

## Algorytmy i Struktury Danych

### Zadanie offline 1 (7.III.2022)

#### Format rozwiązań

Rozwiązanie zadania musi się składać z **krótkiego** opisu algorytmu (wraz z uzasadnieniem poprawności) oraz jego implementacji. Zarówno opis algorytmu jak i implementacja powinny się znajdować w tym samym pliku Pythona (rozszerzenie `.py`). Opis powinien być na początku pliku w formie komentarza (w pierwszej linii w komentarzu powinno być imię i nazwisko studenta). Opis nie musi być długi—wystarczy kilka zdań, jasno opisujących ideę algorytmu. Implementacja musi być zgodna z szablonem kodu źródłowego dostarczonym wraz z zadaniem. Niedopuszczalne jest w szczególności:

1. korzystanie z wbudowanych funkcji sortujących,
2. korzystanie z zaawansowanych struktur danych (np. słowników czy zbiorów),
3. zmienianie nazwy funkcji implementującej algorytm, listy jej argumentów, lub nazwy pliku z rozwiązaniem,
4. modyfikowanie testów dostarczonych wraz z szablonem,
5. wypisywanie na ekranie jakichkolwiek napisów innych niż wypisywane przez dostarczony kod (ew. napisy dodane na potrzeby diagnozowania błędów należy usunąć przed wysłaniem zadania).

Dopuszczalne jest natomiast:

1. korzystanie z następujących elementarnych struktur danych: krotka,
2. korzystanie ze struktur danych dostarczonych razem z zadaniem (jeśli takie są).

Wszystkie inne algorytmy lub struktury danych wymagają implementacji przez studenta. Dopuszczalne jest oczywiście implementowanie dodatkowych funkcji pomocniczych w pliku z szablonem rozwiązania.

Zadania niezgodne z powyższymi ograniczeniami otrzymają ocenę 0 punktów. Rozwiązania w innych formatach (np. `.PDF`, `.DOC`, `.PNG`, `.JPG`) z definicji nie będą sprawdzane i otrzymają ocenę 0 punktów, nawet jeśli będą poprawne.

#### Testowanie rozwiązań

Żeby przetestować rozwiązanie zadania należy wykonać polecenie: `python3 zad1.py`

## Zadanie offline 1.

Szablon rozwiązania: zad1.py

Węzły jednokierunkowej listy odsyłaczowej reprezentowane są w postaci:

```
class Node:
    def __init__(self):
        self.val = None # przechowywana liczba rzeczywista
        self.next = None # odsyłacz do następnego elementu
```

Niech  $p$  będzie wskaźnikiem na niepustą listę odsyłaczową zawierającą parami różne liczby rzeczywiste  $a_1, a_2, \dots, a_n$  (lista nie ma wartownika). Mówimy, że lista jest  $k$ -chaotyczna jeśli dla każdego elementu zachodzi, że po posortowaniu listy znalazłby się na pozycji różniącej się od bieżącej o najwyżej  $k$ . Tak więc 0-chaotyczna lista jest posortowana, przykładem 1-chaotycznej listy jest 1, 0, 3, 2, 4, 6, 5, a  $(n - 1)$ -chaotyczna lista długości  $n$  może zawierać liczby w dowolnej kolejności. Proszę zaimplementować funkcję **SortH(p, k)**, która sortuje  $k$ -chaotyczną listę wskazywaną przez  $p$ . Funkcja powinna zwrócić wskazanie na posortowaną listę. Algorytm powinien być jak najszybszy oraz używać jak najmniej pamięci (w sensie asymptotycznym, mierzonym względem długości  $n$  listy oraz parametru  $k$ ). Proszę skomentować jego złożoność czasową dla  $k = \Theta(1)$ ,  $k = \Theta(\log n)$  oraz  $k = \Theta(n)$ .