

Algorytmy sortowania

Kacper Leporowski¹

Adam Malinowski

W tym sprawozdaniu porównane będą wybrane algorytmy sortowania pod względem ich złożoności, czasu wykonania i wpływu różnych danych wejściowych na te kryteria – danych, dla których algorytm będzie wykonywał się dłużej lub krócej w porównaniu do danych losowych.

Rodzaje danych, które zostały sprawdzone, to dane losowe, dane rosnące, dane malejące, dane A-kształtne i dane stałe. Zestawy liczb przeznaczonych do posortowania miały rozmiary od 1000 do 10000 elementów z krokiem 1000, a w przypadkach, w których taka liczba danych nie pozwalała na miarodajny pomiar czasu wykonania wykorzystano listy o wymiarach od 50000 do 500000 elementów, z krokiem 50000.

Wykorzystane algorytmy sortowania to:

- sortowanie przez wybieranie (selection sort)
- sortowanie przez wstawianie (insertion sort)
- sortowanie Shella (Shell sort)
- sortowanie przez kopcowanie (heap sort)
- sortowanie szybkie (quick sort)

Sortowanie przez wybieranie

Sortowanie przez wybieranie to algorytm sortowania w miejscu o złożoności obliczeniowej $O(n^2)$ – w programie występują dwie zagnieżdżone pętle.

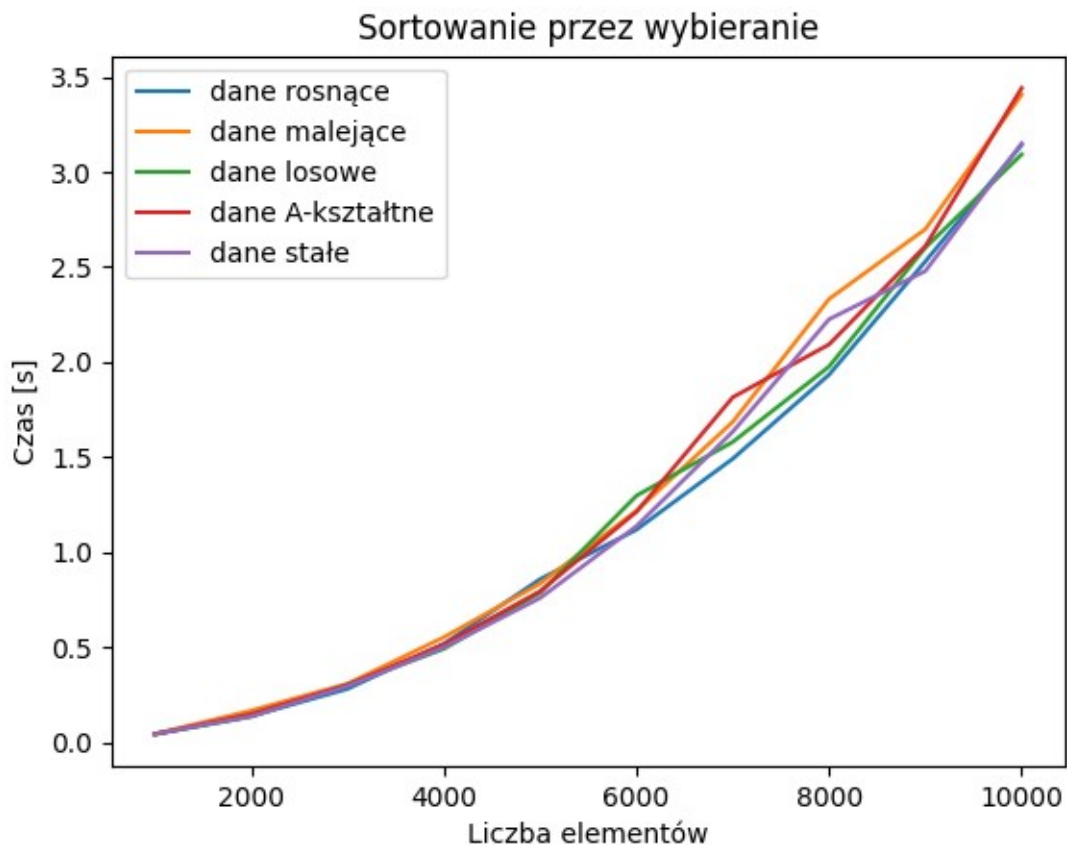


Figura 1: Czas wykonywania - selection sort

Złożoność i czas wykonania algorytmu są niezależne od rodzaju danych wejściowych – dla każdego zbioru algorytm wykonuje się w tym samym czasie.

Sortowanie przez wstawianie

Sortowanie przez wstawianie ma złożoność $O(n^2)$ i sortuje w miejscu.

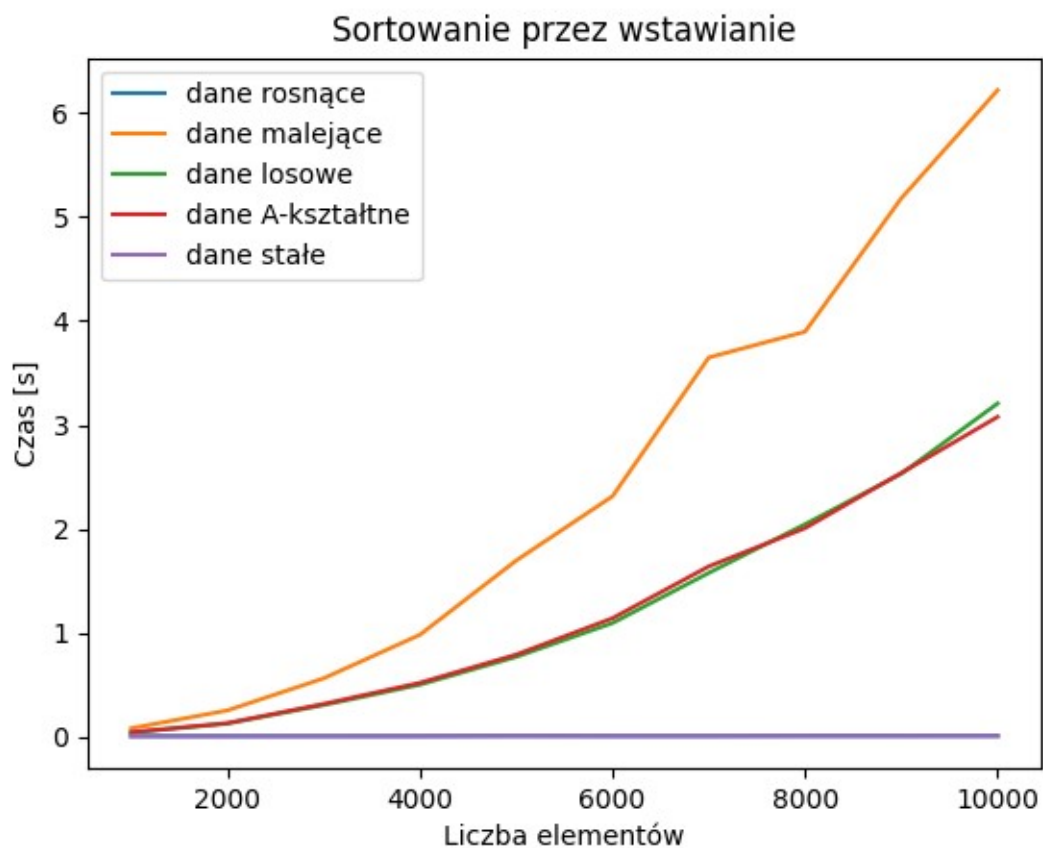


Figura 2: Czas wykonywania *insetion sort*

Dla danych, które są już wstępnie posortowane (dane stałe, dane rosnące) następuje przypadek optymistyczny, w którym algorytm wykonuje się w czasie liniowym. W pozostałych przypadkach czas jest kwadratowy, a algorytm najgorzej radzi sobie z danymi malejącymi.

Sortowanie Shella

Sortowanie Shella jest algorytmem sortującym w miejscu. Jego złożoność to $O(n \log n)$ w przeciętnym przypadku, a w najgorszym – $O(n \log^2 n)$. W tym przypadku początkowy przyrost to połowa długości zestawu danych który po każdym przejściu jest dzielony przez 2.

Algorytm działa najszybciej dla danych rosnących i stałych, niewiele dłużej dla danych A-kształtnych i losowych. Najgorzej algorytm działa dla danych malejących.

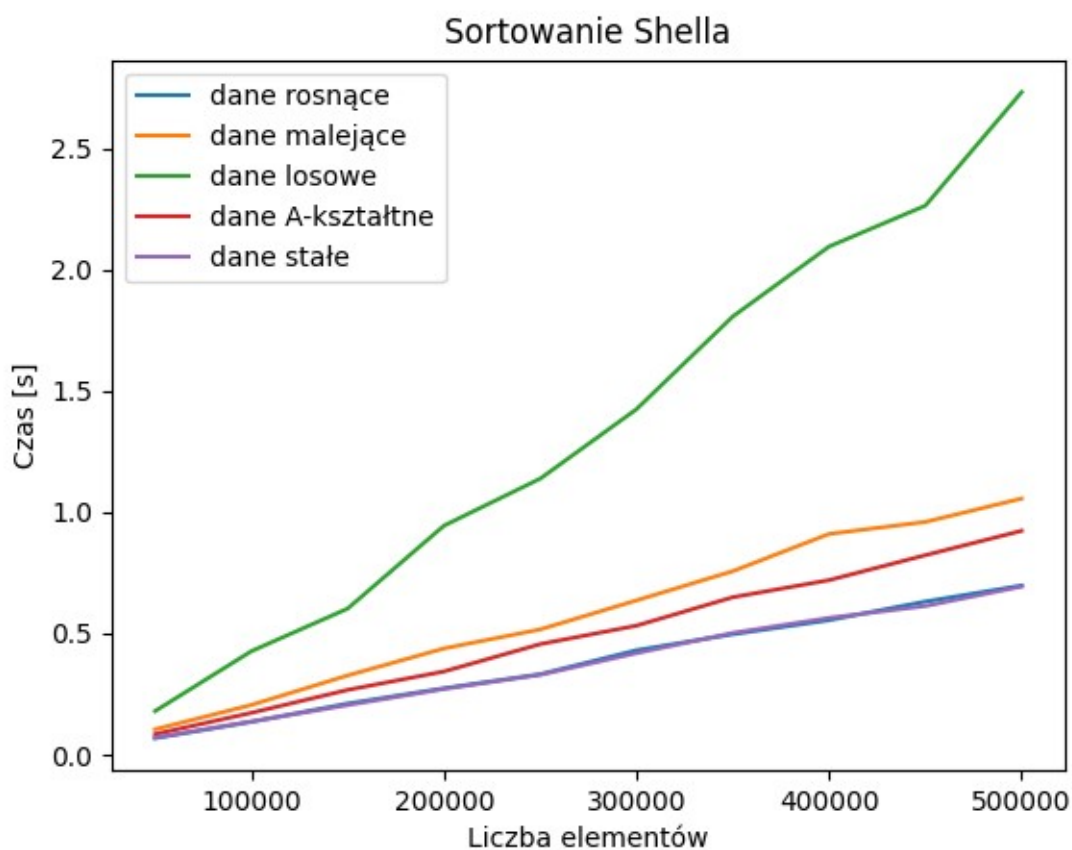


Figura 3: Czas wykonywania Shell sort

Sortowanie przez kopcowanie

Sortowanie przez kopcowanie jest algorytmem sortującym w miejscu, o złożoności $O(n \log n)$. Operacja tworzenia kopca jest wykonywana w czasie liniowym $O(n)$, a działanie na elemencie kopca – $O(\log n)$.

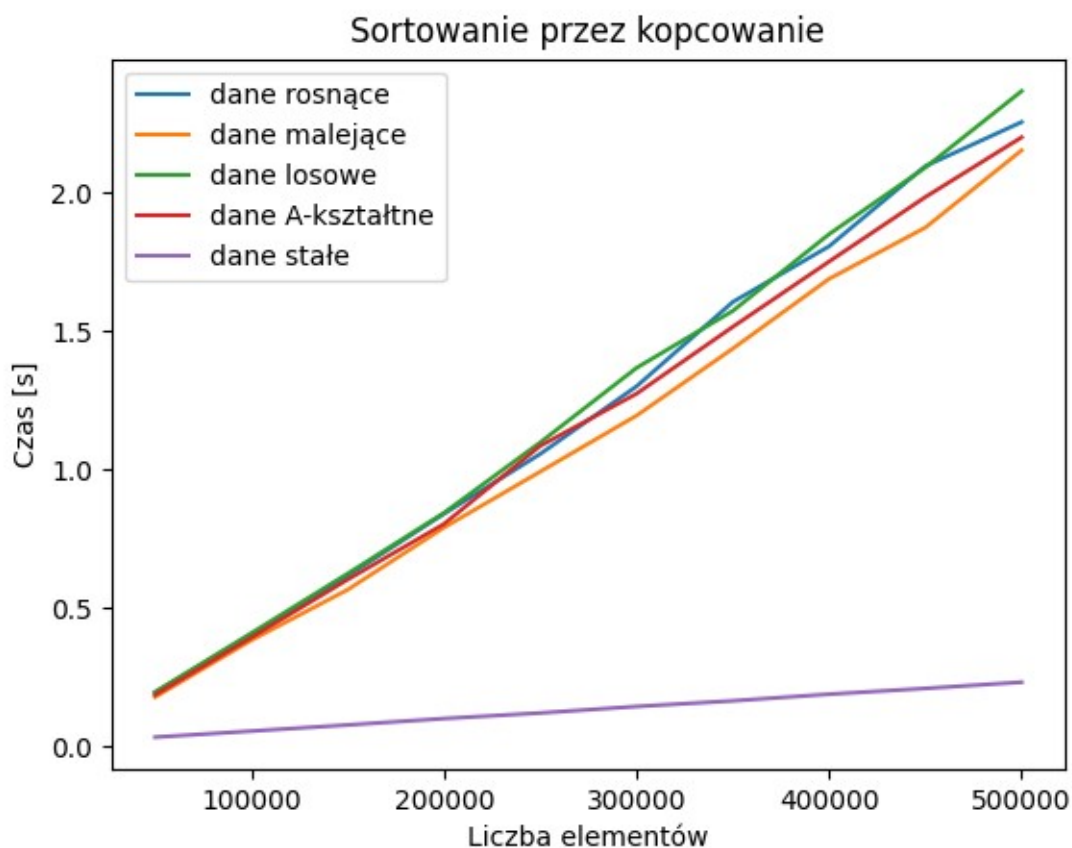


Figura 4: Czas wykonywania heap sort

Dla wszystkich rodzajów danych algorytm sortuje je w takim samym czasie, za wyjątkiem danych stałych – w tym przypadku algorytm wykonuje się znacznie szybciej ($O(n)$).

Sortowanie szybkie

Sortowanie szybkie sortuje w miejscu i ma średnią złożoność $O(n \log n)$. Zależy ona od elementu rozdzielającego zwanego pivotem. W najgorszym przypadku może ona osiągnąć $O(n^2)$.

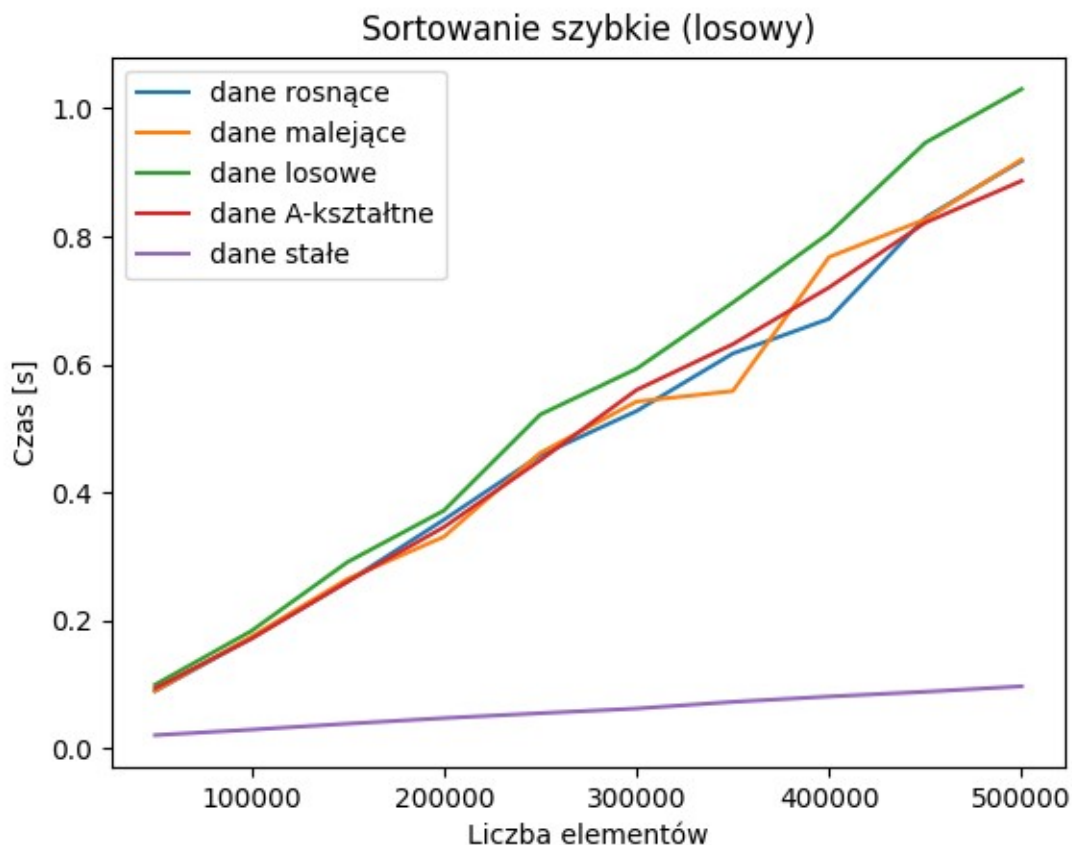


Figura 5: Czas wykonywania quick sort

W przypadku losowego wyboru pivotu algorytm wykonuje się w tym samym czasie dla wszystkich zestawów poza danymi stałymi, dla których wykonuje się znacznie szybciej. W przypadku, gdy pivotem jest pierwszy element w liście sytuacja zmienia się – dla danych losowych algorytm nadal wykonuje się w tym samym czasie, jednak dla pozostałych danych program nie wykonuje się – spowodowane jest to tym, że w tych przypadkach pivotem staje się element najmniejszy bądź największy w danej liście – następuje przepełnienie stosu z powodu zbyt wielu wywołań rekurencyjnych.