Algorytmy i Struktury Danych
Zadanie offline 3 (21.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, 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: `python3 zad3.py`

Zadanie offline 3.

Szablon rozwiązania: `zad3.py`

Mamy daną N elementową tablicę T liczb rzeczywistych, w której liczby zostały wygenerowane z pewnego rozkładu losowego. Rozkład ten mamy zadany jako k przedziałów $[a_1, b_1], [a_2, b_2], \ldots, [a_k, b_k]$ takich, że i-ty przedział jest wybierany z prawdopodobieństwem c_i, a liczba z przedziału ($x \in [a_i, b_i]$) jest losowana zgodnie z rozkładem jednostajnym. Przedziały mogą na siebie nachodzić. Liczby a_i, b_i są liczbami naturalnymi ze zbioru $\{1, \ldots, N\}$.

Proszę zaimplementować funkcję `SortTab(T, P)` sortująca podaną tablicę i zwracająca posortowaną tablicę jako wynik. Pierwszy argument to tablica do posortowania a drugi to opis przedziałów w postaci:

```
P = [(a_1,b_1,c_1), (a_2,b_2,c_2), ..., (a_k,b_k,c_k)]}.
```

Na przykład dla wejścia:

```
T = [6.1, 1.2, 1.5, 3.5, 4.5, 2.5, 3.9, 7.8]
P = [(1, 5, 0.75) , (4, 8, 0.25)]
```

po wywołaniu `SortTab(T,P)` tablica zwrócona w wyniku powinna mieć postaci:

```
T = [1.2, 1.5, 2.5, 3.5, 3.9, 4.5, 6.1, 7.8]
```

Algorytm powinien być możliwie jak najszybszy. Proszę podać złożoność czasową i pamięciową zaproponowanego algorytmu.