Since the node-edge incident matrix has n^2 elements, there can't be a faster method (for complete graphs)

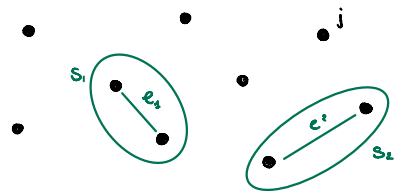
For sparse graphs, we usually have $|E| \sim 3|V|$ ⇒ Can use a better algorithm
↓
$O(|E| \log |V|)$
(best Dijkstra implementation)

Kruscal method (greedy algorithm)

- Sort the edges by increasing costs

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

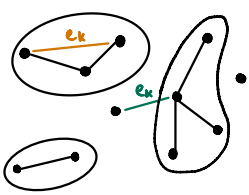
- $T^* = \phi$



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* = ∅
C[i] = {i}, ∀ i = 1, ..., m        O(n)
sort all edges                     O(m log(m)) = O(m log(n²)) = O(m log n)
while |T*| < n-1 do                O(n)
    [i,j] = e_k                    ⇒ O(m log n) + (n-1) O(MERGE)   !
    if C[i] ≠ C[j] then
        T* = T* ∪ {[i,j]}
        merge (C[i], C[j])         → Bottleneck ⇒ must find a fast method
                                                  to merge the two lists
                                                  (O(n) makes the algorithm O(n²))
    else
        k = k+1
```