

Forelesning 1

Problemer og algoritmer

Pensum

- Kap. 1. The role of algorithms in computing
- Kap. 2. Getting started
- Kap. 3. Growth of functions:
Innledning og 3.1

THE SCIENCE BEHIND A CRAZY 6-WAY KIDNEY EXCHANGE

THE SCIENCE BEHIND A CRAZY 6-WAY KIDNEY EXCHANGE

KATIE M. PALMER SCIENCE 03.09.15 2:43 PM

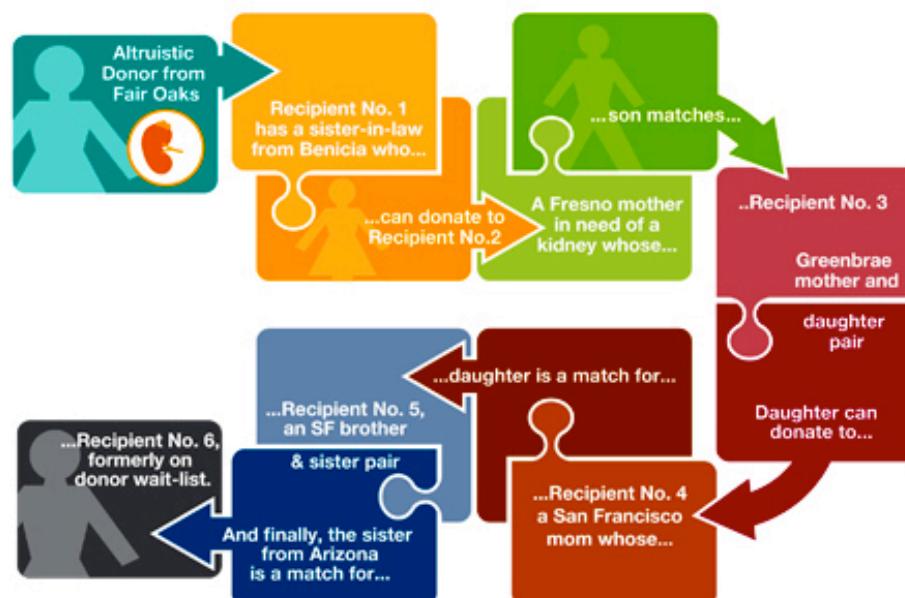
THE SCIENCE BEHIND A CRAZY 6-WAY KIDNEY EXCHANGE

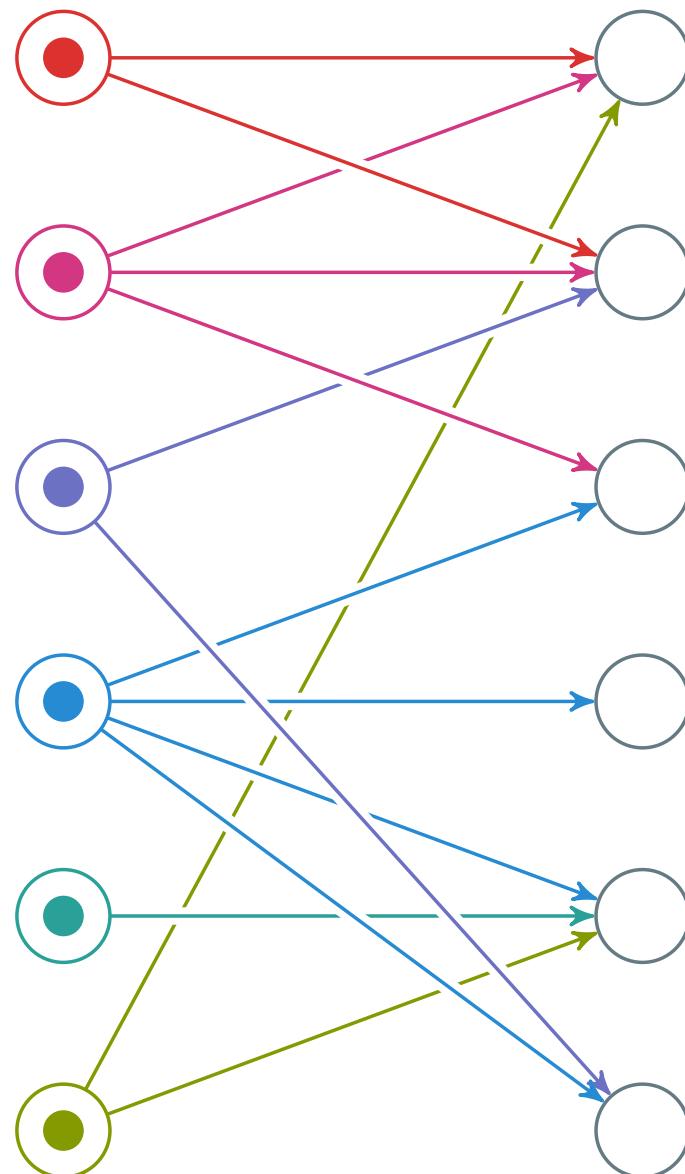
IN A SURGERY worthy of the most convoluted *Grey's Anatomy* plot, surgeons at California Pacific Medical Center in San Francisco completed a two-day, six-way kidney transplant late last week. Five surgeons and dozens of anesthesiologists and nurses daisy-chained 12 patients together in the West Coast's largest paired transplant ever.

Patients who need a new kidney are often subjugated to years-long waiting lists, biding their time until an organ becomes available, typically from a recently deceased donor. But living transplants, from donors who give a kidney to a relative or friend, are far more likely to succeed—and can last twice as long before another transplant is needed. The problem is, that

KATIE M. PALMER SCIENCE 03.09.15 2:43 PM

THE SCIENCE BEHIND A CRAZY 6-WAY KIDNEY EXCHANGE





Bipartitt graf: Graf der nodene kan deles i to grupper, og der det ikke finnes kanter innad i noen av gruppene. I dette tilfelle er gruppene donorer og mottakere.

Bipartitt matching: Finn et subsett av kantene der ingen kanter er koblet til de samme nodene som noen andre. Intuitivt: Match hver donor med nøyaktig én mottaker, og omvendt.

Sikk-sakk

Basert på Ford-Fulkerson

Bare en illustrasjon –
dere skal lære Ford-
Fulkerson-algoritmen
senere.

I hver iterasjon

- Spør hver donor etter tur om de er i stand til å donere.

Kan du ta imot?

- Står vi uten donor? Svar i så fall ja.
- Ellers, hør om vår eksisterende donor kan donere til noen andre.

Kan du donere?

- Spør alle kompatible donorer vi ikke alt har spurt i denne iterasjonen.
- Hvis noen kan ta imot, svar ja. Ellers, svar nei.

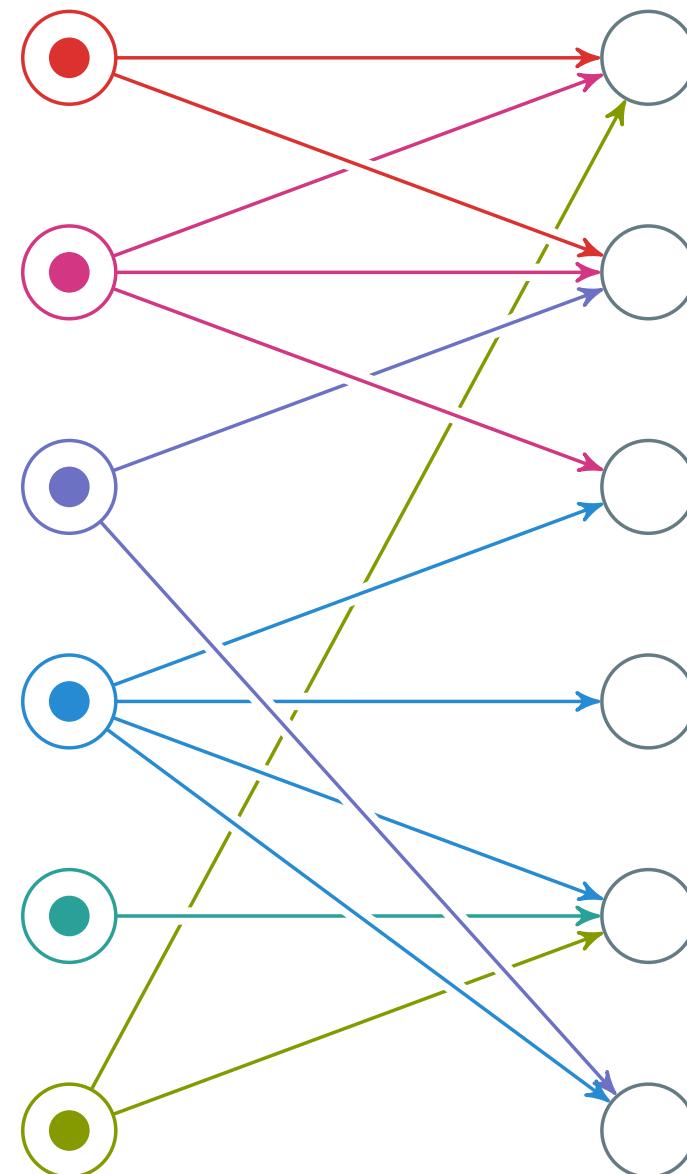
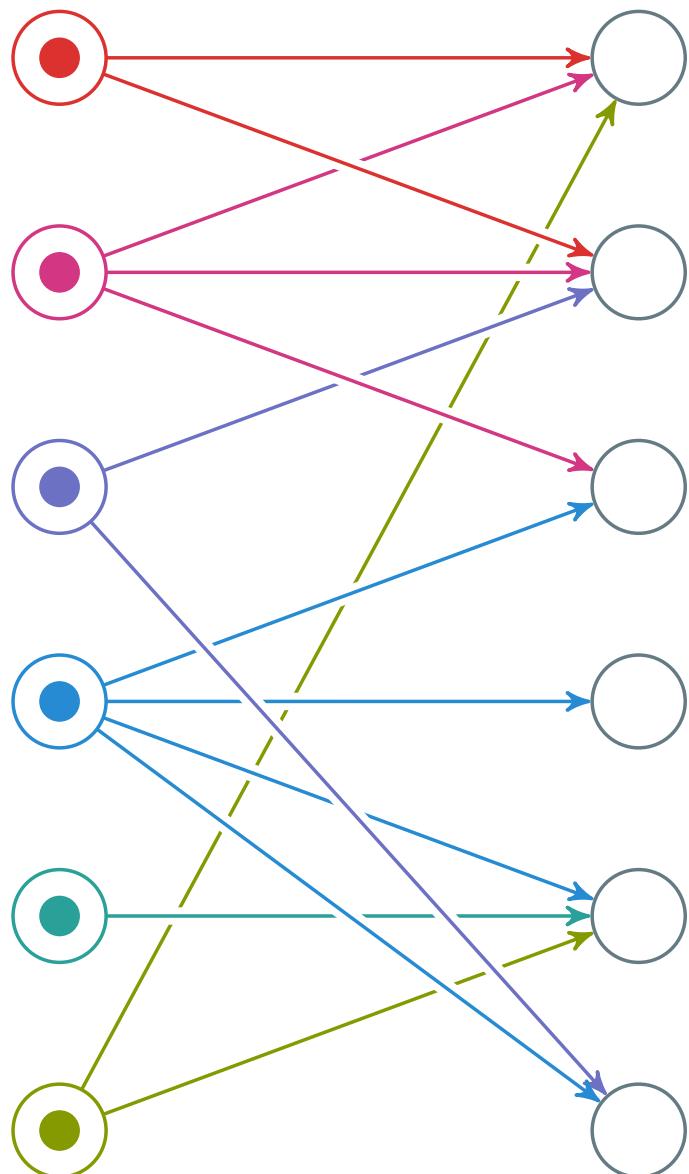
Flipp-flopp

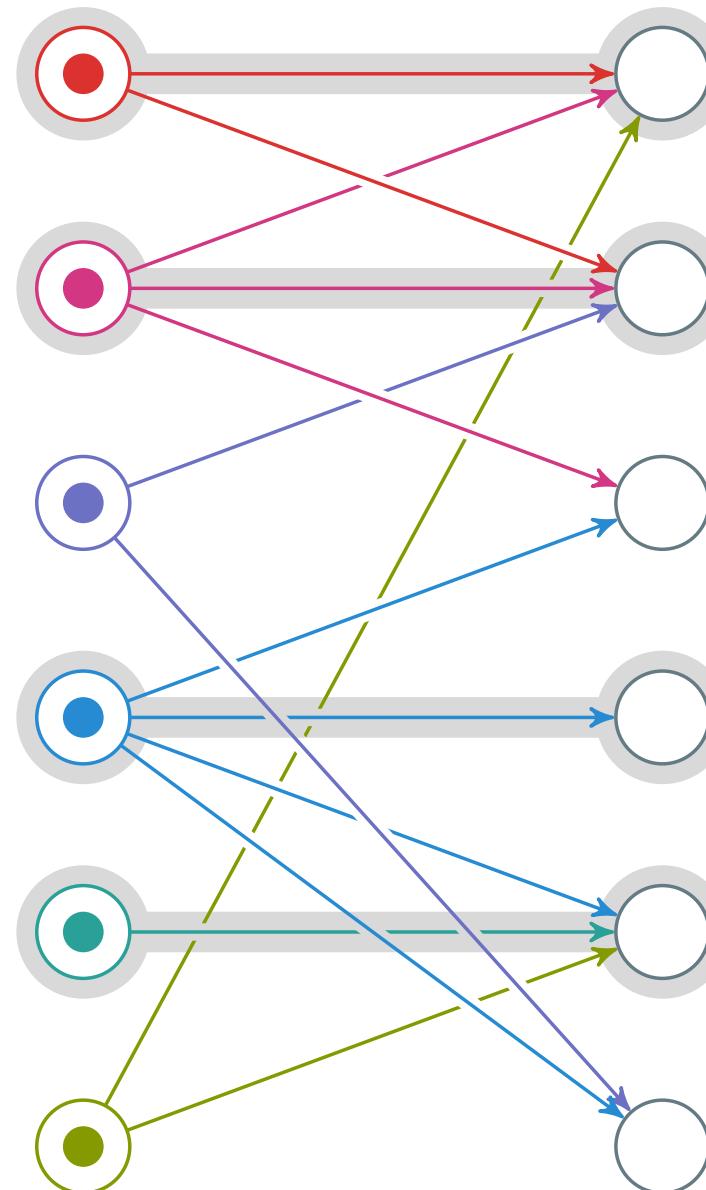
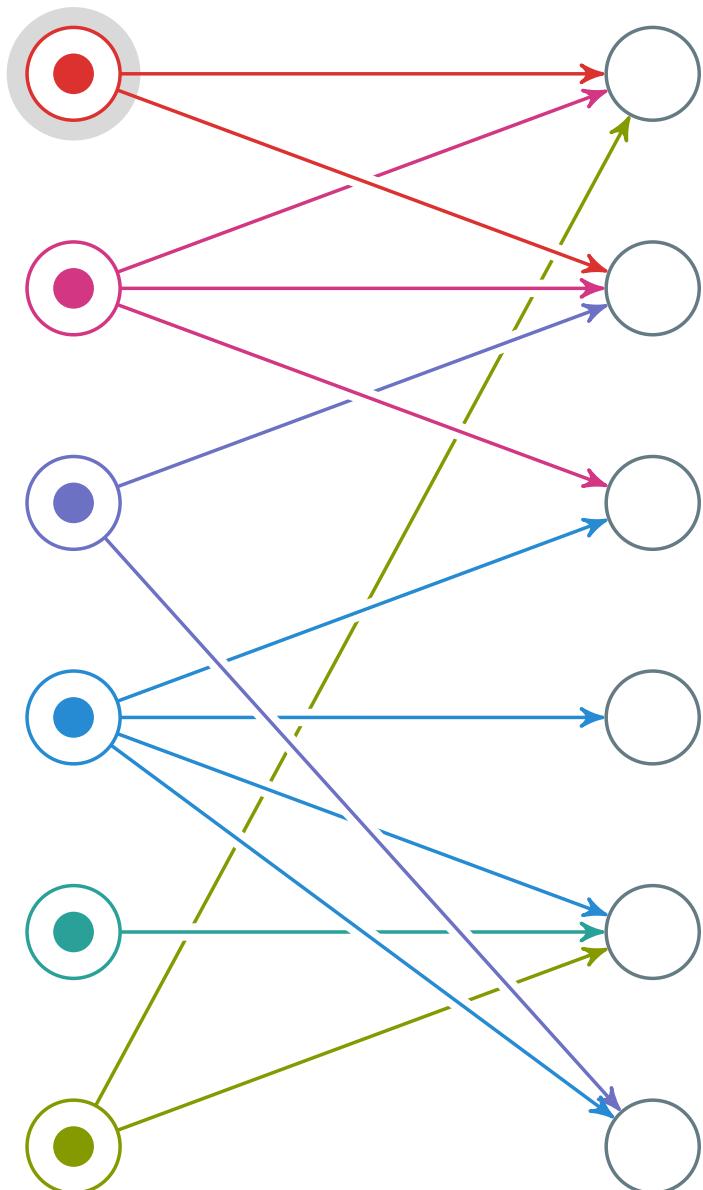
Basert på Heaps algoritme

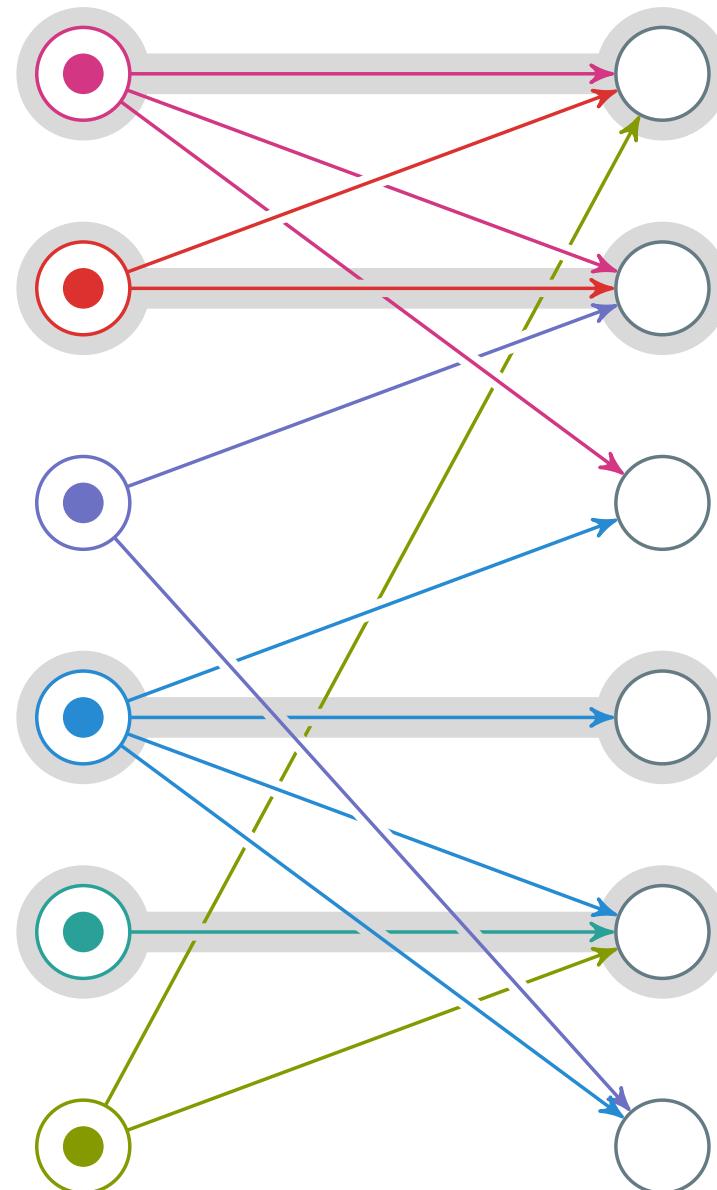
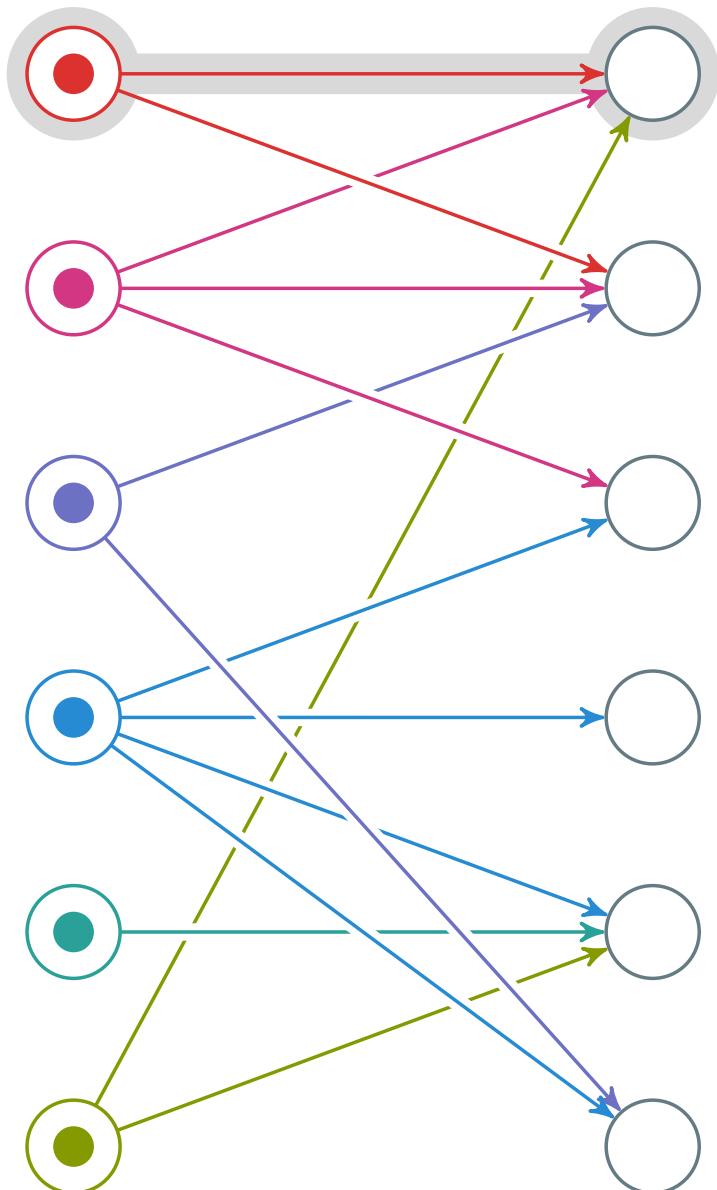
Også bare en illustrasjon
- ikke en
pensumalgoritme.

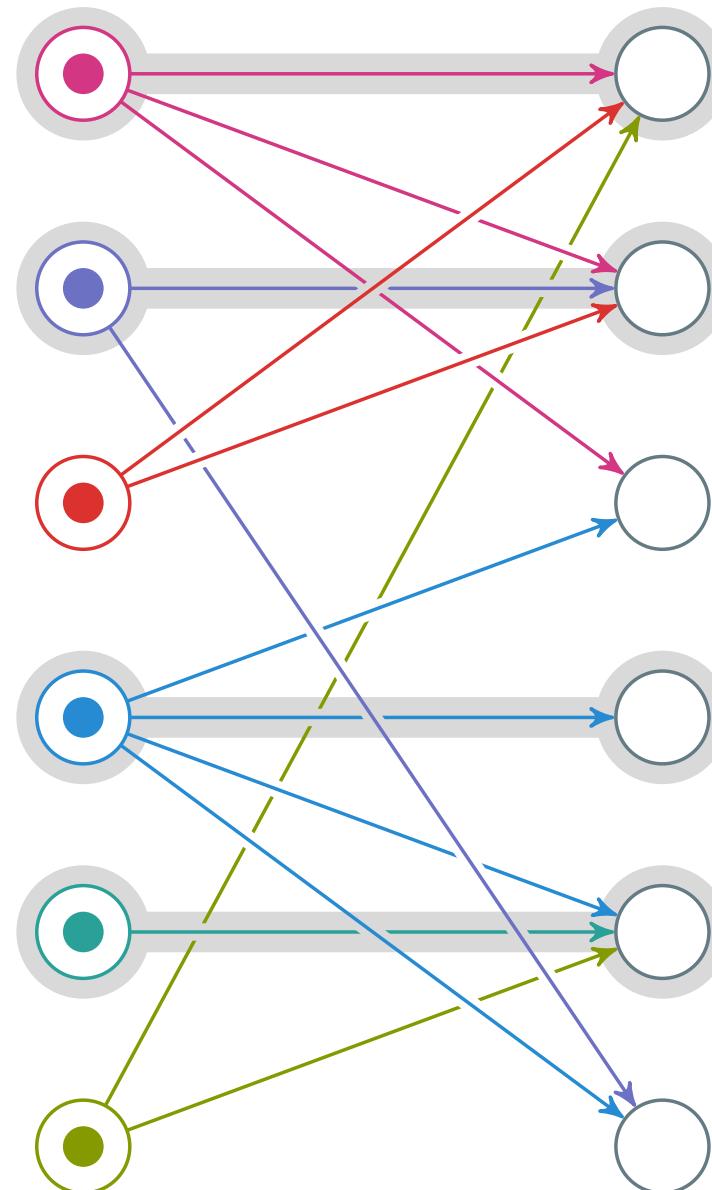
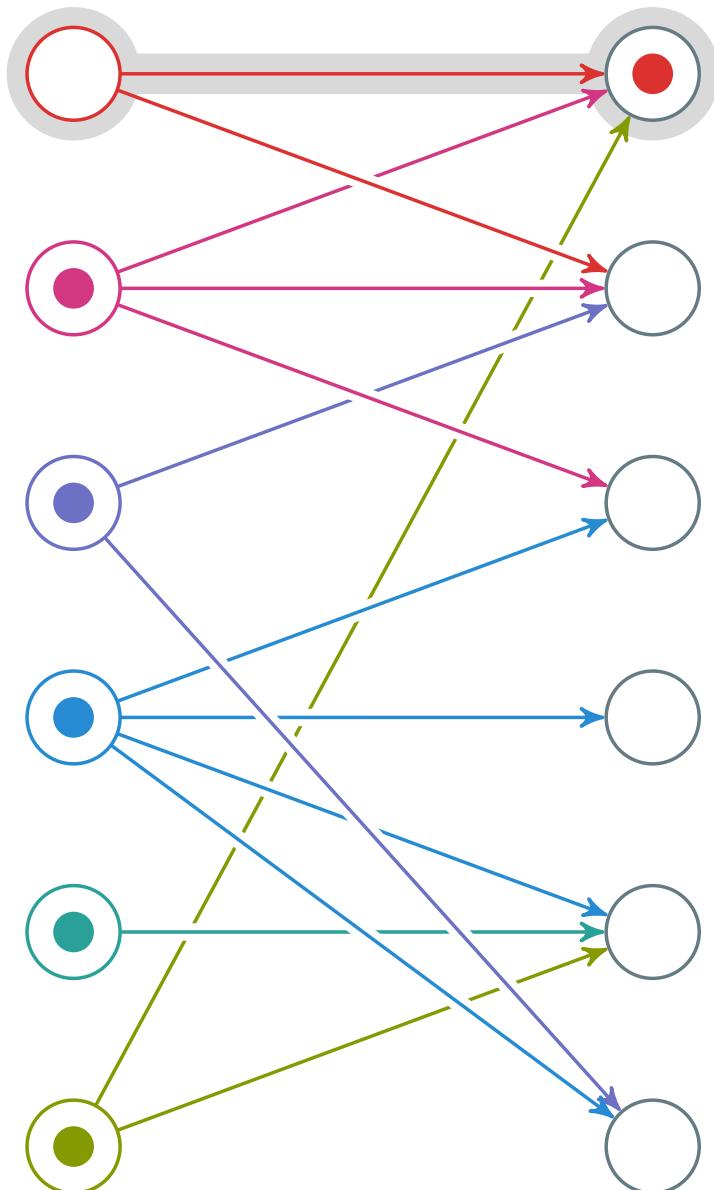
For parameter N

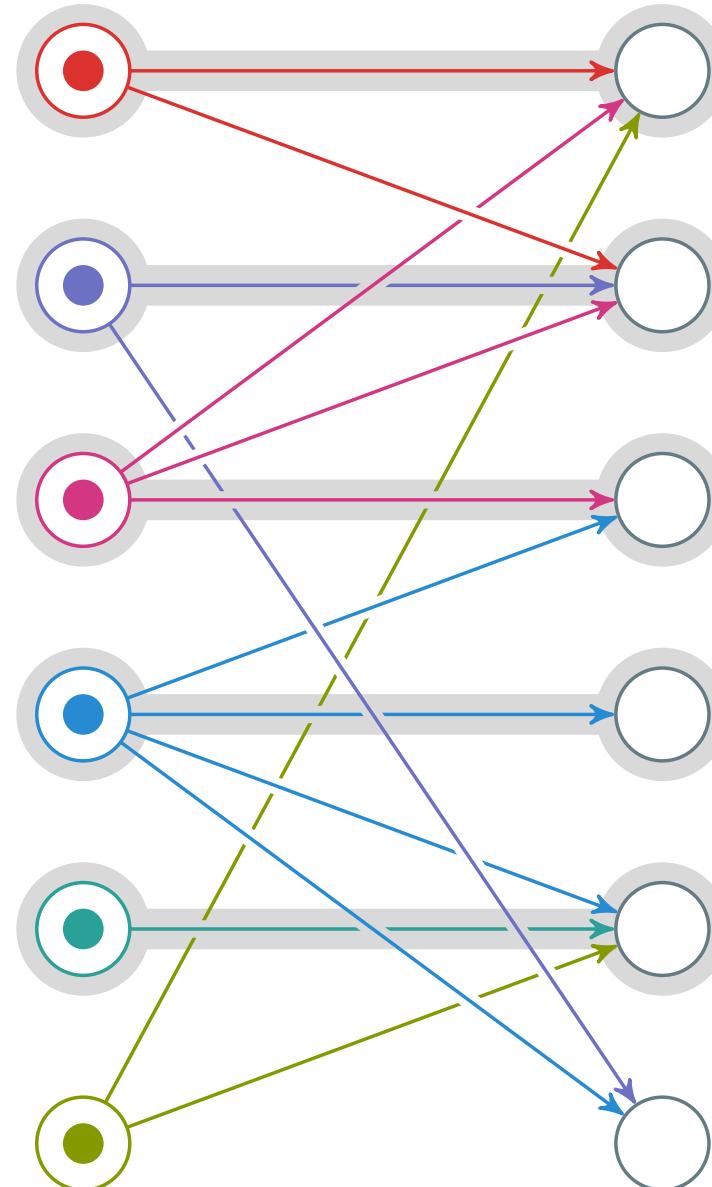
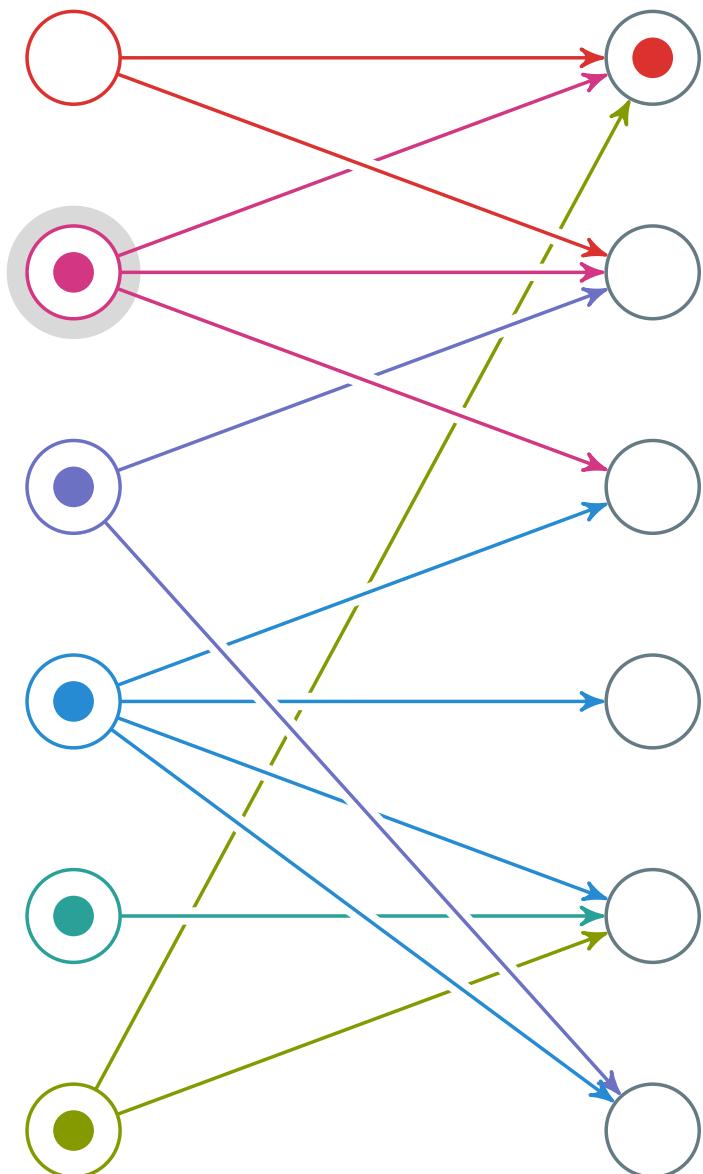
- Til å begynne med, la M være 1
- Hvis N er større enn 2, løs for $N-1$.
- Hvis M er lik N , avslutt.
- Hvis N er et oddetall, bytt plass på donor 1 og N ; ellers bytt M og N .
- Øk M med 1 og begynn en ny runde.

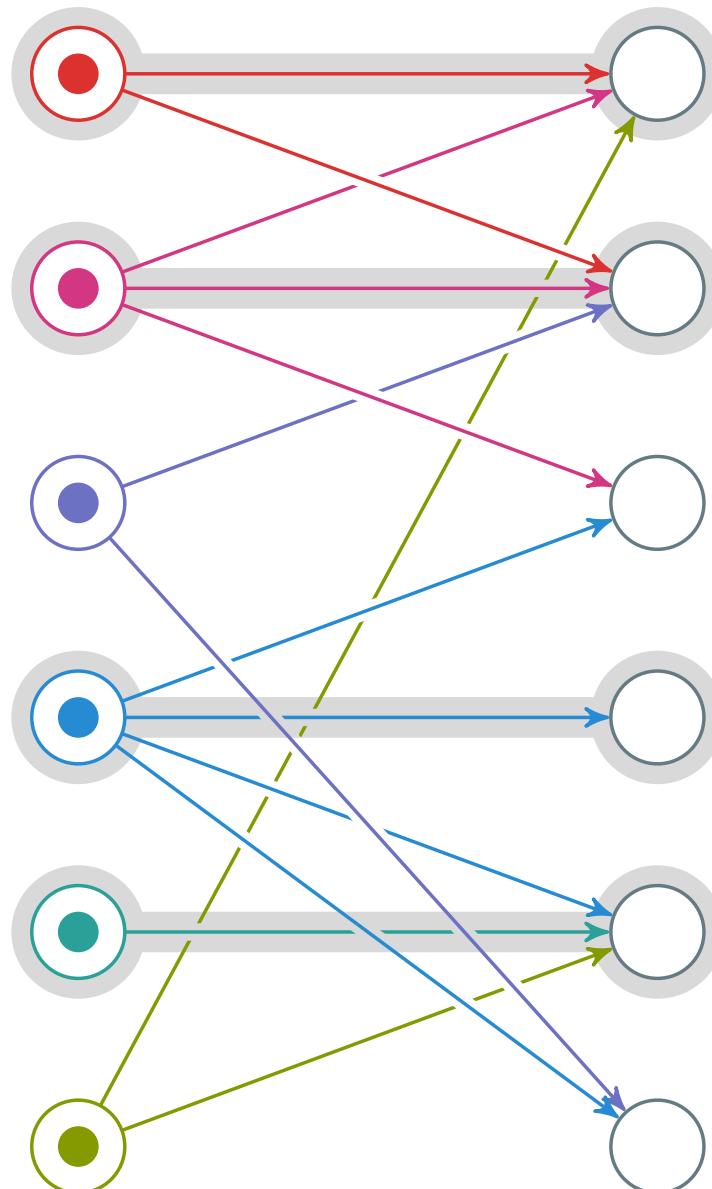
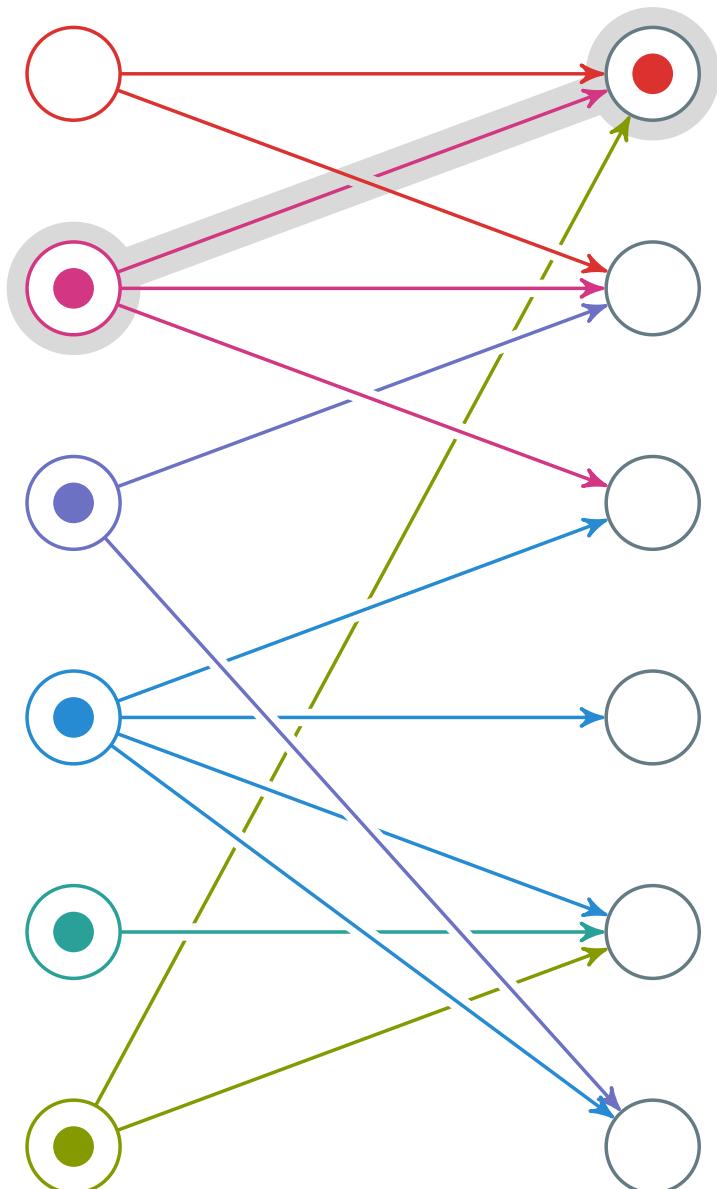


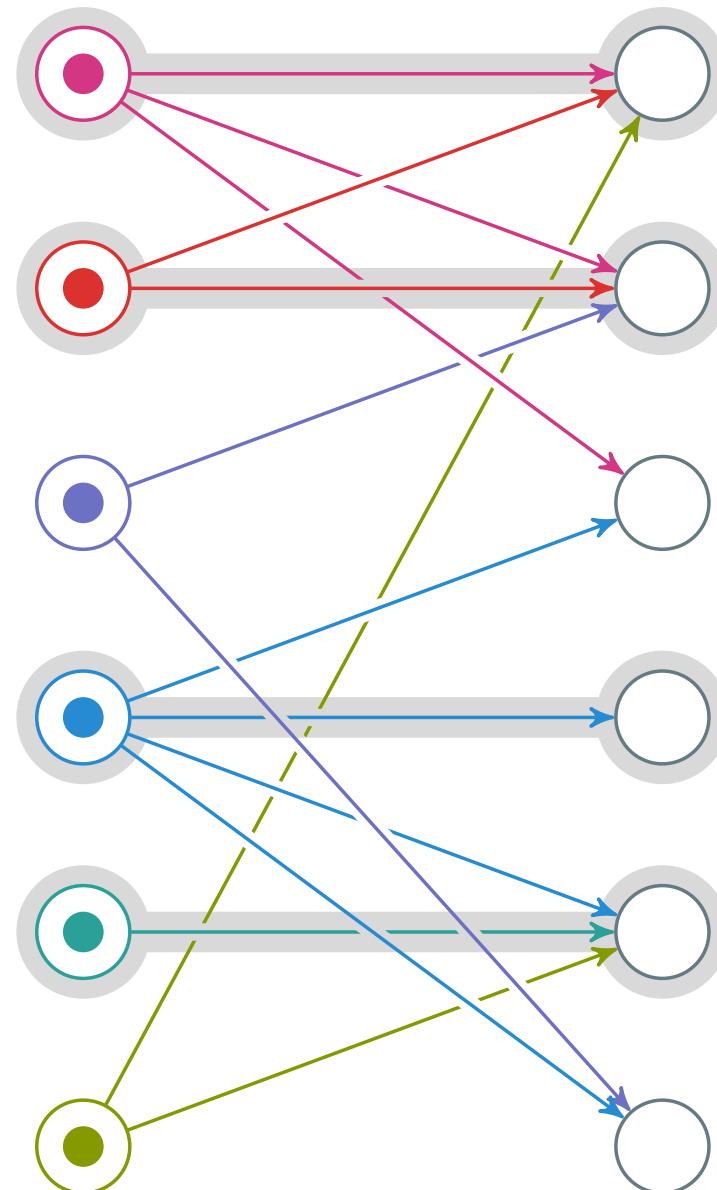
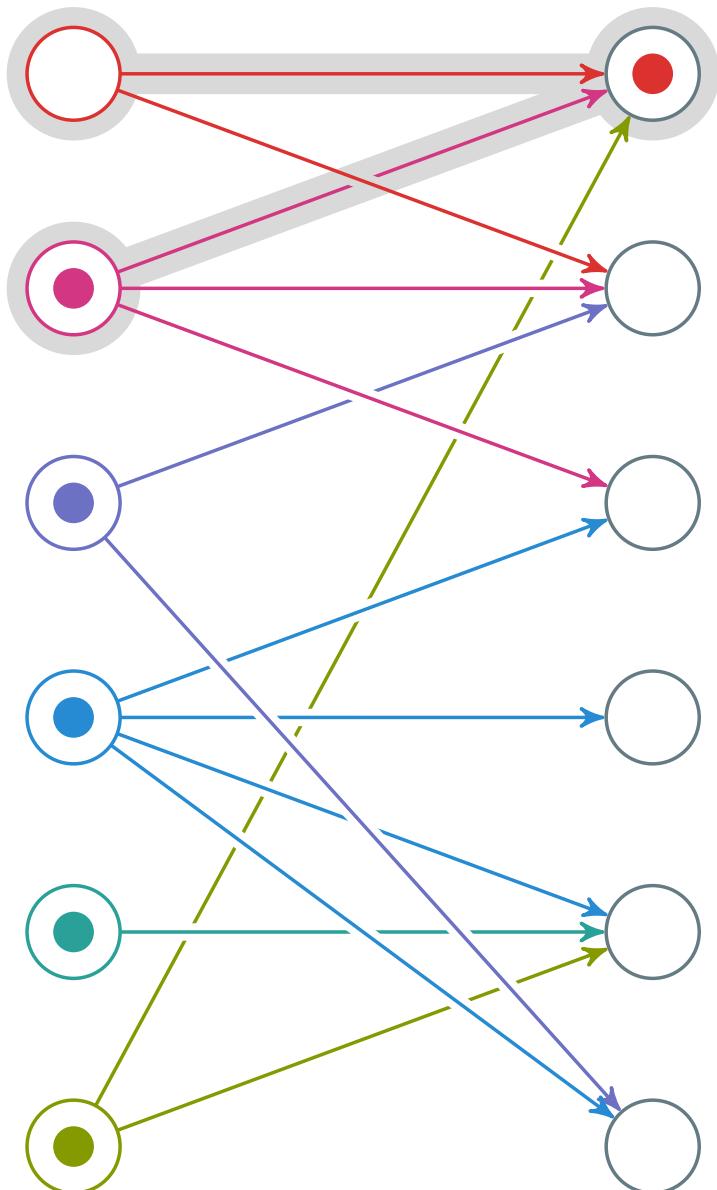


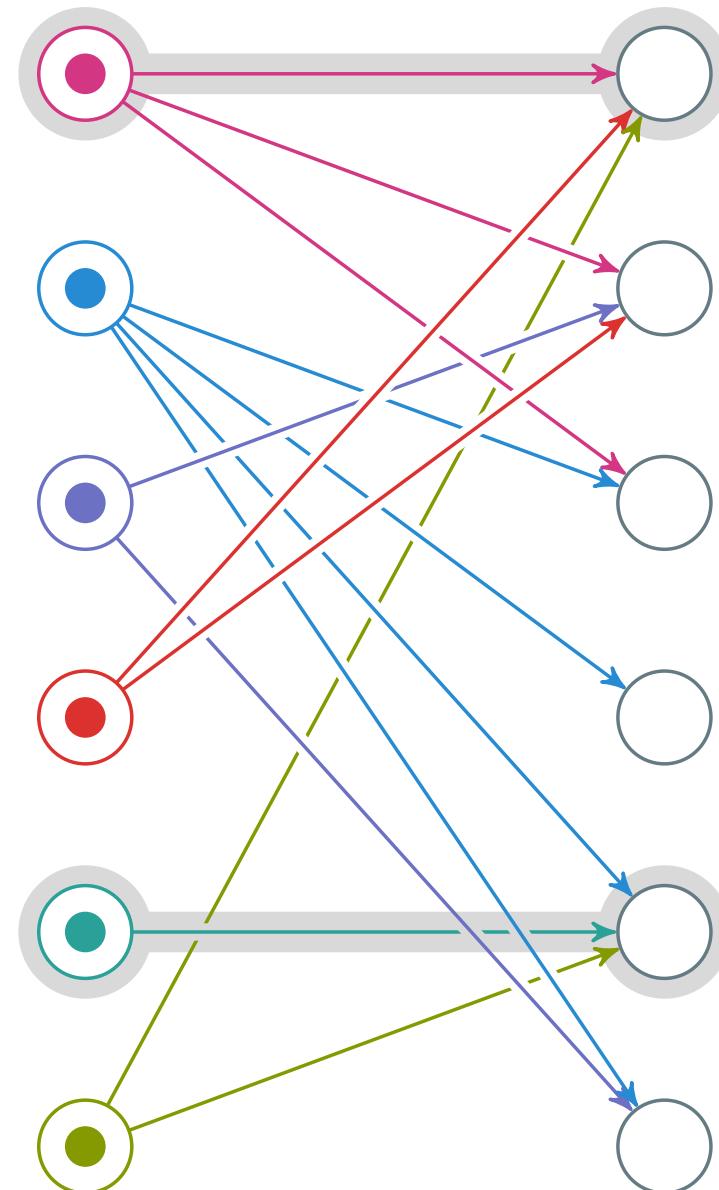
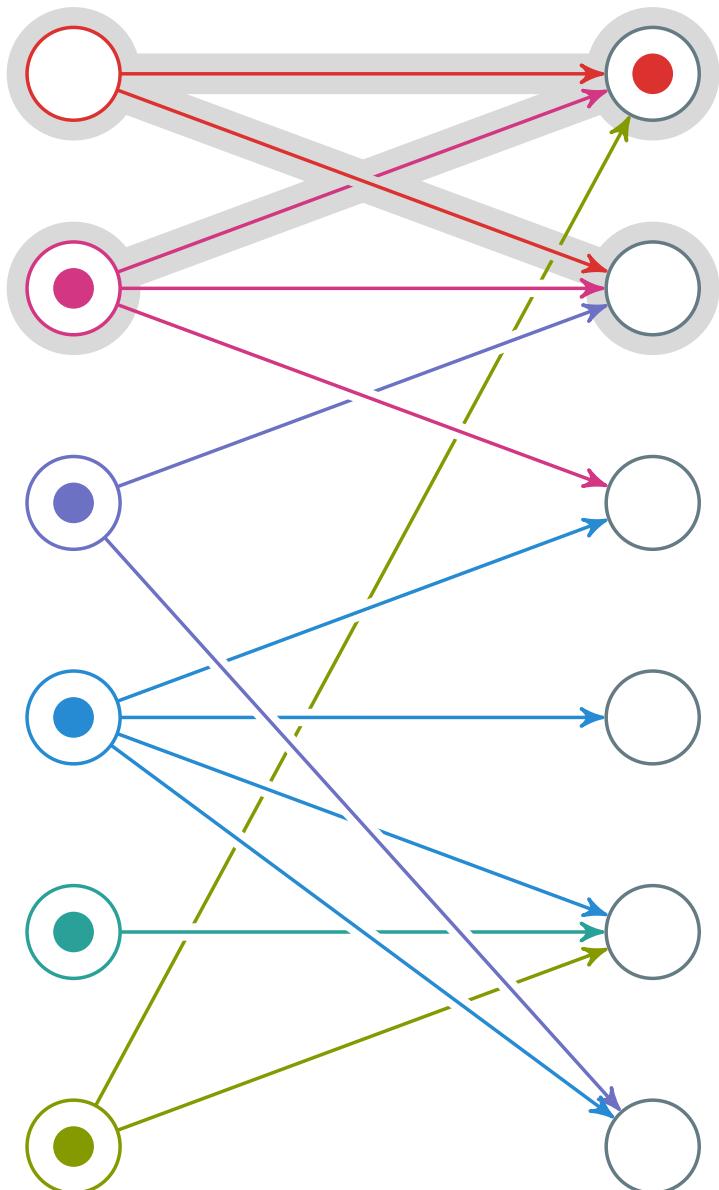


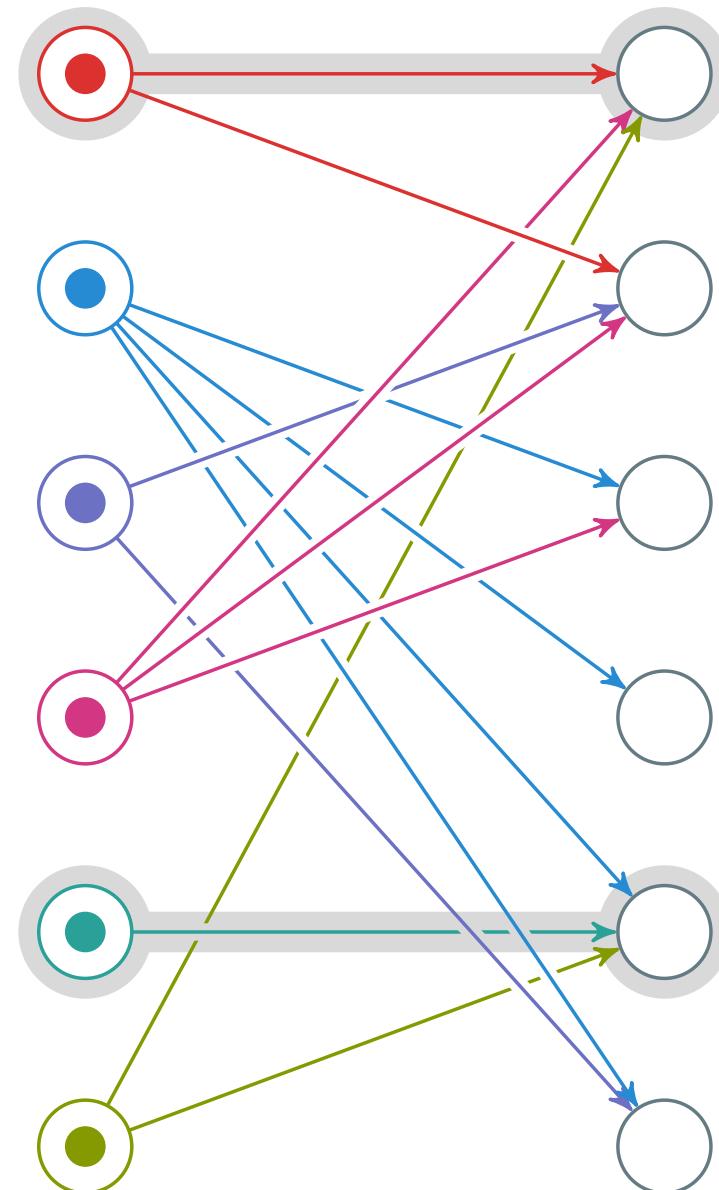
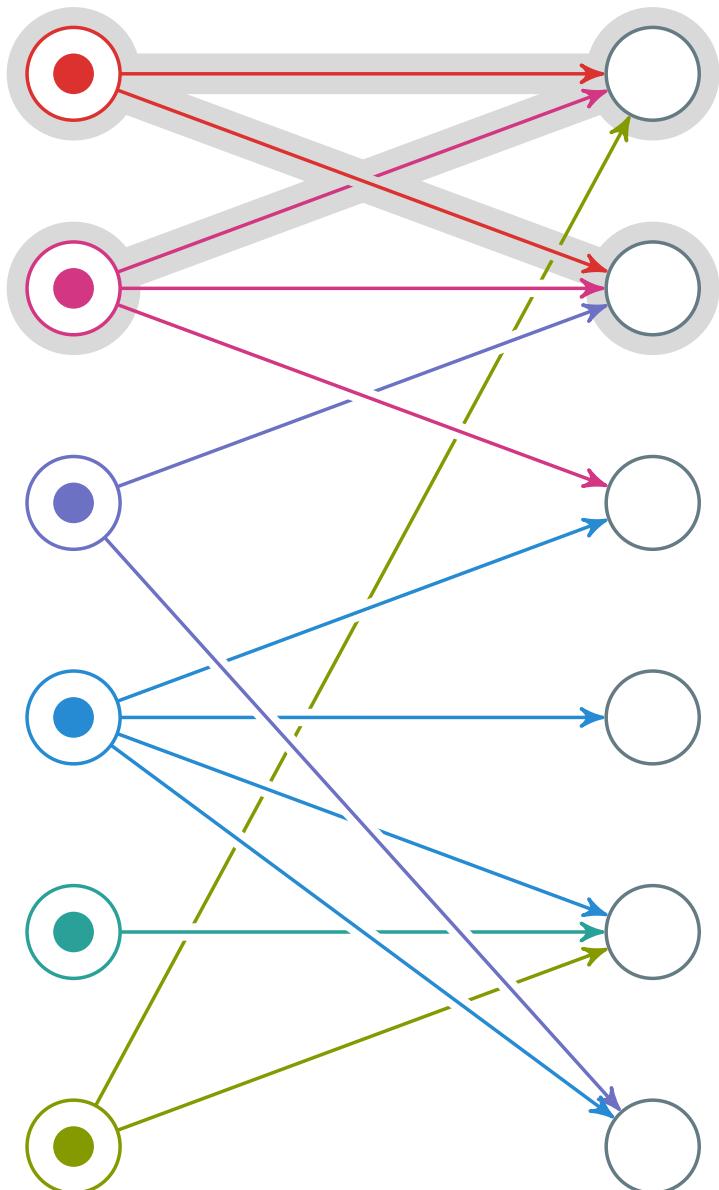


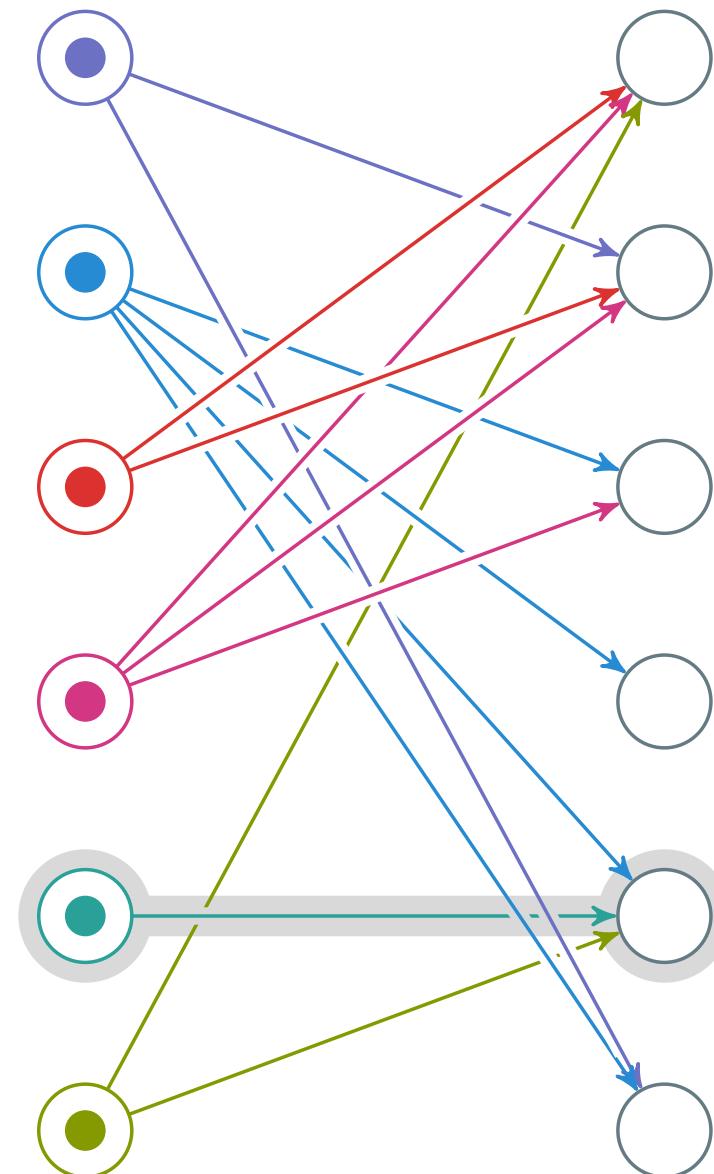
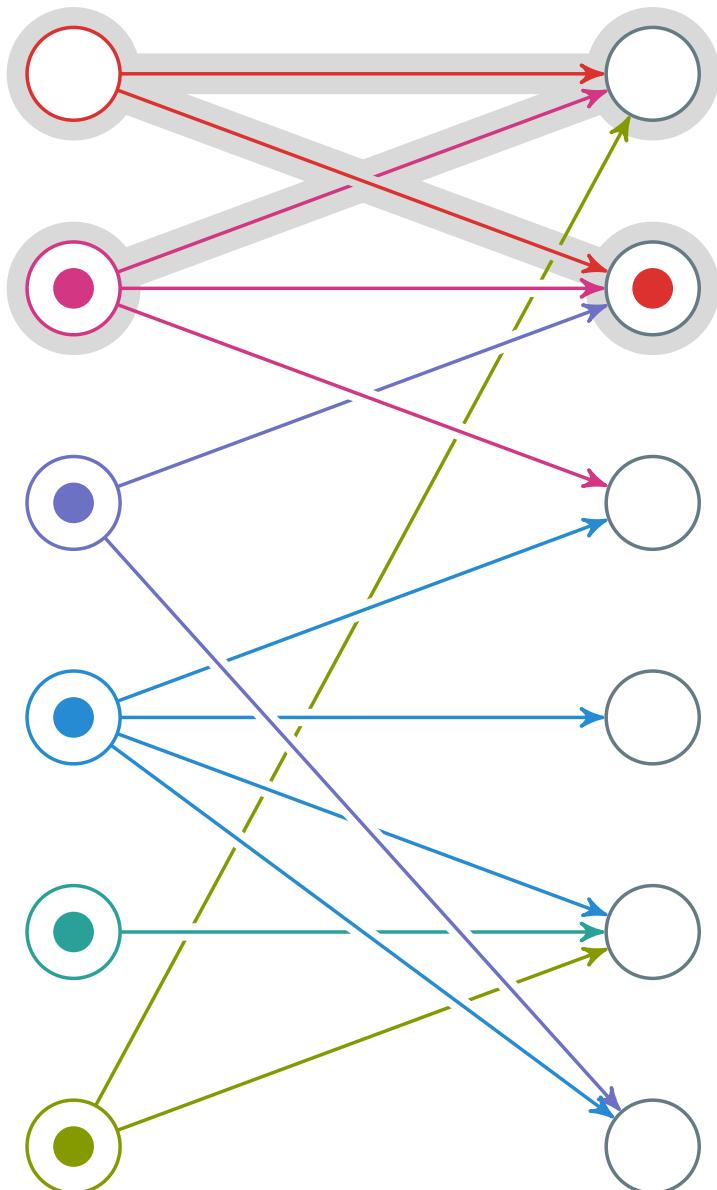


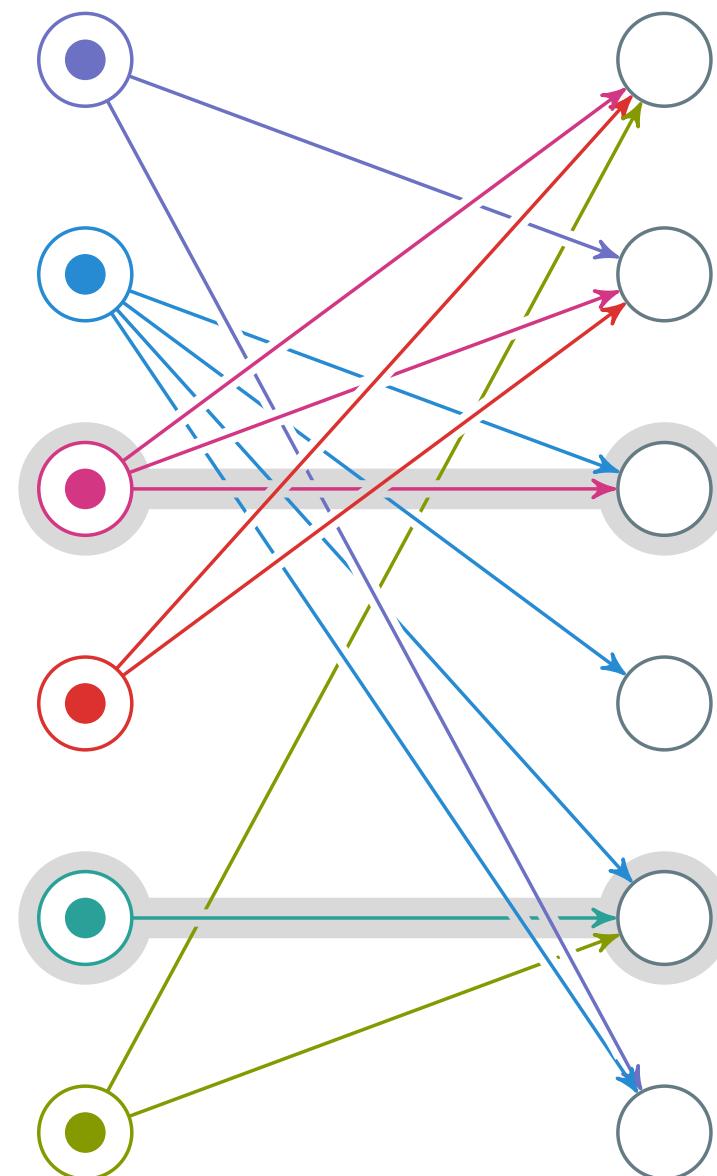
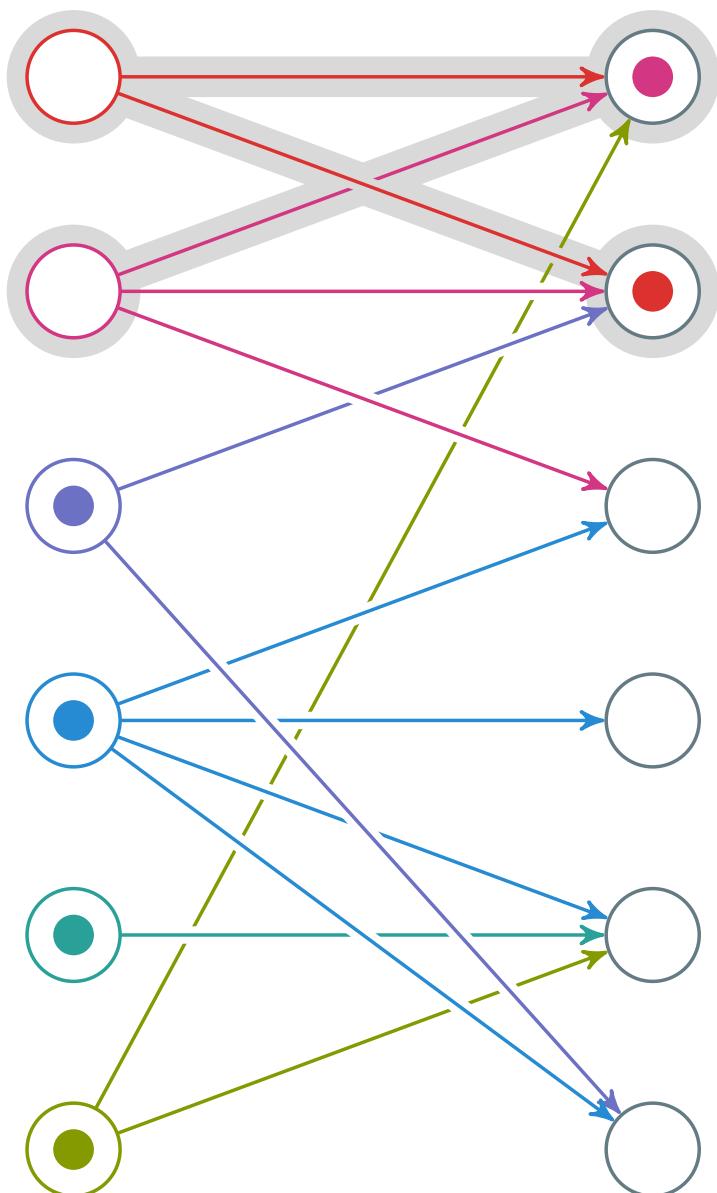


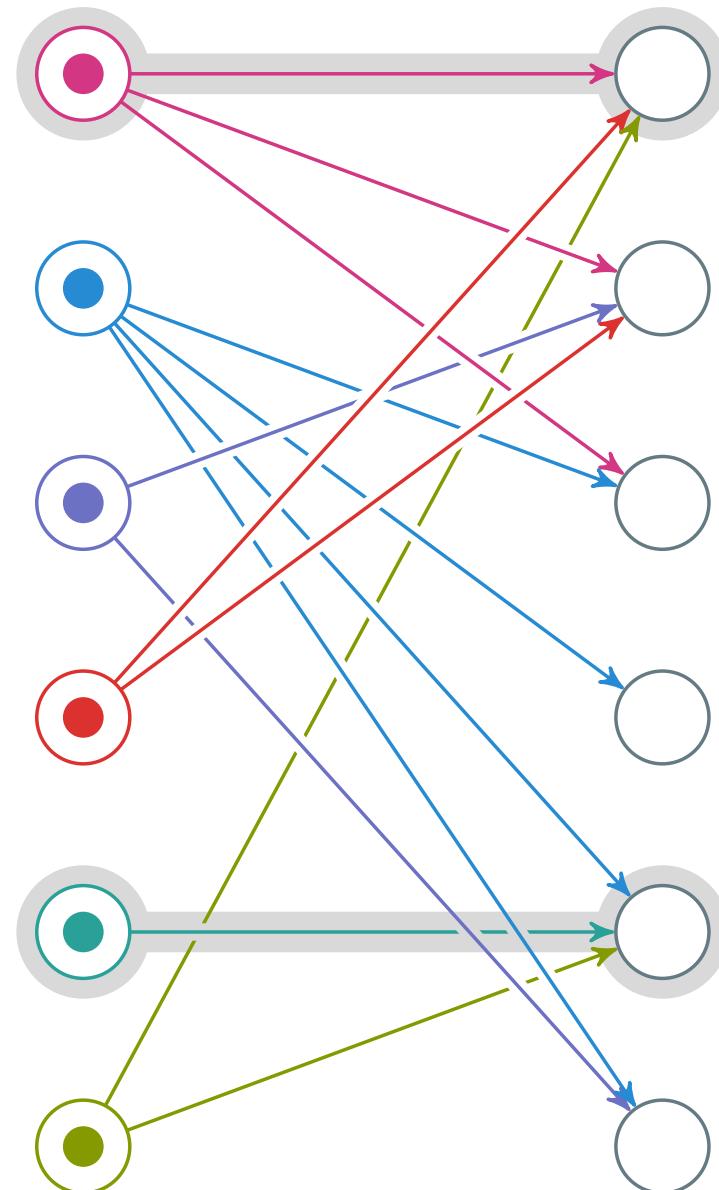
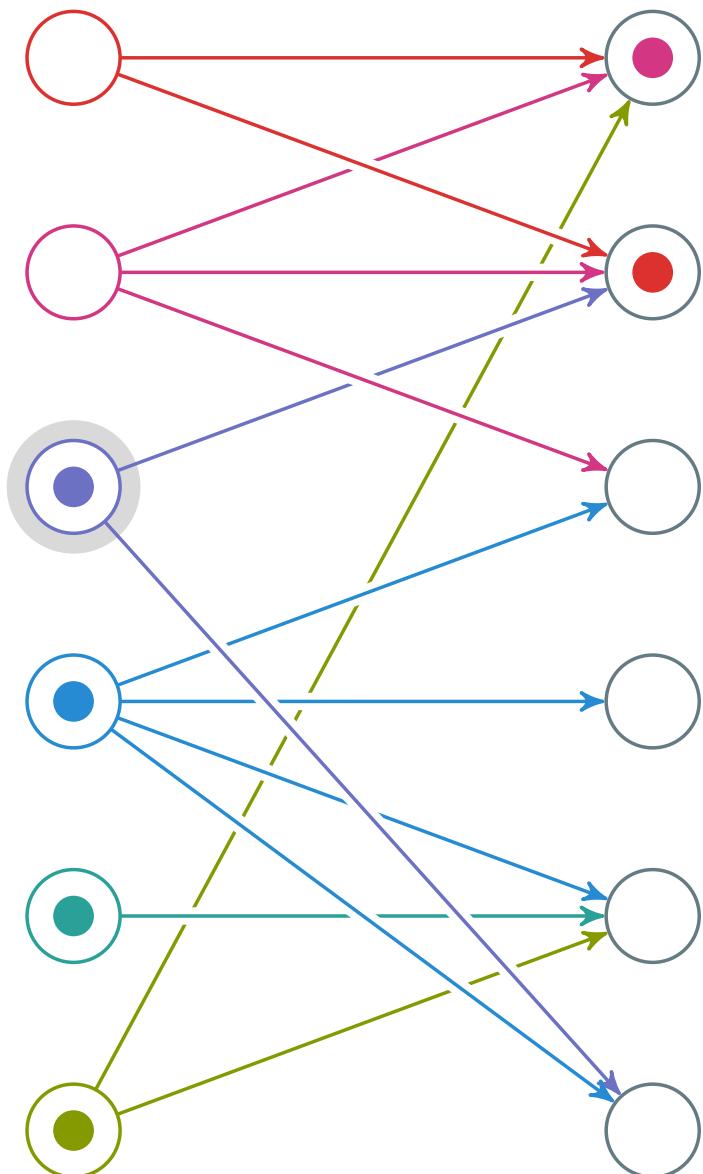


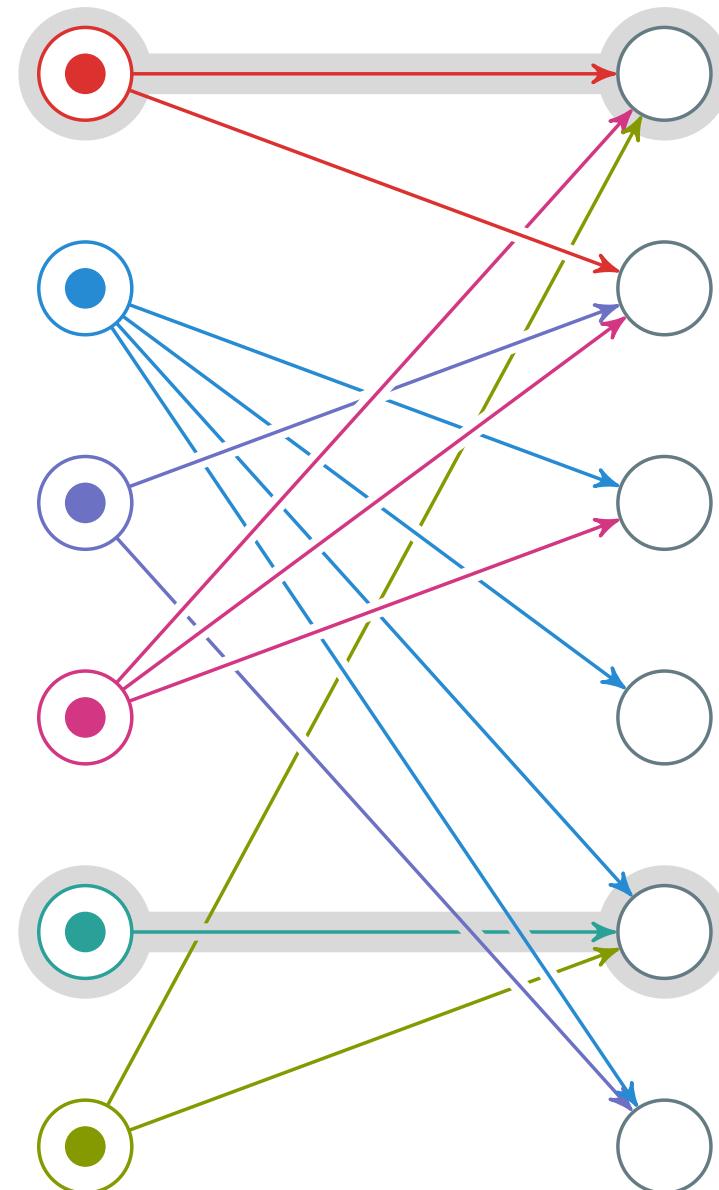
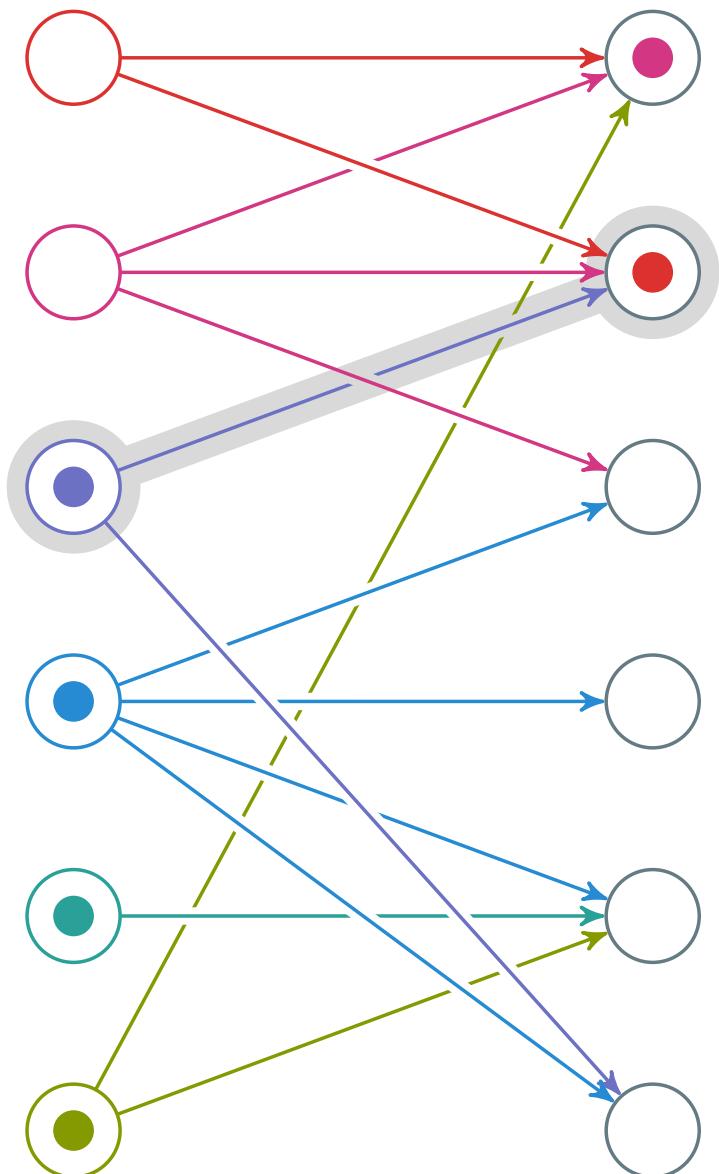


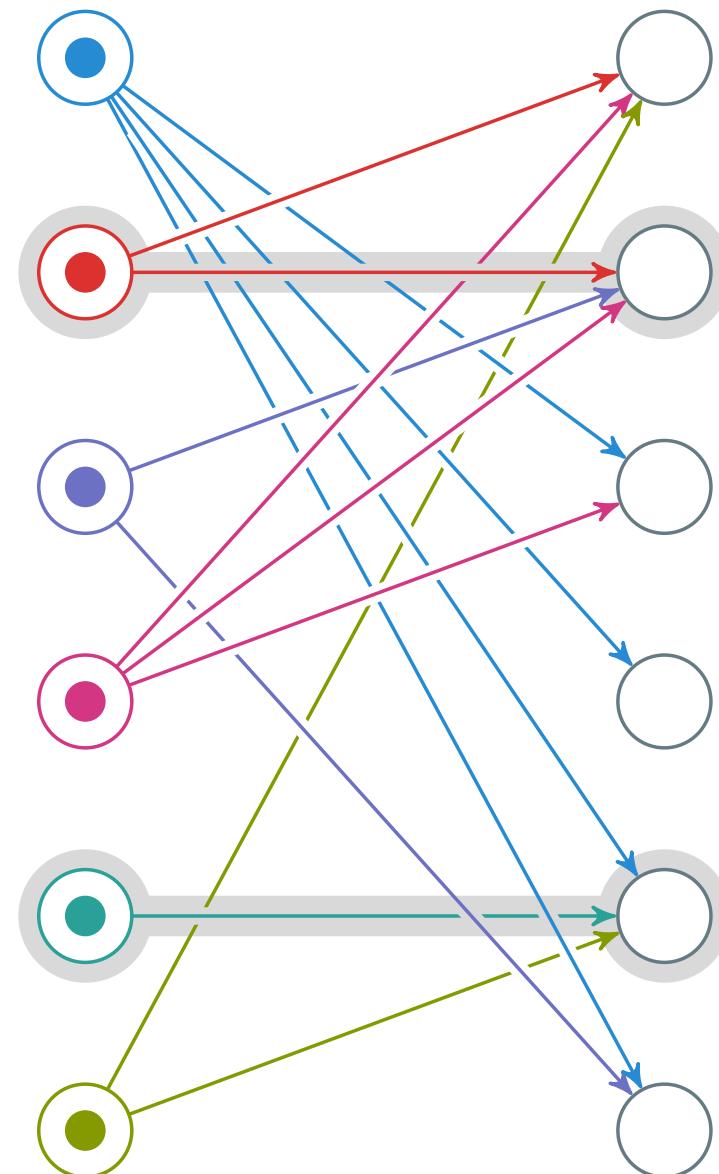
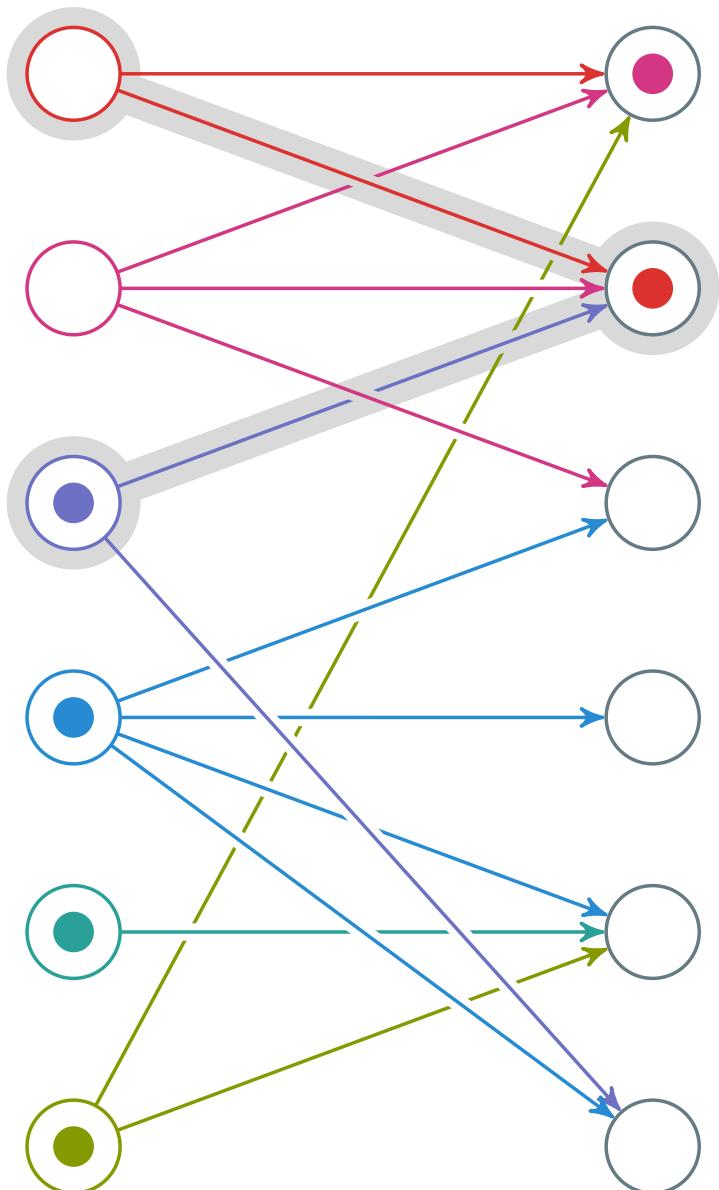


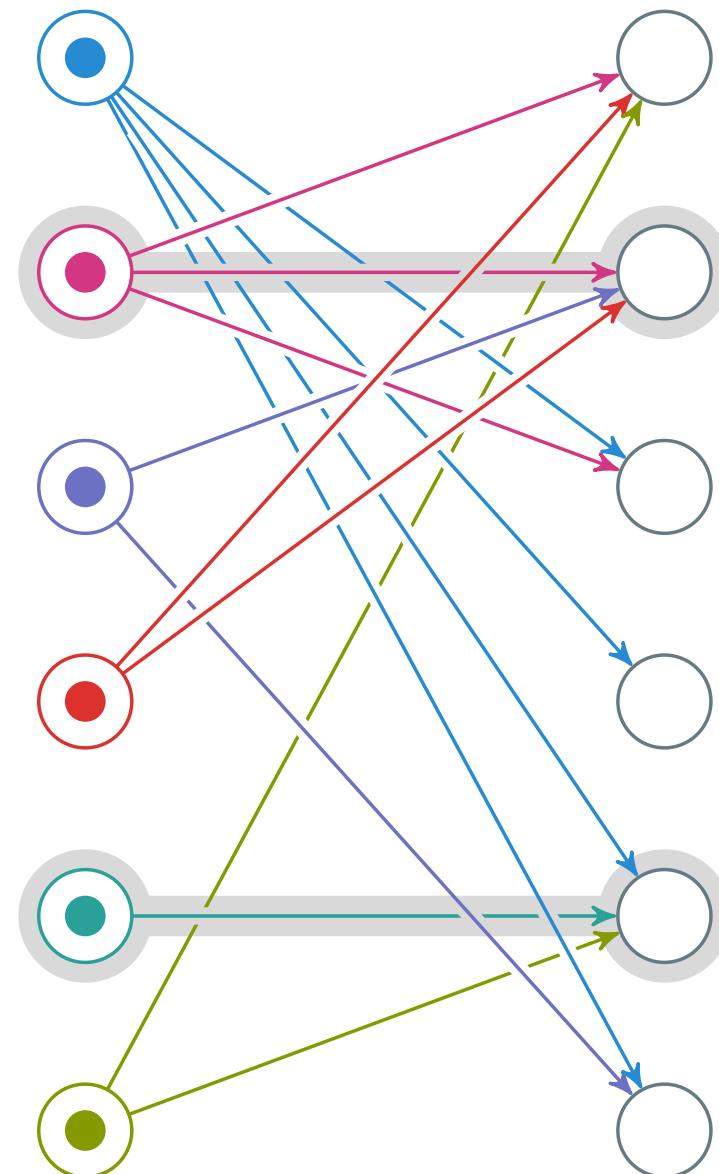
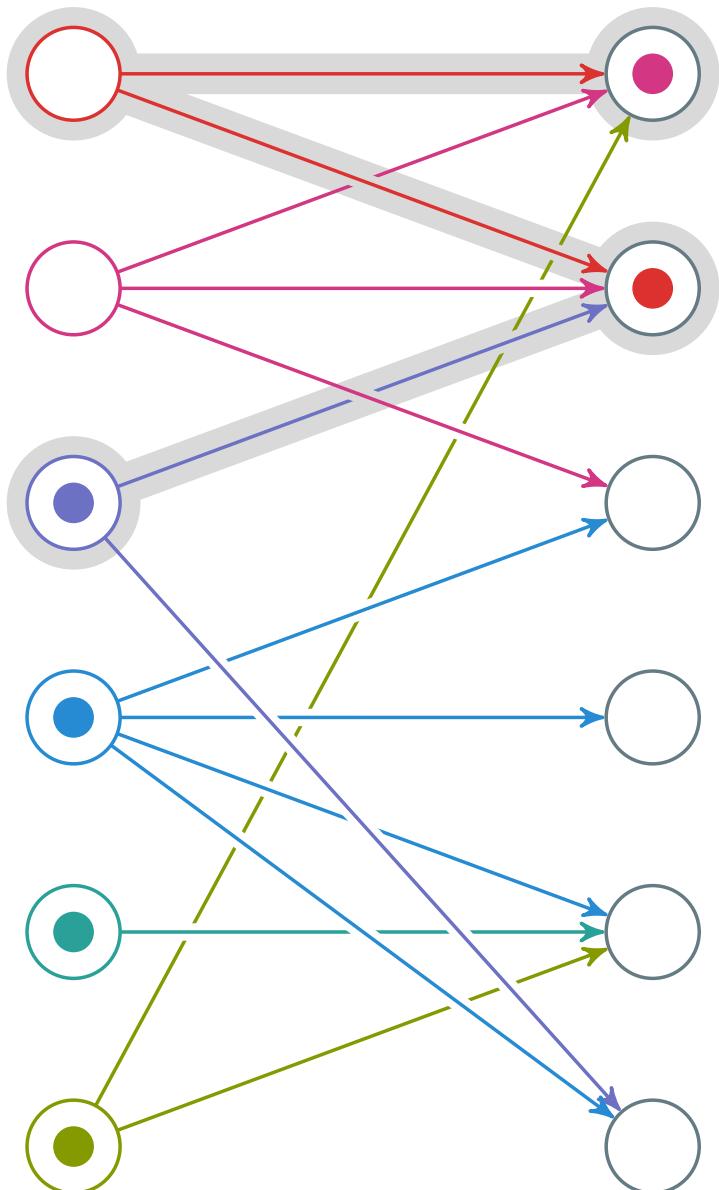


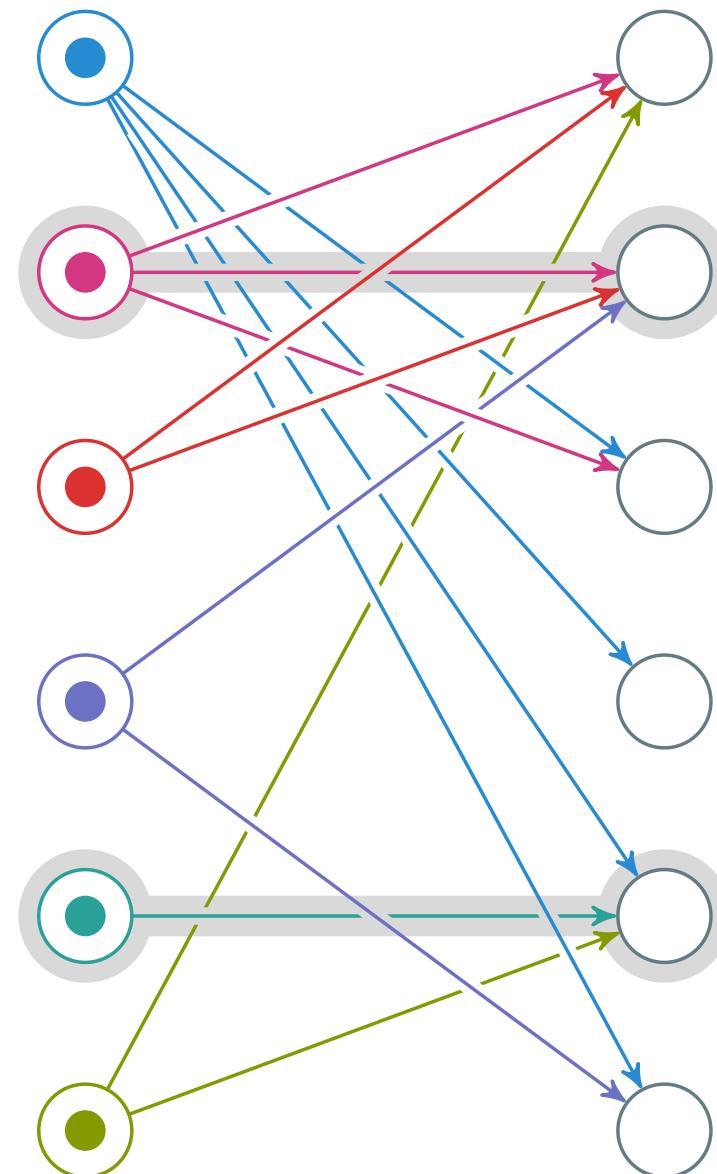
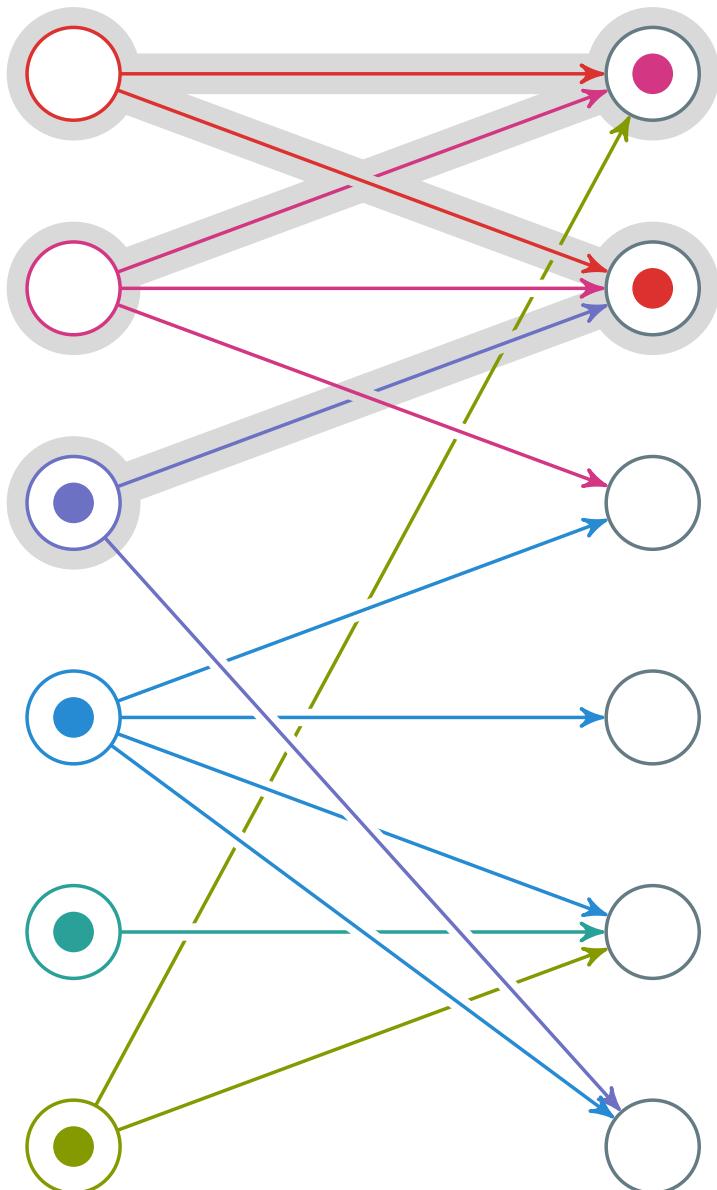


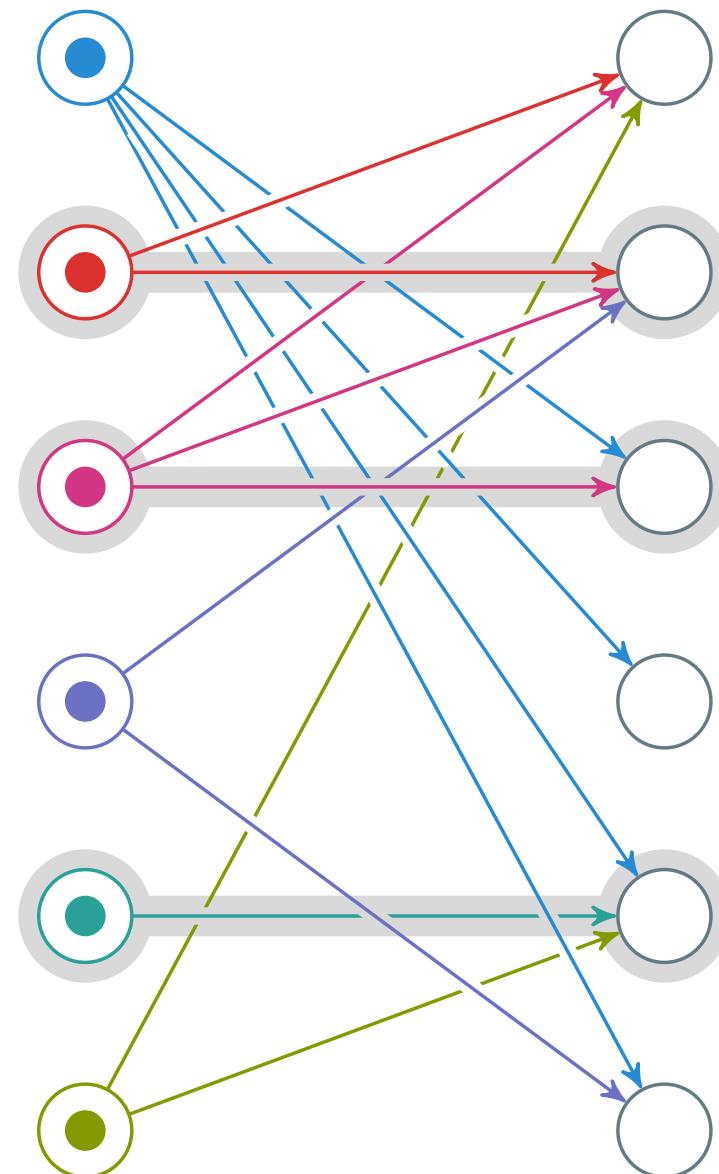
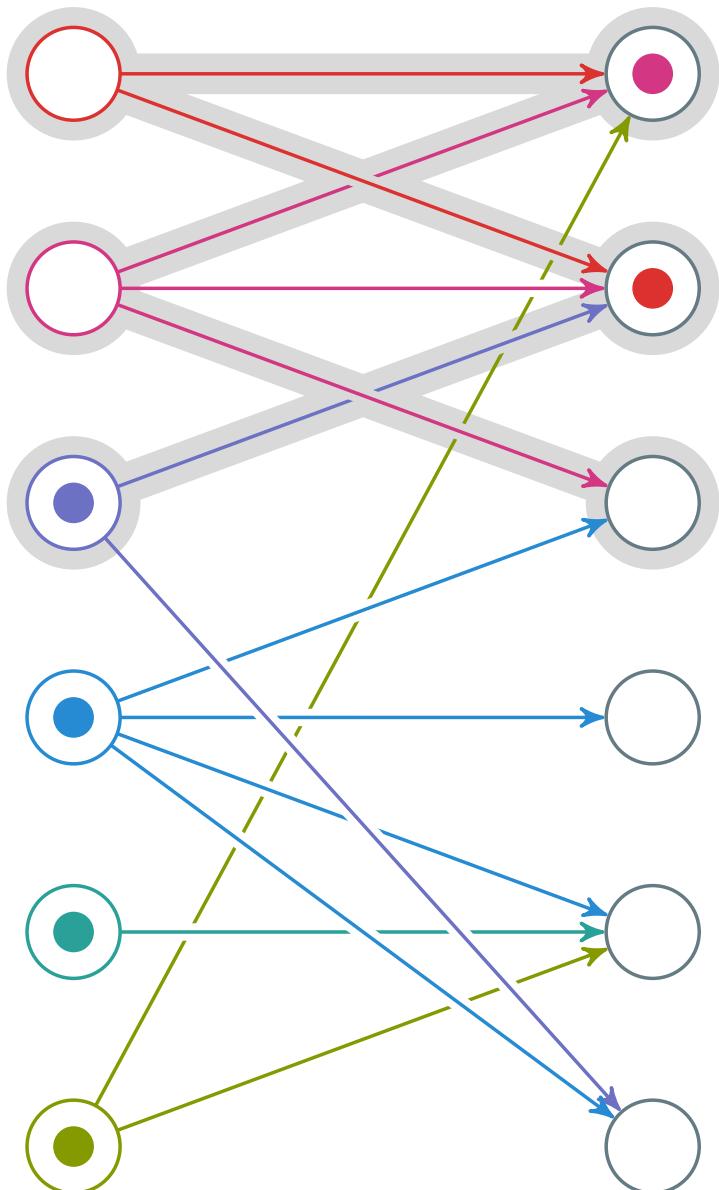


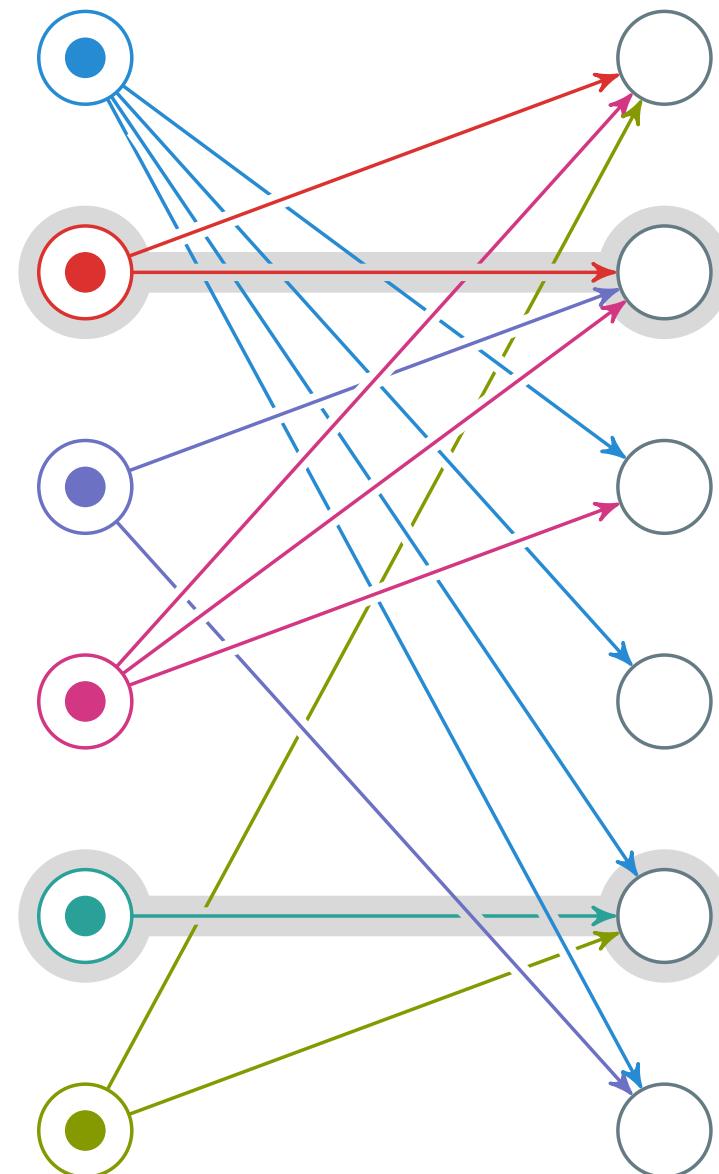
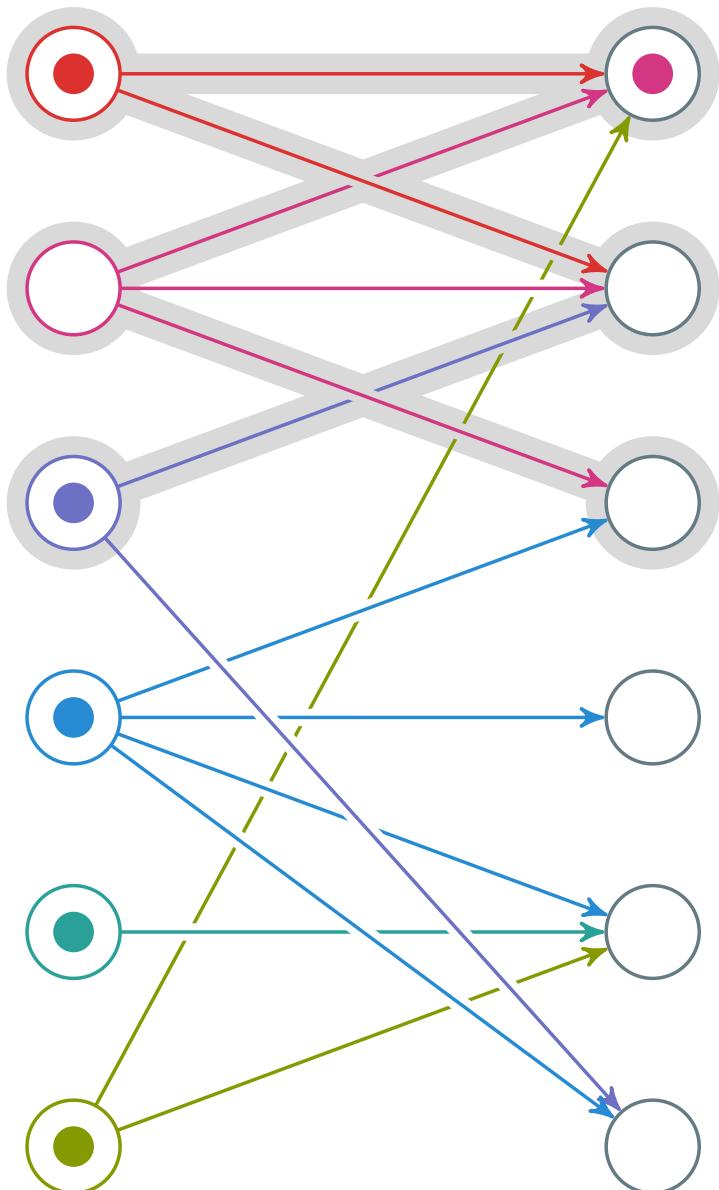


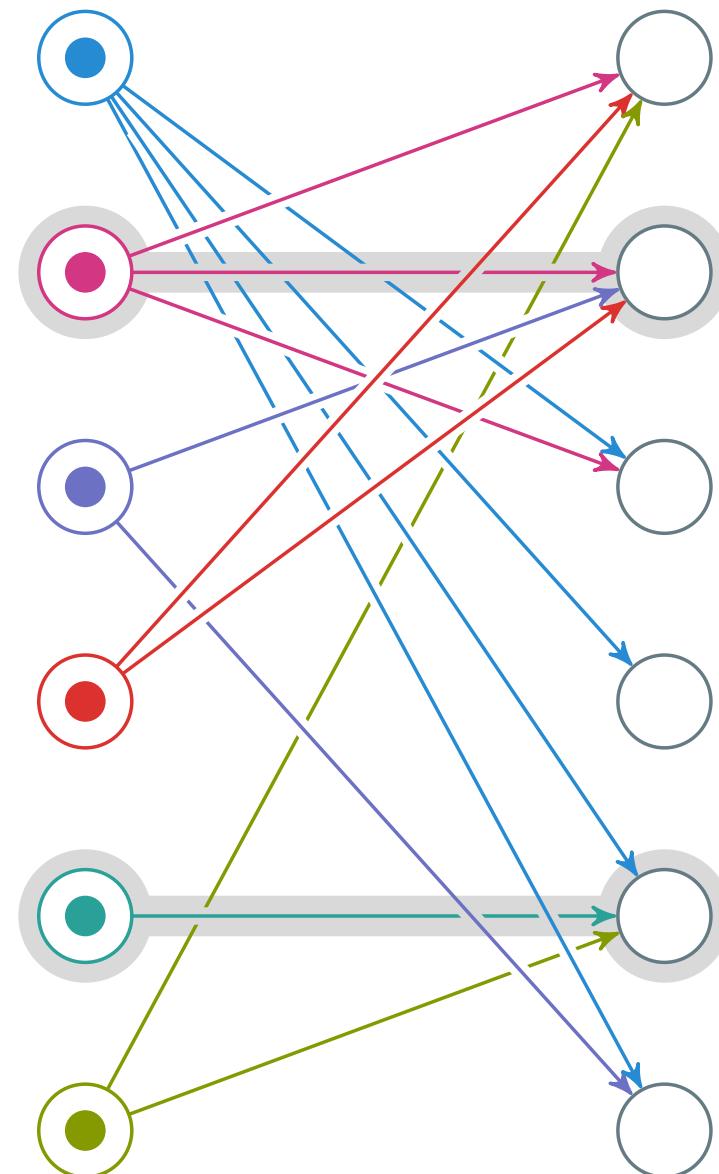
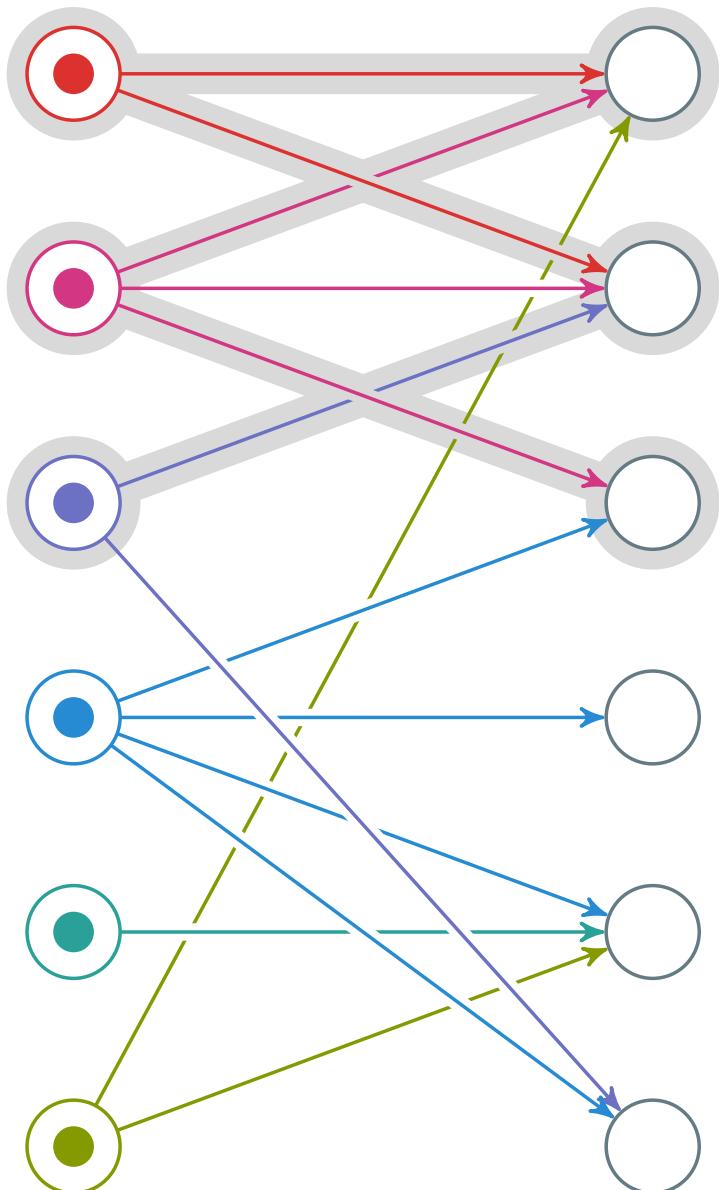


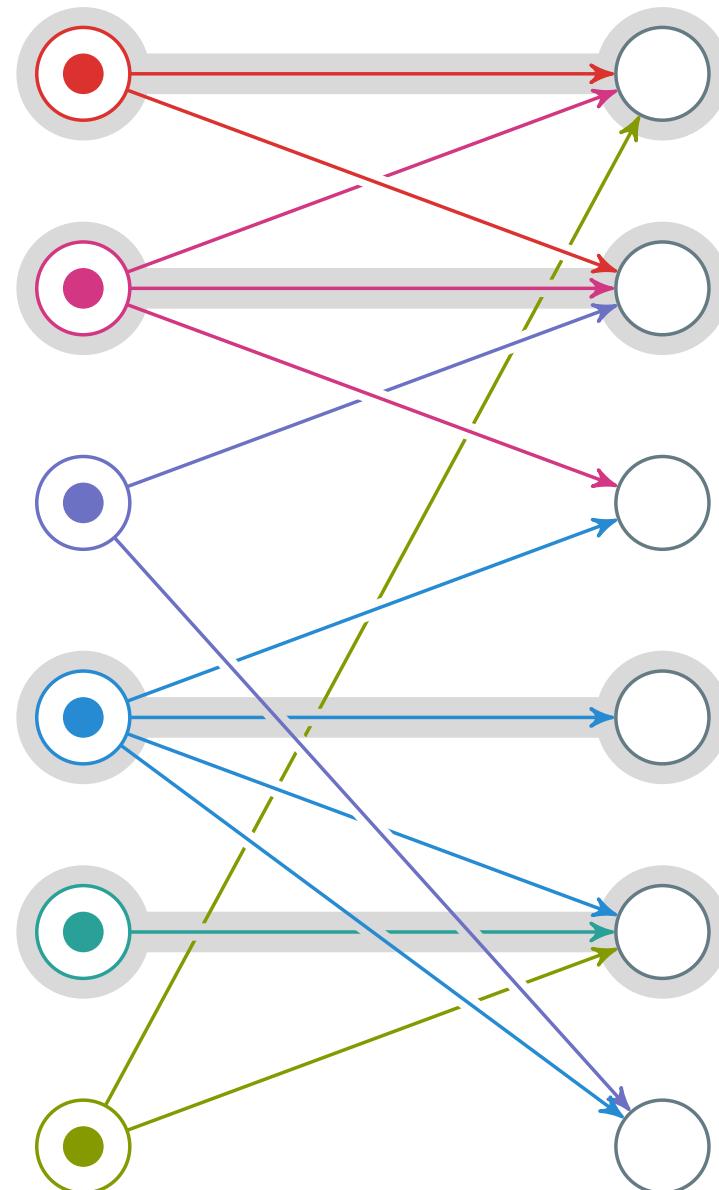
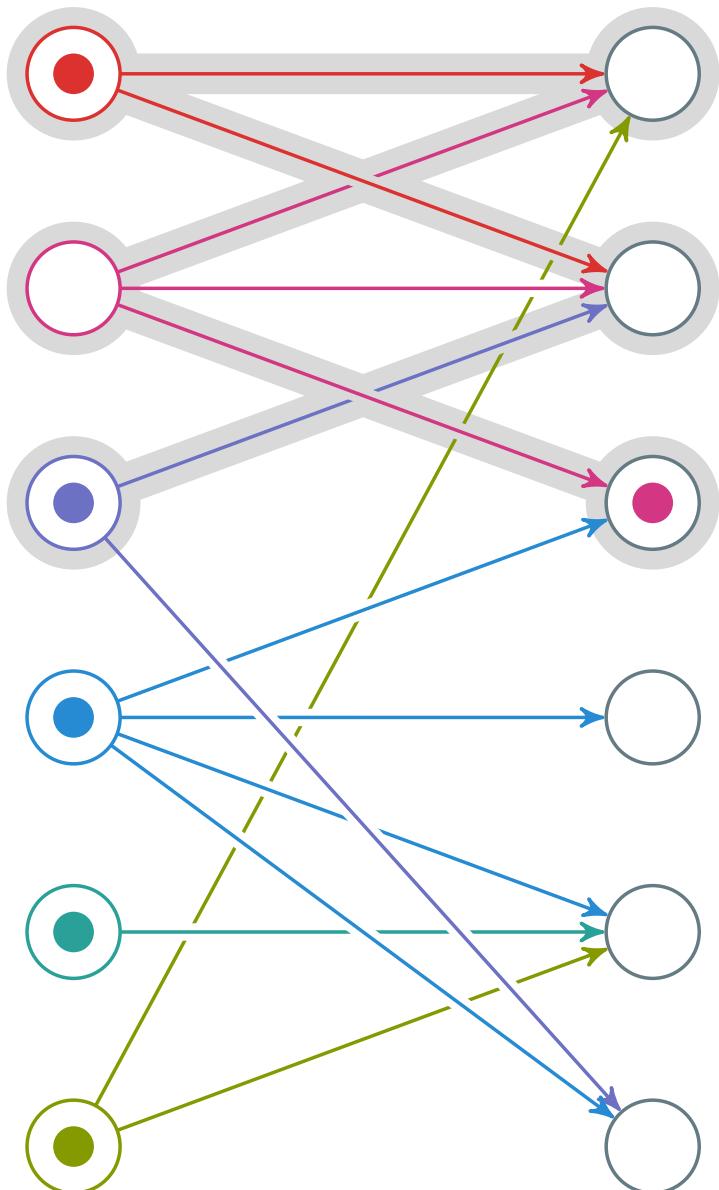


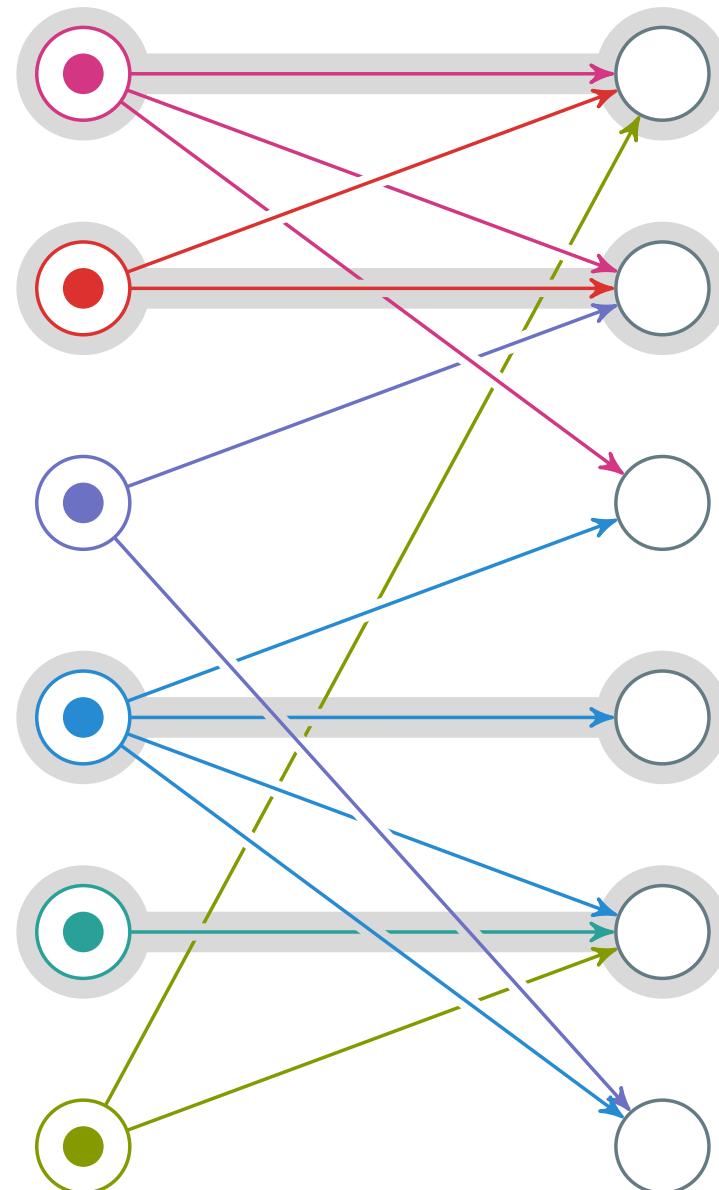
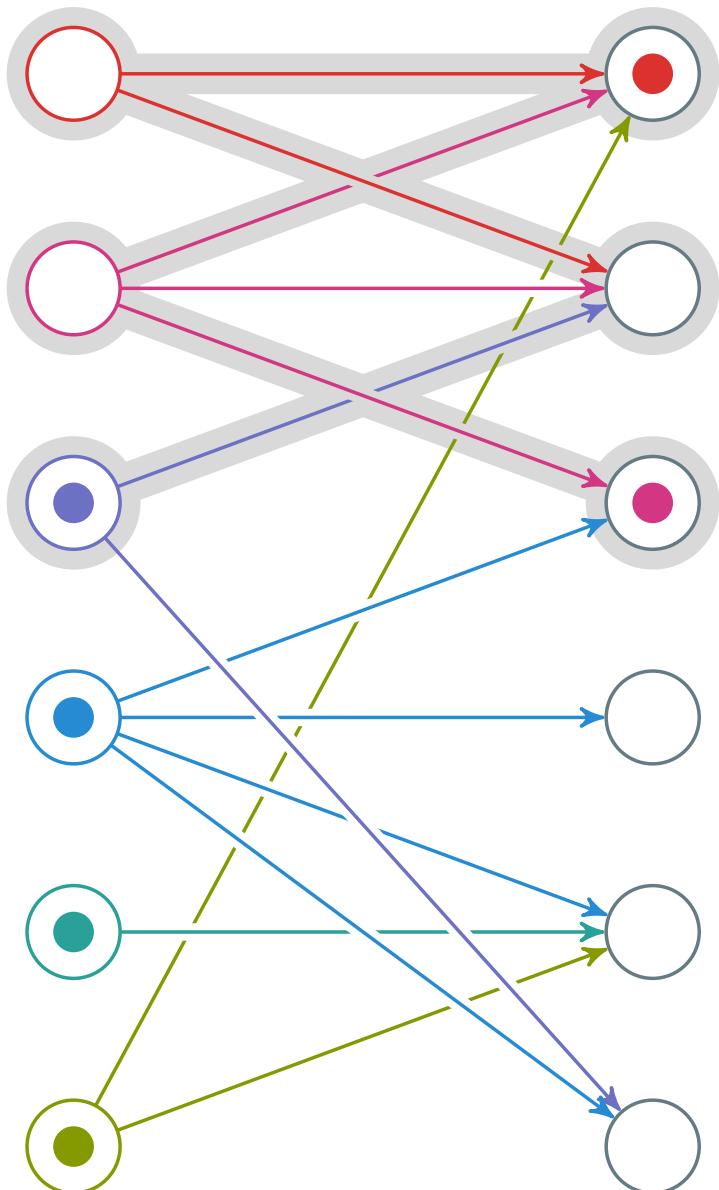


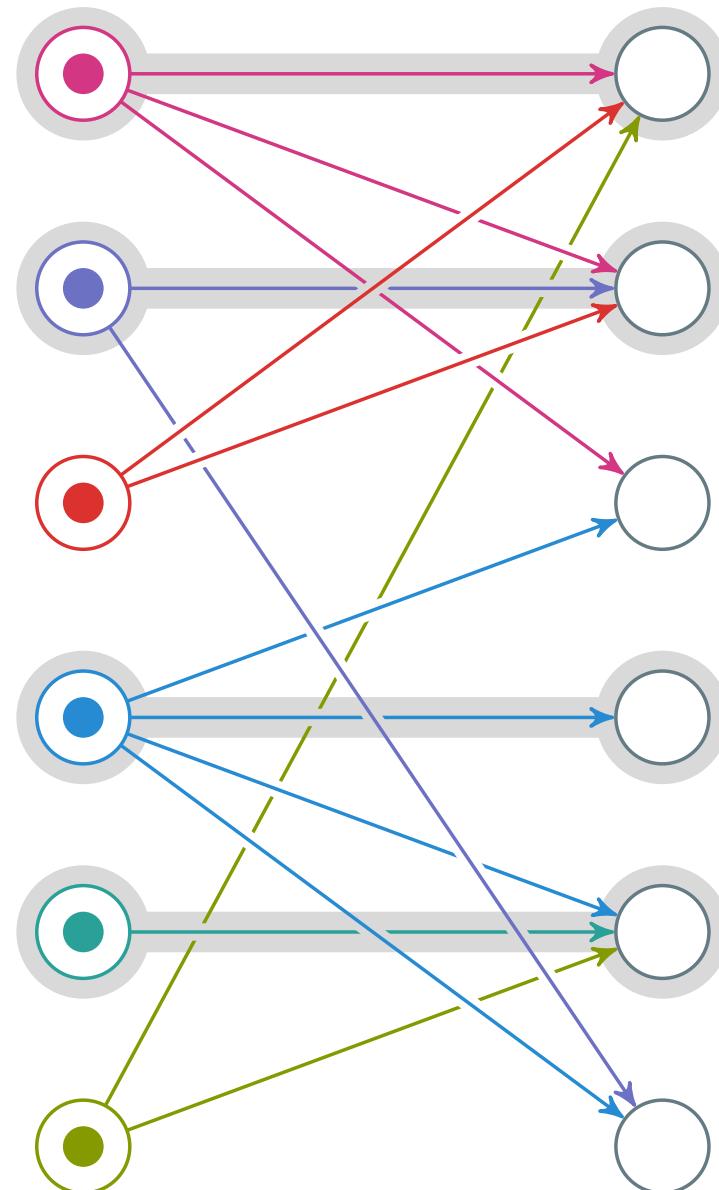
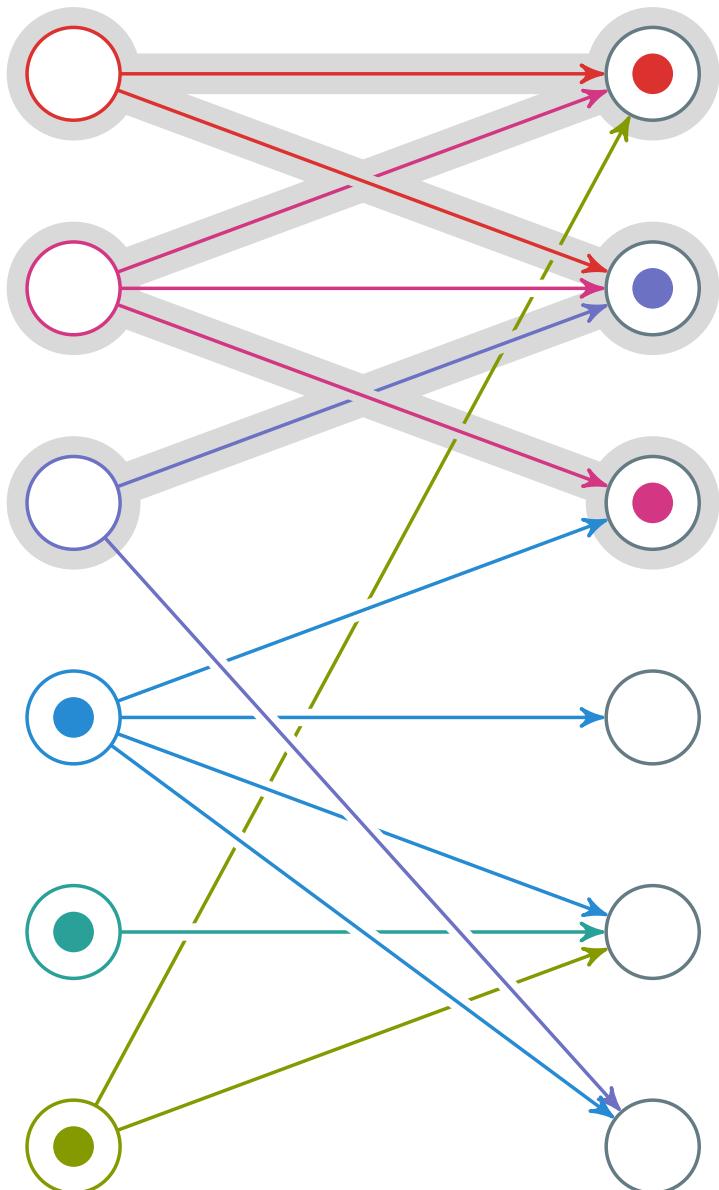


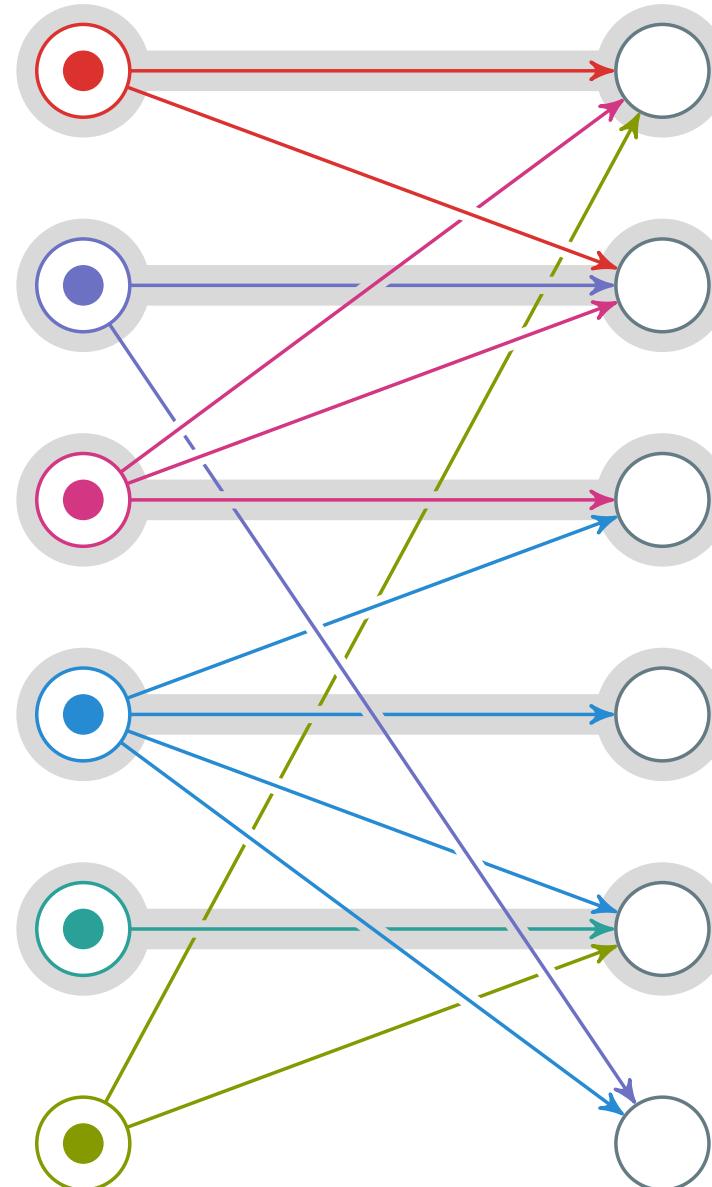
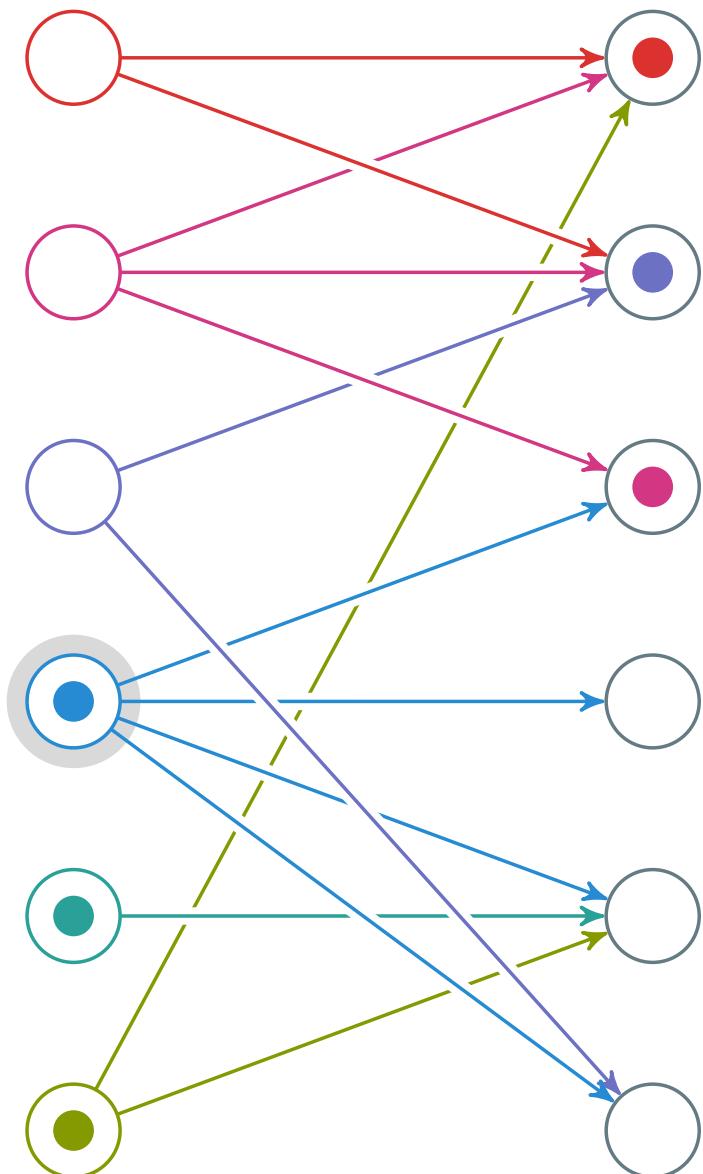


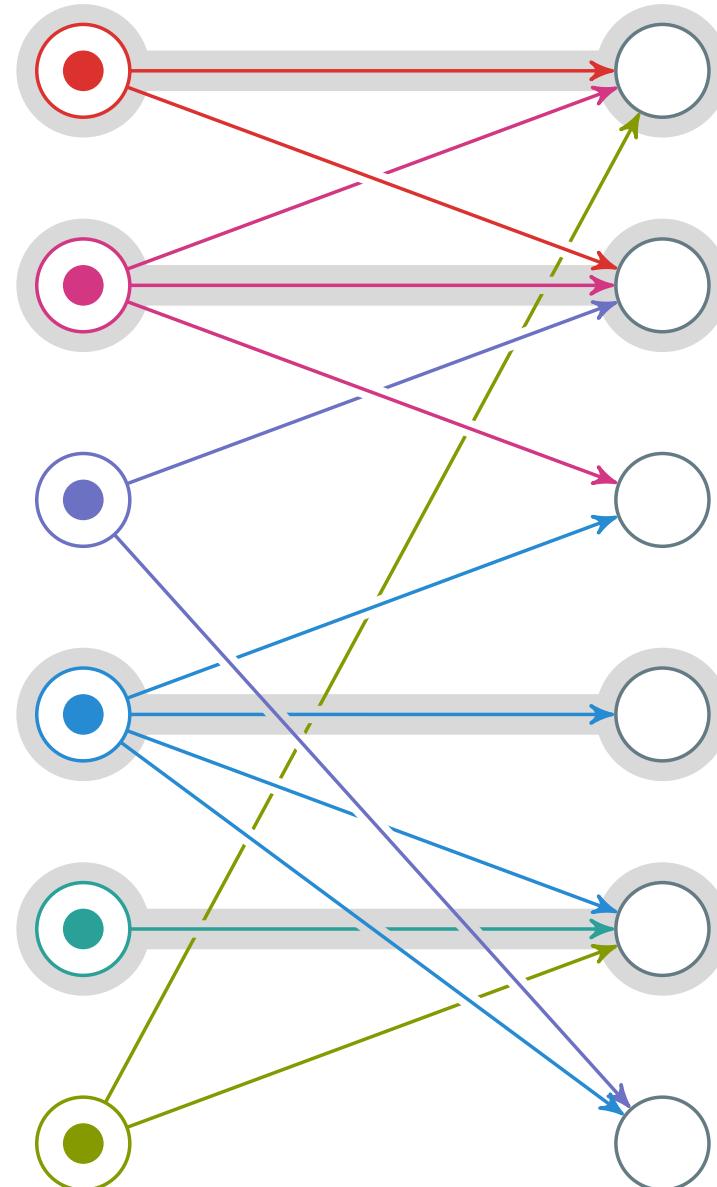
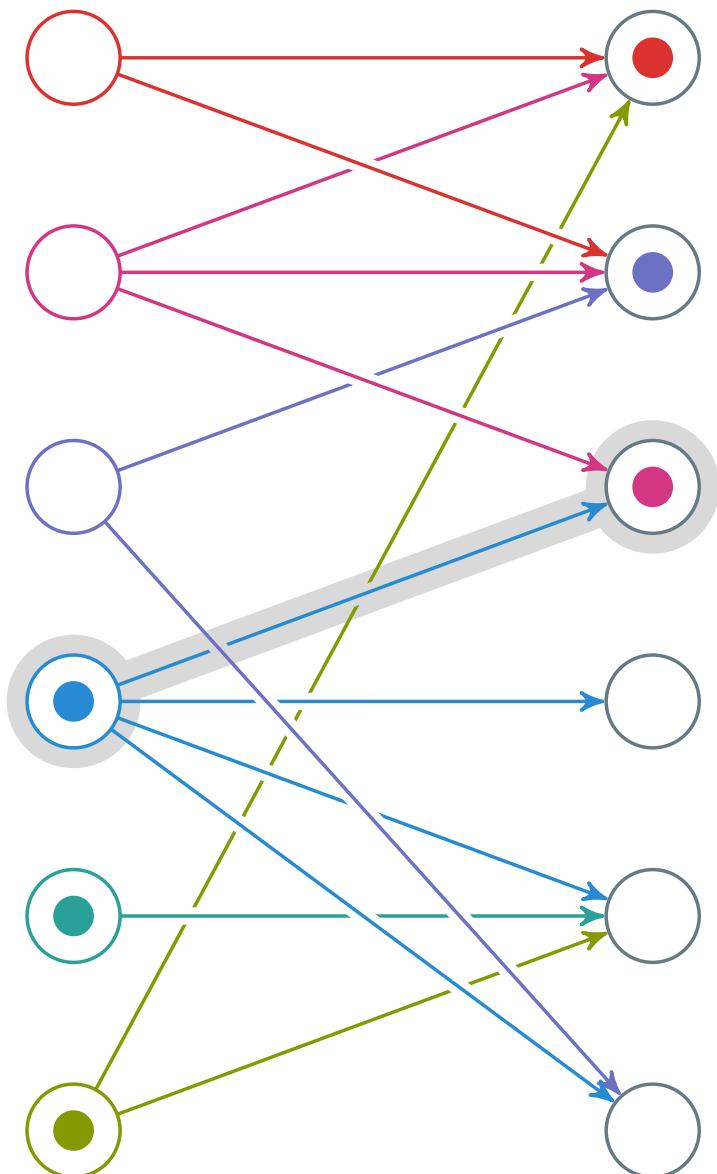


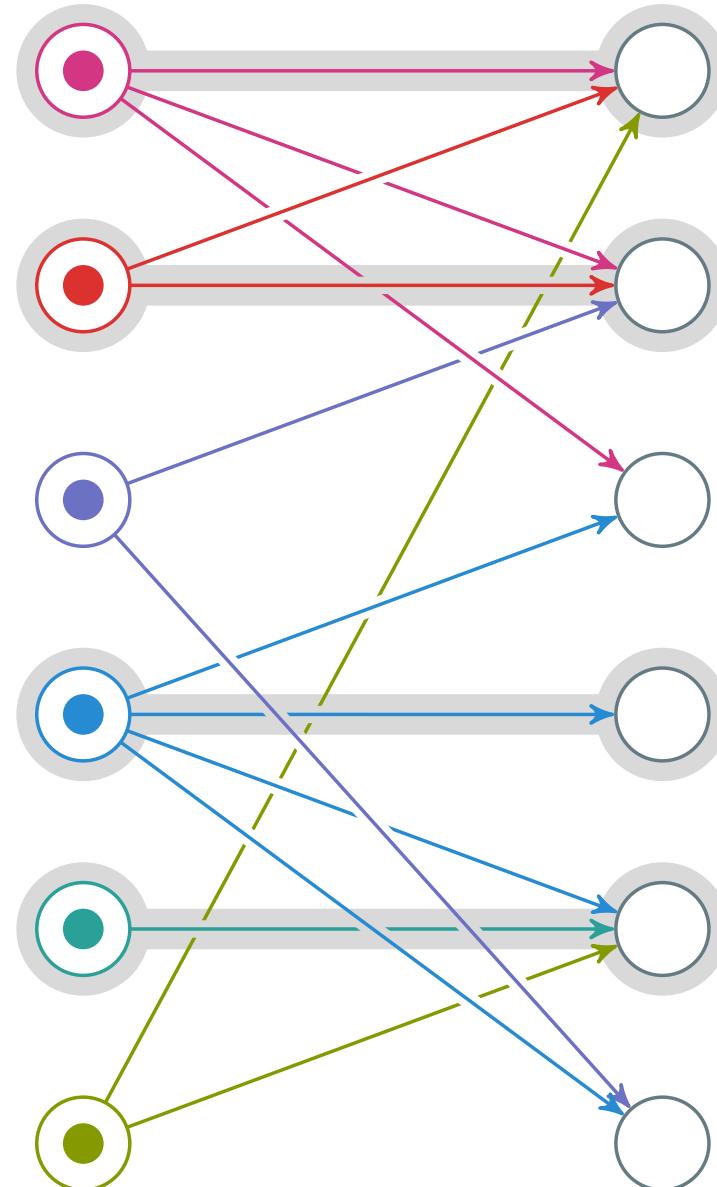
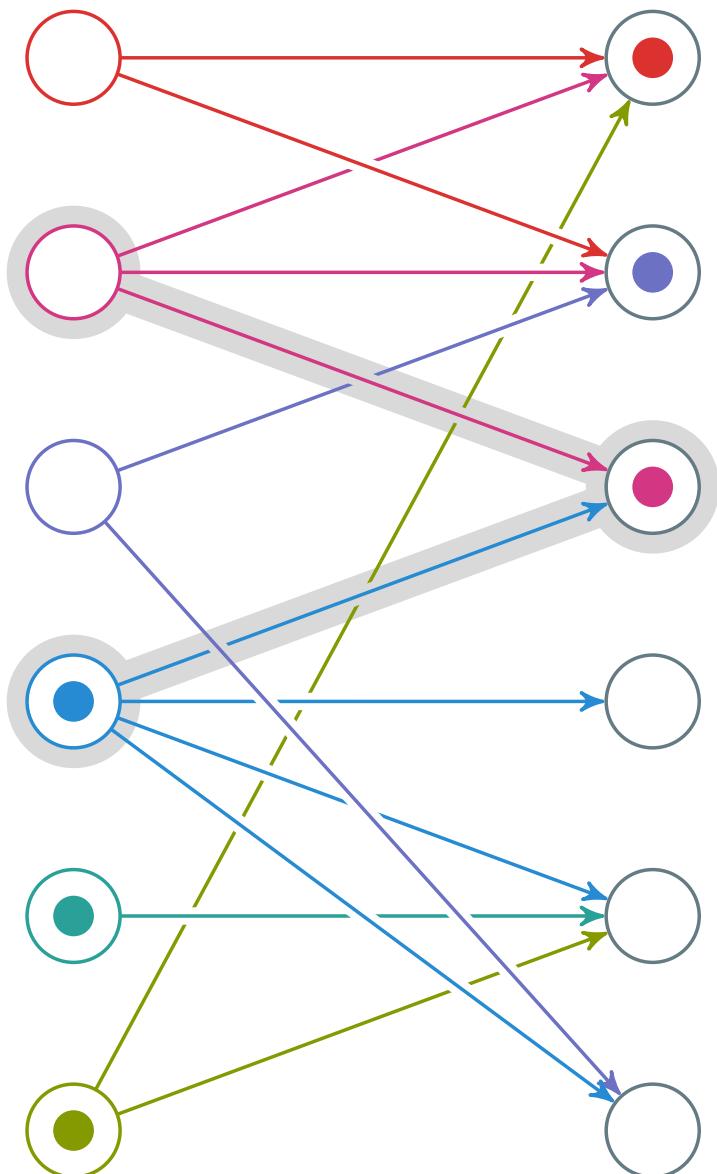


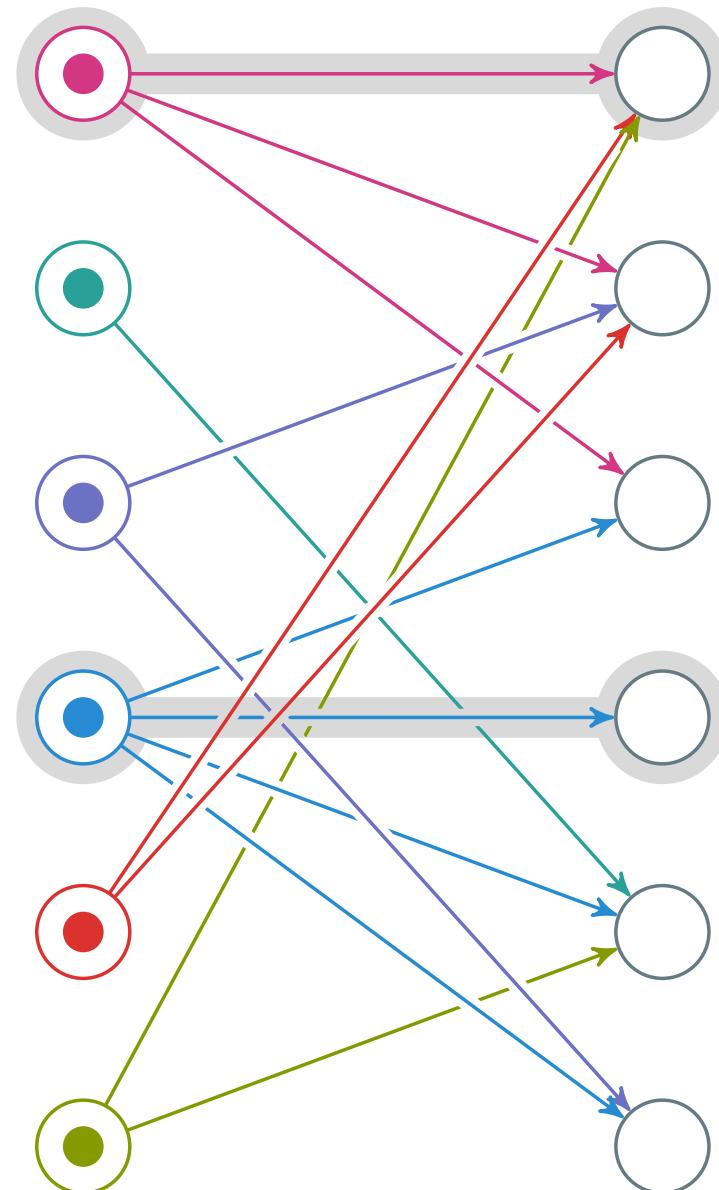
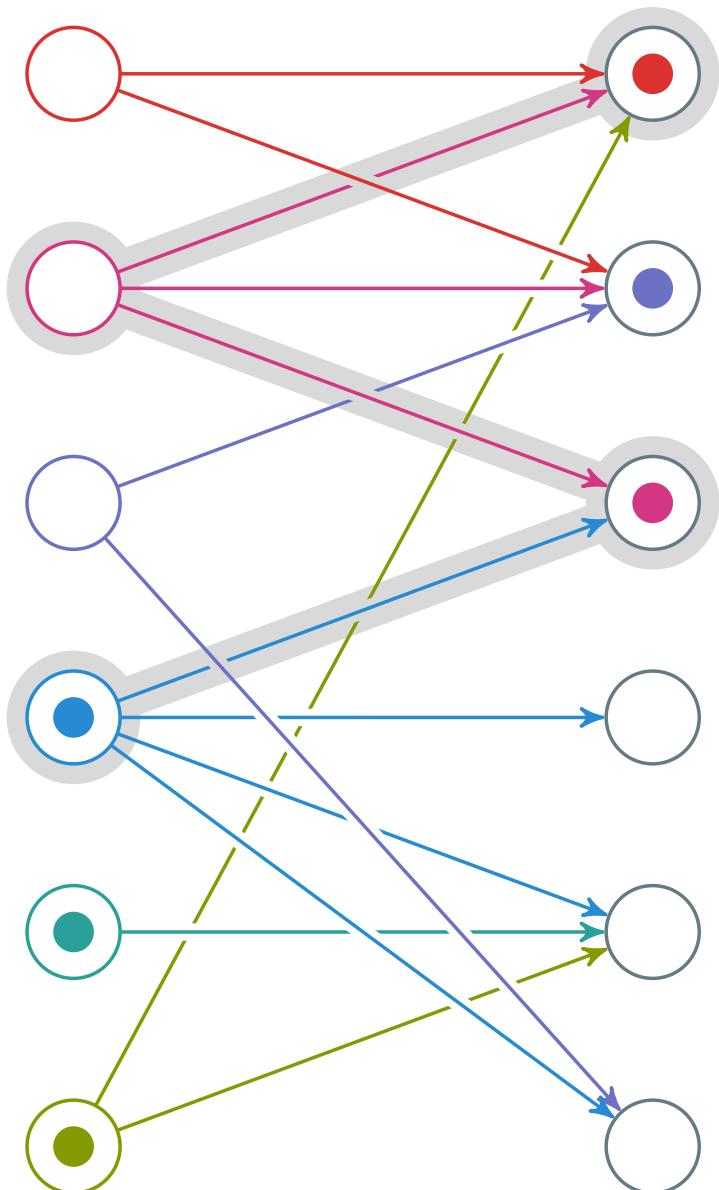


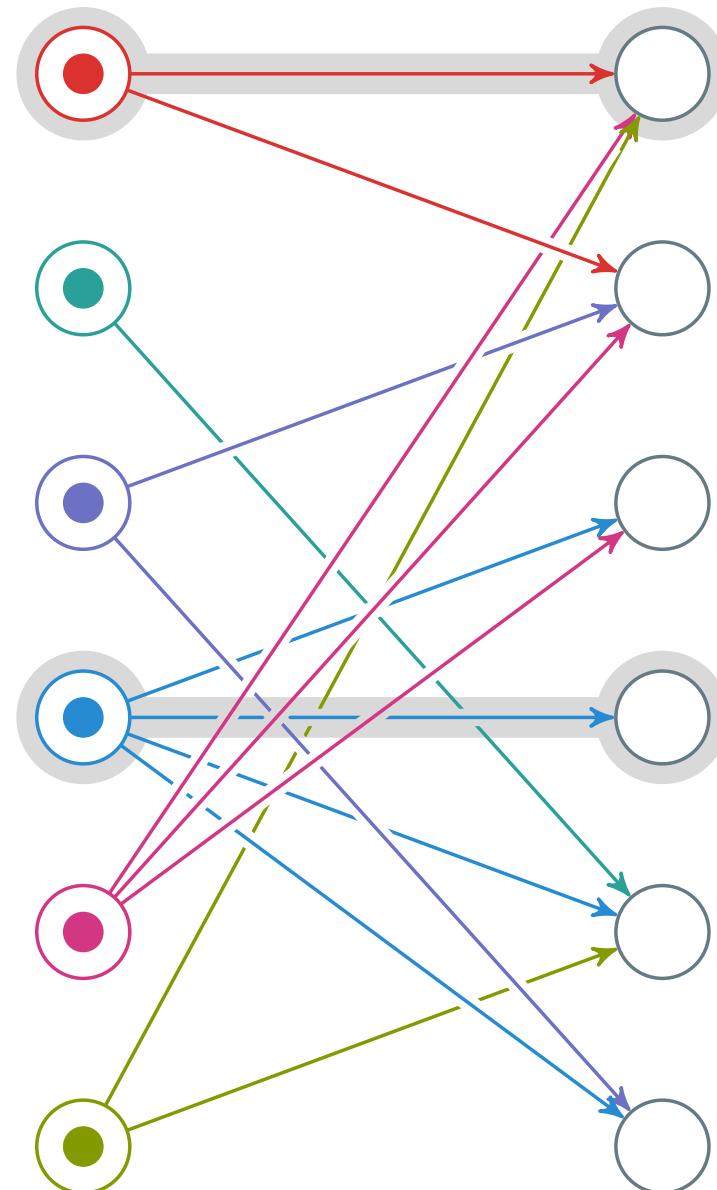
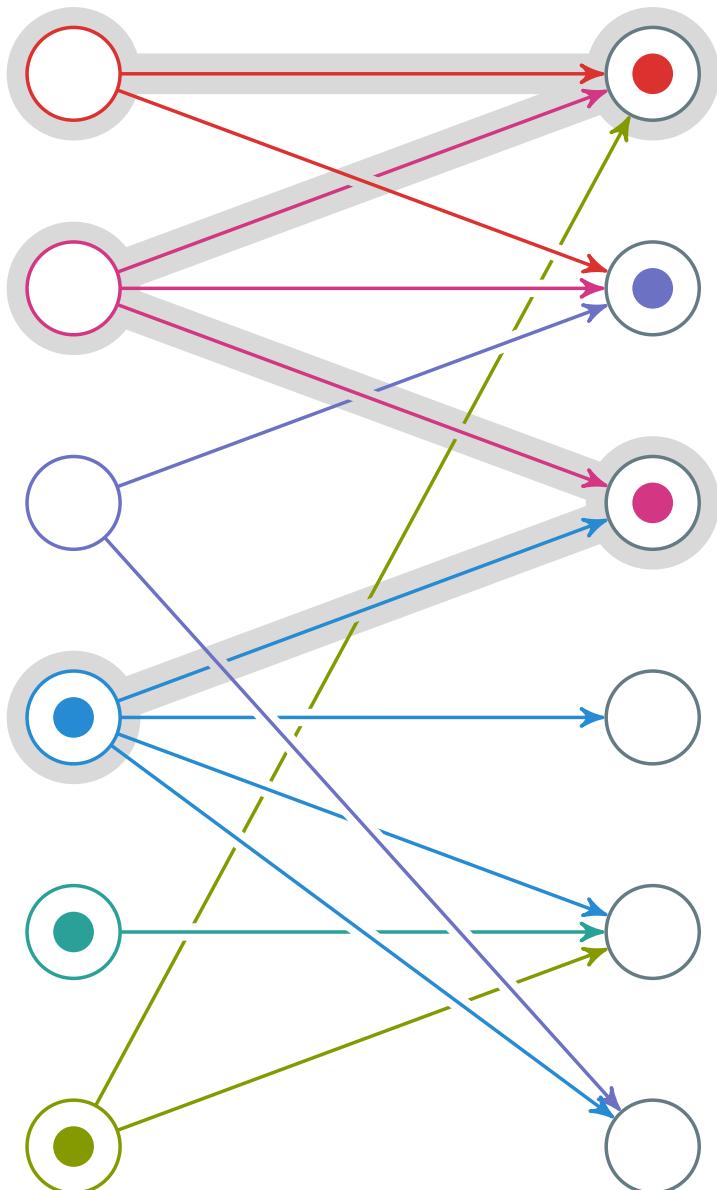


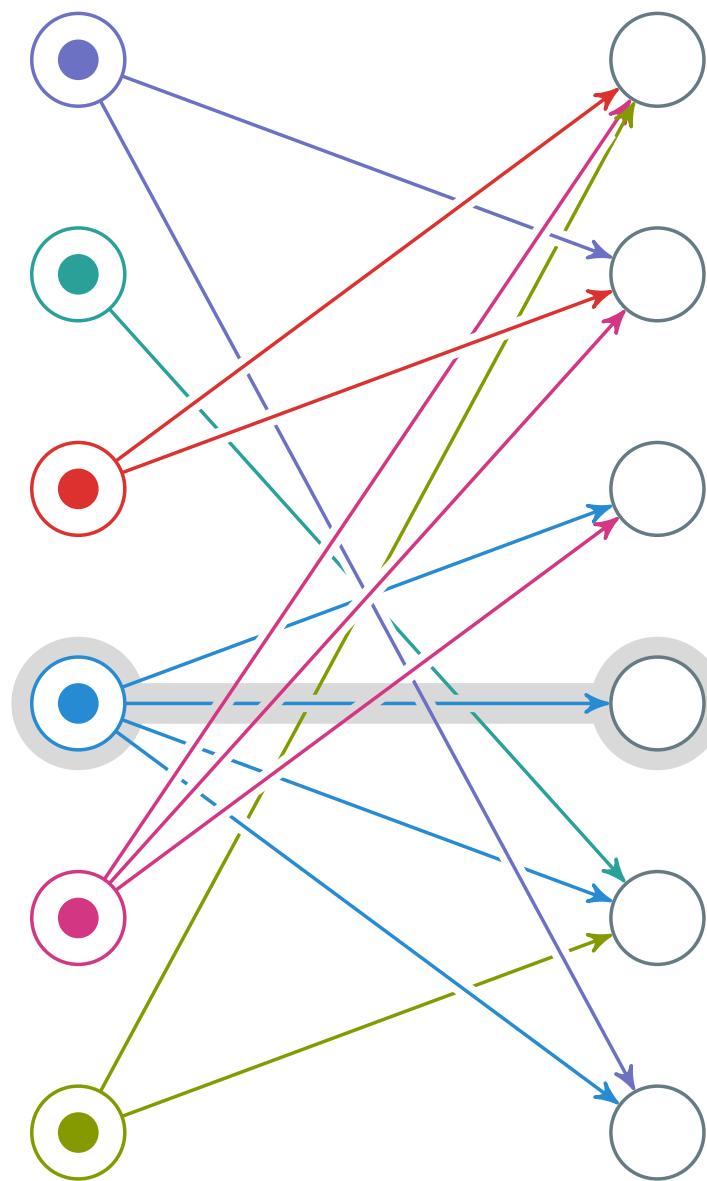
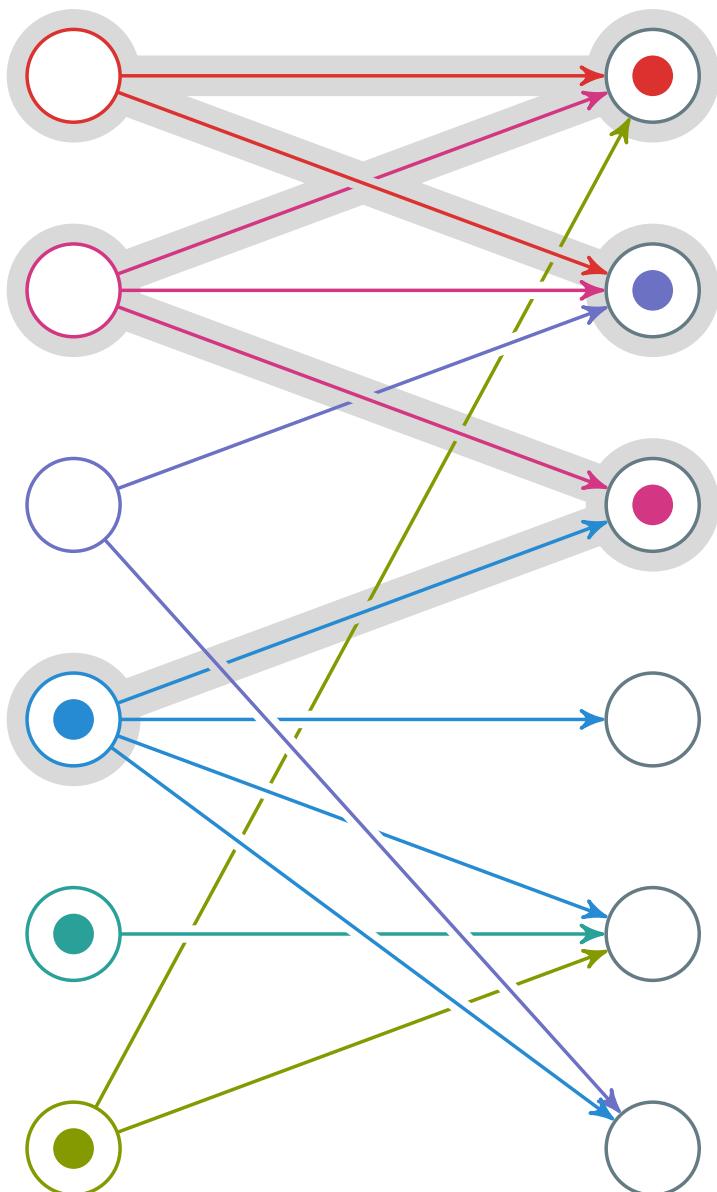


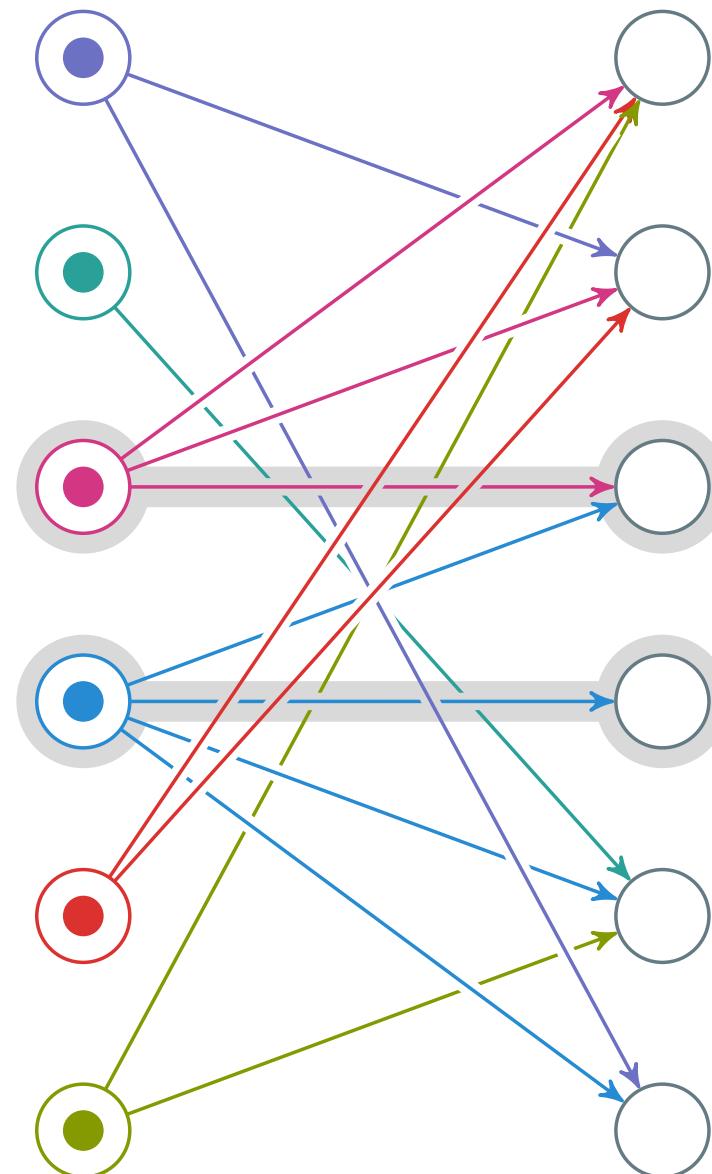
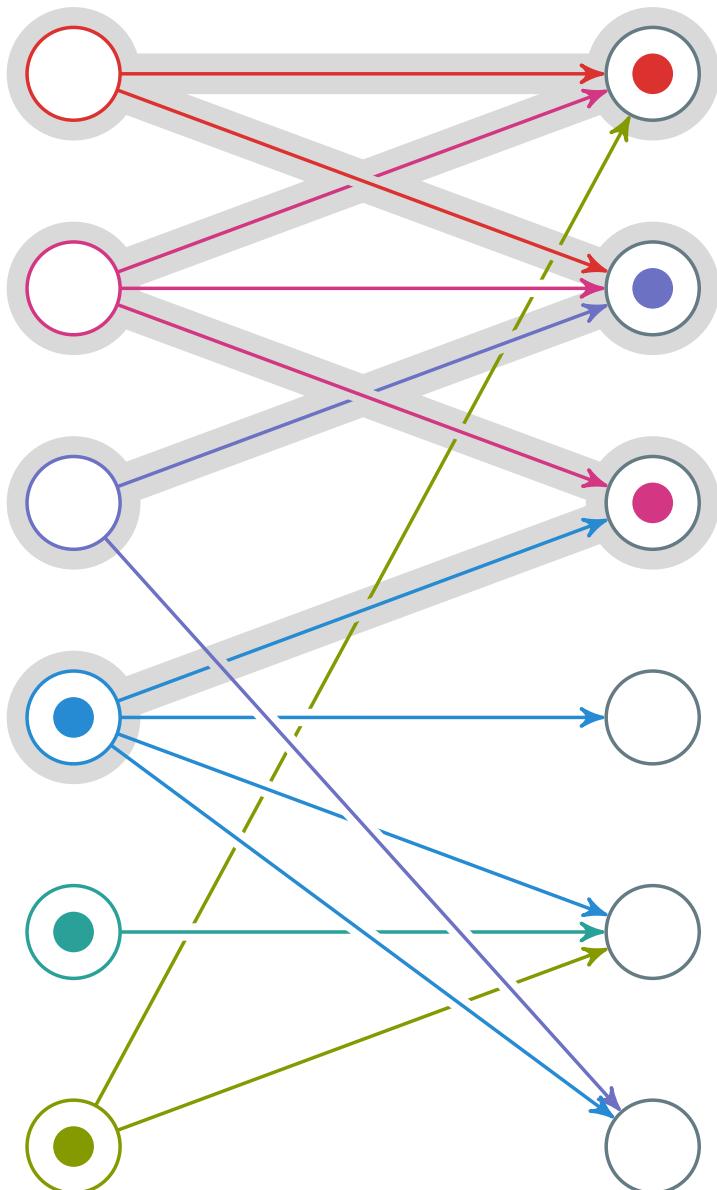


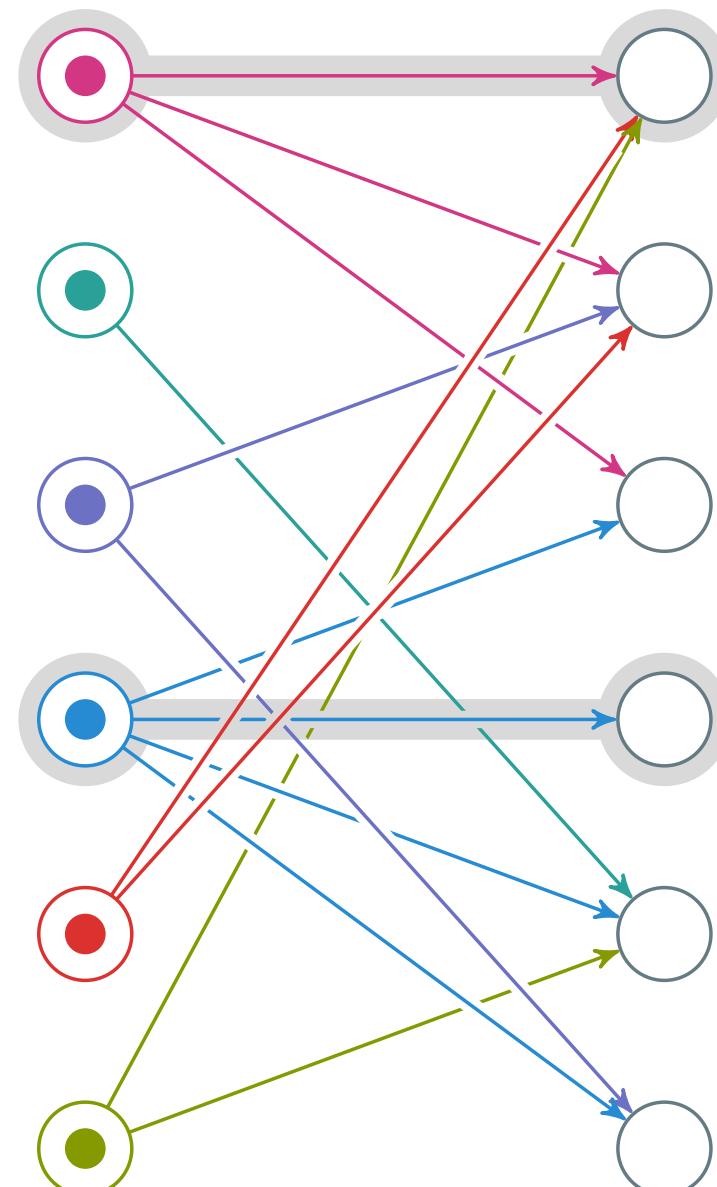
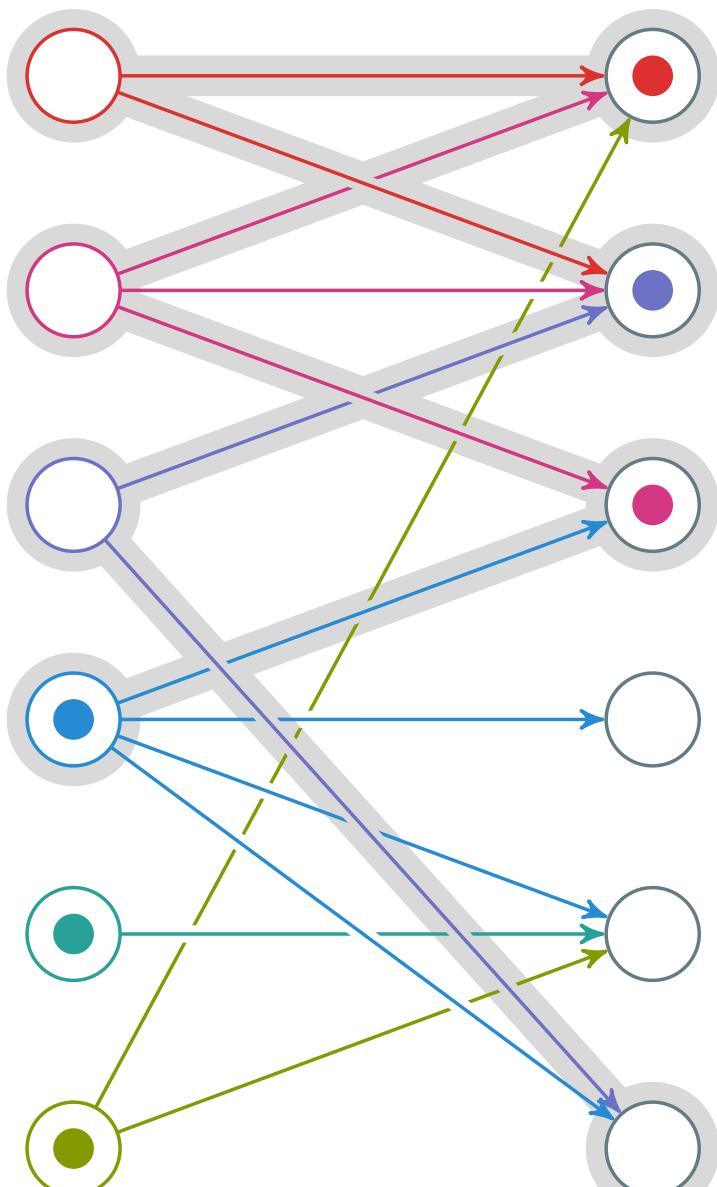


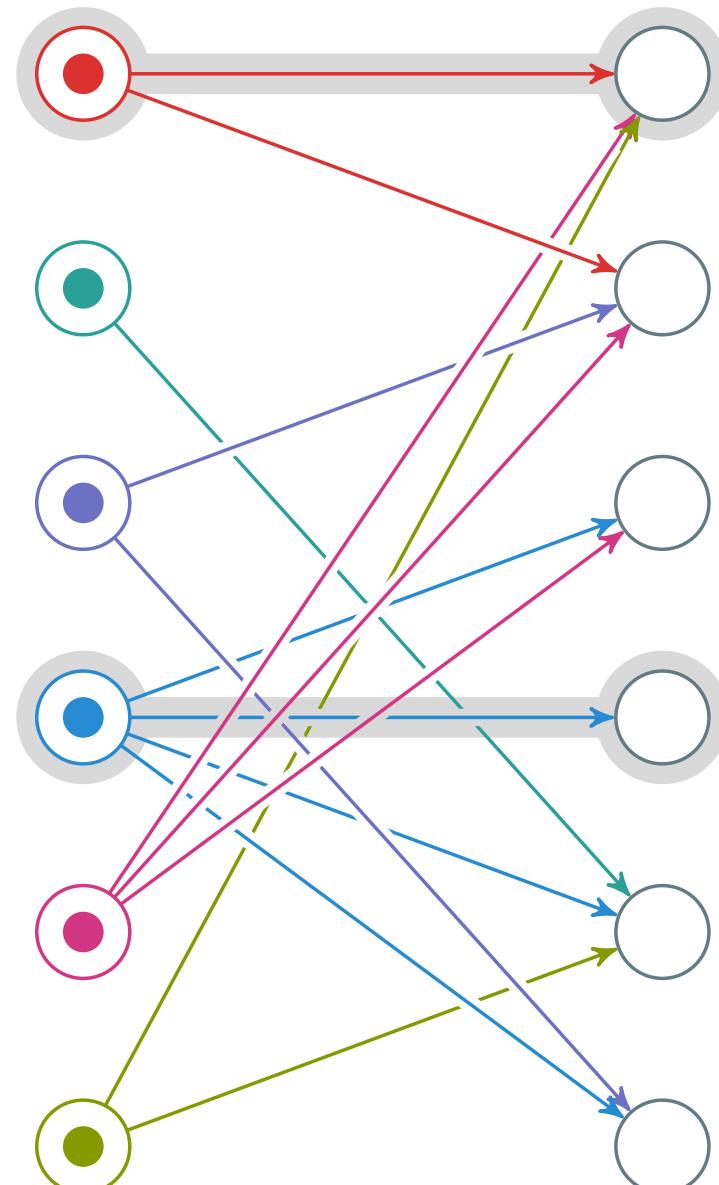
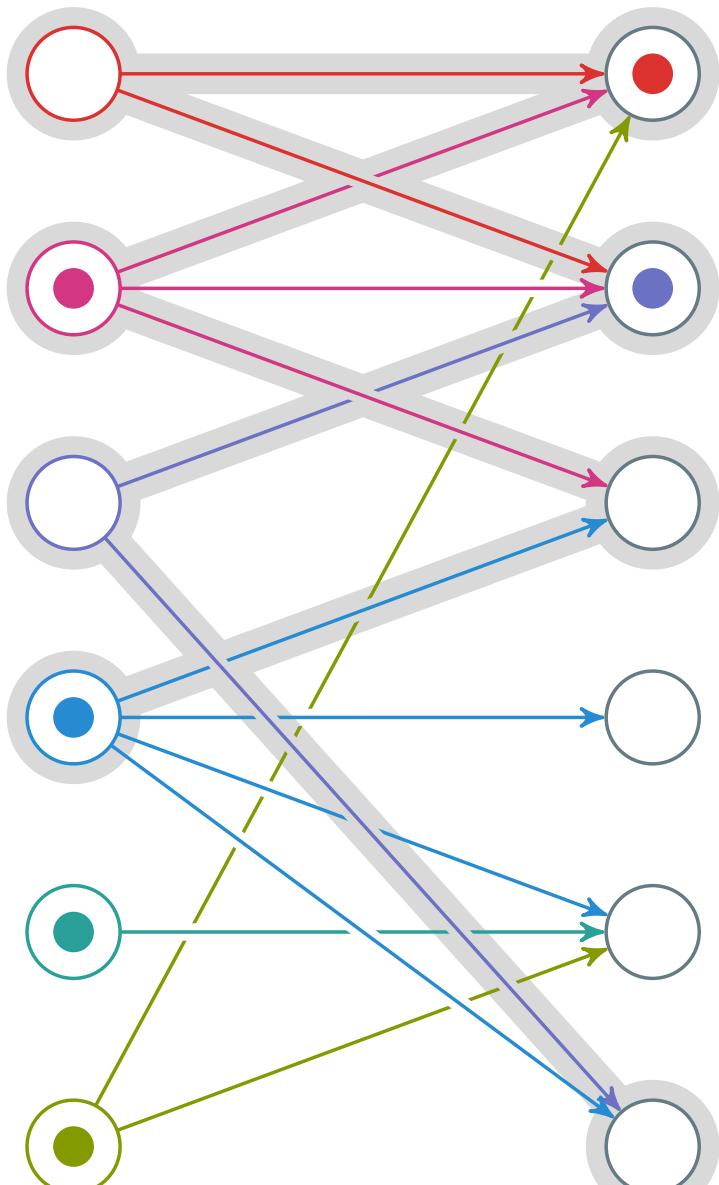


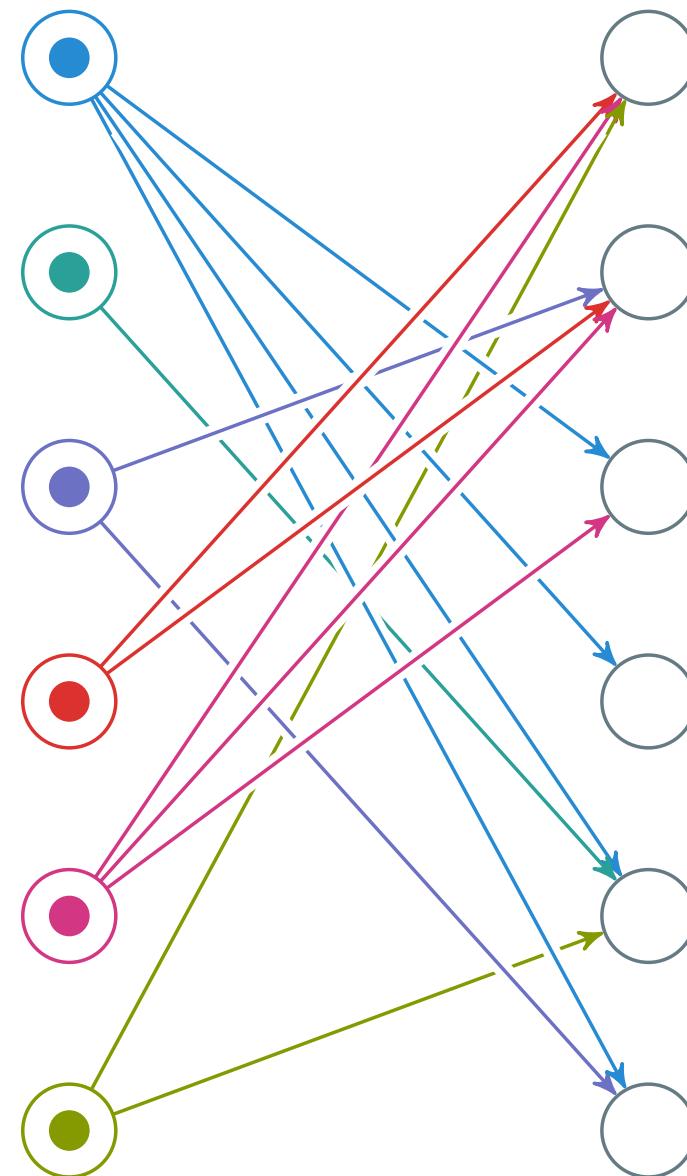
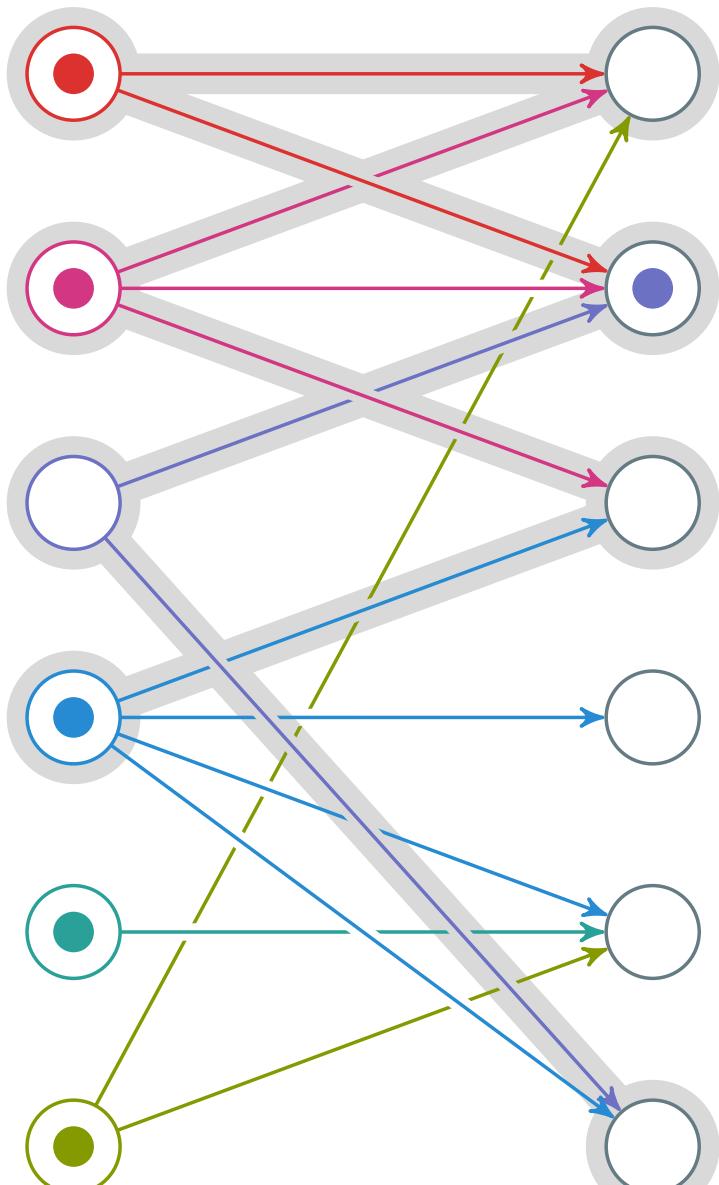


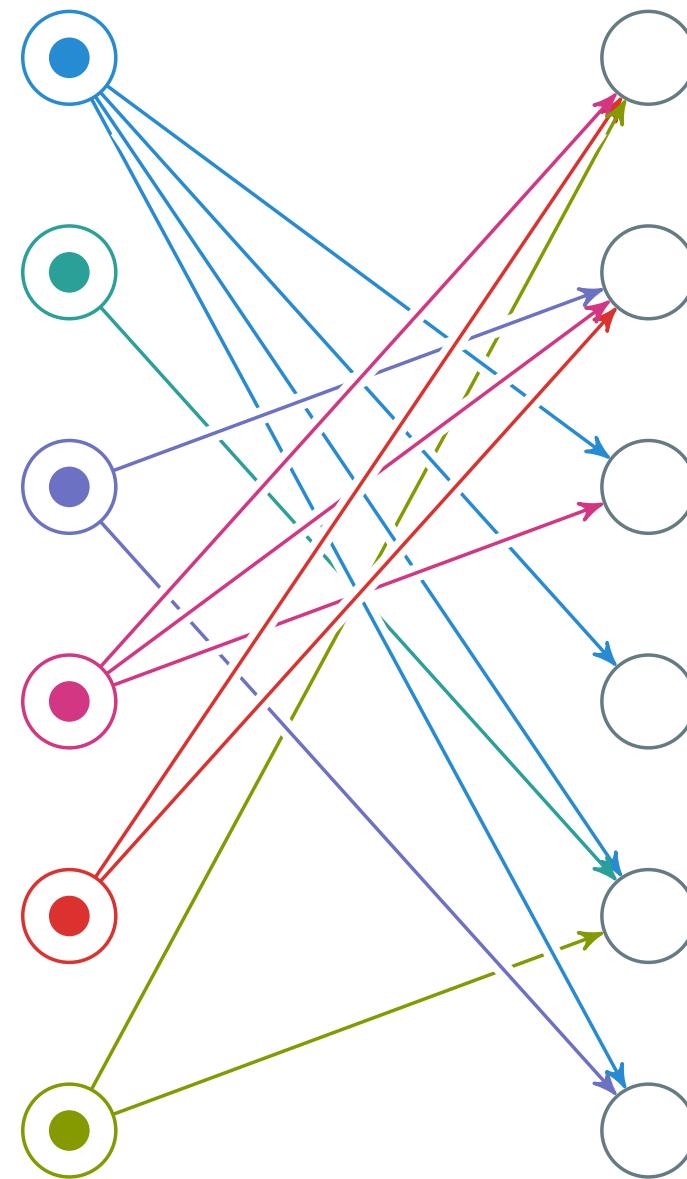
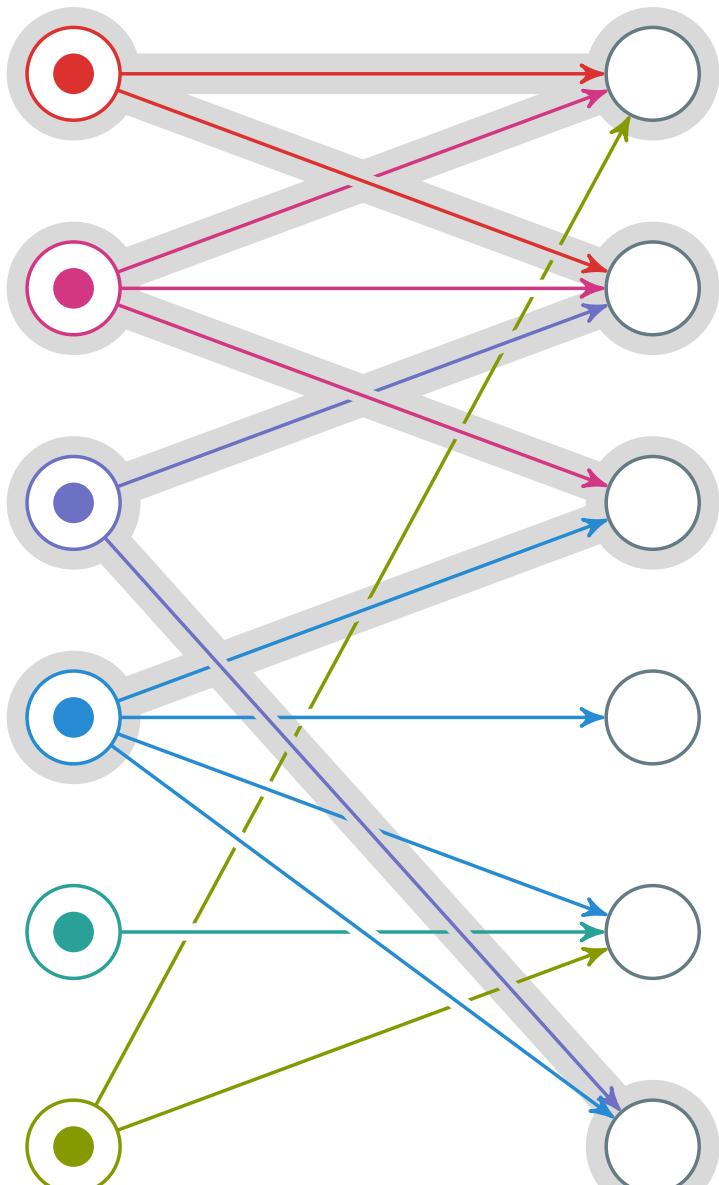


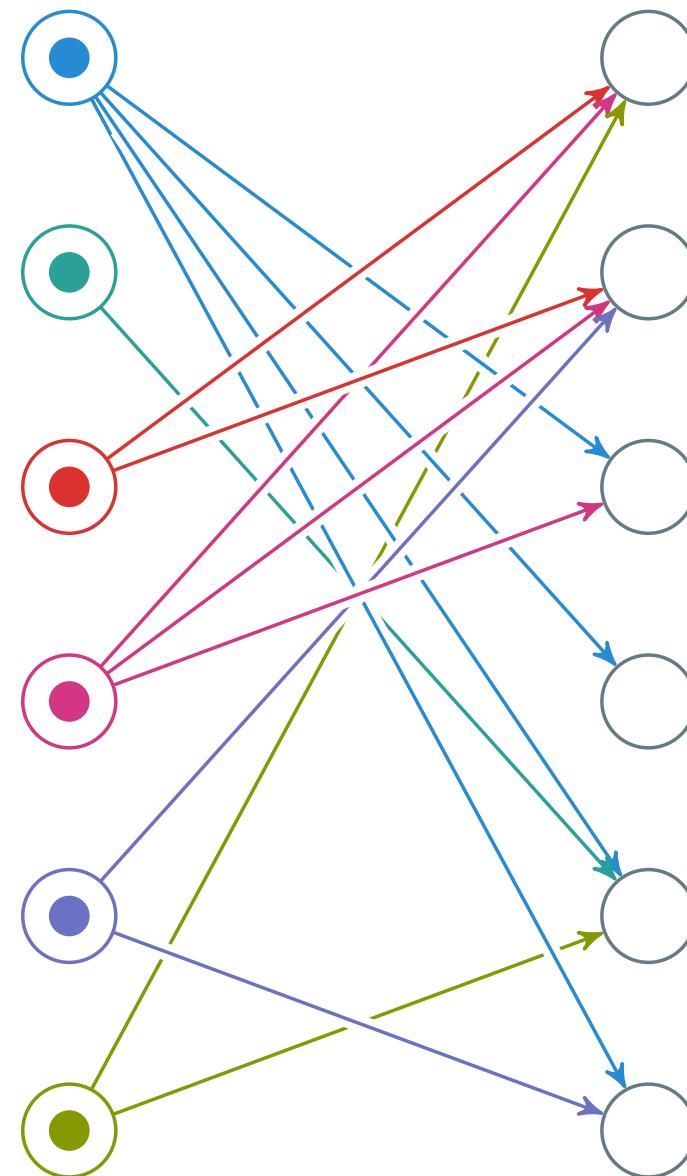
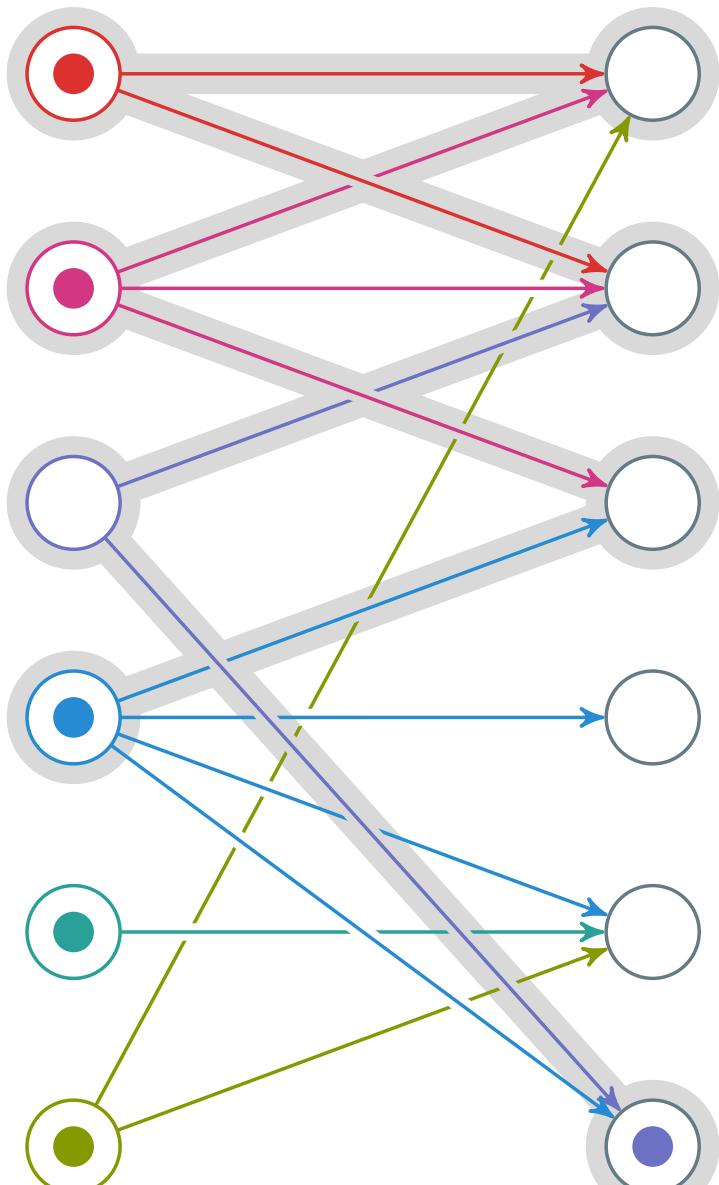


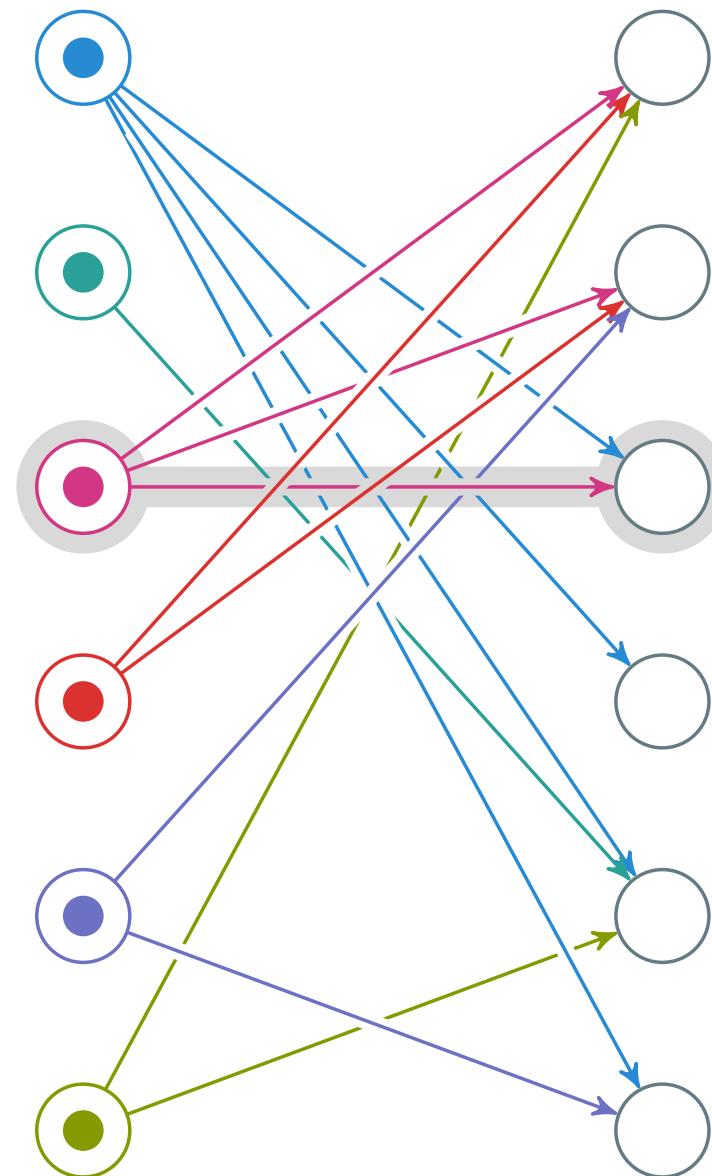
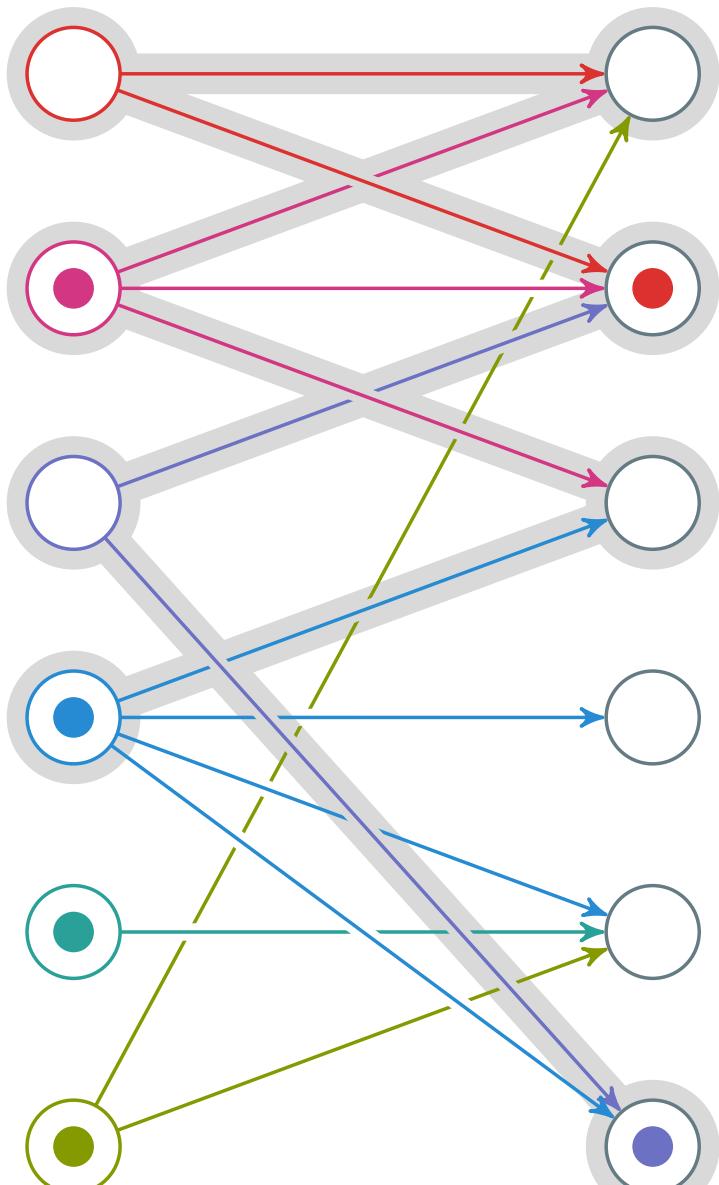


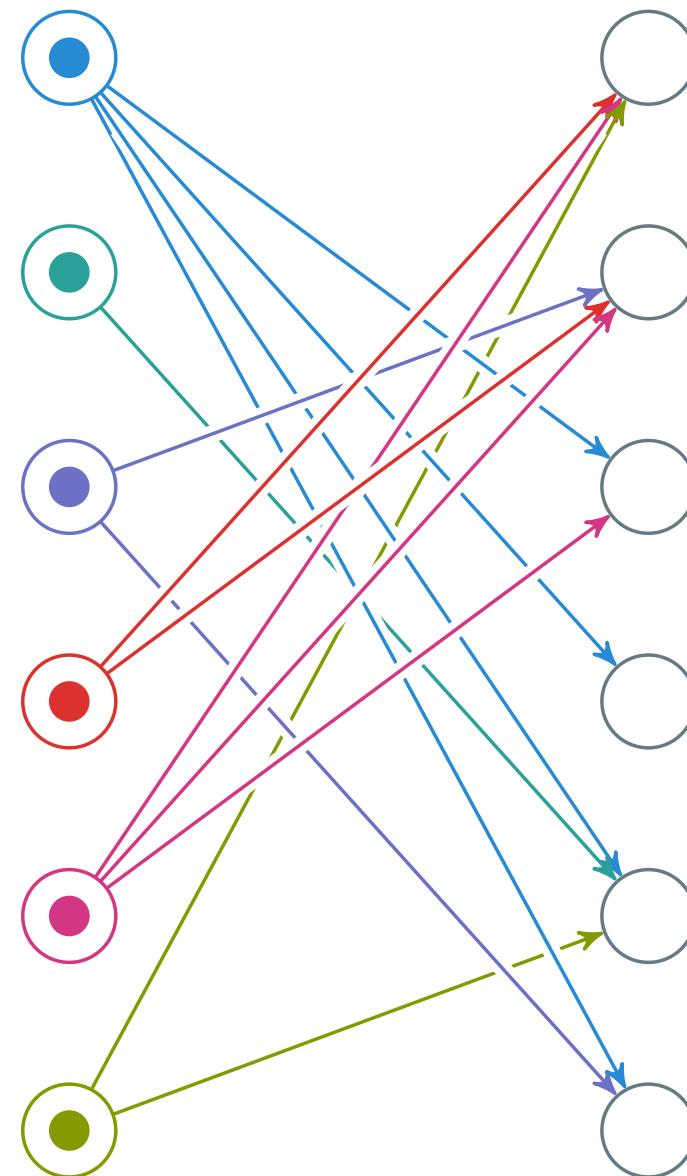
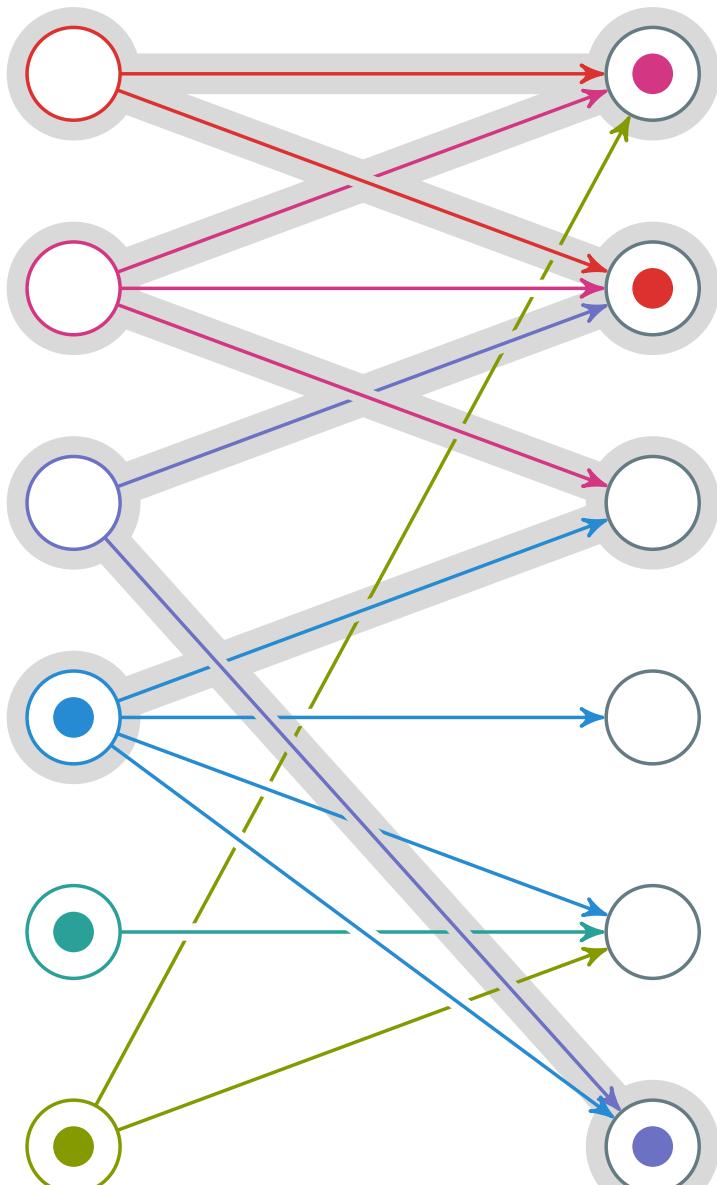


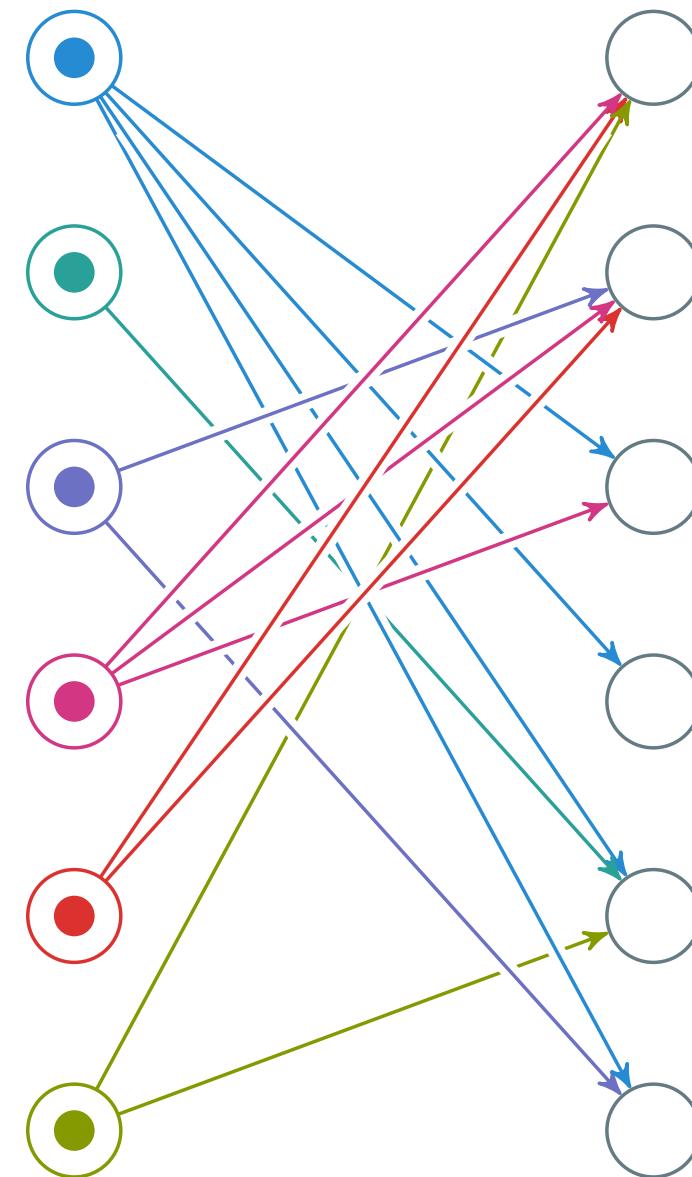
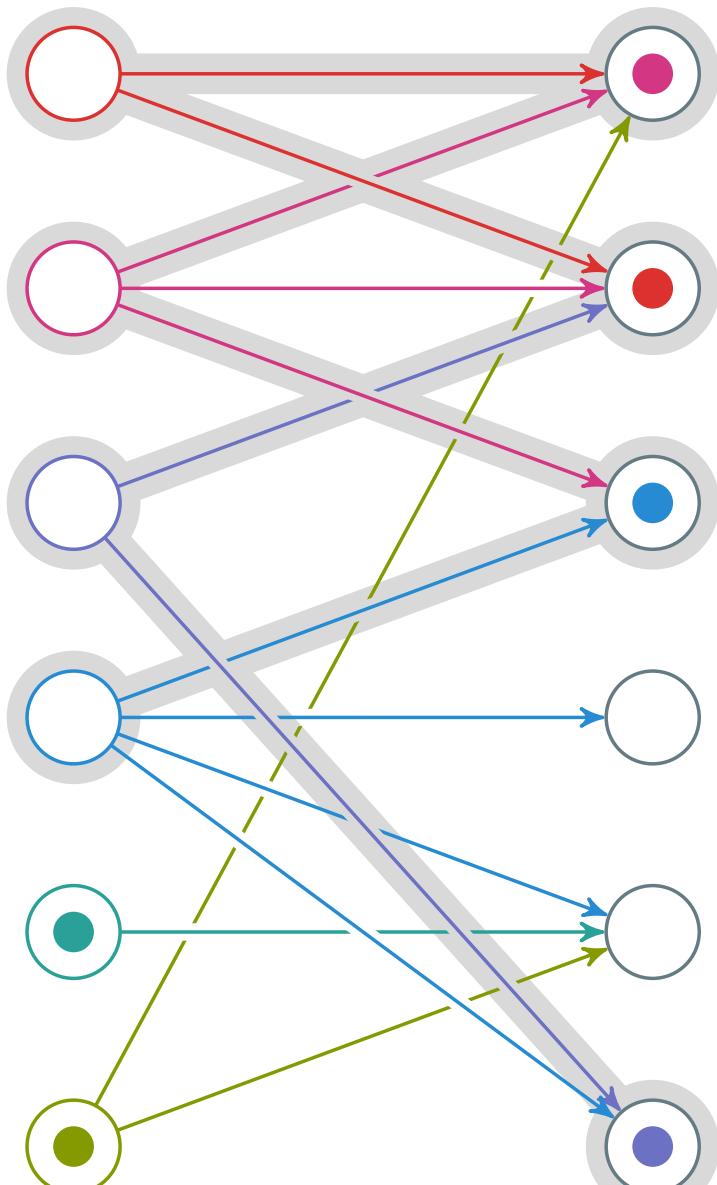


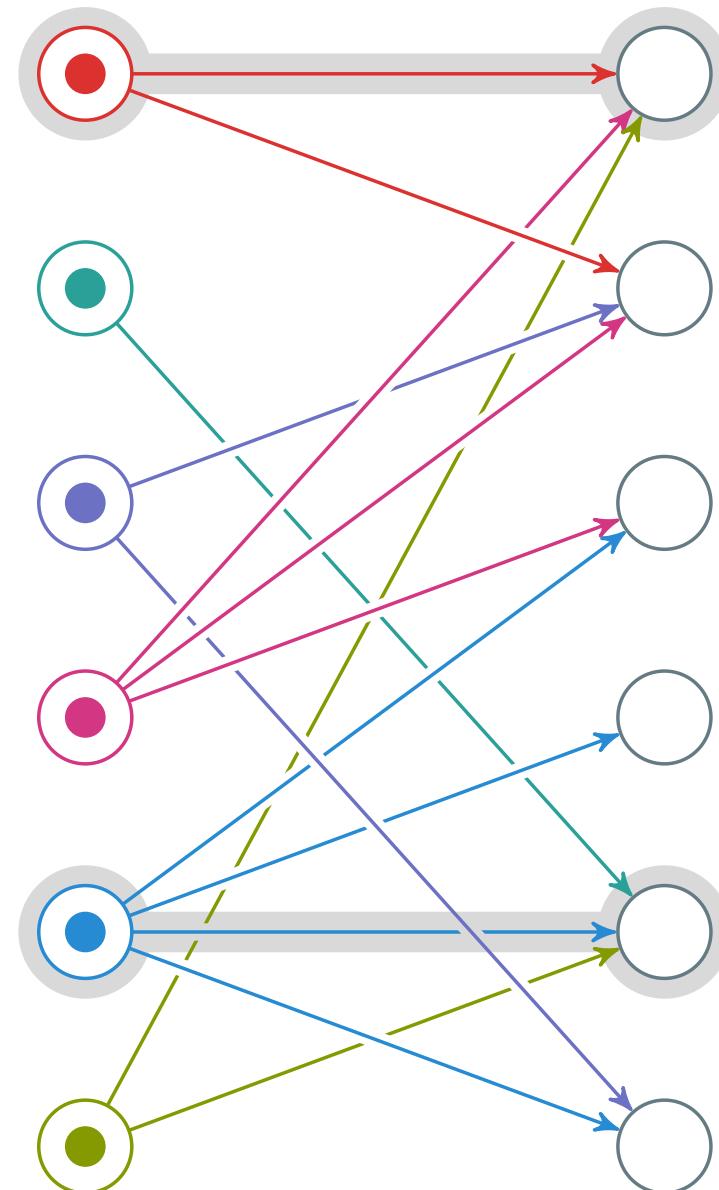
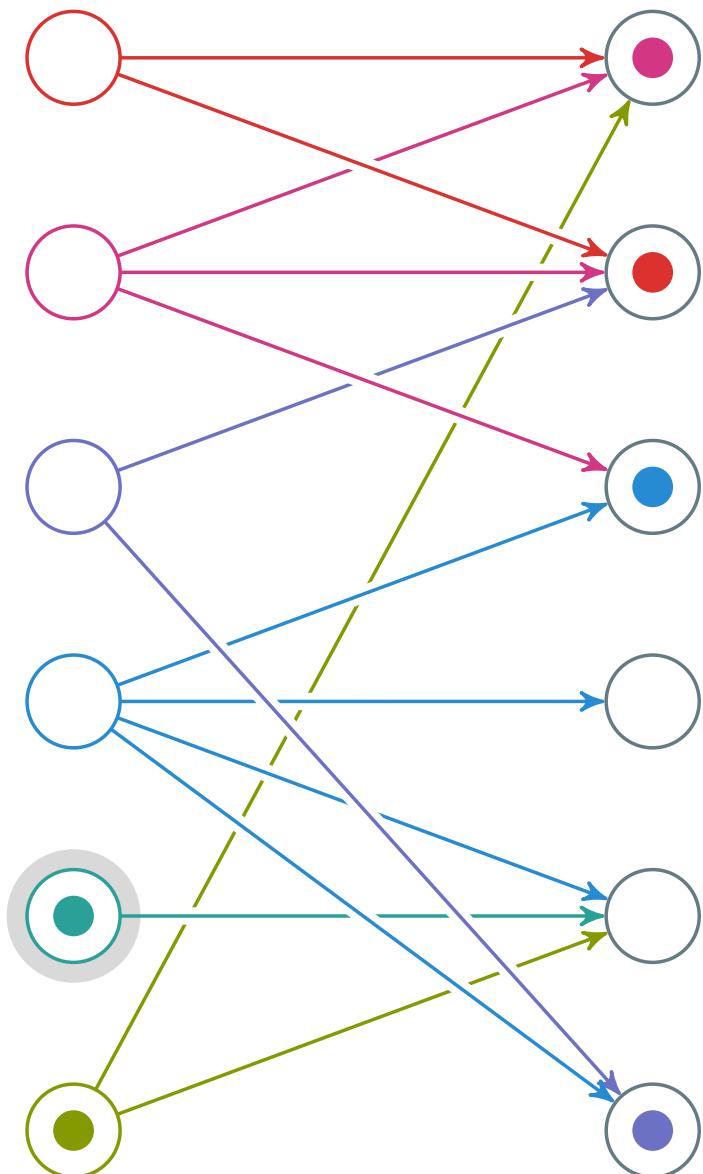


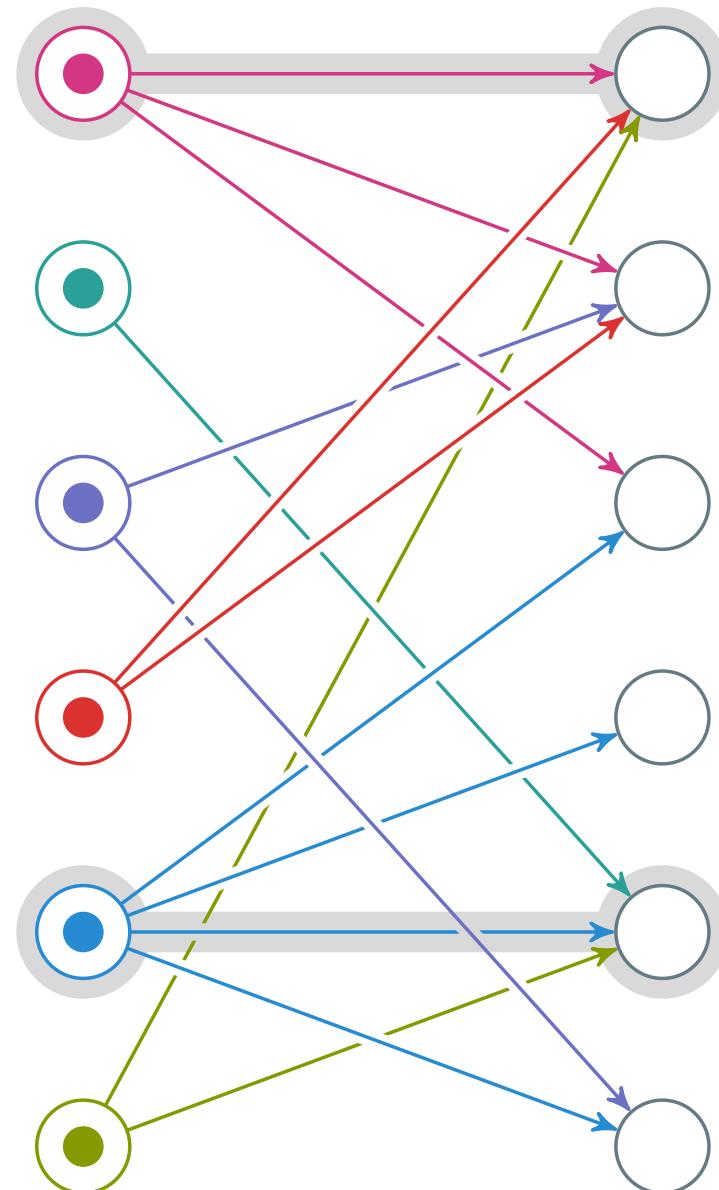
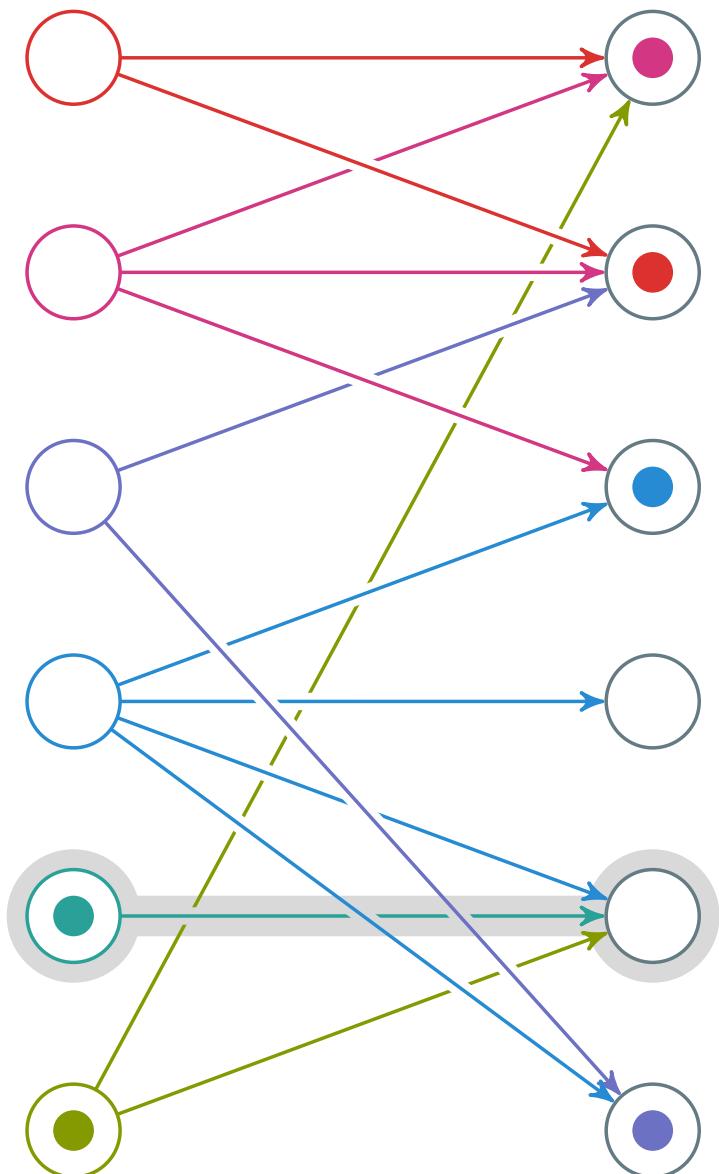


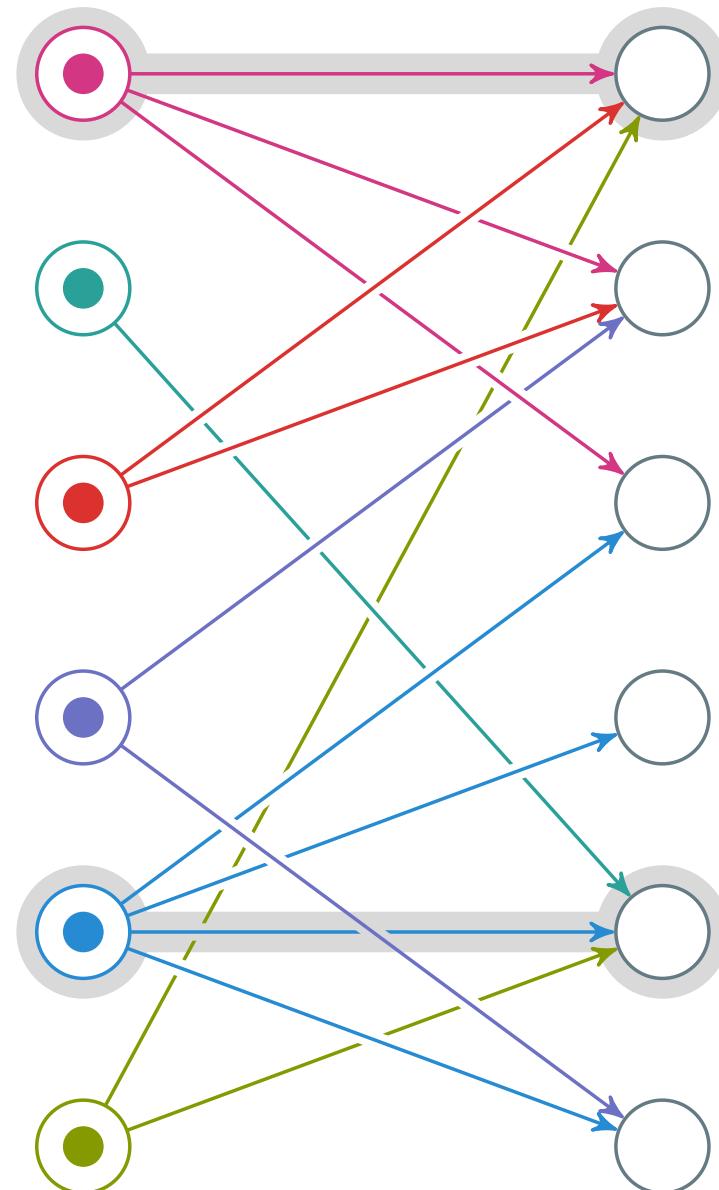
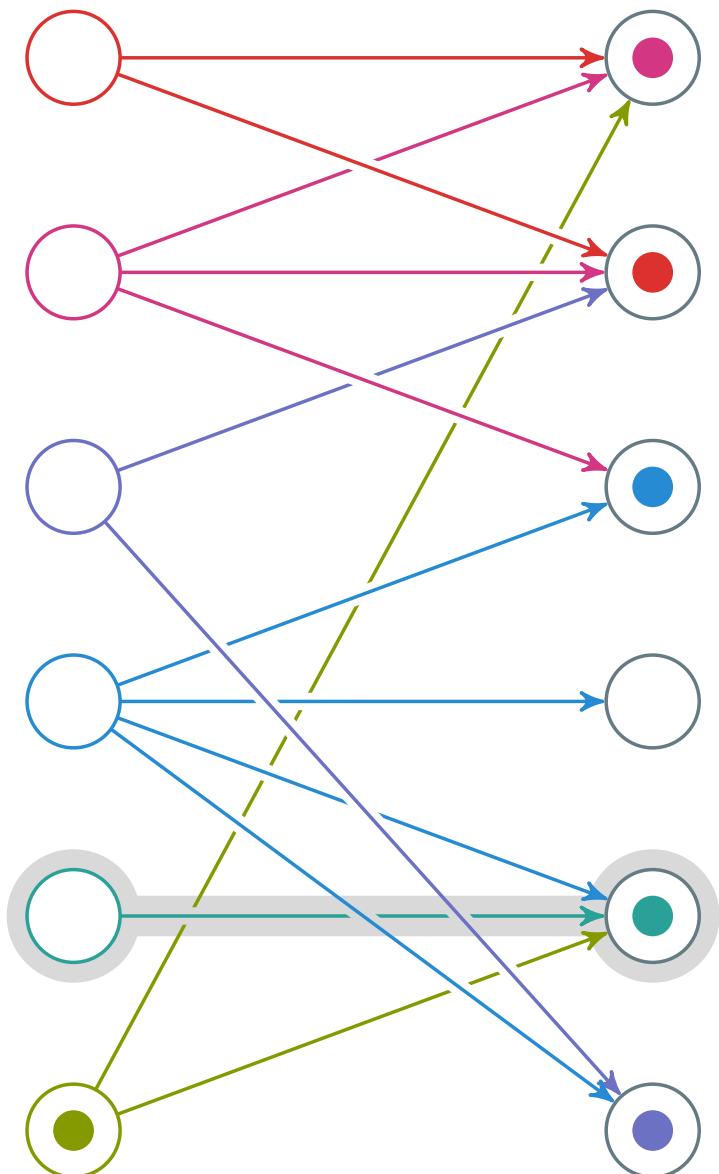


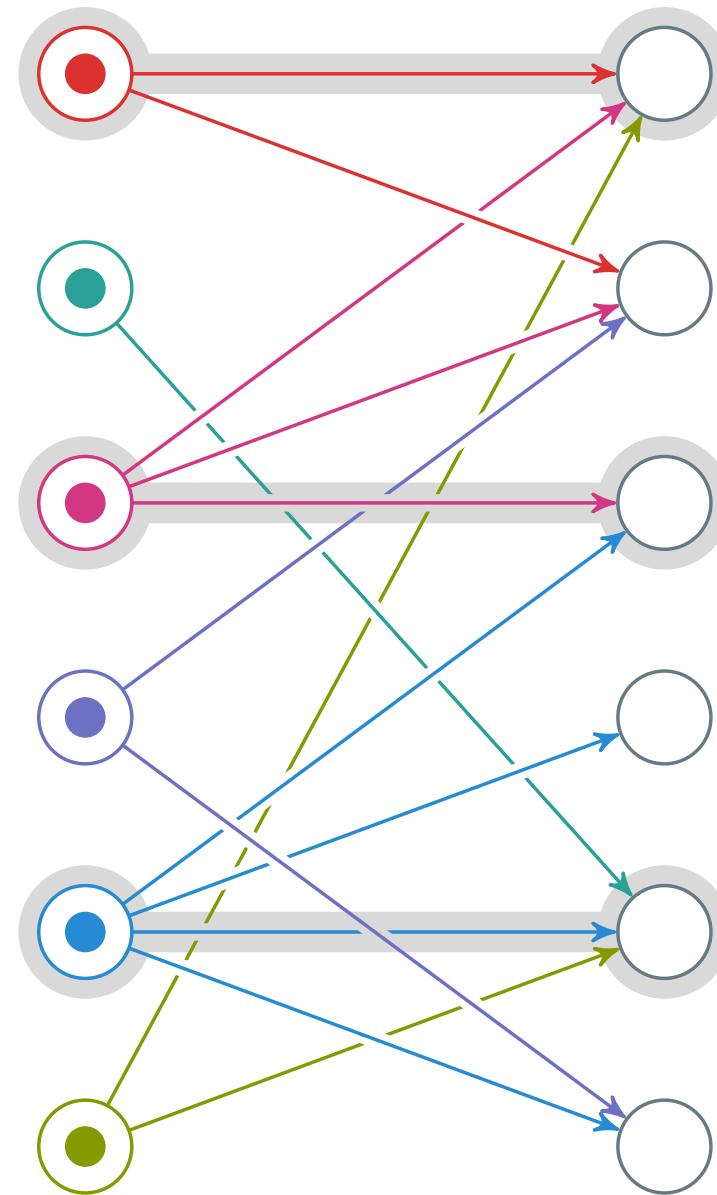
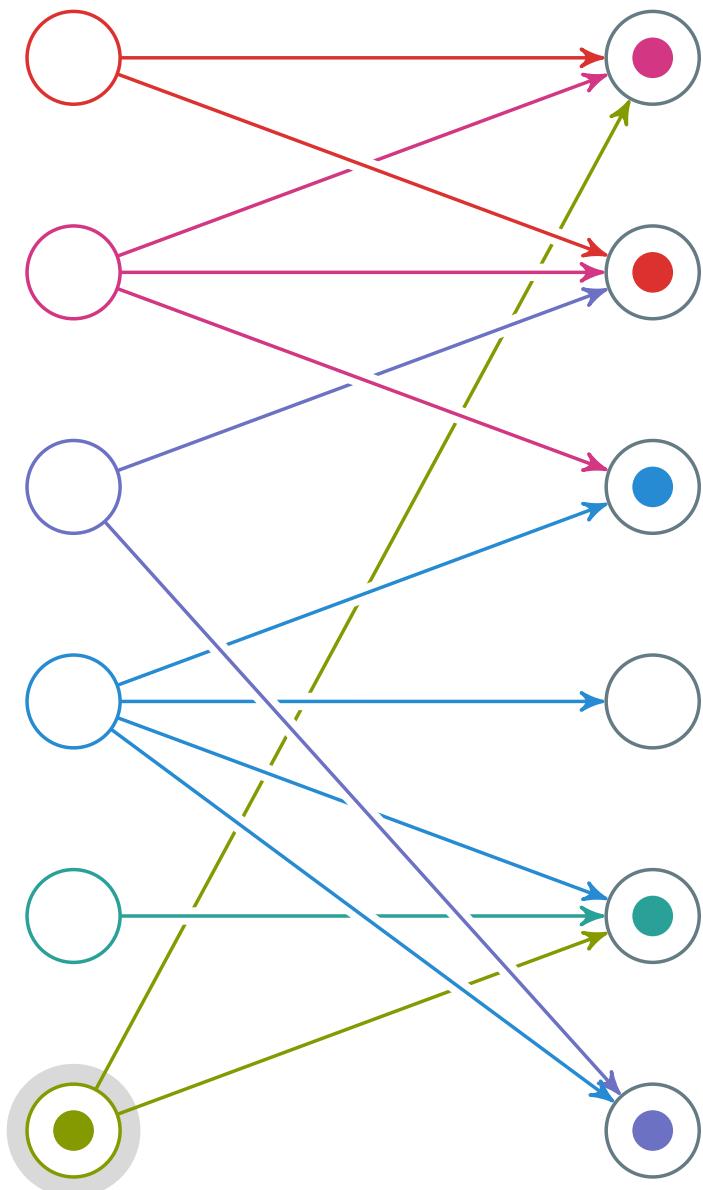


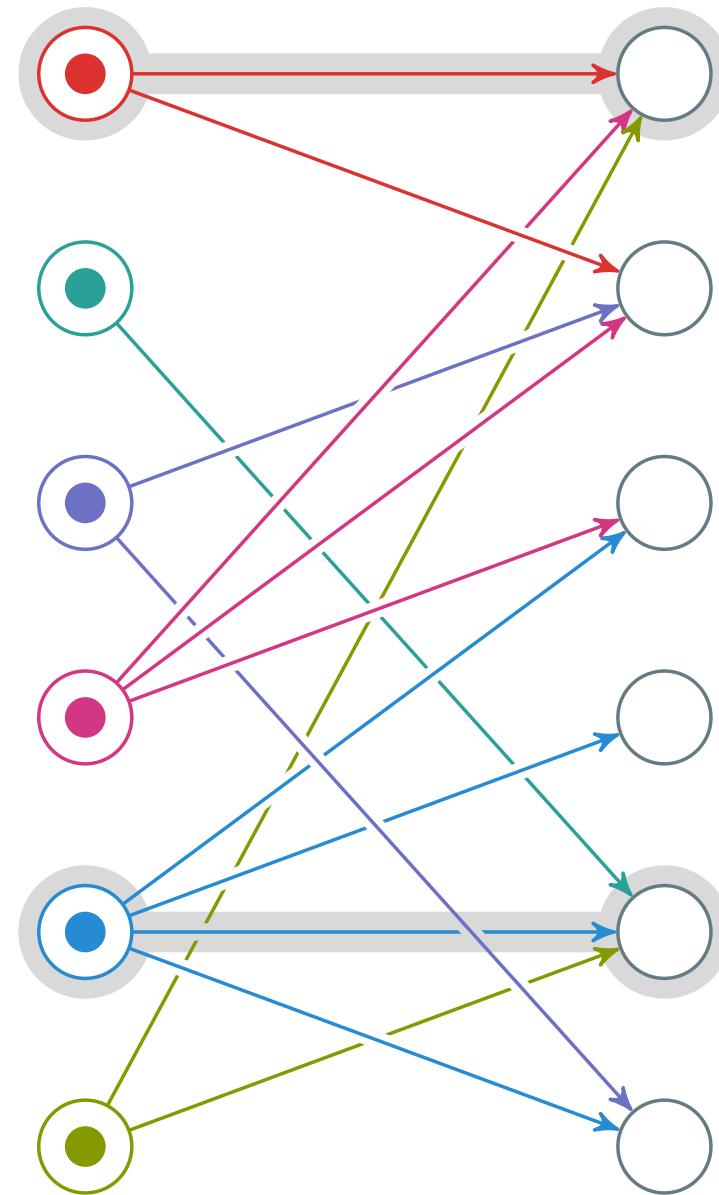
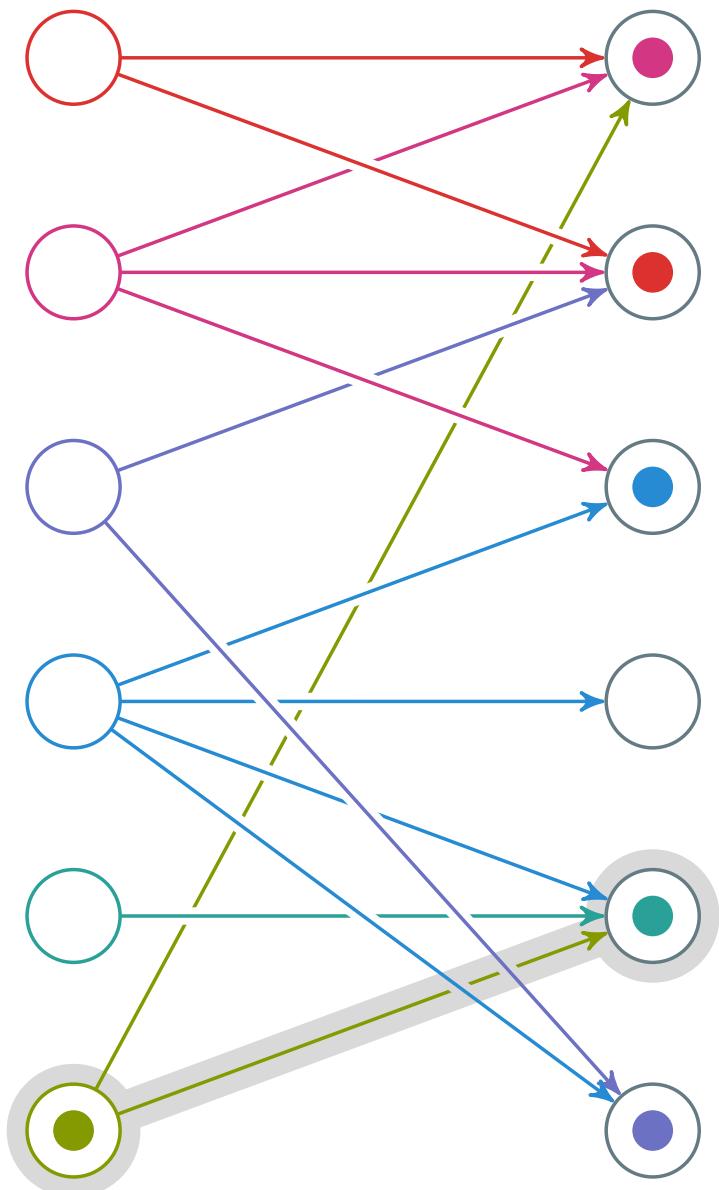


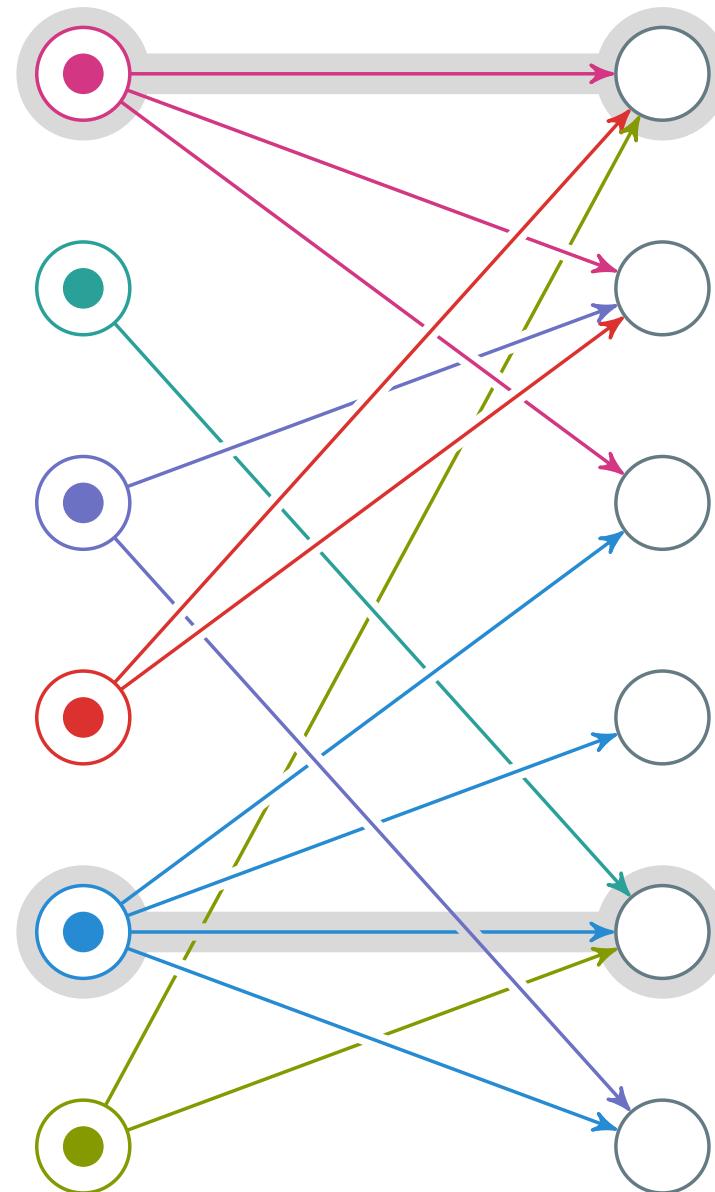
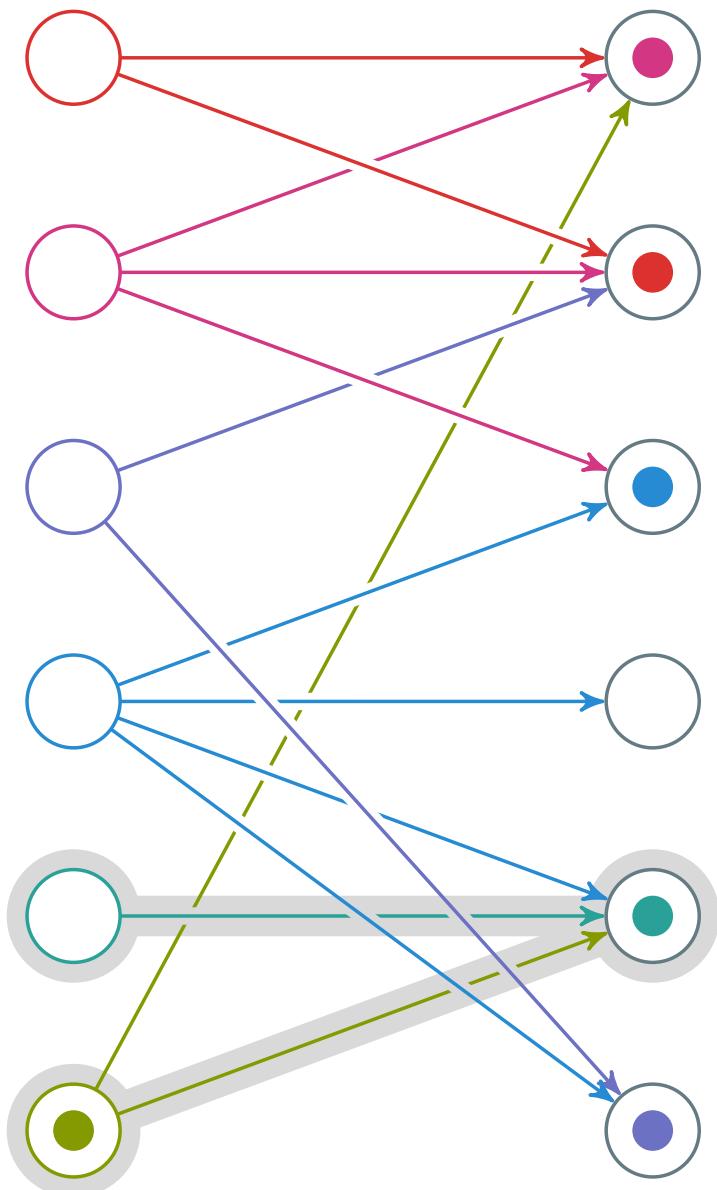


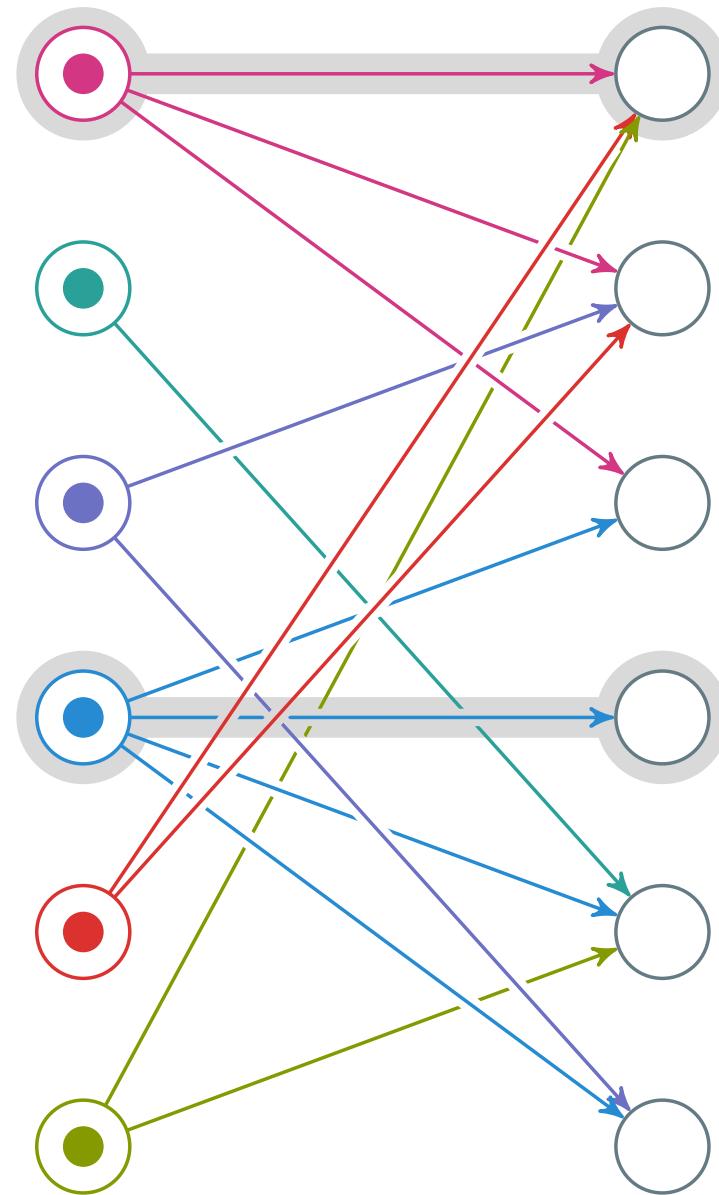
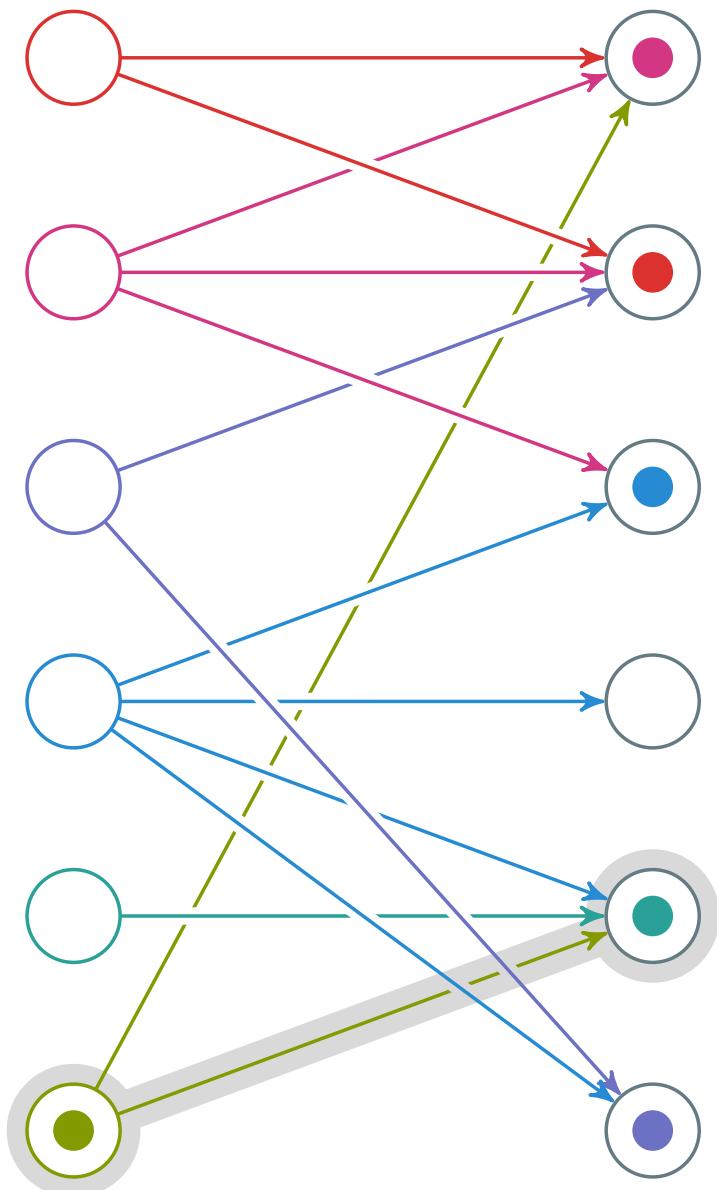


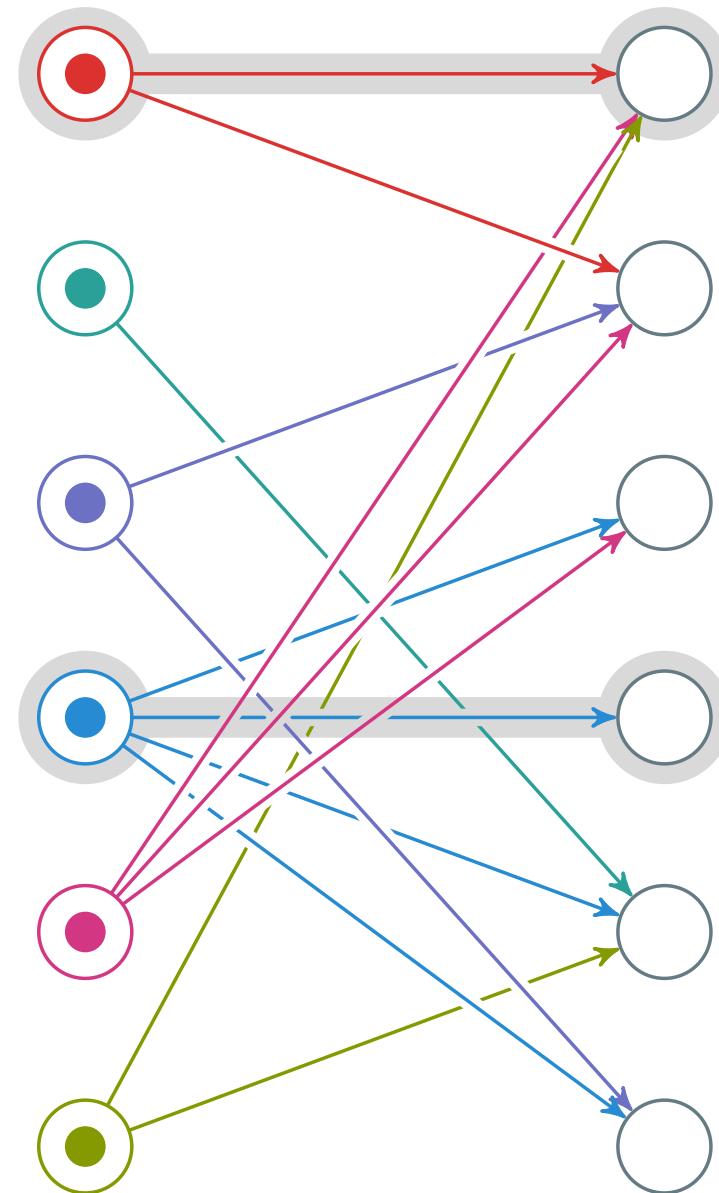
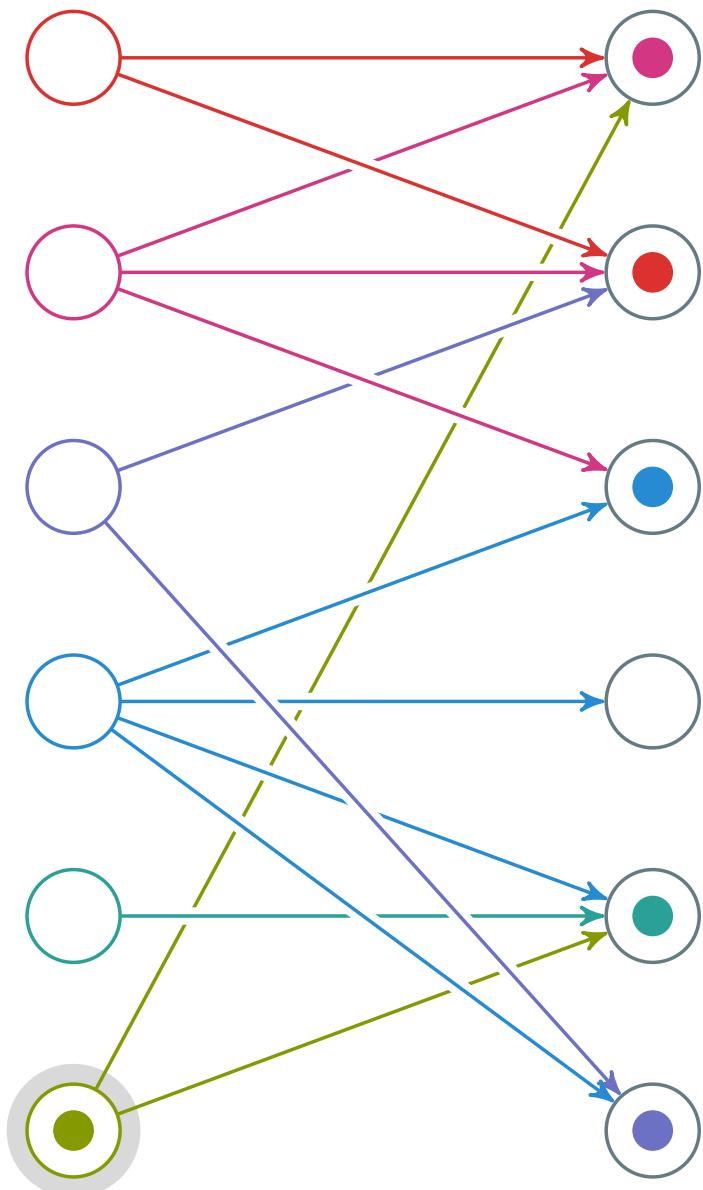


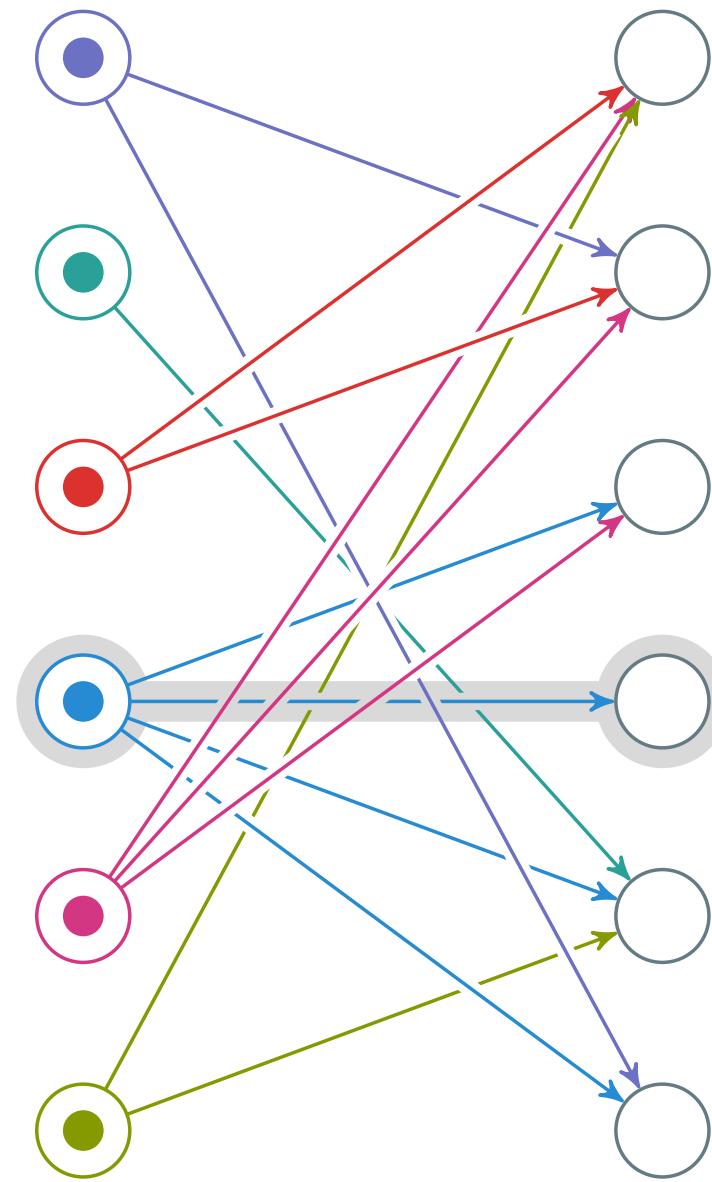
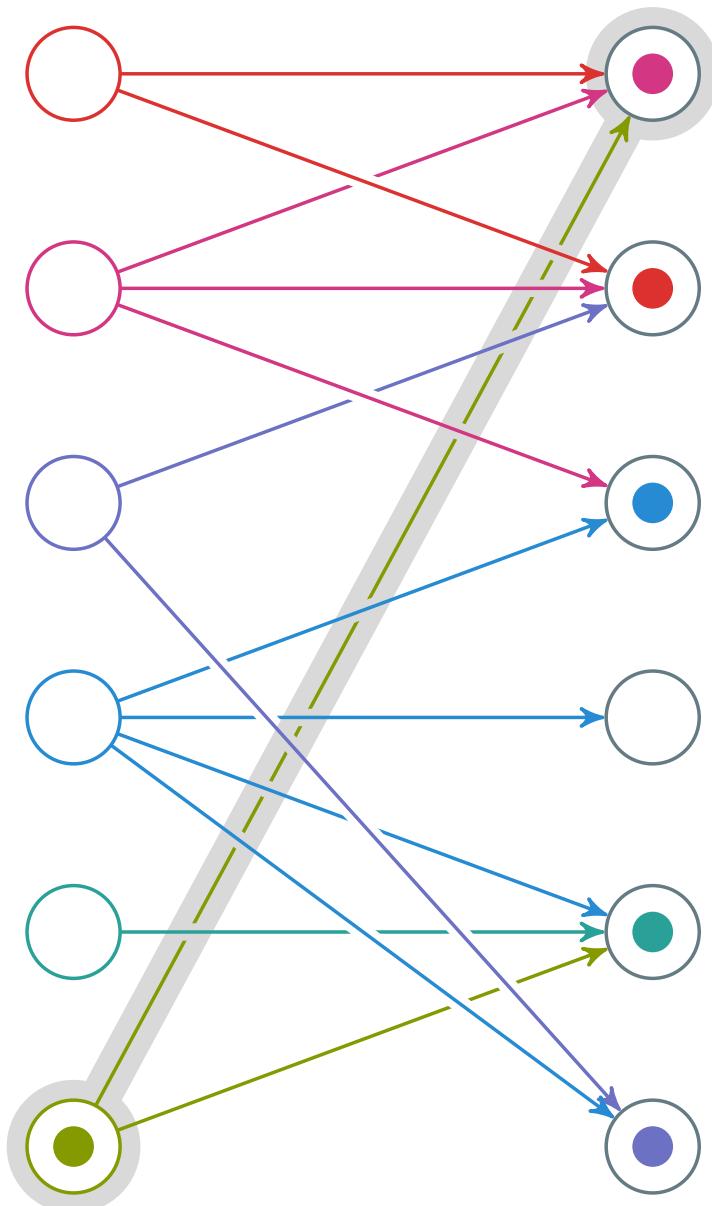


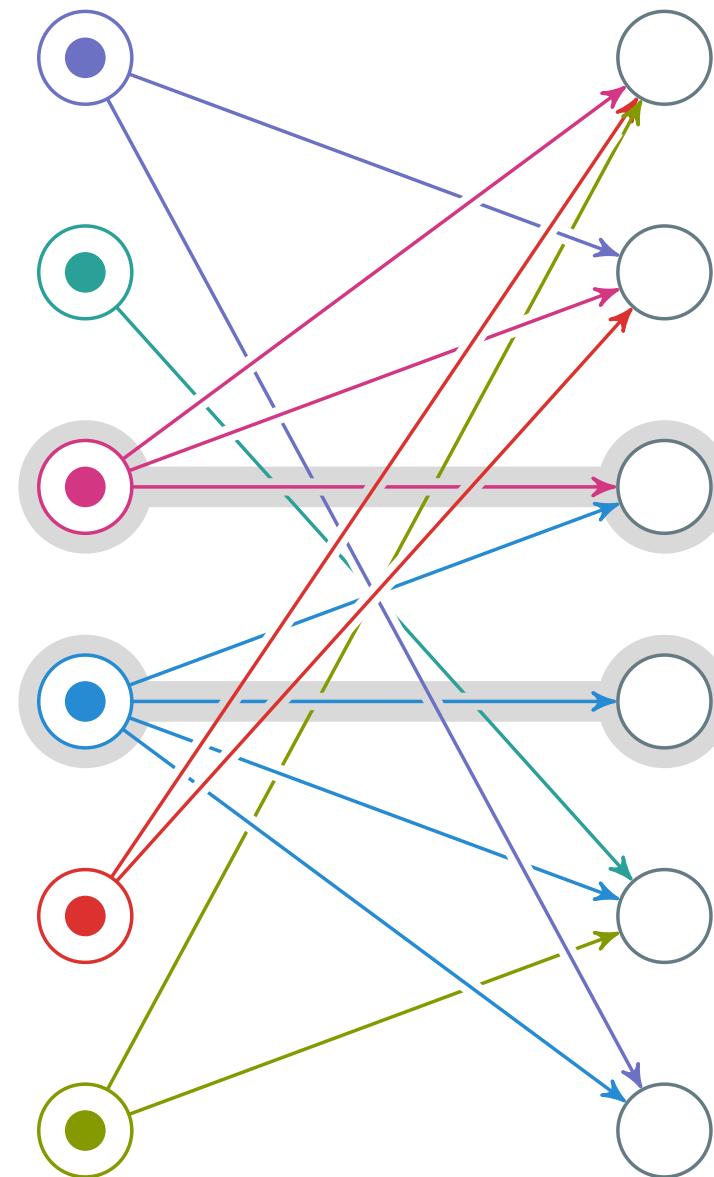
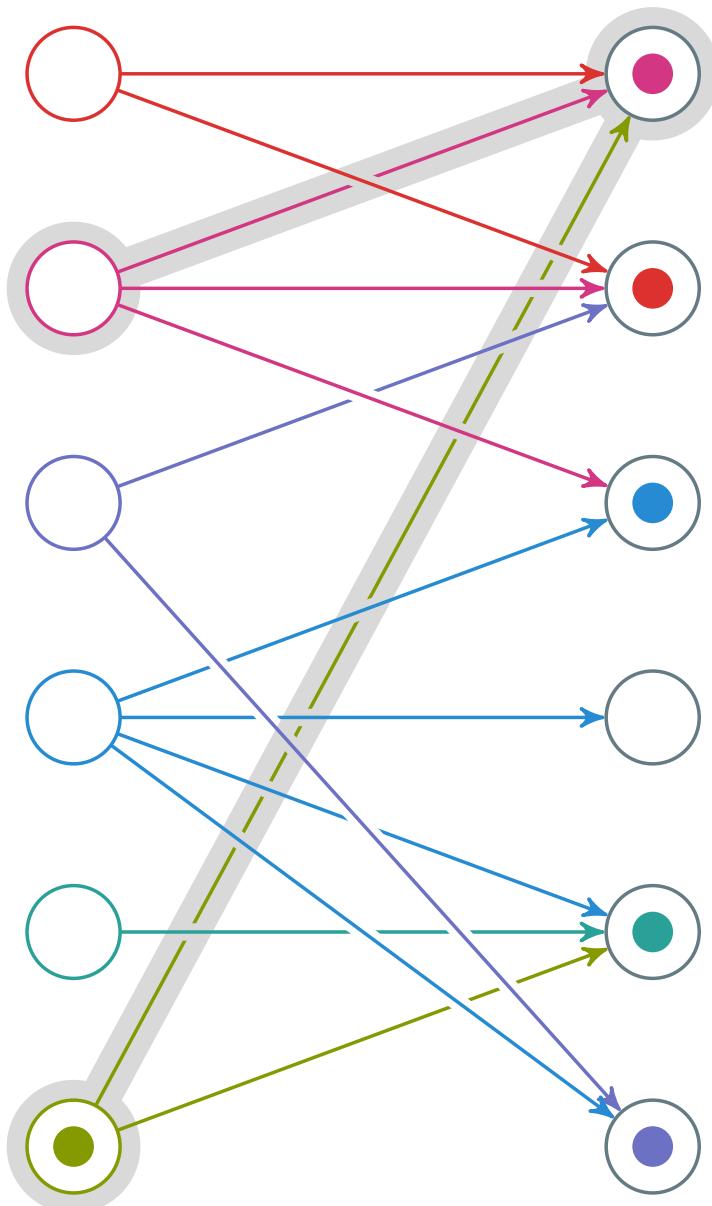


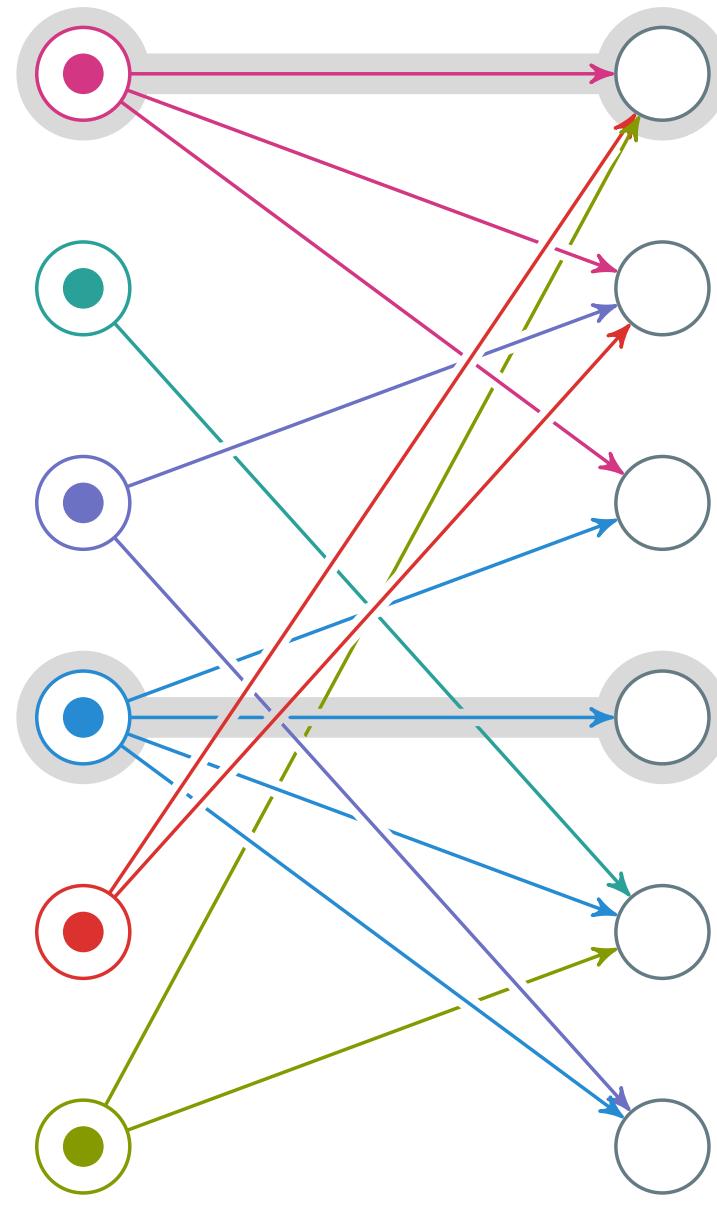
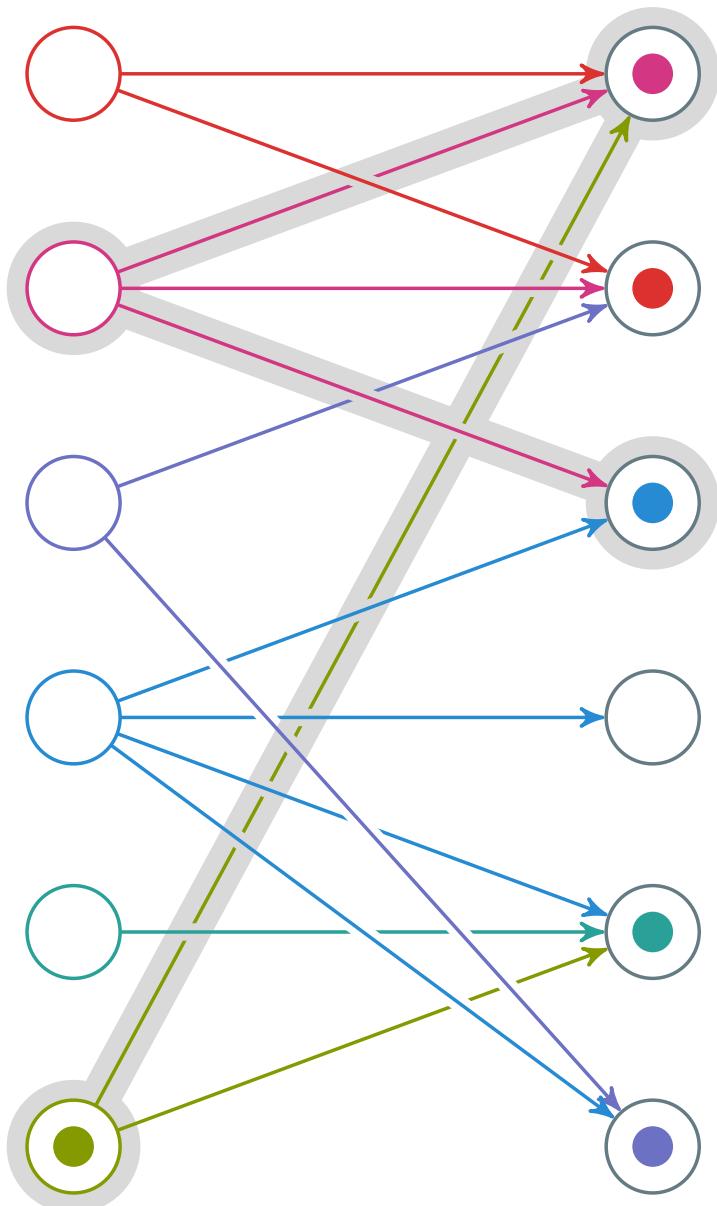


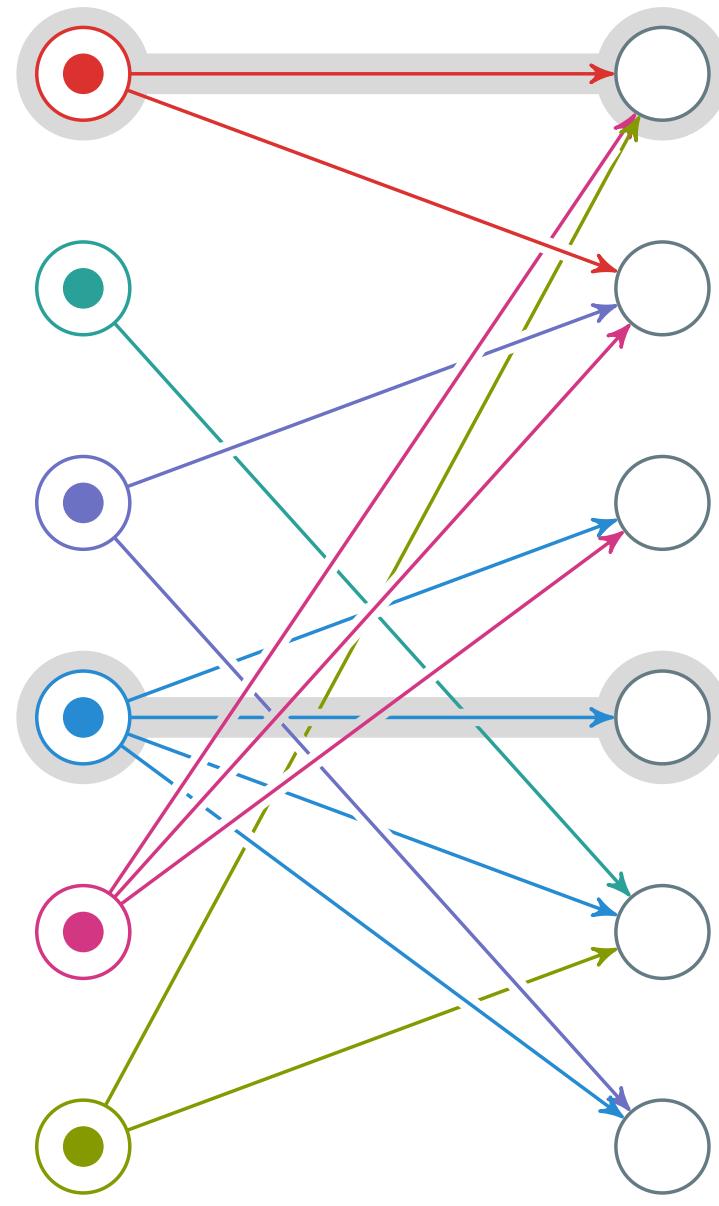
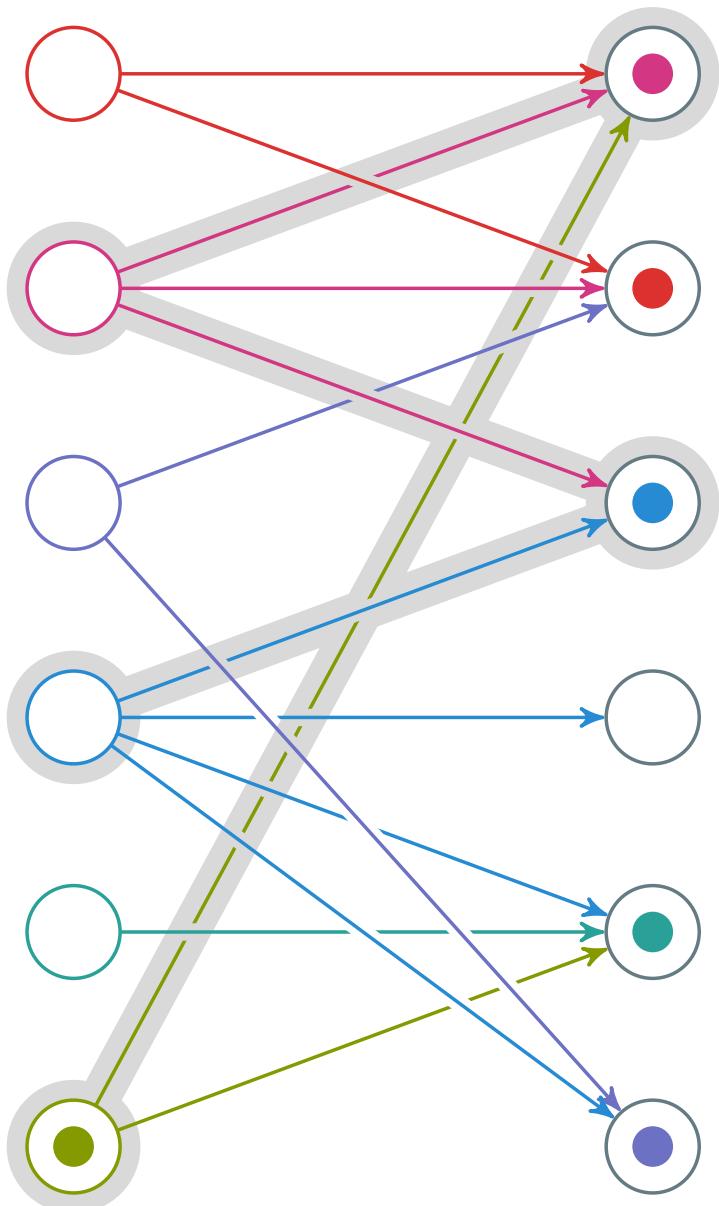


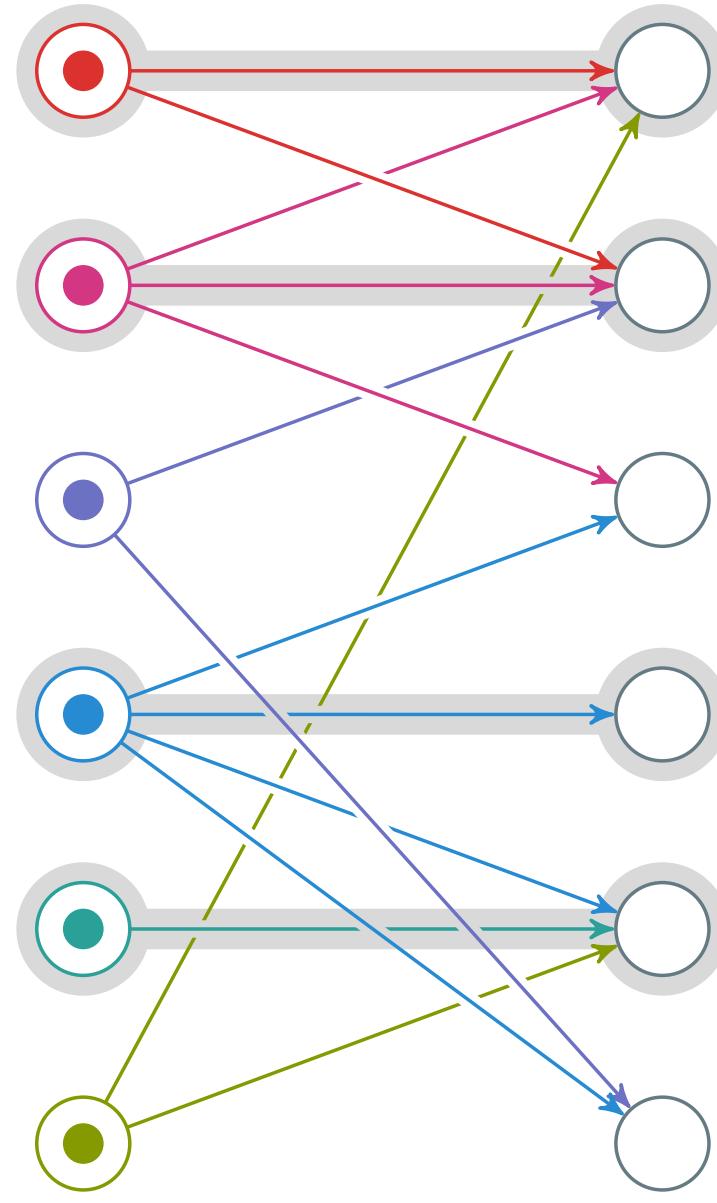
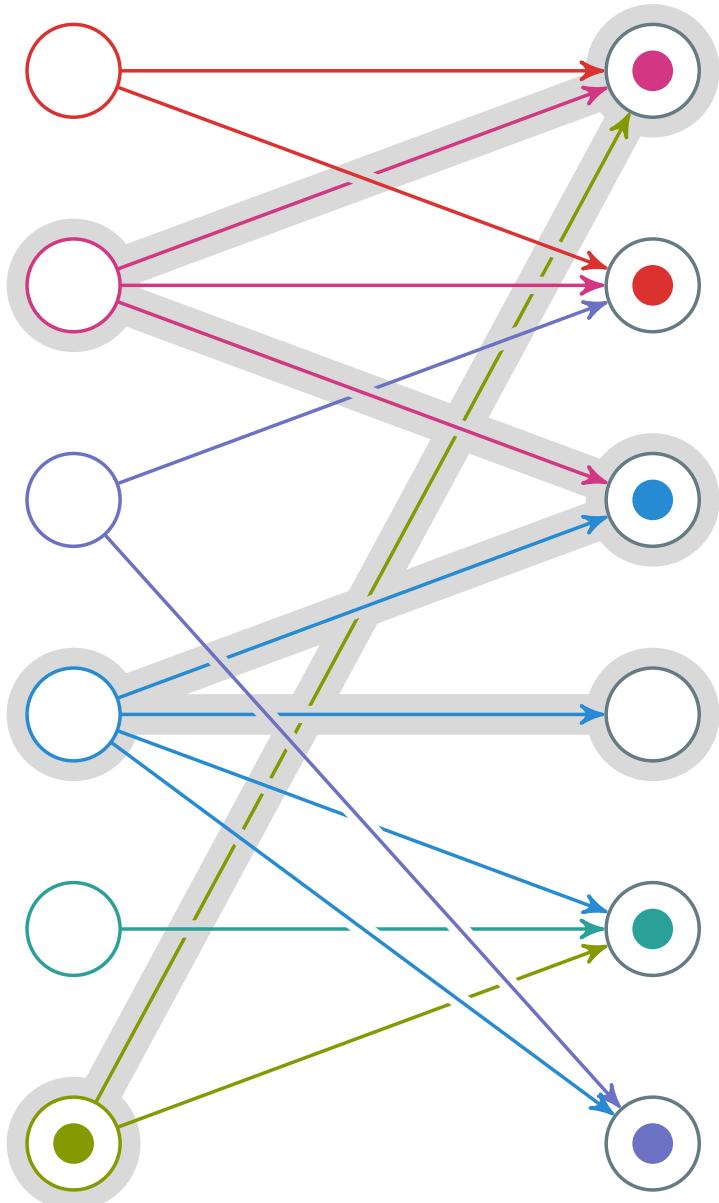


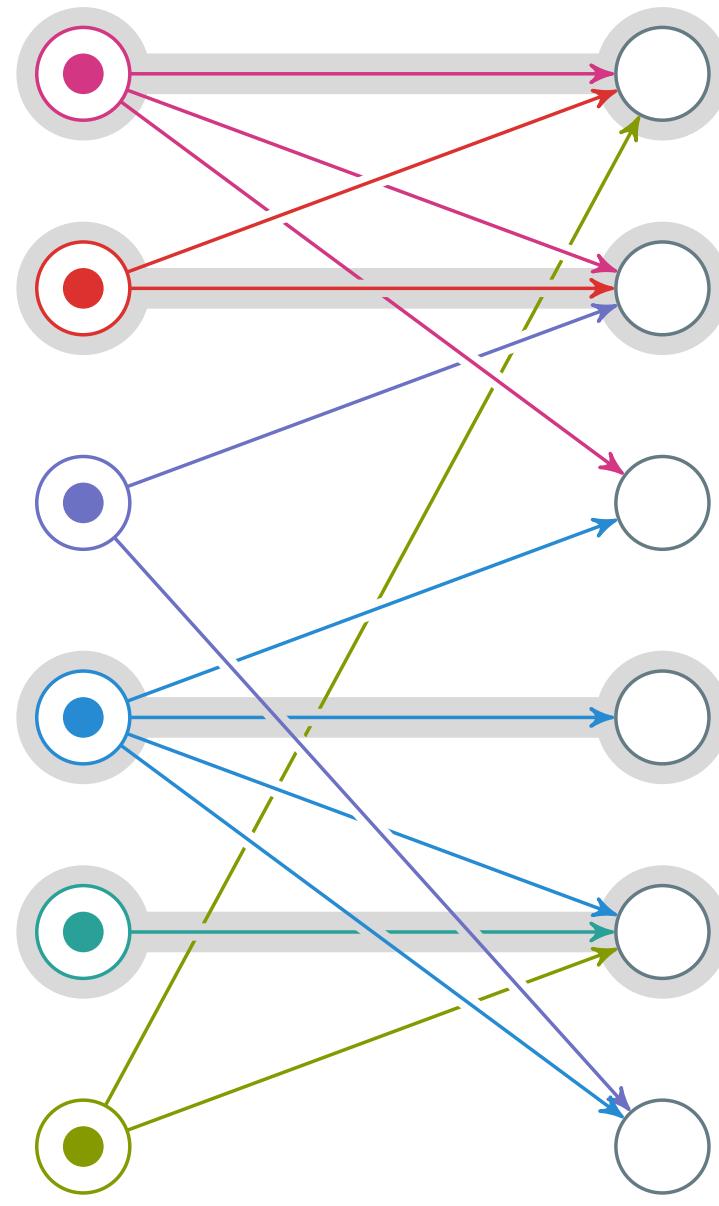
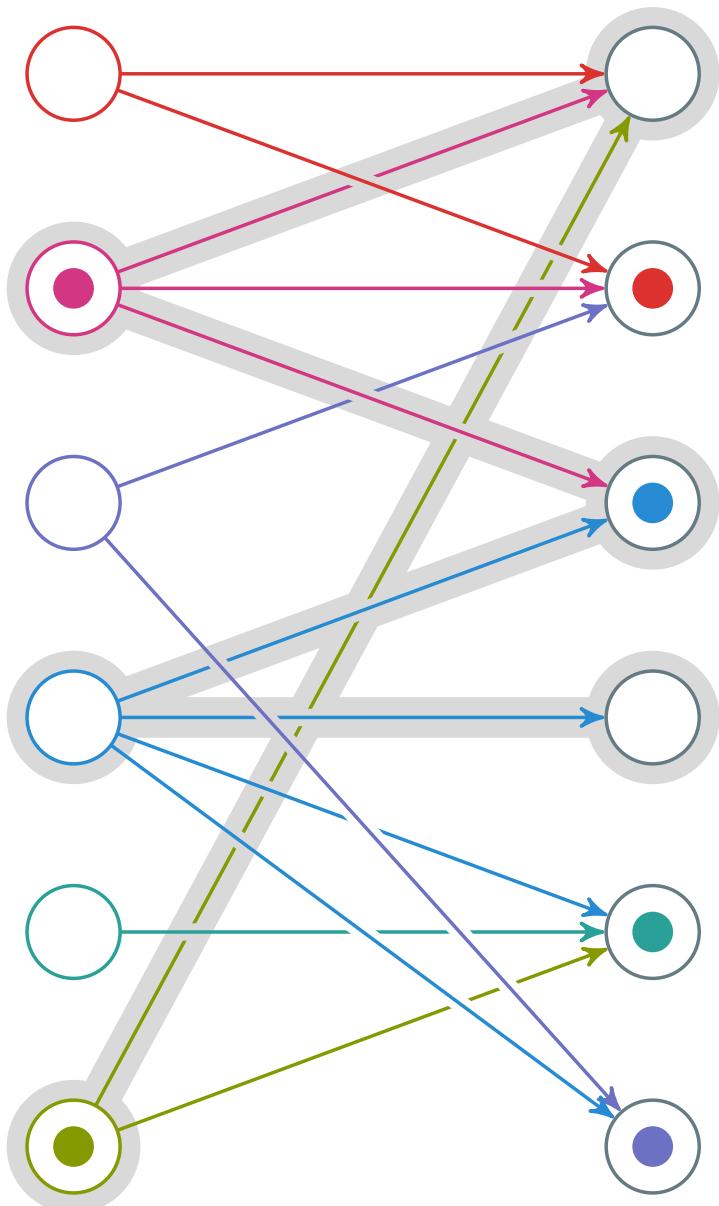


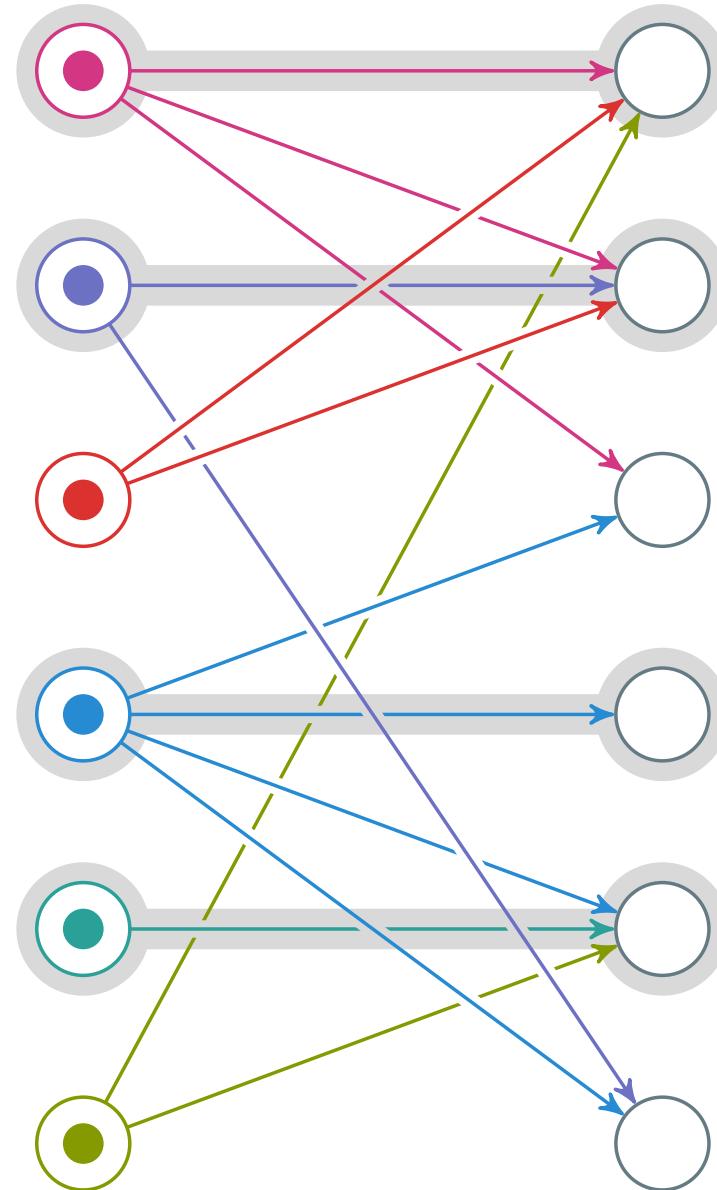
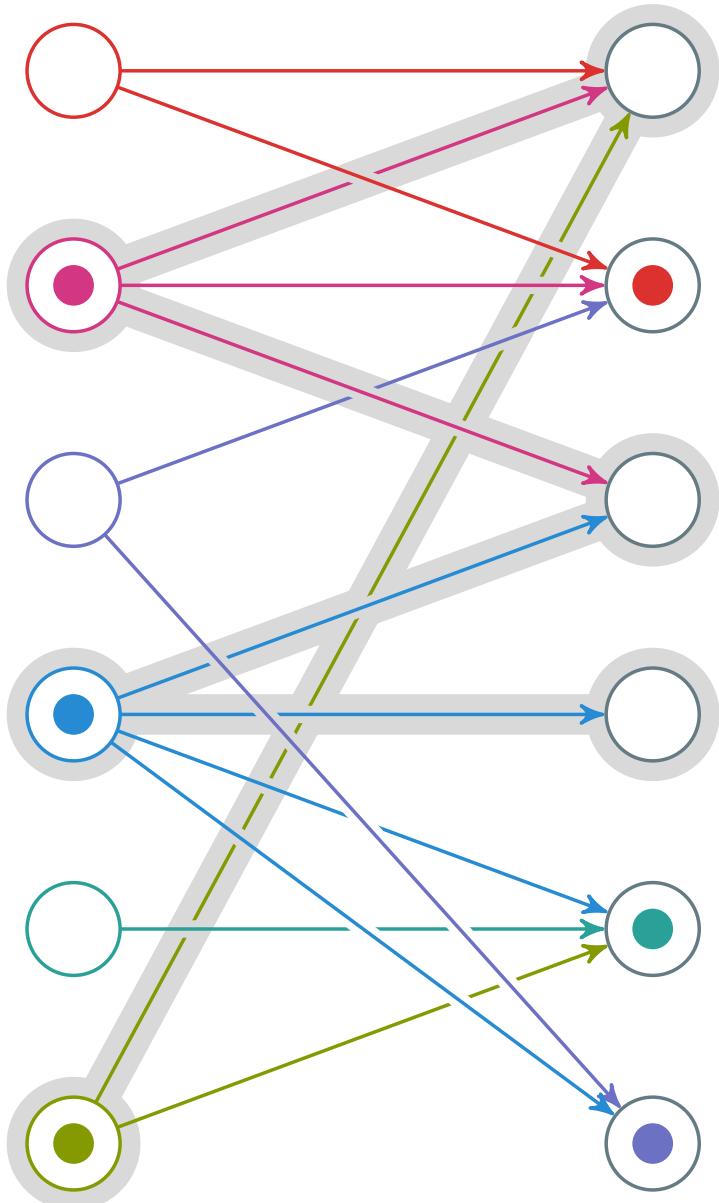


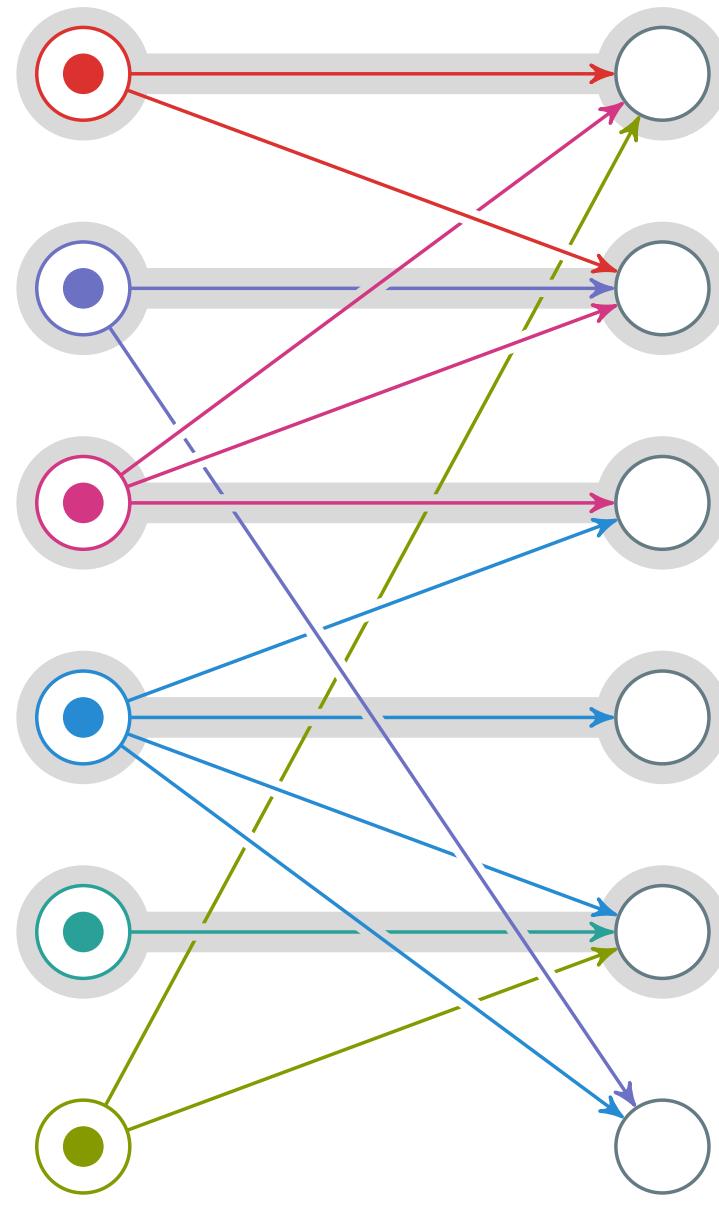
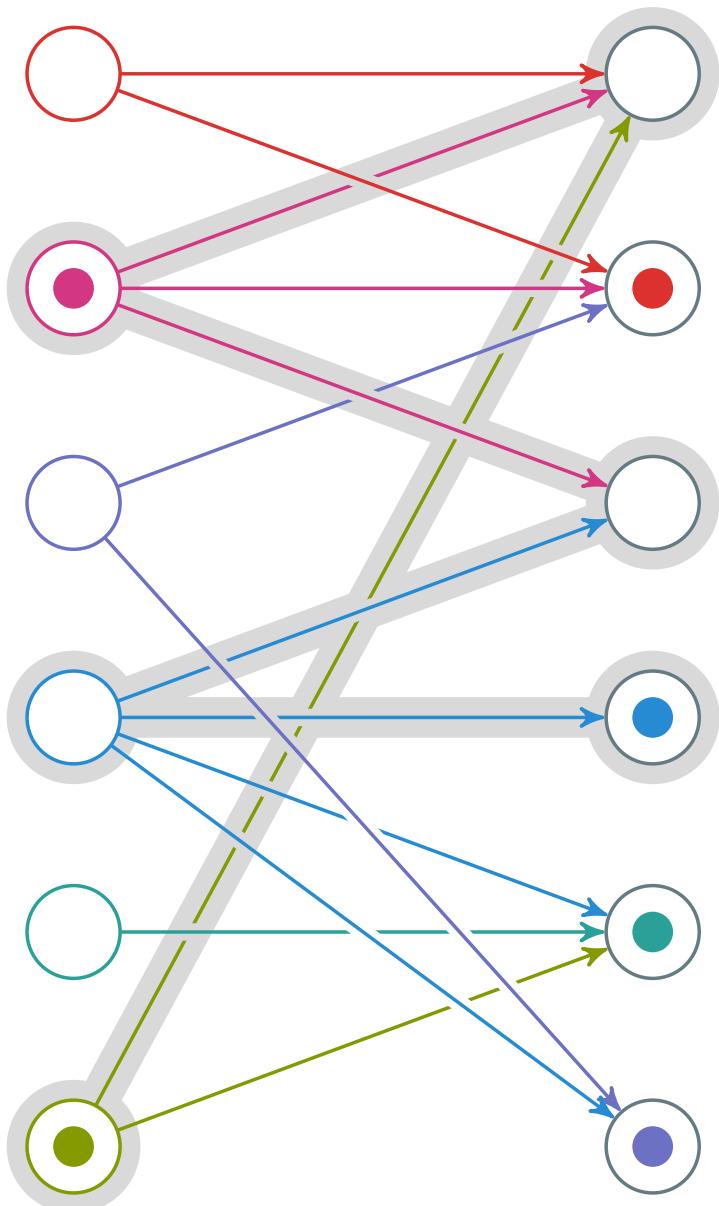


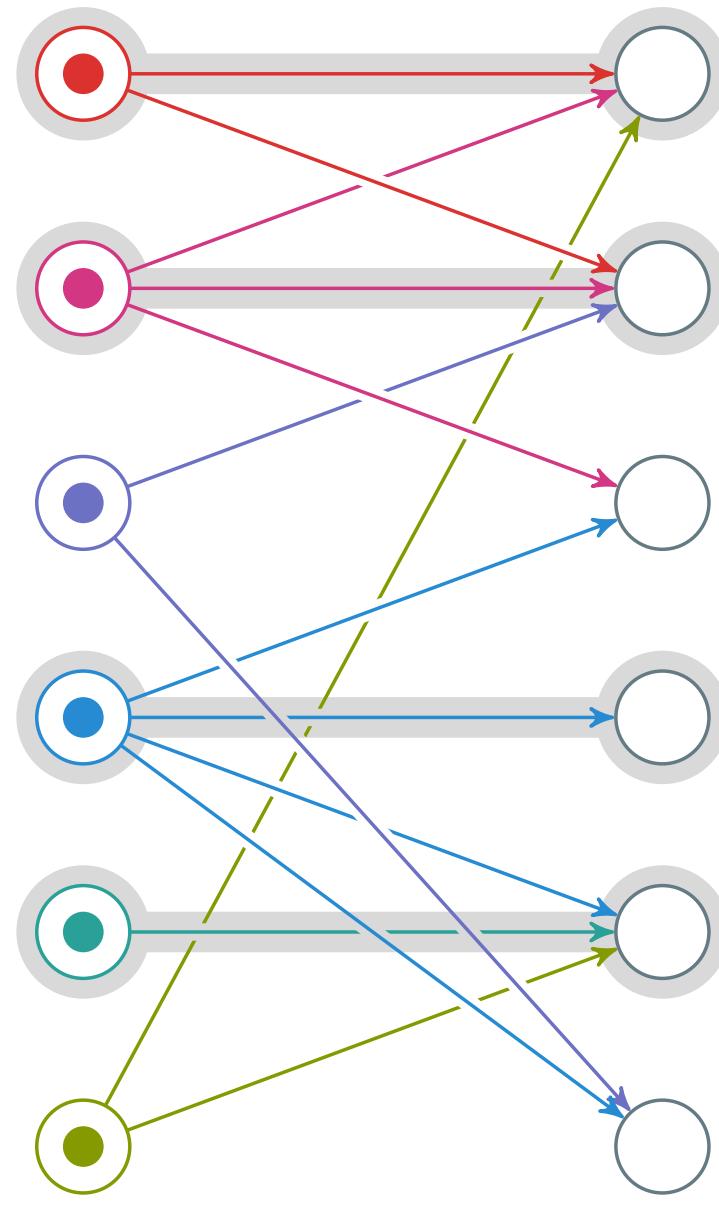
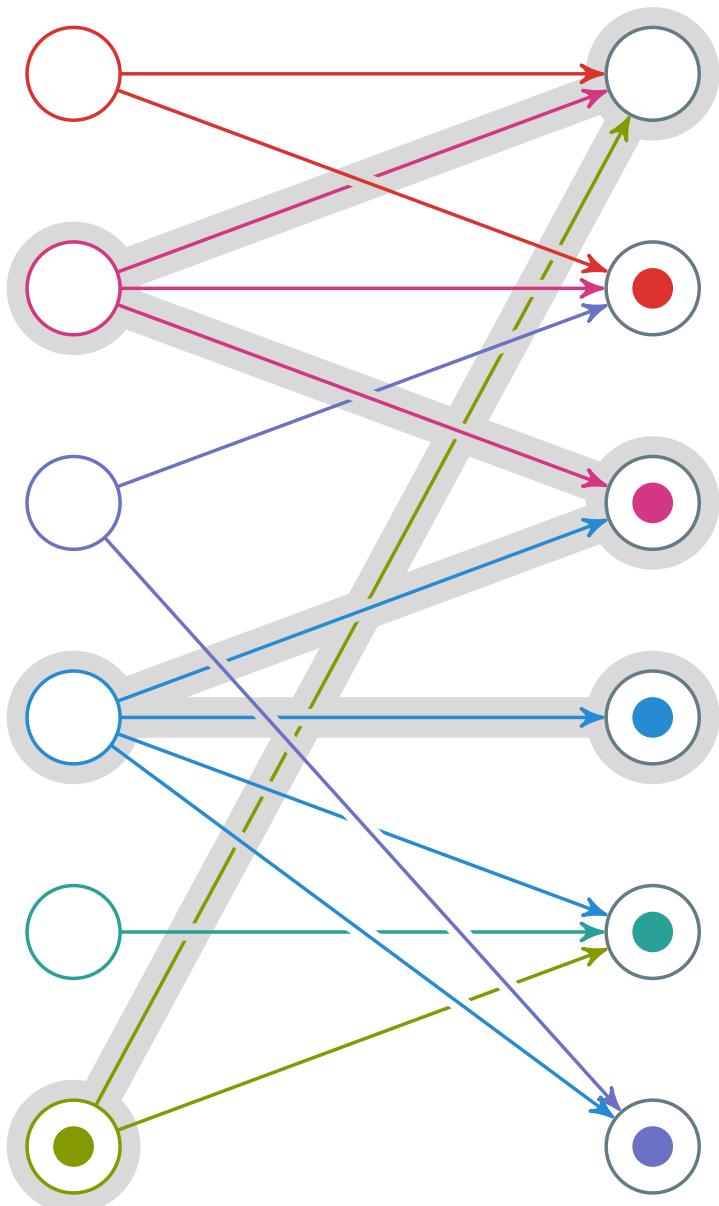


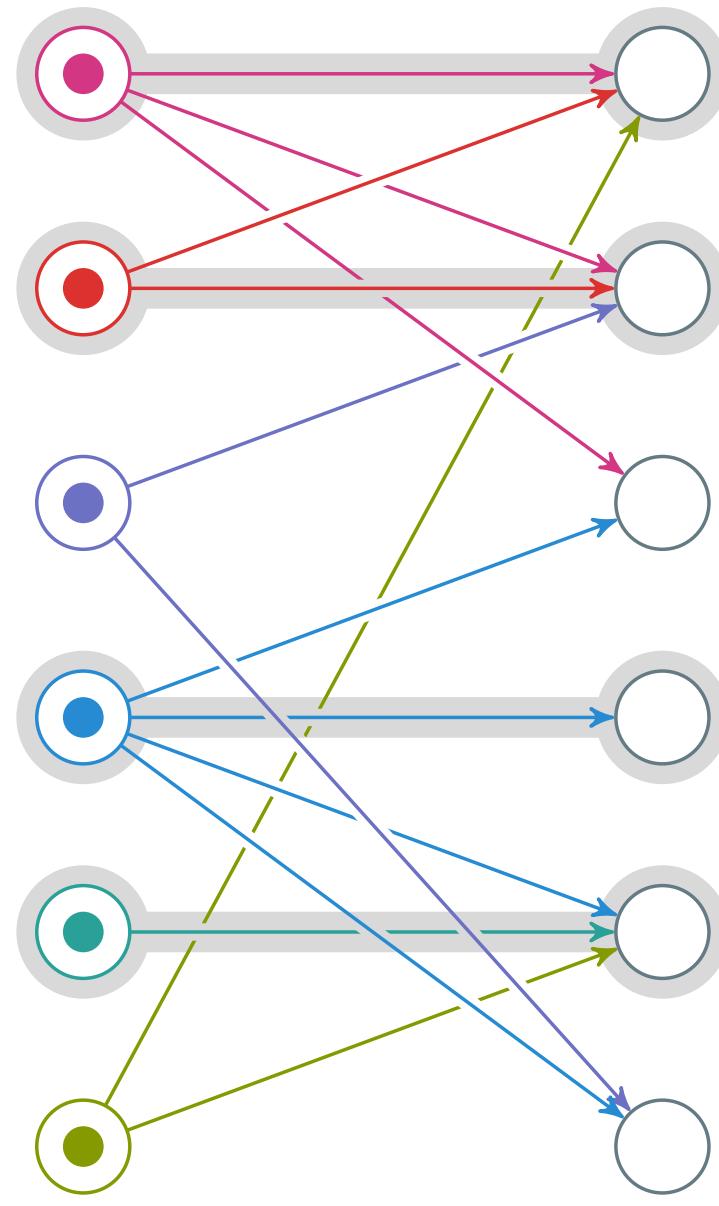
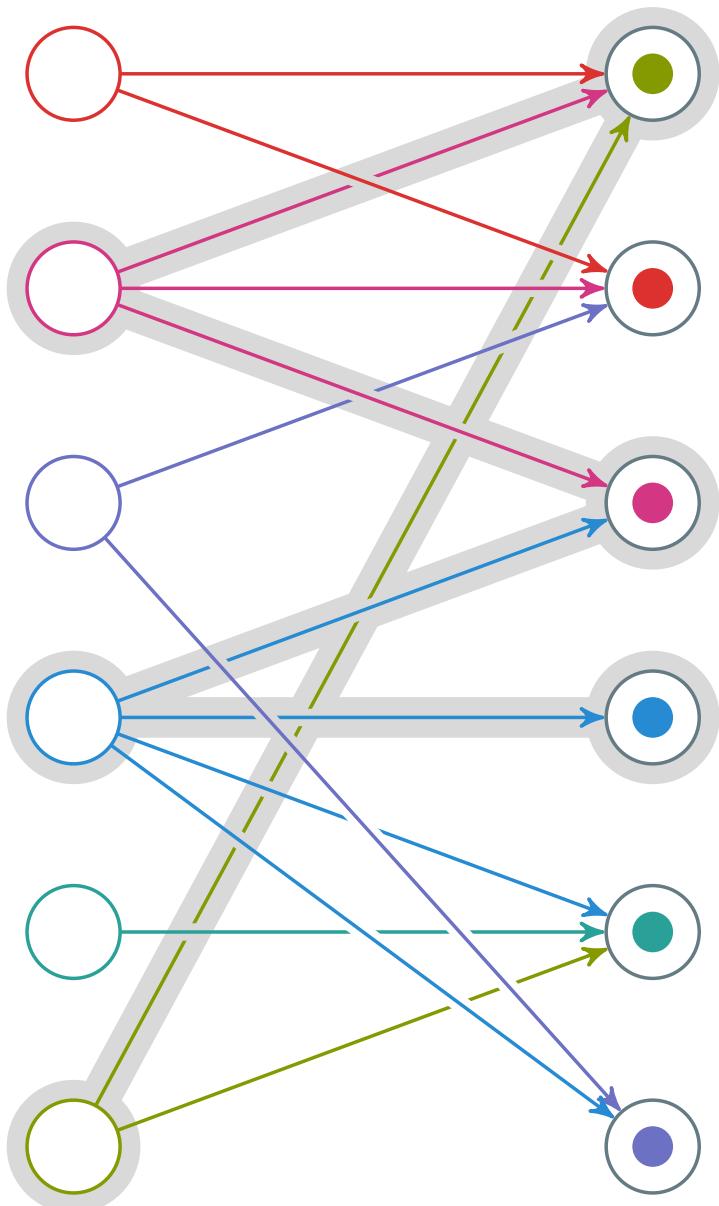


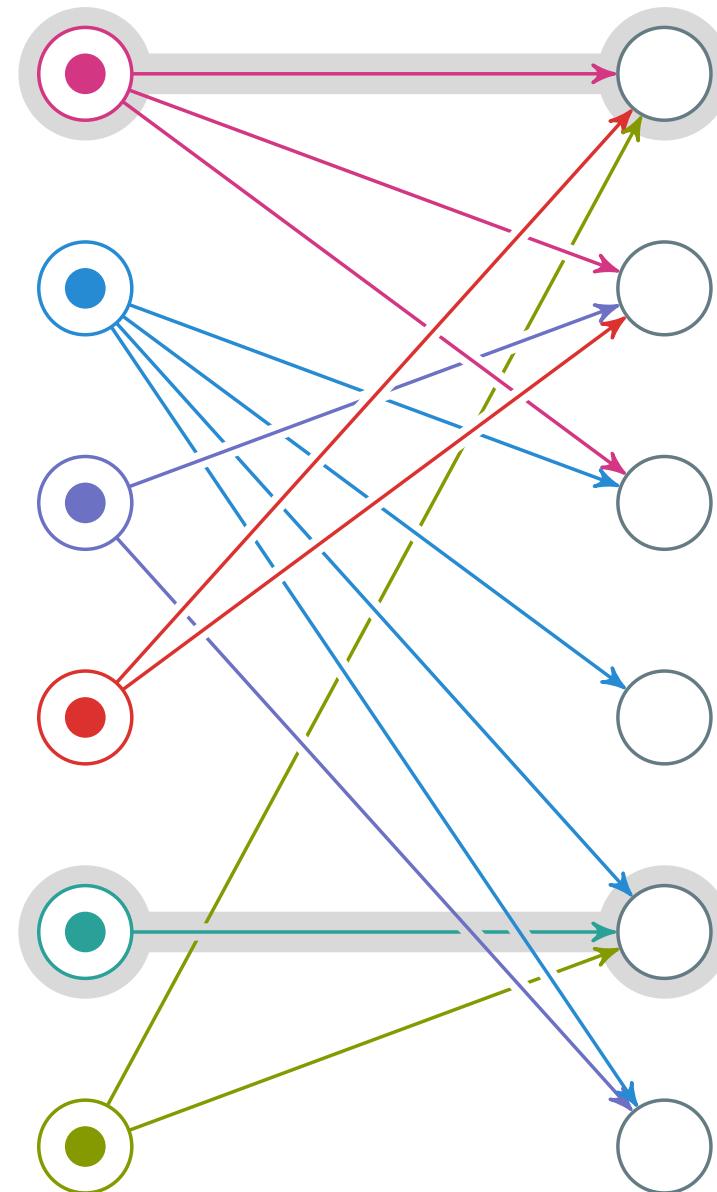
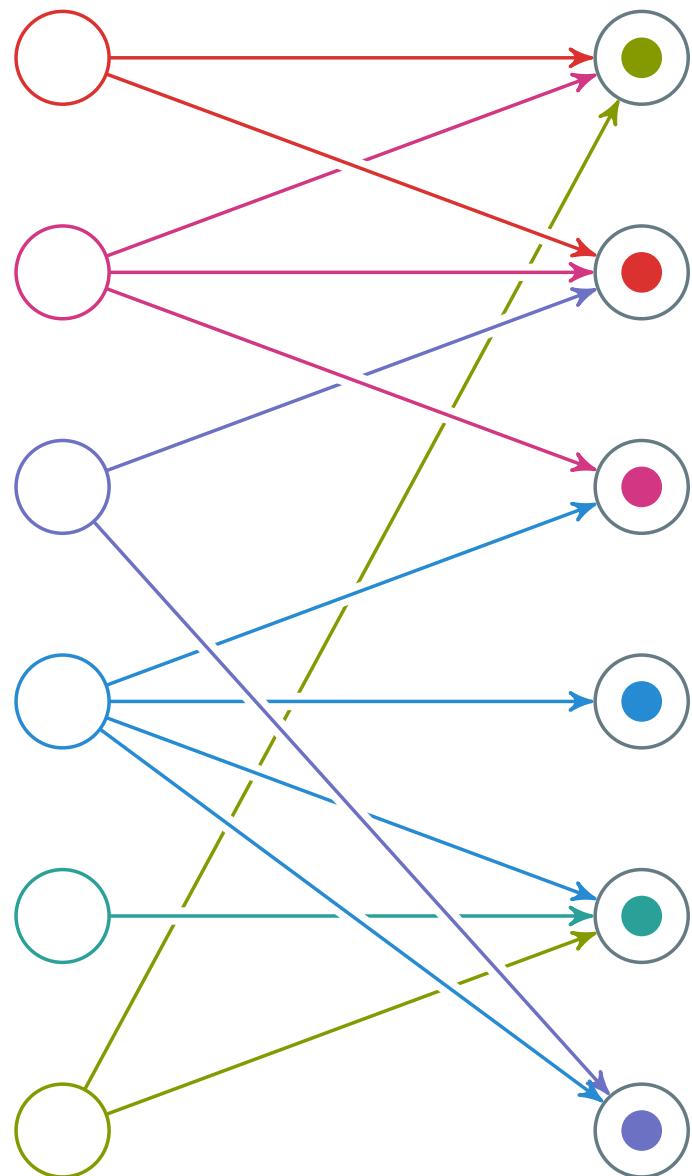


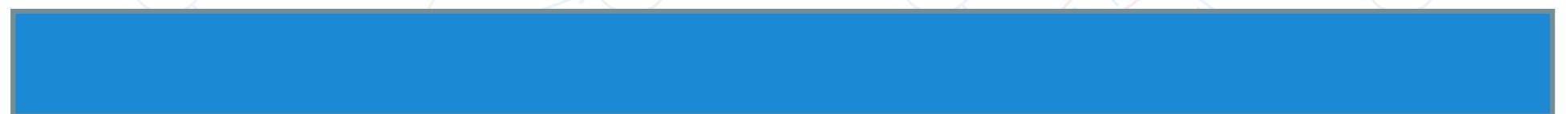


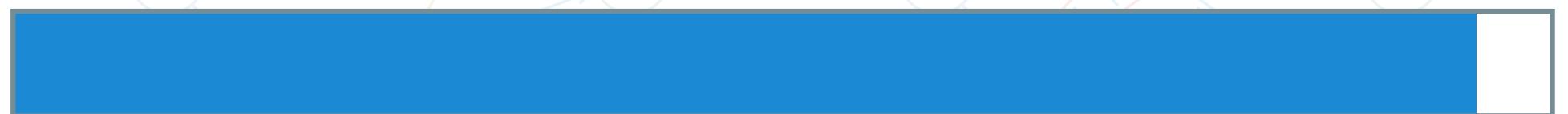


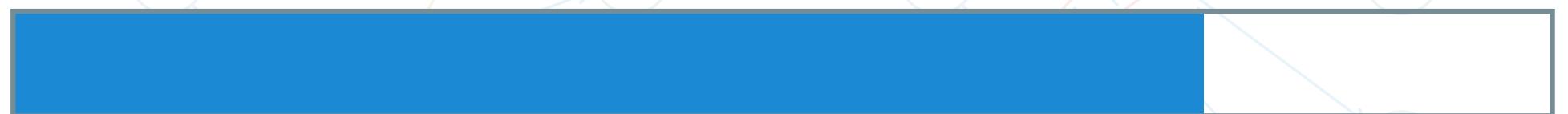






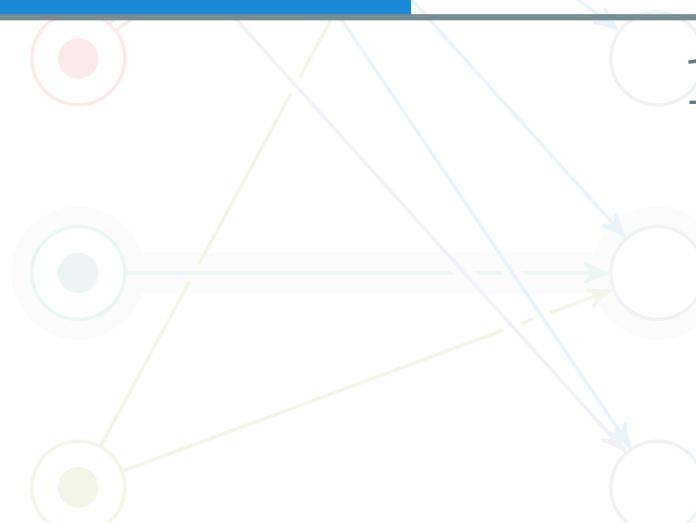
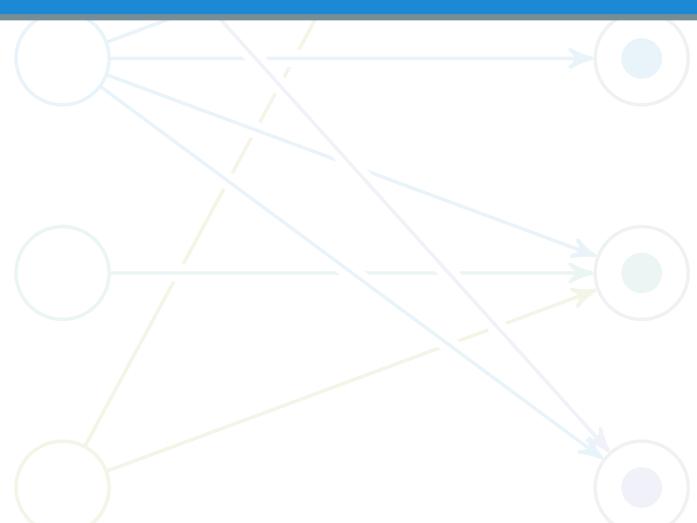






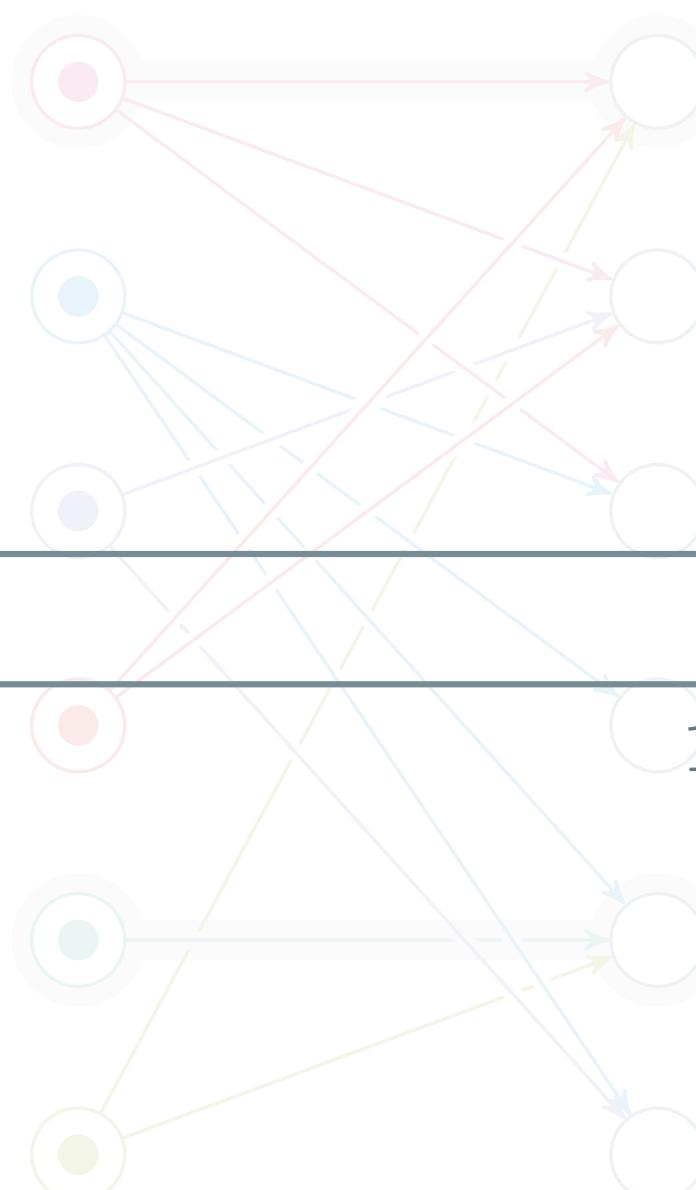
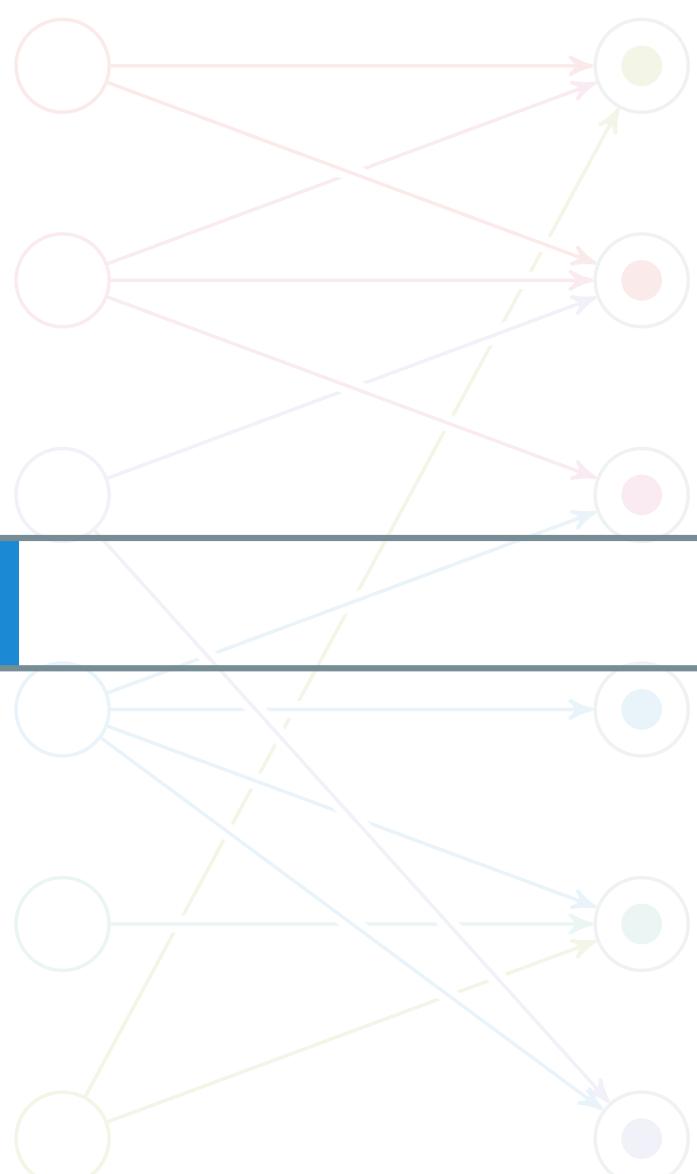
0

100%





0 100%





7.6%

Når sikk-sakk-algoritmen er ferdig har flipp-flopp algoritmen bare gjort seg ferdig med 7.6% av sin kjøretid.



Når måtte vi ha startet med flipp-flopp-algoritmen for å bli ferdige nå, for ulike antall donor-mottaker-par?

1

Antall donor-mottaker-par


$$\frac{d\tilde{E}}{dt} = \int_0^{2\pi} \frac{\partial V}{\partial t} d\theta$$
$$\tilde{E} = \int_0^{2\pi} \frac{e^{i\theta} V}{1 + e^{i\theta}} d\theta$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2\pi} \int_0^{2\pi} \frac{d}{dt} \left(\frac{e^{i\theta} V}{1 + e^{i\theta}} \right) d\theta$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2\pi} \int_0^{2\pi} \frac{e^{i\theta} V}{(1 + e^{i\theta})^2} d\theta$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2\pi} \int_0^{2\pi} \frac{e^{i\theta} V}{1 + 2e^{i\theta} + e^{2i\theta}} d\theta$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2\pi} \int_0^{2\pi} \frac{e^{i\theta} V}{1 + 2e^{i\theta} + e^{2i\theta}} d\theta$$

$-1 \mu\text{s}$

Når måtte vi ha startet med å sjekke alle muligheter, dersom det å sjekke én mulighet tok 1 mikrosekund?

2

$$\frac{d\tilde{E}}{dt} = \frac{\partial \tilde{V}}{\partial t}$$
$$\tilde{E} = \frac{1}{2} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \tilde{V} dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \frac{\partial \tilde{V}}{\partial t} dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \frac{1}{2} \frac{\partial^2 \tilde{V}}{\partial t^2} dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{4} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \frac{\partial^2 \tilde{V}}{\partial t^2} dt$$

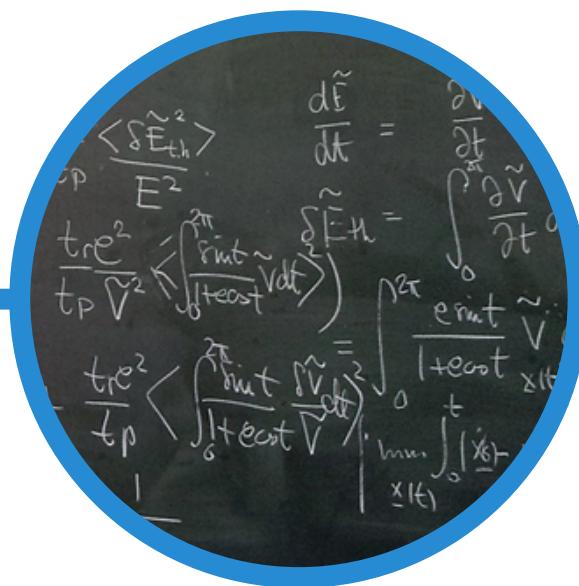
$-2 \mu\text{s}$

3

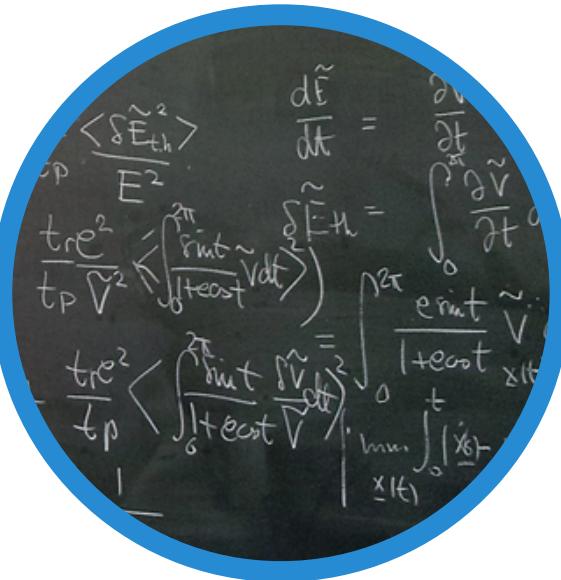
$$\frac{d\tilde{E}}{dt} = \frac{\partial \tilde{V}}{\partial t}$$
$$\tilde{E} = \frac{1}{2} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} V^2 dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \frac{d}{dt} \left(\frac{V^2}{1+e^{i\omega t}} \right) dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \frac{2V}{1+e^{i\omega t}} \frac{dV}{dt} dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \frac{2V^2}{1+e^{i\omega t}} dt$$

$-6 \mu\text{s}$

4


$$\frac{d\tilde{E}}{dt} = \frac{\partial v}{\partial t} + \frac{\partial^2 v}{\partial t^2}$$
$$\tilde{E}_{th} = \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} x(t) dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2\pi} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \frac{d}{dt} x(t) dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2\pi} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \frac{d}{dt} \left(\frac{1}{2} \ln(1+e^{i\omega t}) \right) dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2\pi} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \frac{1}{1+e^{i\omega t}} \omega dt$$

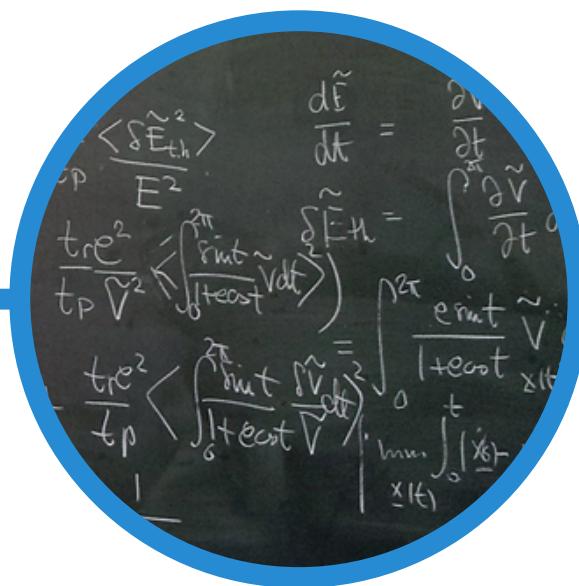
-24 μ s


$$\frac{d\tilde{E}}{dt} = \frac{\partial \tilde{V}}{\partial t}$$
$$\tilde{E} = \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \tilde{V} dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{i\omega} \left(\tilde{V} - \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \frac{\partial \tilde{V}}{\partial t} dt \right)$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{i\omega} \left(\tilde{V} - \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \tilde{V} dt \right)$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{i\omega} \tilde{V}$$

$-120 \mu\text{s}$

$$\begin{aligned}
 \frac{\langle \delta \tilde{E}_{th}^2 \rangle}{E^2} &= \frac{d\tilde{E}}{dt} = \frac{\partial V}{\partial t} \\
 \frac{t \epsilon^2}{t_p V^2} \left\langle \int_0^{\frac{\pi}{2}} \frac{e^{i \omega t}}{1 + e^{i \omega t}} \delta V dt \right\rangle^2 &= \int_0^{\frac{\pi}{2}} \frac{e^{i \omega t}}{1 + e^{i \omega t}} V(t) dt \\
 \frac{t \epsilon^2}{t_p} \left\langle \int_0^{\frac{\pi}{2}} \frac{e^{i \omega t}}{1 + e^{i \omega t}} \frac{\delta V}{V} dt \right\rangle^2 &= \int_0^{\frac{\pi}{2}} \frac{e^{i \omega t}}{1 + e^{i \omega t}} V(t) dt
 \end{aligned}$$

—720 μ s


$$\frac{d\tilde{E}}{dt} = \frac{\partial v}{\partial t} \int_0^{2\pi} \frac{\partial V}{\partial t} dt$$
$$\frac{d\tilde{E}}{dt} = \int_0^{2\pi} \frac{e^{i\omega t} \tilde{V}}{1 + e^{i\omega t}} dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2\pi} \left(\int_0^{2\pi} \frac{e^{i\omega t} \tilde{V}}{1 + e^{i\omega t}} dt + \int_0^{2\pi} \frac{e^{-i\omega t} \tilde{V}}{1 + e^{-i\omega t}} dt \right)$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2\pi} \left(\int_0^{2\pi} \frac{e^{i\omega t} \tilde{V}}{1 + e^{i\omega t}} dt + \int_0^{2\pi} \frac{e^{i\omega t} \tilde{V}}{1 + e^{i\omega t}} dt \right)$$

— 5 ms

$$\frac{d\tilde{E}}{dt} = \frac{\partial V}{\partial t}$$
$$\tilde{E} = \frac{1}{2} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} V^2 dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \frac{d}{dt} \left(\frac{V^2}{2} \right) dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} V \frac{dV}{dt} dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} V^2 dt$$

-40 ms

$$\begin{aligned}
 \frac{\langle \delta \tilde{E}_{th}^2 \rangle}{E^2} &= \frac{d\tilde{E}}{dt} = \frac{\partial V}{\partial t} \\
 \frac{t \epsilon^2}{t_p V^2} \left\langle \int_0^{\frac{\pi}{2}} \frac{e^{i \omega t}}{1 + e^{i \omega t}} \delta V dt \right\rangle^2 &= \int_0^{\frac{\pi}{2}} \frac{e^{i \omega t}}{1 + e^{i \omega t}} V(t) dt \\
 \frac{t \epsilon^2}{t_p} \left\langle \int_0^{\frac{\pi}{2}} \frac{e^{i \omega t}}{1 + e^{i \omega t}} \frac{\delta V}{V} dt \right\rangle^2 &= \int_0^{\frac{\pi}{2}} \frac{e^{i \omega t}}{1 + e^{i \omega t}} V(t) dt
 \end{aligned}$$

—360 ms

$$\begin{aligned}
 \frac{d\tilde{E}}{dt} &= \frac{\partial V}{\partial t} \\
 \frac{d\tilde{E}_{th}}{dt} &= \int_0^{2\pi} \frac{\partial V}{\partial t} dt \\
 \frac{d\tilde{E}_{th}}{dt} &= \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} V dt \\
 \frac{d\tilde{E}_{th}}{dt} &= \int_0^{2\pi} \frac{1}{1+e^{i\omega t}} \left(\frac{V}{2} + \frac{V}{2} \tanh \left(\frac{V}{2} \right) \right) dt
 \end{aligned}$$

—3.6 s

$$\frac{d\tilde{E}}{dt} = \frac{\partial \tilde{V}}{\partial t}$$
$$\tilde{E} = \frac{1}{2} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \tilde{V} dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \frac{\partial \tilde{V}}{\partial t} dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \frac{1}{2} \frac{\partial^2 \tilde{V}}{\partial t^2} dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{4} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \frac{\partial^2 \tilde{V}}{\partial t^2} dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{4} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \frac{\partial^2 \tilde{V}}{\partial t^2} dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{4} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \frac{\partial^2 \tilde{V}}{\partial t^2} dt$$

- 4 S

$$\frac{d\tilde{E}}{dt} = \frac{\partial \tilde{V}}{\partial t}$$
$$\tilde{E} = \frac{1}{2} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \tilde{V} dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2} \int_0^{2\pi} \frac{e^{i\omega t} \tilde{V}}{1+e^{i\omega t}} \omega dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2} \int_0^{2\pi} \tilde{V} dt$$

- 8 min

13



-1.7 h

$$\frac{d\tilde{E}}{dt} = \frac{\partial \tilde{V}}{\partial t}$$
$$\tilde{E} = \frac{1}{2} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \tilde{V} dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \frac{\partial \tilde{V}}{\partial t} dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2} \int_0^{2\pi} \frac{e^{i\omega t}}{1+e^{i\omega t}} \tilde{V}' dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2} \int_0^{2\pi} \tilde{V}' dt$$
$$\frac{d\tilde{E}}{dt} = \frac{1}{2} \tilde{V} \Big|_0^{2\pi}$$
$$\frac{d\tilde{E}}{dt} = \tilde{V}$$

- 1 d

15



-15 d

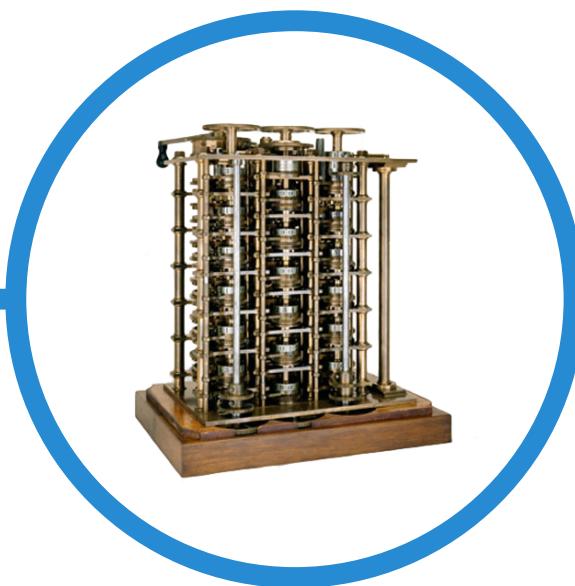
$$\frac{d\tilde{E}}{dt} = \frac{\partial \tilde{V}}{\partial t}$$
$$\frac{\langle \tilde{E}_{th}^2 \rangle}{E^2} = \frac{t \pi e^2}{t_P V^2} \left(\int_0^{2\pi} \frac{e^{i\omega t} \tilde{V} dt}{1 + e^{i\omega t}} \right)^2$$
$$= \int_0^{2\pi} \frac{e^{i\omega t} \tilde{V}}{1 + e^{i\omega t}} \left| \tilde{V} \right|^2 dt$$

- 240 d

17



-11 a



–200 a



-3.9 ka

20

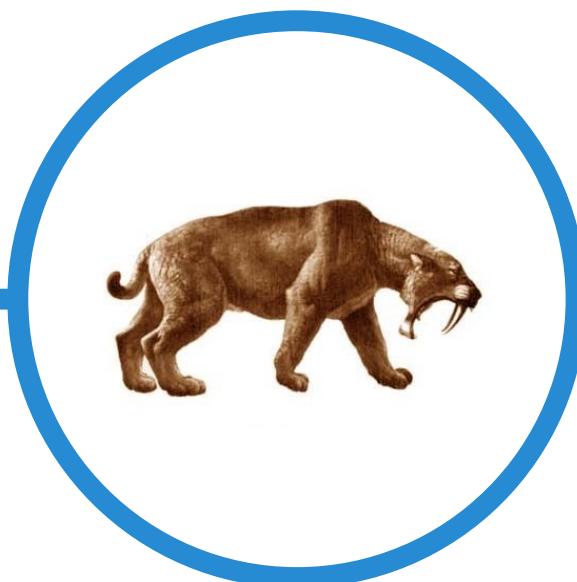


– 77 ka

21

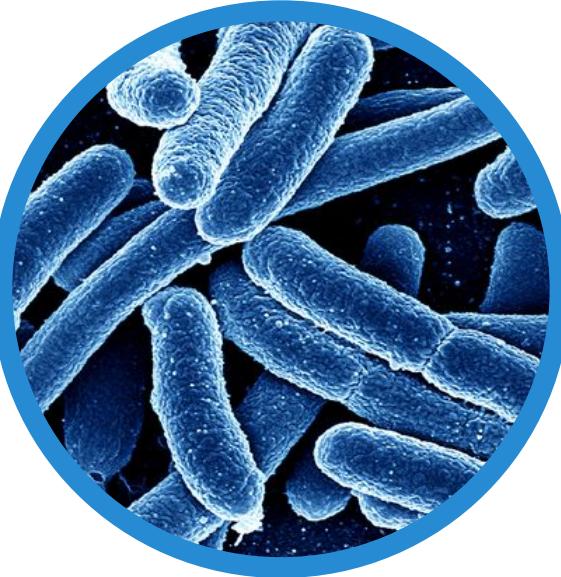


-1.6 Ma

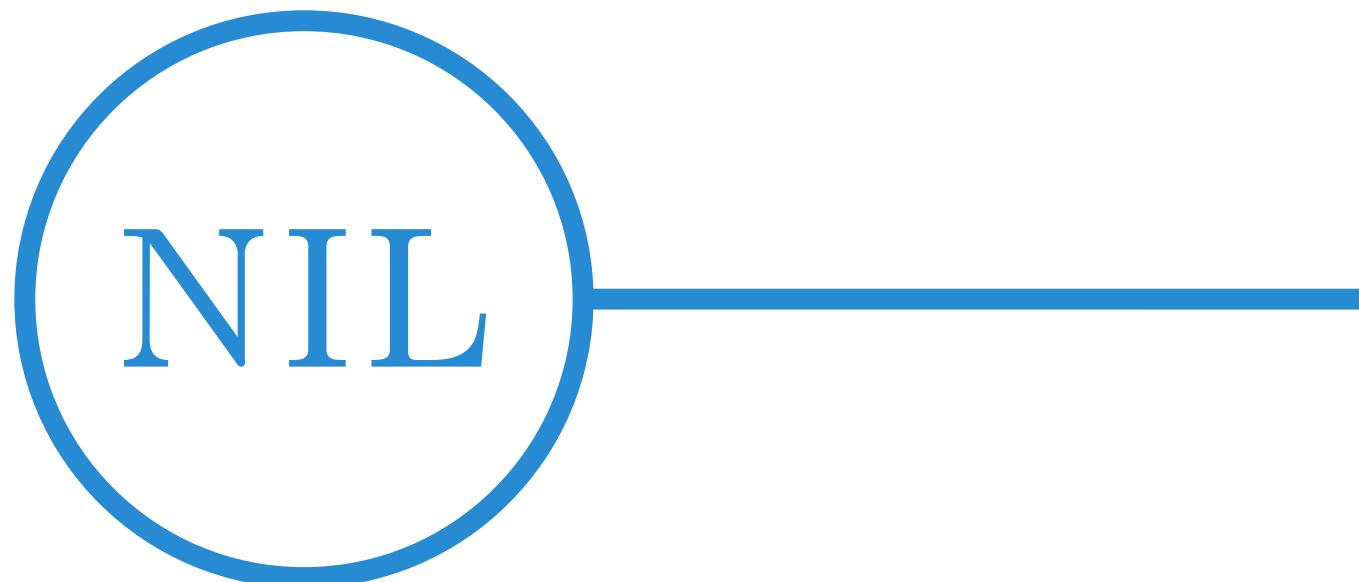


—36 Ma

23



-820 Ma



20 Ga

Reality check

- Vi kunne ha klart 23 par om vi startet ved The Big Bang. Men...
- Reelle tall for USA:
 - 121 678 på venteliste (2016)
 - 17 107 donorer (2014)

$$\begin{pmatrix} 121\,678 \\ 17\,107 \end{pmatrix} \mu\text{s}$$

$$\binom{121\,678}{17\,107} \text{μs} = \frac{121\,678!}{17\,107! \cdot (121\,678 - 17\,107)!} \text{μs}$$

$$\binom{121\,678}{17\,107} \mu\text{s} = \frac{121\,678!}{17\,107! \cdot (121\,678 - 17\,107)!} \mu\text{s}$$
$$= 1.6 \times 10^{21\,454} \mu\text{s}$$

$$\binom{121\,678}{17\,107} \mu\text{s} = \frac{121\,678!}{17\,107! \cdot (121\,678 - 17\,107)!} \mu\text{s}$$

$$= 1.6 \times 10^{21\,454} \mu\text{s}$$

$$= 4.9 \times 10^{21\,440} \text{ a}$$

$$\binom{121\,678}{17\,107} \mu\text{s} = \frac{121\,678!}{17\,107! \cdot (121\,678 - 17\,107)!} \mu\text{s}$$

$$= 1.6 \times 10^{21\,454} \mu\text{s}$$

$$= 4.9 \times 10^{21\,440} \text{ a}$$

$$= 4.9 \times 10^{20\,440} \times T_U$$

Dette er et grovt underestimatt - teller bare antall subsett av mottakerne som kan matches.

$$\binom{121\,678}{17\,107} \mu\text{s} = \frac{121\,678!}{17\,107! \cdot (121\,678 - 17\,107)!} \mu\text{s}$$

$$= 1.6 \times 10^{21\,454} \mu\text{s}$$

$$= 4.9 \times 10^{21\,440} \text{ a}$$

$$= 4.9 \times 10^{20\,440} \times T_U$$

Øk hastigheten med en centillion, og du får bare fjernet 600 av de 20440 nullene i antalle universlevetider...

Tid fra The Big Bang til Heat Death



Enklere problem

- Hver person har én numerisk score
- Match personer som har så lik score som mulig
- Cosinus-likhet: Sum av score-produkter for donor/mottaker
- Vil maksimere total cosinus-likhet
- Kan løses ved sortering!

Problemer

- **Problem:**

Relasjon mellom input og output

- **Instans:** Én bestemt input

- **Problemstørrelse:**

Lagringsplass som trengs for en instans

- Kan variere hvordan vi måler størrelse

- **Kjøretid:** Funksjon av problemstørrelse
- **Best-case:**
Beste mulige kjøretid for en gitt størrelse
- **Worst-case:** Verste mulige
- **Average-case:**
Forventet, gitt en sannsynlighetsfordeling
- Bruker vanligvis worst-case

RAM

- Kun enkle instruksjoner, som aritmetikk, flytting av data og programkontroll
- Disse tar konstant tid
- Vi kan håndtere heltall og flyttall
- Antar vanligvis at heltallene er maks $c \lg n$, for en eller større enn annen c større enn 1

Matematisk program

Spesialtilfelle: LP

min

$$\min x + 4y + 9z$$

$$\min x + 4y + 9z$$

$$\text{s.t. } x + y \leq 5$$

$$x + z \geq 10$$

$$-y - z = 7$$

$$\min x + 4y + 9z$$

$$\text{s.t. } x + y \leq 5$$

$$x + z \geq 10$$

$$-y - z = 7$$

```
from pulp import *
```

$$\min x + 4y + 9z$$

$$\text{s.t. } x + y \leq 5$$

$$x + z \geq 10$$

$$-y + z = 7$$

```
from pulp import *
prob = LpProblem("p", LpMinimize)
```

$$\min x + 4y + 9z$$

$$\text{s.t. } x + y \leq 5$$

$$x + z \geq 10$$

$$-y + z = 7$$

```
from pulp import *

prob = LpProblem("p", LpMinimize)

x = LpVariable("x")
y = LpVariable("y")
z = LpVariable("z")

prob += x + 4*y + 9*z
```

$$\min x + 4y + 9z$$

$$\text{s.t. } x + y \leq 5$$

$$x + z \geq 10$$

$$-y + z = 7$$

```
from pulp import *

prob = LpProblem("p", LpMinimize)

x = LpVariable("x")
y = LpVariable("y")
z = LpVariable("z")

prob += x + 4*y + 9*z

prob += x + y <= 5
prob += x + z >= 10
prob += -y + z == 7
```

$$\min x + 4y + 9z$$

$$\text{s.t. } x + y \leq 5$$

$$x + z \geq 10$$

$$-y + z = 7$$

```
from pulp import *

prob = LpProblem("p", LpMinimize)

x = LpVariable("x")
y = LpVariable("y")
z = LpVariable("z")

prob += x + 4*y + 9*z

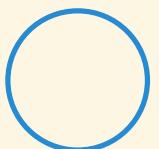
prob += x + y <= 5
prob += x + z >= 10
prob += -y + z == 7

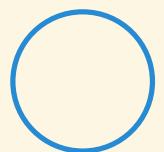
GLPK().solve(prob)
```

Effektive kommersielle
løsere («solvere») er f.eks.
Gurobi og CPLEX.

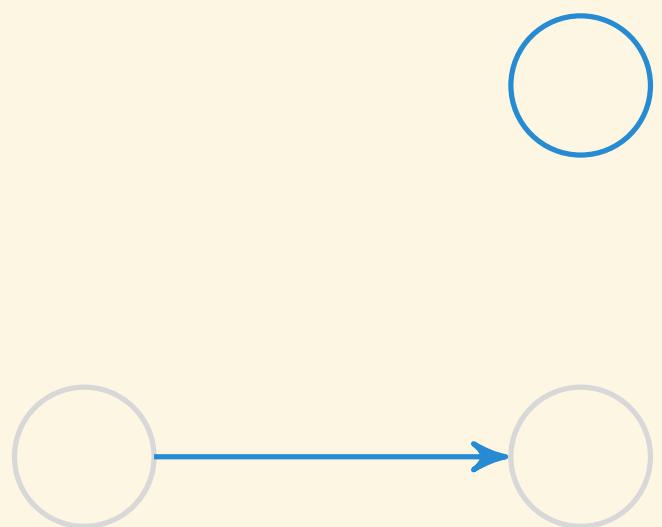
Korteste vei

Gurobi vs Dijkstra



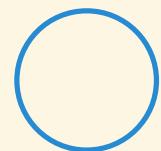


$|V| = 10\,000$



$$|V| = 10\,000$$

Første probleminstans. V
er noder og E er kanter.



$$|V| = 10\,000$$



$$|E| = 1\,000\,000$$

Dijkstra

$$\text{fib.1} = 1 \quad \text{fib.}(n+2) = \text{fib.}(n+1) + \text{fib.}n$$

$$\text{fib.}(X_{\text{gcd}} Y)$$

$$= \underbrace{\text{fib.}(x)}_{\text{gcd}} \text{fib.}(X_{\text{gcd}} Y) = \underbrace{\text{fib.}(y)}_{\text{fib.}(y)}$$

$$\text{fib.}(X_{\text{gcd}} Y) = (\text{fib.}X)_{\text{gcd}}$$

$$\text{fib.}(a+b) \text{ gcd}$$

$$\text{fib.}(a+b)$$

{ FIBONACCI }

$$= \text{fib.}(a-1) \cdot \text{fib.}b +$$

Kjøring med Dijkstras
algoritme.

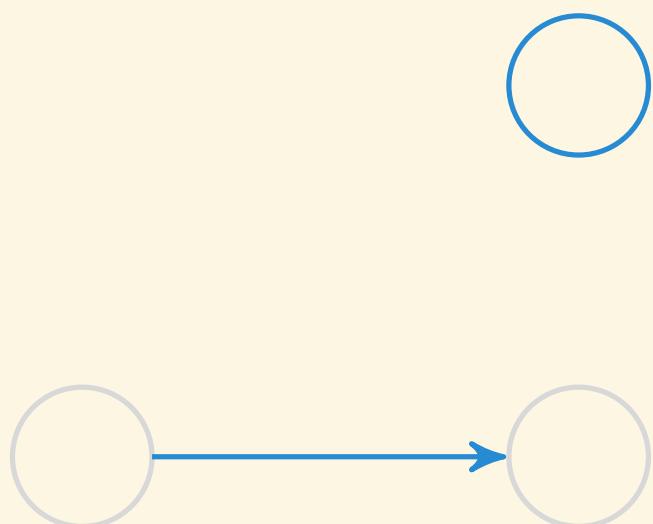
0.04 s



GUROBI
OPTIMIZATION

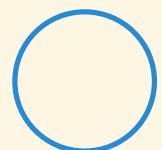
Kjøring med MIP-solver-pakken Gurobi.

31.7 s



$$|V| = 10\,000$$

$$|E| = 1\,000\,000$$



$$|V| = 100\,000$$



$$|E| = 10\,000\,000$$

Større probleminstans

Dijkstra

$$\text{fib.}_1 = 1 \quad \text{fib.}_n = \text{fib.}_{(n-1)} + \text{fib.}_{(n-2)}$$

$$\text{fib.}(X^{\text{gcd}} Y)$$

$$= \underbrace{\text{fib.}(X^{\text{gcd}})}_{\text{gcd}} \text{fib.}(X^{\text{gcd}} Y) = \underbrace{\text{fib.}(Y)}_{\text{final value}}$$

$$\text{fib.}(X^{\text{gcd}} Y) = (\text{fib.}_X)^{\text{gcd}}$$

$$\text{fib.}(a+b) \leq^{\text{gcd}}$$

$$\text{fib.}(a+b)$$

{ FIBONACCI }

$$= \text{fib.}(a-1) \cdot \text{fib.}(b) +$$

Dijkstras algoritme
fungerer fortsatt fint.

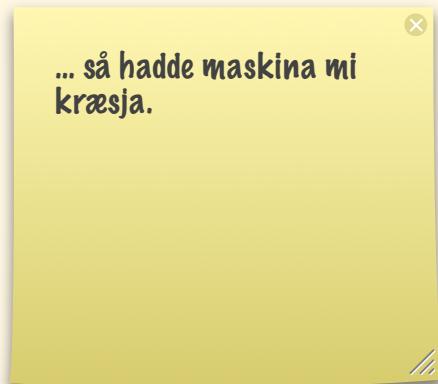
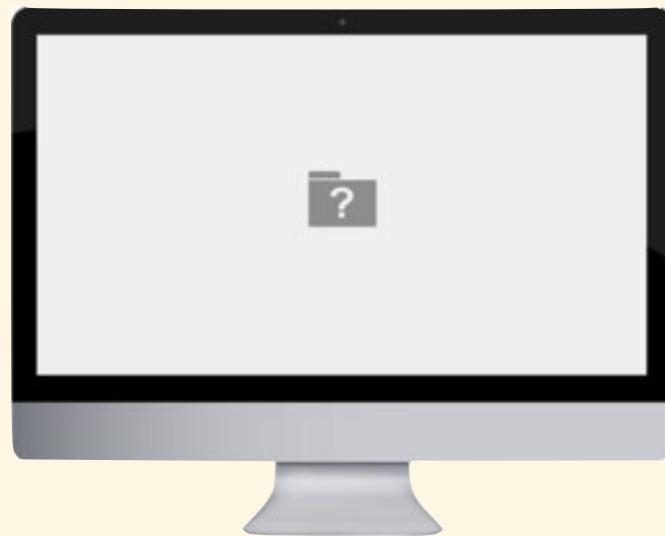
0.53 s



GUROBI
OPTIMIZATION

Meee... Gurobi funker ikke
så bra. Kjørte den over
natten, og ...

Neste dag...



Asymptotisk notasjon

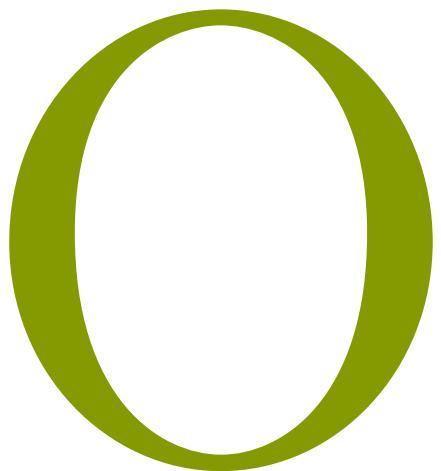
$\lg n$	logaritmisk	
\sqrt{n}	—	1×10^{31}
n	lineær	3×10^{15}
$n \lg n$	linearitmisk	7×10^{13}
n^2	kvadratisk	6×10^7
n^3	kubisk	1×10^5
2^n	eksponentiell	51
$n!$	faktoriell	17

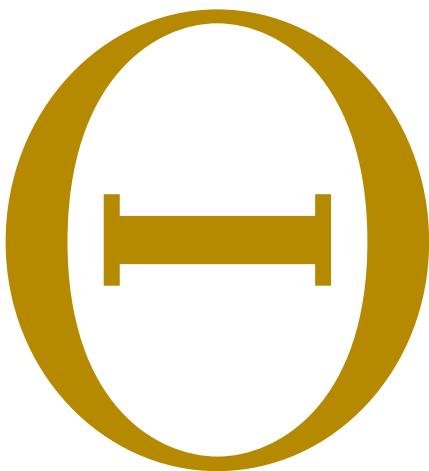
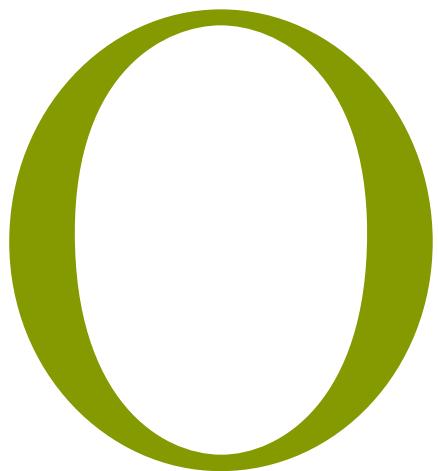
Alt på formen n^k kaller vi *polynomisk*
 (Ta en kikk på Problem 1-1)

$\lg n$	logaritmisk	$2 \times 10^{949\,978\,419\,116\,565}$
\sqrt{n}	—	1×10^{31}
n	lineær	3×10^{15}
$n \lg n$	linearitmisk	7×10^{13}
n^2	kvadratisk	6×10^7
n^3	kubisk	1×10^5
2^n	eksponentiell	51
$n!$	faktoriell	17

✖
 Tallene her er altså hvor
 stor n kan bli før det tar
 mer enn ett år å bli
 ferdig, dersom vi bruker
 $f(n)$ mikrosekunder, der
 $f(n)$ er funksjonen til
 venstre.

Alt på formen n^k kaller vi *polynomisk*
 (Ta en kikk på Problem 1-1)





O

H

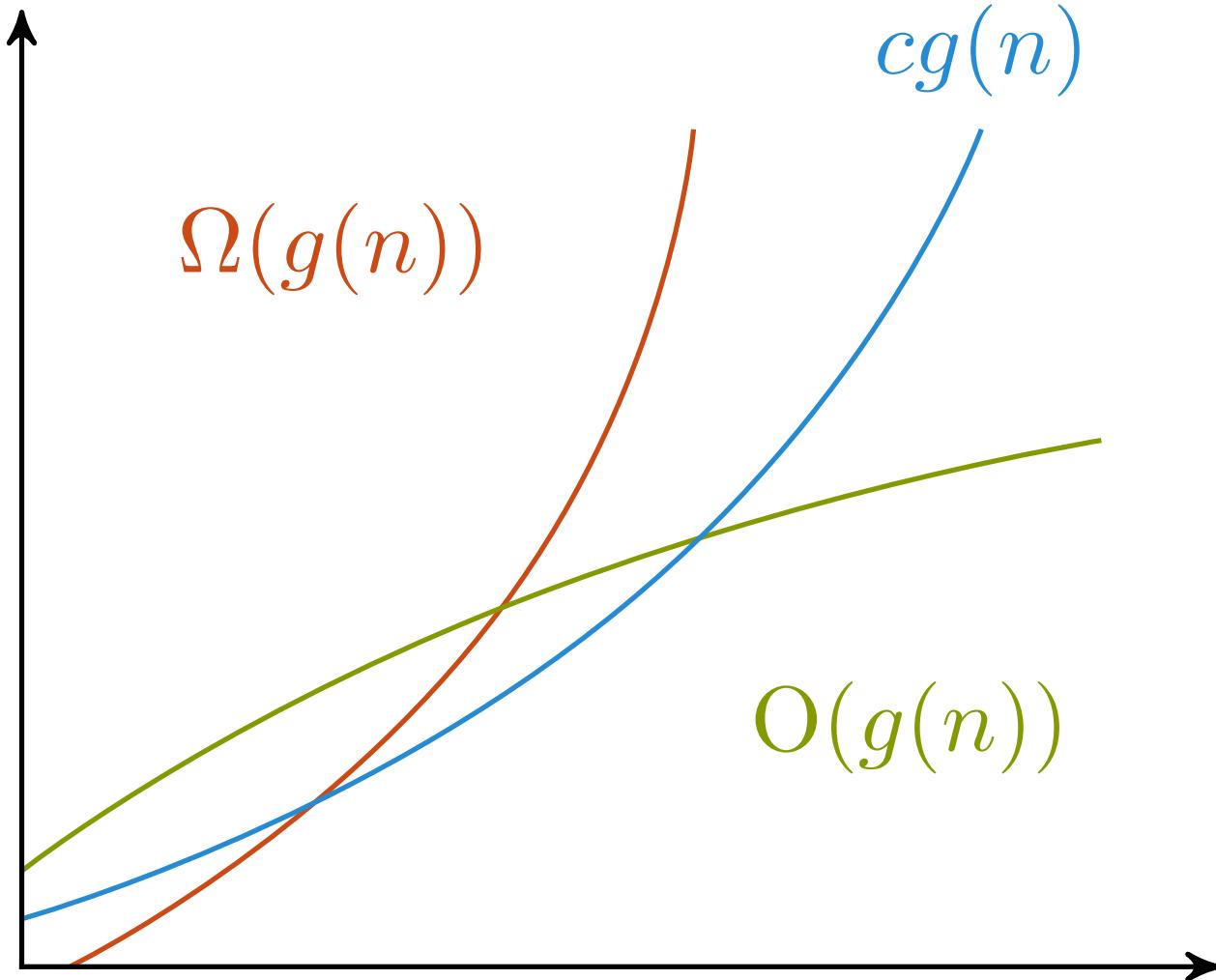
Ω

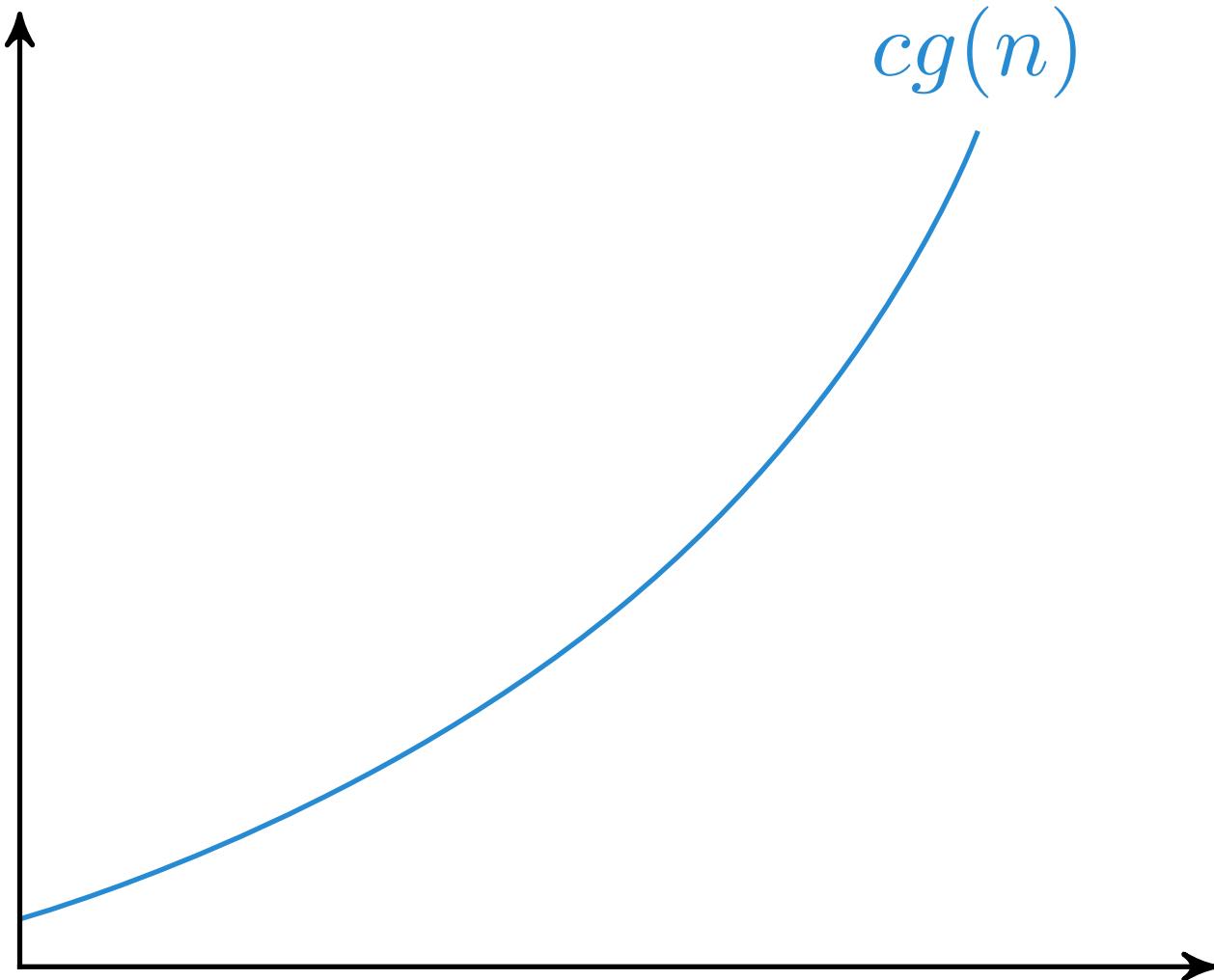
O

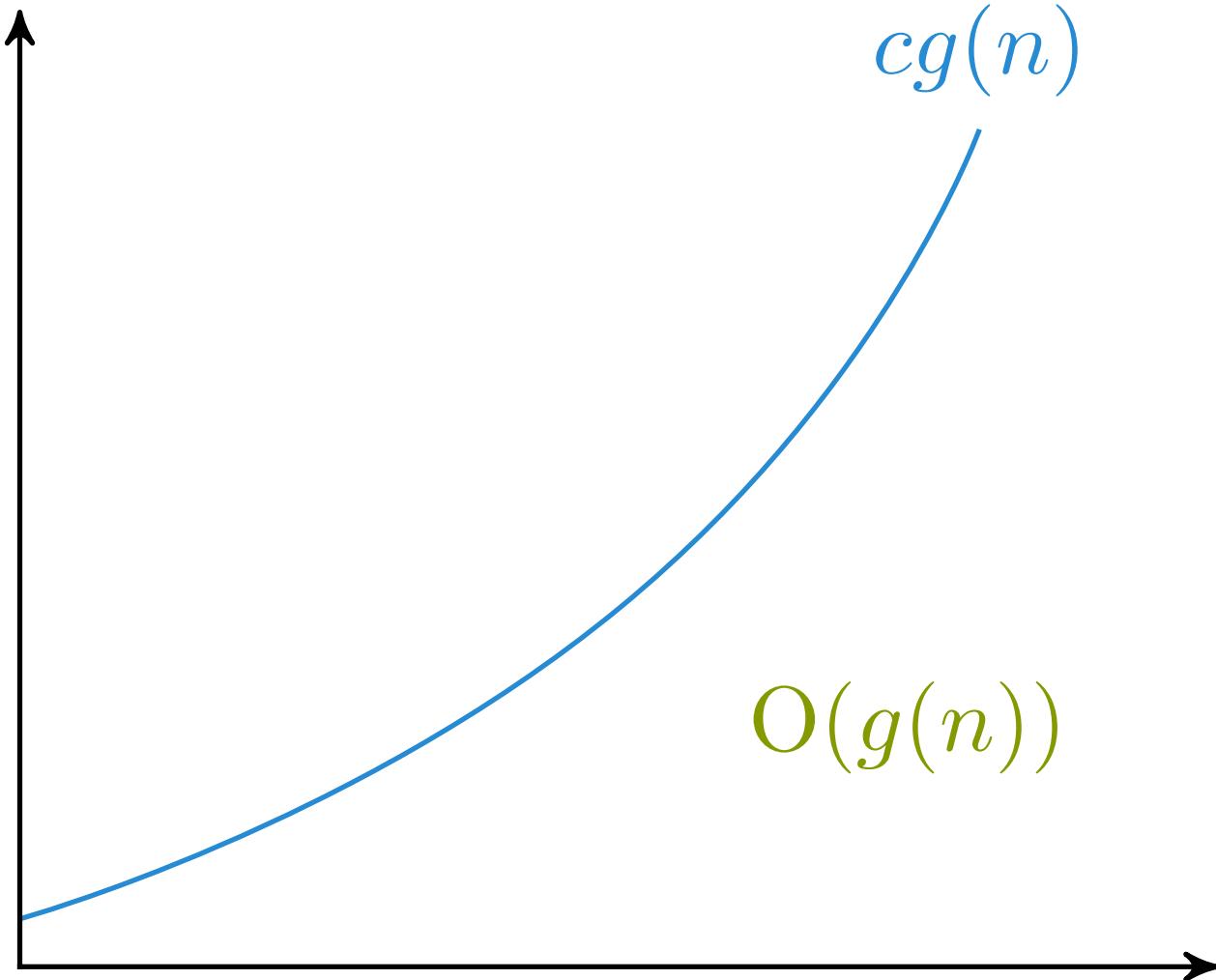
theta

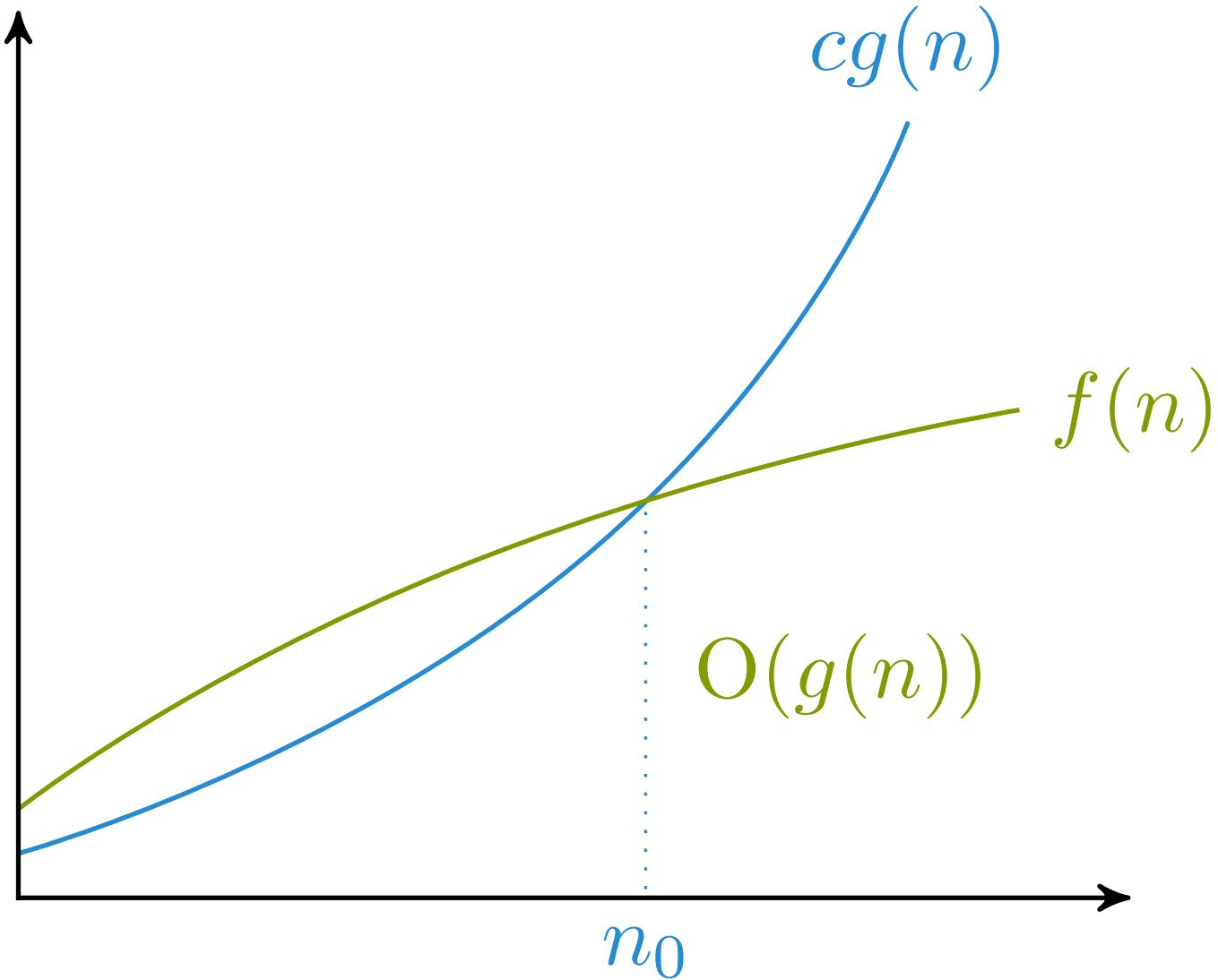
omega

O, theta og omega









$$f(n) = O(g(n))$$

$$f(n) = O(g(n))$$

Hvorfor ikke $f \in O(g)$?

Her gir $g(n)$ en asymptotisk øvre grense for $f(n)$.

$$f(n) = O(g(n))$$

Hvorfor ikke $f \in O(g)$?

Historisk ... og praktisk. $n^2 + O(n)$, etc.

$$f(n) = O(g(n))$$

$$f(n) = O(g(n))$$

$$\mathbb{I}$$

$$f(n) = O(g(n))$$

$$\Updownarrow$$

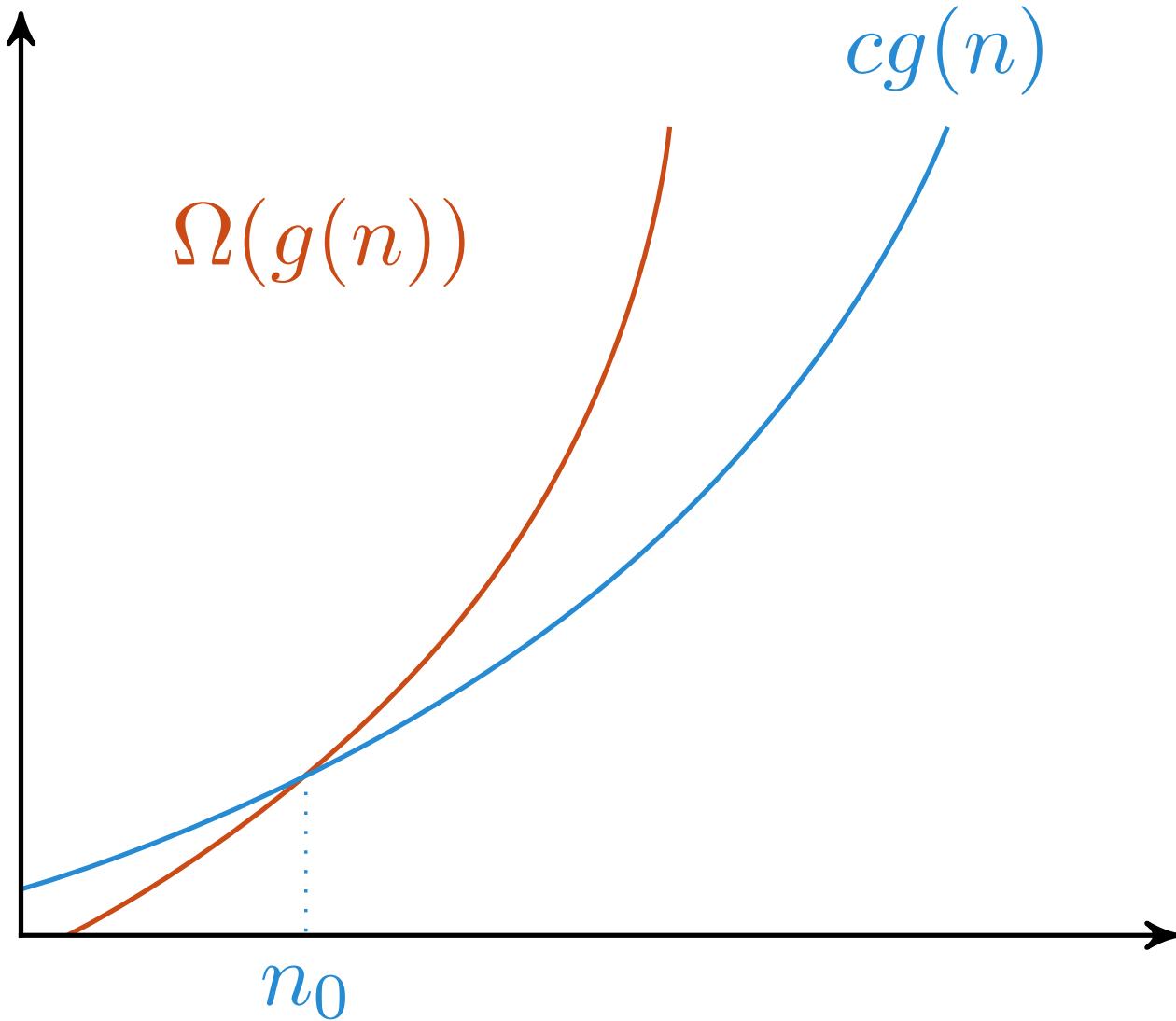
$$0 \leq f(n) \leq cg(n)$$

$$f(n) = O(g(n))$$

$$\Updownarrow$$

$$0 \leq f(n) \leq cg(n)$$

... for alle $n \geq n_0$, for en eller annen c og n_0



$$f(n) = \Omega(g(n))$$

$$f(n) = \Omega(g(n))$$



$$f(n) = \Omega(g(n))$$

$$\Updownarrow$$

$$0 \leq cg(n) \leq f(n)$$

$$f(n) = \Omega(g(n))$$



Her er $g(n)$ en asymptotisk ***nedre*** grense for $f(n)$

$$0 \leq cg(n) \leq f(n)$$

... for alle $n \geq n_0$, for en eller annen c og n_0

$$f(n) = \Theta(g(n))$$

$$f(n) = \Theta(g(n))$$

$$\mathbb{D}$$

$$f(n) = \Theta(g(n))$$



$$f(n) = O(g(n)) \text{ og } f(n) = \Omega(g(n))$$

Vi kan altså definere
Theta ved hjelp av O og
Omega.

$$f(n) = \mathrm{o}\big(g(n)\big)$$

$$f(n) = \omega\big(g(n)\big)$$

$$f(n) = o(g(n))$$

$$f(n) = \omega(g(n))$$

Disse er strengere
varianter av de store
operatorene.

Som før, men for *alle* $c > 0$

ω >

Ω \geq

Θ =

O \leq

O <

- Vet vi om vi har best- worst- eller average-case, kan alle tre brukes
- Vet vi det ikke, må vi velge en notasjon som favner alle muligheter

Dekomponering

- Del opp i delproblemer
- Løs delproblemene
- Lag løsning ut fra del-løsningene

Naturlig induksjon

P

P

•
•
•

Q

P

⋮

Q

$P \Rightarrow Q$

P

⋮

Q

$$\frac{}{P \Rightarrow Q} \text{ (I} \Rightarrow \text{)}$$

Vi kan bevise
implikasjonen "hvis P så
 Q " ved å midlertidig anta
 P og så utlede Q .

(Introduksjon av
implikasjon)

P

H

•
•
•

Q

$$\frac{}{P \Rightarrow Q} (I \Rightarrow) \cancel{H}$$

$$P \implies Q \qquad P$$

$$P \Rightarrow Q \qquad P$$

$$Q$$

$$\frac{P \Rightarrow Q \quad P}{Q} (\text{E} \Rightarrow)$$

Om vi allerede har
implikasjonen "hvis P så
 Q " og finner ut P , så kan
vi utlede Q .

(Eliminering av
implikasjon)

$$P(n)$$

XXX Burde bruke $P(k)$ og \forall n, for å være konsistent med bruken ellers.
Og kanskje til og med kjøre $P(k-1)$...

$$P(n)$$

$$\forall x P(x)$$

$$\frac{P(n)}{\forall x P(x)} (\text{I}\forall)$$

Hvis vi kan vise at P holder for et vilkårlig objekt fra et eller annet univers, uten å bruke noen egenskaper ved dette objektet som skiller det fra andre objekter...

vilkårlig

$$\frac{P(n)}{\forall x P(x)} (\text{I } \forall)$$

... så har vi vist at P gjelder for *alle* objekter fra dette universet.

$$P(n)$$

$$P(n)$$

$$\vdots$$

$$P(n+1)$$

$$P(n)$$

$$\vdots$$

$$P(n + 1)$$

$$\forall x(P(x) \Rightarrow P(x + 1))$$

$$P(n)$$

$$\vdots$$

$$P(n + 1)$$

$$\forall x(P(x) \Rightarrow P(x + 1)) \qquad P(1)$$

$$P(n)$$

$$\vdots$$

$$P(n + 1)$$

$$\forall x(P(x) \Rightarrow P(x + 1)) \qquad \qquad P(1)$$

$$\forall xP(x)$$

$$P(n)$$

$$\vdots$$

$$P(n+1)$$

$$\forall x(P(x) \Rightarrow P(x+1)) \qquad P(1)$$

$$\forall x P(x)$$

Naturlig induksjon:
Universet vårt er de
naturlige tall. Vi viser at
 P holder for 1, og at hvis
 P holder for n så holder
den også for $n+1$.

$$P(n)$$

⋮

for \mathbb{N}

$$P(n+1)$$

$$\frac{\forall x(P(x) \Rightarrow P(x+1)) \quad P(1)}{\forall x P(x)}$$

Da kan vi konkludere med
at P holder for alle
naturlige tall.

Alle hester har samme farge

- Én hest har samme farge som seg selv
- Anta at i enhver flokk med k hester så har alle samme farge
- Da vil alle i en flokk med $k+1$ hester også ha samme farge, siden den består av overlappende sett av k hester



$$\sum_{i=0}^{n-1} i = \frac{n(n-1)}{2}$$

$$\sum_{i=0}^{n-1} i$$

$$\frac{n(n-1)}{2}$$

$$n = 1$$

$$\sum_{i=0}^{n-1} i \qquad \qquad \frac{n(n-1)}{2}$$

$$n = 1$$

$$\sum_{i=0}^{n-1} i \quad \bigg| \quad \frac{n(n-1)}{2}$$

$$\sum_{i=0}^{1-1} i =$$

$$n = 1$$

$$\sum_{i=0}^{n-1} i \qquad \qquad \qquad \frac{n(n-1)}{2}$$

$$\sum_{i=0}^{1-1} i = 0$$

$$n = 1$$

$$\sum_{i=0}^{n-1} i \qquad \qquad \frac{n(n-1)}{2}$$

$$\sum_{i=0}^{1-1} i = 0$$

$$\frac{1(1-1)}{2} =$$

$$n = 1$$

$$\sum_{i=0}^{n-1} i \quad \left| \quad \frac{n(n-1)}{2}$$

$$\sum_{i=0}^{1-1} i = 0$$

$$\frac{1(1-1)}{2} = 0$$

Grunntilfelle: Vi har
klart å vise at venstre og
høyre side i ligningen
holder for $n = 1$, så
formelen stemmer her.

$$n = k$$

$$\sum_{i=0}^{n-1} i \quad \bigg| \quad \frac{n(n-1)}{2}$$

$$\text{H: } \sum_{i=0}^{k-2} i = \frac{(k-1)(k-2)}{2}$$

$$n = k$$

$$\sum_{i=0}^{n-1} i \quad \left| \quad \frac{n(n-1)}{2}$$

$$\text{H: } \sum_{i=0}^{k-2} i = \frac{(k-1)(k-2)}{2}$$

$$\frac{k(k-1)}{2}$$

$$n = k$$

$$\sum_{i=0}^{n-1} i$$

$$\frac{n(n-1)}{2}$$

$$\sum_{i=0}^{k-1} i =$$

$$\frac{k(k-1)}{2}$$

$$\text{H: } \sum_{i=0}^{k-2} i = \frac{(k-1)(k-2)}{2}$$

$$n = k$$

$$\sum_{i=0}^{n-1} i$$

$$\frac{n(n-1)}{2}$$

$$\sum_{i=0}^{k-1} i = \sum_{i=0}^{k-2} i + (k-1)$$

$$\frac{k(k-1)}{2}$$

H: $\sum_{i=0}^{k-2} i = \frac{(k-1)(k-2)}{2}$

$$n = k$$

$$\sum_{i=0}^{n-1} i$$

$$\frac{n(n-1)}{2}$$

$$\sum_{i=0}^{k-1} i = \sum_{i=0}^{k-2} i + (k-1)$$

$$\frac{k(k-1)}{2}$$

$$\text{H: } \sum_{i=0}^{k-2} i = \frac{(k-1)(k-2)}{2}$$

$$n = k$$

$$\sum_{i=0}^{n-1} i$$

$$\frac{n(n-1)}{2}$$

$$\sum_{i=0}^{k-1} i = \sum_{i=0}^{k-2} i + (k-1)$$

$$= \frac{(k-1)(k-2)}{2} + (k-1)$$

$$\frac{k(k-1)}{2}$$

H: $\sum_{i=0}^{k-2} i = \frac{(k-1)(k-2)}{2}$

$$n = k$$

$$\sum_{i=0}^{n-1} i$$

$$\frac{n(n-1)}{2}$$

$$\sum_{i=0}^{k-1} i = \sum_{i=0}^{k-2} i + (k-1)$$

$$= \frac{(k-1)(k-2)}{2} + (k-1)$$

$$= \frac{(k-2)(k-1) + 2(k-1)}{2}$$

$$\frac{k(k-1)}{2}$$

H: $\sum_{i=0}^{k-2} i = \frac{(k-1)(k-2)}{2}$

$$n = k$$

$$\sum_{i=0}^{n-1} i$$

$$\frac{n(n-1)}{2}$$

$$\sum_{i=0}^{k-1} i = \sum_{i=0}^{k-2} i + (k-1)$$

$$= \frac{(k-1)(k-2)}{2} + (k-1)$$

$$= \frac{(k-2)(k-1) + 2(k-1)}{2}$$

$$= \frac{k(k-1)}{2}$$

$$\text{H: } \sum_{i=0}^{k-2} i = \frac{(k-1)(k-2)}{2}$$

$$\frac{k(k-1)}{2}$$

Induktivt trinn: Vi antar at formelen holder for $n = k-1$ (induktiv hypotese), og viser at høyre og venstre side da blir like for $n = k$. Beviset er da komplett.

$$\sum_{i=0}^{n-1} 2^i = 2^n - 1$$

$$\sum_{i=0}^{n-1} 2^i$$

$$2^n - 1$$

$$n = 1$$

$$\sum_{i=0}^{n-1} 2^i$$

$$2^n - 1$$

$$n = 1$$

$$\sum_{i=0}^{n-1} 2^i$$

$$2^n - 1$$

$$\sum_{i=0}^{1-1} 2^i =$$

$$n = 1$$

$$\sum_{i=0}^{n-1} 2^i$$

$$2^n - 1$$

$$\sum_{i=0}^{1-1} 2^i = 1$$

$$n = 1$$

$$\sum_{i=0}^{n-1} 2^i$$

$$2^n - 1$$

$$\sum_{i=0}^{1-1} 2^i = 1$$

$$2^1 - 1 =$$

$n = 1$

$$\sum_{i=0}^{n-1} 2^i \quad \boxed{2^n - 1}$$

Grunntilfellet stemmer,
siden venstre og høyre
side er like når $n = 1$.

$$\sum_{i=0}^{1-1} 2^i = 1$$

$$2^1 - 1 = 1$$

$$n = k$$

$$\sum_{i=0}^{n-1} 2^i$$

$$2^n - 1$$

$$\text{H: } \sum_{i=0}^{k-2} 2^i = 2^{k-1} - 1$$

$$n = k$$

$$\sum_{i=0}^{n-1} 2^i$$

$$2^n - 1$$

$$2^k - 1$$

$$\text{H: } \sum_{i=0}^{k-2} 2^i = 2^{k-1} - 1$$

$$n = k$$

$$\sum_{i=0}^{n-1} 2^i$$

$$2^n - 1$$

$$\sum_{i=0}^{k-1} 2^i =$$

$$2^k - 1$$

$$\text{H: } \sum_{i=0}^{k-2} 2^i = 2^{k-1} - 1$$

$$n = k$$

$$\sum_{i=0}^{n-1} 2^i$$

$$2^n - 1$$

$$\sum_{i=0}^{k-1} 2^i = \sum_{i=0}^{k-2} 2^i + 2^{k-1}$$

$$2^k - 1$$

H: $\sum_{i=0}^{k-2} 2^i = 2^{k-1} - 1$

$$n = k$$

$$\sum_{i=0}^{n-1} 2^i$$

$$2^n - 1$$

$$\sum_{i=0}^{k-1} 2^i = \sum_{i=0}^{k-2} 2^i + 2^{k-1}$$

$$2^k - 1$$

$$\text{H: } \sum_{i=0}^{k-2} 2^i = 2^{k-1} - 1$$

$$n = k$$

$$\sum_{i=0}^{n-1} 2^i$$

$$2^n - 1$$

$$\sum_{i=0}^{k-1} 2^i = \sum_{i=0}^{k-2} 2^i + 2^{k-1}$$

$$= 2^{k-1} - 1 + 2^{k-1}$$

$$2^k - 1$$

H: $\sum_{i=0}^{k-2} 2^i = 2^{k-1} - 1$

$$n = k$$

$$\sum_{i=0}^{n-1} 2^i$$

$$2^n - 1$$

$$\sum_{i=0}^{k-1} 2^i = \sum_{i=0}^{k-2} 2^i + 2^{k-1}$$

$$= 2^{k-1} - 1 + 2^{k-1}$$

$$= 2^k - 1$$

$$\text{H: } \sum_{i=0}^{k-2} 2^i = 2^{k-1} - 1$$

$$2^k - 1$$

Induktivt trinn: Antar at ligningen holder for $n = k-1$ og viser at den da også må holde for $n = k$.

Sterk induksjon
Antagelsen gjelder
alle lavere tall

Løkkeinvarianter

- Brukes til bevis for løkker
- **Initialisering:** Den holder før vi starter
- **Vedlikehold:**
Holder den før en iterasjon, så holder den etter
- **Terminering:**
Når løkka terminerer sier den noe nyttig

Insertion-Sort



Dekomponering/
induksjon: Anta at vi
kan sortere en kortere
sekvens. Hvordan kan vi
da sortere hele
sekvensen?

Invariant:

Prefikset så langt er sortert

INSERTION-SORT(A)

```
INSERTION-SORT( $A$ )
1  for  $j = 2$  to  $A.length$ 
```

INSERTION-SORT(A)

1 **for** $j = 2$ **to** $A.length$
2 $key = A[j]$

INSERTION-SORT(A)

1 **for** $j = 2$ **to** $A.length$
2 $key = A[j]$
3 $i = j - 1$

INSERTION-SORT(A)

- 1 **for** $j = 2$ **to** $A.length$
- 2 $key = A[j]$
- 3 $i = j - 1$
- 4 **while** $i > 0$ and $A[i] > key$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
```

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
```

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

Sett hvert tall inn i det initiatelle segmentet som allerede er sortert, slik at det resulterende segmentet også er sortert.

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

$i, j, key = -, -, -$

5	1
2	2
4	3
6	4
1	5
3	6

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

5	1
2	2
4	3
6	4
1	5
3	6

$i, j, key = -, 2, -$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

j	5	1
	2	2
	4	3
	6	4
	1	5
	3	6

$i, j, key = -, 2, 2$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

$i, j, key = 1, 2, 2$

i	5	1
j	2	2
	4	3
	6	4
	1	5
	3	6

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

i	5	1
j	2	2
	4	3
	6	4
	1	5
	3	6

$i, j, key = 1, 2, 2$

INSERTION-SORT(A)

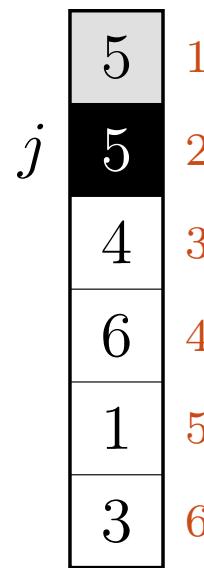
```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

i	5	1
j	5	2
	4	3
	6	4
	1	5
	3	6

$i, j, key = 1, 2, 2$

INSERTION-SORT(A)

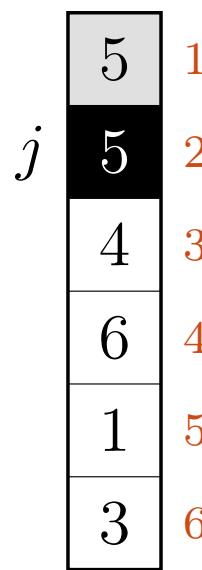
```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```



$i, j, key = 0, 2, 2$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```



$i, j, key = 0, 2, 2$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

$i, j, key = 0, 2, 2$

2	1
5	2
4	3
6	4
1	5
3	6

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

2	1
5	2
4	3
6	4
1	5
3	6

$i, j, key = 0, 3, 2$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

2	1
5	2
4	3
6	4
1	5
3	6

j

$i, j, key = 0, 3, 4$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

	2	1
i	5	2
j	4	3
	6	4
	1	5
	3	6

$i, j, key = 2, 3, 4$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

	2	1
i	5	2
j	4	3
	6	4
	1	5
	3	6

$i, j, key = 2, 3, 4$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

	2	1
i	5	2
j	5	3
	6	4
	1	5
	3	6

$i, j, key = 2, 3, 4$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

i	2	1
	5	2
j	5	3
	6	4
	1	5
	3	6

$i, j, key = 1, 3, 4$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

$i, j, key = 1, 3, 4$

i	2	1
	5	2
j	5	3
	6	4
	1	5
	3	6

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

i	2	1
	4	2
j	5	3
	6	4
	1	5
	3	6

$i, j, key = 1, 3, 4$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

i	2	1
	4	2
	5	3
	6	4
j	1	5
	3	6

$i, j, key = 1, 4, 4$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

$i, j, key = 1, 4, 6$

i	2	1
	4	2
	5	3
j	6	4
	1	5
	3	6

INSERTION-SORT(A)

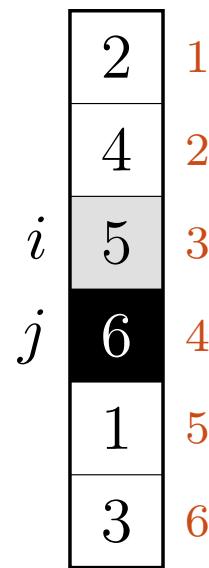
```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

2	1
4	2
i	5
j	6
1	4
3	5
6	6

$i, j, key = 3, 4, 6$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```



$i, j, key = 3, 4, 6$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

2	1
4	2
i	3
j	4
6	5
1	6
3	

$i, j, key = 3, 4, 6$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

2	1
4	2
5	3
6	4
1	5
3	6

$i, j, key = 3, 5, 6$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

2	1
4	2
5	3
6	4
1	5
3	6

$i, j, key = 3, 5, 1$

INSERTION-SORT(A)

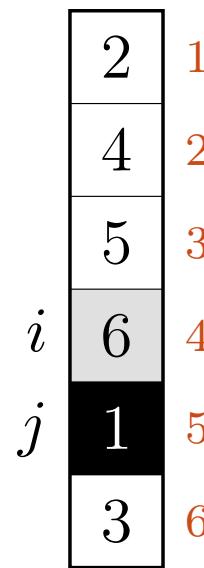
```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

2	1
4	2
5	3
i	4
j	5
1	6
3	

$i, j, key = 4, 5, 1$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```



$i, j, key = 4, 5, 1$

INSERTION-SORT(A)

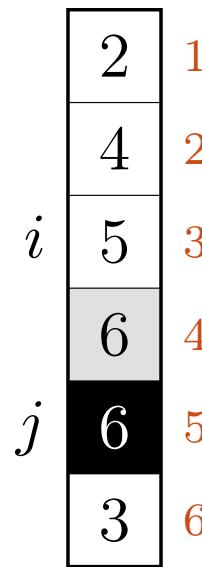
```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

2	1
4	2
5	3
i	6
j	5
	3

$$i, j, key = 4, 5, 1$$

INSERTION-SORT(A)

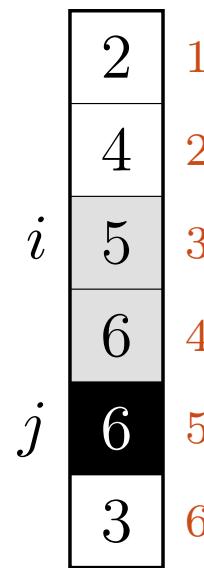
```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```



$i, j, key = 3, 5, 1$

INSERTION-SORT(A)

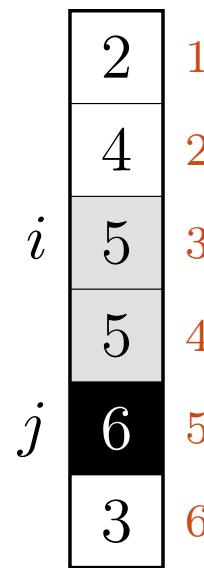
```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```



$i, j, key = 3, 5, 1$

INSERTION-SORT(A)

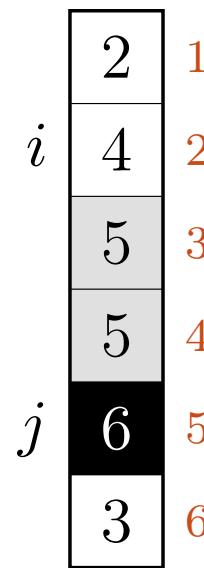
```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```



$i, j, key = 3, 5, 1$

INSERTION-SORT(A)

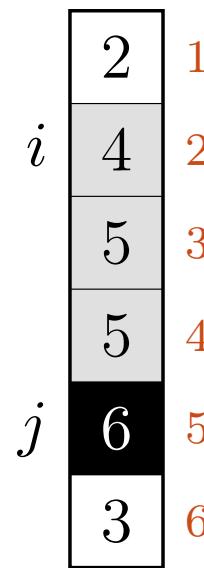
```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```



$i, j, key = 2, 5, 1$

INSERTION-SORT(A)

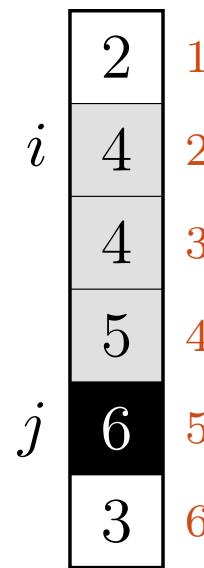
```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```



$i, j, key = 2, 5, 1$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```



$i, j, key = 2, 5, 1$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

i	2	1
	4	2
	4	3
	5	4
j	6	5
	3	6

$i, j, key = 1, 5, 1$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

i	2	1
	4	2
	4	3
	5	4
j	6	5
	3	6

$i, j, key = 1, 5, 1$

INSERTION-SORT(A)

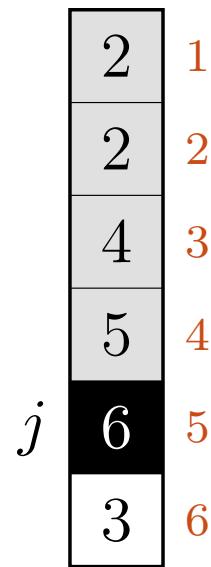
```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

i	2	1
	2	2
	4	3
	5	4
j	6	5
	3	6

$i, j, key = 1, 5, 1$

INSERTION-SORT(A)

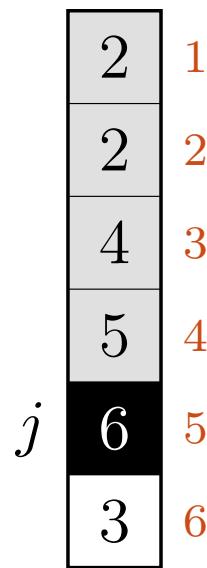
```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```



$i, j, key = 0, 5, 1$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```



$i, j, key = 0, 5, 1$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

1	1
2	2
4	3
5	4
6	5
3	6

j

$i, j, key = 0, 5, 1$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

1	1
2	2
4	3
5	4
6	5
j	3
	6

$i, j, key = 0, 6, 1$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

1	1
2	2
4	3
5	4
6	5
j	3
	6

$i, j, key = 0, 6, 3$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

1	1
2	2
4	3
5	4
i	5
6	5
j	3
	6

$i, j, key = 5, 6, 3$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

1	1
2	2
4	3
5	4
i	5
6	5
j	3
	6

$i, j, key = 5, 6, 3$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

1	1
2	2
4	3
5	4
i	5
j	6

$i, j, key = 5, 6, 3$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

1	1
2	2
4	3
5	4
6	5
i	
j	6

$i, j, key = 4, 6, 3$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

1	1
2	2
4	3
5	4
6	5
i	
j	6

$i, j, key = 4, 6, 3$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

1	1
2	2
4	3
5	4
5	5
i	
j	6

$i, j, key = 4, 6, 3$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

1	1
2	2
4	3
5	4
5	5
6	6

i

j

$i, j, key = 3, 6, 3$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

1	1
2	2
4	3
5	4
5	5
6	6

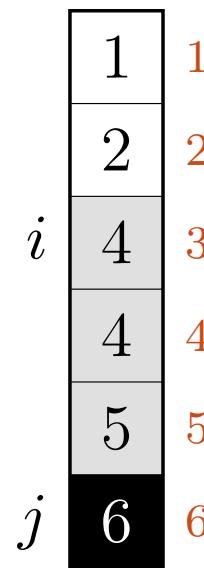
i

j

$i, j, key = 3, 6, 3$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```



$i, j, key = 3, 6, 3$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

i	1	1
	2	2
	4	3
	4	4
	5	5
j	6	6

$i, j, key = 2, 6, 3$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

i	1	1
	2	2
	4	3
	4	4
	5	5
j	6	6

$i, j, key = 2, 6, 3$

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

1	1
2	2
3	3
4	4
5	5
6	6

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

INSERTION-SORT(A)

```

1  for  $j = 2$  to  $A.length$  .....  $n$ 
2       $key = A[j]$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 

```

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$  .....  $n$ 
2       $key = A[j]$  .....  $n - 1$ 
3       $i = j - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$  .....  $n$ 
2       $key = A[j]$  .....  $n - 1$ 
3       $i = j - 1$  .....  $n - 1$ 
4      while  $i > 0$  and  $A[i] > key$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

INSERTION-SORT(A)

```
1  for  $j = 2$  to  $A.length$  .....  $n$ 
2       $key = A[j]$  .....  $n - 1$ 
3       $i = j - 1$  .....  $n - 1$ 
4      while  $i > 0$  and  $A[i] > key$  .....  $\sum_{j=2}^n t_j$ 
5           $A[i + 1] = A[i]$ 
6           $i = i - 1$ 
7       $A[i + 1] = key$ 
```

INSERTION-SORT(A)

1	for $j = 2$ to $A.length$	n
2	$key = A[j]$	$n - 1$
3	$i = j - 1$	$n - 1$
4	while $i > 0$ and $A[i] > key$	$\sum_{j=2}^n t_j$
5	$A[i + 1] = A[i]$	$\sum_{j=2}^n (t_j - 1)$
6	$i = i - 1$		
7	$A[i + 1] = key$		

INSERTION-SORT(A)

1	for $j = 2$ to $A.length$	n
2	$key = A[j]$	$n - 1$
3	$i = j - 1$	$n - 1$
4	while $i > 0$ and $A[i] > key$	$\sum_{j=2}^n t_j$
5	$A[i + 1] = A[i]$	$\sum_{j=2}^n (t_j - 1)$
6	$i = i - 1$	$\sum_{j=2}^n (t_j - 1)$
7	$A[i + 1] = key$		

INSERTION-SORT(A)

```

1  for  $j = 2$  to  $A.length$  .....  $n$ 
2     $key = A[j]$  .....  $n - 1$ 
3     $i = j - 1$  .....  $n - 1$ 
4    while  $i > 0$  and  $A[i] > key$  .....  $\sum_{j=2}^n t_j$ 
5       $A[i + 1] = A[i]$  .....  $\sum_{j=2}^n (t_j - 1)$ 
6       $i = i - 1$  .....  $\sum_{j=2}^n (t_j - 1)$ 
7     $A[i + 1] = key$  .....  $n - 1$ 

```

INSERTION-SORT(A)

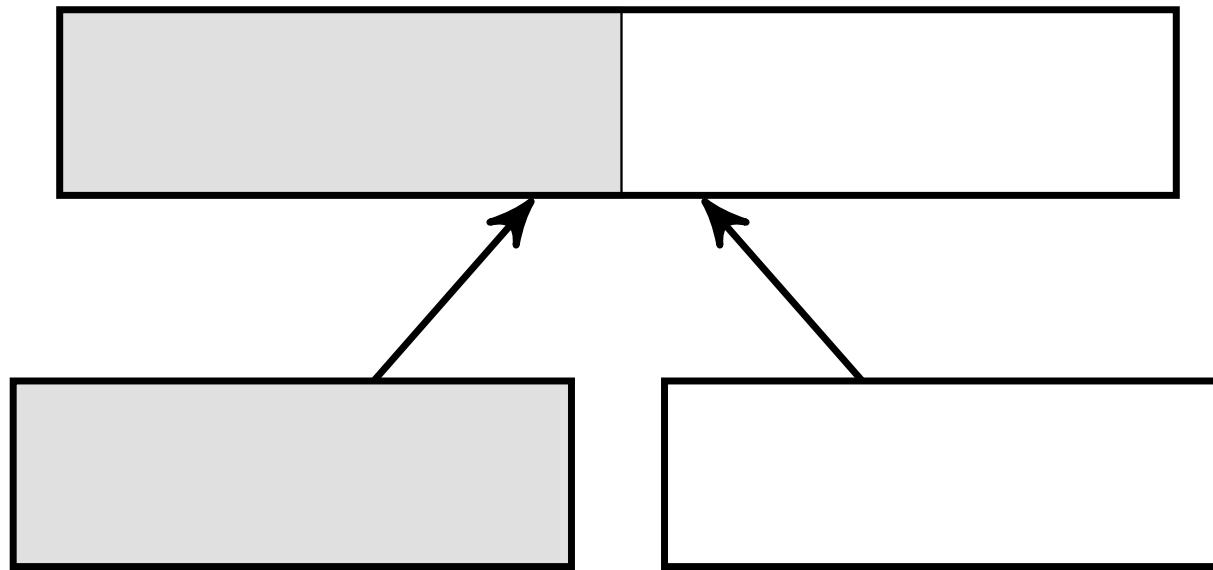
```

1  for  $j = 2$  to  $A.length$  .....  $n$ 
2     $key = A[j]$  .....  $n - 1$ 
3     $i = j - 1$  .....  $n - 1$ 
4    while  $i > 0$  and  $A[i] > key$  .....  $\sum_{j=2}^n t_j$ 
5       $A[i + 1] = A[i]$  .....  $\sum_{j=2}^n (t_j - 1)$ 
6       $i = i - 1$  .....  $\sum_{j=2}^n (t_j - 1)$ 
7     $A[i + 1] = key$  .....  $n - 1$ 

```

$$T(n) = \Theta(n^2)$$

Merge-Sort



Anta at vi kan sortere
alle kortere sekvenser.
Mer spesifikt, anta at vi
sorterer hver halvdel.
Hvordan kan vi så få
sortert hele sekvensen?

Ikke løkkeinvariant, men
induksjonshypotese:
Hver halvdel er sortert

Evt.: Induksjon på antall elementer

Divide, Conquer & Combine

MERGE-SORT(A, p, r)

MERGE-SORT(A, p, r)
1 **if** $p < r$

MERGE-SORT(A, p, r)

1 **if** $p < r$

2 $q = \lfloor (p + r)/2 \rfloor$

```
MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
```

```
MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
```

```
MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )
```

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8$

p	5	1
	2	2
	4	3
	7	4
	1	5
	3	6
	2	7
r	6	8

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8$

p	5	1
	2	2
	4	3
	7	4
	1	5
	3	6
	2	7
r	6	8

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8$

p	5	1
	2	2
	4	3
q	7	4
	1	5
	3	6
	2	7
r	6	8

MERGE-SORT(A, p, r)

```
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )
```

$p, r = 1, 8 \rightarrow 1, 4$

p	5	1
	2	2
	4	3
r	7	4
	1	5
	3	6
	2	7
	6	8

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 \rightarrow 1, 4$

p	5	1
	2	2
	4	3
r	7	4
	1	5
	3	6
	2	7
	6	8

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 \rightarrow 1, 4$

p	5	1
q	2	2
	4	3
r	7	4
	1	5
	3	6
	2	7
	6	8

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 1, 2$

p	5	1
r	2	2
	4	3
	7	4
	1	5
	3	6
	2	7
	6	8

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 1, 2$

p	5	1
r	2	2
	4	3
	7	4
	1	5
	3	6
	2	7
	6	8

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

p, q	5	1
r	2	2
	4	3
	7	4
	1	5
	3	6
	2	7
	6	8

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 1, 2$

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

p, r	5	1
	2	2
	4	3
	7	4
	1	5
	3	6
	2	7
	6	8

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 1, 2 \rightarrow 1, 1$

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 1, 2$

p	5	1
r	2	2
	4	3
	7	4
	1	5
	3	6
	2	7
	6	8

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

p, r	1
5	2
2	2
4	3
7	4
1	5
3	6
2	7
6	8

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 1, 2 \rightarrow 2, 2$

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 1, 2$

p	5	1
r	2	2
	4	3
	7	4
	1	5
	3	6
	2	7
	6	8

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 \rightarrow 1, 4$

p	2	1
	5	2
	4	3
r	7	4
	1	5
	3	6
	2	7
	6	8

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 3, 4$

2	1
5	2
4	3
7	4
1	5
3	6
2	7
6	8

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 3, 4$

2	1
5	2
4	3
7	4
1	5
3	6
2	7
6	8

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

p, q	2	1
	5	2
r	4	3
	7	4
	1	5
	3	6
	2	7
	6	8

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 3, 4$

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

p, r	2	1
	5	2
	4	3
	7	4
	1	5
	3	6
	2	7
	6	8

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 3, 4 \rightarrow 3, 3$

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 3, 4$

2	1
5	2
4	3
7	4
1	5
3	6
2	7
6	8

MERGE-SORT(A, p, r)

```
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )
```

2	1
5	2
4	3
7	4
1	5
3	6
2	7
6	8

p, r

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 3, 4 \rightarrow 4, 4$

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 3, 4$

2	1
5	2
4	3
p	
7	4
1	5
3	6
2	7
6	8
r	

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 \rightarrow 1, 4$

p	2	1
	5	2
	4	3
r	7	4
	1	5
	3	6
	2	7
	6	8

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8$

p	2	1
	4	2
	5	3
	7	4
	1	5
	3	6
	2	7
r	6	8

MERGE-SORT(A, p, r)

```
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )
```

$p, r = 1, 8 > 5, 8$

2	1
4	2
5	3
7	4
1	5
3	6
2	7
6	8

p

r

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 > 5, 8$

2	1
4	2
5	3
7	4
1	5
3	6
2	7
6	8

p

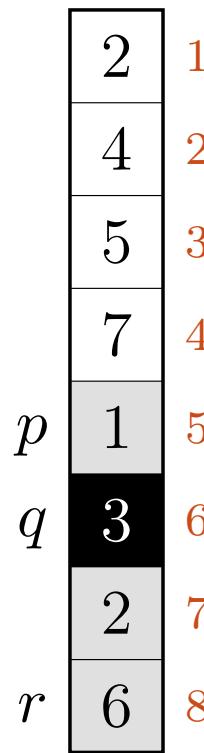
r

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 > 5, 8$



MERGE-SORT(A, p, r)

```
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )
```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 5, 6$

2	1
4	2
5	3
7	4
1	5
3	6
2	7
6	8

p

r

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 5, 6$

2	1
4	2
5	3
7	4
1	5
3	6
2	7
6	8

p

r

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

2	1
4	2
5	3
7	4
1	5
3	6
2	7
6	8

p, q

r

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 5, 6$

MERGE-SORT(A, p, r)

```
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )
```

2	1
4	2
5	3
7	4
1	5
3	6
2	7
6	8

p, r

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 5, 6 \rightarrow 5, 5$

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 5, 6$

2	1
4	2
5	3
7	4
1	5
3	6
2	7
6	8

p

r

MERGE-SORT(A, p, r)

```
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )
```

p, r

2	1
4	2
5	3
7	4
1	5
3	6
2	7
6	8

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 5, 6 \rightarrow 6, 6$

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 5, 6$

2	1
4	2
5	3
7	4
1	5
3	6
2	7
6	8

p

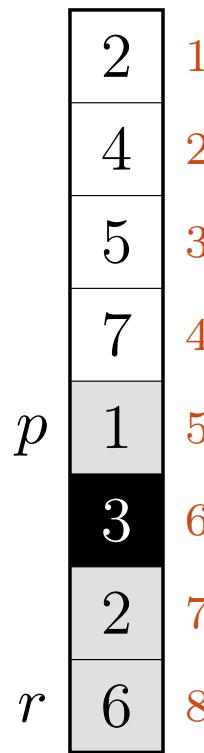
r

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 \rightarrow 5, 8$



MERGE-SORT(A, p, r)

```
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )
```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 7, 8$

2	1
4	2
5	3
7	4
1	5
3	6
p	7
r	8

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 7, 8$

2	1
4	2
5	3
7	4
1	5
3	6
p	7
r	8

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

2	1
4	2
5	3
7	4
1	5
3	6
2	7
6	8

p, q

r

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 7, 8$

MERGE-SORT(A, p, r)

```
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )
```

2	1
4	2
5	3
7	4
1	5
3	6
2	7
6	8

p, r

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 7, 8 \rightarrow 7, 7$

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 7, 8$

2	1
4	2
5	3
7	4
1	5
3	6
2	7
6	8

p

r

MERGE-SORT(A, p, r)

```
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )
```

2	1
4	2
5	3
7	4
1	5
3	6
2	7
6	8

p, r

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 7, 8 \rightarrow 8, 8$

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 7, 8$

2	1
4	2
5	3
7	4
1	5
3	6
2	7
6	8

p

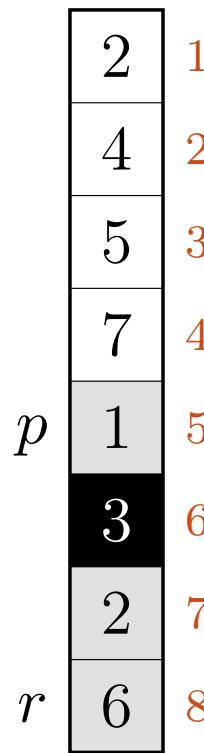
r

```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8 \rightarrow 5, 8$

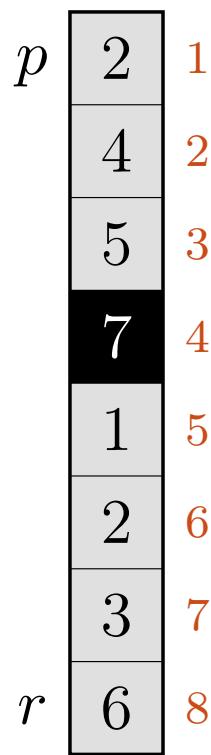


```

MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )

```

$p, r = 1, 8$



```
MERGE-SORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )
```

1	1
2	2
2	3
3	4
4	5
5	6
6	7
7	8

MERGE-SORT(A, p, r)

1 **if** $p < r$

2 $q = \lfloor (p + r)/2 \rfloor$

3 MERGE-SORT(A, p, q)

4 MERGE-SORT($A, q + 1, r$)

5 MERGE(A, p, q, r)

MERGE-SORT(A, p, r)

- 1 **if** $p < r$ $\Theta(1)$
- 2 $q = \lfloor (p + r)/2 \rfloor$
- 3 MERGE-SORT(A, p, q)
- 4 MERGE-SORT($A, q + 1, r$)
- 5 MERGE(A, p, q, r)

MERGE-SORT(A, p, r)

- 1 **if** $p < r$ $\Theta(1)$
- 2 $q = \lfloor (p + r)/2 \rfloor$ $\Theta(1)$
- 3 MERGE-SORT(A, p, q)
- 4 MERGE-SORT($A, q + 1, r$)
- 5 MERGE(A, p, q, r)

MERGE-SORT(A, p, r)

- 1 **if** $p < r$ $\Theta(1)$
- 2 $q = \lfloor (p + r)/2 \rfloor$ $\Theta(1)$
- 3 MERGE-SORT(A, p, q) $T(n/2)$
- 4 MERGE-SORT($A, q + 1, r$)
- 5 MERGE(A, p, q, r)

MERGE-SORT(A, p, r)

- 1 **if** $p < r$ $\Theta(1)$
- 2 $q = \lfloor (p + r)/2 \rfloor$ $\Theta(1)$
- 3 MERGE-SORT(A, p, q) $T(n/2)$
- 4 MERGE-SORT($A, q + 1, r$) $T(n/2)$
- 5 MERGE(A, p, q, r)

MERGE-SORT(A, p, r)

- 1 **if** $p < r$ $\Theta(1)$
- 2 $q = \lfloor (p + r)/2 \rfloor$ $\Theta(1)$
- 3 MERGE-SORT(A, p, q) $T(n/2)$
- 4 MERGE-SORT($A, q + 1, r$) $T(n/2)$
- 5 MERGE(A, p, q, r) $\Theta(n)$

MERGE-SORT(A, p, r)

- | | | | |
|---|---------------------------------|-------|-------------|
| 1 | if $p < r$ | | $\Theta(1)$ |
| 2 | $q = \lfloor (p + r)/2 \rfloor$ | | $\Theta(1)$ |
| 3 | MERGE-SORT(A, p, q) | | $T(n/2)$ |
| 4 | MERGE-SORT($A, q + 1, r$) | | $T(n/2)$ |
| 5 | MERGE(A, p, q, r) | | $\Theta(n)$ |

$$T(n) = 2T(n/2) + \Theta(n)$$

$$T(n) = \Theta(n \lg n)$$

Kjøretiden til Merge Sort, både i beste, verste og gjennomsnittlig tilfelle.

I motsetning til Insertion Sort, som har kvadratisk kjøretid.



$$T(n) = \Theta(n \lg n)$$