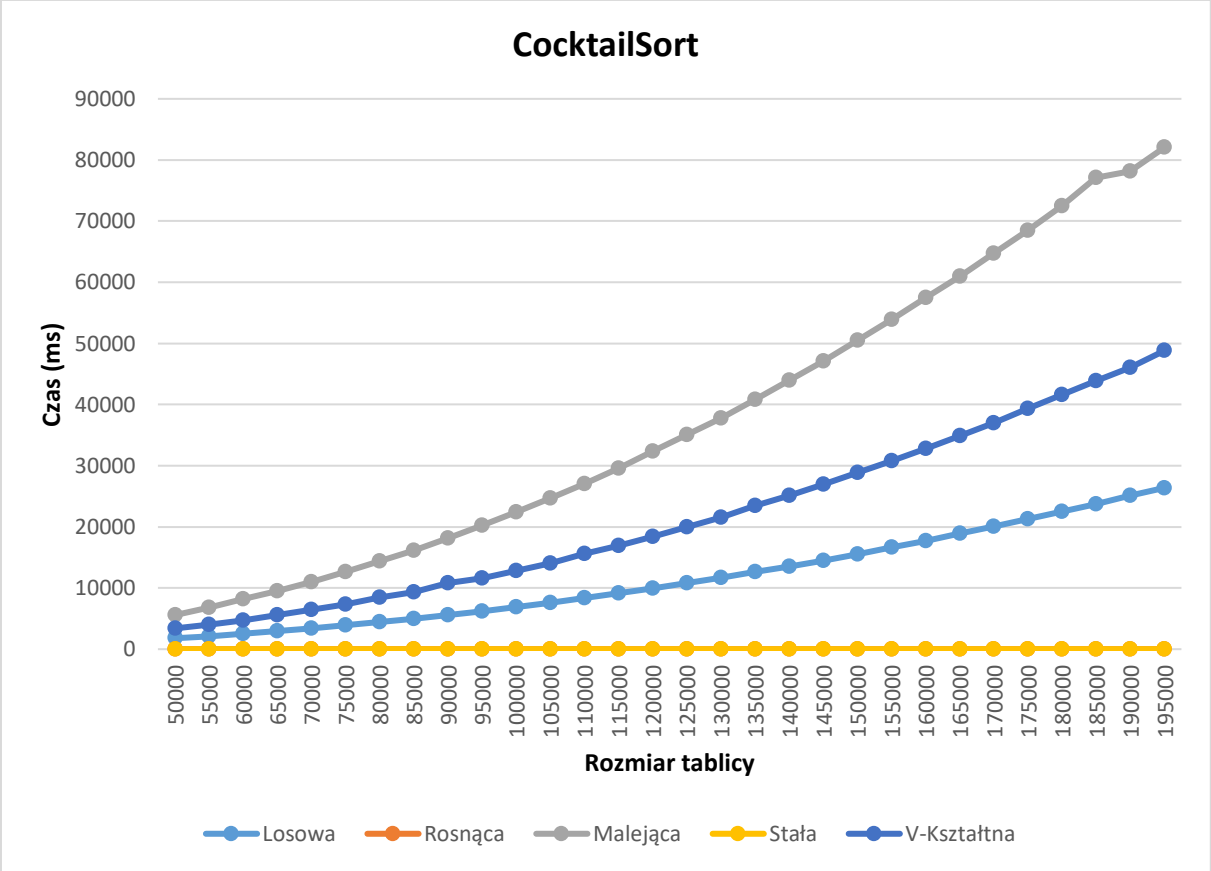
Optymistyczna złożoność obliczeniowa wynosi O(n) wtedy kiedy ciąg wejściowy jest rosnący. Jednak złożoność obliczeniowa zmianie w przypadkach gdy tablica jest malejąca lub posortowana w mniejszym stopniu. Złożoność w takich przypadkach będzie wynosić $O(n^2)$.

b) InsertionSort



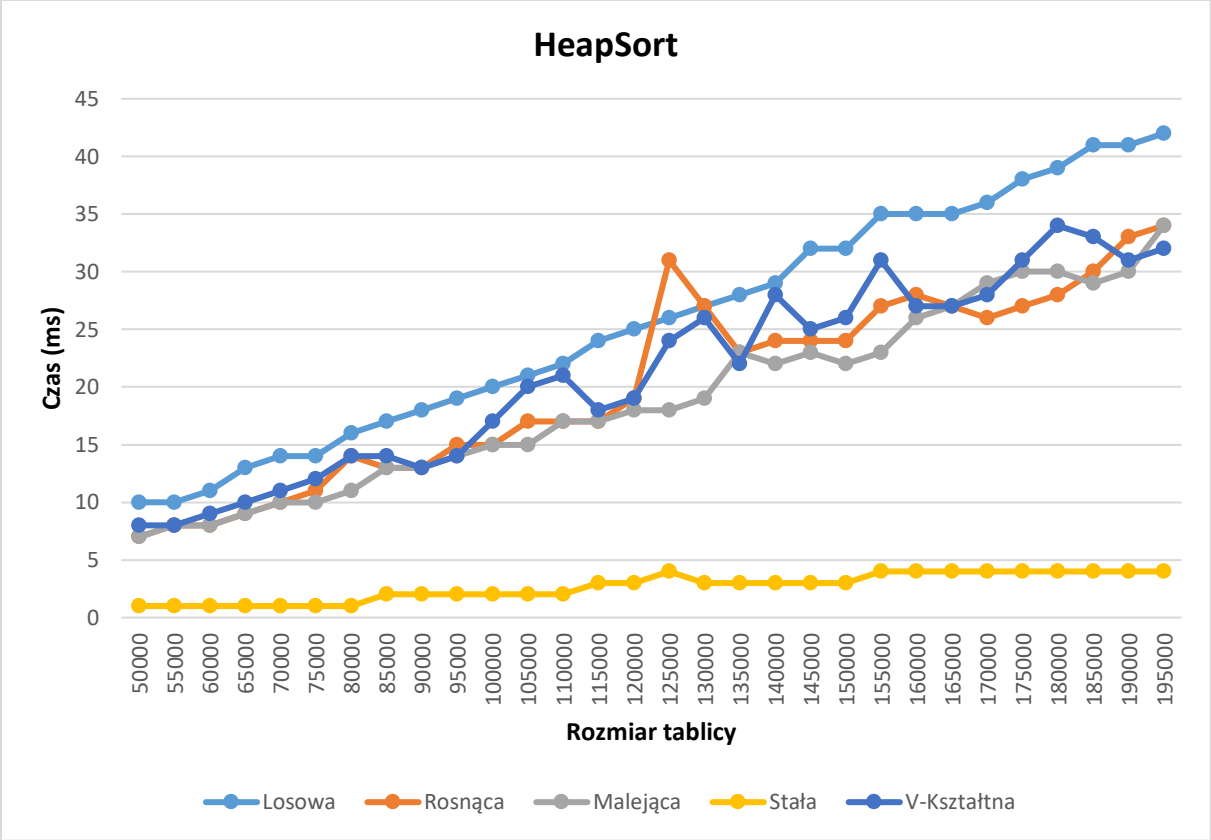
Na wykresie czas tablicy stałej wynosi 1ms tak samo jak dla tablicy rosnącej. Jak widać na wykresie algorytm jest wpełni zależny od danych wejściowych. Im mniej posortowana tablica tym więcej czasu potrzebuje algorytm na jego wykonanie. W najgorszym przypadku jego złożoność obliczeniowa nadal wynosi $O(n^2)$.

c) CocktailSort



Na wykresie czas tablicy stałej wynosi 1ms tak samo jak dla tablicy rosnącej. Jak widać na wykresie algorytm jest wpełni zależny od danych wejściowych. Im mniej posortowana tablica tym więcej czasu potrzebuje algorytm na jego wykonanie. Złożoność obliczeniowa wynosi $O(n^2)$.

d) HeapSort

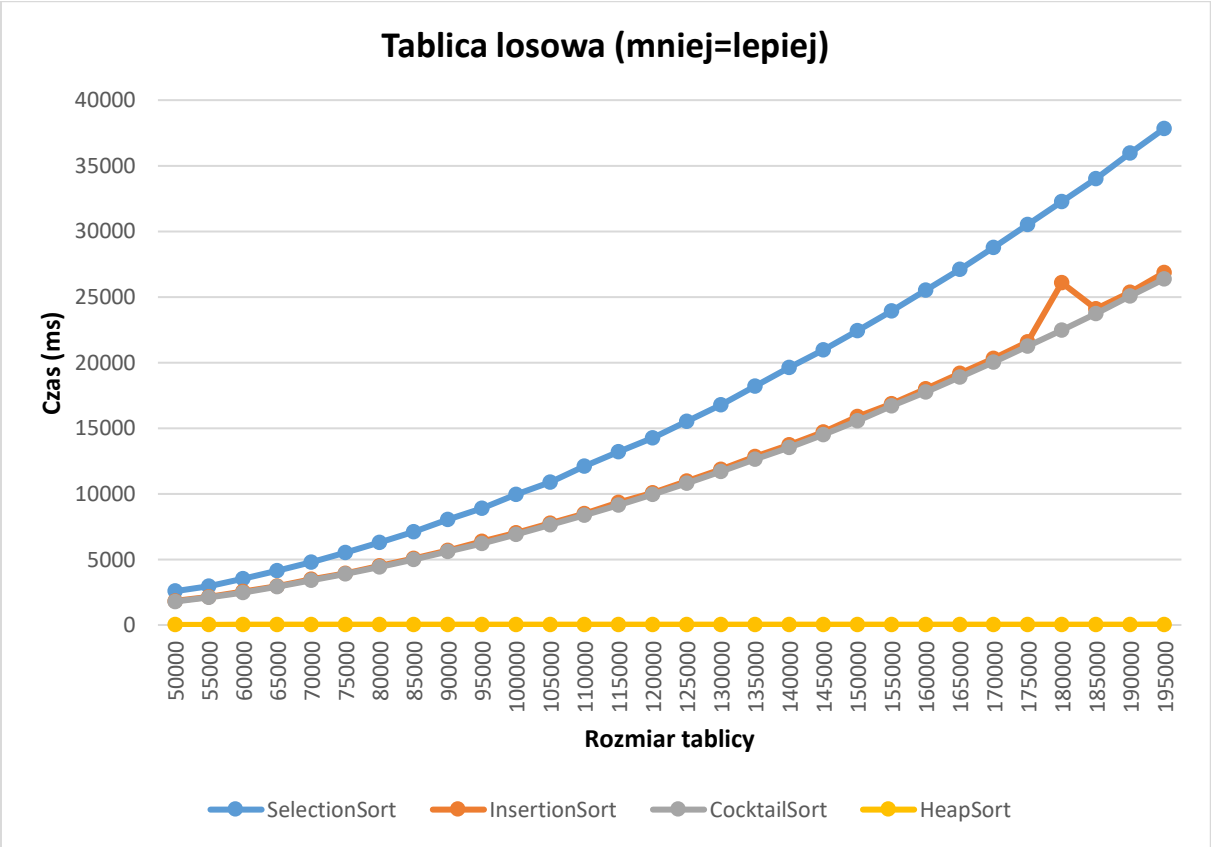


Złożoność algorytmu sortującego HeapSort wynosi $O(n \cdot log(n))$. Różni się tylko w momencie gdy tablica już była posortowana. Wtedy algorytm tylko sprawdza poprawność tablicy. Czas wykonania operacji sortowania nie jest zależny od rodzaju tablicy (z wyjątkiem już posortowanej).

II. Część druga

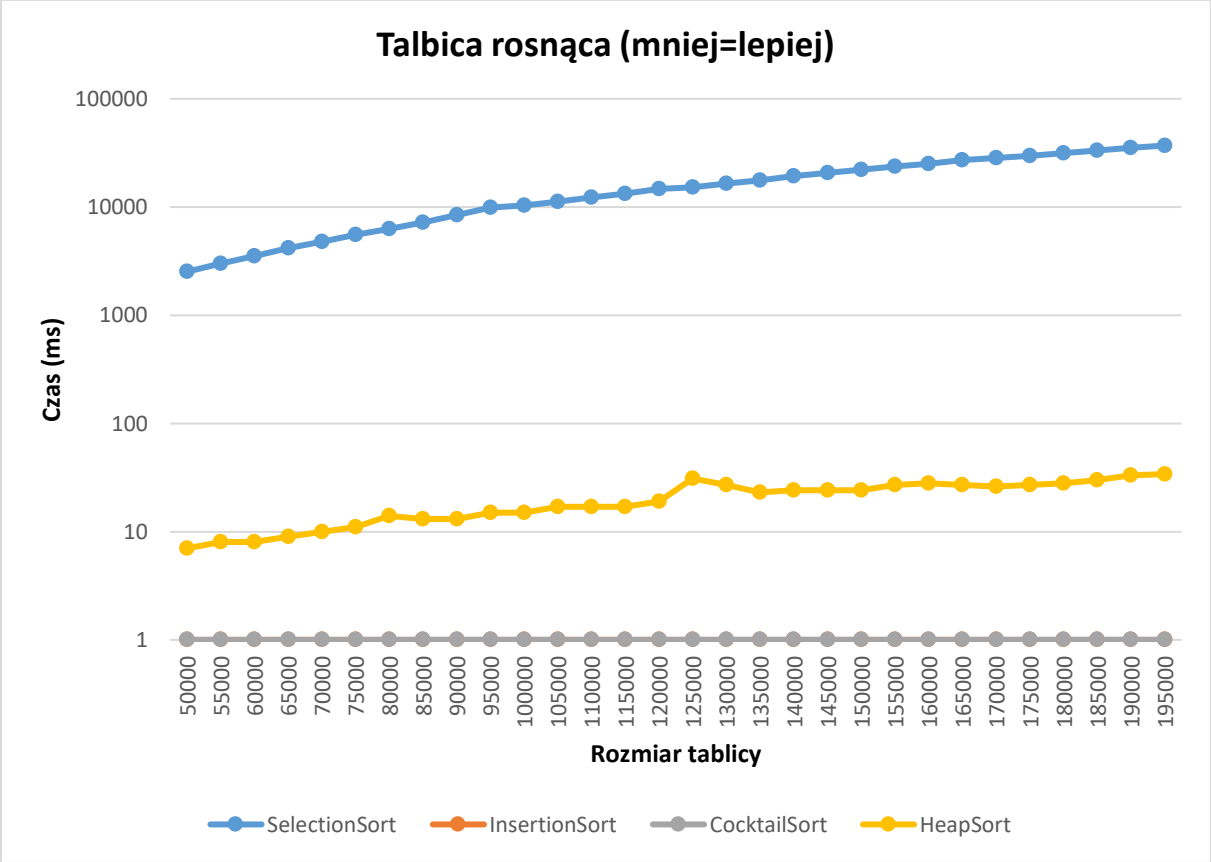
Część druga projektu ma wykazać różnice w czasach 4 algorytmów sortujących dla konkretnej tablicy.

a) Tablica losowa



Dla tablicy losowej zdecydowanym liderem jest algorytm sortujący HeapSort. Reszta algorytmów niestety wraz ze wzrostem rozmiaru tablicy potrzebuje coraz to więcej czasu do zakończenia sortowania.

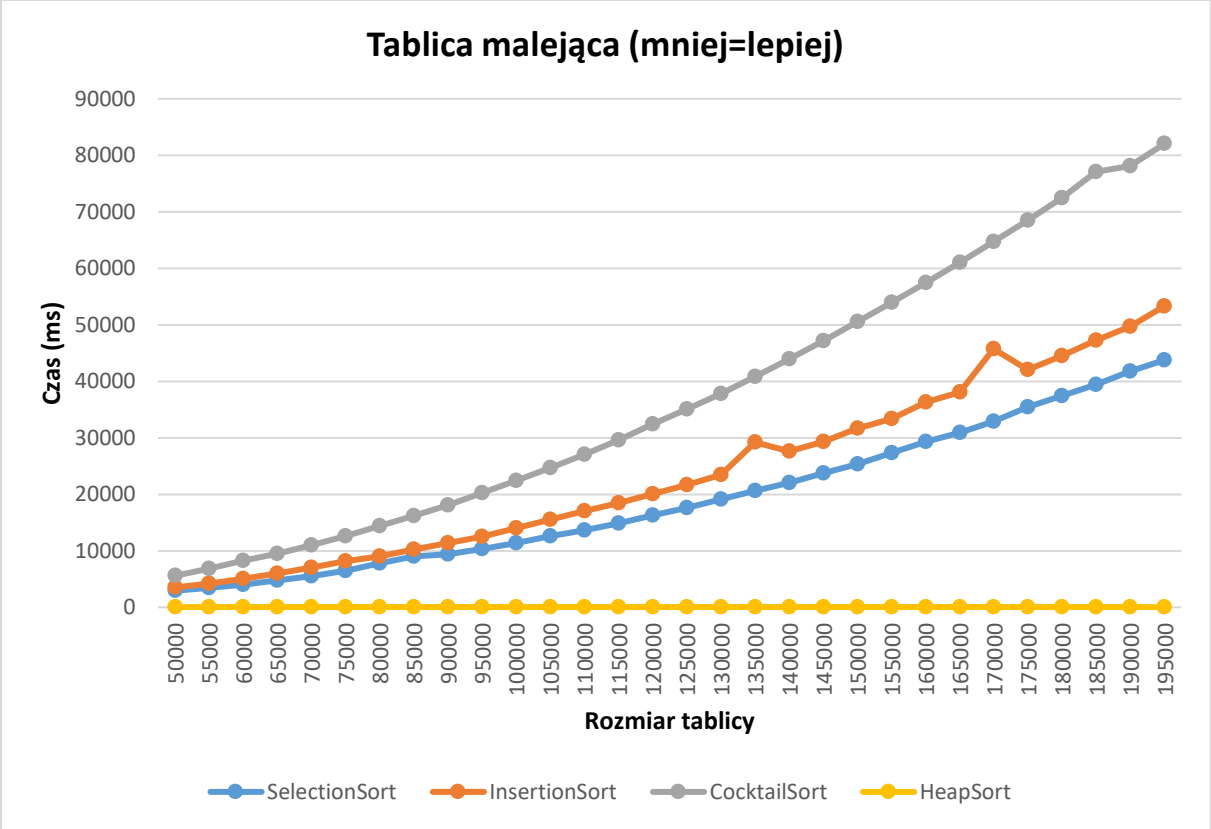
b) Tablica rosnąca



Wykres został przedstawiony w skali logarytmicznej ze względu na małe wartości jakie wygenerowały algorytmy HeapSort oraz CocktailSort. CocktailSort oraz InsertionSort mają wynik na poziomie 1ms. Złożoność obliczeniowa SelectionSort nadal wynosi $O(n^2)$.

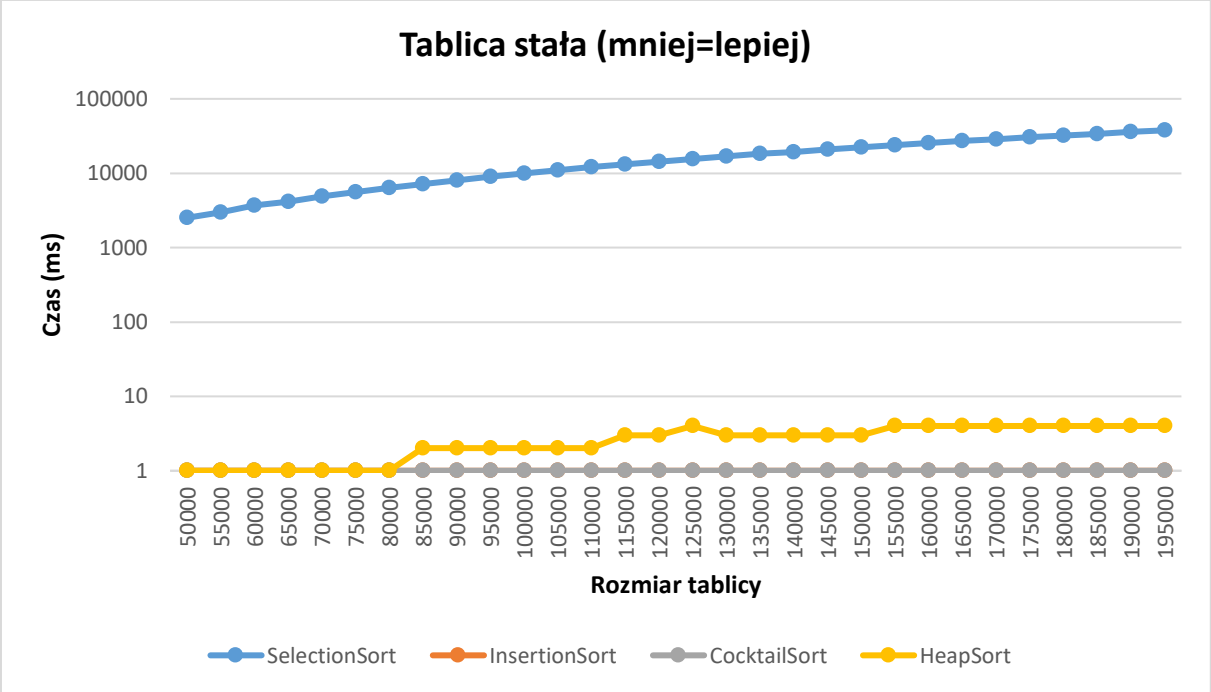
Liderem przy tablicy rosnącej są CocktailSort oraz InsertionSort gdyż najszybciej sprawdzają już posortowane talbice.

c) Tablica malejąca



W przypadku odwracania tablicy posortowanej malejąco najszybciej z posortowaniem radzi sobie algorytm HeapSort. Reszta algorytmów wraz ze wzrostem rozmiaru tablicy potrzebuje coraz więcej czasu. Najgorzej radzi sobie algorytm CocktailSort.

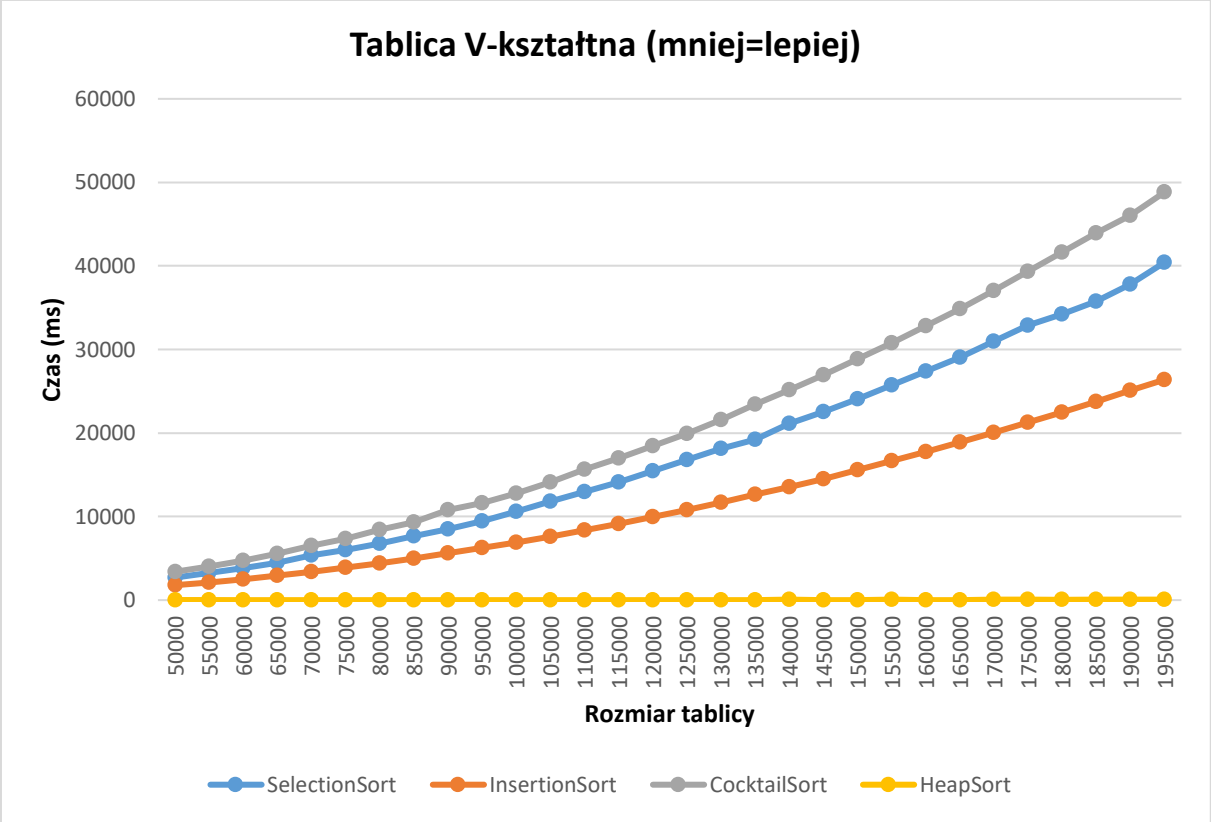
d) Tablica stała



Wykres został przedstawiony w postaci logarytmicznej ze względu na małe wartości uzyskane przy badaniu 3 z 4 algorytmów.

Tak jak w przypadku tablicy rosnącej HeapSort potrzebuje parę milisekund więcej niż algorytm CocktailSort oraz InsertionSort (CocktailSort = InsertionSort = 1ms). Zdecydowanym przegranym jest znowu SelectionSort gdyż im więcej elementów w tablicy tym czas znacznie się wydłuża.

e) Tablica V-kształtna



Tablica V-Kształtna jest idealną tablicą do testów gdyż daje porównywalne wyniki dla każdego powtórzonego testu. Można zauważyć, iż **HeapSort** jest zdecydowanym liderem pod względem sortowania tego typu tablicy **(1-32ms).** Najgorszy w tym przypadku jest CocktailSort.

3. Podsumowanie

W eksperymencie przeprowadzonym można wysnuć następujący wniosek. Do sortowania tablic zdecydowanym liderem jest algorytm HeapSort. Utrzymuje on bardzo przyzwoite wyniki nawet przy bardzo dużych tablicach. Jest minimalnie wolniejszy w przypadkach gdy trzeba sprawdzić czy tablica jest posortowana.

Najgorszym algorytmem do sortowania jest CocktailSort. Ma on bardzo duże czasy gdy trzeba wykonać dużo operacji przenoszenia liczb. Z kolei InsertionSort jest najgorszym algorytmem do sprawdzania tablic stałych lub już posortowanych.