Szablon rozwiązania:	`egz1b.py`
Złożoność akceptowalna (1.5pkt):	$O(n^2)$, gdzie n to rozmiar drzewa.
Złożoność wzorcowa (+2.5pkt):	$O(n)$, gdzie n to rozmiar drzewa.

Dane jest drzewo binarne opisane przez następujące klasy:

```
class Node:
  def __init__( self ):
    self.left = None     # lewe poddrzewo
    self.right = None    # prawe poddrzewo
    self.x = None        # pole do wykorzystania przez studentów
```

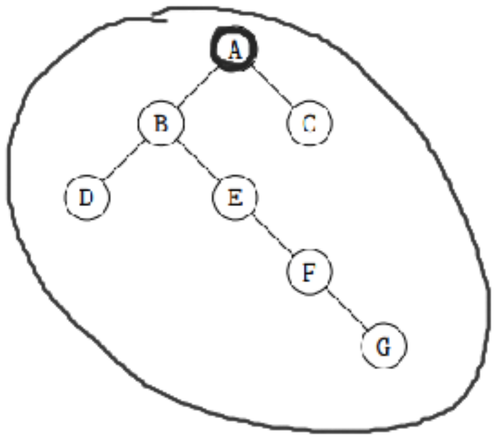
Mówimy, że takie drzewo jest *ładne* jeśli wszystkie jego liście znajdują się na jednym poziomie. Szerokością ładnego drzewa jest jego liczba liści a wysokością poziom, na którym te liście się znajdują (korzeń jest na poziomie 0, jego dzieci na poziomie 1, jego wnuki na poziomie 2 itd.). Zadanie polega na zaimplementowaniu funkcji:

```
def widentall( T )
```

która dla danego drzewa T zwraca minimalną liczbę krawędzi, które trzeba usunąć, żeby powstało ładne drzewo, którego szerokość jest jak największa i którego wysokość jest największa wśród drzew o maksymalnej szerokości. Usunięcie krawędzi odcina całe poddrzewo poniżej tej krawędzi.

Rozważmy następujące dane wejściowe:

```
A = Node()
B = Node()
C = Node()
A.left = B
A.right = C
D = Node()
E = Node()
B.left = D
B.right = E
F = Node()
E.right = F
G = Node()
F.right = G
```



Wywołanie `widentall(A)` powinno zwrócić wynik 2 (ucinamy krawędzie między `A` i `C` oraz między `E` i `F`. Ucięcie krawędzi między `B` i `D` oraz między `B` i `E` doprowadziłoby do ładnego drzewa o tej samej szerokości, ale mniejszej wysokości.

[2pkt.] Zadanie 3.

Szablon rozwiązania: `zad3.py`

Dany jest zbiór przedziałów $A = \{(a_0, b_0), \ldots, (a_{n-1}, b_{n-1})\}$. Proszę zaimplementować funkcję:

```
def kintersect( A, k ):
    ...
```

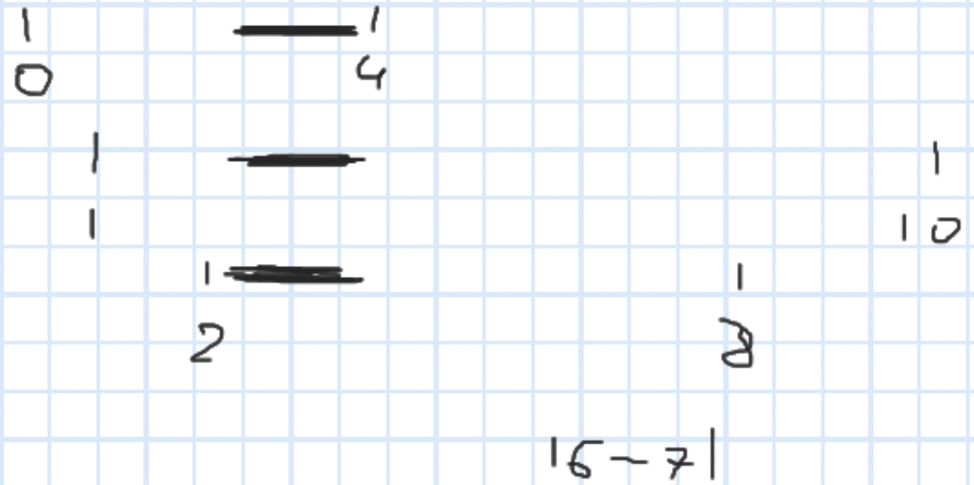
która wyznacza k przedziałów, których przecięcie jest jak najdłuższym przedziałem. Zbiór A jest reprezentowany jako lista par. Końce przedziałów to liczby całkowite. Można założyć, że $k \geq 1$ oraz k jest mniejsze lub równe łącznej liczbie przedziałów w A. Funkcja powinna zwracać listę numerów przedziałów, które należą do rozwiązania.

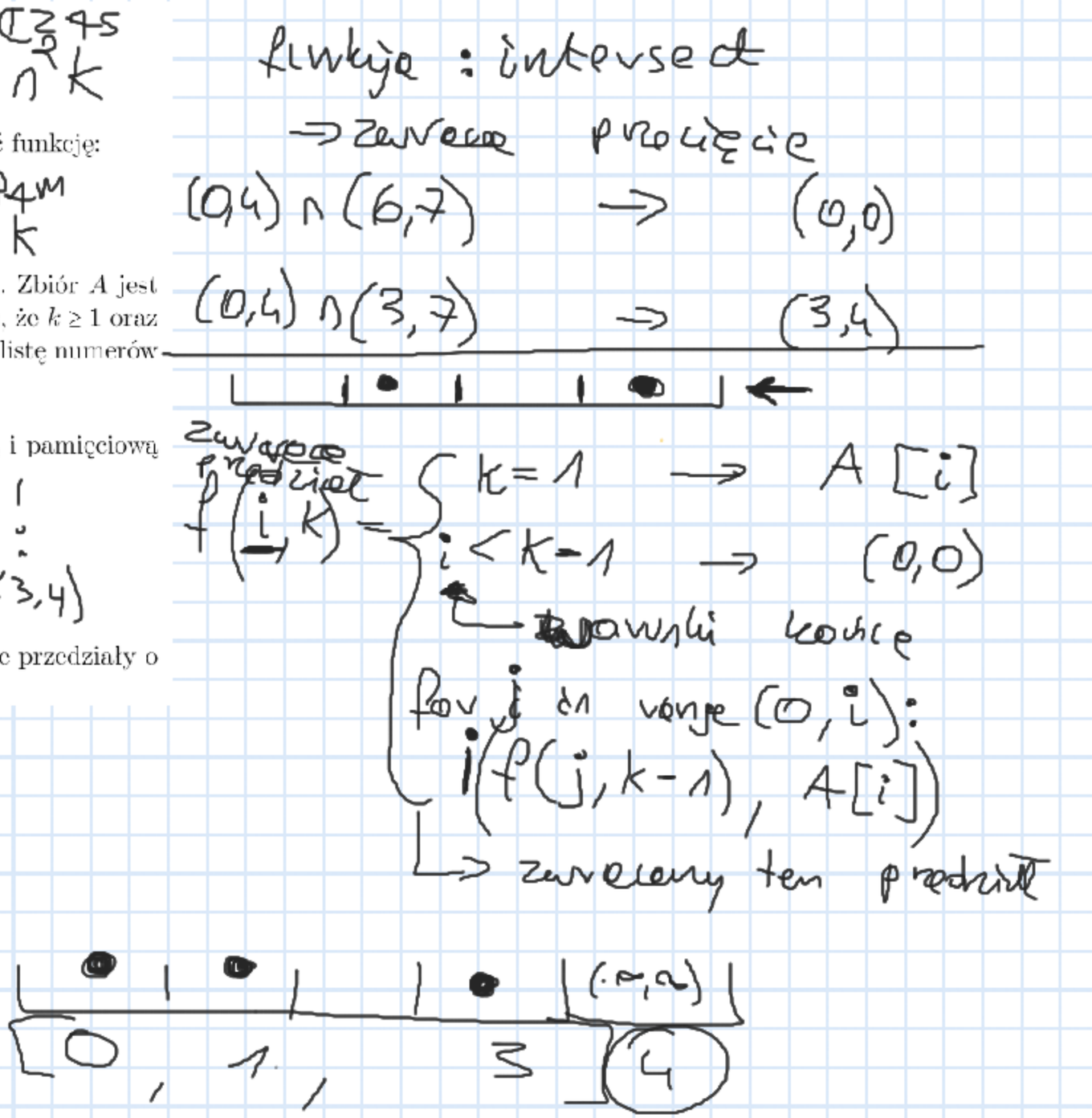
Funkcja powinna być możliwie jak najszybsza. Proszę oszacować złożoność czasową i pamięciową użytego algorytmu.

Przykład: Rozważmy listę przedziałóoe:

```
A = [(0,4),(1,10),(6,7),(2,8)]
```

Dla $k = 3$ wynikiem powinno być `[0,1,3]` (lub dowolna permutacja tej listy), co daje przedziały o przecięciu $[2, 4]$, o długości $4 - 2 = 2$.

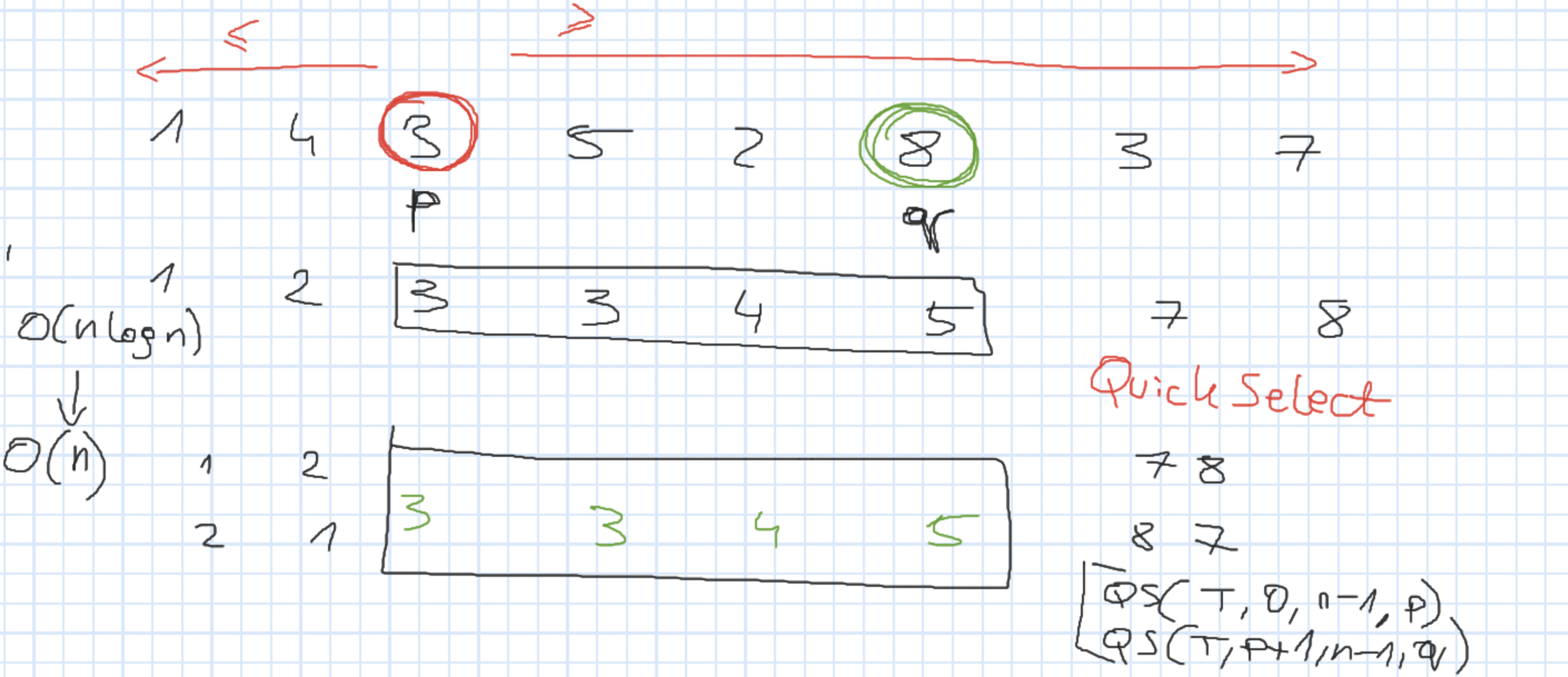




[2pkt.] Zadanie 2. Mamy n żołnierzy różnego wzrostu i nieuporządkowaną tablicę, w której podano wzrosty żołnierzy. Żołnierze zostaną ustawieni na placu w szeregu malejąco względem wzrostu. Proszę zaimplementować funkcję:

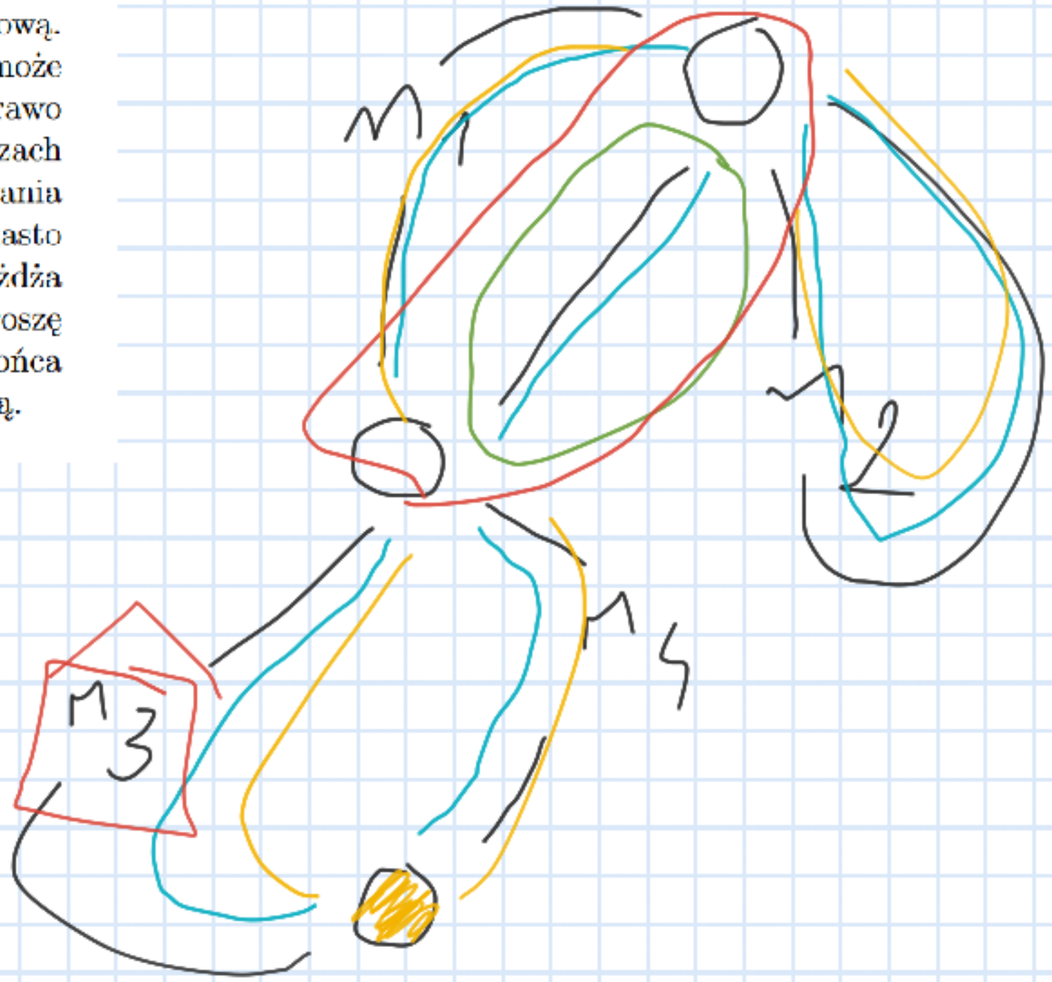
```
section(T,p,q)
```

która zwróci tablicę ze wzrostami żołnierzy na pozycjach od p do q włącznie. Użyty algorytm powinien być możliwie jak najszybszy. Proszę w rozwiązaniu umieścić 1-2 zdaniowy opis algorytmu oraz proszę oszacować jego złożoność czasową.



[2pkt.] Zadanie 2. Algocja leży na wielkiej pustyni i składa się z miast oraz oaz połączonych drogami. Każde miasto jest otoczone murem i ma tylko dwie bramy—północną i południową. Z każdej bramy prowadzi dokładnie jedna droga do jednej oazy (ale do danej oazy może dochodzić dowolnie wiele dróg; oazy mogą też być połączone drogami między sobą). Prawo Algocji wymaga, że jeśli ktoś wjechał do miasta jedną bramą, to musi go opuścić drugą. Szach Algocji postanowił wysłać gońca, który w każdym mieście kraju odczyta zakaz formułowania zadań "o szachownicy" (obraza majestatu). Szach chce, żeby goniec odwiedził każde miasto dokładnie raz (ale nie ma ograniczeń na to ile razy odwiedzi każdą z oaz). Goniec wyjeżdża ze stolicji Algocji, miasta x, i po odwiedzeniu wszystkich miast ma do niej wrócić. Proszę przedstawić (bez implementacji) algorytm, który stwierdza czy odpowiednia trasa gońca istnicje. Proszę uzasadnić poprawność algorytmu oraz oszacować jego złożoność czasową.

1) Miasta
2) Super_oazy
3) DFS



M4
M2
M1
M3

M4 M2 M1 M3

2. Proszę zaimplementować funkcję:

```
int SumBetween(int T[], int from, int to, int n);
```

Zadaniem tej funkcji jest obliczyć sumę liczb z n elementowej tablicy `T`, które w posortowanej tablicy znajdywałyby się na pozycjach o indeksach od `from` do `to` (włącznie). Można przyjąć, że liczby w tablicy `T` są parami różne (ale nie można przyjmować żadnego innego rozkładu danych). Zaimplementowana funkcja powinna być możliwie jak najszybsza. Proszę oszacować jej złożoność czasową (oraz bardzo krótko uzasadnić to oszacowanie).

2. Złodziej widzi na wystawie po kolei n przedmiotów o wartościach $A[0], A[1], ..., A[n-1]$. Złodziej chce wybrać przedmioty o jak największej wartości, ale resztki przyzwoitości zabraniają mu ukraść dwa przedmioty leżące obok siebie. Proszę zaimplementować funkcję:

```
int goodThief( int A[], int n );
```

która zwraca maksymalną wartość przedmiotów, które złodziej może ukraść zgodnie ze swoim kodeksem moralnym oraz wypisuje numery przemiotów które powinien wybrać. Proszę uzasadnić poprawność algorytmu oraz oszacować jego złożoność czasową. Algorytm powinien być możliwie jak najszybszy (ale przede wszystkim poprawny).

max(T[i-1], T[i] + T[i-2]) * T[i-1] == T[i-2]
= 0

	1	3	-1	0	2	-1	5	1
V	1	3	3	3	5	5	10	10
P	-1	-1	1	1	1	1	4	4