Algorytmy i Struktury Danych
Zadanie offline 7 (3.VI.2024)

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 zad7.py`

Zadanie offline 7.

Szablon rozwiązania: `zad7.py`

Indianin Jonasz wędruje po labiryncie szukając śladów dawnej cywilizacji. Labirynt jest kwadratowy i złożony z $n \times n$ komnat. Jonasz rozpoczyna wędrówkę w komnacie o współrzędnych $(0,0)$ znajdującej się na planie w lewym górnym rogu i musi dotrzeć do komnaty o współrzędnych $(n-1, n-1)$ w prawym dolnym rogu. Niektóre komnaty (zaznaczone na planie znakiem #) są niedostępne i nie można do nich się dostać. Jonaszowi wolno poruszać się tylko w trzech kierunkach, opisanych na planie jako Góra, Prawo i Dół oraz nie wolno mu wrócić do komnaty w której już był. Jonasz chce odwiedzić jak największej komnat, żeby potencjalnie znaleźć jak najwięcej interesujących artefaktów. Zadanie polega na zaimplementowaniu funkcji:

```
def maze ( L )
```

która otrzymuje na wejściu tablicę L opisującą labirynt i zwraca największą liczbę komnat, które może odwiedzić Jonasz na swojej drodze lub -1 jeśli dotarcie do końca drogi jest niemożliwe. Komnaty początkowej nie liczymy jako odwiedzonej. Funkcja powinna być możliwie jak najszybsza.

Labirynt opisuje lista $L = [W_0, W_1, W_2, \ldots, W_{n-1}]$, gdzie każde W_i to napis o długości n znaków. Znak kropki '.' oznacza dostępną komnatę a znak '#'oznacza komnatę niedostępną.

Przykład. Rozważmy następujący labirynt:

```
L = [ "....",
      "..#.",
      "..#.",
      "...." ]
```

Optymalna droga wojownika to: DDDPGGGPPDDD, podczas której Wojownik odwiedził 12 komnat.