INF 3/4130
21. september 2006

- Dagens temaer:
  - Avsluttende kap. 14 (Pensum: Alt untatt 14.1.3 og 14.2.6)
  - Om å finne den største matchingen i en ikke-bipartit graf (Notat om dette kommer, og pensum for dette stoffet er angitt på siste foil)
- Oblig 1 er lagt ut.
  - Leveringsanvisning kommer. Frist fredag 5 oktober.
- Neste uke:
  - Balanserte søketrær (noe stoff taes fra M.A.Weiss, boka til INF1020)

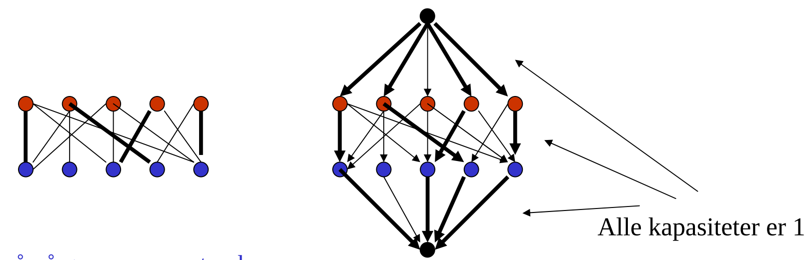
Avslutning av kapittel 14

Kap. 14.2.7: En sammenheng mellom flyt i grafer og matchinger i bipartite grafer

Enkelt men viktig lemma:

1. Ved heltallige kapasiteter kan man alltid finne en maksimal flyt som er heltallig.
2. Når alle kapasitetene er 1 kan vi altså finne en maksimal flyt der hver kant har flyt 0 eller 1 (og FordFulkerson vil alltid finne en slik!)

En slik flyt kan dermed tolkes som et *utplukk av kanter* (de som har flyt)



Se på på gruppene neste uke:

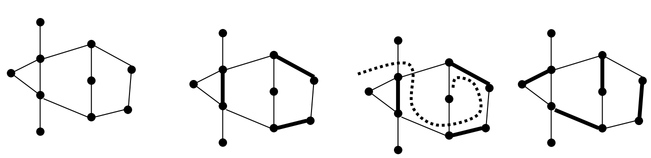
- Leting etter forbedringsvei i flytnettverket tilsvarer leting etter forbedringsvei ut fra en matching i grafen
- Et utplukk av noder som dekker alle kanter (gruppeoppgave denne uken) tilsvarer et kutt i flytnettverket

Maks-matching-problemet for generelle grafer

Vi skal bare se på "flest mulig kanter". Algoritmen for det veiede problemet er mer komplisert.

Eksempler på at dette problemet oppstår:

- Lage 2-personers lag i en klasse der man vet hvem som jobber godt sammen
  - Noder: Elevene. Kanter: Mellom alle par av elever som jobber godt sammen
- Sokker som skal lages til par etter vasken
  - Noder: Sokkene, Kanter: Mellom sokker som er like nok til kunne være et akseptabelt par.
- Det homoseksuelle partner-problemet
  - Blir ofte nevnt, men er vel like unaturlig som eksempel som det (hetroseksuelle) gifte-problemet
- I mange andre problemer oppstår det å lage en maksimal matching i en generell graf som et essensielt underproblem.



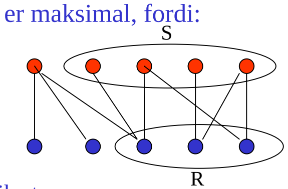
Noe bedre?

En maks-matching-algoritme for generelle grafer

Det kommer et notat om dette (litt oppdatert fra i fjor).

Vi trenger et nytt kriterium for å vise at en matching er maksimal, fordi:

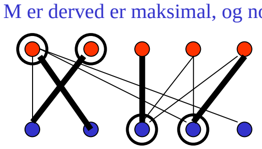
Hall's teorem har bare mening for bipartite grafer:



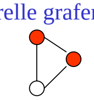
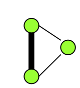
Kønig-Egervàrys-teorem (fra gruppeøvelsene):

I en bipartit graf kan man finne et node-utplukk X slik at:

- X dekker alle kanter i grafen
- Det finnes en matching M med like mange kanter som X har noder
- M er derved er maksimal, og noe mindre slikt overdekkende utplukk finnes ikke

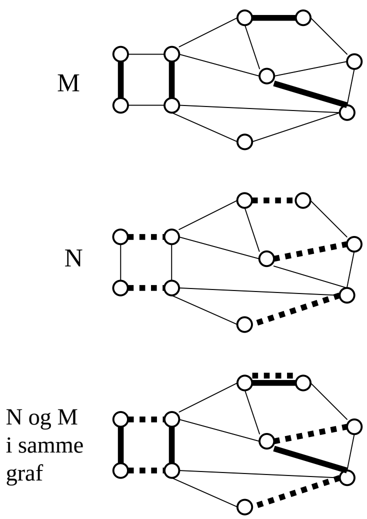


Dette teoremet gjelder dessverre ikke for generelle grafer:

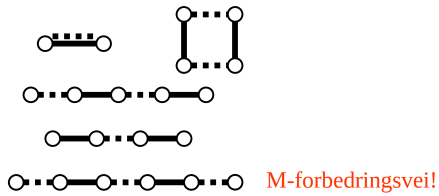
Minste kant-overdekkene node-utplukk (rødt):  Største matching: 

Nytt kriterium: Forbedringsveier

- For generelle grafer: Gitt en matching M. Dersom det finnes en større matching N, så finnes også en forbedrings-vei i M. Bevis:



Mulige mønstre som inneholder kanter fra M og/eller N:



Bare det siste mønstret har flere N-kanter enn M-kanter. Derfor må det finnes minst ett slikt. Og det er altså en forbedringsvei for M!

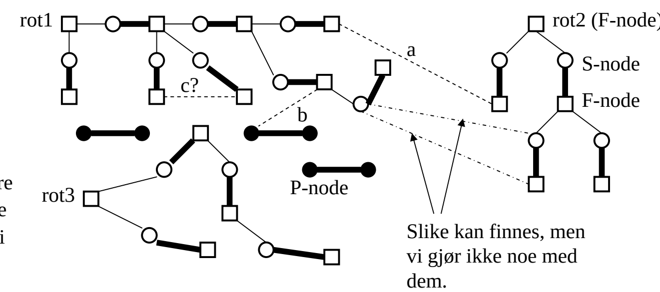
Altså, nytt kriterium: Om en matching ikke har forbedringsveier, så er den maksimal

Nok en maks-matching-algoritme for bipartite grafer

(Litt mer komplisert, men den lar seg generalisere til generelle grafer)

- Idé: Gror *alternerende trær* på "vanlig måte", men fra *alle* umatchede noder samtidig
  - Både fra de i X og de i Y (siden noe slikt skille ikke finnes i generelle grafer)
  - Nodene i trærne er enten F(irkant)-noder eller S(irkel)-noder, og alle noder utenfor trærne er P(rikk)-noder. Røttene i trærne er pr. def. F-noder.
  - Når vi går fra en rot utover en vei i treet vil annenhver node være F-node og S-node, og alle kanter fra S-node til F-node vil være matching-kanter.
  - Merk: F-noder og S-noder tilsvarer ikke X og Y, siden røtter kan være både X- og Y-noder
- Initialisering: Alle ikke-matchede noder (både i X og Y) settes til røtter i hvert sitt tre, og dermed til F-noder. Resten blir P-noder. Alle P-noder er altså matchede.
- Steget: Se etter kanter fra F-noder til P-noder eller fra F-noder til F-noder:

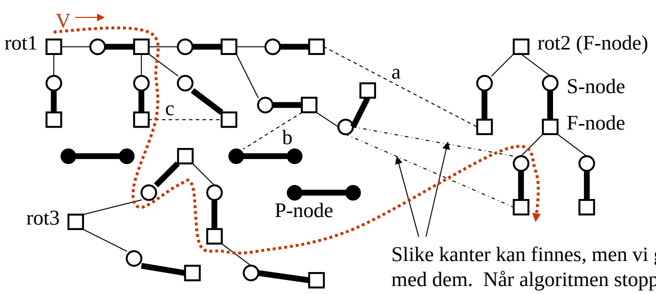
- (a): Kanten går til F-node i annet tre: Da er forb.vei funnet
- (b): Kanten til P-node. Utvid treet på vanlig måte
- Kant til F-node i eget tre (c) finnes ikke (gir odde løkke, som ikke finnes i bipartite grafer).



Avslutningssituasjonen har ikke forbedringsvei

Anta at algoritmen stopper uten å finne en F- til F-kontakt mellom to trær. Gitt en alternerende vei V, som starter i en rot. Vi vil vise at V aldri kan komme til en annen rot (og derved ikke kan være en forbedrings-vei). Vi viser dette gjennom to lemmaer:

Lemma 1: Om V starter i en rot-node (som er en F-node) eller kommer utenifra og inn i et tre i en S-node, så vil V, så lenge den går mellom noder i dette treet, alltid følge matchede kanter fra en S- til en F-node, og den vil ikke komme til roten av treet.

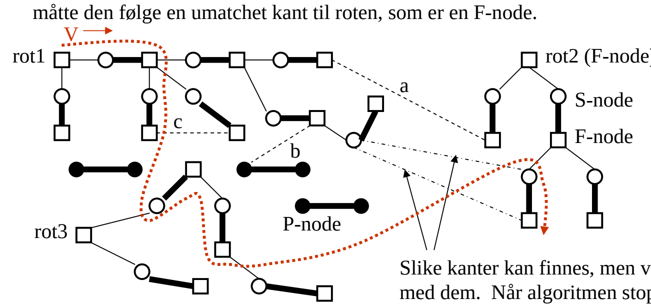


Slike kanter kan finnes, men vi gjør ikke noe med dem. Når algoritmen stopper finnes ikke kanter av typen a eller b (og hvertfall ikke c!). Merk at V også kan gå mellom noder innen et tre langs kanter som ikke er med i treet (ikke vist)

Lemma 1: Om V starter i en rot-node (en F-node) i et tre eller kommer utenifra og inn i et tre i en S-node, så vil V, så lenge den går mellom noder i dette treet, alltid følge matchede kanter fra en S- til en F-node, og motsatt for umatchede kanter. Den vil (derved) ikke komme til roten av treet.

Bevis:

- Innen *ett og samme tre* kan det ikke være F-F-kanter eller S-S-kanter (fordi grafen er bipartit)
- Derfor, så lenge V går mellom nodene i ett tre må den annenhver gang ha en F- og en S-node
- Om V starter i roten av treet, og forblir i treet må den først følge en umatchet kant til en S-node.
- Om V kommer utenfra treet (og altså til en S-node i treet), kommer den langs en umatchet kant.
- I begge disse tilfellene går den altså langs en umatchet kant til en S-node i treet. Den neste kanten i V må da være matchet, og gå til en F-node, så må V følge en umerket kant til S-node osv.
- Dermed vil V, så lange den går mellom noder i dette treet, følge matchede kanter fra S- til F-noder, og umatchede fra F- til S-noder. Dermed vil den heller ikke kunne komme til roten av treet, for da måtte den følge en umatchet kant til roten, som er en F-node.



Slike kanter kan finnes, men vi gjør ikke noe med dem. Når algoritmen stopper finnes ikke kanter av typen a eller b (og hvertfall ikke c!)

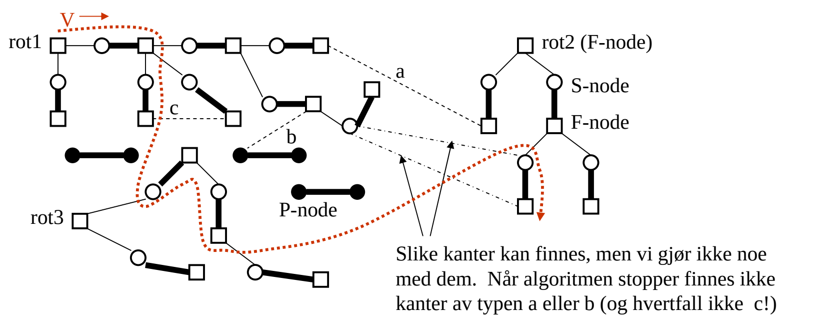
Lemma 2: Dersom V startet i roten av et tre T eller kom inn i treet T til i S-node, så kan V bare forlate T i en F-node, langs en umatchet kant som fører til en S-node i et annet tre.

Bevis:

- Den siste kanten V fulgte i T måtte være en matchet kant.
- Ut fra Lemma 1: Så lenge V er i treet, må den følge matchede kanter fra en S- til en F-node. Dermed vil den altså måtte forlate treet i en F-node.
- Den kommer da til en S-node siden algoritmen stoppet nettopp fordi det ikke var noen kanter fra F-noder i T til F-noder i andre trær, eller fra F-noder til P-noder.

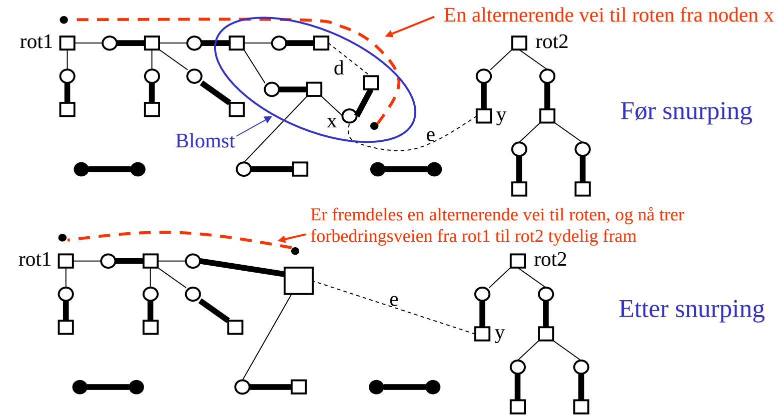
Dermed: Lemma 1 og 2 sier til sammen at V ikke kan nå en rotnode i et annet tre, og altså ikke være en forbedringsvei.

Grovbeskrivelse av situasjonen: Hver gang V går langs en matchet kant, så beveger den seg bort fra roten i det treet som kanten er med i.



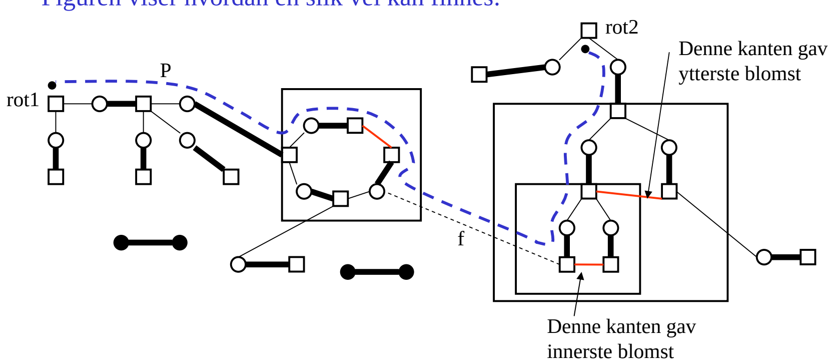
Algoritmen for generelle grafer

- Da *kan* det plutselig finnes kanter (som *d* i figuren, *c* i forrige) mellom to F-noder i samme tre. Da dannes odde løkker med treet, og disse kalles ”blomster”.
- Merk at det da går alternerende veier fra *alle* noder i blomsten tilbake til roten
- Dermed vil kanten *e* ”plutselig” gi en forbedringsvei fra rot2 til rot1
- Derfor: ”Snurp” en blomst sammen til en firkantnode så fort den oppstår.
- Merk: Den opprinnelige grafen med hver enkelt blomst snurpet sammen til en enkelt F-node kaller vi den ”(sammen-)snurpede grafen”. Alle indere kanter i blomstene er da vekk.

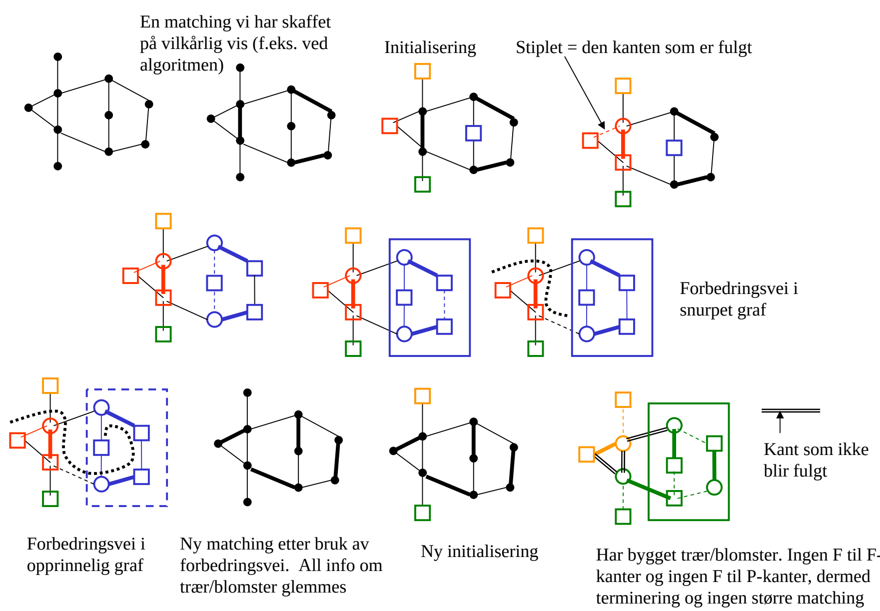


Forbedringsveier i den opprinnelige grafen

- Det kan etter hvert bli mange blomster, også inne i hverandre
- Det er da viktig at en forbedringsvei som oppstår i den snurpete grafen, også angir en forbedringsvei i den opprinnelige grafen.
- Figuren viser hvordan en slik vei kan finnes:



Et eksempel



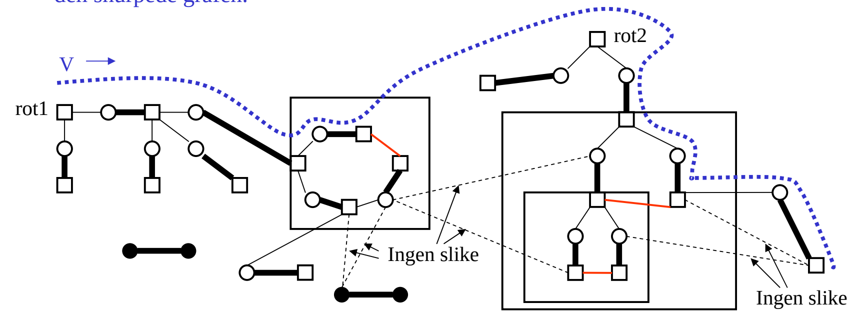
Ingen forbedringsveier ved terminering

(Se figur neste foil)

- Anta at denne algoritmen stopper uten at det blir funnet noen forbedringsvei.
  - Må da vise at ingen forbedringsveier finnes (i den opprinnelige grafen), slik at matchingen altså er maksimal.
- På den sammensnurpede grafen går argumentet som for bipartite grafer. Der gjelder Lemma 1 og 2 som før, og det er ingen forbedringsveier i den ut fra den gitte matchingen
  - Ut fra Lemma 1 sier dette også noe vi trenger lenger ned, nemlig at V alltid må gå inn i en blomst langs en matchet kant, siden en blomst er en F-node i den snurpede grafen
- Om da alle forbedringsveier V i den opprinnelige grafen også blir forbedringsveier V' i den snurpede grafen, ville vi være i mål. For å vise at det er slik, viser vi følgende:
  - V kan bare være innom en gitt blomst én gang.
    - Dette er riktig fordi V bare kan entre blomsten langs en *matchet* kant (se over), og slike finnes det bare én av for hver blomst.
  - V sin første kant i en blomst er alltid umatchet, og V forlater en blomst langs en umatchet kant.
    - Det første er opplagt siden den kom inn langs en matchet kant, og det andre må gjelde siden V brukte den eneste matchede kanten med én node i og én node utenfor blomsten til å komme inn i blomsten.
- Dermed vil det stykket som V går inni en blomst bestå av kanter: (umatchet, matchet, umatchet,…, matchet).
  - Slike stykker av V vil dermed forsvinne når man snurper grafen, men den resterende V' vil da fremdeles være en alternerende vei i den snurpede grafen.
  - Dersom V var en forbedringsvei i den opprinnelige grafen ville dermed også V' være en forbedringsvei i den snurpede grafen. Og dermed er saken bevist.

Ingen forbedringsveier ved terminering

Figur til beviset (på forrige foil) for at det ikke er forbedringsveier i den opprinnelige grafen når algoritmen stopper uten å finne F-til-F kant mellom trær i den snurpede grafen.

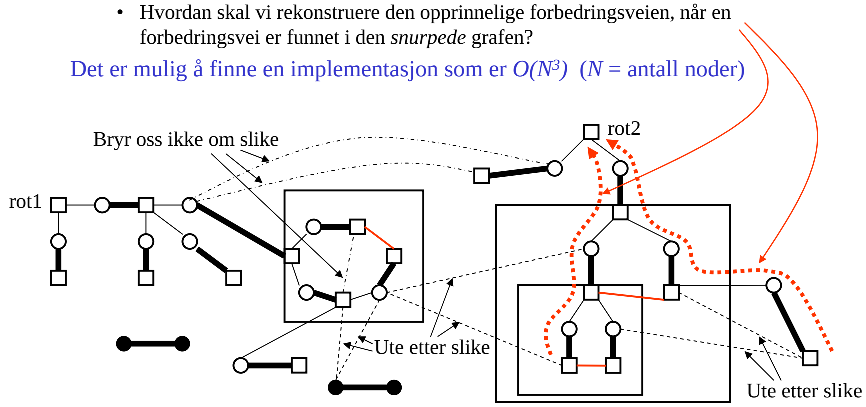


Implementasjon

Dere skal se litt på dette på gruppene neste uke.

- Hvordan skal man få testet om to noder er i samme tre?
- Hvordan skal man få testet om to noder er i samme blomst? Det oppstår stadig blomster, og blomster innlemmes i større blomster
- Hvordan skal "den snurpede grafen" lagres? Med egne snurpe-noder?
- Hvordan skal vi rekonstruere den opprinnelige forbedringsveien, når en forbedringsvei er funnet i den *snurpede* grafen?

Det er mulig å finne en implementasjon som er $O(N^3)$ (N = antall noder)



Pensum angående matchinger i generelle grafer

- Det man skal kunne til eksamen om algoritmen som finner en maksimal matching i en generell graf er følgende:
  - Man skal kunne selve algoritmen, og kunne utføre den pr hånd på en generell graf
  - Man skal kunne *beviset* for at om en matching i en gitt graf ikke er maksimal, så finnes en forbedrings-vei i grafen i forhold til denne matchingen
  - Man skal *vite* at om algoritmen stopper uten at matchingen er perfekt, så vil vi ut fra stoppsituasjonen kunne vise at en alternerende vei som starter i en umatchet node aldri kan nå en annen umatchet node. Altså er det ingen forbedringsvei, og derved ingen matching med flere kanter.