Sammenhængskomponenter i grafer

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 "$\sim$".

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 "$\sim$".

En relation er en ækvivalensrelation hvis der for alle $x, y, z \in S$ gælder:

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

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 "$\sim$".

En relation er en ækvivalensrelation hvis der for alle $x, y, z \in S$ gælder:

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

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

Ækvivalensrelationer på en grafs knuder

Uorienterede grafer:

For $v, u \in V$:

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

Ækvivalensrelationer på en grafs knuder

Uorienterede grafer:

For $v, u \in V$:

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

Ses let at være en ækvivalensrelation, så giver en partition af grafens knuder V.

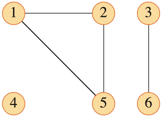
Ækvivalensrelationer på en grafs knuder

Uorienterede grafer:

For $v, u \in V$:

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

Ses let at være en ækvivalensrelation, så giver en partition af grafens knuder V. Her er en graf med en partition af størrelse tre:



Partitions mængder kaldes grafens sammenhængskomponenter (CC'er).

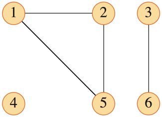
Ækvivalensrelationer på en grafs knuder

Uorienterede grafer:

For $v, u \in V$:

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

Ses let at være en ækvivalensrelation, så giver en partition af grafens knuder V. Her er en graf med en partition af størrelse tre:



Partitions mængder kaldes grafens sammenhængskomponenter (CC'er).

Finde dem?

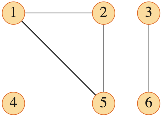
Ækvivalensrelationer på en grafs knuder

Uorienterede grafer:

For $v, u \in V$:

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

Ses let at være en ækvivalensrelation, så giver en partition af grafens knuder V. Her er en graf med en partition af størrelse tre:



Partitions mængder kaldes grafens sammenhængskomponenter (CC'er).

Finde dem? Via DFS eller BFS med $\textsc{Global}$ ydre loop. Hvert kald fra ydre loop opdager præcis knuderne i én sammenhængskomponent (fra tidligere sætninger om $\textsc{GenericGraphTraversal}2/3(s)$ ses, at et kald opdager præcis de knuder, som kan nås fra s via en sti af hvide knuder på tidspunktet for kaldet).

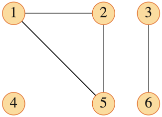
Ækvivalensrelationer på en grafs knuder

Uorienterede grafer:

For $v, u \in V$:

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

Ses let at være en ækvivalensrelation, så giver en partition af grafens knuder V. Her er en graf med en partition af størrelse tre:



Partitions mængder kaldes grafens sammenhængskomponenter (CC'er).

Finde dem? Via DFS eller BFS med $\textsc{Global}$ ydre loop. Hvert kald fra ydre loop opdager præcis knuderne i én sammenhængskomponent (fra tidligere sætninger om $\textsc{GenericGraphTraversal}2/3(s)$ ses, at et kald opdager præcis de knuder, som kan nås fra s via en sti af hvide knuder på tidspunktet for kaldet). Tid?

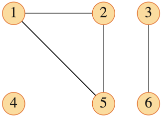
Ækvivalensrelationer på en grafs knuder

Uorienterede grafer:

For $v, u \in V$:

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

Ses let at være en ækvivalensrelation, så giver en partition af grafens knuder V. Her er en graf med en partition af størrelse tre:



Partitions mængder kaldes grafens sammenhængskomponenter (CC'er).

Finde dem? Via DFS eller BFS med Global ydre loop. Hvert kald fra ydre loop opdager præcis knuderne i én sammenhængskomponent (fra tidligere sætninger om GenericGraphTraversal$2/3(s)$ ses, at et kald opdager præcis de knuder, som kan nås fra s via en sti af hvide knuder på tidspunktet for kaldet). Tid? $O(n + m)$.

Anvendelser af at finde sammenhængskomponenter

Eksempler:

- Er et elektrisk netværk forbundet, så strøm kan nå alle vegne rundt?
- Find regioner i billeder med en bestemt type farve (jvf. "magic wand" tool i Photoshop), billedanalyse.
- Analyse af sociale netværk.

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 $v, u \in V$:

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

Ses let at være en ækvivalensrelation, så giver en partition af grafens knuder V.

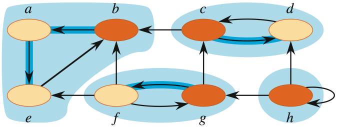
Ækvivalensrelationer på en grafs knuder

Orienterede grafer:

For $v, u \in V$:

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

Ses let at være en ækvivalensrelation, så giver en partition af grafens knuder V. Her er en graf med en partition af størrelse fire:



Partitions mængder kaldes grafens stærke sammenhængskomponenter (SCC'er).

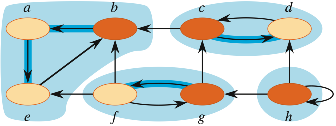
Ækvivalensrelationer på en grafs knuder

Orienterede grafer:

For $v, u \in V$:

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

Ses let at være en ækvivalensrelation, så giver en partition af grafens knuder V. Her er en graf med en partition af størrelse fire:



Partitions mængder kaldes grafens stærke sammenhængskomponenter (SCC'er). Finde dem?

Anvendelser af at finde stærke sammenhængskomponenter

Eksempler:

- For vej-netværk bør antallet af stærke sammenhængskomponenter være én, ellers opstår "black holes", hvor køretøjer fanges.
- Den ergodiske hovedsætning for Markov kæder (graf-model for mange stokastiske processer—bl.a. Google's Pagerank er baseret på begrebet Markov kæder) har to betingelser, hvoraf den ene er, at antal stærke sammenhængskomponenter i grafen er én.
- Problemet 2-CNF-SAT kan løses tid via algoritmer til at finde stærke sammenhængskomponenter. En del problemer, f.eks. indenfor VLSI (computer chip) design, data clustering og scheduling, kan modelleres som instanser af 2-CNF-SAT.

Finde stærke sammenhængskomponenter

Algoritme: [Sharir 1981, Kosaraju]

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.

Finde stærke sammenhængskomponenter

Algoritme: [Sharir 1981, Kosaraju]

```
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: [Sharir 1981, Kosaraju]

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: [Sharir 1981, Kosaraju]

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?

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: Over 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.

Bevis: Over de næste sider.

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: Over de næste sider.

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: Over de næste sider.

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

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

Lemma 1:

> 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 $f(C) > f(C')$.

Bevis for Lemma 1 gives på næste side.

Korrekthed af SCC algoritme

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

Lemma 1:

> 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 $f(C) > f(C')$.

Bevis for Lemma 1 gives på næste side.

Da G^T er G med alle kanter vendt, og da SCC'erne er de samme i G^T og G, kan lemmaet også formuleres således:

Lemma 2:

> 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 1):

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

Korrekthed af SCC algoritme

Bevis (Lemma 1):

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 hvid-sti lemma.

Korrekthed af SCC algoritme

Bevis (Lemma 1):

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 hvid-sti lemma.

Case 2: $u \in C'$. Her er der en sti fra u til w for alle $w \in C'$, så af hvid-sti lemma følger $f(C') = u.f$.

Korrekthed af SCC algoritme

Bevis (Lemma 1):

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 hvid-sti lemma.

Case 2: $u \in C'$. Her er der en sti fra u til w for alle $w \in C'$, så af hvid-sti lemma følger $f(C') = u.f$.

Antag, at der fandtes en knude $v \in C$ med $v.d < u.f$. Da $u.d < v.d$ (eftersom u var den først opdagede i $C \cup C'$) giver parentesstrukturen for d- og f-tider at $u.d < v.d < v.f < u.f$. Dvs. at v og u er på stakken samtidig, med v øverst (push'et senest). 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 1):

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 hvid-sti lemma.

Case 2: $u \in C'$. Her er der en sti fra u til w for alle $w \in C'$, så af hvid-sti lemma følger $f(C') = u.f$.

Antag, at der fandtes en knude $v \in C$ med $v.d < u.f$. Da $u.d < v.d$ (eftersom u var den først opdagede i $C \cup C'$) giver parentesstrukturen for d- og f-tider at $u.d < v.d < v.f < u.f$. Dvs. at v og u er på stakken samtidig, med v øverst (push'et senest). 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.

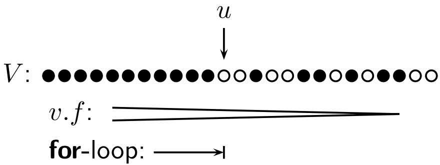
Vi viser ovenstående udsagn via induktion på k.

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

Det $(k+1)$'te træ genereres ved det $(k+1)$'te kald til DFS-VISIT 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 $w \in C$, fås $C \subseteq T$ af tidligere sætning om $\textsc{GenericGraphTraversal2}(s)$.

Korrekthed af SCC algoritme

Lad C' være en vilkårlig hvid SCC forskellig fra C. Pga. **for**-løkkens rækkefølge ses $u.f = f(C) > f(C')$.

Hvis der var en kant i G^T, som gik fra C til C', ville Lemma 2 give $f(C) < f(C')$.

Så ingen kant i G^T kan gå fra C til C'. Da $\mathrm{DFS\text{-}V\textsc{isit}}$ 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 $u.f = f(C) > f(C')$.

Hvis der var en kant i G^T, som gik fra C til C', ville Lemma 2 give $f(C) < f(C')$.

Så 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 $k = 1$. $\square$