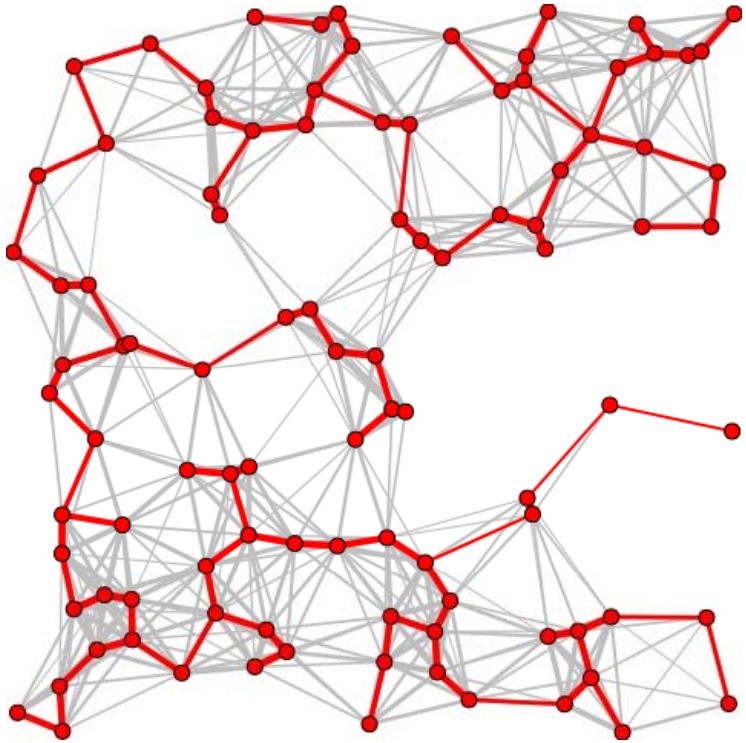


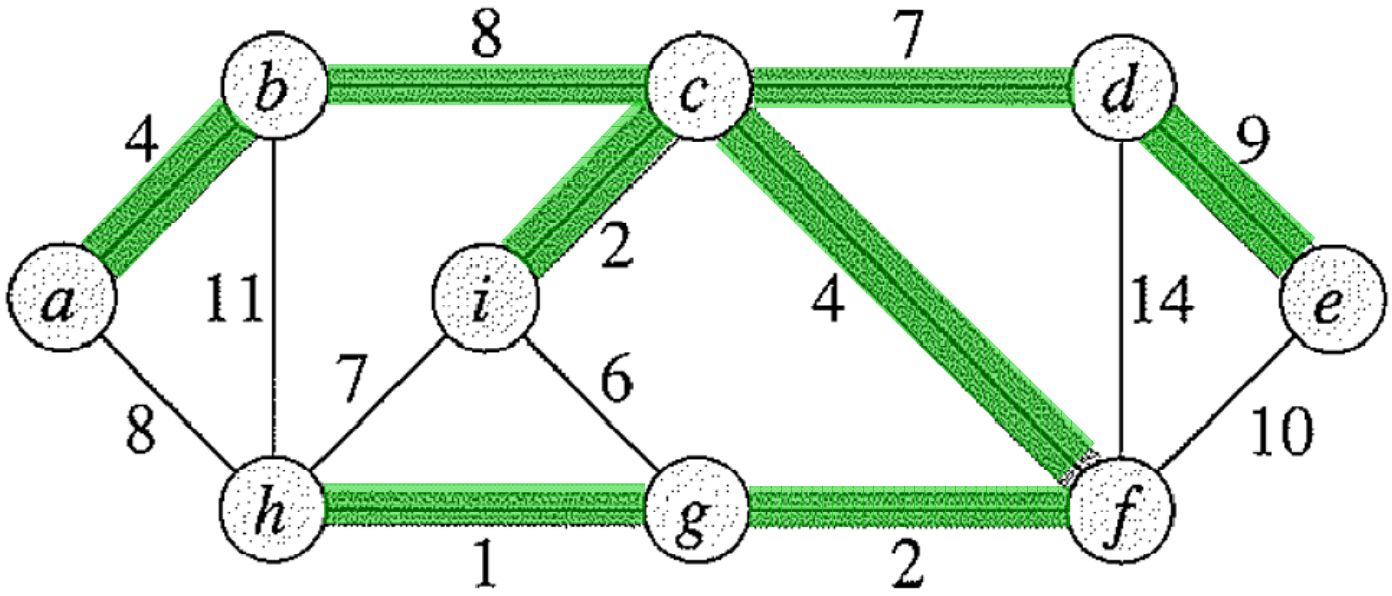
Algoritmer og Datastrukturer 2

Gerth Stølting Brodal

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

AARHUS UNIVERSITET

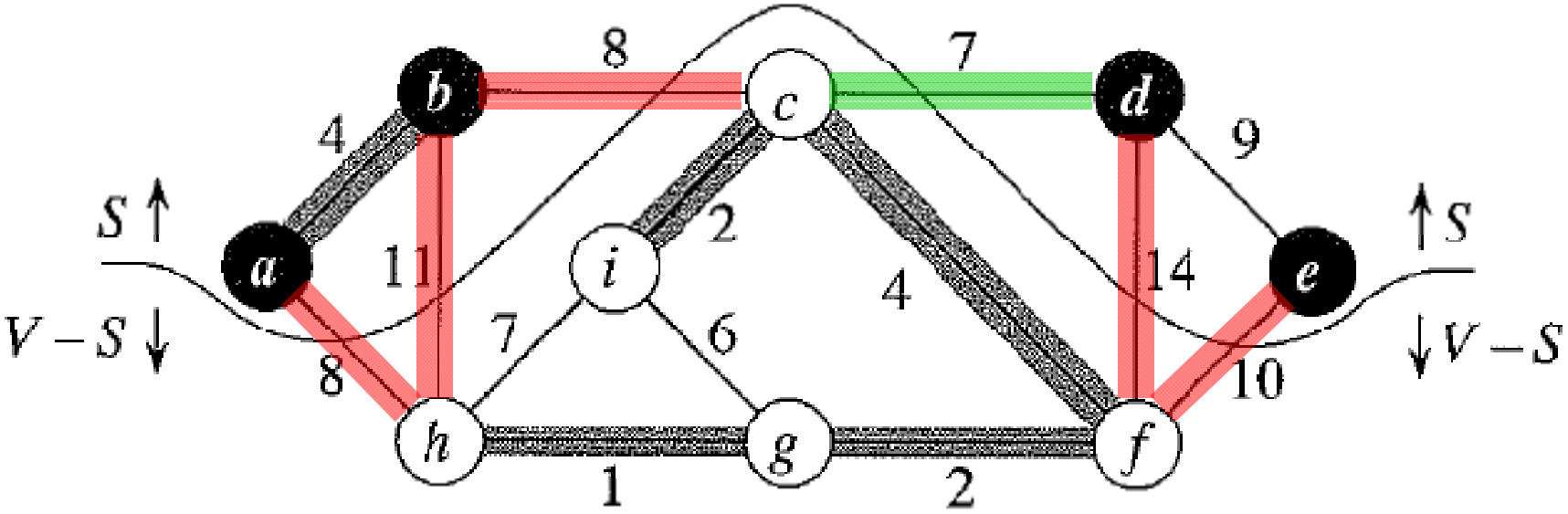
Minimum Udspændende Træ (MST)



Problem

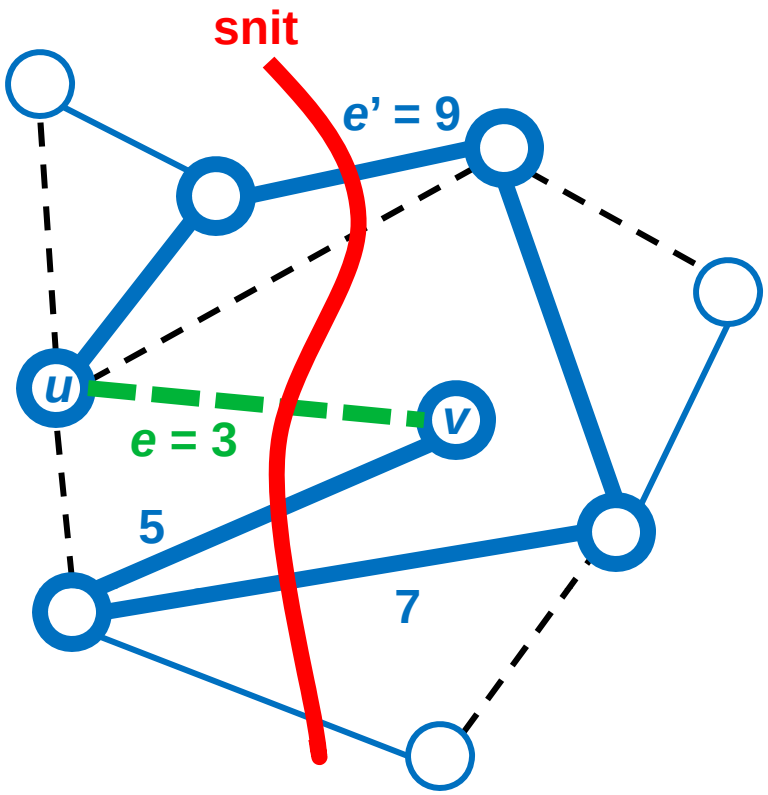
Find et udspændende træ for en sammenhængende uorienteret vægtet graf således at summen af kanterne er mindst mulig

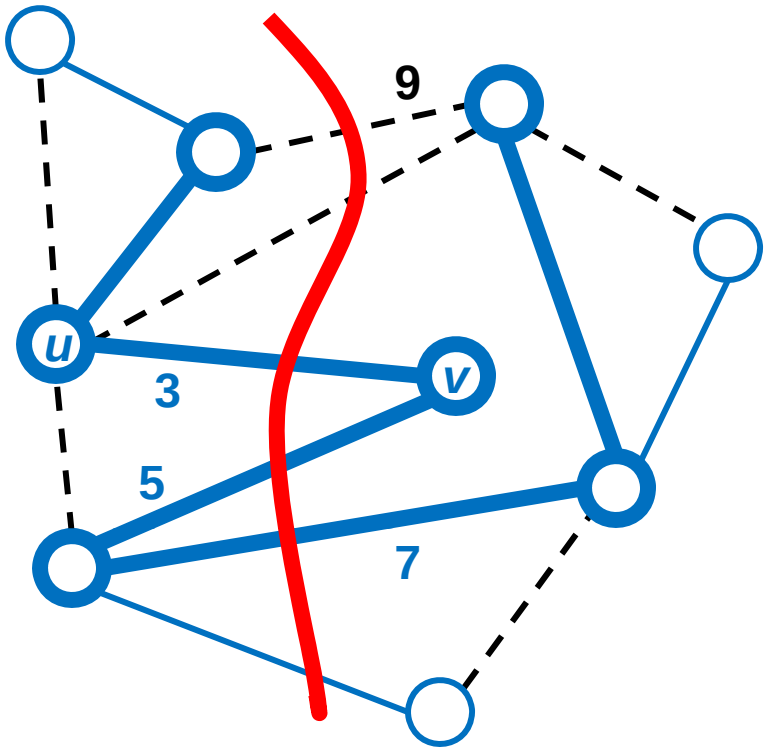
Minimum Udspændende Træer: Snit



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

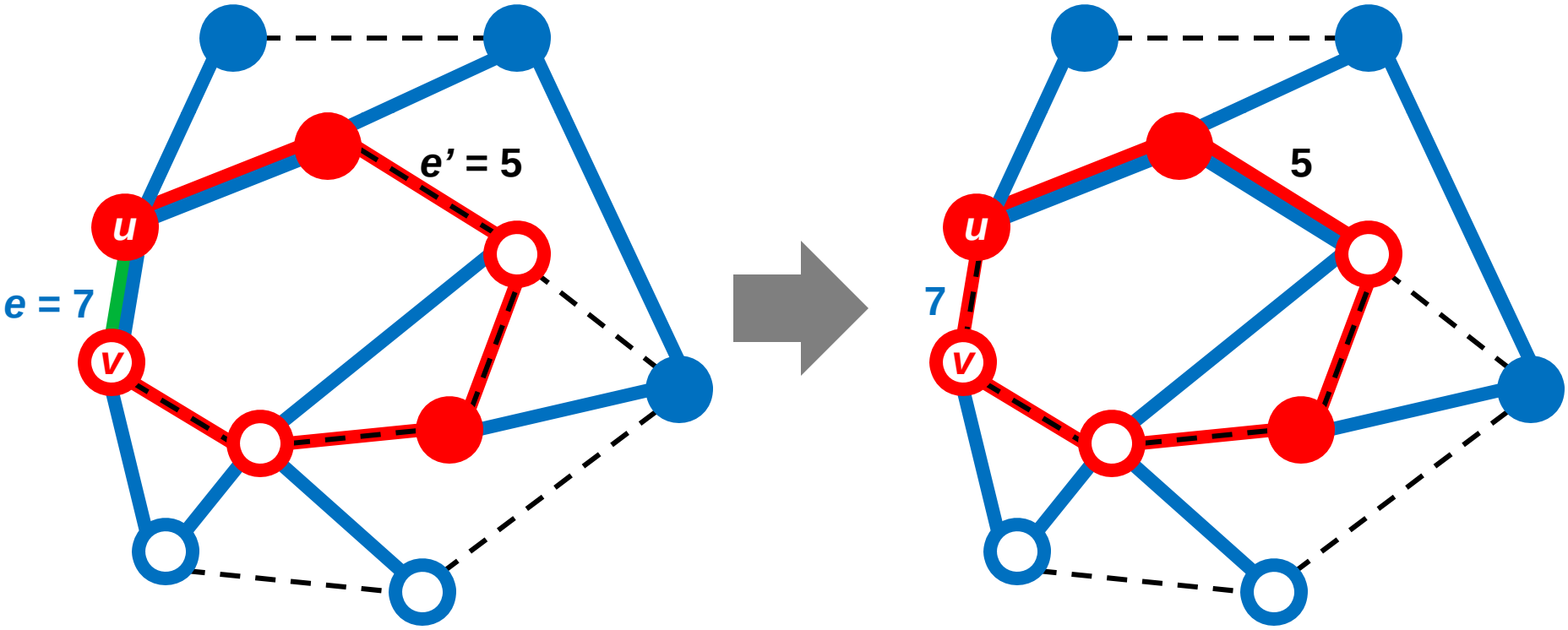
Nyt udspændende træ med mindre vægt

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

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

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

Kruskall's Algoritme

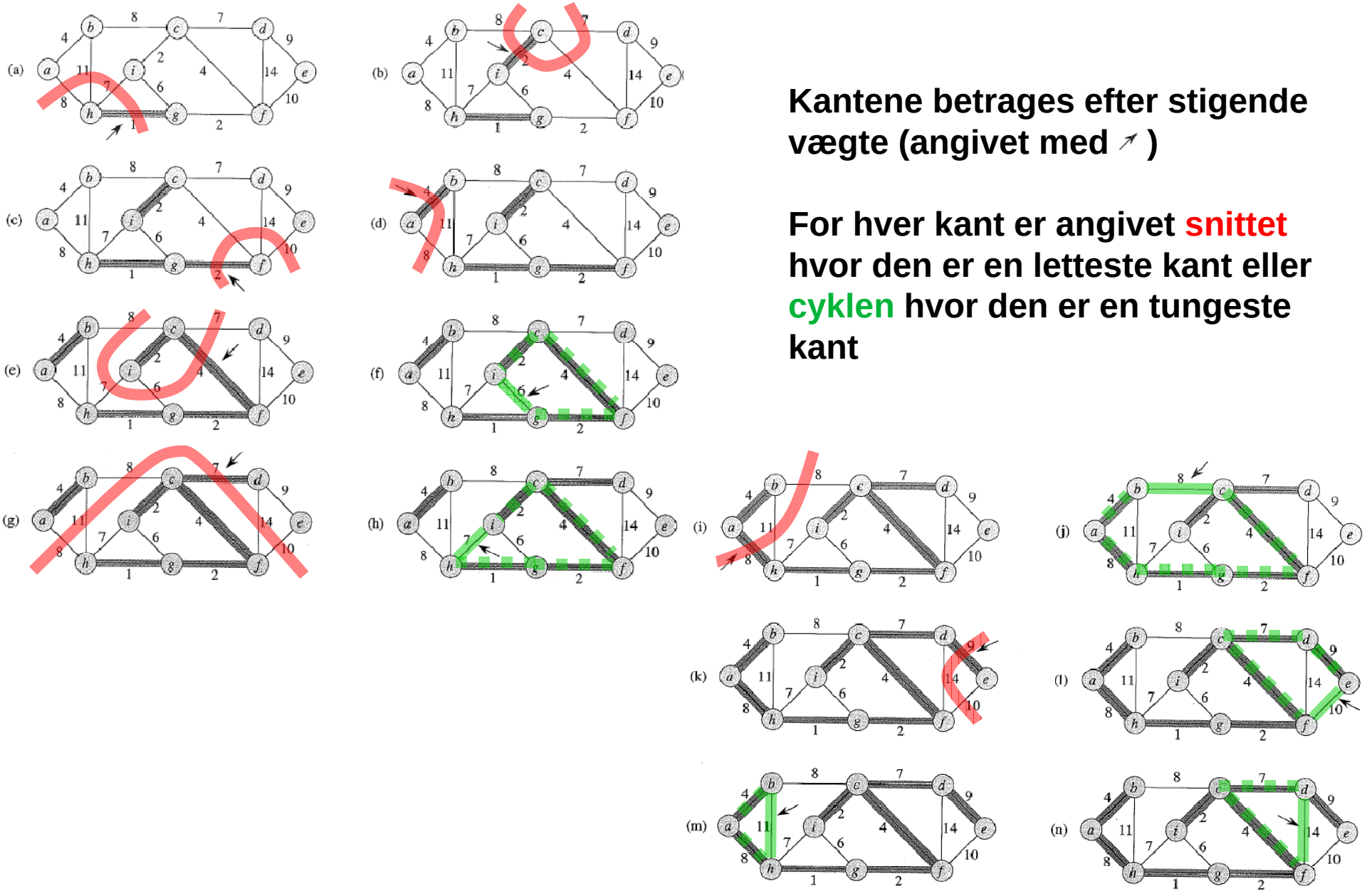
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort the edges of G.E into nondecreasing order by weight w
5  for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```

flaskehals i algoritmen

Union-Find datastruktur

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

Kruskall's Algoritme: Eksempel

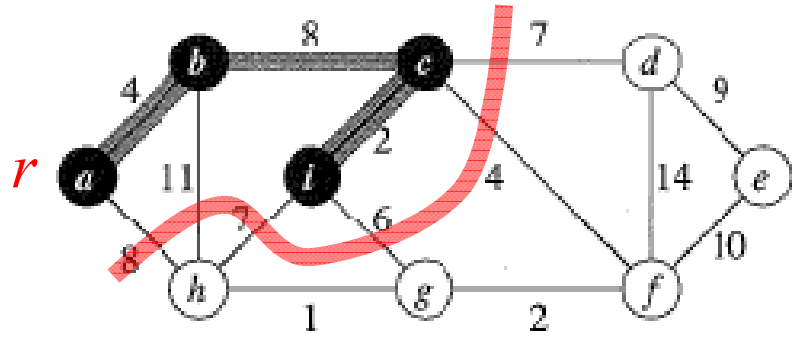


Kantene betrages efter stigende vægte (angivet med ↗)

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

Prim's Algoritme

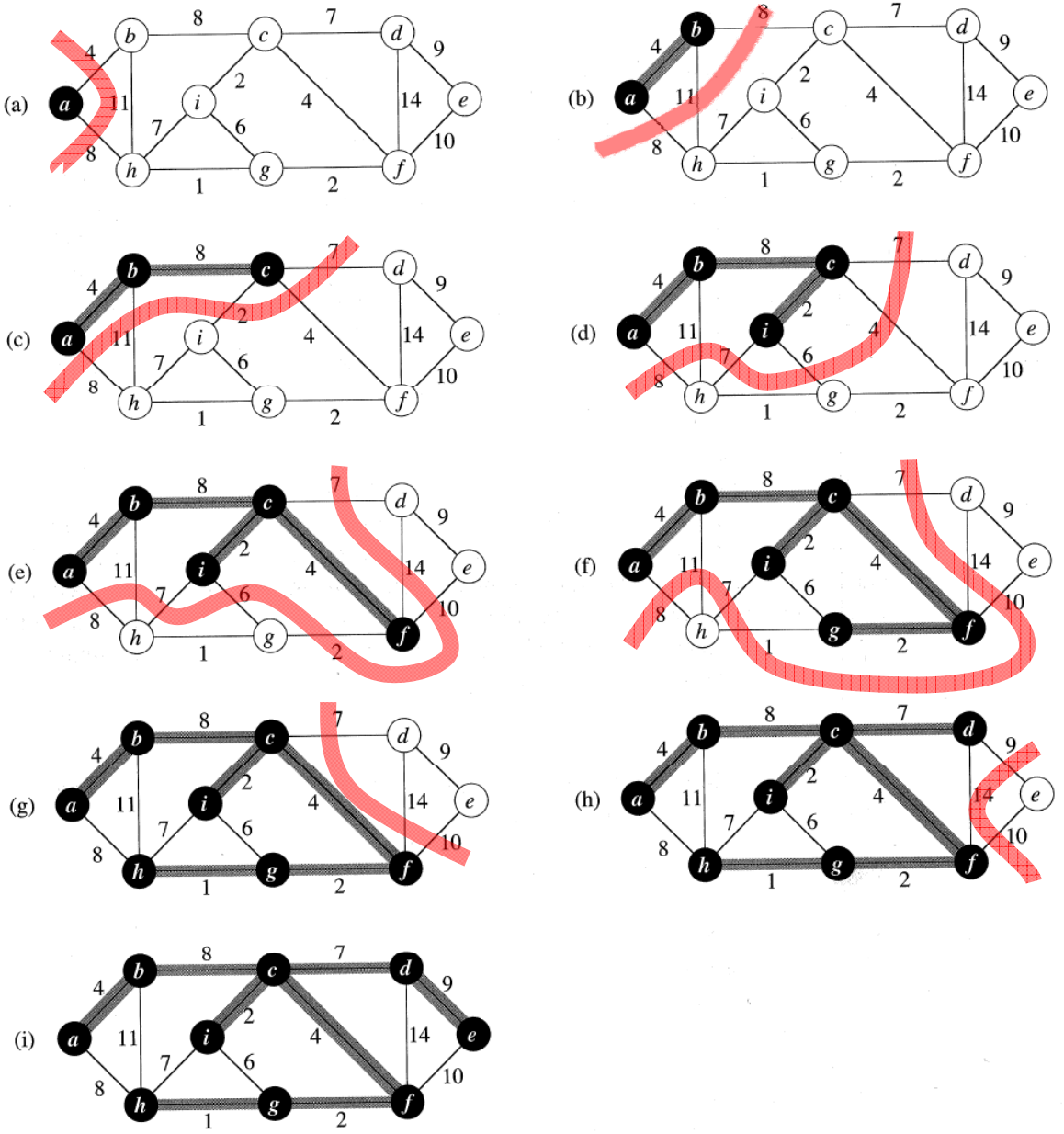
```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```



flaskehals i algoritmen
- prioritetskø

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

Prim's Algoritme: Eksempel



Minimum Udspænding Træer

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