

## 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. 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, lista,
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: `python kol1b.py`

## Kilka uwag o Pythonie

```
x = "abcdefg"
y = x[::-1]      # y staje się nowym napisem "gfedcba"
z = y            # z i y wskazują na ten sam napis (koszt tego przypisania to O(1))
                # niezależnie od długości napisu y)
ord("a")        # wynikiem jest kod litery 'a' (97)
chr(97)         # wynikiem jest litera o kodzie 97 ("a")
```

<b>Szablon rozwiązania:</b>	<code>kol1b.py</code>
<b>Pierwszy próg złożoności:</b>	$O(N)$ , gdzie $N$ to łączna długość napisów w tablicy wejściowej.
<b>Drugi próg złożoności:</b>	$O(N \log N)$

Dwa słowa są anagramami jeśli składają się z dokładnie tej samej liczby tych samych liter. Na przykład anagramami są słowa “algorytm” i “logarytm”, podczas gdy słowa “katar” i “totem” anagramami nie są. Jeśli dwa słowa są anagramami, to są tej samej długości.

Dana jest tablica  $T$  składająca się z pewnej liczby słów, gdzie każde słowo składa się z małych liter alfabetu łacińskiego. Popularnością anagramową słowa  $T[i]$  nazywamy liczbę takich indeksów  $j$ , że słowo  $T[j]$  jest anagramem słowa  $T[i]$ .

Proszę zaimplementować funkcję  $f(T)$ , która zwraca popularność najpopularniejszego anagramu.

Na przykład dla wejścia:

```
#      0      1      2      3      4      5      6      7  
T = ["tygrys", "kot", "wilk", "trysyg", "wlik", "sygryt", "likw", "tygrys"]
```

wywołanie  $f(T)$  powinno zwrócić liczbę 4. Algorytm powinien być możliwie jak najszybszy. Proszę podać złożoność czasową i pamięciową zaproponowanego algorytmu.