Wstęp do programowania
Pracownia 12
(przedostatnia)

Uwaga: Wszystkie zadania z tej listy będzie można oddawać do końca semestru. Do maksimum wliczają się 3 punkty.

Premia za tę listę wynosi 0.5, przyznawana jest osobom, które zdobyły co najmniej 2p za zadania z tej listy. Pierwsze 3 zadania odwołują się do rzeczy omówionych podczas Wykładu 13, który będzie opublikowany 12.01.2021

Zadanie 1.(1pkt) W zadaniu zastanowimy się, jak zamienić `wodę` w `wino`. A dokładniej, rozważamy następującą łamigłówkę, w której *ruchem* jest zmiana jednej litery w czteroliterowym słowie polskim (lista dozwolonych słów będzie na stronie) w taki sposób, żeby powstało inne czteroliterowe polskie słowo. Czynności należy powtarzać, aż otrzyma się docelowe słowo. Przykładowy ciąg wyrazów zamieniających `mąka` w `keks`:
`['mąka', 'mika', 'miks', 'kiks', 'keks']`

Napisz funkcję, która dla pary słów znajduje najkrótszą ścieżkę zamieniającą jedno w drugie (zwracaną jako listę słów) lub listę pustą, jeżeli ścieżka nie istnieje. Sprawdź, jaką wartość zwraca ona dla pary `woda` i `wino`

Zadanie 2.(0.5+0.5pkt) W algorytmie DFS kolejność sprawdzania kolejnych węzłów nie jest ustalona. W tym zadaniu powinieneś zmodyfikować program z Wykładu 13 w tej sposób, by algorytm najpierw sprawdzał te wioski, które są bliższe punktowi docelowemu (wg normalnej odległości euklidesowej). Zaprezentuj działanie tak zmodyfikowanego algorytmu rysując jednocześnie ścieżkę znalezioną przez zwykły DFS i DFS zmodyfikowany (0.5).

Druga część zadania stanowi rozwinięcie pierwszej, a celem jest stworzenie ciekawego rysunku, który pokazuje różnicę między zmodyfikowanym DFS a algorytmem BFS (lub innym, znajdującym ścieżkę składającą się z możliwie najmniejszej liczby węzłów). W tym celu należy:

a) Powtórzyć wiele razy losowanie punktów a i b (start i cel)

b) Sprawdzić dla nich wynik działania obu algorytmów

c) Narysować tę parę ścieżek, które dużo się od siebie różnią (jak najmniej procentowo krawędzi jest wspólnych, zob. Współczynnik Jackarda)

d) Wymyślić sposób takiego rysowania krawędzi, żeby było widać, że między niektórymi węzłami przechodzą obie ścieżki

Zadanie 3.(1pkt) Popraw klasę `Set` z wykładu 11, by obsługiwała również takie konstrukcje jak `len(s)`, `s1 & s2`, `s1 - s2` (uwaga: częścia tego zadania jest samodzielne znalezienie, jaki nazwy muszą mieć `__specjalne_metody__` przeciążające operatory.

Zadanie 4.(1pkt) Szyfr przestawieniowy to taki szyfr, w którym każdej literce z polskiego alfabetu przypisana jest inna literka (konsekwentnie, w ramach całego komunikatu). W tym i kolejnym zadaniu, będziemy łamać takie szyfry (czyli pisać programy, które znajdują komunikat, w sytuacji, gdy mamy znamy jedynie szyfrogram). Będziemy zakładać, że słowa w szyfrogramie oddzielone są spacjami i (dla zwiększenia czytelności komunikatu), między nimi czasami znajdują się znaki interpunkcyjne (niezaszyfrowane, otoczone spacjami). Zakładamy również, że wszystkie słowa w komunikacie występują w słowniku (z polskimi słowami z jednej z poprzednich list) i że nie mamy żadnych dodatkowych informacji o języku (np. o częstościach liter, czy wyrazów).

Napisz program, który umie rozszyfrować dwa pierwsze szyfrogramy ze SKOS-u. Uwaga: w obu tych szyfrogramach wszystkie słowa mają unikalną permutacyjną postać normalną (to znaczy, że znajomość tejże postaci pozwala jednoznacznie wybrać słowo). Uwaga2: każdy szyfrogram jest w osobnym wierszu, każdy był też szyfrowany osobną permutacją.

Zadanie 5.(0.5, $\star$pkt) Zmodyfikuj program, by poradził sobie z jeszcze jakimś przykładem, w którym nie wszystkie słowa mają unikalną permutacyjną postać normalną (na przykład trzeci szyfrogram ma tę własność, będąc zarazem bardzo łatwym do odszyfrowania)

Zadanie 6.(1, ⋆pkt) Zmodyfikuj program, by poradził sobie z wszystkimi przykładami ze SKOS-u. Uwaga: to już trochę trudniejsze zadanie. W rozwiązaniu można wykorzystać fakt, że nawet nieunikalna normalna postać permutacyjna daje pewne wskazówki o możliwych przyporządkowaniach liter. Szyfrogram 'óśóś' mówi na przykład, że literze 'ó' nie można przypisać litery 'ą' (bo żadne słowo nie zaczyna się na 'ą'). Można też z niego wydedukować inne rzeczy, patrząc na wszystkie słowa o postaci permutacyjnej 1-2-1-2.

Szyfrogramy ze SKOS-u da się odszyfrować w czasie kilku sekund każdy (a niektóre w ułamki sekundy).