

### zadanie 3

#### Intuicja:

- wybieramy jakiś  $v \in V$  i, dopóki możemy, przenosimy z niego DFS-em.
- gdy (w DFS) skończymy przeniesieniem wierzchołek, wstawiamy go na początek listy TopOrd.
- gdy skończymy przenosieniem a w grafie zostały jeszcze nieprzeniesione wierzchołki  
Wpp. zwracamy TopOrd

#### Algorytm:

```

TopOrd ← empty-list()
n ← liczba-wierzch. tab. długości n
V = [false...false] // potrzebna do DFS

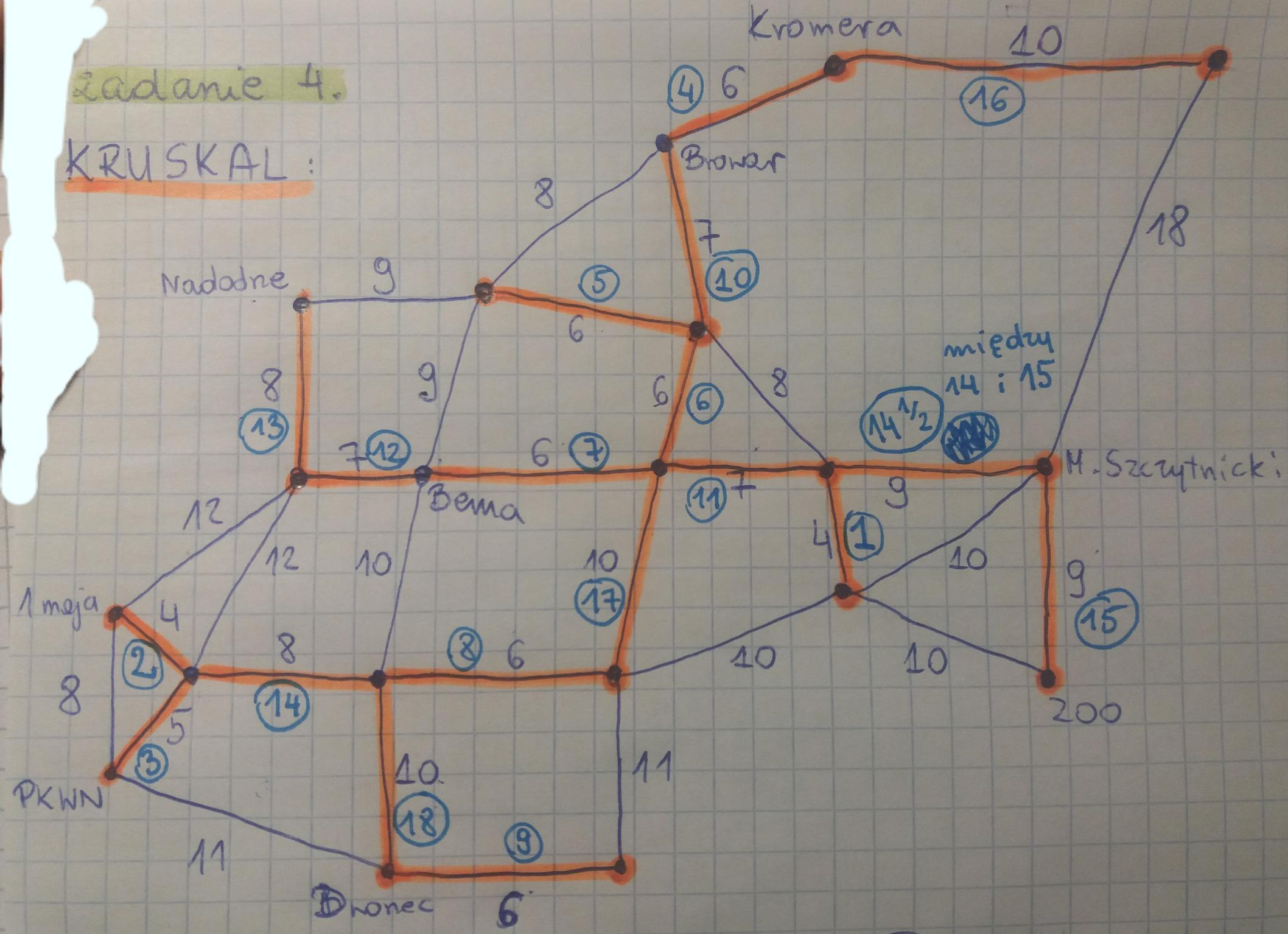
for (i=0; i < n; i++)           nieprzeniesiony
    if (V[i] == false) // nieodwiedzony
        visited ← empty-list();
        DFS(i, V, visited, G)
        // DFS odznaczył przeniesione wierzch.
        // w V i wypełnił listę visited
        for-each v in visited
            TopOrd.push-front(v)

return TopOrd.

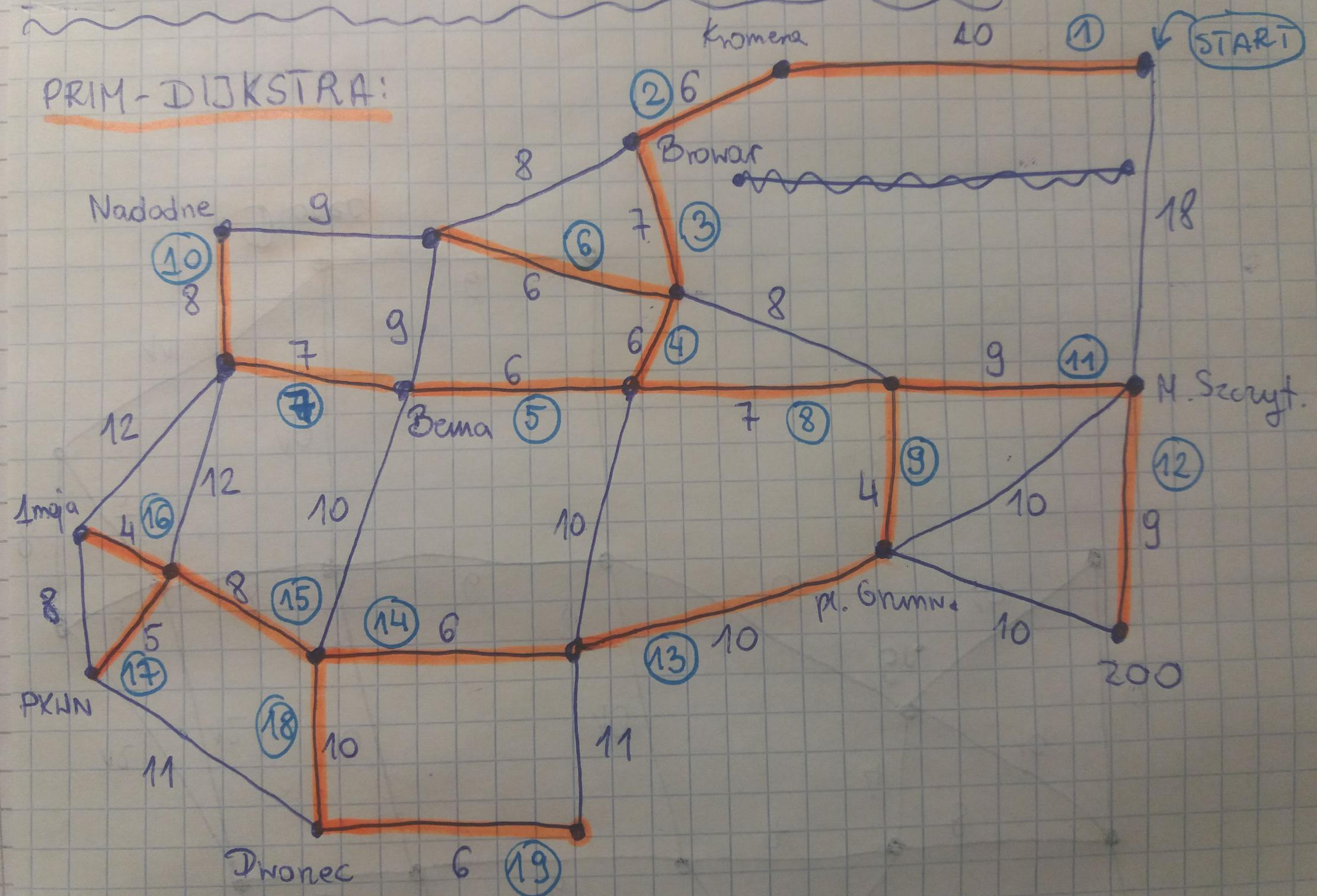
```

## Zadanie 4.

## KRUSKAL :



## PRIM - DIJKSTRA:



## Zadanie 6

Wiemy, że jeśli algorytm:

- ma  $n-1$  kroków
- w każdym z nich dodaje najkrótszą krawędź \* między niepołączonymi  $S \in V(G) \setminus S$  do  $T$  (gdzie na początku  $T$  - puste)

wyznacza najkrótsze dnewo spinające.

Analogicznie można stwierdzić, że jeśli algorytm:

- ma  $m-(m-1)$  kroków (1)
- w każdym z nich zabiera najdłuższą krawędź, która nie powoduje rozspójnienia grafu  $T$  (gdy T na początku - wszystkie krawędzie)

to też wyznacza najkrótsze dnewo spinające (2)  
dzielając „z drugiej strony”.

~~~~~

Nasz algorytm usuwa wszystkie krawędzie, które nie są konieczne do zachowania spójności. T to więc (popr. po wykonaniu algorytmu) graf spójny, bez cykli, czyli dnewo. Ma ono  $n$  wierzchołków, co oznacza  $n-1$  krawędzi, co oznacza, że musimy usunąć  $m-(m-1)$  krawędzi, co spełnia (1). Ponadto ponieważ  $c(e_1) > \dots > c(e_m)$ , to zawsze zabieramy najdłuższą krawędź, co spełnia (2). ■

## zadanie 7

Postępujemy prawie identycznie jak poszukując najmniejszego  
drewa rozpinającego

Zmodyfikowany algorytm Kruskala:

1. sortujemy krawędzie malejco (pod względem wag)  
mamy więc  $c(e_1) \geq c(e_2) \geq \dots \geq c(e_m)$
2.  $T \leftarrow \emptyset$

3. dla  $i = 1 \dots m$

jeśli dodanie  $e_i$  do  $T$  nie tworzy cyklu :

$$T \leftarrow T \cup \{e_i\}$$

zwrócić  $T$

## zadanie 8

Dane wejściowe: uint32-t G[32]

for k = 1 ... n

for i = 1 ... n

for j = 1 ... n

$G[i] := G[i] | (G[i] \& (1 << (32-k))) \quad \& \quad G[k] \& (1 << (32-j))$

## Zadanie 1

Wykonujemy algorytm Kruskala, po drodze formując odniesione krawędzi o najmniejszej wage.

1. sortujemy krawędzie rosnąco :  $c(e_1) \leq c(e_2) \leq \dots \leq c(e_m)$

2.  $T_1 \leftarrow \emptyset$ , ~~min~~  $\min \leftarrow \infty$

3. dla  $i=1 \dots m$

jeśli dodanie  $e_i$  do  $T_1$  nie tworzy cyklu :

$$T_1 \leftarrow T_1 \cup \{e_i\}$$

wpp.

jeśli  $c(e_i) < \min$

$$\min \leftarrow e_i$$

4. jeśli  $\min = \infty$  return NULL

// graf był dnewem, więc nie  
istnieje 1-dnewo spinające

wpp return  $T_1 \cup \{\min\}$

W poniższym algorytmie najpierw znalezliśmy najkrótsze dnewo spinające, a na samym koncu dodaliśmy krawędzi do niego niewłaściwą, która ma najmniejszą wagę. Wobec tego uzyskaliśmy najkrótsze 1-dnewo rozpinające.

## zadanie 12

I)

$$\text{MST}(G) \leq \text{TSP}(G)$$

$\text{MST}(G)$  to suma wag krawędzi okregu spinającego, czyli okregu, którego wierzchołkami są wszystkie  $v \in V(G)$ .

Ponadto, jest ona najmniejsza z możliwych.

Komisjoner nie może więc odwiedzić wszystkich wierzchołków z sumą "uznaczonych" wag mniejszą niż  $\text{MST}(G)$ .

II)

~~TSP(G) > 2 \* MST(G)~~

~~TSP(G) < 2 \* MST(G)~~

$$\text{TSP}(G) \leq 2 \cdot \text{MST}(G)$$

Rozważamy najgorszy przypadek dla komisjazera.

Aby odwiedzić kożdy  $v \in V(G)$  może poszukać się algorytmem DFS unikomionym na MST. Wtedy, w najgorszym przypadku, każda krawędź w tym okregu przejdzie dwukrotnie, wobec czego

$$\text{TSP}(G) \leq 2 \cdot \text{MST}(G)$$