Københavns Universitet
Introduktion til diskret matematik og algoritmer - Problem set 4

Victor Vangkilde Jørgensen - kft410
kft410@alumni.ku.dk

March 22, 2025

Contents

1

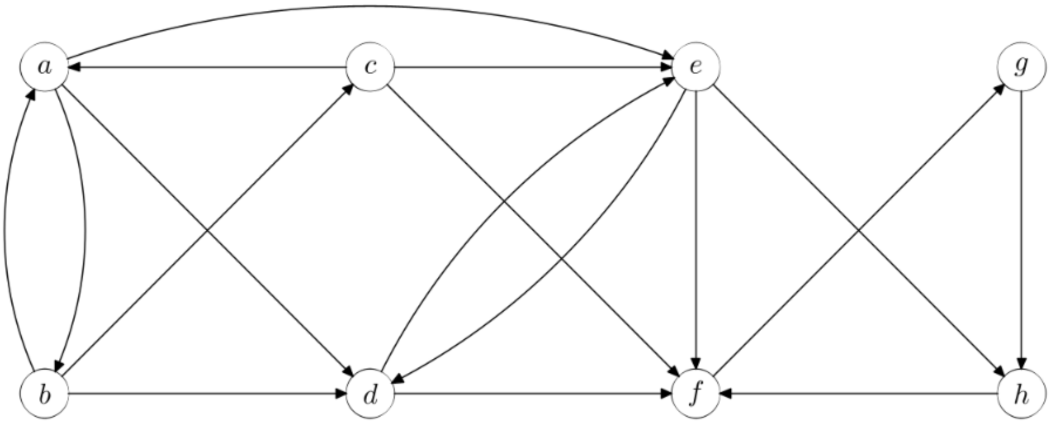


Figure 1: Directed graph G for which to compute strongly connected components in Problem 1a.

1.a

Vi opskriver vores directed graph som en adjacency list representation i lexicographic order:

$$
\begin{aligned}
a &\to (b, d, e)\\
b &\to (a, c, d)\\
c &\to (a, e, f)\\
d &\to (e, f)\\
e &\to (d, f, h)\\
f &\to (g)\\
g &\to (h)\\
h &\to (f)
\end{aligned}
$$

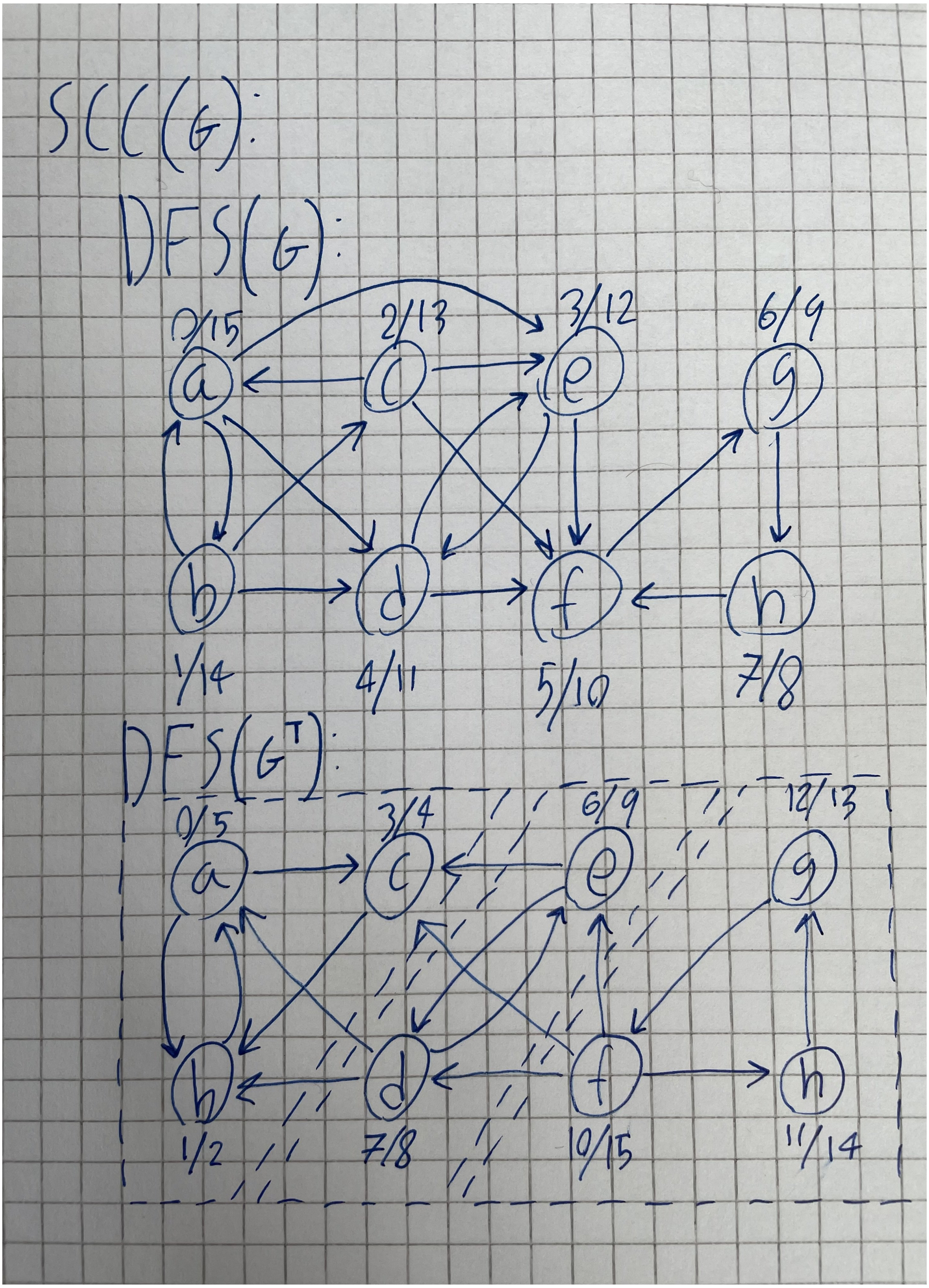


Figure 2: Gennemgang af $SCC(G)$.

For at beregne alle strongly connected components i en graf, skal vi køre DFS på grafen, og opbevare finish time'en for alle recursions. Dette har jeg gjort som vist i billedet ovenover hvor der står $DFS(G)$. Jeg har noteret start og sluttiderne hvor hver vertex med $starttid/sluttid$ over hver vertex. Starttiden bliver talt når recursion'en bliver kaldt på vertex'en, og sluttiden bliver noteret når recursion'en er færdig (DFS peger på en sort vertex).

Efter vi har kørt DFS noterer vi sluttiderne i decending order:

$$a, b, c, e, d, f, g, h$$

Nu transposer vi grafen, hvilket rent grafisk blot betyder, at vi 'vender' hver directed edge i grafen, således at $(v, u) \in E \rightarrow (u, v) \in E$. Herefter kører vi igen DPS, men vi køerer algoritmet på G^T. Vores start vertex sætter vi til den edge med højst sluttid fra tidligere (a i vores tildfælde). Herfra noterer vi hver visited vertex, og når en vertex peger på en sort vertex i en recursion, putter vi alle vertices, som ikke er i en ssc sammen i en ny ssc.

Vi ser, at når vi har besøgt $\{a, b, c\}$, peger c på b, som er sort, hvorefter a peger på c, som er sort. Vi noterer derfor $\{a, b, c\}$ som en ssc. Vi fortsætter med samme metode, så DFS nu køres på e, da e nu er den med højst sluttid, som ikke er blevet kørt endnu. Efter vi har kørt $DFS(G^T)$ ender vi med følgende subsets:

$$\{a, b, c\}\{e, d\}\{f, g, h\}$$

1.b

2

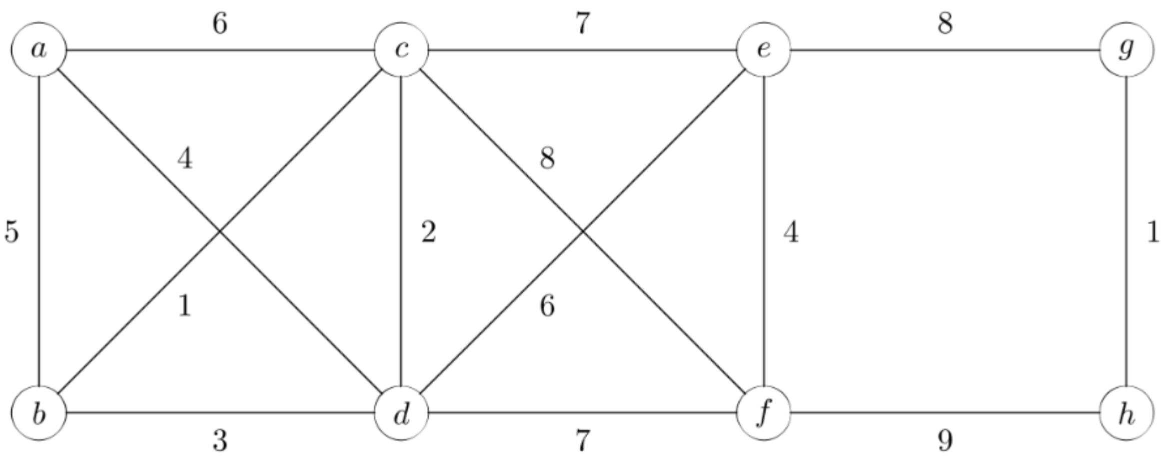
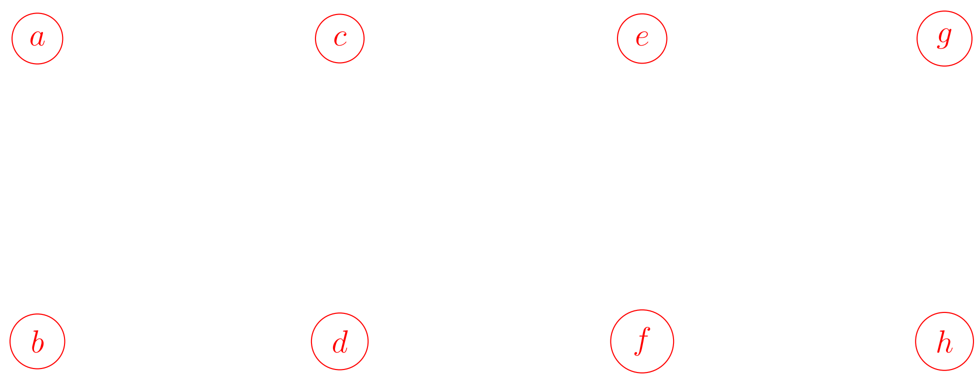


Figure 3: Undirected graph for which to compute minimum spanning tree in Problem 2a.

2.a

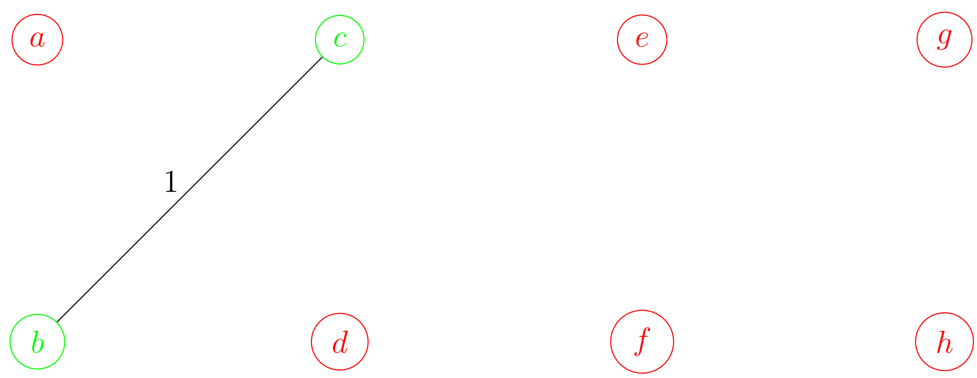
Vi har følgende vertices:



som vi kan skrive op som subsets med alle connectede vertices. I begyndelsen har vi bare:

$$\{a\}\{b\}\{c\}\{d\}\{e\}\{f\}\{g\}\{h\}$$

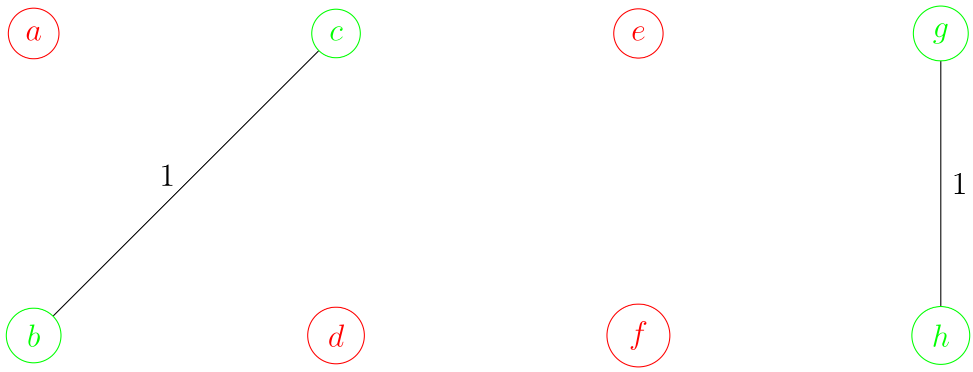
Vi ser, at $(b, c) \in E$ har lavest vægt, sammen med $(g, h) \in E$. I lexicographic order vælger vi først $(b, c) \in E$.
$(b, c) \in E$ connecter kun ikke-connetede vertices sammen, da disse er de eneste elementer i deres subsets, så vi connecter dem.



Nu har vi nogle nye connectede vertices, så vi samler deres subsets:

$$\{b, c\}\{a\}\{d\}\{e\}\{f\}\{g\}\{h\}$$

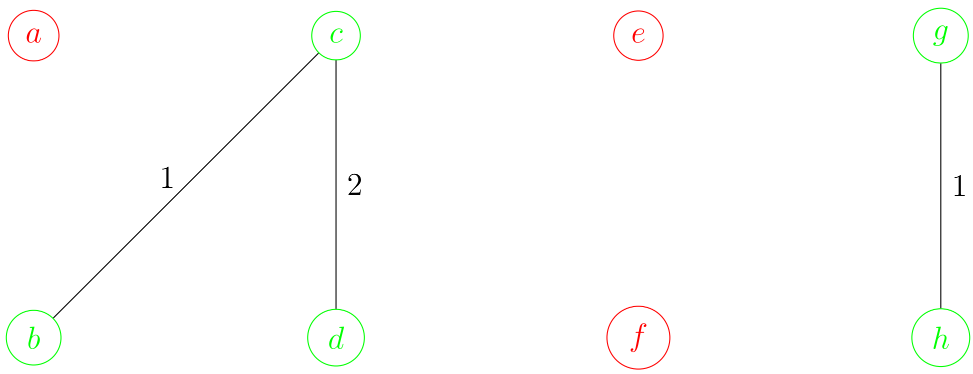
Vi kigger derefter på $(g, h) \in E$, da denne edge er den næste i acending order af weight.
$(g, h) \in E$ connecter kun ikke-connetede vertices sammen, da disse vertices er i forskellige subsets, så vi connecter dem.



Nu har vi nogle nye connectede vertices, så vi samler deres subsets:

$$\{b, c\}\{g, h\}\{a\}\{d\}\{e\}\{f\}$$

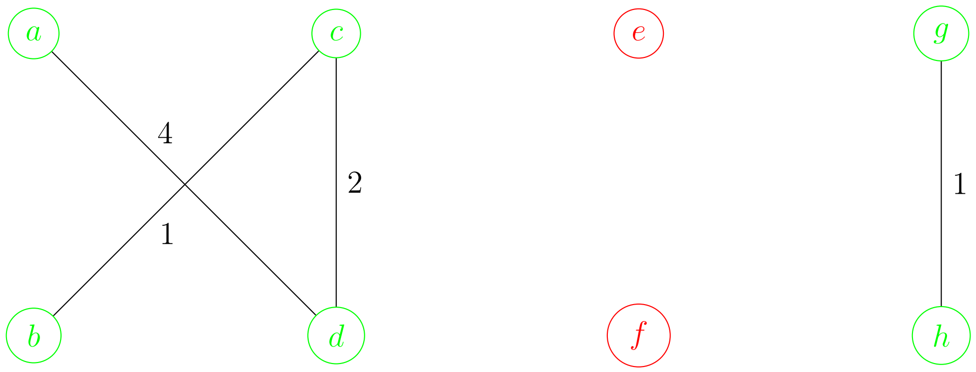
Vi kigger derefter på $(c, d) \in E$, da denne edge er den næste i acending order af weight. $(c, d) \in E$ connecter d, som ikke er connected gennem andre edges, da disse vertices er i forskellige subsets, så vi connecter dem.



Nu har vi nogle nye connectede vertices, så vi samler deres subsets:

$$\{b, c, d\}\{g, h\}\{a\}\{e\}\{f\}$$

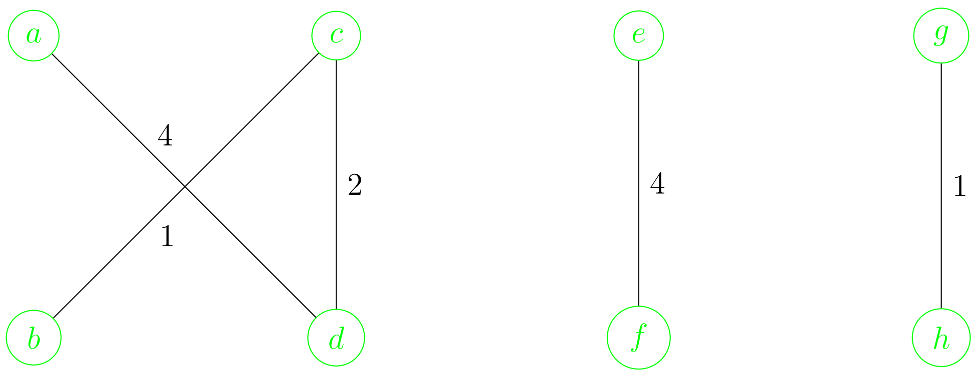
Vi kigger derefter på $(b, d) \in E$, da denne edge er den næste i acending order af weight. b og d, er allerede connected gennem andre edges, da de er i samme subset, så vi ignorerer den edge.
Vi kigger derefter på $(a, d) \in E$, da denne edge er den næste i acending order af weight. $(a, d) \in E$ connecter a, som ikke er connected gennem andre edges, da disse vertices er i forskellige subsets, så vi connecter dem.



Nu har vi nogle nye connectede vertices, så vi samler deres subsets:

$$\{a, b, c, d\}\{g, h\}\{e\}\{f\}$$

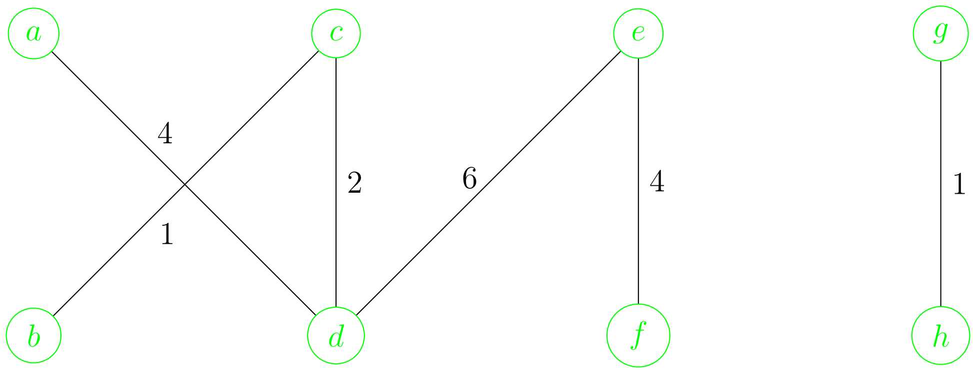
Vi kigger derefter på $(e, f) \in E$, da denne edge er den næste i acending order af weight. $(e, f) \in E$ connecter kun ikke-connetede vertices sammen, da disse vertices er i forskellige subsets, så vi connecter dem.



Nu har vi nogle nye connectede vertices, så vi samler deres subsets:

$$\{a, b, c, d\}\{e, f\}\{g, h\}$$

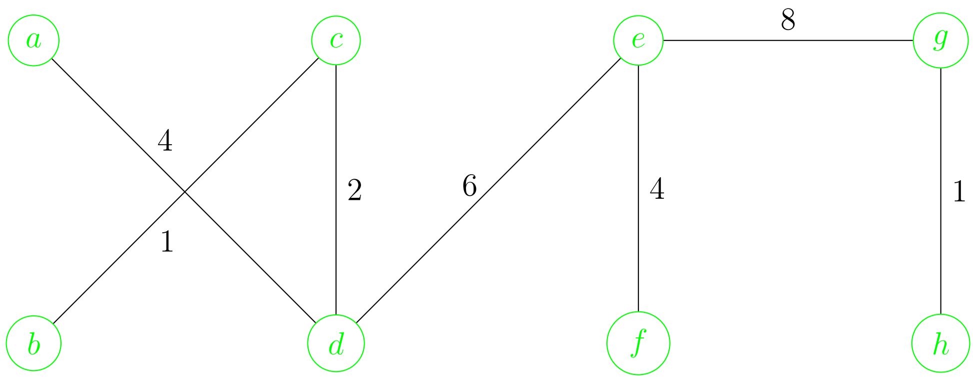
Vi kigger derefter på $(a, b) \in E$, da denne edge er den næste i acending order af weight. a og b, er allerede connected gennem andre edges, da de er i samme subset, så vi ignorerer den edge.
Vi kigger derefter på $(a, c) \in E$, da denne edge er den næste i acending order af weight. a og c, er allerede connected gennem andre edges, da de er i samme subset, så vi ignorerer den edge.
Vi kigger derefter på $(d, e) \in E$, da denne edge er den næste i acending order af weight. $(d, e) \in E$ connecter både e og f, som ikke er connected gennem andre edges, da disse vertices er i forskellige subsets, så vi connecter dem.



Nu har vi nogle nye connectede vertices, så vi samler deres subsets:

$$\{a, b, c, d, e, f\}\{g, h\}$$

Vi kigger derefter på $(c, e) \in E$, da denne edge er den næste i acending order af weight. c og e, er allerede connected gennem andre edges, da de er i samme subset, så vi ignorerer den edge.
Vi kigger derefter på $(d, f) \in E$, da denne edge er den næste i acending order af weight. d og f, er allerede connected gennem andre edges, da de er i samme subset, så vi ignorerer den edge.
Vi kigger derefter på $(c, f) \in E$, da denne edge er den næste i acending order af weight. c og f, er allerede connected gennem andre edges, da de er i samme subset, så vi ignorerer den edge.
Vi kigger derefter på $(e, g) \in E$, da denne edge er den næste i acending order af weight. $(e, g) \in E$ connecter både e og g, som ikke er connected gennem andre edges, da disse vertices er i forskellige subsets, så vi connecter dem.



Nu har vi nogle nye connectede vertices, så vi samler deres subsets:

$$\{a, b, c, d, e, f, g, h\}$$

Vi ser, at vi har 7 edges og 8 vertices. Vi er dermed færdige, da antallet af edges svarer til antallet af vertices - 1, og alle vores vertices er i samme set, hvilket vil sige, at de er connected.

2.b

Proof by contradiction:
For et minimum spanning tree gælder det:

$$\forall cut \in G(V,E) : light\ edge(cut) \in MST(G)$$

Ifølge informationen i opgave, har vi en edge $(u,v) \in E$, hvor u,v er unikke vertices og $w(u,v) \in E$ er strængt mindre end weight'en af alle andre edges connected til v.
Antag, at

$$((u,v) \in E) \notin MST(G)$$

så må det betyde

$$light\ edge(cut(v)) \neq (u,v) \in E$$

og

$$\exists((k,j) \in E) \in G \ : \ (w((k,j) \in E) \leq w(u,v) \in E) \in G$$

men ifølge vores information fra opgavebeskrivelsen kan dette ikke være tilfældet, da $w(u,v) \in E$ er strængt mindre end alle andre edges connected til v. Hvis $(u,v) \notin E$ er en del af $MST(G)$ går dette imod vores definition af et minimum spanning tree, og vi ender dermed en crontradiction $\square$

Vi er nu færdige, da vi ved, at $(u,v) \in E$ altid er med i $MST(G)$.

2.c

Proof by contradiction:
Præcist som før, gælder der for ethvert minimum spanning tree:

$$\forall cut \in G(V,E) : light\ edge(cut) \in MST(G)$$

Ifølge informationen i opgave, har vi en edge $(u,v) \in E$, hvor u,v er unikke vertices og $w(u,v) \in E$ er strængt større end weight'en af alle andre edges connected til v.
Antag, at

$$((u,v) \in E) \in MST(G)$$

så må det betyde

$$light\ edge(cut(v)) = (u,v) \in E$$

og

$$\nexists((k,j) \in E) \in G \ : \ (w((k,j) \in E) \leq w(u,v) \in E) \in G$$

men ifølge vores information fra opgavebeskrivelsen kan dette ikke være tilfældet, da v har andre unikke naboer, og $w(u,v) \in E$ er strængt større end alle andre edges connected til v. Hvis $(u,v) \in E$ er en del af $MST(G)$ går dette imod vores definition af et minimum spanning tree, og vi ender dermed en crontradiction $\square$

Vi er nu færdige, da vi ved, at $(u,v) \in E$ aldrig er en del af $MST(G)$.

2.d

3

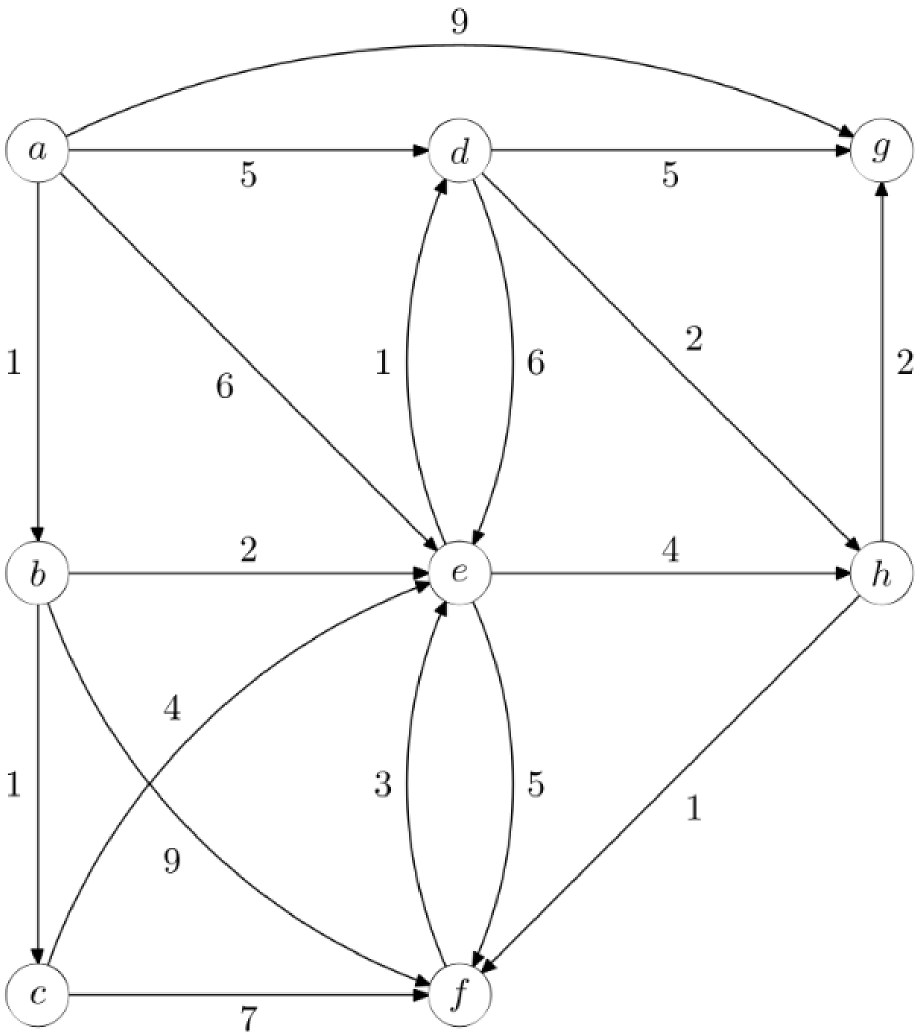


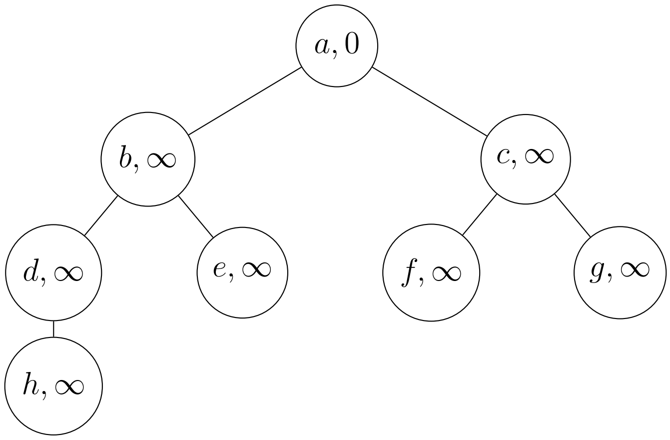
Figure 4: Directed graph Dijkstra's algorithm in Problem 3a.

3.a

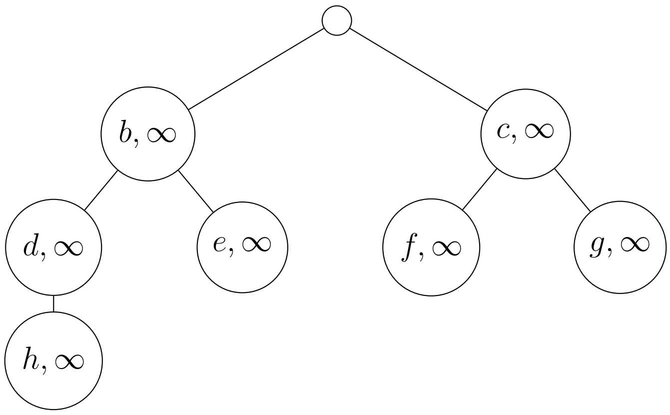
Jeg opskriver vores weighted directed graph som adjacency list representation $(vertex, weight)$:

$$
\begin{aligned}
&(a, 0) \to ((b, 1), (d, 5), (e, 6), (g, 9)) \\
&(b, 0) \to ((c, 1), (e, 2), (f, 9)) \\
&(c, 0) \to ((e, 4), (f, 7)) \\
&(d, 0) \to ((e, 6), (g, 5), (h, 2)) \\
&(e, 0) \to ((d, 1)(f, 5), (h, 4)) \\
&(f, 0) \to (e, 3) \\
&(g, 0) \to () \\
&(h, 0) \to (f, 1)
\end{aligned}
$$

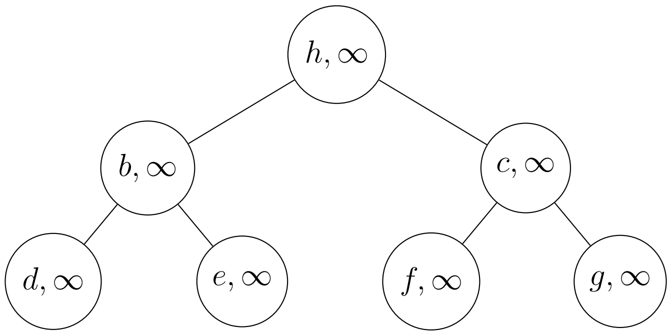
Vi tilføjer alle vores vertices til vores priority queue, som vi laver som en min-heap:



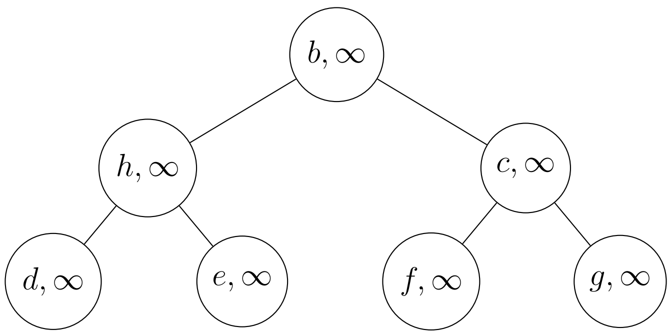
Vi extracter a:

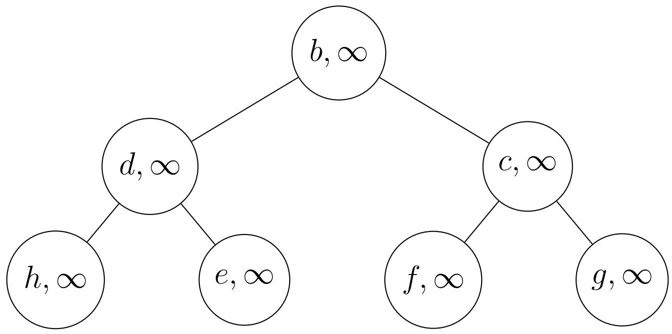


og flytter sidste element forest:



vi kører min-heapify:

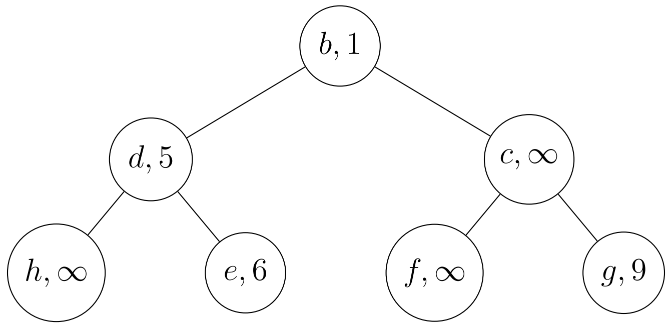




Vi sætter vores extracted vertex i et set $S = \{(a, 0)\}$ Udregning af distancer for $a's$ naboer:

$$[(b, 1), (d, 5), (e, 6), (g, 9)]$$

Vi opdaterer nu alle værdier der er lavere i vores priority queue:



a er nu besøgt.

Vi ser, at b er den ubesøgte vertex med kortest afstand fra a.
Udregning af nye distancer for $b's$ naboer:

$$[(c, 1+1), (e, 1+2), (f, 1+9)] = [(c, 2), (e, 3), (f, 10)]$$

Vi opdaterer nu de laveste værdier i grafen:

$$[(A, 0), (B, 1), (c, 2), (e, 3), (d, 5), (g, 9), (f, 10), (h, \infty)]$$

b er nu besøgt, så det har jeg markeret ved at skrive B i stedet.

Vi ser, at c er den ubesøgte vertex med kortest afstand fra a.
Udregning af nye distancer for $c's$ naboer:

$$[(e, 2+4), (f, 2+7)] = [(e, 6), (f, 9)]$$

Vi opdaterer nu de laveste værdier i grafen:

$$[(A, 0), (B, 1), (C, 2), (e, 3), (d, 5), (f, 9), (g, 9), (h, \infty)]$$

c er nu besøgt, så det har jeg markeret ved at skrive C i stedet.

Vi ser, at e er den ubesøgte vertex med kortest afstand fra a.
Udregning af nye distancer for $e's$ naboer:

$$[(d, 3+1), (f, 3+5), (h, 3+4)] = [(d, 4), (f, 8), (h, 7)]$$

Vi opdaterer nu de laveste værdier i grafen:

$$[(A,0),(B,1),(C,2),(E,3),(d,4),(h,7),(f,8),(g,9)]$$

e er nu besøgt, så det har jeg markeret ved at skrive E i stedet.

Vi ser, at d er den ubesøgte vertex med kortest afstand fra a.
Udregning af nye distancer for $d's$ naboer:

$$[(g,5+5),(h,5+5),(e,5+6)] = [(g,10),(h,10),(e,11)]$$

Vi opdaterer nu de laveste værdier i grafen, men ser, at der ikke nogen kortere afstande:

$$[(A,0),(B,1),(C,2),(E,3),(D,4),(h,7),(f,8),(g,9)]$$

d er nu besøgt, så det har jeg markeret ved at skrive D i stedet.

Vi ser, at h er den ubesøgte vertex med kortest afstand fra a.
Udregning af nye distancer for $h's$ naboer:

$$[(f,7+1)] = [(f,8)]$$

Vi opdaterer nu de laveste værdier i grafen, men ser, at der ikke nogen kortere afstande:

$$[(A,0),(B,1),(C,2),(E,3),(D,4),(H,7),(f,8),(g,9)]$$

h er nu besøgt, så det har jeg markeret ved at skrive H i stedet.

Vi ser, at f er den ubesøgte vertex med kortest afstand fra a.
Udregning af nye distancer for $f's$ naboer:

$$[(e,8+3)] = [(e,11)]$$

Vi opdaterer nu de laveste værdier i grafen, men ser, at der ikke nogen kortere afstande:

$$[(A,0),(B,1),(C,2),(E,3),(D,4),(H,7),(F,8),(g,9)]$$

f er nu besøgt, så det har jeg markeret ved at skrive F i stedet.

Vi ser, at g er den sidste ubesøgte vertex, og at g ikke har nogen naboer.

$$[(A,0),(B,1),(C,2),(E,3),(D,4),(H,7),(F,8),(G,9)]$$

Vi markerer g som besøgt med G.

3.b

3.c

3.d