Szablon rozwiązania	`egzP8a.py`
Złożoność akceptowalna (1.5 pkt):	$O(n^2)$
Złożoność wzorcowa (+2.5 pkt):	O(nlogn), gdzie **n** to liczba firm biorących udział w przetargu

Telewizja Polska część swoich zysków uzyskuje ze sprzedaży czasu reklamowego. Zgodnie z nowym prawem każdy okres rozliczeniowy będzie obejmował pewną ilość dni, każdorazowo ustaloną rozporządzeniem. W poprzednich latach dochody te były coraz mniejsze, dlatego w tym roku prezes spółki postanowił zmienić strategię przetargową tak, aby je zmaksymalizować. Wiadome jest, że największe zyski przynosi pierwszy blok reklamowy wyświetlany po wieczornym wydaniu wiadomości. Prezes zadecydował, że w każdym okresie rozliczeniowym maksymalnie dwie różne firmy będą mogły zdecydować się wyświetlać reklamy w ramach tego bloku. Dodatkowo, każda z nich musi wskazać jeden określony zakres dni, w którym będą one transmitowane codziennie. W ramach przetargu zgłosiła się pewna ilość firm, każda z nich wskazując wybrany przez siebie zakres dni, w których chciałyby wyświetlać swoją reklamę oraz oferując pewne wynagrodzenie za cały ten czas. Wybrane zostały oczywiście te, które dały łącznie największy zysk. Prezes spółki chciałby dowiedzieć się, jaki zysk z reklam pierwszego bloku osiągnie w najbliższym okresie rozliczeniowym.

Zadanie polega na zaimplementowaniu funkcji:

```
reklamy( T, S, o )
```

która zwróci zysk, który Telewizja Polska osiągnie z tych reklam, przy następujących założeniach:

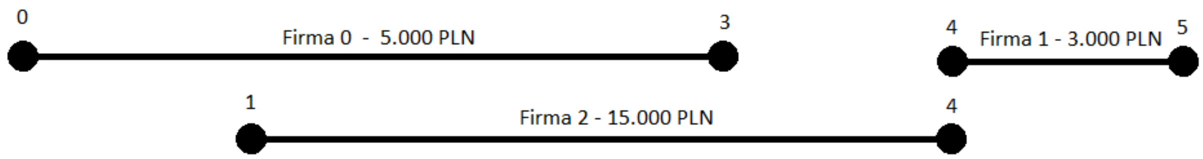
1. Tablica **T** zawiera okresy, w których firmy byłyby chętne wyświetlać swoje reklamy. Tablica ma postać $[(p_0, k_0), (p_1, k_1), ..., (p_{n-1}, k_{n-1})]$ gdzie p_i oznacza początek zakresu dni w danym okresie rozliczeniowym, a k_i koniec tego zakresu (włącznie) dla firmy i-tej.

2. Tablica **S** określa stawki, które zostały zaoferowane przez firmy za cały okres wyświetlania reklam i ma postać $[s_0, s_1, ..., s_{n-1}]$ gdzie s_i to stawka firmy i-tej za okres z tablicy **T**

3. Zmienna **o** to określona rozporządzeniem ilość dni, którą będzie trwał okres rozliczeniowy.

Rozważmy następujące dane:

```
T = [ (0, 3), (4, 5), (1, 4) ]
S = [ 5000, 3000, 15000 ]
```



Wywołanie funkcji `reklamy( T, S, 6 )` powinno zwrócić wynik **15000**. Najbardziej opłacalna opcja to wybór jedynie firmy 2, a w dniu zerowym oraz piątym nie wyświetlanie bloku reklamowego tuż po wieczornym wydaniu wiadomości.

Szablon rozwiązania	`egzP8b.py`
Złożoność akceptowalna (1.5pkt):	$O(n^3 \log n)$
Złożoność wzorcowa (+2.5pkt):	$O(n^3)$, gdzie **n** to liczba punktów na hali

Pewna duża amerykańska firma technologiczna postanowiła zorganizować wyścigi autonomicznych robotów. Wyścigi odbyły się na potężnej prostokątnej hali, która dla tego celu została w pełni opróżniona. Na podłodze zostały umieszczone naklejki w dwóch kolorach. Białe oznaczają punkty zmiany kierunku, a szare to punkty kontrolne (które oczywiście jednocześnie pełnią również role punktów białych). Roboty wyposażone są w kamery oraz dobre oświetlenie, co pozwala im rozpoznać, że znajdują się na punkcie danego koloru. Ze względu na zastosowane technologie, autonomiczne roboty po najechaniu na punkt, wybierają kolejny punkt swojej trasy, a następnie zmieniają swój kierunek oraz decydują o odległości, którą przebędą (tj. jeżeli określą następny punkt, to wszystkie punkty na trasie do niego zostaną zignorowane). Między dwoma punktami zawsze poruszają się w linii prostej. W trakcie trasy muszą odwiedzić one wszystkie punkty kontrolne w określonej przez organizatora kolejności, przebywając łącznie jak najkrótszy dystans. Rozwiązanie zadania byłoby oczywiste, gdyby nie to, że na hali zostały rozmieszczone przeszkody. Do pamięci robota została wgrana przygotowana mapa w postaci grafu ważonego i musi zdecydować on o kolejności odwiedzenia punktów.

Zadanie polega na zaimplementowaniu funkcji:

```
def robot( G, P )
```

która oblicza dystans, który pokona robot, przy następujących założeniach.

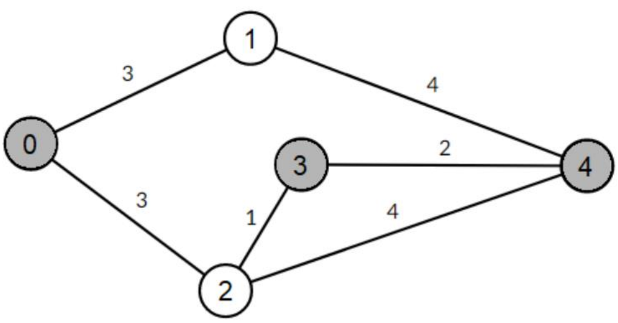
1. Graf ważony **G** jest wyrażony jako lista sąsiedztwa, dla każdych dwóch punktów **u** oraz **v**, pomiędzy którymi nie znajduje się żadna przeszkoda, oddalonych od siebie o **x cm**, lista **G[u]** będzie zawierała krotkę ***(v, x)*** oraz lista G[v] będzie zawierała krotkę ***(u, x)***

2. Tablica **P** zawiera listę szarych punktów, ich numeracja odpowiada wierzchołkom z grafu **G**, w kolejności, w której muszą zostać odwiedzone. Robot zaczyna swoją trasę w pierwszym punkcie, a kończy w ostatnim. Wierzchołki w tablicy **P** są unikalne.

3. Można założyć, że przeszkody zostały umieszczone w taki sposób, że dana przeszkoda wyklucza poruszanie się tylko między jedną parą punktów na hali, a także, że ich ilość jest marginalna względem ilości wszystkich możliwych par punktów.

Rozważmy następujące dane:

```
G = [
  [(1, 3), (2, 3)],
  [(0, 3), (4, 4)],
  [(0, 3), (3, 1), (4, 4)],
  [(2, 1), (4, 2)],
  [(1, 4), (2, 4), (3, 2)]
]
P = [0, 3, 4]
```



Wywołanie `robot( G, P )` powinno zwrócić wynik **6** (Odwiedzamy punkty 0 – 2 – 3 – 4)