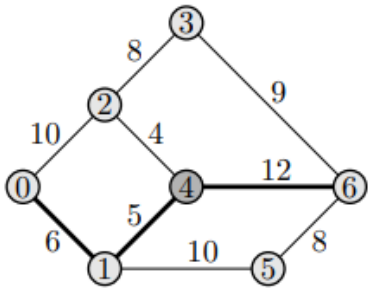
Magiczny wojownik chce się przedostać przez góry Bajtocji. Wyrusza z miasteczka s i chce się dostać do miasteczka t. Wojownik dysponuje mapą z zaznaczonymi schroniskami, miasteczkami *s* i *t*, oraz łączącymi je szlakami (mapa ma formę grafu gdzie miasteczka i schroniska to wierzchołki a szlaki to krawędzie). Każdy szlak ma przypisaną liczbę godzin potrzebnych, żeby go przebyć (są to liczby naturalne z zakresu od 1 do 16, zapisane jako wagi krawędzi grafu). Kodeks honorowy magicznych wojowników mówi, że wojownik nie może być w drodze bez odpoczynku dłużej niż 16 godzin. Taki odpoczynek musi trwać 8 godzin i musi się odbyć w schronisku. Wojownik chce się dostać z *s* do *t* jak najszybciej, ale nie może łamać zasad kodeksu. Gdy wojownik rusza z *s* jest w pełni wypoczęty, ale nie musi być wypoczęty gdy dotrze do *t*.

Proszę zaimplementować funkcję **warrior(*G, s, t*)**, która zwraca ile godzin trwa najszybsza droga wojownika z *s* do *t*, przy użyciu mapy opisanej jako graf *G*. Graf *G* reprezentowany jest jako lista krawędzi. Każda krawędź to trójka postaci (*u*, *v*, *w*), gdzie *u* i *v* to numery wierzchołków a *w* to liczba godzin potrzebna na przebycie drogi z *u* do *v* (oraz z *v* do *u*; graf jest nieskierowany). Numery wierzchołków to kolejne liczby naturalne od *0* do *n − 1*, gdzie *n* to liczba wierzchołków. Algorytm powinien być możliwie jak najszybszy. Proszę uzasadnić poprawność zaproponowanego algorytmu oraz oszacować jego złożoność czasową i pamięciową.

Przykład 1. Dla wejścia:
G = [(1,5,10), (4,6,12), (3,2,8),
(2,4,4) , (2,0,10), (1,4,5),
(1,0,6) , (5,6,8) , (6,3,9)]
s = 0
t = 6



wywołanie **warrior(G,s,t)** powinno zwrócić wartość 31, odpowiadającą trasie
0 ⟶ 1 ⟶ 4(nocleg) ⟶ 6, której przebycie trwa 6 + 5 + 8 + 12 = 31 godzin.