College 8, Opspannende bomen

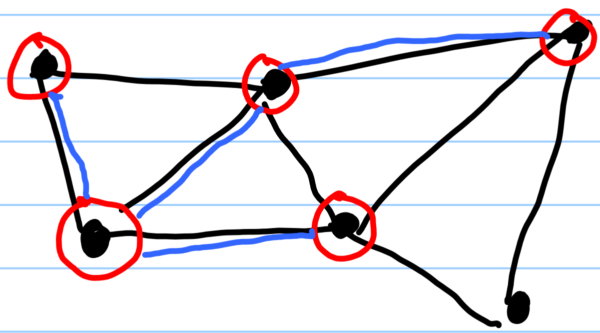
Gegeven is een ongerichte graaf $G=(V,E)$

<u>Definitie</u>: Boom = samenhangende graaf $G'=(V',T)$ zonder kringen, $V' \subseteq V$, $T \subseteq E$
$|V'|$ knopen $\Rightarrow$ $|V'|-1$ kanten.

Als de boom alle knopen in G bevat, dwz als $V'=V$, dan is de boom een <u>opspannende</u> boom in G

Voorbeeld boom:
(geen opspannende)

$G=(V,E)$



O : V'
— : T

Het "minimum spanning tree" (MST) probleem:

Gegeven een ongerichte volledige graaf $G=(V,E)$ en een symmetrische afstandenmatrix $[d_{ij}]$, bepaal een boom (V,T) waarbij

$$\sum_{[i,j]\in T} d_{ij}$$

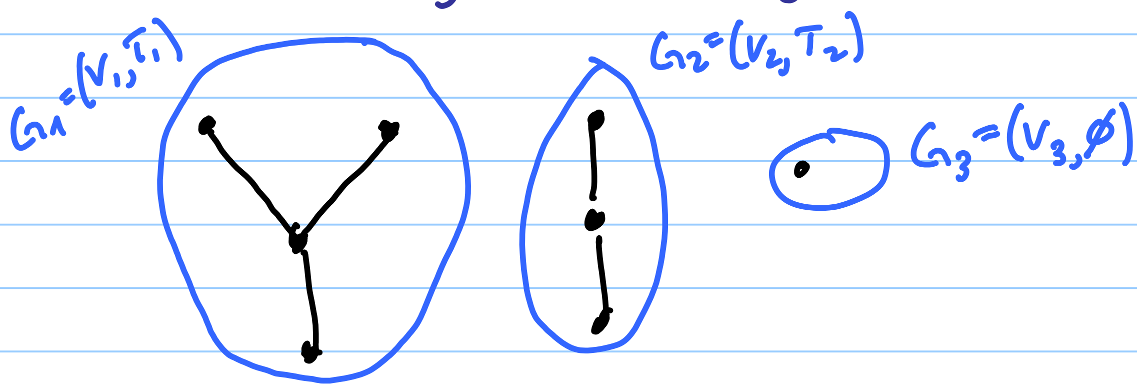
minimaal is.

Oorspronkelijke graaf niet volledig $\Rightarrow$
Zet $d_{ij} = \infty$ als $[i,j] \notin E$.

Het MST-probleem kent vele toepassingen en is ook en theoretisch interessant probleem.

Toepassingen: Vooral netwerk-ontwerp problemen: Bepaal een zo goedkoop mogelijk netwerk dat alle "knopen" met elkaar verbindt.
"Knopen" = telefooncentrales, computer-servers, etc.

Definitie: Een forest ("bos") is een verzameling knoop-disjuncte bomen.



Hoe kunnen we algoritmisch een opspannende boom "bouwen"?

Maak gebruik van de volgende mooie eigenschap!

(Stelling 12.1 in het boek)

Gegeven is $G=(V,E)$.

Stelling. Zij $\{(U_1,T_1),(U_2,T_2),\ldots,(U_k,T_k)\}$ een bos die V spant, en $[u,v]$ de kortste kant met precies een eindpunt in U_1. Onder alle opspannende bomen die alle kanten in $T=\bigcup_{i=1}^{k} T_i$ bevatten, is er een minimale opspannende boom die $[u,v]$ bevat.

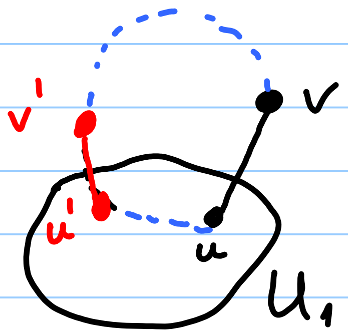
Bewijs:

Stel, er is een opsp. boom (V,B) met

$$B \supseteq T \text{ en } [u,v] \notin B$$

die korter is dan alle opsp. bomen die T en $[u,v]$ bevatten.

Voeg $[u,v]$ toe aan B $\Rightarrow$ we krijgen nu een kring!



NB! kring bevat niet alleen U_1-knopen, want $v \notin U_1$

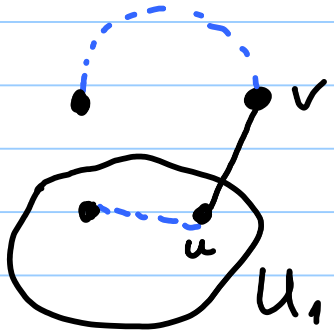
Daardoor bestaat er een kant $[u',v'] \notin T$ met $u' \in U_1$ en $v' \in V \setminus U_1$, en $[u,v] \neq [u',v']$

Omdat $[u,v]$ de kortste kant is met precies een eindpunt in U_1 geldt

$$d_{uv} \leq d_{u'v'}$$

Maak nieuwe boom (V, B') met

$$B' = B \cup \{[u,v]\} \setminus \{[u',v']\}$$



B' bevat T en $[u,v]$. De lengte van B' is kleiner dan of gelijk aan de lengte van B.

$\blacksquare$

Deze stelling kunnen we nu algoritmisch gebruiken!

Algoritme 1 (Jarnik (1930), Prim (1957), Dijkstra (1959))

Begin met $F = \{(v_1, \emptyset), (v_2, \emptyset), \ldots, (v_{|V|}, \emptyset)\}$, $U = \{v_1\}$ (alleen geïsoleerde knopen)

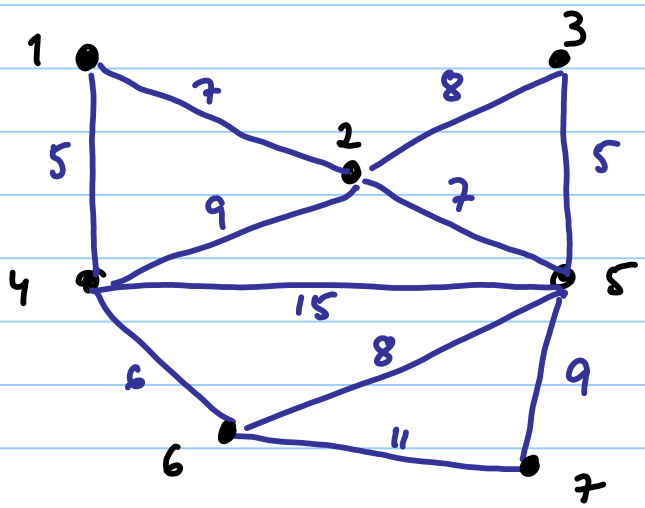
Er bestaat een MST die de kortste kant grenzend aan v_1 bevat (zie stelling), zeg $[v_1, v_2]$

$\Rightarrow F = \{ (v_1, v_2, [v_1, v_2]), (v_3, \emptyset), \ldots, (v_{|V|}, \emptyset) \}$

Vervolgens, voeg de kortste kant grenzend aan v_1 <u>of</u> v_2 (maar <u>niet</u> $[v_1, v_2]$) toe aan het eerste "komponent". Ga zo door tot dat alle knopen in een "komponent" zit. De resulterende graaf vormt een MST.

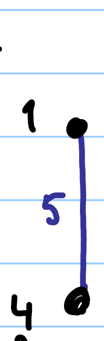
<u>Correctheidsbewijs:</u> Stelling 12.1 en inductie.

<u>Voorbeeld</u>:



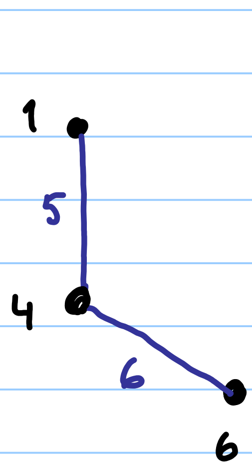
Iteratie

1



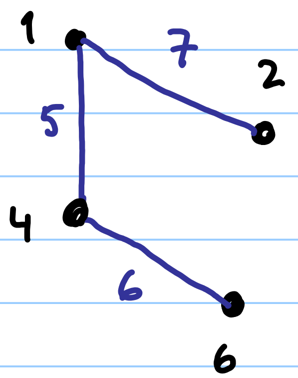
$u = \{1, 4\}$

2



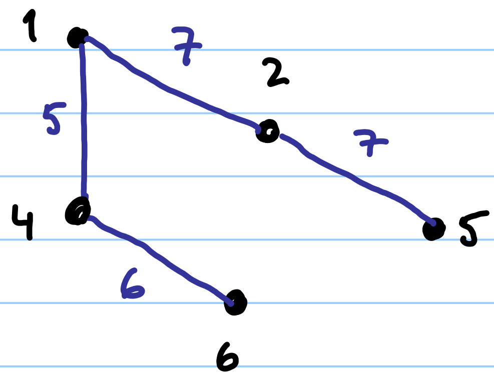
$u = \{1, 4, 6\}$

3



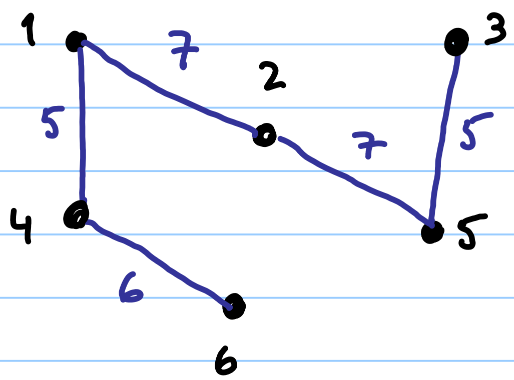
$U = \{1, 4, 6, 2\}$

4



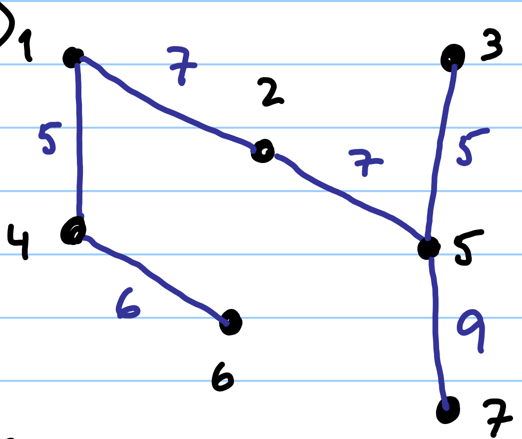
$U = \{1, 4, 6, 2, 5\}$

5



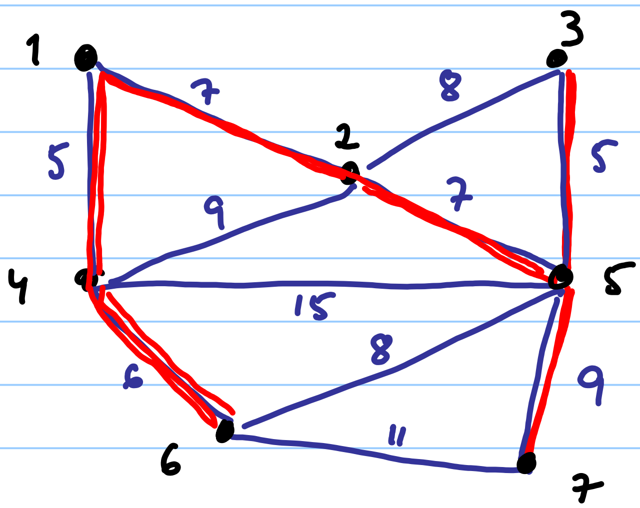
$U = \{1, 4, 6, 2, 5, 3\}$

6



$U = \{1, 4, 6, 2, 5, 3, 7\} = V$

lengte van MST = 39.



Wat is de runtime van Prim's algoritme?

Kijk naar implementatie

Voor iedere $v \in V \setminus U$: bepaal punt in U die het dichtstbij v is, en de bijbehorende lengte v/d edge.
[closest[v], d(v, closest[v])]

Bij begin: $U = \{1\}$

$$\left.\begin{array}{ll} 2: [1, 7] & 5: [1, \infty] \\ 3: [1, \infty] & 6: [1, \infty] \\ 4: [1, 5] & 7: [1, \infty] \end{array}\right\} O(|V|)$$

$\Rightarrow$ Initialiseren : $O(|V|)$ tijd.

In elke iteratie:

* Loop de lijst

v: [closest[v], d(v, closest[v])]) $\forall\, v \in V \setminus U$

door om kleinste d(v, closest[v]) te vinden.

Noem de vertex met de kleinste d(v, closest[v]) "nieuw". $U := U \cup \{\text{nieuw}\}$

Dit kost $O(|V|)$ tijd.

Bv. iteratie 1:

2: [1, 7]	5: [1, ∞]
3: [1, ∞]	6: [1, ∞]
→ 4: [1, 5]	7: [1, ∞]

"nieuw = 4 $U := U \cup \{4\} = \{1, 4\}$

* Stel [closest[v], d(v, closest[v])] bij :
Als d(v, closest[v]) > d(v, nieuw) :

d(v, closest[v]) := d(v, nieuw)
closest[v] := nieuw

Dit kost $O(|V|)$ tijd

Bv. in iter. 1:

2: [1, 7]	5: [4, 15]	bijgesteld
3: [1, ∞]	6: [4, 6]	
weg → ~~4: [1, 5]~~	7: [1, ∞]	

Samengevat : Initialiseren : $O(|V|)$
In elke iteratie : $O(|V|)$
$|V|-1$ iteraties
$\Rightarrow$ $O(|V|^2)$ tijd.

Algoritme 2 (hoofdst. 12.2)

Een "slimmere" versie vergeleken met Alg. 1. In plaats van alleen één component (u) uit te breiden met een nieuwe kant, breid alle samenhangende componenten uit tegelijk.

$\Rightarrow$ $O(|E| \cdot \log |V|)$ algoritme

$\Rightarrow$ asymptotisch beter dan Alg. 1 als we minder dan $\Theta(|V|^2 / \log |V|)$ kanten hebben in de graaf.

Zie werkcollege W8.

Algoritme 3 (Kruskal (1956))

Dit is een "greedy" - algoritme

Greedy: "maak de lokaal optimale keuze in elke iteratie."

Bekijk dit algoritme vanuit het algemenere probleem "maximale gewogen bos" oftewel "maximum weight forest":

$G = (V, E)$, kant $[v_i, v_j] \in E$ heeft
gewicht $w_{ij} \geq 0$

Twee mogelijkheden: G wel of niet samenhangend.

a) Stel $G = (V, E)$ samenhangend
Omdat $w_{ij} \geq 0$ kan elke optimale MWF maximaal worden gemaakt.
Een maximale bos is een boom

Vertaling MWF -> MST :

$$W = \max_{i,j} \{w_{ij}\}$$

$$d_{ij} := W - w_{ij} \qquad (1)$$

MST onder d_{ij} $\forall (i,j) \in E$ (1)
$\Leftrightarrow$
MWF onder w_{ij} $\forall (i,j) \in E$

b) $G = (V, E)$ niet samenhangend

MWF onder w $\Leftrightarrow$
vereniging van de MSTs van alle samenhangende componenten van G onder d_{ij} $\forall (i,j) \in E$ (1).

Greedy voor MWF:

Begin met $F = \emptyset$
Orden alle kanten in E in orde van niet-toenemende lengte.

while $E \neq \emptyset$ (of $|F| < |V| - 1$)
  Neem kant $[u,v]$ in E met grootste gewicht
  en laat $E := E \setminus [u,v]$
  Als $[u,v]$ geen kring creëert, $F := F \cup [u,v]$

Rekentijd

Initialiseren: Orden alle kanten:
$O(|E| \cdot \log |E|)$

In elke iteratie: Controleer of u en v in dezelfde samenhangende component zitten.
$O(|T|) = O(|V|)$
Aantal iteraties: $O(|E|)$

$\Rightarrow$ Rekentijd $= O(|E| \cdot |V|)$
(Dat domineert $O(|E| \cdot \log |E|)$)

Waarom is MWF interessant?

Het MWF probleem is een zogenaamde matroïde-probleem.

Één belangrijke reden waarom matroïden interessant zijn:

Stelling: Problemen op matroïden worden optimaal opgelost m bv Greedyalgoritmen.

De rest van de aantekeningen is voor de geïnteresseerde lezer!!!

Definitie: Een paar $(S, \mathcal{F})$ bestaand uit een eindige verzameling S en een niet-lege collectie $\mathcal{F}$ van deelverzamelingen van S noemen we een matroïde als $(S, \mathcal{F})$ voldoet aan

(i) $T \in \mathcal{F}$ en $U \subseteq T \Rightarrow U \in \mathcal{F}$

(ii) als $T, U \in \mathcal{F}$ en $|T| < |U|$, dan is $T \cup \{u\} \in \mathcal{F}$ voor een element $u \in U \setminus T$

De collectie $\mathcal{F}$ noemen we de independent sets van de matroïde

Er zijn verschillende soorten matroïden, bv:

a) "Matric Matroids": A: $m \times n$ matrix
S = verzameling kolommen van A
$\mathcal{F}$ = collectie van lineair onafhankelijke kolommen van A

b) "Graphic Matroids" : $G=(V,E)$
Laat $F \in \mathcal{F}$ als $G_F=(V,F)$ geen kring bevat. Dus $\mathcal{F}$ is de collectie van bossen van G.

c) "Partition Matroids": Zij Π een partitie van S; $\Pi=\{S_1, S_2, \ldots, S_p\}$ zdd $S_i \cap S_j = \emptyset \ \forall \ i \neq j$, en $\bigcup_{i=1}^{p} S_i = S$

Een deelverzameling $I \in \mathcal{F}$ d.e.s.d.a $|I \cap S_i| \leq 1 \quad \forall \ i=1,\ldots,p$

Het paar $M_\Pi = (S, \mathcal{F})$ is dan een partition matroid.