Algorytmy i Struktury Danych
Egzamin/Zaliczenie 2 (7.IX.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. zmienianie nazwy funkcji implementującej algorytm, listy jej argumentów, lub nazwy pliku z rozwiązaniem,
2. modyfikowanie testów dostarczonych wraz z szablonem,
3. 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, słownik, 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 (należy 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 egz2a.py`

Szablon rozwiązania:	`egz2a.py`
Złożoność akceptowalna (2pkt):	$O(n^2)$
Złożoność lepsza (+1pkt):	$O(n \log n)$
Złożoność wzorcowa (+1pkt):	$O(n)$

Dany jest zbiór $P = \{p_1, \ldots, p_n\}$ punktów na płaszczyźnie. Współrzędne punktów to liczby naturalne ze zbioru $\{1, \ldots, n\}$. Mówimy, że punkt $p_i = (x_i, y_i)$ dominuje punkt $p_j = (x_j, y_j)$ jeśli zachodzi:

$$x_i > x_j \text{ oraz } y_i > y_j.$$

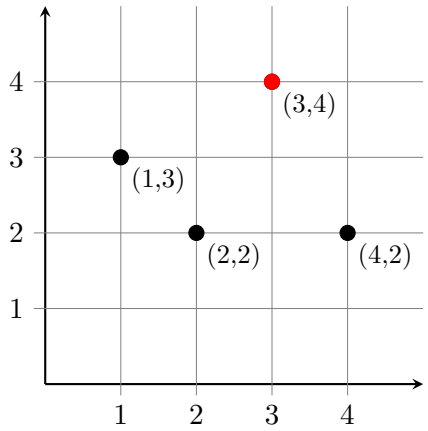
Siłą danego punktu jest to ile punktów dominuje. Zadanie polega na implementacji funkcji:

```
dominance( P )
```

która na wejściu otrzymuje listę `P` zawierającą n punktów (każdy reprezentowany jako para liczb ze zbioru $\{1, \ldots, n\}$) i zwraca siłę najsilniejszego z nich. Funkcja powinna być możliwie jak najszybsza.

Przykład. Dla wejścia:

```
P = [(1,3),
     (3,4),
     (4,2),
     (2,2)]
```



wynikiem jest 2. Punkt o współrzędnych $(3, 4)$ dominuje punkty o współrzędnych $(1, 3)$ oraz $(2, 2)$.

Podpowiedź. W realizacji algorytmu o złożoności $O(n)$ przydatne może być policzenie tablicy T takiej, że $T[i]$ to liczba punktów, których współrzędna y jest większa lub równa i.