

Since the node-edge incident matrix has  $n'$  elements, there can't be a faster method (for complete graphs)

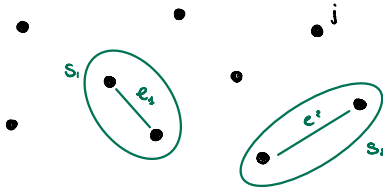
For sparse graphs, we usually have  $|E| \sim 2|V| \Rightarrow$  Can use a better algorithm  
 $\downarrow$   
 $O(|E| \log |V|)$   
 (best Dijkstra implementation)

## Kruskal method

- Sort the edges by increasing costs

$$c_{e_1} < c_{e_2} < \dots < c_{e_m}$$

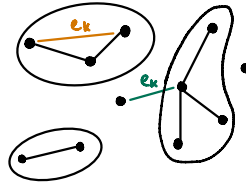
$$T^* = \emptyset$$



Every node  $j$  corresponds to a connected component

$$T^* = \{e_1, e_2, \dots\}$$

- Iteration  $k$ :



- if  $e_k$  connects two different components:  
create the connection
- if  $e_k$  connects two nodes in the same component: ignore the edge

## Pseudocode:

$$T^* = \emptyset$$

$$C[i] = \{i\}, \forall i = 1, \dots, m \quad O(n)$$

$$\text{sort all edges} \quad O(m \log(m)) = O(m \log(n^2)) = O(m \log n)$$

$$\text{while } |T^*| < n-1 \text{ do} \quad O(n)$$

$$[i, j] = e_k$$

$$\Rightarrow O(m \log n) + (n-1) O(n \log n)$$

if  $C[i] \neq C[j]$  then

$$T^* = T^* \cup \{[i, j]\}$$

merge  $(C[i], C[j]) \rightarrow$  Bottleneck  $\Rightarrow$  must find a fast method

to merge the two lists

$(O(n))$  makes the algorithm  $O(n^2)$

else

$$k = k + 1$$

OK