Wstęp do Algorytmów

Kierunek: Inżynieria Systemów

Semestr Letni – 2024/2025

Lista 2 – Złożoność Obliczeniowa – Wprowadzenie

Do pomiaru czasu na liście można skorzystać wprost z time.time()), jak na stronie 11 w tych materiałach: `prac.im.pwr.wroc.pl/~szwabin/assets/algo/lectures/1.pdf`, albo z funkcji timeit (`https://www.geeksforgeeks.org/timeit-python-examples/`).
`https://www.c-sharpcorner.com/article/selection-insertion-and-bubble-sort-in-python/` – kody i opisy podstawowych algorytmów sortowania.

1. Przeanalizować podstawowe algorytmy sortowania (sortowanie bąbelkowe, sortowanie przez wstawianie, sortowanie przez wybór).

- Narysować schematy blokowe tych algorytmów i przeanalizować ich złożoność czasową.
- Napisać program obliczający średni i maksymalny (z dziesięciu przebiegów) czas działania poszczególnych algorytmów sortowania dla losowo wygenerowanej listy n liczb.
- Wyświetlić na wykresie średnie i maksymalne czasy działania poszczególnych algorytmów dla długości ciągów: 10, 20, 50, 100, 200, 500, 1000. Zadbać o dobre opisanie wykresu. (w przypadku braku pomysłów na lib do wykresów, można skorzystać np. z `https://plot.ly/python/line-charts/`)
- Wprowadzić dwie modyfikacje do sortowania bąbelkowego z linka:
  - taką, gdzie algorytm jest przerywany, gdy w bieżącym przejściu (przez ciąg) nie nastąpiła żadna zamiana,
  - taką, gdzie w każdym przejściu po ciągu porównania naiwnie realizujemy do końca ciągu (w wersji z linka w każdym kolejnym przejściu robimy o jedno porównanie mniej).
- Zestawić na wykresie czasy działania wszystkich trzech wersji implementacyjnych algorytmu sortowania bąbelkowego.
- Dla 3 algorytmów sortowania: bąbelkowego, sortowania przez wstawianie oraz sortowania przez wybór oraz dla wbudowanej funkcji sort() w języku Python:
  - przyjąć $n = 10, 20, 50, 100, 200, 500, 1000$,
  - wylosować n liczb,
  - wykonać testy wydajnościowe i wykreślić czasy działania poszczególnych algorytmów,
  - zastanowić się, czy dla poszczególnych algorytmów (włączając w to różne wersje sortowania bąbelkowego) potrzebne jest uśrednianie wyników czasowych z wielu przebiegow dla różnych danych początkowych – wprowadzić uśrednienie czasu trwania z wielu przebiegów (dla różnych danych wejściowych) tam, gdzie wydaje się to konieczne.