# Odległości

27 marca 2025

## Słowo wstępne

**Lista 1.** pozwoliła na zapoznanie się z algorytmem *Odległości Edycyjnej* tj.: dla dwóch napisów $s_1$ i $s_2$, określ minimalny koszt, który pozwoli na przekształcenie $s_1$ w $s_2$. Dopuściliśmy trzy operacje:

- Dodanie znaku
- Usunięcie znaku
- Zamiana znaku

Następnie rozważyliśmy kolejną operację:

- Transpozycja dwóch sąsiednich znaków

Można dostrzec przynajmniej dwa mankamanety naiwenego działania algorytmu:

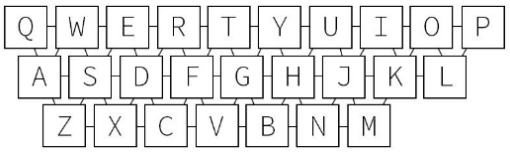
- Wszystkie możliwości były traktowane na równi
- Czas przeszukiwania przestrzeni był niezadowalający

W poniższych zadaniach, dla uproszczenia, rozważamy tylko język angielski (słownik jest podany w materiałach do zadania).

## Zadanie 1. Ciężar Wagi Edycyjnej (2 pkt)

Wagi w obu powyższych wagi przypadkach były określane arbitralnie. Rozważmy następującą propozycję wag, opartą na układzie klawiatury *QWERTY*.

Odległość w tym układzie pomiędzy poszczególnymi klawiszami wynosi od 1 do 9, żeby uzyskać wartości od 0 do 2 każdą z tych wartości mnożymy przez $\frac{2}{9}$. Przy tak ujętej odległości możemy zdefiniować konszty następująco:




Rysunek 1: Układ klawiatury QWERTY wraz z zaznaczonym sąsiedztwem

- Dodanie znaku - bez zmian, 1.
- Usunięcie znaku:
  - dla napisu długości *1* lub *2* koszt usunięcia znaku wynosi 1.0
  - koszt usunięcia poszczególnego znaku dla napisów długości $\geq 2$:
    - dla pierwszego i ostatniego znaku napisu jest równy odległości pomiędzy znakiem do usunięcia a sąsiednim znakiem
    - dla pozostałych jest równy średniej z odgległości (na klawiaturze) sąsiednich znaków
- Zamiana znaku - koszt zamiany znaków jest równy odległości (na klawiaturze) pomiędzy wstawianym znakiem a zastępowanym znakiem.

Zaimplementuj powyższe rozwiązanie i porównaj wyniki otrzymane z implementacją z **Listy 1.** (bez transpozycji).

# Testy

Zakładamy, że podawane słowa na wejściu są nad alfabetem angielskim, nie zawierają cyfy ani znaków specjalnych (w tym spacji) oraz są to napisy małymi literami.
Dla pliku z rozwiązaniem **z1.py** dostarczone są testy **t1.py**. Każdy test składa się słów *str1*, *str2*, oczekiwanego wyniku oraz nazwy testu. W pliku **z1.py** należy zdefiniować funkcję **solution** przyjmująca na wejściu dwa napisy i zwracającą liczbę rzeczywistą jako odpowiedź:

```
1    def solution(str1: str, str2: str) -> float:
2        ...
```

# Dodatkowe testy (1 pkt)

Napisz swoje własne testy trzymając się konwencji pliku **t1.py**.

# Przykład

Transformacja „algorytm” na „aforyzm”: Całkowity koszt: 2,4444 Operacje: 8 łącznie (0 wstawek, 1 usunięcie, 2 podstawienia, 5 dopasowań)

- Dopasowanie „a” (koszt: 0,0000)
- Usunięcie „l” (koszt: 1,3333)
- Zamiana „g” na „f” (koszt: 0,2222)
- Dopasowanie „o” (koszt: 0,0000)
- Dopasowanie „r” (koszt: 0,0000)
- Dopasowanie „y” (koszt: 0,0000)
- Zamiana „t” na „z” (koszt: 0,8889)
- Dopasowanie „m” (koszt: 0,0000)