Algorytmy i Struktury Danych
Egzamin 2: Zadanie A (5.IX.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. zmienianie nazwy funkcji implementującej algorytm, listy jej argumentów, lub nazwy pliku z rozwiązaniem,
2. 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 zaawansowanych struktur danych (np. słowników czy zbiorów),
2. korzystanie z następujących elementarnych struktur danych: krotka, lista, kolejka `collections.deque`, kolejka priorytetowa (`queue.PriorityQueue` lub `heapq`),
3. korzystanie ze struktur danych dostarczonych razem z zadaniem (jeśli takie są),
4. korzystanie z wbudowanych funkcji sortujących (można założyć, że mają złożoność $O(n \log n)$).

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 egz2a.py`

Szablon rozwiązania:	`egz2a.py`
Złożoność akceptowalna (1.5pkt):	$O(n^2)$, gdzie n to liczba transportów.
Złożoność wzorcowa (+2.5pkt):	$O(n \log n)$, gdzie n to liczba transportów.

Do elektrowni ma przyjechać seria długo oczekiwanych tranportów węgla. Każdy transport zawiera pewną liczbę ton węgla, które muszą być składowane w jednym z magazynów o numerach 0, 1, 2, ... (magazynów jest bardzo dużo i na pewno wystarczą na cały węgiel). Każdy magazyn ma pojemność T ton i węgiel z każdego transportu musi być przechowywany razem, w jednym magazynie (ale w danym magazynie może być węgiel z kilku różnych transportów). Dyrekcja elektrowni przyjęła zasadę, że gdy przyjeżdża kolejny transport, to węgiel umieszczany jest w magazynie o najniższym numerze, w którym się mieści (na szczęście żaden transport nie ma więcej niż T ton). Proszę napisać funkcję:

```
def coal( A, T )
```

która przyjmuje na wejściu tablicę $A = [a_0, \ldots, a_{n-1}]$ zawierającą ilości węgla w kolejnych transportach (wyrażoną w tonach, jako liczby naturalne) oraz liczbę naturalną T oznaczającą pojemność każdego z magazynów. Funkcja powinna zwrócić numer magazynu, w którym umieszczono ostatni transport. Funkcja powinna być możliwie jak najszybsza.

Przykład. Rozważmy następujące dane:

```
A = [1, 6, 2, 10, 8, 7, 1]
T = 10
```

Wywołanie `coal(A, T)` powinno zwrócić 0 (pierwsze trzy transporty zostaną umieszczone w magazynie nr 0, kolejny w magazynie nr 1, transport 8 ton zostanie umieszczony w magazynie nr 2, transport 7 ton w magazynie nr 3 i ostatni transport zmieści się jeszcze w magazynie nr 0).