Föreläsning 16

Uppspännande träd
Bredden förstsökning
Djupet först sökning
Minimala uppspännande träd

Definition Ett träd T är ett *uppspännande träd* för G om T är en delgraf av G som innehåller alla hörn.

Sats En graf G har (minst) ett uppspännande träd om och endast om G är sammanhängande.

Bevis: '$\Rightarrow$' Anta att G har ett uppspännande träd T.
Då finns det för varje par av hörn v, w en väg i T mellan dem.
Den vägen ligger också i G, alltså är G sammanhängande.

Bevis: '$\Leftarrow$' Anta att G är sammanhängande.
Om G inte har någon cykel så är G ett träd enligt tidigare sats.
Om G har en cykel C bilda en delgraf G_1 till G genom att ta bort en kant, vilken som helst, i C.
G_1 kommer fortfarande vara sammanhängande, och innehåller alla hörn i G.
Om G_1 saknar cykel kommer G_1 vara ett uppspännande träd för G.
Om G_1 har en cykel upprepar vi proceduren och bildar G_2.
Eftersom G har ett ändligt antal kanter måste vi till slut få ett uppspännande träd G_n. $\square$

Beskriver två olika algoritmer för att hitta uppspännande träd i en graf G.
Pseudokoder för dessa finns i kursboken.
Vi antar att vi gör detta för enkla grafer, genom att ta bort alla loopar och spar endast en av parallella kanter har vi fortfarande en sammanhängande graf.

Bredden först sökning

Ordna hörnen i V, $V = \{v_0, \ldots, v_k\}$

steg 0

Låt $T_0 = (V_0, \emptyset)$ vara ett rotat träd som bara består av ett hörn $V_0 = \{v_0\}$ och som inte har några kanter.

steg 1

Bilda trädet $T_1 = (V_1, E_1)$ av höjd 1 från T_0 genom att lägga till alla hörn i $V \setminus V_0$ som är grannar med v_0 och alla kanter mellan v_0 och dessa hörn.

De nya hörnen i steg 1 kommer ha nivå 1 i T_1

steg 2

Låt $w_1, \ldots, w_j$ vara hörnen av nivå 1 i T_1 (ordning från ovanför)

Bilda trädet $T_2 = (V_2, E_2)$ av höjd 2 från T_1 genom att först lägga till alla hörn i $V \setminus V_1$ som är grannar med w_1, och kanterna mellan w_1 och dessa grannar.

Sedan alla hörn som är grannar med w_2 **och inte lagts till ännu**, och alla kanter mellan w_2 och dessa grannar. Osv tills detta är gjort för w_j.

De nya hörnen i steg 2 kommer ha nivå 2 i T_2

Upprepa!

((steg n

Låt $\{z_1, \ldots, z_i\}$ vara hörnen på nivå $n-1$ i T_{n-1}

Bilda trädet $T_n = (V_n, E_n)$ av höjd n från T_{n-1} genom att lägga till alla hörn i $V \setminus V_{n-1}$ som är grannar med z_1, och kanterna mellan z_1 och dessa grannar.

Därefter alla hörn som är grannar med z_2 **och inte lagts till ännu**, och alla kanter mellan dem. något hörn av nivå $n-1$ i V_{n-1}, och alla kanter mellan sådana par av hörn.

De nya hörnen i steg n kommer ha nivå n i T_n))

När $V \setminus V_n = \emptyset$, dvs det inte finns några hörn kvar, är T_n ett uppspännande träd till T.

Djupet först sökning

Gör trädet så 'högt' som möjligt.

Steg 0

Låt återigen $T_0 = (V_0, \emptyset)$, $V_0 = \{v_0\}$

steg 1 skapa en väg p_1 från v_0 med följande regel:
'Gå till det grannhörn med lägst ordning som ännu inte besökts.'
Detta skapar en enkel väg $p_1 = v_0, w_1, \ldots, w_k$
Låt $T_1 = p_1$

Steg 2
'Gå bakåt (nedåt) i trädet' från slutpunkten w_k till p_1 tills du står i ett hörn som har en granne som inte är med i T_1.
Skapa en väg p_2 från detta hörn enligt samma regel som för p_1
Låt T_2 vara T_1 tillsammans med p_2.

Upprepa!
Steg n.
'Gå bakåt (nedåt) i trädet' från slutpunkten till p_{n-1} tills du står i ett hörn som har en granne som inte är med i T_{n-1}.
Skapa en väg p_n från detta hörn enligt samma regel som p_1.
Låt T_n vara T_{n-1} tillsammans med p_n.

T_n är ett uppspännande träd när alla hörn har besökts.

Definition Låt G vara en viktad graf. Ett *minimalt uppspännande träd* (billigaste uppspännande träd) T för G är ett uppspännande träd så att summan av vikterna av alla kanter i T (Kostnaden för T)

$$\sum_{e \in T} w(e)$$

är minimal bland alla uppspännande träd.

Ett minimalt uppspännande träd kan hittas genom
Prim's algoritm
steg 0 Låt $T_0 = (V_0, \emptyset)$, $V_0 = \{v_0\}$
steg 1 Bilda T_1 genom att lägga till kanten $e = (v_0, v')$ som har lägst vikt av alla kanter med v_0 som ena ändpunkt och den andra ändpunkten $v' \neq v_0$.
$T_1 = (\{v_0, v'\}, \{e\})$.

steg 2 Bilda T_2 genom att lägga till den kant (w, w') som har minimal vikt av alla kanter med ett hörn i $w \in T_1$ och $w' \notin T_2$. Lägg även till w'. Upprepa!

Fortsätt tills T_n innehåller alla hörn. Då är T_n ett minimalt uppspännande träd.

Steg n

Bilda T_n genom att lägga till den kant (w, w') som har minimal vikt av alla kanter med ett hörn i $w \in T_{n-1}$ och $w' \notin T_n$. Lägg även till w'.

Vår version av Prim's algoritm har worst-case tid $\Theta(n^3)$, men det går att förbättra till $\Theta(n^2)$.