Szablon rozwiązania	`egzP1a.py`
Złożoność akceptowalna (1.5pkt):	O(*nm*)
Złożoność wzorcowa (+2.5pkt):	O(*n*log*m*), gdzie *n* to łączna ilość znaków w zapisie Morse'a wiadomości, a *m* to ilość dostępnych liter (m < n)

W nocy z 14 na 15 kwietnia 1912 roku swój dziewiczy rejs odbywał statek RMS Titanic. Historia jego jest niestety znana, jednak badacze w późniejszych latach zastanawiali się, czy katastrofy można było uniknąć. Operatorzy telegrafów w tamtych czasach posługiwali się alfabetem Morse'a. Niestety sprzęt na pokładzie statku nie funkcjonował poprawnie, przez co po każdej przesłanej literze trzeba było zrobić kilkusekundową przerwę, co gorsza, część liter nie była wcale rozpoznawana przez telegraf i wywoływała błąd. Po zderzeniu z górą lodową minuty dzieliły statek od utraty elektryczności, więc wiadomość z prośbą o pomoc trzeba było wysłać jak najszybciej. Jeden z operatorów wpadł wtedy na pomysł, aby zignorować odstępy między literami i zakodować wiadomość korzystając z tych, które były rozpoznawalne, przy okazji minimalizując ilość zużytych liter – zakładamy, że w porcie pracują najlepsi kryptografowie i będą w stanie w mgnieniu oka rozszyfrować każdą nadaną w ten sposób wiadomość.

W ramach zadania należy zaimplementować funkcję:

```
def titanic(W, M, D)
```

która oblicza minimalną ilość liter, która musi zostać użyta do przesłania w ten sposób wiadomości **W** przy następujących założeniach:

1. **M** jest tablicą zawierającą wszystkie oznaczenia liter w alfabecie Morse'a w postaci krotek składających się z litery i odpowiadającej jej znaków, tj. M = [('A', '.-'), ('B', '-...'), ('C', '-.-.'), ...] itd. Jedna litera składa się z maksymalnie 4 znaków. Występują dwa typy znaków: '**.**' oraz '-'

2. **W** jest stringiem wyrażającym wiadomość w języku naturalnym w wielkich literach alfabetu angielskiego (Od A do Z).

3. **D** jest tablicą indeksów rozpoznawalnych przez telegraf liter w tablicy M. Można założyć, że rozwiązanie zawsze istnieje, tj. **D** zawsze zawiera litery E (.) oraz T (-)

Rozważmy następujące dane:

```
W = 'SOS'
#Czyli w zapisie Morse'a: . . . - - - . . .

D = [0, 4, 13, 19, 25]
#Czyli litery: A (.-), E (.), N (-.), T (-) oraz Z (--..)
```

Wywołanie `titanic(W, M, D)` powinno zwrócić wynik 5, wysyłamy wiadomość np. jako **EEAZE** (Może być kilka różnych rozwiązań, zwrócić wystarczy jedynie minimalną ilość liter)

```
. | . | . - | - - . . | .

E | E |  A  |    Z    | E
```

Uwaga. W pliku w funkcji testującej pojawiło **recursion=False**. Jeżeli Twoja implementacja powoduje błąd „recursion depth" – zmień na True, aby wywołać dostosowane mniejsze testy.

Szablon rozwiązania	`egzP1b.py`
Złożoność akceptowalna (1.5pkt):	O(n^2)
Złożoność wzorcowa (+2.5pkt):	O($m \log n$), gdzie n wyraża łączną liczbę punktów, a m liczbę połączeń komunikacyjnych między nimi

Biuro podróży WRSS WIEiT zaplanowało wycieczkę do Warszawy dla studentów informatyki. Mapa turystyczna stolicy ma postać grafu nieskierowanego G = (V, E), gdzie wierzchołki oznaczają punkty turystyczne warte odwiedzenia, dworzec oraz lotnisko, a krawędzie połączenia komunikacyjne między tymi miejscami. Każde z połączeń ma pewien stały czas, wyrażony w minutach, który jest wymagany na podróż. W ofercie wycieczki przygotowanej przez samorząd jest napisane, że w trakcie wycieczki odwiedzą dokładnie 3 punkty turystyczne. Ponadto ze względu na oszczędności, do Warszawy dostaną się koleją, a wrócą samolotem, aby zdążyć na sesję egzaminacyjną. Aby maksymalnie wykorzystać wycieczkę, biuro chce zminimalizować czas spędzony w podróży między punktami (dodatkowo nie ma znaczenia, które 3 punkty odwiedzimy, ponieważ na mapie turystycznej zaznaczone są tylko te warte odwiedzenia).

W ramach zadania należy zaimplementować funkcję:

```
def turysta(G, D, L)
```

która oblicza minimalny czas, który zostanie poświęcony na transport, zakładając, że:

1. **G** zawiera graf wyrażony jako lista krawędzi, czyli dla każdego połączenia między punktami *u* oraz *v* (*u* < *v*) o czasie przejazdu *p*, **G** zawiera krotkę *(u, v, p)*

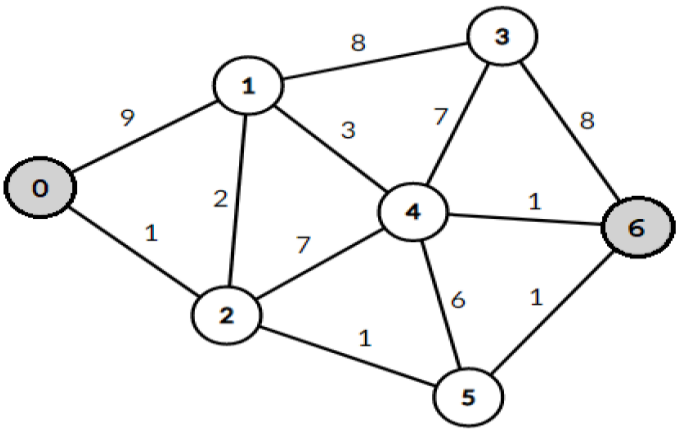
2. Dworzec na którym wysiedli studenci jest oznaczony jako **D**, a lotnisko jako **L**

3. Studenci oprócz dworca i lotniska mają odwiedzić DOKŁADNIE 3 inne punkty turystyczne. Dworzec i lotnisko mogą zostać „odwiedzone" tylko raz. Dworzec tuż po przyjeździe na wycieczkę, a lotnisko tuż przed odlotem.

Rozważmy następujące dane:

```
G = [
(0, 1, 9), (0, 2, 1),
(1, 2, 2), (1, 3, 8),
(1, 4, 3), (2, 4, 7),
(2, 5, 1), (3, 4, 7),
(4, 5, 6), (3, 6, 8),
(4, 6, 1), (5, 6, 1)
]

D = 0
L = 6
```



Wywołanie `turysta(G, D, L)` powinno zwrócić wynik 7 (Zaczynamy na dworcu w punkcie 0, odwiedzamy kolejno punkty 2, 1, 4 i dochodzimy do lotniska w punkcie 6). Proszę zauważyć, że w teorii ścieżka 0-2-5-6 jest najkrótsza, jednak zawiera tylko dwa punkty turystyczne.