Hej alla,
För detta VG-spåret av projektet så kan ni gärna titta på föreläsningsanteckning nr 10 (här hittar ni hur ni kan representera grafer i datorer - antingen som närhetsmatris eller närhetslista), och titta gärna även på min föreläsningsanteckning nr 11 (här hittar ni två välkända och ofta använda graf-traverseringsalgoritmer som heter Bredden-Först och Djupet-Först) för att få en ide om vilka algoritmer och vilka datastrukturer som är användbara för projektet. Nedan hittar ni även en utökad diskussion om hur Ford Fulkerson's algoritm (som ju är en algoritm som hittar ett max-flow i ett nätverk) fungerar, det är denna algoritm för att hitta ett max-flow (och som då samtidigt hittar vår eftersökta max bipartit matchning på vår reducerade instans) som ni implementerar i vårt VG-projekt "Bipartit matchning via Max flow". Observera att i Ford Fulkerson's max-flow-algoritm så är begreppen "residual network" och "augumenting paths" viktiga att förstå vid implementationen av Ford Fulkerson's max flow algoritm. Och har ni flera frågor här så bara hör av er (kommande veckor så kommer jag att gå igenom grafalgoritmer på våra föreläsningar om ni skulle ha frågor här också ang projektet). Många lycka till! /Mvh, Mia

Låt oss nu titta på steg 2 i Ford Fulkerson algoritmen, d v s steget ”Så länge det existerar en väg p från s till t i residual graf G_f” (Notera här att en väg mellan nod s till nod t är en mängd av ”sammanhängande” bågar och dess mellanliggande noder som därmed sammankopplar dessa bågarna så att vägen mellan s-t blir sammanhängande, t ex noderna s-u-v-t utgör en väg p ifall bågarna (s,u), (u,v), (v,t) finns i grafen.)

Titta på residual network G_f och välj ut den ”svagaste länken” i G_f:
/*Residual network (eller residual graf) som vi betecknar G_f är en ny graf innehållande alla noderna i ursprungsgrafen G men med nya bågvikter, alla dessa nya bågvikterna (d v s mängden av c_f(u,v)) i residualgrafen G(f) får du fram genom följande formel: c_f(u,v):=c(u,v) – f(u,v) där c(u,v) är kapaciteten för bågen e_(u,v) mellan nodparet (u,v), och f(u,v) är aktuella flödet (d v s flow) mellan nod u och nod v. Ett residual graf G_f är en graf som intuitivt kan ses som en graf som visar mängden av tillgänglig resterande kapacitet som kan användas som flöde och som därmed kan läggas till det aktuella flödet (d v s utöka flödet, vi vill ju hitta maximum flow) ifrån nod s till t. Då är det ju givetvis så att vi, när vi tittar på hur

mycket vi kan utöka flödet längs en väg p (d v s väg ifrån s till t) måste titta på ”flaskhalsen” på denna vägen p, d v s den ”svagaste länken” på vägen p som är den båge (länk) som p g a sin kapacitet kan utökas allra minst med en ny flödesmängd (d v s minsta värdet i c_f(u,v)-värdena på vägen p i grafen G_f, och detta minsta flödet kallar vi c_f(p) i Ford Fulkerson algoritmen, detta är just steget cf (p) ← min{cf (u, v)|(u, v) is in p} som du frågar efter, d v s den lättaste bågen(d v s med minsta värdet) i vägen p i residual grafen G_f). När du har hittat detta c_f(p) så uppdaterar du nu flödet f(u,v) med detta värde på c_f(p) för alla bågar e_(u,v) som ingår i vägen p (och flödet f(v,u)) . Denna kommentar förklarar alltså vad du gör i rad 2.1 t o m 2.2.0.2) i Ford Fulkerson algoritmen i projektet). */
Nu behöver du alltså, inför nästa iteration av while-satsen (d v s rad 2), uppdatera din residual graf G_f enligt formeln c_f(u,v):=c(u,v) – f(u,v), eftersom ju f(u,v) har uppdaterats nu i steg 2.2.0.1. När vi inte längre kan hitta någon väg p (ifrån s-t) i residual grafen G_f där alla ingående bågar i p har positiva bågvikter så stoppar vi för då har vi hittat maximum flow (vi kan inte hitta flera ”utökande vägar”).*/
Tips! För att få en bra illustrerande figur på ovan nämda diskussionen, d v s en bra illustrerande figur på ett nätverk med kapaciteter samt ett flöde inritat, och dess motsvarande residual graf, titta gärna på Flow network - wikipedia (här finns ett mycket bra exempel).
Många lycka till!
Mvh,
Mia