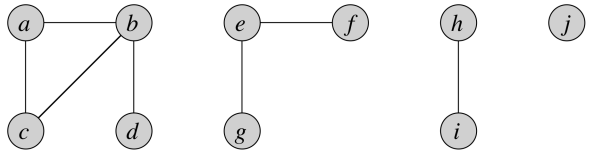
Disjoint Sets

Recap: Kruskal MST-algoritmen

Forsøger at tilføje kanter til A i letteste-først-orden. Tilføjer kun kant (u, v) til A hvis der ikke laves en kreds, dvs. hvis u og v ligger i forskellige træer. Hvis (u, v) tilføjes, vil disse to træer blive til eet bagefter.



(a)

Edge processed	Collection of disjoint sets									
initial sets	{a}	{b}	{c}	{d}	{e}	{f}	{g}	{h}	{i}	{j}
(b,d)	{a}	{b,d}	{c}		{e}	{f}	{g}	{h}	{i}	{j}
(e,g)	{a}	{b,d}	{c}		{e,g}	{f}		{h}	{i}	{j}
(a,c)	{a,c}	{b,d}			{e,g}	{f}		{h}	{i}	{j}
(h,i)	{a,c}	{b,d}			{e,g}	{f}		{h,i}		{j}
(a,b)	{a,b,c,d}				{e,g}	{f}		{h,i}		{j}
(e,f)	{a,b,c,d}				{e,f,g}			{h,i}		{j}
(b,c)	{a,b,c,d}				{e,f,g}			{h,i}		{j}

(b)

Vedligeholder opdelingen i træer i A ved hjælp af en disjoint-set datastruktur på V:

$$\textsc{Make-Set}(x),\ \textsc{Union}(x, y)\ \textsc{Find-Set}(x)$$

Disjoint Sets operationer

MAKE-SET(x):

Opret $\{x\}$ som en mængde.

UNION(x, y):

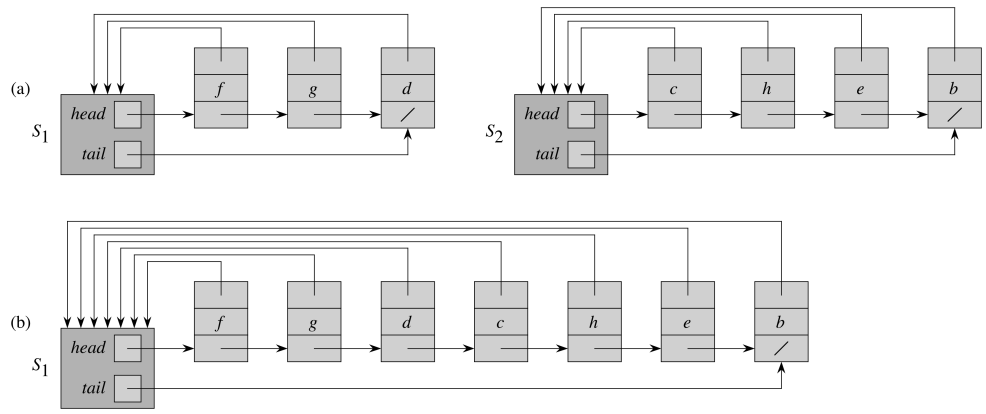
Slå $\{a, b, c, \ldots, x\}$ og $\{h, i, j, \ldots, y\}$ sammen til $\{a, b, c \ldots, h, i, j, \ldots, x, y\}$.

FIND-SET(x):

Returner en ID for mængden indeholdende x.

Datastruktur for Disjoint Sets

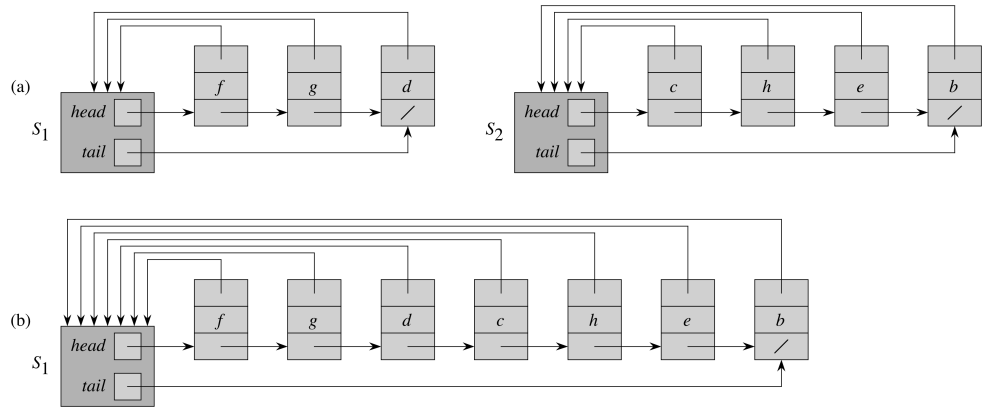
En simpel løsning:



- MAKE-SET(x): opret ny liste.
- UNION(x, y): slå lister sammen, behold een header, ændrer alle header-pointere i den anden liste.
- FIND-SET(x): returner pointer til header.

Datastruktur for Disjoint Sets

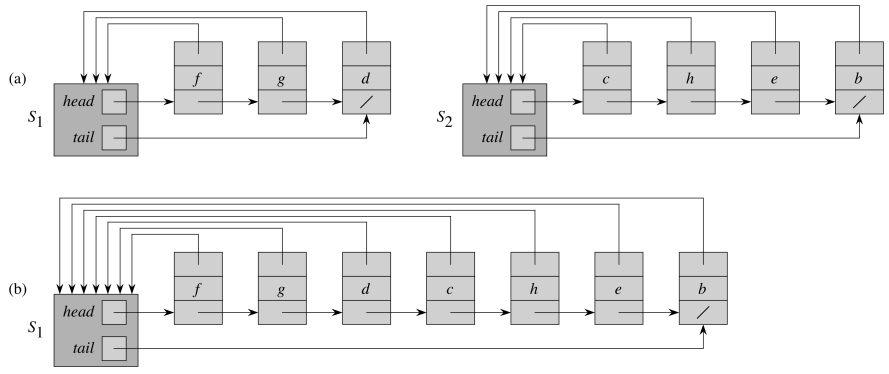
Køretid (efter n MAKE-SET):



- MAKE-SET(x): opret ny liste: $O(1)$.
- UNION(x, y): slå lister sammen, behold een header, ændre alle header-pointere i den anden liste: $O(n)$.
- FIND-SET(x): returner pointer til header: $O(1)$.

Naiv analyse: n MAKE-SET, op til $n-1$ UNION, og m FIND-SET koster $O(m+n^2)$.

Datastruktur for Disjoint Sets

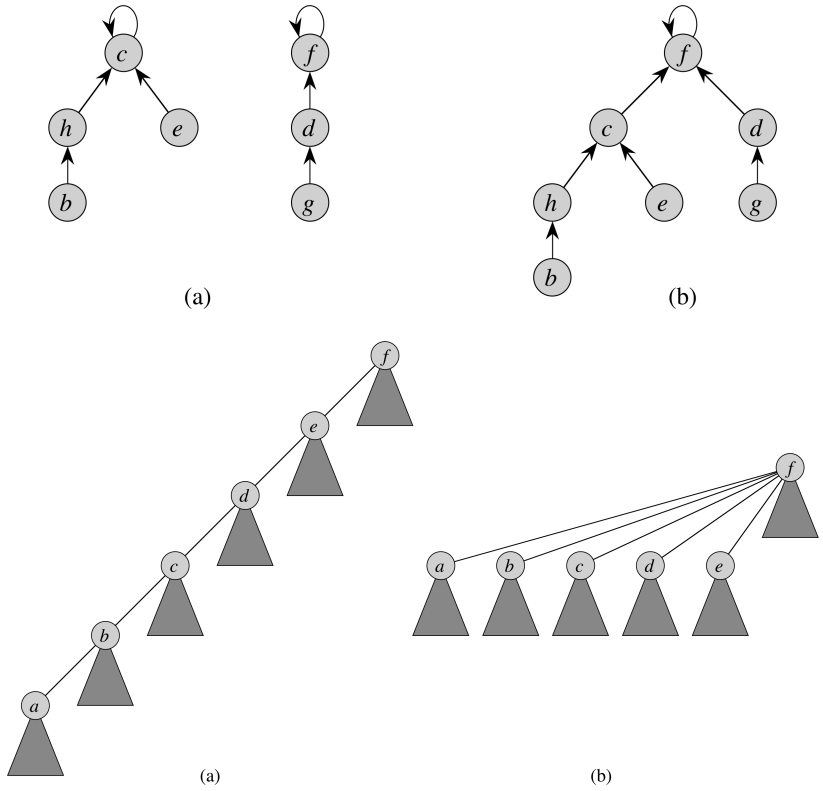


- MAKE-SET(x): opret ny liste: $O(1)$.
- UNION(x, y): slå lister sammen, behold header af længste liste, ændre alle header-pointere i korteste liste: $O(n)$.
- FIND-SET(x): returner pointer til header: $O(1)$.

En knude kan kun ændre sin header-pointer $k = \log n$ gange, da størrelsen af dens mængde hver gang vokser mindst en faktor to ($1 \cdot 2^k \leq n \Leftrightarrow k \leq \log n$).

Bedre analyse: n MAKE-SET, op til $n - 1$ UNION, og m FIND-SET koster $O(m + n \log n)$.

En anden datastruktur



Union by rank + path compression $\Rightarrow$ meget tæt på $O(m+n)$ tid. Mere præcist $O(m \cdot \alpha(n))$.

Analyse: i et senere kursus.