Sammenhængskomponenter i grafer

Ækvivalensrelationer

$S = \{1, 2, 3, 4\}$

$R = \{(1,2)\ (3,3), (3,4), (4,3)\}$

$1 \sim 2$

Repetition:

En relation R på en mængde S er en delmængde af $S \times S$. Når $(x, y) \in R$ siges x at stå i relation til y. Ofte skrives $x \sim y$, og relationen selv betegnes "$\sim$".

Ækvivalensrelationer

Repetition:

En relation R på en mængde $\overset{V}{S}$ er en delmængde af $\overbrace{S \times S}^{E}$. Når $(x, y) \in R$ siges x at stå i relation til y. Ofte skrives $x \sim y$, og relationen selv betegnes "$\sim$".

Relation kaldes en ækvivalensrelation hvis der for alle $x, y, z \in S$ gælder:

- $x \sim x$. $\quad (x,x) \in R$
- $x \sim y \Rightarrow y \sim x$. $\quad (x,y) \in R \Rightarrow (y,x) \in R$
- $x \sim y \wedge y \sim z \Rightarrow x \sim z$.

$$\left.\begin{array}{l}(x,y) \in R \\ (y,z) \in R\end{array}\right\} \Rightarrow (x,z) \in R$$

Ækvivalensrelationer

Repetition:

En relation R på en mængde S er en delmængde af $S \times S$. Når $(x, y) \in R$ siges x at stå i relation til y. Ofte skrives $x \sim y$, og relationen selv betegnes "$\sim$".

Relation kaldes en ækvivalensrelation hvis der for alle $x, y, z \in S$ gælder:

$[x] = \{y \mid (x,y) \in R\}$

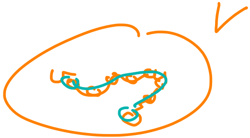
- $x \sim x$.
- $x \sim y \Rightarrow y \sim x$. $[y]$
- $x \sim y \wedge y \sim z \Rightarrow x \sim z$.

$x \sim y$

S

En ækvivalensrelation deler S i disjunkte delmængder (hver bestående af elementer som er i relation til hinanden, men ikke til andre elementer), og kaldes derfor også en partition.

Ækvivalensrelationer på en grafs knuder



Uorienterede grafer:

For $v, u \in V$:

$$v \sim u \boxed{\Leftrightarrow} \text{ der er en (uorienteret) sti mellem } u \text{ og } v$$

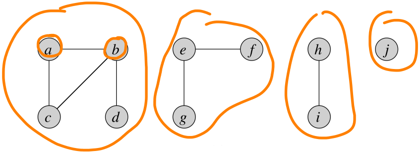
Ækvivalensrelationer på en grafs knuder

Uorienterede grafer:

For $v, u \in V$:

$$v \sim u \Leftrightarrow \text{der er en (uorienteret) sti mellem } u \text{ og } v$$

Giver en partition af grafens knuder V:



De kaldes grafens sammenhængskomponenter (CC'er).

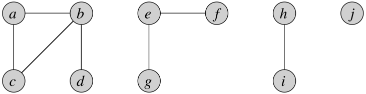
Ækvivalensrelationer på en grafs knuder

Uorienterede grafer:

For $v, u \in V$:

$$v \sim u \Leftrightarrow \text{ der er en (uorienteret) sti mellem } u \text{ og } v$$

Giver en partition af grafens knuder V:

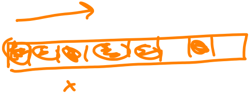


De kaldes grafens sammenhængskomponenter (CC'er).

Finde dem?

Ækvivalensrelationer på en grafs knuder

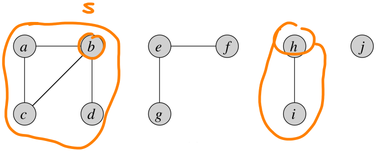
Uorienterede grafer:



For $v, u \in V$:

$$v \sim u \Leftrightarrow \text{ der er en (uorienteret) sti mellem } u \text{ og } v$$

Giver en partition af grafens knuder V:



De kaldes grafens sammenhængskomponenter (CC'er).

Finde dem? Via DFS eller BFS (med GLOBAL ydre loop). Knuderne i hvert genereret træ er en CC, da alle knuder, som kan nås fra en startknude s, vil blive nået (jvf. sætning om GENERICGRAPHTRAVERSAL1(s)), og ingen andre knuder kan nås .

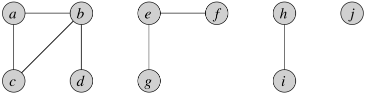
Ækvivalensrelationer på en grafs knuder

Uorienterede grafer:

For $v, u \in V$:

$$v \sim u \Leftrightarrow \text{ der er en (uorienteret) sti mellem } u \text{ og } v$$

Giver en partition af grafens knuder V:



De kaldes grafens sammenhængskomponenter (CC'er).

Finde dem? Via DFS eller BFS (med GLOBAL ydre loop). Knuderne i hvert genereret træ er en CC, da alle knuder, som kan nås fra en startknude s, vil blive nået (jvf. sætning om GENERICGRAPHTRAVERSAL1(s)), og ingen andre knuder kan nås .

Tid?

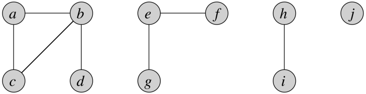
Ækvivalensrelationer på en grafs knuder

Uorienterede grafer:

For $v, u \in V$:

$$v \sim u \Leftrightarrow \text{ der er en (uorienteret) sti mellem } u \text{ og } v$$

Giver en partition af grafens knuder V:

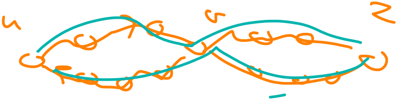


De kaldes grafens sammenhængskomponenter (CC'er).

Finde dem? Via DFS eller BFS (med GLOBAL ydre loop). Knuderne i hvert genereret træ er en CC, da alle knuder, som kan nås fra en startknude s, vil blive nået (jvf. sætning om GENERICGRAPHTRAVERSAL1(s)), og ingen andre knuder kan nås .

Tid? $O(n + m)$.

Ækvivalensrelationer på en grafs knuder



Orienterede grafer:

For $v, u \in V$:

$$v \sim u \quad \Leftrightarrow \quad \begin{array}{c} \text{der er en (orienteret) sti fra } u \text{ til } v \\ \text{OG} \\ \text{der er en (orienteret) sti fra } v \text{ til } u \end{array}$$

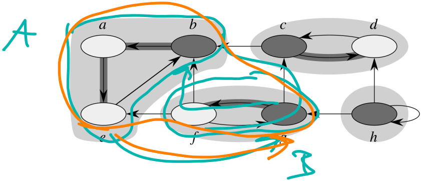
Ækvivalensrelationer på en grafs knuder

Orienterede grafer:

For $v, u \in V$:

$$v \sim u \quad \Leftrightarrow \quad \begin{array}{c}\text{der er en (orienteret) sti fra } u \text{ til } v\\ \text{OG}\\ \text{der er en (orienteret) sti fra } v \text{ til } u\end{array}$$

Giver en partition af grafens knuder V:



De kaldes grafens stærke sammenhængskomponenter (SCC'er).

Finde dem?

Finde stærke sammenhængskomponenter

Algoritme:

$u.f$

SCC(G)

call DFS(G) to compute finishing times $u.f$ for all u → Top. Sort.

compute G^{T}

call DFS(G^{T}), but in the main loop, consider vertices in order of decreasing $u.f$

(as computed in first DFS)

output the vertices in each tree of the depth-first forest formed in second DFS

as a separate SCC

Her er G^T grafen G med alle kanter vendt.

Finde stærke sammenhængskomponenter

Algoritme:

SCC(G)

call DFS(G) to compute finishing times $u.f$ for all u
compute G^{T}
call DFS(G^{T}), but in the main loop, consider vertices in order of decreasing $u.f$
(as computed in first DFS)
output the vertices in each tree of the depth-first forest formed in second DFS
as a separate SCC

Her er G^T grafen G med alle kanter vendt.

Tid?

Finde stærke sammenhængskomponenter

Algoritme:

SCC(G)
call DFS(G) to compute finishing times $u.f$ for all u
compute G^{T}
call DFS(G^{T}), but in the main loop, consider vertices in order of decreasing $u.f$
(as computed in first DFS)
output the vertices in each tree of the depth-first forest formed in second DFS
as a separate SCC

Her er G^T grafen G med alle kanter vendt.

Tid? $O(n+m)$.

Finde stærke sammenhængskomponenter

Algoritme:

SCC(G)

call DFS(G) to compute finishing times $u.f$ for all u
compute G^{T}
call DFS(G^{T}), but in the main loop, consider vertices in order of decreasing $u.f$
(as computed in first DFS)
output the vertices in each tree of the depth-first forest formed in second DFS
as a separate SCC

Her er G^T grafen G med alle kanter vendt.

Tid? $O(n+m)$.

Korrekthed?

Finde stærke sammenhængskomponenter

Algoritme:

SCC(G)
call DFS(G) to compute finishing times $u.f$ for all u
compute G^{T}
call DFS(G^{T}), but in the main loop, consider vertices in order of decreasing $u.f$
(as computed in first DFS)
output the vertices in each tree of the depth-first forest formed in second DFS
as a separate SCC

Her er G^T grafen G med alle kanter vendt.

Tid? $O(n+m)$.

Korrekthed? De næste sider...

Korrekthed af SCC algoritme

Sætning:

> Algoritmen SCC ovenfor er korrekt, dvs. træerne returneret fra det andet kald til DFS repræsenterer præcis G's SCC'er.

Korrekthed af SCC algoritme

Sætning:

Algoritmen SCC ovenfor er korrekt, dvs. træerne returneret fra det andet kald til DFS repræsenterer præcis G's SCC'er.

Bevis:

Korrekthed af SCC algoritme

Sætning:

Algoritmen SCC ovenfor er korrekt, dvs. træerne returneret fra det andet kald til DFS repræsenterer præcis G's SCC'er.

Bevis:

Bemærk først at:

$$\text{Der er en sti } u \rightsquigarrow v \text{ i } G \quad \Leftrightarrow \quad \text{Der er en sti } v \rightsquigarrow u \text{ i } G^T$$

Korrekthed af SCC algoritme

Sætning:

> Algoritmen SCC ovenfor er korrekt, dvs. træerne returneret fra det andet kald til DFS repræsenterer præcis G's SCC'er.

Bevis:

Bemærk først at:

$$\text{Der er en sti } u \rightsquigarrow v \text{ i } G \quad \Leftrightarrow \quad \text{Der er en sti } v \rightsquigarrow u \text{ i } G^T$$

Heraf følger:

$$u \text{ og } v \text{ i samme SCC i } G \quad \Leftrightarrow \quad u \text{ og } v \text{ i samme SCC i } G^T$$

Korrekthed af SCC algoritme

Sætning:

> Algoritmen SCC ovenfor er korrekt, dvs. træerne returneret fra det andet kald til DFS repræsenterer præcis G's SCC'er.

Bevis:

Bemærk først at:

$$\text{Der er en sti } u \rightsquigarrow v \text{ i } G \quad \Leftrightarrow \quad \text{Der er en sti } v \rightsquigarrow u \text{ i } G^T$$

Heraf følger:

$$u \text{ og } v \text{ i samme SCC i } G \quad \Leftrightarrow \quad u \text{ og } v \text{ i samme SCC i } G^T$$

Så G og G^T har de samme SCC'er.

Korrekthed af SCC algoritme

Hvid-sti lemma:

Hvis findes en sti af hvide knuder (inkl. w) fra u til w når u kommer på stakken (dvs. til tid $u.d$), da kommer w på stakken inden u kommer af.

Korrekthed af SCC algoritme

Hvid-sti lemma:

> Hvis findes en sti af hvide knuder (inkl. w) fra u til w når u kommer på stakken (dvs. til tid $u.d$), da kommer w på stakken inden u kommer af.

Bevis (lemma):

Antag at der findes knuder på stien som ikke kommer på stakken før u kommer af, og lad y være den første (set fra u). Dvs. y's forgænger x kom på stakken inden u kom af. Men så kan u ikke komme af inden x gør, og x kan ikke komme af før y har været på stakken (da y er nabo til x vil y blive opdaget/sat på stakken af x, hvis det ikke er sket tidligere). Modstrid. $\square$

Korrekthed af SCC algoritme

Hvid-sti lemma:

> Hvis findes en sti af hvide knuder (inkl. w) fra u til w når u kommer på stakken (dvs. til tid $u.d$), da kommer w på stakken inden u kommer af.

Bevis (lemma):

Antag at der findes knuder på stien som ikke kommer på stakken før u kommer af, og lad y være den første (set fra u). Dvs. y's forgænger x kom på stakken inden u kom af. Men så kan u ikke komme af inden x gør, og x kan ikke komme af før y har været på stakken (da y er nabo til x vil y blive opdaget/sat på stakken af x, hvis det ikke er sket tidligere). Modstrid. $\square$

Da w kommer på stakken efter at u kommer på stakken (w var hvid til tid $u.d$) fås:

Korollar:

> I situationen ovenfor haves $u.d < w.d < w.f < u.f$.

Korrekthed af SCC algoritme

For en knudemængde $C \subseteq V$ defineres $f(C) = \max_{v \in C} v.f$ (hvor f angiver tiden fra første DFS i SCC-algoritmen).

Lemma:

> Hvis C, C' er to forskellige SCC'er i G, og (x, y) er en kant i G med $x \in C$ og $y \in C'$, da gælder $f(C) > f(C')$.

Korrekthed af SCC algoritme

For en knudemængde $C \subseteq V$ defineres $f(C) = \max_{v \in C} v.f$ (hvor f angiver tiden fra første DFS i SCC-algoritmen).

Lemma:

> Hvis C, C' er to forskellige SCC'er i G, og (x, y) er en kant i G med $x \in C$ og $y \in C'$, da gælder $f(C) > f(C')$.

Da G^T er G med alle kanter vendt, og da SCC'erne er de samme i G^T og G, haves:

Korollar:

> Hvis C, C' er to forskellige SCC'er i G^T, og (x, y) er en kant i G^T med $x \in C$ og $y \in C'$, da gælder $f(C) < f(C')$.

Korrekthed af SCC algoritme

Bevis (lemma):

Lad u være den første knude i $C \cup C'$ som opdages.

Korrekthed af SCC algoritme

Bevis (lemma):

Lad u være den første knude i $C \cup C'$ som opdages.

Case 1: $u \in C$. Her er der en sti fra u til w for alle $w \in C \cup C'$, så udsagnet følger af korollar til hvid-sti lemma.

Korrekthed af SCC algoritme

Bevis (lemma):

Lad u være den første knude i $C \cup C'$ som opdages.

Case 1: $u \in C$. Her er der en sti fra u til w for alle $w \in C \cup C'$, så udsagnet følger af korollar til hvid-sti lemma.

Case 2: $u \in C'$. Her er der en sti fra u til w for alle $w \in C'$, så af samme korollar haves $f(C') = u.f$.

Korrekthed af SCC algoritme

Bevis (lemma):

Lad u være den første knude i $C \cup C'$ som opdages.

Case 1: $u \in C$. Her er der en sti fra u til w for alle $w \in C \cup C'$, så udsagnet følger af korollar til hvid-sti lemma.

Case 2: $u \in C'$. Her er der en sti fra u til w for alle $w \in C'$, så af samme korollar haves $f(C') = u.f$.

Hvis der fandtes en knude $v \in C$ med $v.d < u.f$, ville der være et tidpunkt hvor v og u var på stakken samtidig (da $u.d < v.d$, eftersom u var den først opdagede i $C \cup C'$). Da det er en invariant under DFS at der i grafen findes en sti mellem knuderne på stakken (fra tidligere til senere push'ede knuder), ville dette betyde en sti fra $u \in C'$ til $v \in C$. Sammen med kanten (x, y) ville dette medføre at alle knuder i $C \cup C'$ var i samme SCC, i modstrid med at C og C' er to forskellige SCC'er.

Korrekthed af SCC algoritme

Bevis (lemma):

Lad u være den første knude i $C \cup C'$ som opdages.

Case 1: $u \in C$. Her er der en sti fra u til w for alle $w \in C \cup C'$, så udsagnet følger af korollar til hvid-sti lemma.

Case 2: $u \in C'$. Her er der en sti fra u til w for alle $w \in C'$, så af samme korollar haves $f(C') = u.f$.

Hvis der fandtes en knude $v \in C$ med $v.d < u.f$, ville der være et tidpunkt hvor v og u var på stakken samtidig (da $u.d < v.d$, eftersom u var den først opdagede i $C \cup C'$). Da det er en invariant under DFS at der i grafen findes en sti mellem knuderne på stakken (fra tidligere til senere push'ede knuder), ville dette betyde en sti fra $u \in C'$ til $v \in C$. Sammen med kanten (x, y) ville dette medføre at alle knuder i $C \cup C'$ var i samme SCC, i modstrid med at C og C' er to forskellige SCC'er.

Derfor haves $v.d > u.f$ for alle $v \in C$, så $f(C) > u.f = f(C')$. $\square$

Korrekthed af SCC algoritme

Vi viser nu sætningen om korrekthed af SCC-algoritmen ved at vise at for alle k gælder:

> Knuderne i de k første træer genereret under den anden DFS i SCC-algoritmen udgør hver især en SCC i G^T.

Da SCC'erne i G og G^T er de samme, og da alle knuder i grafen er i et af træerne, viser dette korrektheden.

Korrekthed af SCC algoritme

Vi viser nu sætningen om korrekthed af SCC-algoritmen ved at vise at for alle k gælder:

> Knuderne i de k første træer genereret under den anden DFS i SCC-algoritmen udgør hver især en SCC i G^T.

Da SCC'erne i G og G^T er de samme, og da alle knuder i grafen er i et af træerne, viser dette korrektheden.

Vi viser ovenstående udsagn via induktion på k.

Korrekthed af SCC algoritme

Vi viser nu sætningen om korrekthed af SCC-algoritmen ved at vise at for alle k gælder:

> Knuderne i de k første træer genereret under den anden DFS i SCC-algoritmen udgør hver især en SCC i G^T.

Da SCC'erne i G og G^T er de samme, og da alle knuder i grafen er i et af træerne, viser dette korrektheden.

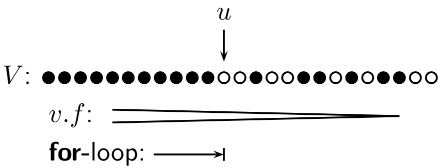
Vi viser ovenstående udsagn via induktion på k.

Skridt: Antag sandt for k, vis sandt for $k+1$.

Det $(k+1)$'te træ genereres ved det $(k+1)$'te kald til $\text{DFS-V}\textsc{isit}$ i **for**-løkken i det ydre loop i DFS. Lad u være knude, der kaldes på.

Hvis vi stiller knuderne op i **for**-løkkens rækkefølge (efter aftagende $v.f$-værdi), ser situationen sådan ud på tidspunktet for dette kald:

Korrekthed af SCC algoritme



Sorte knuder er de indtil nu opdagede under DFS, hvide er de uopdagede.

Lad C være SCC'en indeholdende u, og lad T være træet genereret af kaldet på u. Af induktionsantagelsen udgør de sorte knuder præcis k af grafens SCC'er. Derfor må alle andre SCC'er ligge inden i de hvide knuder, og C er en af disse (da u er hvid).

Eftersom der ved starten af kaldet er en hvid sti fra u til alle $w \in C$, giver hvid-sti lemma at $C \subseteq T$.

Korrekthed af SCC algoritme

Lad C' være en vilkårlig hvid SCC forskellig fra C. Pga. **for**-løkkens rækkefølge ses $u.f = f(C) > f(C')$, så af korollar til lemma ovenfor fås at ingen kant i G^T kan gå fra C til C'. Da DFS-VISIT ikke besøger de sorte knuder, kan den derfor ikke forlade C. Heraf ses $T \subseteq C$.

Vi har i alt vist $T = C$, hvilket viser udsagnet for $k + 1$.

Korrekthed af SCC algoritme

Lad C' være en vilkårlig hvid SCC forskellig fra C. Pga. **for**-løkkens rækkefølge ses $u.f = f(C) > f(C')$, så af korollar til lemma ovenfor fås at ingen kant i G^T kan gå fra C til C'. Da DFS-VISIT ikke besøger de sorte knuder, kan den derfor ikke forlade C. Heraf ses $T \subseteq C$.

Vi har i alt vist $T = C$, hvilket viser udsagnet for $k + 1$.

Basis: Samme argument, blot lidt simplere (der er ingen sorte knuder, og u er første knude i rækkefølgen), viser udsagnet for $k = 1$. □