Algorytmy i Struktury Danych
Zadanie offline 6 (16.V.2023)

## 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, kolejka `collections.deque`, kolejka priorytetowa (`queue.PriorityQueue, heapq`),
2. korzystanie ze struktur danych dostarczonych razem z zadaniem (jeśli takie są).
3. 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: `python3 zad6.py`

## Zadanie offline 6.

**Szablon rozwiązania:** `zad6.py`

Firma Binary Works zajmuje się produkcją kubełków do sortowania liczb. W firmie jest $n$ pracowników oraz $n$ maszyn pozwalających na tworzenie kubełków. Niestety praktycznie każde urządzenie pochodzi od innego wytwórcy i wymaga innych umiejętności. Stąd nie wszyscy pracownicy potrafią obsługiwać każde z nich. Zadanie polega na zaimplementowaniu funkcji: Zadanie polega na zaimplementowaniu funkcji:

```
def binworker( M )
```

gdzie $M$ to tablica tablic o następującej interpretacji: Dla każdego $i \in \{0, \ldots, n-1\}$ $M[i]$ to lista numerów maszyn (ze zbioru $\{0, \ldots, n\}$ na których pracownik numer $i$ potrafi pracować. Funkcja powinna zwrócić maksymalną liczbę pracowników, którzy mogą jednocześnie pracować (na danej maszynie może na raz pracować jednocześnie tylko jeden pracownik).

**Przykład.** Rozważmy następujące dane:

```
M = [ [ 0, 1, 3],   # 0
      [ 2, 4],      # 1
      [ 0, 2],      # 2
      [ 3 ],        # 3
      [ 3, 2] ]     # 4
```

Wynikiem wywołania `binworker(M)` powinno być 5. W szczególności:

- pracownik 0 może pracować na maszynie 1,
- pracownik 1 może pracować na maszynie 4,
- pracownik 2 może pracować na maszynie 0,
- pracownik 3 może pracować na maszynie 3,
- pracownik 4 może pracować na maszynie 2.