Algorytmy i Struktury Danych
Zadanie offline 2 (14.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 zad2.py`

Zadanie offline 2.

Szablon rozwiązania: `zad2.py`

Dany jest ciąg przedziałów domkniętych $L = [[a_1, b_1], \ldots, [a_n, b_n]]$. Początki i końce przedziałów są liczbami naturalnymi. Poziomem przedziału $c \in L$ nazywamy liczbę przedziałów w L, które w całości zawierają się w c (nie licząc samego c). Proszę zaproponować i zaimplementować algorytm, który zwraca maksimum z poziomów przedziałów znajdujących się w L. Proszę uzasadnić poprawność algorytmu i oszacować jego złożoność obliczeniową.

Algorytm należy zaimplementować jako funkcję postaci:

```
def depth( L ):
  ...
```

która przyjmuje listę przedziałów `L` i zwraca maksimum z poziomów przedziałów w `L`.

Przykład. Dla listy przedziałów:

```
L = [ [1, 6],
      [5, 6],
      [2, 5],
      [8, 9],
      [1, 6]]
```

wynikiem jest liczba 3.