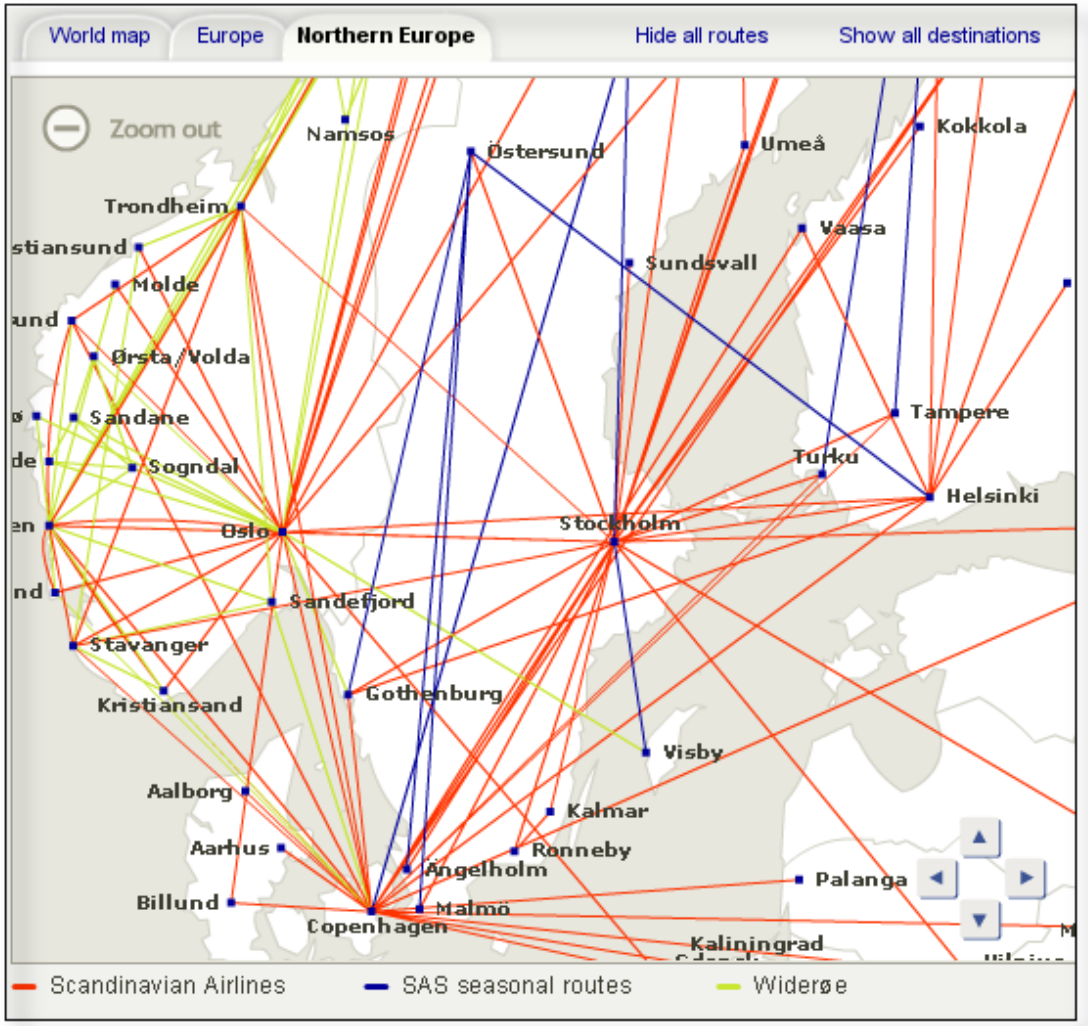
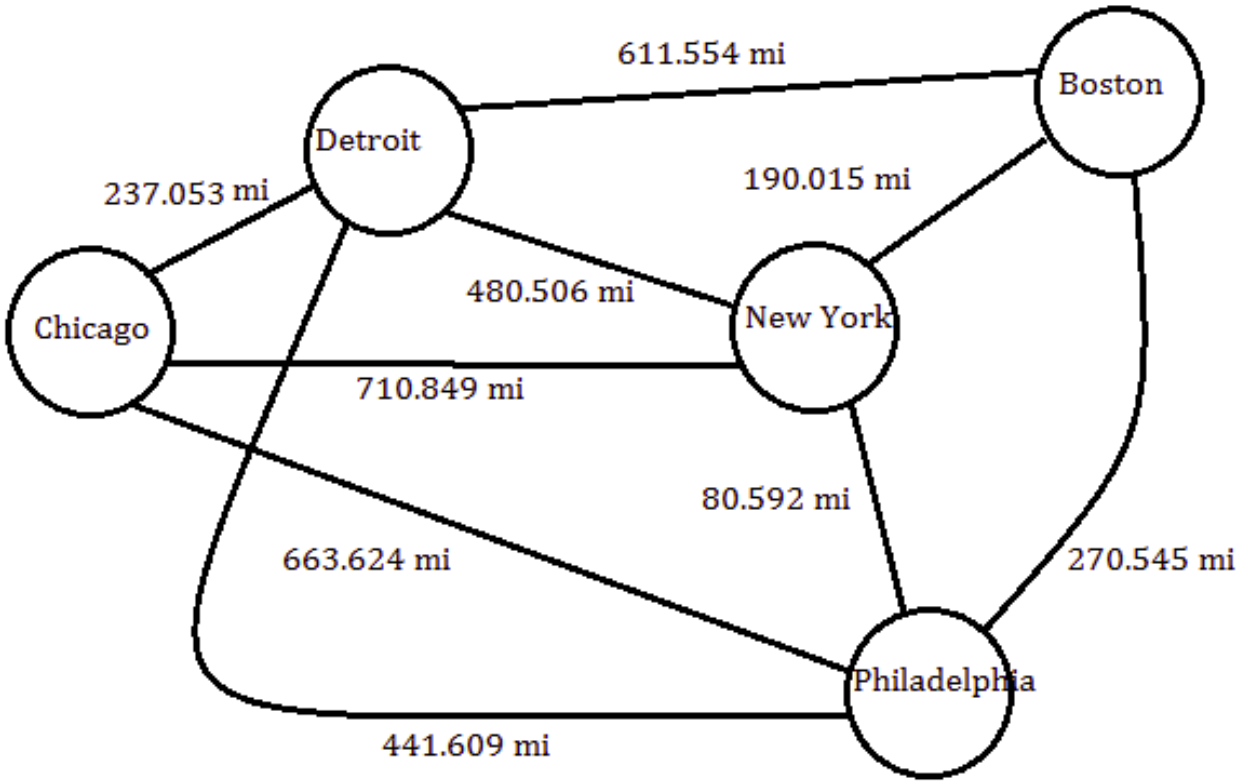
Grafer


World map
Europe
Northern Europe
Hide all routes
Show all destinations
Zoom out
Namsos
Östersund
Umeå
Kokkola
Trondheim
Vaasa
Sundsvall
Molde
Ørsta/Volda
Sandane
Tampere
Sogndal
Turku
Helsinki
Oslo
Stockholm
Sandefjord
Stavanger
Gothenburg
Kristiansand
Visby
Aalborg
Kalmar
Aarhus
Ronneby
Ängelholm
Palanga
Billund
Malmö
Copenhagen
Kaliningrad
Scandinavian Airlines
SAS seasonal routes
Widerøe

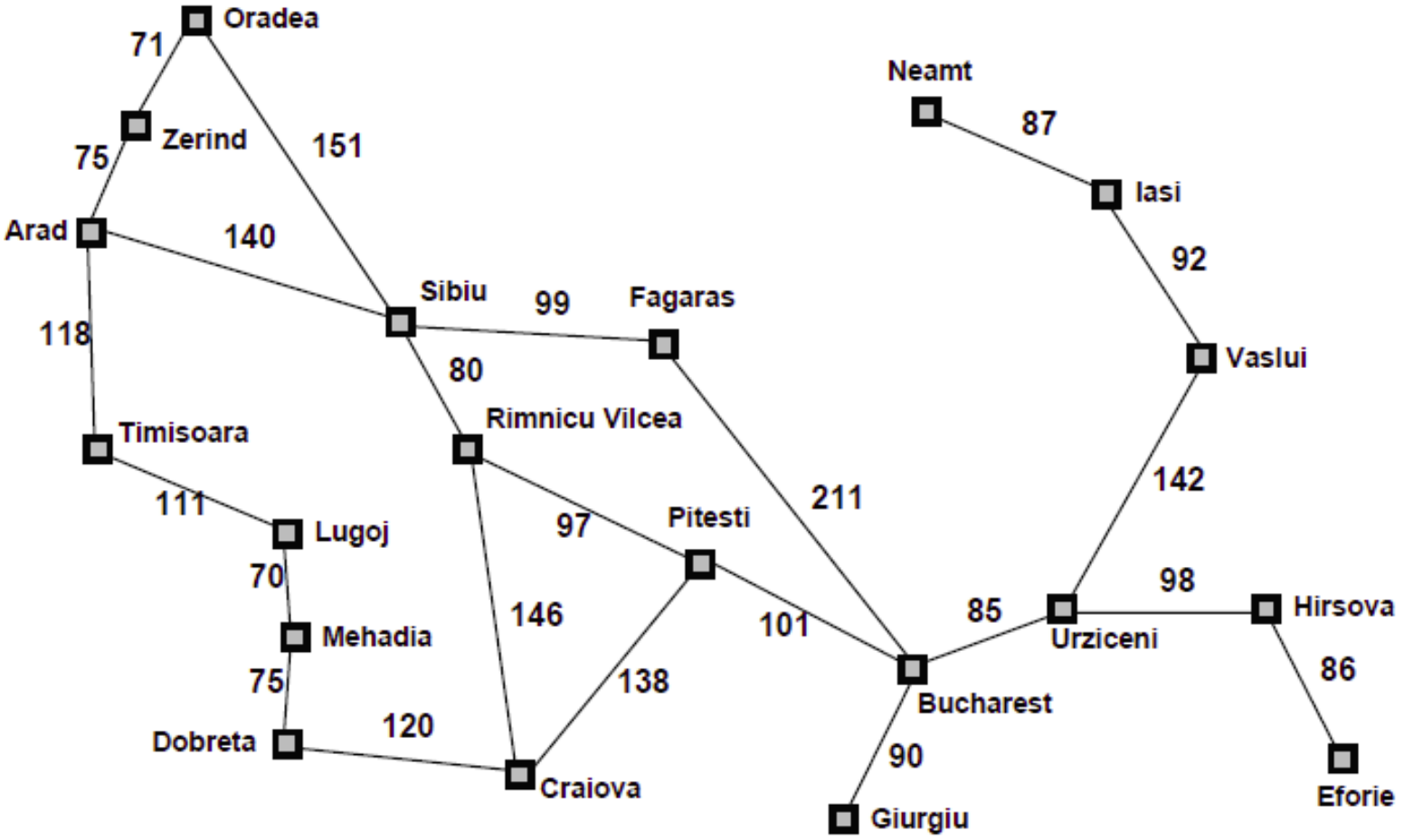
Anvendelser av grafer

- Brukes for datasett med ikke-lineære og ikke-hierarkiske forbindelser mellom dataobjektene
- Forbindelsene i en graf er ofte usystematiske
- Typisk anvendelser er modellering av nettverk:
  - Veisystemer/rutekart
  - Prosesser
  - Sosiale forhold
  - Organisasjonsmodeller
  - Internett / LAN

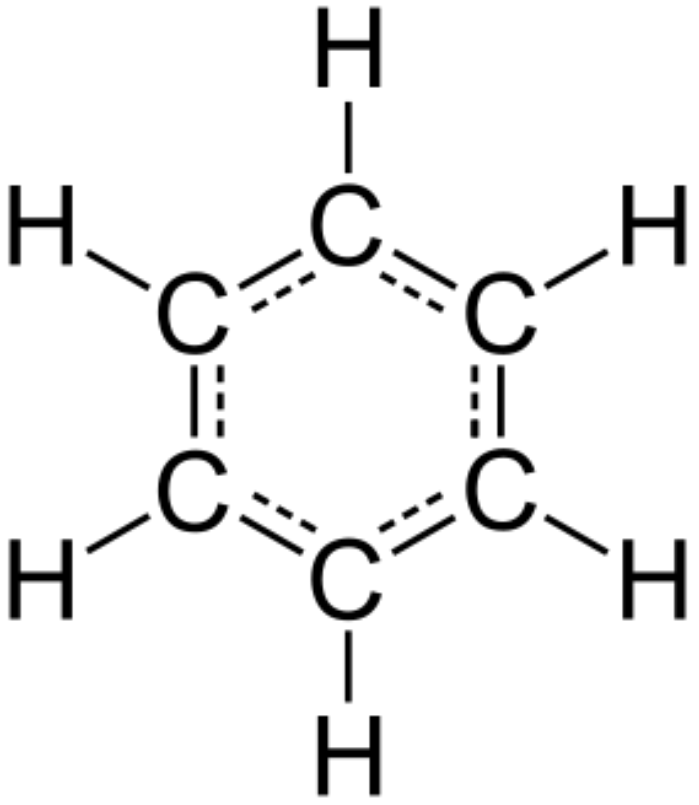
Flyruter


611.554 mi
Boston
Detroit
237.053 mi
190.015 mi
480.506 mi
New York
Chicago
710.849 mi
80.592 mi
663.624 mi
270.545 mi
Philadelphia
441.609 mi

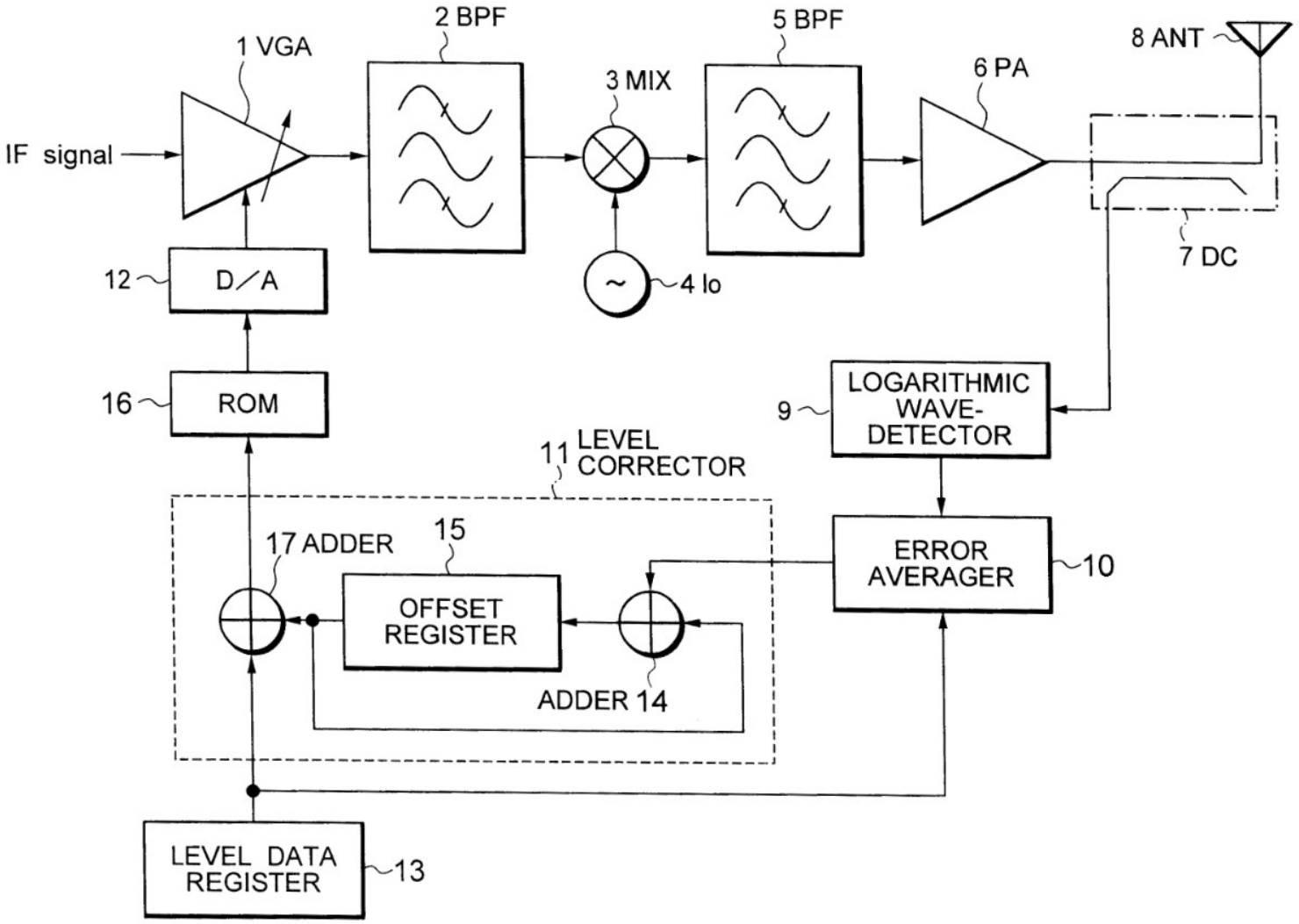
Veikart med avstandsangivelse



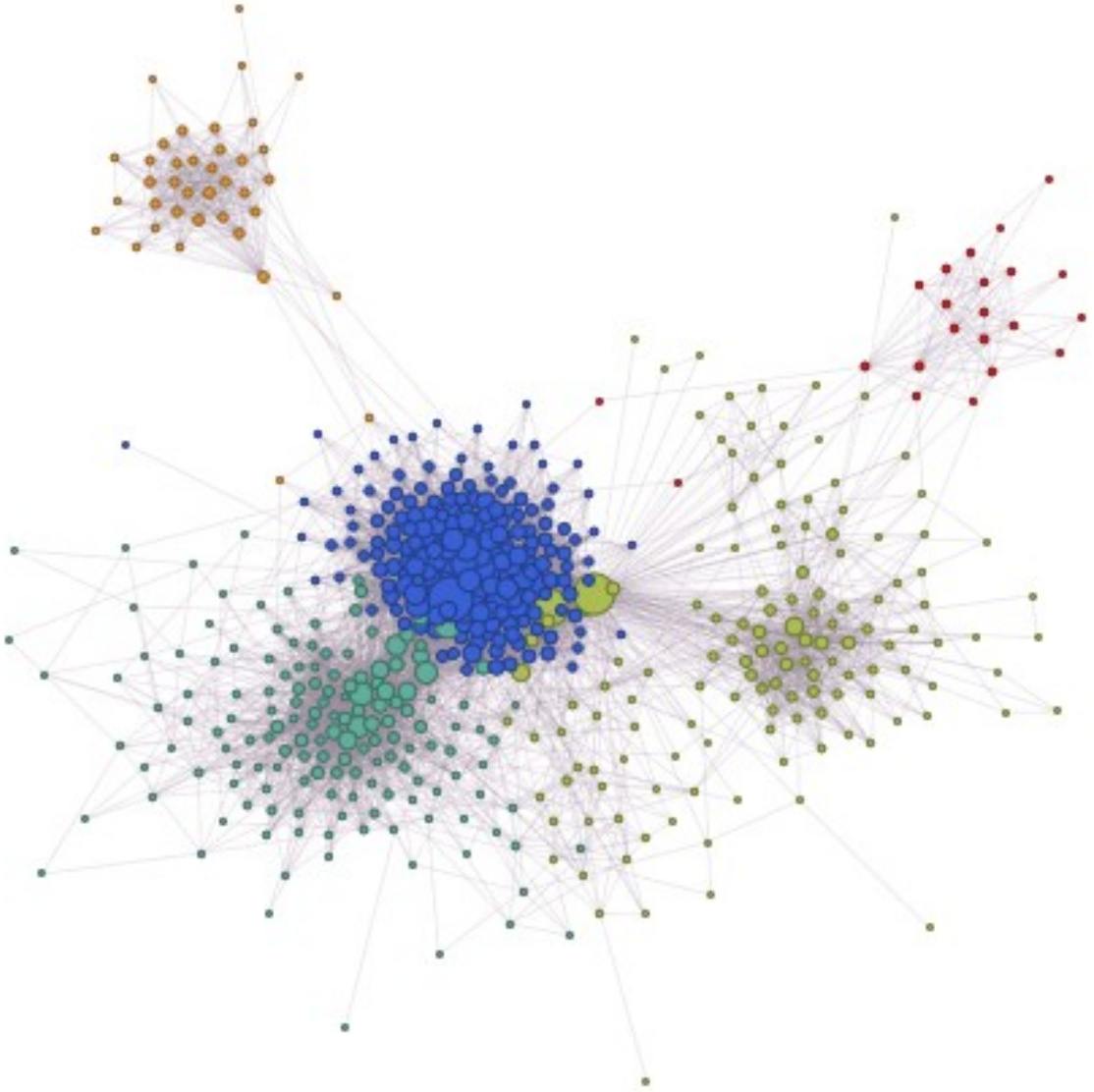
Molekylmodell



Nettverk for signalbehandling



Venner på Facebook



Grafer: Definisjon

- En graf *G* består av en mengde *V* med noder (vertices) sammen med en mengde *E* med kanter (edges), $G = (V, E)$
- En kant er et par (v, w) av noder, som angir at det er en forbindelse mellom nodene
- Hvis det er en kant mellom nodene *v* og *w*, dvs. at $(v, w) \in E$, sier vi at de to nodene er naboer i grafen *G*

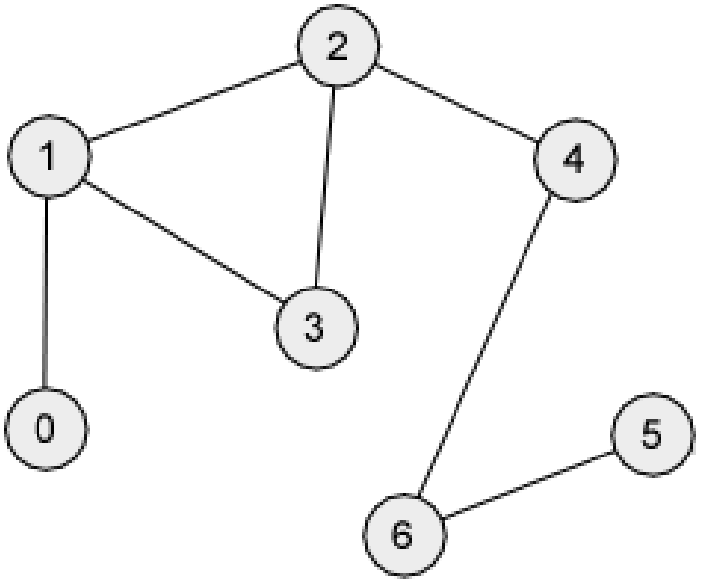
Rettede og urettede grafer

- En graf er rettet hvis kantene er ordnede par:
  - Kantene er énveis
  - En kant (*v*, *w*) angir at det er en forbindelse fra *v* til *w,* men *ikke* fra *w* til *v*
- Grafen er urettet hvis kantene ikke er ordnet:
  - Kantene er toveis
  - En kant (*v*, *w*) angir at det er en forbindelse både fra *v* til *w og* fra *w* til *v*

Sammenhengende urettede grafer

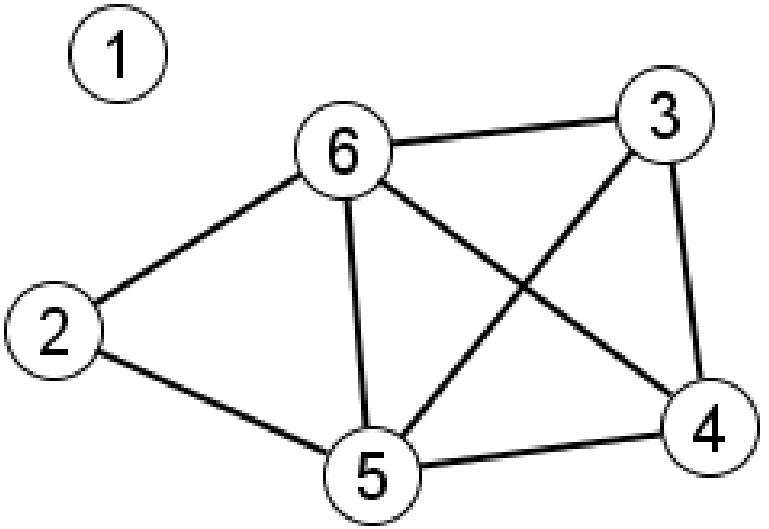
- En vei i grafen er en sekvens av noder der hver node er nabo til neste node i sekvensen
- En cykel er en vei med minst 3 noder, der siste node på veien er nabo med første
- En urettet graf er sammenhengende hvis det finnes en vei fra enhver node i grafen til enhver annen node, ellers er den usammenhengende
- En urettet graf kalles for et tre hvis den er sammenhengende og uten cykler

Sammenhengende, urettet graf



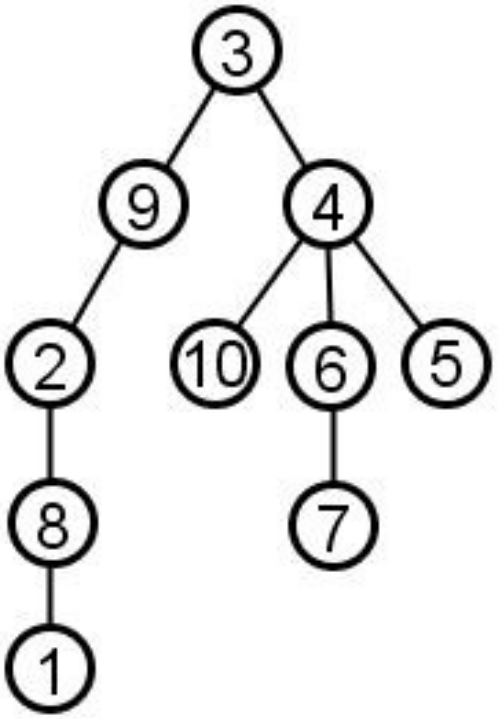
- Noder: 0, 1, 2, 3, 4, 5, 6
- Kanter: (0, 1), (1, 2), (1, 3), (2, 3), (2, 4), (4, 6), (5, 6)
- Vei: 0, 1, 2, 4, 6, 5
- Cykel: 1, 3, 2

Usammenhengende, urettet graf



Tre

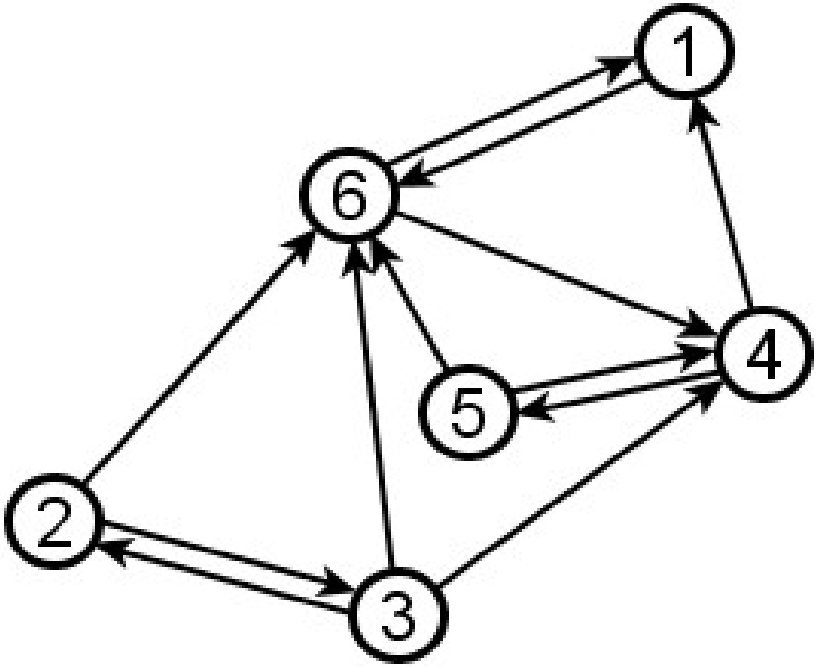
Grafen er sammenhengende og uten cykler: Et tre



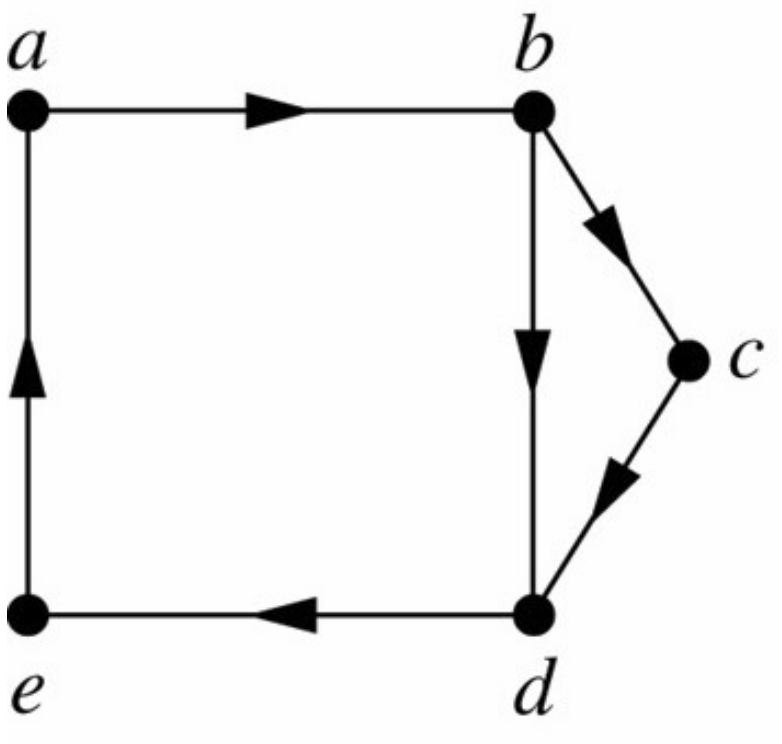
Samenhengende rettede grafer

- Kantene i grafen har en *retning*:
  - Må ha *to* kanter for å ha forbindelse begge veier mellom to noder
- Veiene i grafen blir rettede (har en retning)
- En rettet graf er sterkt sammenhengende hvis det finnes en rettet vei fra enhver node til enhver annen node i grafen
- En rettet graf er svakt sammenhengende hvis det finnes en vei fra enhver node til enhver annen node i den tilsvarende urettede grafen

Svakt sammenhengende rettet graf



Sterkt sammenhengede rettet graf

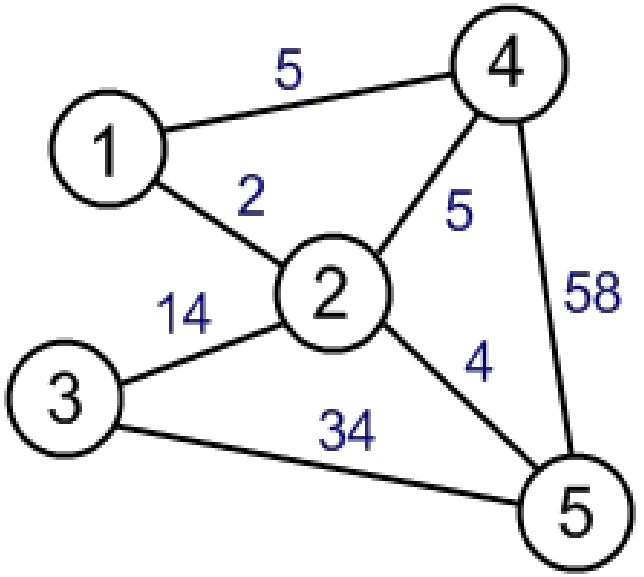


Vektede grafer

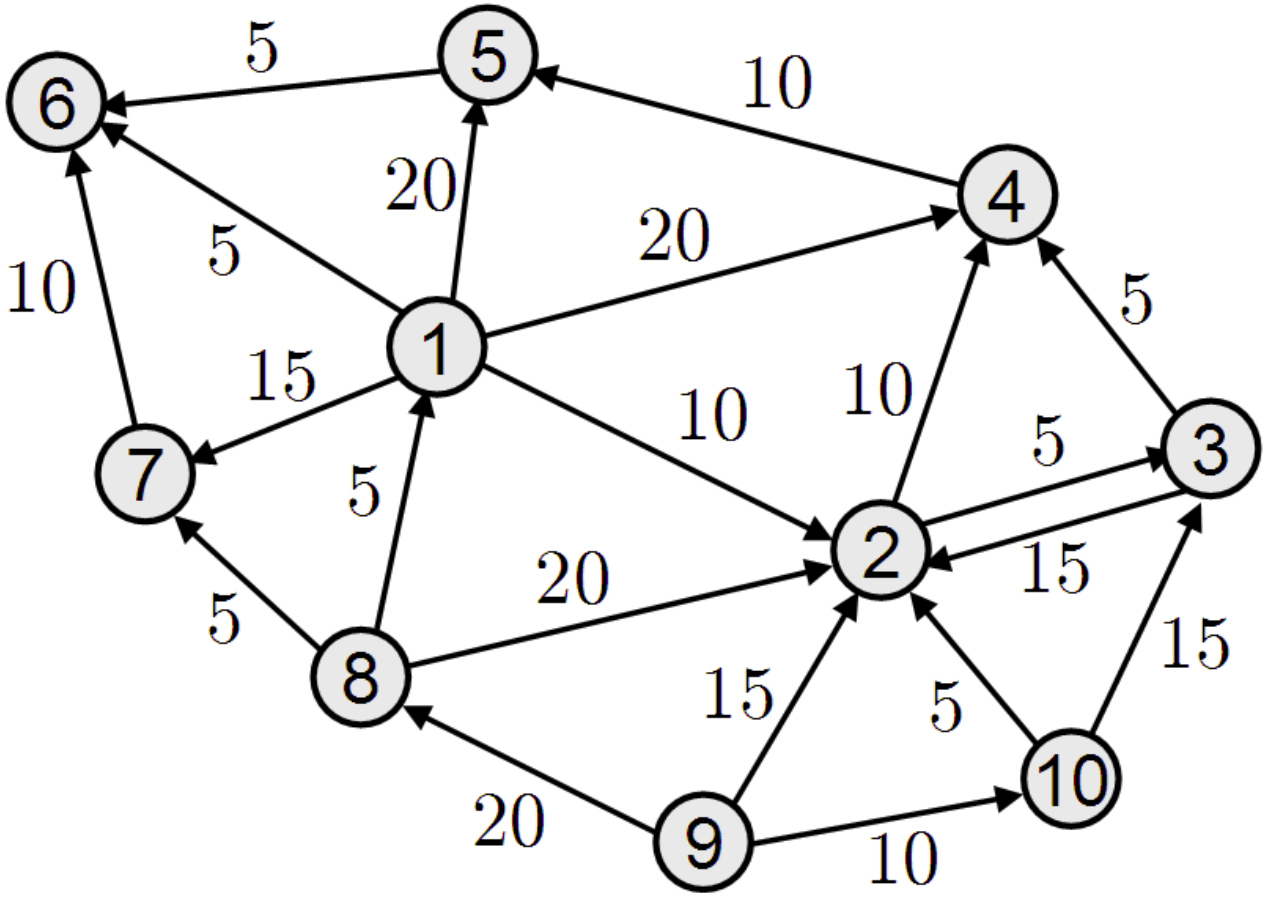
- Hver kant har en *lengde* (eller *kostnad*)
- En kant består av tre deler:
  - To noder
  - En verdi som angir kantlengden
- En vektet graf kan være enten rettet eller urettet
- Veiene i grafen har en *vektet veilengde**:
  - Summen av lengdene av alle kantene langs veien

*: Uvektet veilengde er *antall* kanter langs veien

Urettet vektet graf



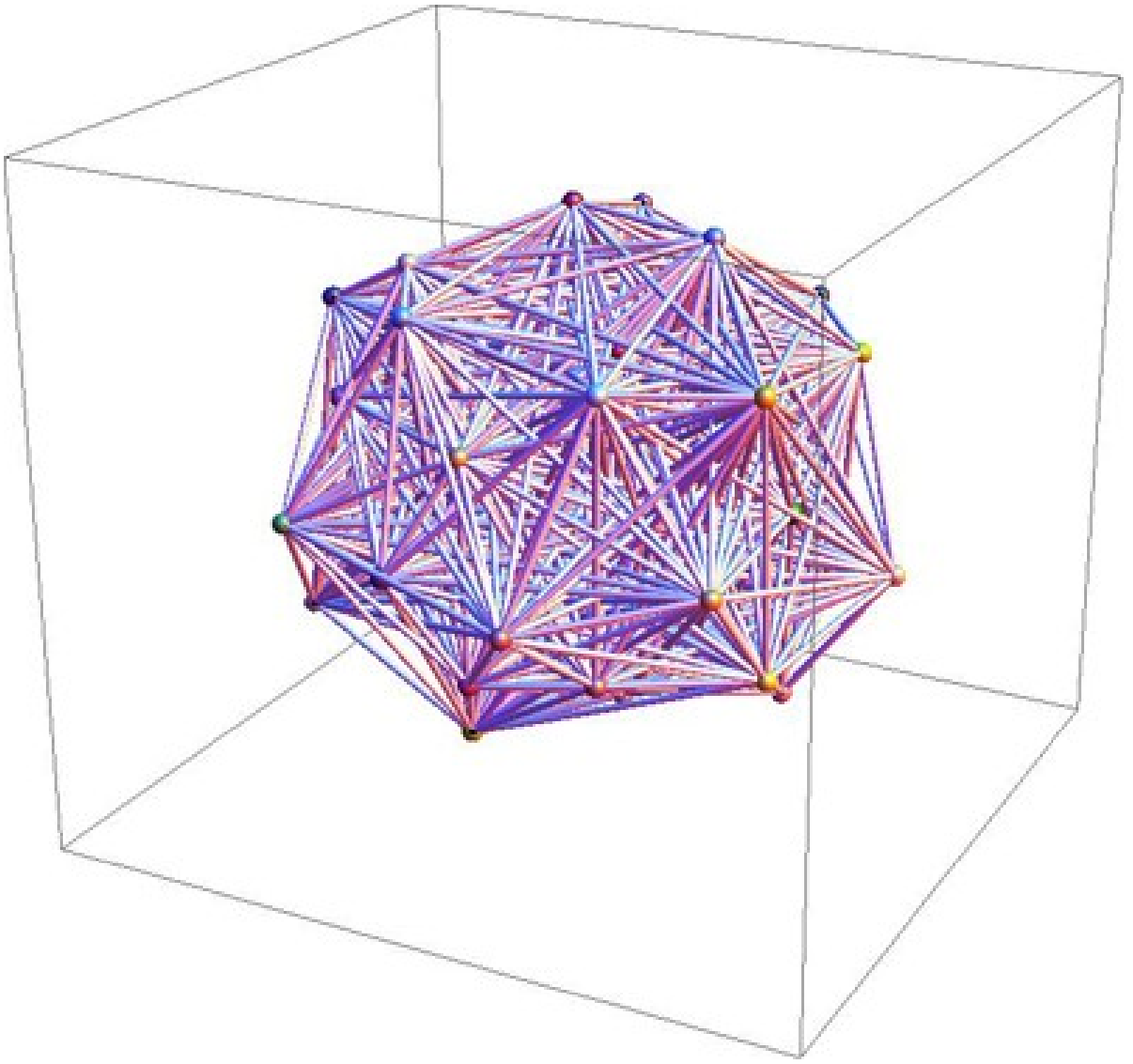
Rettet vektet graf



Antall kanter i en graf med *n* noder

- Anta at det aldri er mer enn én kant fra en node til enhver annen node i grafen (?)
- Maksimalt antall kanter er da n^2
- Grafer og nettverk er oftest "tynt befolket", dvs, at antall kanter er mye mindre enn n^2, vanligvis $O(n)$
- Sparse graph: Antall kanter er $O(n)$
- Dense graph: Antall kanter er $O(n^2)$
- Viktig at implementasjon av grafer og grafalgoritmer er effektive når grafen er "sparse"

3-D dense graph



Datastruktur for grafer

- Trenger å lagre:
  - Dataene i hver node
  - Hvilke noder som er naboer i grafen (kantene)
  - Evt. lengde/kostnad for hver kant for vektet graf
- Rettede og urettede grafer lagres på samme måte:
  - En urettet kant (*v*, *w*) lagres som de to rettede kantene (*v*, *w*) og (*w*, *v*)
- To standard måter å lagre grafen på:
  - Nabomatrise
  - Nabolister

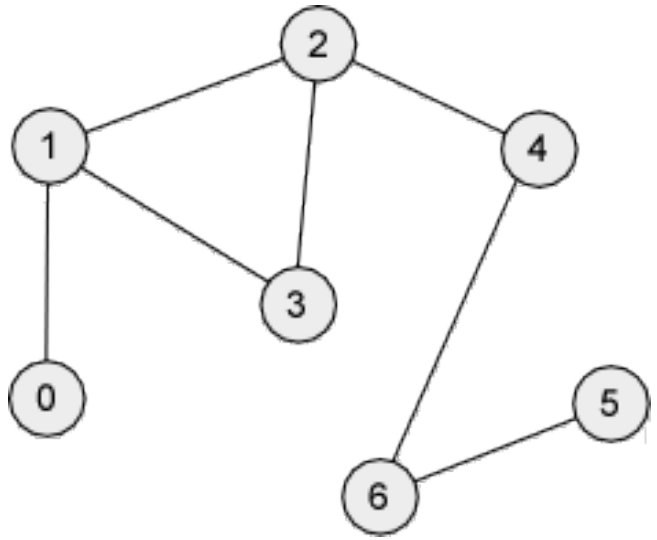
Nabomatriser *

- Nummererer de *n* nodene i en graf, fra 0 til *n* – 1
- Hele grafen kan representeres med en to-dimensjonal boolsk *n* x *n* tabell/matrise G der:
  - `G[i][j]` er `true` hvis og bare hvis det går en kant fra node `i` til node `j`, og `false` ellers
- For vektede grafer kan det brukes en matrise med kantlengder (heltall eller reelle tall), der:
  - `G[i][j]` er lik kantlengden hvis det går en kant fra node `i` til node `j`, ellers lik "uendelig" (ingen kant)

*: Engelsk: "adjacency matrices"

Nabomatrise for urettet graf

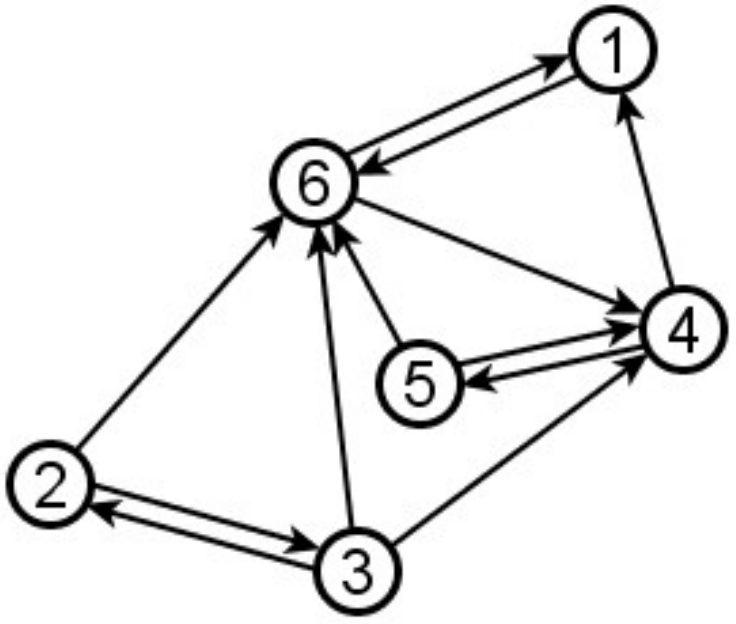
For urettede grafer blir nabomatrisen symmetrisk om hoveddiagonalen, `G[i][j] = G[j][i]` (redundans)



	0	1	2	3	4	5	6
0	T	**T**	F	F	F	F	F
1	**T**	T	**T**	**T**	F	F	F
2	F	**T**	T	**T**	**T**	F	F
3	F	**T**	**T**	T	F	F	F
4	F	F	**T**	F	T	F	**T**
5	F	F	F	F	F	T	**T**
6	F	F	F	F	**T**	**T**	T

Nabomatrise for rettet graf

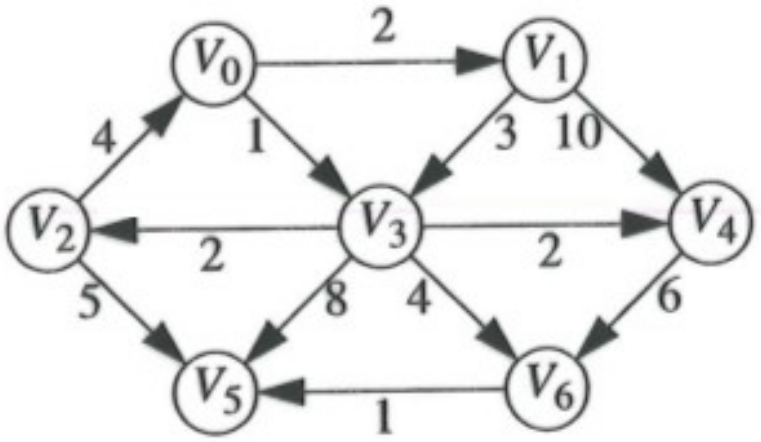
Rettede grafer har usymmetrisk nabomatrise



	1	2	3	4	5	6
1	T	F	F	F	F	**T**
2	F	T	**T**	F	F	**T**
3	F	**T**	T	**T**	F	**T**
4	**T**	F	F	T	**T**	F
5	F	F	F	**T**	T	**T**
6	**T**	F	F	**T**	F	T

(nodene nummereres med start i 1 i figuren ovenfor)

Nabo- / kantmatrise for vektet graf



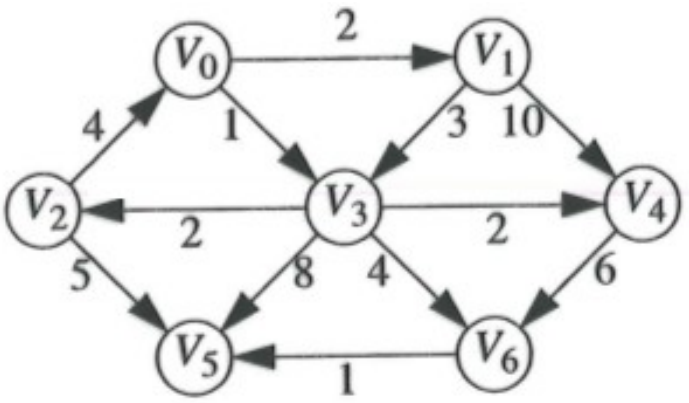
	0	1	2	3	4	5	6
0	0	**2**	∞	**1**	∞	∞	∞
1	∞	0	∞	**3**	**10**	∞	∞
2	**4**	∞	0	∞	∞	**5**	∞
3	∞	∞	**2**	0	**2**	**8**	**4**
4	∞	∞	∞	∞	0	∞	**6**
5	∞	∞	∞	∞	∞	0	∞
6	∞	∞	∞	∞	∞	**1**	0

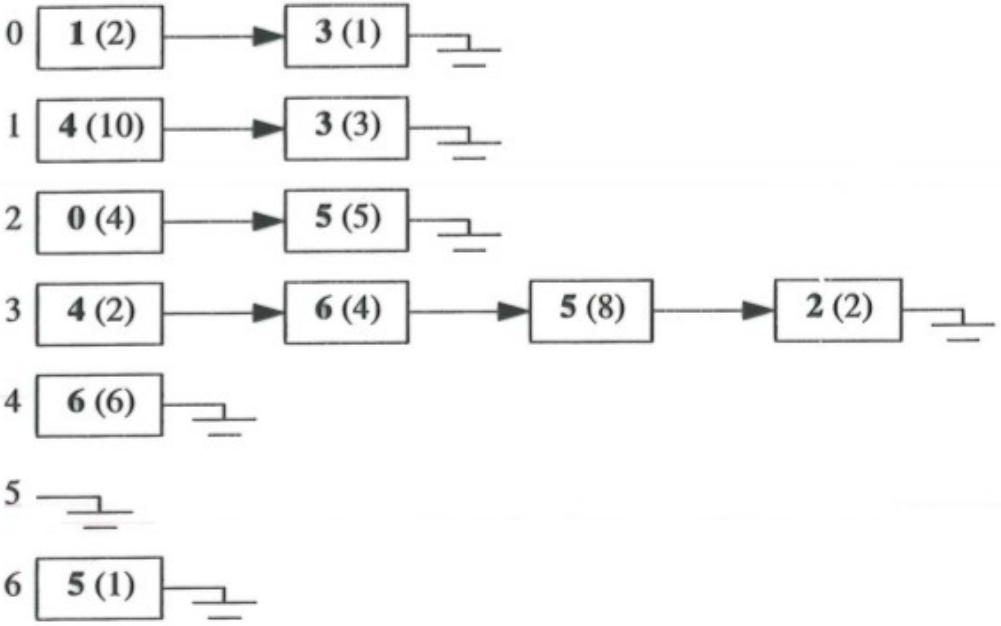
Nabolister *

- Nabomatriser bruker $O(n^2)$ hukommelse, sløsing med plass siden de fleste grafer er “sparse”
- Nabolister:
  - Grafen representeres som en array med lister, én liste for hver node i grafen
  - Listen for en node inneholder nodens direkte naboer i grafen
  - Rettede og urettede grafer kan behandles likt, urettede kanter lagres to ganger
- Nabolisten for en “sparse” graf med med n noder krever $O(n)$ hukommelse

Engelsk: “adjacency lists / edge lists”

Nabolister for en rettet, vektet graf

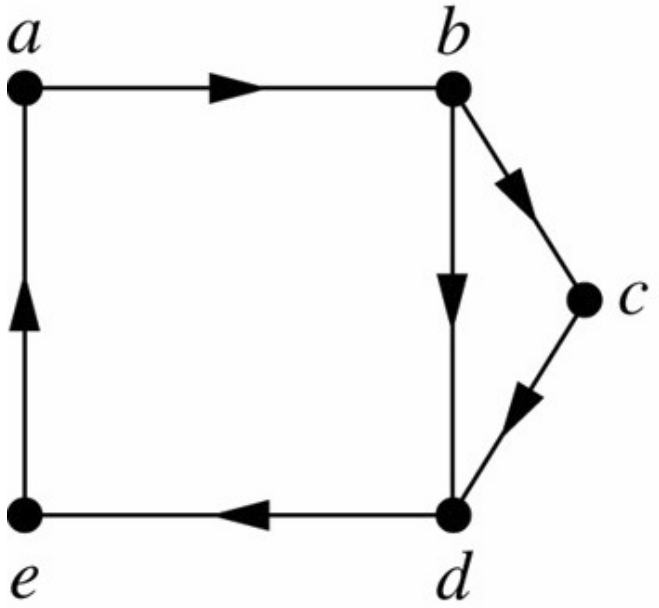




Lagring av grafdata på filer

- Naboliste-formatet er velegnet til lagre en graf som en tekstfil
- Nodene nummereres fra 0 til $n - 1$
- Legger antall noder i grafen, *n*, først på filen
- Deretter én linje for hver node, med:
  - Nodenummer
  - Dataene som er lagret i noden
  - Antall naboer
  - En liste med nodenummere for alle naboene
  - I tillegg kantlengder for *vektede* grafer

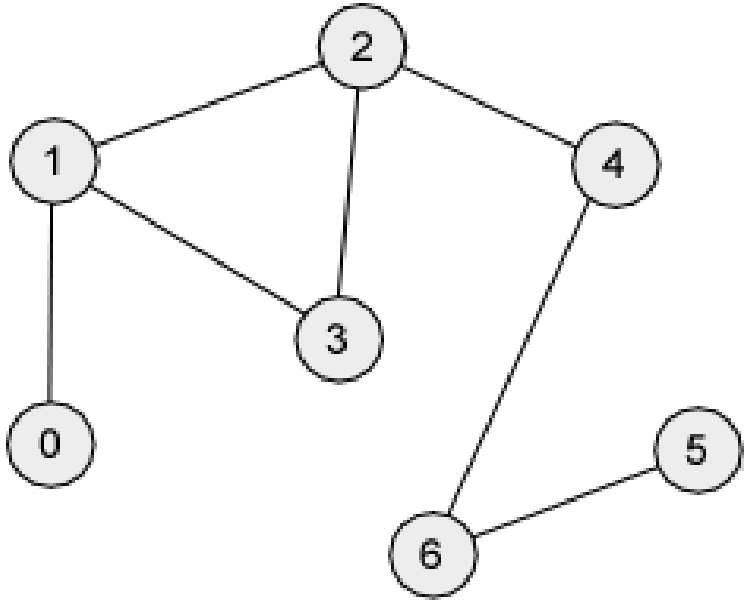
Eksempel, lagring av rettet graf




```
5
0 a   1     1
1 b   2     2 3
2 c   1     3
3 d   1     4
4 e   1     0
```

Fil: graf_5.txt

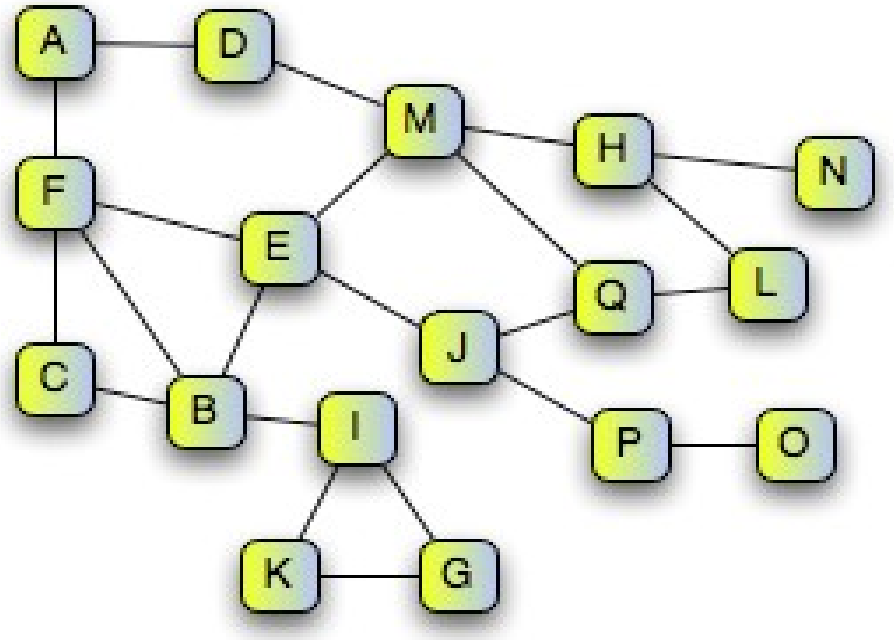
Eksempel, lagring av urettet graf




```
7
0  0   1      1
1  1   3      0  2  3
2  2   3      1  3  4
3  3   2      1  2
4  4   2      2  6
5  5   1      6
6  6   2      4  5
```

Fil: graf_7.txt

Eksempel, lagring av urettet graf



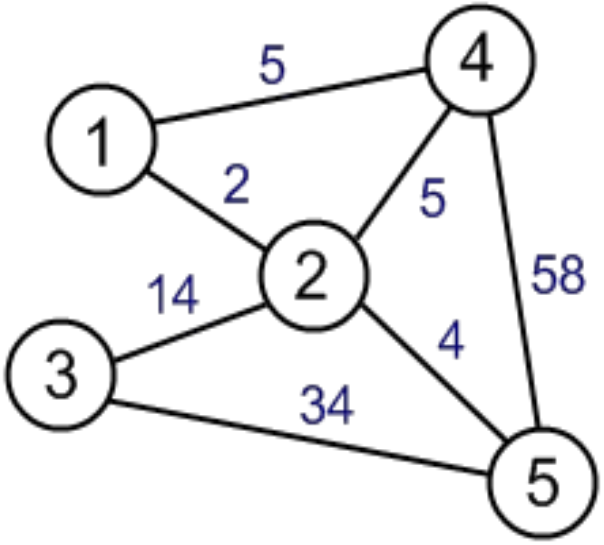

```
17
 0 A   2     3  5
 1 B   4     2  4  5  8
 2 C   2     1  5
 3 D   2     0 12
 4 E   4     1  5  9 12
 5 F   4     0  1  2  4
 6 G   2     8 10
 7 H   3    11 12 13
 8 I   3     1  6 10
 9 J   3     4 15 16
10 K   2     6  8
11 L   2     7 16
12 M   4     3  4  7 16
13 N   1     7
14 O   1    15
15 P   2     9 14
16 Q   3     9 11 12
```

Fil: graf_17.txt

Uvektet graf: Enkel implementasjon

- Noder med data som bare er enkel tekst-streng
- Representerer grafen som en Java-klasse med:
  - En array med dataene i hver node
  - En nabomatrise, der rettede og urettede grafer lagres på samme måte
- Leser inn hele grafen fra fil ved opprettelse av et nytt graf-objekt – ingen metoder for innsetting eller fjerning av data i grafen
- Java-kode: `enkelGraf.java`

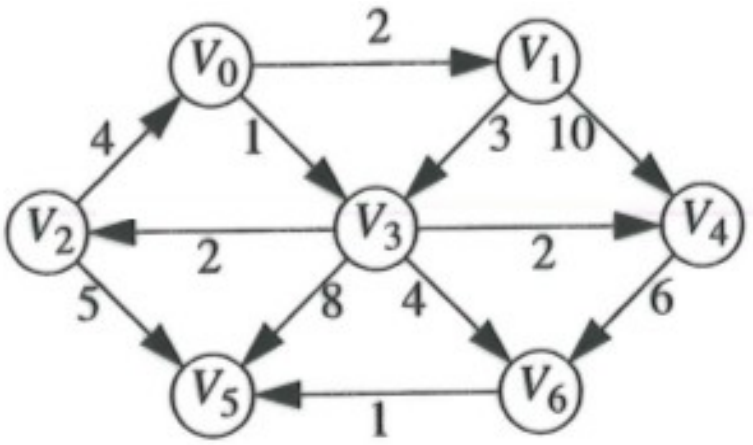
Eksempel, lagring av urettet, vektet graf




```
5
0 1   2      1   2      3   5
1 2   4      0   2      2 14      3   5      4   4
2 3   2      1 14      4 34
3 4   3      0   5      1   5      4 58
4 5   3      1   4      2 34      3 58
```

Fil: vgraf_5.txt

Eksempel, lagring av rettet, vektet graf




```
7
0 V0   2     1   2     3   1
1 V1   2     3   3     4  10
2 V2   2     0   4     5   5
3 V3   4     2   2     4   2     5   8     6   4
4 V4   1     6   6
5 V5   0
6 V6   1     5   1
```

Fil: vgraf_7.txt

Vektet graf: Enkel implementasjon

- Noder med data som bare er enkel tekst-streng
- Representerer grafen som en Java-klasse med:
  - En array med dataene i hver node
  - En vekt-/kantlengde-matrise, der rettede og urettede kanter lagres på samme måte
  - “Ingen kant” representeres med verdien “uendelig”
- Leser inn hele grafen fra fil ved opprettelse av et nytt graf-objekt – ingen metoder for innsetting eller fjerning av data i grafen
- Java-kode: `enkelVektetGraf.java`