

5. Verkkoalgoritmeja

Eräs keskeinen algoritmien suunnittelutekniikka on

”Palauta ongelma johonkin tunnettuun verkko-ongelmaan.”

Palauttaminen edellyttää usein ongelman ja algoritmin pientä modifioimista, joten on tärkeä ymmärtää hyvin perusalgoritmien keskeiset toimintaperiaatteet.

Tämän luvun jälkeen opiskelija

- tuntee yksityiskohtaisesti syvyysuuntaisen haun (DFS) ja sen analyysin suunnatussa ja suuntaamattomassa verkossa
- osaa soveltaa DFS:ää osana muiden verkko-ongelmien ratkaisua
- tuntee maksimivuo-ongelman peruspiirteet ja osaa palauttaa muita ongelmia maksimivuo-ongelmaan
- osaa ratkaista maksimivuo-ongelman Edmondsin-Karpin algoritmilla
- tuntee kehittyneempien maksimivuoalgoritmien yleisperiaatteet

Ei pidä unohtaa aiemminkaan opittuja algoritmeja (osa kurssilla Tietorakenteet): leveysuuntainen haku, Dijkstra, Floyd-Warshall, Kruskal, Prim.

5.1 Syvyysuuntainen haku

Perusalgoritmi on tuttu kurssilta Tietorakenteet.

Oletetaan, että verkko $G = (V, E)$ on annettu vieruslistoina $L[v]$, $v \in V$.

Tuloksena saadaan verkon syvyysuuntainen virittävä metsä $T \subseteq E$ ja solmujen esi- ja jälkinumerot $prenum[v]$ ja $postnum[v]$, $v \in V$.

DFS:

```
 $T := \emptyset$ ;  $pre := 1$ ;  $post := 1$   
for  $v \in V$  do  $new[v] := True$   
for  $v \in V$  do  
    if  $new[v]$  then DFS-Visit( $v$ )
```

DFS-Visit(v):

```
 $prenum[v] := pre$ ;  $pre := pre + 1$ ;  
 $new[v] := False$   
for  $w \in L[v]$  do  
    if  $new[w]$  then  
         $T := T \cup \{ (v, w) \}$   
        DFS-Visit( $w$ )  
 $postnum[v] := post$ ;  $post := post + 1$ ;
```

Sama algoritmi toimii suunnatuille ja suuntaamattomille verkoille. (Jälkimmäisessä tapauksessa vain aina $w \in L[v] \Leftrightarrow v \in L[w]$.) Lopputuloksen tulkinta kuitenkin on hieman erilainen.

Tarkastellaan ensin suunnattuja verkkoja.

Lause Verkon $G = (V, E)$ syvyysuuntainen haun aikavaativuus on $O(|V| + |E|)$.

Todistus Muut osat kuin *DFS-Visit*-kutsut vievät selvästi ajan $O(|V|)$.

Kullakin $v \in V$ kutsu *DFS-Visit*(v) suoritetaan tasan kerran, sillä kutsu merkitsee solmun v heti vanhaksi eikä siihen enää mennä.

Kutsun *DFS-Visit*(v) aikavaativuus lukuunottamatta rekursiivisia *DFS-Visit*-kutsuja on selvästi $O(1 + |L[v]|)$.

Siis kaikkiaan *DFS-Visit*-kutsut vievät ajan

$$O\left(\sum_{v \in V} (1 + |L[v]|)\right) = O(|V| + |E|).$$

□

Syvyysuuntainen haku jakaa kaaret puukaariin T ja poikittaiskaariin $E - T$.

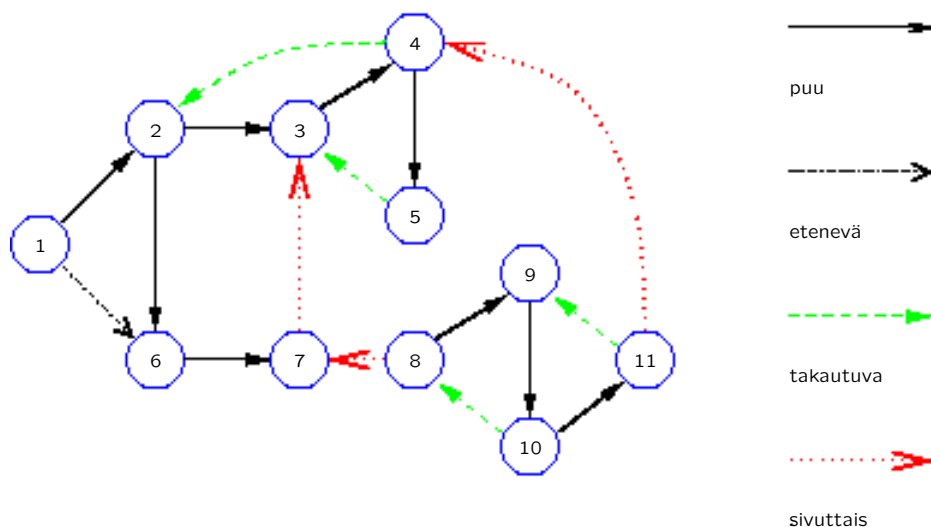
Tämä jako, ja solmujen numeroinnit, eivät ole yksikäsitteisiä, vaan riippuvat vieruslistojen järjestyksestä ja solmujen valintajärjestyksestä proseduurin *DFS for*-silmukassa.

Suunnatussa verkossa poikittaiskaaret jaetaan edelleen:

etenevät kaaret: solmusta sen aitoihin jälkeläisiin

takautuvat kaaret: solmusta sen esivanhempiin (mukaanlukien solmusta itseensä)

sivuttaiskaaret: kaikki muut



Eräs mahdollinen esinumerointi ja kaarten jaottelu verkossa

Lemma Jos (v, w) on sivuttaiskaari, niin $prenum[v] > prenum[w]$ ja $postnum[v] > postnum[w]$.

Todistus Oletetaan $(v, w) \in E$.

Jos $v = w$, niin (v, w) on määritelmän mukaan takautuva. Oletetaan $v \neq w$.

Jos $prenum[v] < prenum[w]$, niin kutsun $DFS-Visit(v)$ tapahtuessa w on uusi. Siis w tulee solmun v jälkeläiseksi.

Olkoon toisaalta $prenum[v] > prenum[w]$ mutta $postnum[v] < postnum[w]$. Siis kutsun $DFS-Visit(v)$ päättyessä kutsu $DFS-Visit(w)$ on vielä kesken, joten v on solmun w jälkeläinen. \square

Topologinen järjestäminen

Annettu: suunnattu syklitön verkko (Directed Acyclic Graph, DAG) $G = (V, E)$.

Määrättävä: kullekin solmulle $v \in V$ järjestysnumero $s[v]$ siten, että $s[v] < s[w]$ jos $(v, w) \in E$

Ratkaisu: $s[v] = |V| - \text{postnum}(v)$

Perustelu: Olkoon $(v, w) \in E$.

Syklittömyyden takia takautuvia kaaria ei ole.

Jos (v, w) on etenevä tai puukaari, algoritmista nähdään suoraan $\text{postnum}[v] > \text{postnum}[w]$.

Jos (v, w) on sivuttaiskaari, niin $\text{postnum}[v] > \text{postnum}[w]$ edellisen lemmän perusteella. \square

Muita helppoja DFS:n sovelluksia:

saavutettavuuden testaaminen, syklittömyyden testaaminen

Vahvasti yhtenäiset komponentit

Suunnatussa verkossa G merkitään $u \rightsquigarrow v$ jos solmusta u on polku solmuun v .

Määritellään ekvivalenssirelaatio

$$u \sim v \iff (u \rightsquigarrow v \text{ ja } v \rightsquigarrow u).$$

Ekvivalenssirelaation " \sim " ekvivalenssiluokat ovat verkon **vahvasti yhtenäiset komponentit**.

Algoritmi: verkon G vahvasti yhtenäiset komponentit

1. Laske verkon G solmujen jälkinumerot $postnum[v]$.
2. Muodosta G^r , joka on muuten kuin G mutta kaarten suunnat on vaihdettu.
3. Suorita verkossa G^r syvyysuuntainen haku siten, että proseduurissa DFS solmut käsitellään jälkinumeroiden $postnum$ mukaan laskevassa järjestyksessä.
4. Syntyvän virittävän metsän jokainen puu on samalla verkon G vahvasti yhtenäinen komponentti.

Lause Algoritmi laskee vahvasti yhtenäiset komponentit oikein ja ajassa $O(|V| + |E|)$.

Todistus Aikavaativuus on selvä.

Pitää osoittaa, että u ja v ovat samassa vahvasti yhtenäisessä komponentissa joss ne ovat samassa verkon G^r syvyysuuntaisessa virittävässä puussa. Jatkossa polut ja numeroinnit viittaavat alkuperäiseen verkkoon G ellei muuta mainita.

" \Rightarrow ": Jos verkossa G sekä $u \rightsquigarrow v$ että $v \rightsquigarrow u$, niin sama pätee myös verkossa G^r . Siis u ja v päätyvät samaan puuhun.

" \Leftarrow ": Olkoot u ja v samassa virittävässä puussa, jonka juuri on x . Siis $u \rightsquigarrow x$ ja $v \rightsquigarrow x$. Riittää osoittaa, että myös $x \rightsquigarrow u$ ja $x \rightsquigarrow v$.

Osoitetaan $x \rightsquigarrow v$; väite $x \rightsquigarrow u$ menee samaan tapaan.

Jos $v = x$, asia on selvä. Muuten $postnum[x] > postnum[v]$. Toisaalta koska $v \rightsquigarrow x$, kutsu $DFS-Visit[x]$ alkaa ennen kuin $DFS-Visit[v]$ päättyy. Siis v on solmun x jälkeläinen verkon G syvyysuuntaisessa virittävässä metsässä, ja erityisesti $x \rightsquigarrow v$. \square

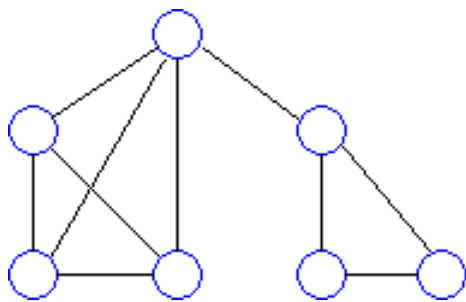
Syvyyssuuntainen haku suuntaamattomassa verkossa

Algoritmi ja sen aikavaativuus ovat samat kuin suunnatussa tapauksessa.

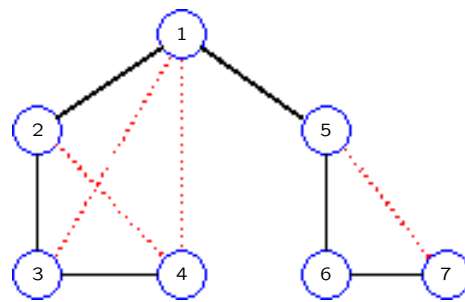
Kaarten suuntaamattomuudesta seuraa, että sivuttaiskaaria ei synny. Haku siis jakaa kaaret kahteen luokkaan:

puukaaret (joukko T) muodostavat virittävän metsän

takautuvat kaaret (joukko $E - T$) yhdistävät esivanhempia ja jälkeläisiä.



suuntaamaton verkko



eräs esinumerointi, jako puukaariin (yhtenäinen) ja takautuviin (pisteet)

2-yhtenäiset komponentit ja artikulaatiopisteet:

esimerkkisovellus syvyysuuntaiselle haulle
suuntaamattomassa verkossa

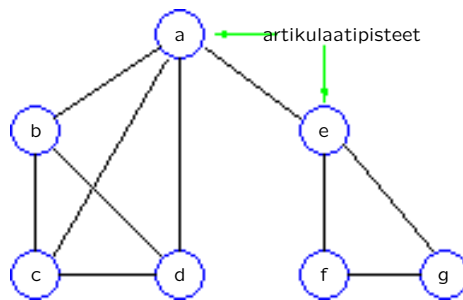
Oletamme jatkossa, että käsiteltävä verkko on yhtenäinen. (Muuten käsitellään kukin yhtenäinen komponentti erikseen.)

Yhtenäisen verkon solmu on **artikulaatiopiste** jos sen poistaminen (siihen liittyvine kaarineen) tekee verkosta epäyhtenäisen.

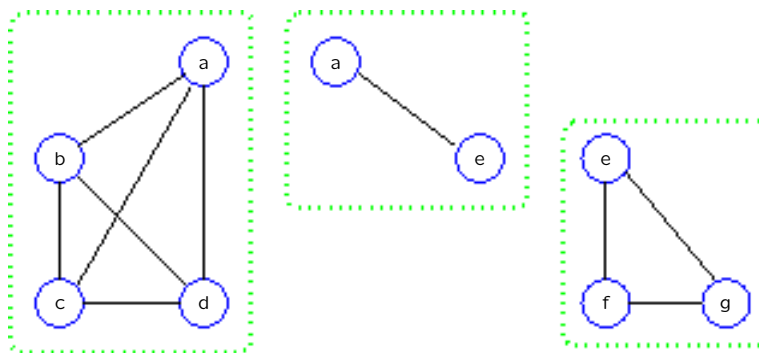
Verkko on **2-yhtenäinen** jos sillä ei ole artikulaatiopisteitä.

Esim. tietoliikenneverkon tapauksessa 2-yhtenäisyys tarkoittaa, että yhden solmun poistuminen ei estä muiden solmujen välisiä yhteyksiä. Tämä yleistyy ilmeisellä tavalla k -yhtenäisyydeksi, $k \geq 2$.

Alustava intuitiivinen määrittely: verkon 2-yhtenäiset komponentit ovat sen maksimaaliset 2-yhtenäiset osaverkot.



Artikulaatiopisteet



2-yhtenäiset komponentit
(solmujoukot $\{a, b, c, d\}$, $\{a, e\}$ ja $\{e, f, g\}$)

Kerrataan määritelmiä:

- solmujono (v_1, \dots, v_k) on **polku** jos $(v_i, v_{i+1}) \in E$ kaikilla $1 \leq i \leq k - 1$
- polku (v_1, \dots, v_k) on **yksinkertainen** jos v_1, v_2, \dots, v_k ovat kaikki eri solmuja
- polku (v_1, \dots, v_k) on **kehä** jos $v_1 = v_k$
- kehä (v_1, \dots, v_k) on **yksinkertainen** jos v_1, v_2, \dots, v_{k-1} ovat kaikki eri solmuja

Huomaa erikoistapaus: jos $(u, v) \in E$ niin (u, v) on yksinkertainen kehä.

Määritellään nyt kaarijoukossa E ekvivalenssirelaatio " \sim " seuraavasti:

$$e \sim e' \Leftrightarrow \text{jokin yksinkertainen kehä sisältää sekä kaaren } e \text{ että kaaren } e'.$$

Olkoot relaation " \sim " ekvivalenssiluokat E_1, \dots, E_k . Kun $i = 1, \dots, k$, muodostukoon $V_i \subseteq V$ niistä solmuista, jotka esiintyvät luokan E_i kaaren päätepisteenä.

Verkon G **2-yhtenäiset komponentit** ovat nyt osaverkot $G_i = (V_i, E_i)$, $i = 1, \dots, k$.

Lemma Olkoon $G = (V, E)$ yhtenäinen, ja sen 2-yhtenäiset komponentit $G_i = (V_i, E_i)$, $i = 1, \dots, k$.

1. G_i on 2-yhtenäinen kaikilla i .
2. Jos $i \neq j$ niin $|V_i \cap V_j| \leq 1$.
3. a on verkon G artikulaatiopiste joss $a \in V_i \cap V_j$ joillain $i \neq j$.

Todistus Selvästi kaikki G_i ovat yhtenäisiä.

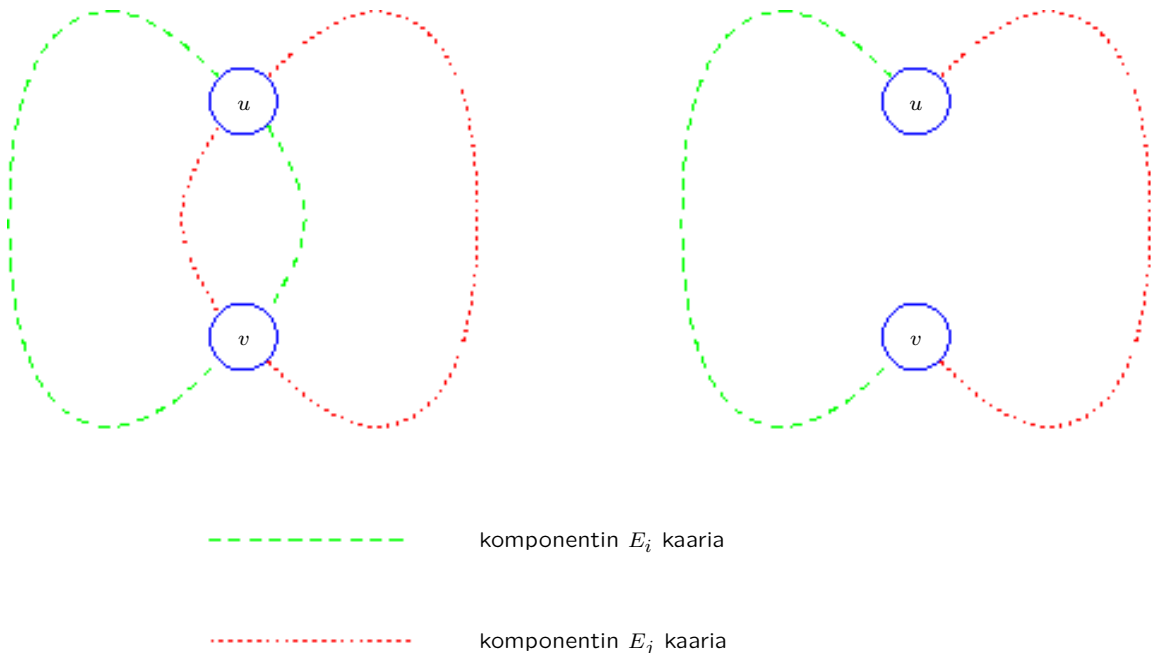
(1): Olkoon $a, u, v \in V_i$. Väitetään, että jos $a \notin \{u, v\}$ niin solmujen u ja v välillä on ainakin yksi polku joka ei kulje solmun a kautta. Jos $(u, v) \in E_i$, asia on selvä. Muuten on olemassa kaksi eri kaarta $(u, u') \in E_i$ ja $(v, v') \in E_i$, ja nämä kaaret ovat jollakin yksinkertaisella kehällä. Kehä antaa kaksi eri polkua solmujen u ja v välille, ja a voi olla vain toisella niistä.

(2): Tehdään vastaoletus $\{u, v\} \subseteq V_i \cap V_j$, $i \neq j$, $u \neq v$.

Koska ei voi olla $V_i \subseteq V_j$, on olemassa jokin $x \in V_i - \{u, v\}$. Tällä x on olemassa joukon E_i kaarista koostuva kehä, joka kulkee solmujen u , v ja x kautta.

Vastaavasti jollain $y \in V_j - \{u, v\}$ jokin joukon E_j kaarista koostuva kehä kulkee pisteiden u , v ja y kautta.

Näiden kehien paloja yhdistämällä saadaan yksinkertainen kehä, joka sisältää kaaria sekä joukosta E_i että joukosta E_j ; ristiriita.



(3), " \Rightarrow ": Olkoon a verkon G artikulaatiopiste.

Siis joillain u, v kaikki polut $u \rightsquigarrow v$ kulkevat pisteen a kautta.

Olkoon jollain tällaisella polulla x ja y solmua a välittömästi ympäröivät solmut:

$$u \rightsquigarrow x \rightarrow a \rightarrow y \rightsquigarrow v.$$

Jos kaaret (x, a) ja (a, y) kuuluisivat samaan komponenttiin, ne olisivat jollain yksinkertaisella kehällä. Tästä kehästä saataisiin solmujen u ja v välille vaihtoehtoinen polku joka ei kulje solmun a kautta; ristiriita.

Siis $(x, a) \in E_i$ ja $(a, y) \in E_j$ joillain $i \neq j$, ja tällöin $a \in V_i \cap V_j$.

(3), " \Leftarrow ": Olkoon $a \in V_i \cap V_j$, $i \neq j$.

Siis $(x, a) \in E_i$ ja $(a, y) \in E_j$ joillain x, y . Jos solmujen x ja y välillä olisi polku, joka ei kulje solmun a kautta, syntyisi kehä jolla olisi kaaret $(x, a) \in E_i$ ja $(a, y) \in E_j$; ristiriita.

Siis kaikki polut $x \rightsquigarrow y$ kulkevat solmun a kautta ja se on artikulaatiopiste.



Tavoitteena on löytää artikulaatiopisteet ja 2-yhtenäiset komponentit syvyysuuntaisen haun yhteydessä.

Tarkastellaan ensin, miten artikulaatiopisteet tunnustetaan, jos syvyysuuntainen virittävä puu T ja takautuvat kaaret $E - T$ on valmiiksi annettu.

Jos a ei ole artikulaatiopiste, erityisesti solmun a jokaisesta lapsesta on solmun a isään polku, joka ei kulje solmun a kautta. Koska sivuttaiskaaria ei ole, tällä polulla on oltava kaari solmun a jälkeläisestä solmun a aitoon esivanhempaan.

Tarkemmin pätee seuraava:

Lause Olkoon verkko $G = (V, E)$ yhtenäinen ja $S = (V, T)$ sen syvyysuuntainen virittävä puu.

Nyt $a \in V$ on verkon G artikulaatiopiste joss

1. a on puun S juuri ja sillä on ainakin kaksi lasta, tai
2. a ei ole puun S juuri ja sillä on lapsi s , jonka mistään jälkeläisestä (s itse mukaanlukien) ei ole takautuvaa kaarta mihinkään solmun a aitoon esivanhempaan.

Todistus Tapaus jossa a on juuri on selvä. Oletetaan että a ei ole juuri.

" \Leftarrow ": Oletetaan, että a ei ole artikulaatiopiste ja s on solmun a lapsi.

Siis erityisesti solmusta s on polku (v_1, \dots, v_k) solmun a vanhempaan p kulkematta solmun a kautta.

Olkoon j pienin, jolla v_j ei ole solmun s jälkeläinen. Tällainen j on olemassa, koska $v_k = p$ ei ole solmun s jälkeläinen.

Koska sivuttaiskaaria ei ole, v_j on solmun s aito esivanhempi.

Koska lisäksi $v_j \neq a$, kaari (v_{j-1}, v_j) on takautuva kaari solmun s jälkeläisestä solmun a aitoon esivanhempaan.

" \Rightarrow ": Oletetaan että jokaisella solmun a lapsella s on jälkeläinen v , josta on takautuva kaari johonkin solmun a aitoon esivanhempaan. Väitetään, että kaikilla $x, y \in V$ on polku $x \rightsquigarrow y$ kulkematta solmun a kautta.

Väitetään, että kummastakin solmusta x ja y on puun juureen polku kulkematta solmun a kautta. Väite selvästi seuraa tästä. Todistus on sama solmuille x ja y , tarkastellaan tapausta x .

Jos x ei ole solmun a jälkeläinen, haluttu polku muodostuu suoraan puukaarista.

Olkoon x nyt solmun a jälkeläinen. Olkoon s se solmun a lapsi, jonka jälkeläinen x on, ja olkoon v solmun s jälkeläinen, josta on takautuva kaari solmun a aitoon esivanhempaan w .

Solmusta x solmuun v päästään puukaaria pitkin solmuun v nousematta solmun s "yläpuolelle" ja erityisesti käymättä solmussa a .

Kaari (v, w) oletuksen mukaan ei sisällä solmua a .

Solmusta w päästään juureen suoraan puukaaria pitkin kulkematta solmun a kautta.

□

Edellä esitetyn ehdon tehokas tarkastaminen perustuu ns. Low -arvojen laskemiseen.

Käytetään yksinkertaisuuden vuoksi solmujen niminä niiden esinumeroita; siis $v = prenum[v]$.

Kullakin v olkoon $A(v)$ niiden solmujen joukko, joihin on kaari jostain solmun v jälkeläisestä. Huomaa, että joukossa $A(v)$ on vain solmun v jälkeläisiä ja esivanhempia.

Asetetaan

$$Low[v] = \min(\{v\} \cup A(v)).$$

Siis $Low[v]$ on juurta lähinnä oleva solmu, johon pääsee solmusta v kulkemalla ensin puussa alaspäin ja sitten mahdollisesti *kerran* ylöspäin pitkin takautuvaa kaarta.

Erityisesti solmun a lapsen s jostain jälkeläisestä on takautuva kaari johonkin solmun a aitoon esivanhempaan joss $Low[s] < a$.

Siis a on artikulaatiopiste joss jollain sen lapsella s pätee $Low[s] \geq a$.

Low -arvot saadaan selvästi palautuskaavasta

$$Low[v] = \min(\{v\} \cup \{w \mid (v, w) \text{ takautuva}\} \cup \{Low[w] \mid w \text{ solmun } v \text{ lapsi}\})$$

Lasketaan nyt *Low*-arvot tällä palautuskaavalla syvyysuuntaisen haun yhteydessä. Koska verkko oletetaan yhtenäiseksi, yksi ylätasoinen kutsu seuraavaan proseduriin *DFS-Visit2* millä tahansa solmulla käy koko verkon läpi.

DFS-Visit2(v):

```

pre := pre + 1; prenum[v] := pre
new[v] := False
Low[v] := prenum[v]
for w ∈ L[v] do
  if new[w] then
    T := T ∪ { (v, w) }
    p[w] := v
    DFS-Visit2(w)
    if Low[w] ≥ prenum[v] then
      v on artikulaatiopiste tai puun juuri
      Low[v] := min { Low[v], Low[w] }
  else if w ≠ p[v] then
    % (v, w) takautuva
    Low[v] := min { Low[v], prenum[w] }

```

Edellä esitettyjen ominaisuuksien perusteella helppo induktio osoittaa, että algoritmi laskee *Low*-arvot ja tunnistaa artikulaatiopisteet oikein, kunhan vielä puun juuri käsitellään erikoistapauksena.

Katsomme vielä, miten saamme tulostetuksi 2-yhtenäiset komponentit.

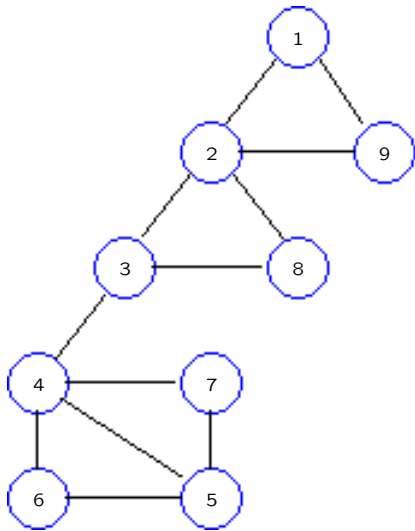
Algoritmi: 2-yhtenäiset komponentit

1. Alusta $T := \emptyset$, $pre := 0$; $P :=$ tyhjä pino, $new[v] := True$ kaikilla v .
2. Suorita $DFS-Visit2(v_0)$ mielivaltaisella $v_0 \in V$.

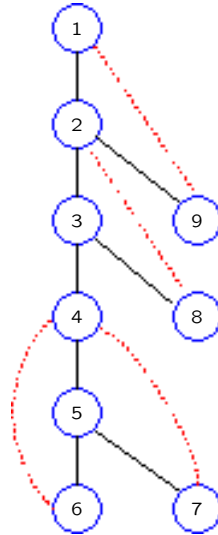
Aina kun algoritmi löytää solmun $w \in L[v]$, paina kaari (v, w) pinoon P ellei sitä ole jo aiemmin painettu.

Kun $DFS-Visit2(v)$ solmua w käsitellessään toteaa, että v on artikulaatiopiste tai juuri, poista pinosta kaaret kaareen (v, w) asti (tämä mukaanlukien). Nämä kaaret muodostavat yhden 2-yhtenäisen komponentin.

Seuraavaksi katsomme ensin esimerkin algoritmin toiminnasta ja todistamme sitten sen aikavaativuuden ja oikeellisuuden.



Verkko



Virittävä puu ja
takautuvat kaaret

v	1	2	3	4	5	6	7	8	9
$Low[v]$	1	1	2	4	4	4	4	2	1

(1, 2)(2, 3)(3, 4) (4, 5)(5, 6)(6, 4)(5, 7)(7, 4)

(1, 2)(2, 3) (3, 4)

(1, 2) (2, 3)(3, 8)(8, 2)

(1, 2)(2, 9)(9, 1)

\emptyset

Pinon kehitys ja tulostetut komponentit

Aikavaativuus: Syvyysuuntaisen haun analyysistä seuraa, että aikavaativuus on $O(|V| + |E|)$, kunhan osoitetaan miten vakioajassa testataan onko kaari (v, w) joskus aiemmin painettu pinoon.

1. Jos $new[w] = True$, kaarta ei ole aiemmin kohdattu eikä siis etenkään painettu pinoon.
2. Jos $new[w] = False$ ja $v < w$, niin w on solmun v jälkeläinen, joten (v, w) on viety pinoon.
3. Jos $new[w] = False$, $w < v$ ja $w = p[v]$, niin kaari (v, w) on viety pinoon juuri ennen kutsua $DFS-Visit2(v)$.
4. Jos $new[w] = False$, $w < v$ ja $w \neq p[v]$, niin w on solmun v kaukaisempi esivanhempi josta kuitenkään ei tultu suoraan solmuun v , joten kutsussa $DFS-Visit2(w)$ ei vielä ole ehditty listan $L[w]$ alkioon v eikä kaarta (v, w) siis ole painettu pinoon.

Siis voidaan testata vakioajassa, onko kaari (v, w) aiemmin viety pinoon.

(Tapauksessa 2 se on voitu jo ehtiä poistaakin pinosta.)

Koska verkko on yhtenäinen, $|E| \geq |V| - 1$ ja aikavaativuudeksi tulee $O(|E|)$.

Lause Algoritmi tulostaa oikeat 2-yhtenäiset komponentit.

Todistus Kaikilla w pätee $Low[w] \geq v_0$, missä v_0 on puun juuri. Siis kaikilla juuren lapsilla w pino tyhjennetään kutsun $DFS-Visit2(w)$ päätyttyä. Erityisesti lopuksi pino on tyhjä ja jokainen kaari on tulostettu tasan kerran.

Todistetaan induktiolla komponenttien lukumäärän b suhteen, että kullakin kerralla tulostettavat kaaret muodostavat yhden kokonaisen komponentin.

Tapauksessa $b = 1$ verkossa ei ole artikulaatiopisteitä, joten juurella on tasan yksi lapsi. Kun tämä lapsi on käsitelty, pinossa on kaikki verkon kaaret, jotka tulostetaan.

Oletetaan, että algoritmi toimii, kun komponentteja on $b - 1$.

Olkoot v ja w ne solmut, joiden kohdalla ensimmäisen kerran käy $Low[w] \leq v$ kun $v = p[w]$. Siis v on ensimmäinen löydetty artikulaatiopiste, ja ennen kutsun $DFS-Visit2(w)$ päättymistä pinosta ei ole poistettu mitään.

Täten kutsun $DFS-Visit2(w)$ päätyttyä tulostetaan ja poistetaan tasan ne kaaret, joilla ainakin toinen päätepiste on solmun v jälkeläinen. Selvästi nämä kaaret muodostavat yhden 2-yhtenäisen komponentin.

Ensimmäisen komponentin tulostamisen jälkeen algoritmi toimii kuten sillä verkolla, joka saadaan poistamalla ensimmäinen komponentti mutta jättämällä solmu v . Induktio-oletuksesta siis seuraa, että loputkin komponentit tulostuvat oikein. \square