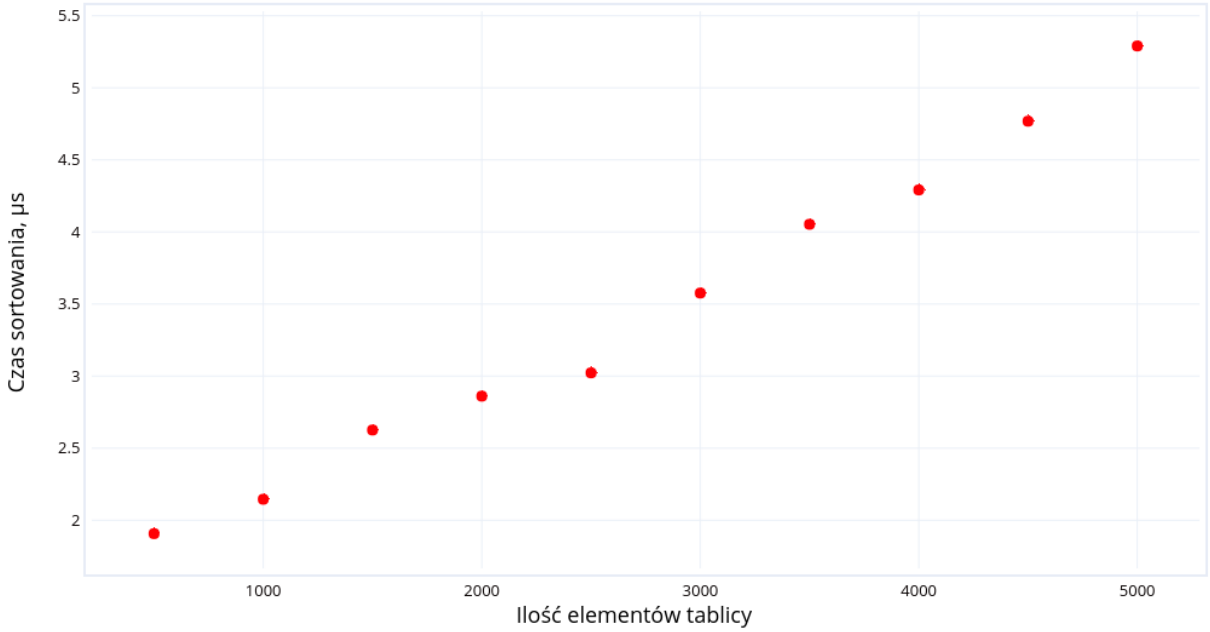
Czas wykonania programu dla różnej ilośći danych(μs)

Tablica losowych liczb w przedziale [0, 100] o rozmiarze n(od 500 do 5000)

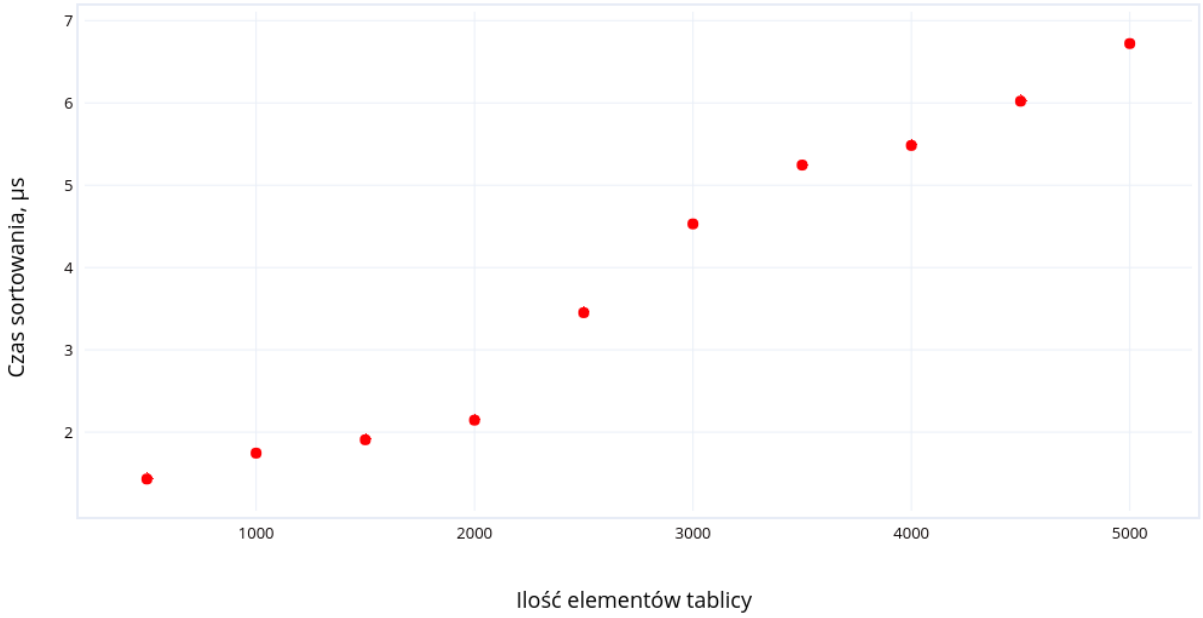
n	*Selection Sort*	*Insertion Sort*	*Bubble Sort*	*Quick Sort*	*Merge Sort*	*Bucket Sort*	*Heap Sort*	*Radix Sort*
500	1.907	1.431	1.192	1.192	1.431	1.192	4.053	1.192
1000	2.146	1.745	1.669	1.669	1.907	1.192	5.010	0.954
1500	2.626	1.907	1.907	1.667	2.165	1.669	5.245	1.430
2000	2.861	2.146	1.907	1.907	2.146	1.667	5.722	1.669
2500	3.023	3.451	2.157	2.146	2.146	2.146	7.106	1.450
3000	3.576	4.530	2.861	2.384	2.384	2.146	9.298	1.907
3500	4.053	5.245	3.099	2.861	2.861	2.623	15.259	2.146
4000	4.296	5.484	4.291	3.099	3.338	3.219	18.597	2.384
4500	4.768	6.021	4.786	4.053	3.815	3.576	20.980	2.861
5000	5.290	6.721	5.007	4.530	4.053	4.292	23.841	3.099

Selection Sort



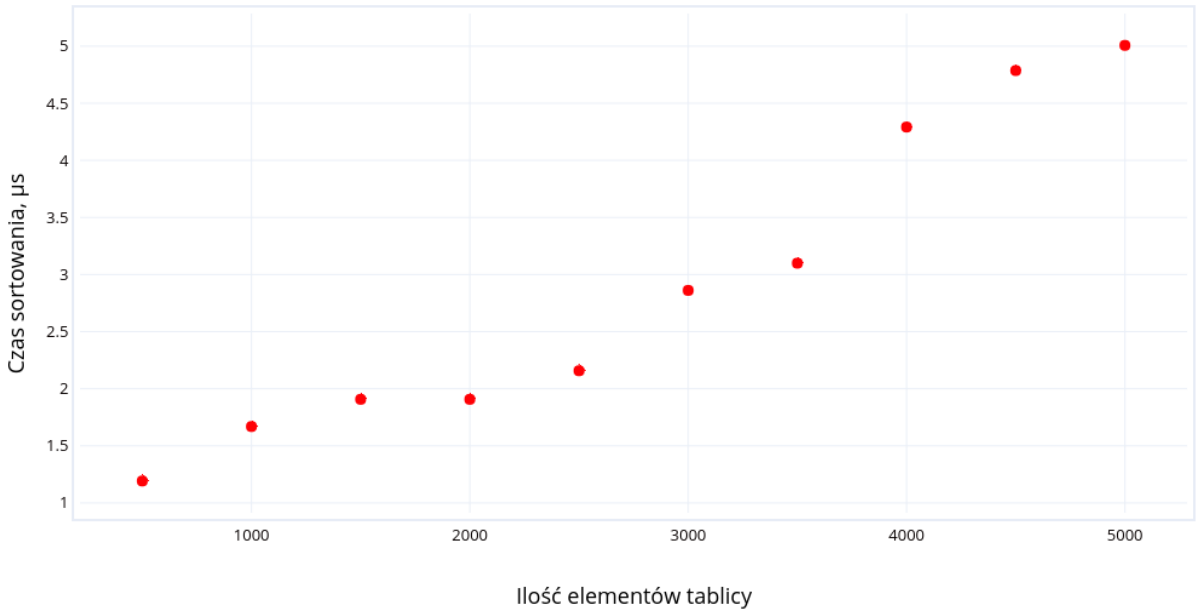
- Z internetu:
  - Złożoność czasowa: $O(n^2)$
  - Złożoność pamięciowa: $O(1)$
- Z własnej implementacji:
  - Złożoność czasowa: $O(n^2)$
  - Złożoność pamięciowa: $O(1)$
- Warto użyć do sortowania niewielkiej ilości danych

Insertion Sort



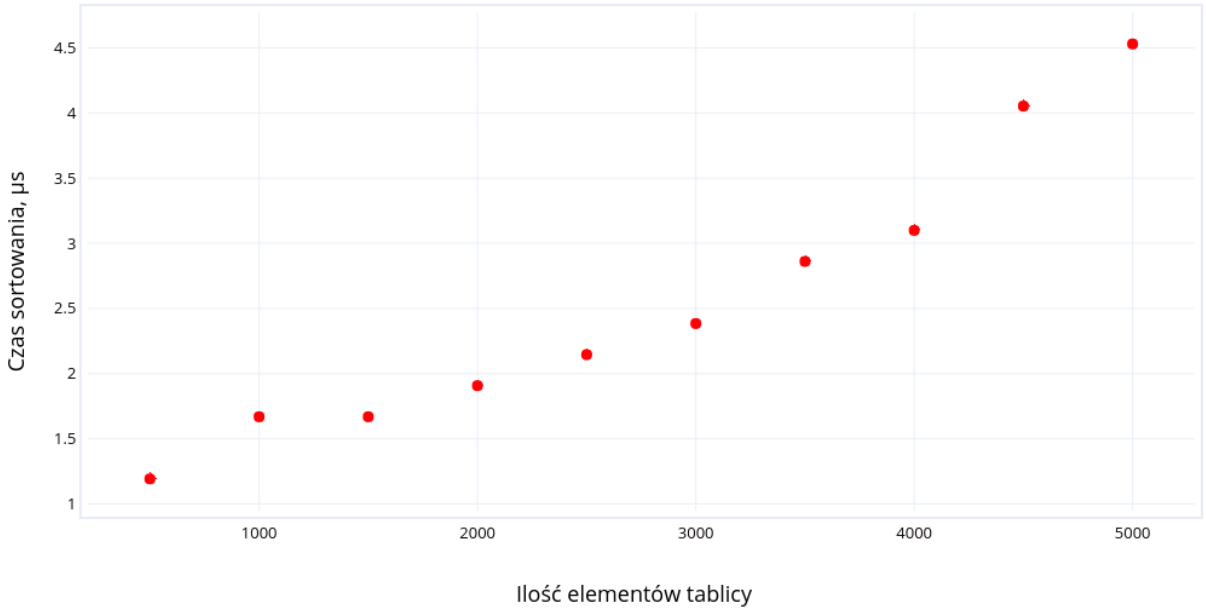
- Z internetu:
  - Złożoność czasowa: $O(n^2)$
  - Złożoność pamięciowa: $O(1)$
- Z własnej implementacji:
  - Złożoność czasowa: $O(n^2)$
  - Złożoność pamięciowa: $O(1)$
- Warto używać do sortowania niewielkiej ilości danych lub sortowania tablic, które już są prawie posortowane(tzn. potrzebują niewiele zamian)

Bubble Sort



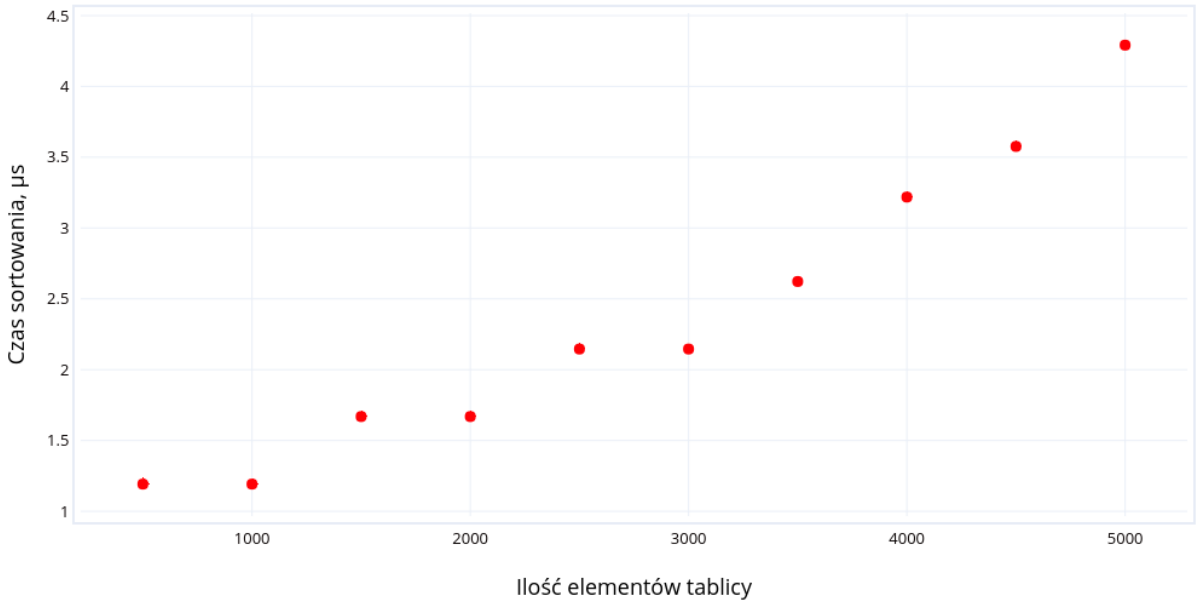
- Z internetu:
  - Złożoność czasowa: $O(n^2)/O(n)$
  - Złożoność pamięciowa: $O(1)$
- Z własnej implementacji:
  - Złożoność czasowa: $O(n^2)$
  - Złożoność pamięciowa: $O(1)$
- Jeden z najszybszych algorytmów dla sortowania małych lub częściowo posortowanych zbiorów danych.

Quick Sort



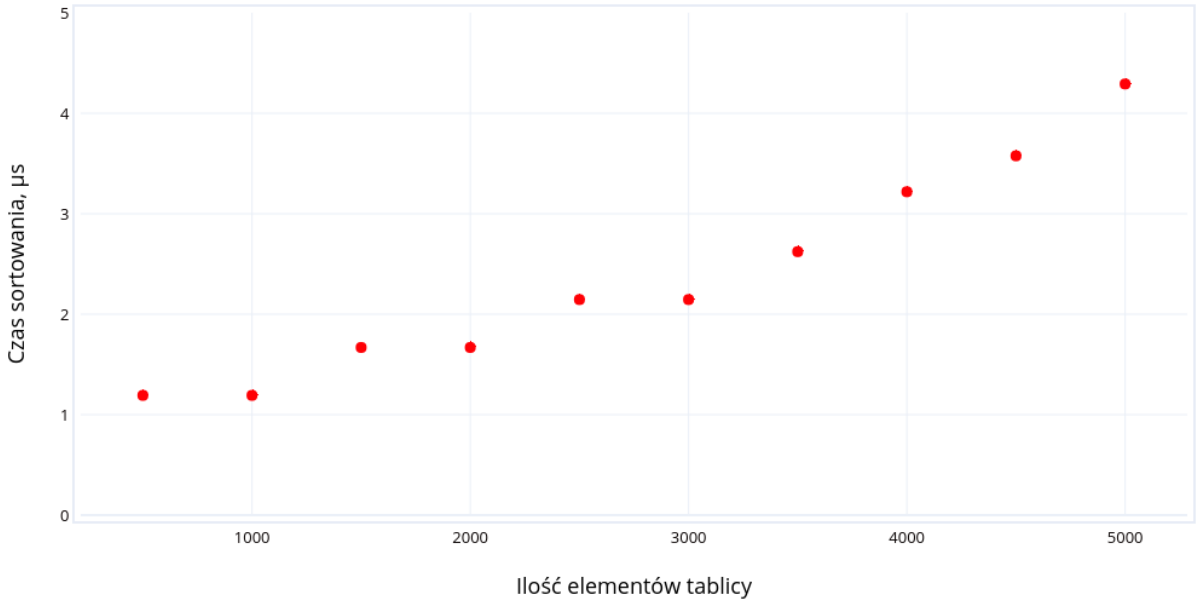
- Z internetu:
  - Złożoność czasowa: $O(n^2)/O(n\ln(n))$
  - Złożoność pamięciowa: $O(1)$
- Z własnej implementacji:
  - Złożoność czasowa: $O(n\ln(n))$
  - Złożoność pamięciowa: $O(1)$
- Jest często używany dla sortowania tablic ze względu na szybkość i niską złożoność pamięciową.

Merge Sort



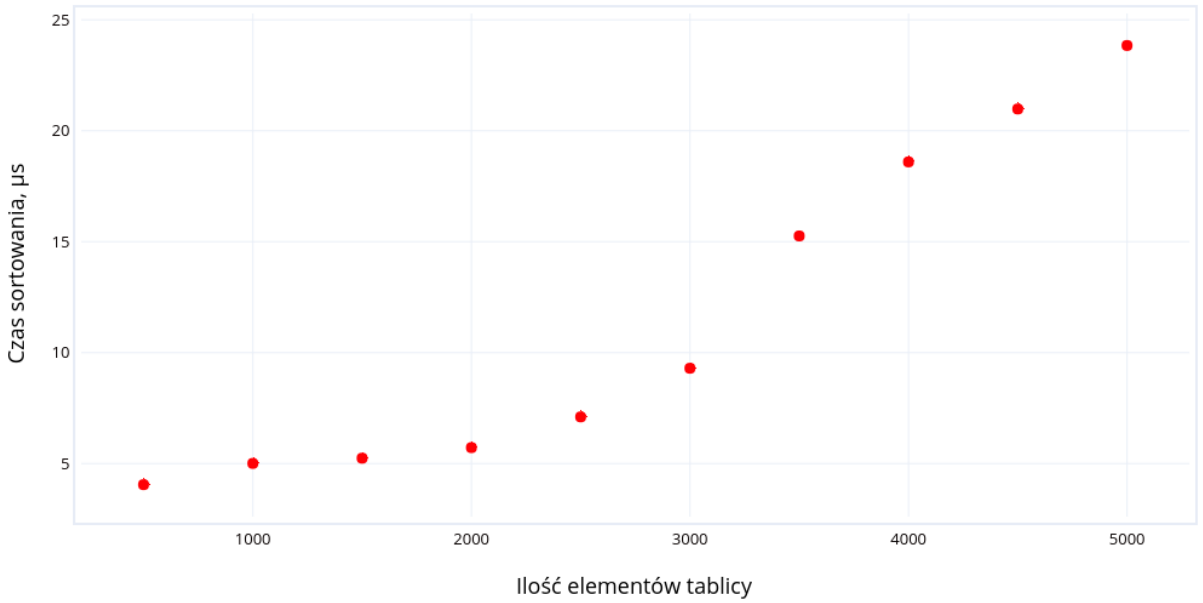
- Z internetu:
  - Złożoność czasowa: $O(n \ln(n))$
  - Złożoność pamięciowa: $O(n)$
- Z własnej implementacji:
  - Złożoność czasowa: $O(n \ln(n))$
  - Złożoność pamięciowa: $O(n)$
- Używa się do sortowania list oraz jest preferowany jeśli sortuje się zbiór liczb o niewielkiej różnice(ponieważ w tym przypadku Quick Sort ma złożoność obliczeniową $O(n^2)$)

Bucket Sort



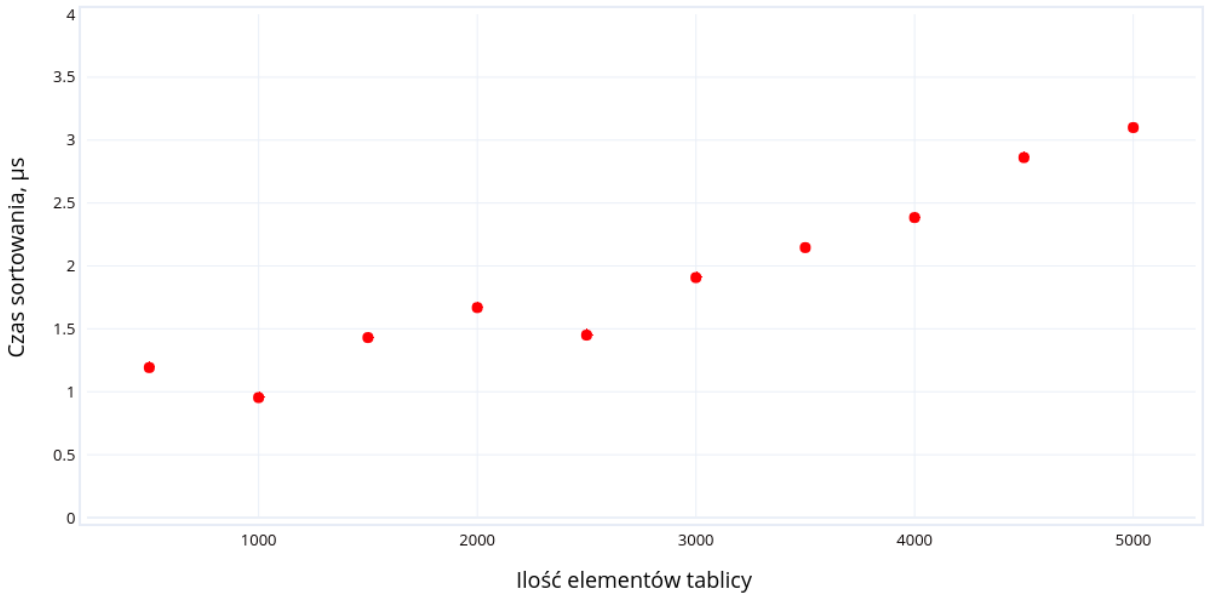
- Z internetu:
  - Złożoność czasowa: $O(n+k)/O(n^2)$
  - Złożoność pamięciowa: $O(n+k)$
- Z własnej implementacji:
  - Złożoność czasowa: $O(n+k)$
  - Złożoność pamięciowa: $O(n+k)$
- Jest najszybszym sortowaniem, gdy sortowane elementy można równomiernie podzielić na partycje.

Heap Sort



- Z internetu:
  - Złożoność czasowa: $O(n \ln(n))$
  - Złożoność pamięciowa: $O(n)$
- Z własnej implementacji:
  - Złożoność czasowa: $O(n \ln(n)$
  - Złożoność pamięciowa: $O(n)$
- Rzadko używany ze względu na szybkość algorytmów Quick Sort i Merge Sort. Czasami używa się do poszukiwania największego i najmniejszego elementów tablicy.

Radix Sort



- Z internetu:
  - Złożoność czasowa: $O(nk)$
  - Złożoność pamięciowa: $O(n+k)$
- Z własnej implementacji:
  - Złożoność czasowa: $O(nk)$
  - Złożoność pamięciowa: $O(n+k)$
- Warto użyć do sortowania zbiorów zawierających bardzo duże i/lub bardzo podobne liczby.