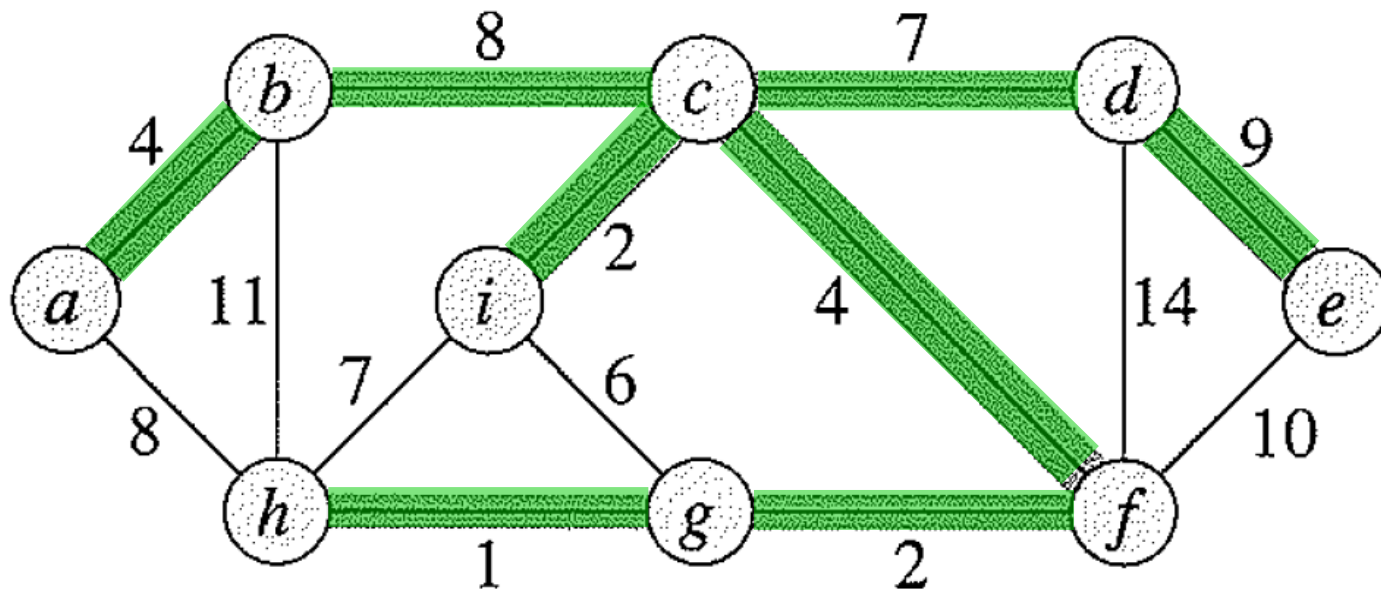


# Algoritmer og Datastrukturer

Minimum Udspændende Træer (MST)  
[CLRS, kapitel 23]

# Minimum Udspændende Træ (MST)

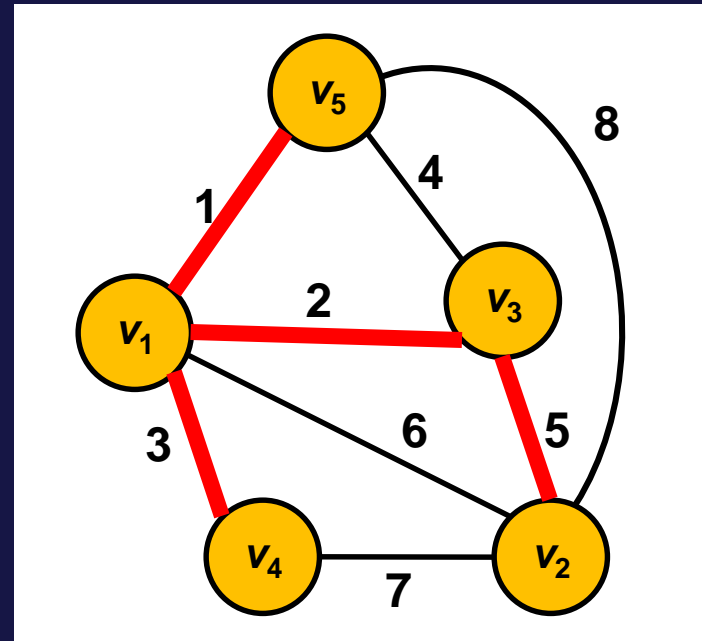


## Problem

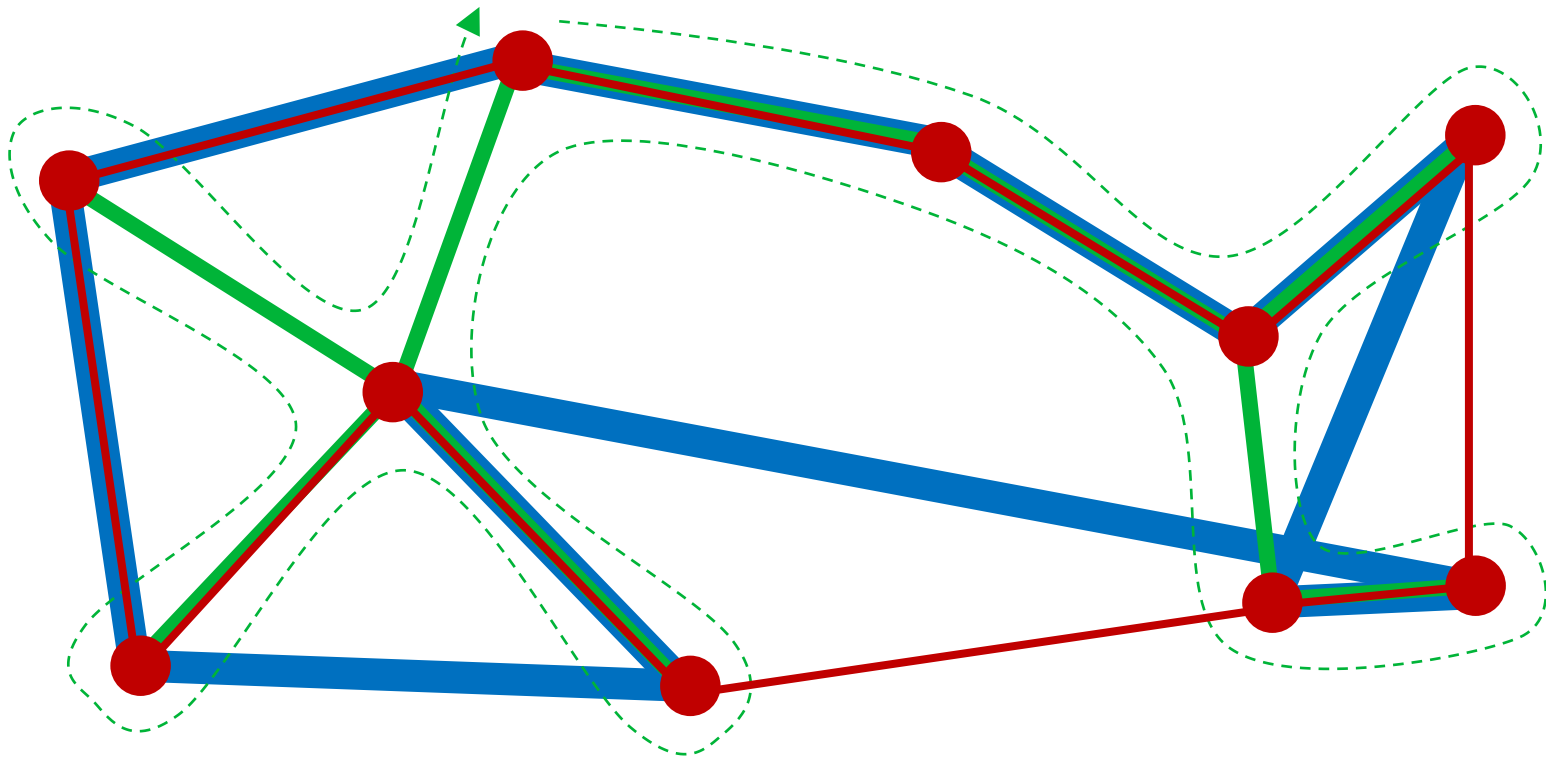
Find et **udspændende træ** for en **sammenhængende uorienteret vægtet** graf således at **summen af kanterne er mindst mulig**

# Vægten af et minimum udspændende træ?

- a) 10
- b) 11
- c) 12
- d) 13
- e) 14
- f) 15
- g) 16
- h) Ved ikke



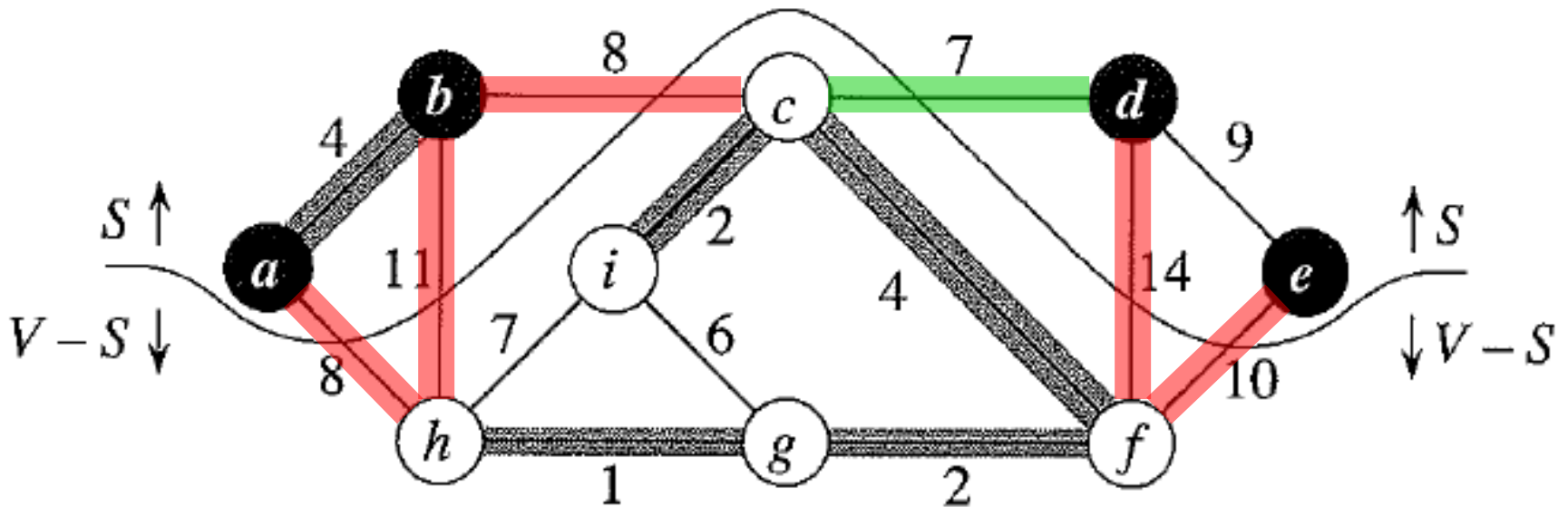
# En anvendelse af MST : (Euklidisk) Korteste Hamiltonske Cykel



**Sætning :** **Touren** fundet vha. et **MST**  
er en 2-approximation til en **korteste cykel**

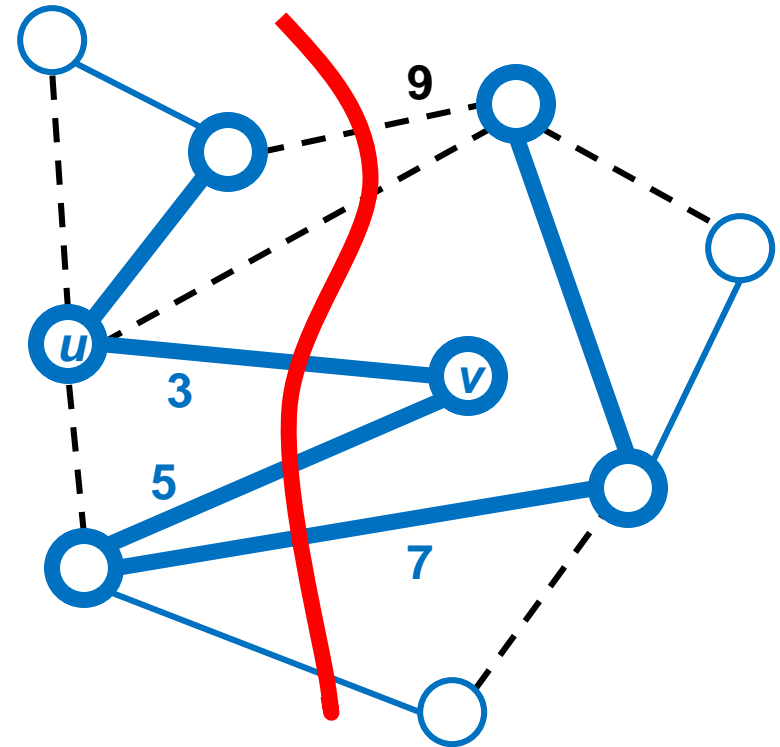
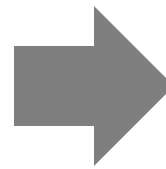
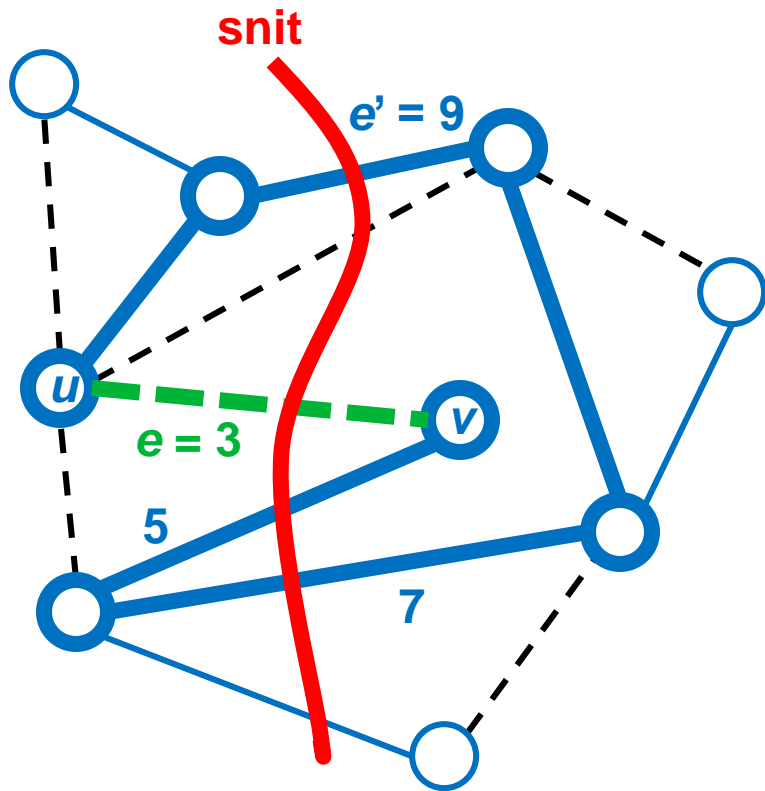
$$( |\text{Touren}| \leq 2 \cdot |\text{MST}| \quad \text{og} \quad |\text{MST}| \leq |\text{korteste cykel}| )$$

# Minimum Udspændende Træer: Snit



## Sætning

Hvis alle vægte er forskellige, så gælder der for *ethvert* **snit (S, V-S)** at den **letteste kant** der krydser snittet er med i et minimum udspændende træ



Antag modsætningsvis et MST med et snit hvor mindste kant  $e$  ikke er med i MST

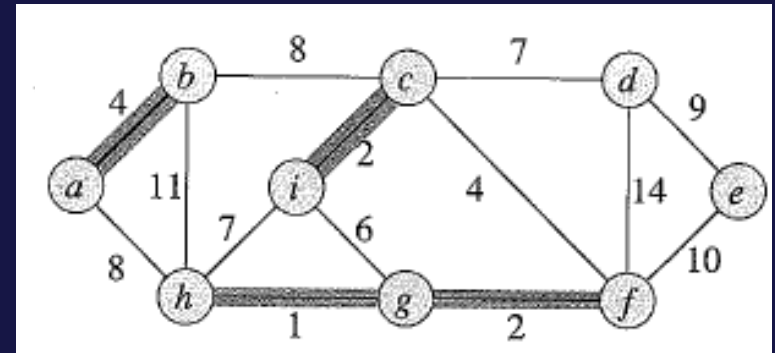
Nyt udspændende træ med mindre vægt ⚡

## Sætning

Hvis alle vægte er forskellige, så gælder der for *ethvert* snit  $(S, V-S)$  at den **letteste kant** der krydser snittet er med i et minimum udspændende træ

Hvis man for en sammenhængende graf endnu ikke har fundet tilstrækkeligt mange kanter til at de udgør et MST, kan man så altid anvende snit sætningen ?

- a) Ja
- b) Nej
- c) Ved ikke

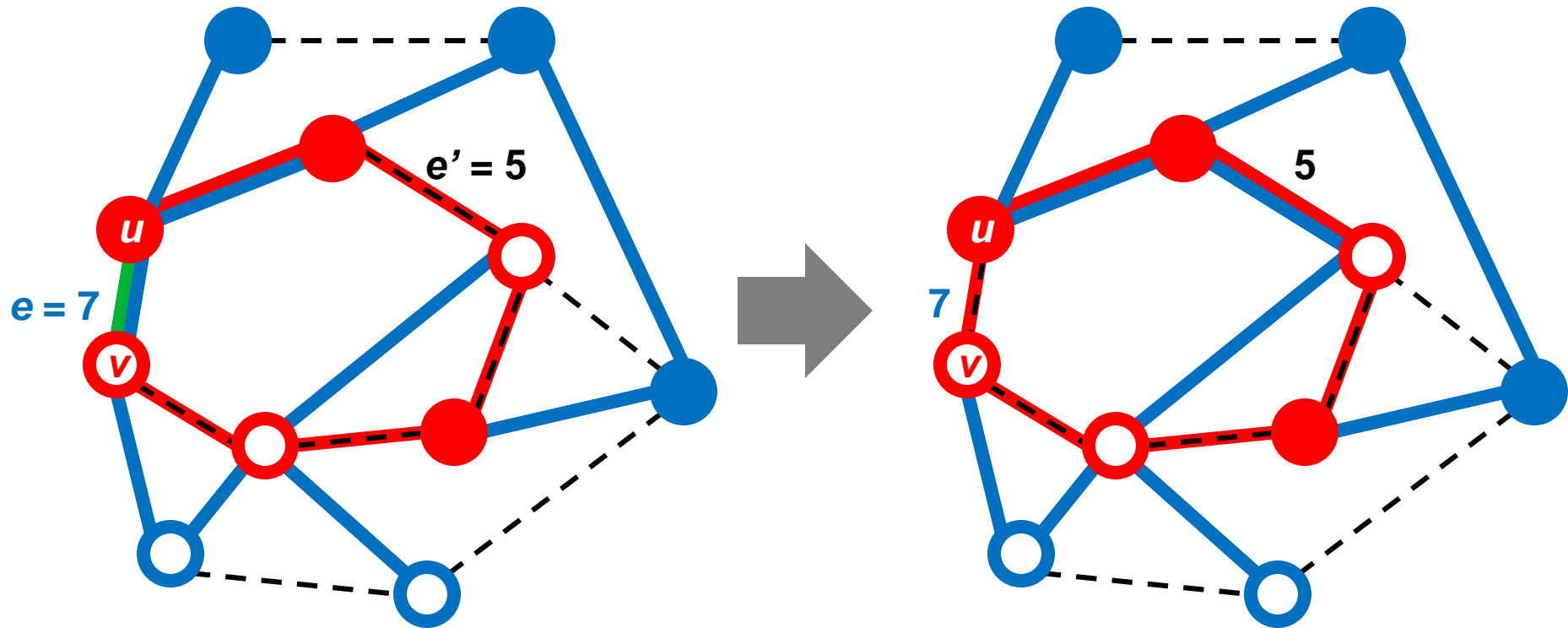


### Sætning

Hvis alle vægte er forskellige, så gælder der for *ethvert* **snit (S, V-S)** at den **letteste kant** der krydser snittet er med i et minimum udspændende træ

## Sætning

Hvis alle vægte er forskellige, så gælder der for *enhver cykel* at den **tungeste kant i cyklen** ikke er med i et minimum udspændende træ



Antag modsætningsvis et MST og *cykel*  
hvor **tungeste kant**  $e$  er med i MST


Nyt udspændende træ  
med mindre vægt ⚡



# Minimum Udspændende Træer: Grådig generel algoritme

GENERIC-MST( $G, w$ )

```
1   $A = \emptyset$ 
2  while  $A$  does not form a spanning tree
3      find an edge  $(u, v)$  that is safe for  $A$ 
4       $A = A \cup \{(u, v)\}$ 
5  return  $A$ 
```



En letteste kant over et  
snit (som ikke allerede  
indeholder kanter fra  $A$ )

# Kruskall's Algoritme

MST-KRUSKAL( $G, w$ )

1  $A = \emptyset$

2 **for** each vertex  $v \in G.V$

3     MAKE-SET( $v$ )

flaskehals i algoritmen

4 sort the edges of  $G.E$  into nondecreasing order by weight  $w$

5 **for** each edge  $(u, v) \in G.E$ , taken in nondecreasing order by weight

6     **if** FIND-SET( $u$ )  $\neq$  FIND-SET( $v$ )

7          $A = A \cup \{(u, v)\}$

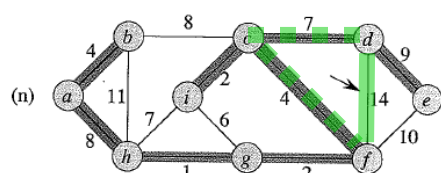
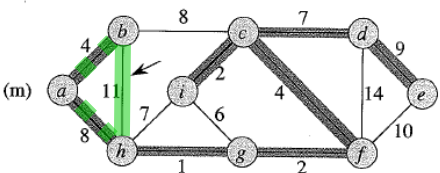
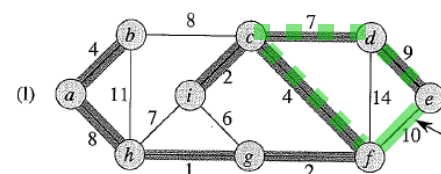
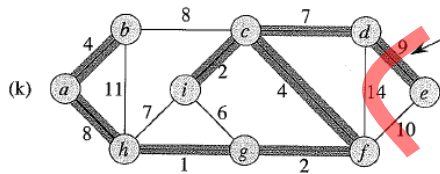
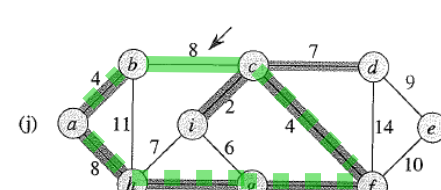
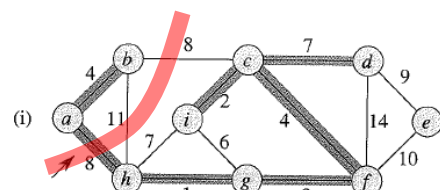
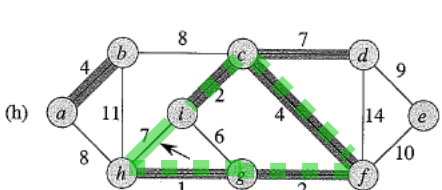
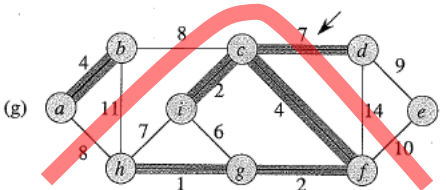
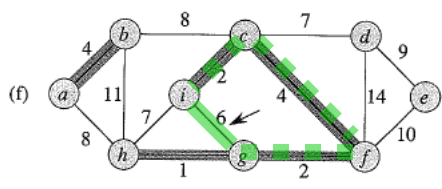
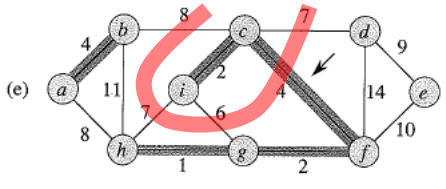
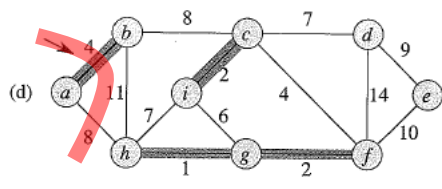
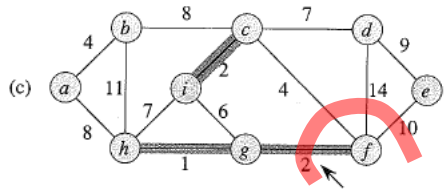
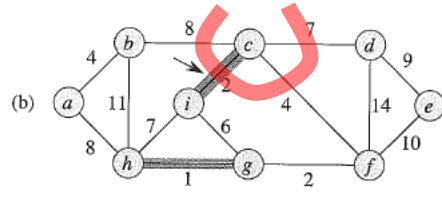
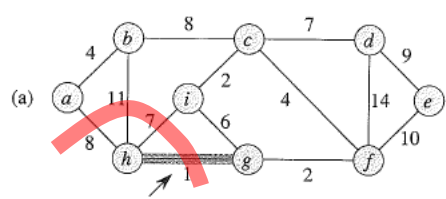
8         UNION( $u, v$ )

9 **return**  $A$

Union-Find  
datastruktur

Tid  $O(m \cdot \log n)$

# Kruskall's Algoritme: Eksempel



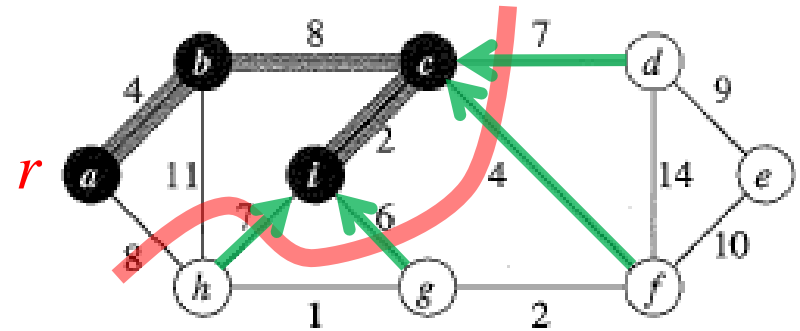
Kantene betrages efter stigende vægte (angivet med ↗)

For hver kant er angivet **snittet** hvor den er en letteste kant eller **cyklen** hvor den er en tungeste kant

# Prim's Algorithm

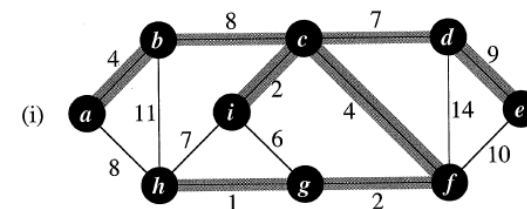
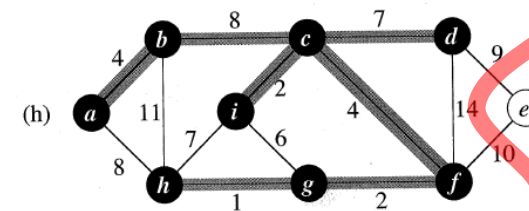
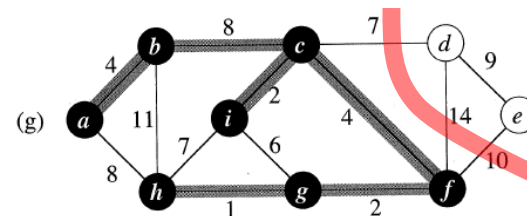
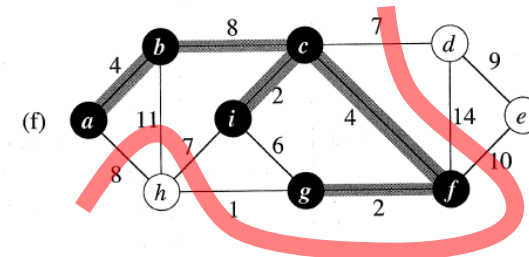
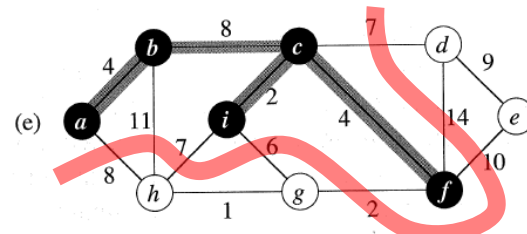
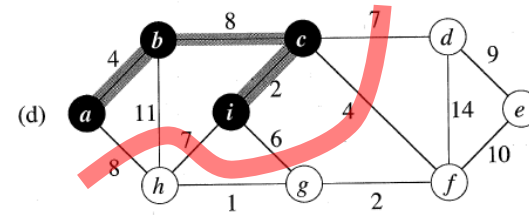
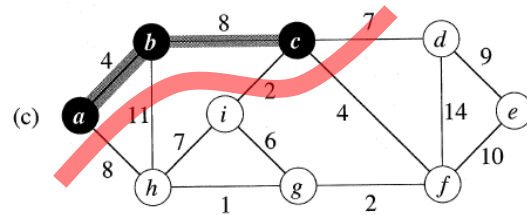
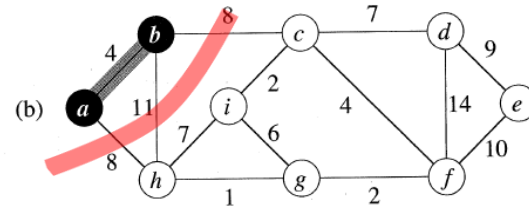
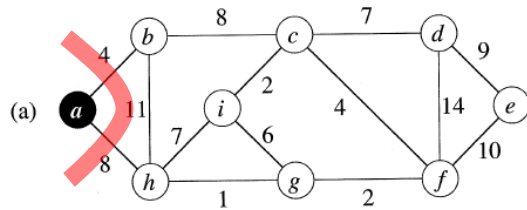
MST-PRIM( $G, w, r$ )

```
1  for each  $u \in G.V$ 
2       $u.key = \infty$ 
3       $u.\pi = \text{NIL}$ 
4   $r.key = 0$ 
5   $Q = G.V$ 
6  while  $Q \neq \emptyset$ 
7       $u = \text{EXTRACT-MIN}(Q)$ 
8      for each  $v \in G.Adj[u]$ 
9          if  $v \in Q$  and  $w(u, v) < v.key$ 
10              $v.\pi = u$ 
11              $v.key = w(u, v)$ 
```



flaskehals i algoritmen  
- prioritetskø

# Prim's Algorithm: Eksempel



# Eksamensopgave 3(c)

## august 2005

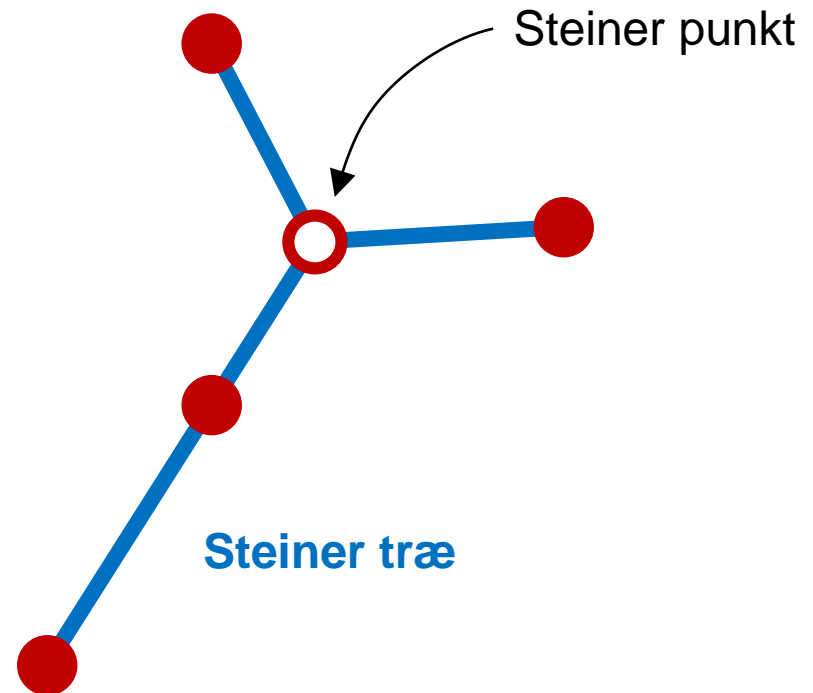
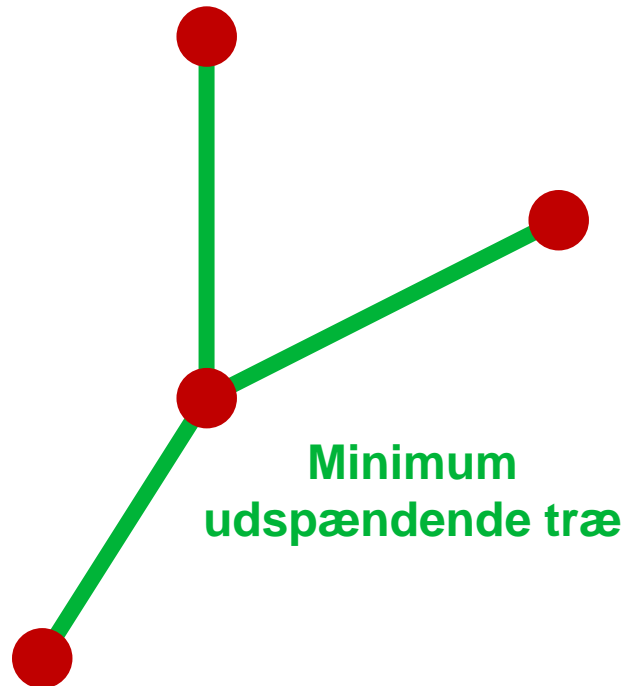
**Spørgsmål c:** Beskriv en algoritme, der givet en vægtet graf  $G$  og en kant  $e$  i grafen, afgør om  $e$  er indeholdt i et minimum udspændende træ for  $G$ . Det antages at alle vægtene er forskellige. Udførelstiden for algoritmen skal være  $O(m)$ , hvor  $m$  er antal kanter i grafen. □

# Minimum Udspænding Træer

<b>Kruskall (1956)</b> (mange træer; sorterer kanterne)	$O(m \cdot \log n)$
<b>Prim (1930)</b> (et træ; prioritetskø over naboknuder)	$O(m \cdot \log n)$
	$O(m + n \cdot \log n)$ (Fibonacci heaps [CLRS, kapitel 19] (1984))
<b>Borůvka (1926)</b> (mange træer samtidigt; kontraktion)	$O(m \cdot \log n)$
<b>Fredman, Tarjan (1984)</b> (Borůvka (1926) + Fibonacci heaps)	$O(m \cdot \log^* n)$
<b>Chazelle (1997)</b>	$O(m \cdot \alpha(m, n))$
<b>Pettie, Ramachandran (2000)</b>	? (men optimal determinisk sammenligningsbaseret)
<b>Karger, Klein, Tarjan (1995)</b> (Randomiseret) <b>Fredman, Willard (1994)</b> (RAM)	$O(m)$

# (Euklidiske) Steiner Træer

Givet en mængde punkter, find et træ med minimum vægt som forbinder alle knuder, muligvis ved at tilføje **nye punkter**



**Conjecture (Steiner Ratio):**  $|MST| \leq \frac{2}{\sqrt{3}} \cdot |\text{Steiner Tree}|$

NP hårdt; polynomial-time approximation scheme (PTAS)