Algorytmy i Struktury Danych
Kolokwium 1 (4.IV.2024)

Format rozwiązań

Wysłać należy tylko jeden plik: `kol1.py`

Plik można wysyłać wielokrotnie, liczy się ostatnia wersja zapisana w systemie.

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, kolejka `collections.deque`, kolejka priorytetowa (`queue.PriorityQueue`),

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. `.ZIP`, `.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 kol1.py`

Szablon rozwiązania:	`kol1.py`
Złożoność akceptowalna (1.0pkt):	$O(n^2)$, gdzie n to liczba elementów w tablicy.
Złożoność wzorcowa (+3.0pkt):	$O(n \log n)$, gdzie n to liczba elementów w tablicy.

Dana jest n-elementowa tablica liczb naturalnych `T`. Dla każdego indeksu $i < n$, rangą elementu na pozycji i określamy liczbę elementów, które w tablicy występują przed elementem i-tym, a ich wartość jest mniejsza od $T[i]$.

Doprecyzowanie: Rozważmy tablicę `[5,3,9,4]`. W tej tablicy dwa pierwsze elementy mają rangę 0 (nie poprzedza ich żaden mniejszy element), 3-ci element ma rangę 2 (przed nim w tablicy znajdują się wartości 5 oraz 3), a ostatni ma rangę 1 (przed nim w tablicy jedynie 3 jest mniejsze).

Proszę zaimplementować funkcję `maxrank(T)`, która dla tablicy `T` o rozmiarze n elementów zwróci maksymalną rangę pośród wszystkich elementów tablicy.

Przykład. Dla wejścia:

```
T = [5,3,9,4]
```

wywołanie `maxrank(T)` powinno zwrócić wartość 2 (odpowiadającą randze elementu na trzeciej pozycji).

Algorytm powinien być możliwie jak najszybszy. Proszę uzasadnić poprawność zaproponowanego algorytmu oraz oszacować jego złożoność czasową i pamięciową.