Sammenhengende Grafer



IN2010 – Uke 8

Litt eksamensoppgaver

(h) En prioritetskø med konstant tid på alle operasjoner ville gjort at Dijkstra hadde kjøretidskompleksitet $\mathcal{O}(|V| + |E|)$.

Sann!

Dijkstra - Kompleksitet

- Det er |V| på prioritetskøen
- Så vi gjør RemoveMin |V| ganger
- O(|V| * log(|V|))

- Vi besøker alle kanter (E) èn gang
- Så vi gjør DecreasePriority |E| ganger
- O(|E| * log(|V|))

ALGORITHM: DIJKSTRAS ALGORITME FOR KORTESTE STIER (T

Input: En vektet graf $G = (V, E)$ med vektfunksjon w og en startnc
Output: Et map `dist` som angir korteste vei fra s til alle noder i G

```
1  Procedure Dijkstra(G, s)
2      queue ← empty priority queue
3      dist ← empty map
4      for v ∈ V do
5          dist[v] ← ∞
6          Insert(queue, v) with priority ∞
7      dist[s] ← 0
8      DecreasePriority(queue, s, 0)
9
10     while queue is not empty do
11         u ← RemoveMin(queue)
12         for (u, v) ∈ E do
13             c ← dist[u] + w(u, v)
14             if c < dist[v] then
15                 dist[v] ← c
16                 DecreasePriority(queue, v, c)
17     return dist
```

Litt eksamensoppgaver

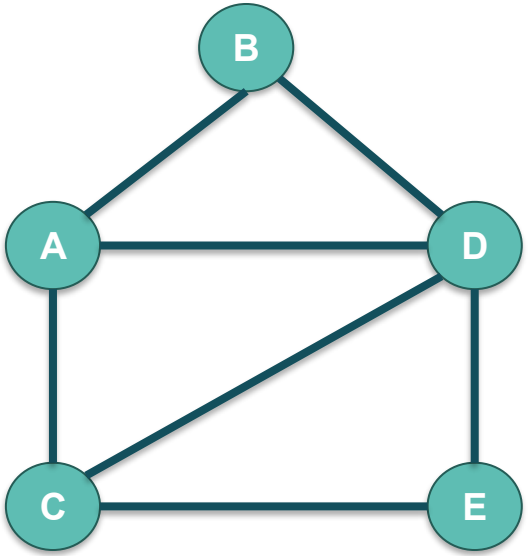
(h) En prioritetskø med konstant tid på alle operasjoner ville gjort at Dijkstra hadde kjøretidskompleksitet $\mathcal{O}(|V| + |E|)$.

- Det er |V| på prioritetskøen
- Så vi gjør RemoveMin |V| ganger
- O(|V| * 1) = O(|V|)

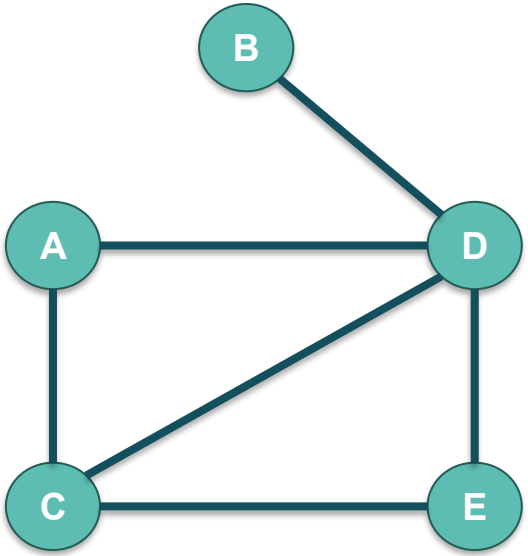
- Vi besøker alle kanter (E) èn gang
- Så vi gjør DecreasePriority |E| ganger
- O(|E| * 1) = O(|E|)

Sammenhengende grafer

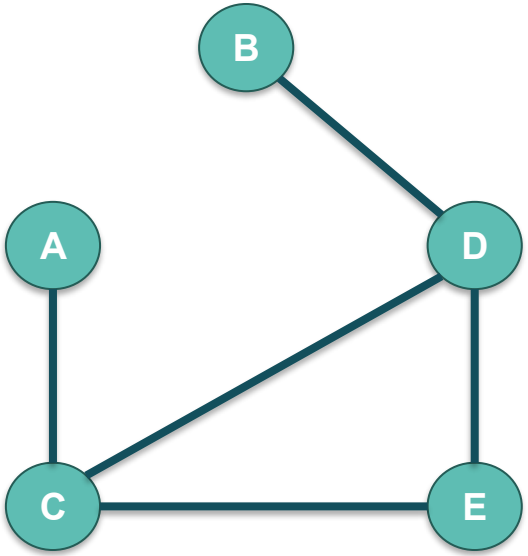
Sammenhengende



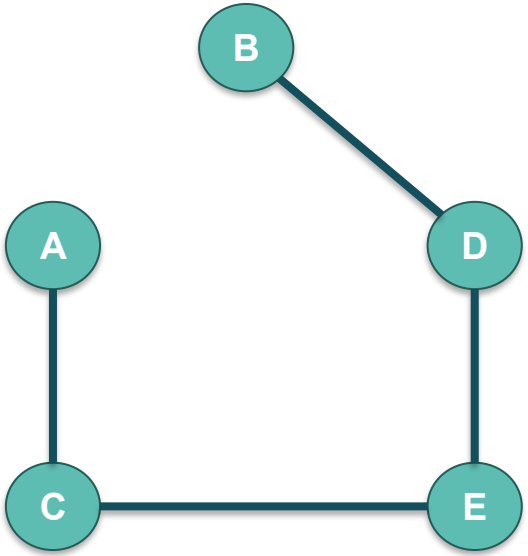
Fortsatt sammenhengende



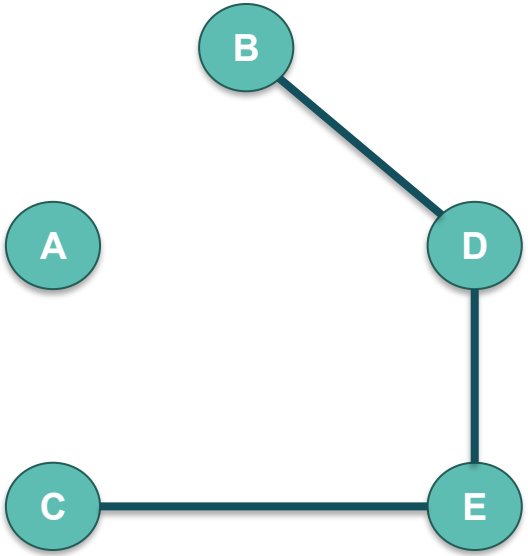
Fortsatt sammenhengende



Fortsatt sammenhengende



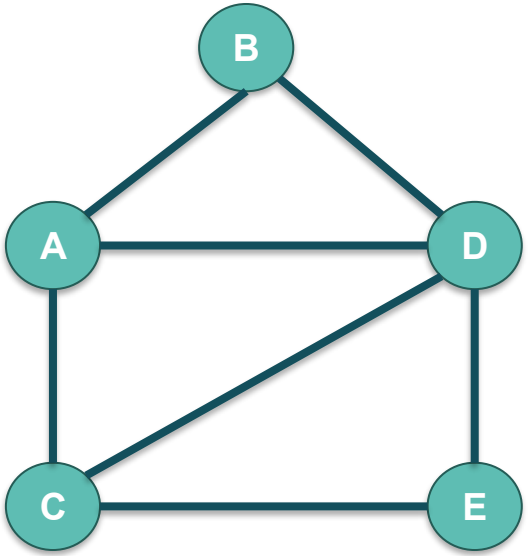
IKKE sammenhengende



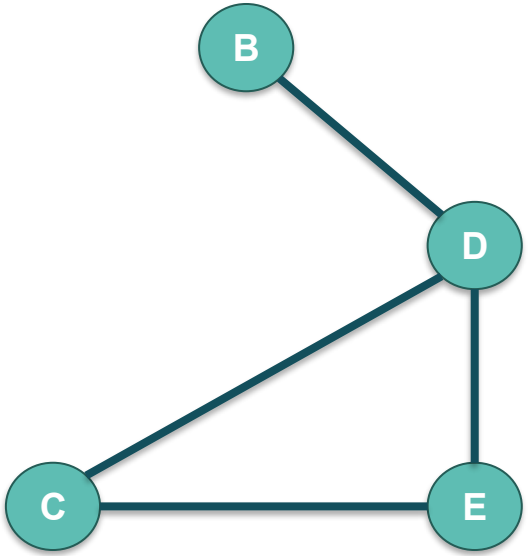
Sammenhengde graf

For at en graf skal være sammenhengende må det finnes en sti mellom alle noder

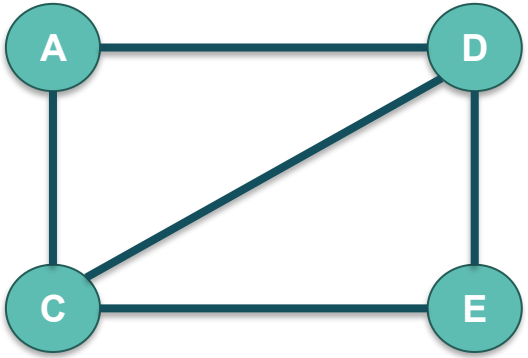
2-Sammenhengende



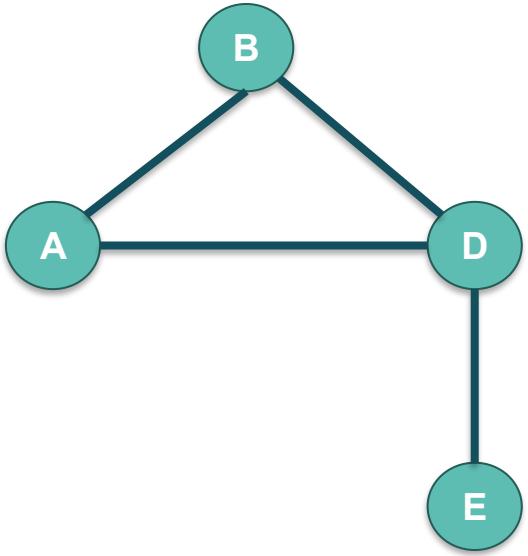
2-Sammenhengende



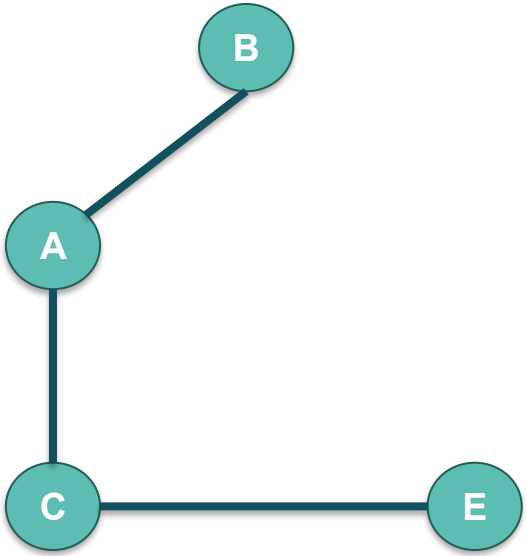
2-Sammenhengende



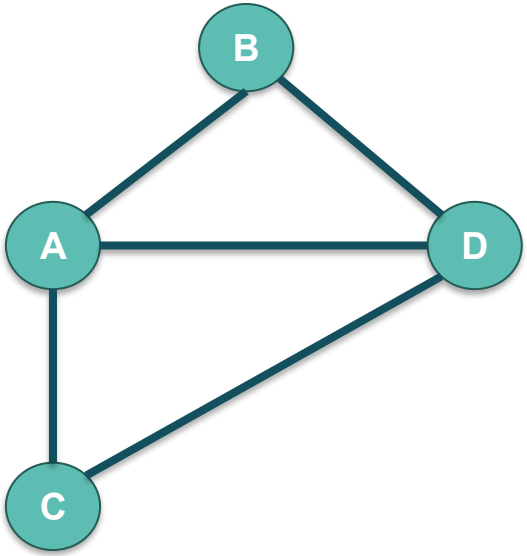
2-Sammenhengende



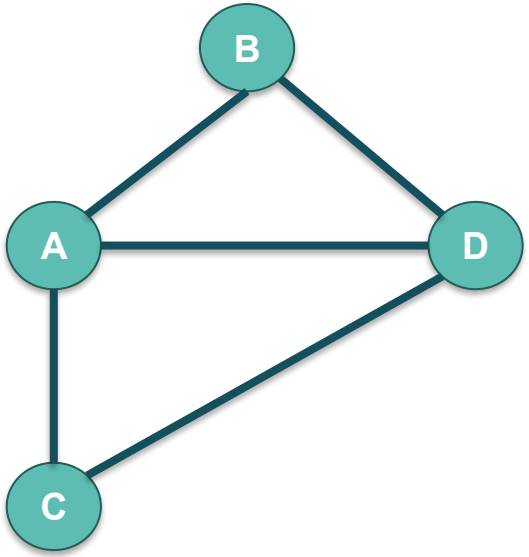
2-Sammenhengende



2-Sammenhengende



2-Sammenhengende



2-sammenhengende dersom grafen fortsatt er sammenhengende, selv om du fjerner en hvilken som helst node fra grafen.

Altså:
Det finnes minst to distinkte stier mellom alle par av noder

2-sammenhengende grafer

(d) Det siste året har det vært to hendelser der det har vært full stopp ved en holdeplass. Det ledet til at reisende ikke kunne nå destinasjonen sin. Bystyret har vedtatt en garanti: dersom det blir full stopp ved en holdeplass, skal det fremdeles være mulig for reisende å nå sin destinasjon via en alternativ rute. Garantien gjelder for alle holdeplasser.

Forklar hvordan vi kan bruke en algoritme kjent fra pensum til å sjekke om kollektivtilbudet leverer denne garantien.

2-sammenhengende grafer

Aller enkleste løsningen?

Vi fjerner en node fra G, sjekker om den fortsatt er sammenhengende, og gjør dette for alle noder.

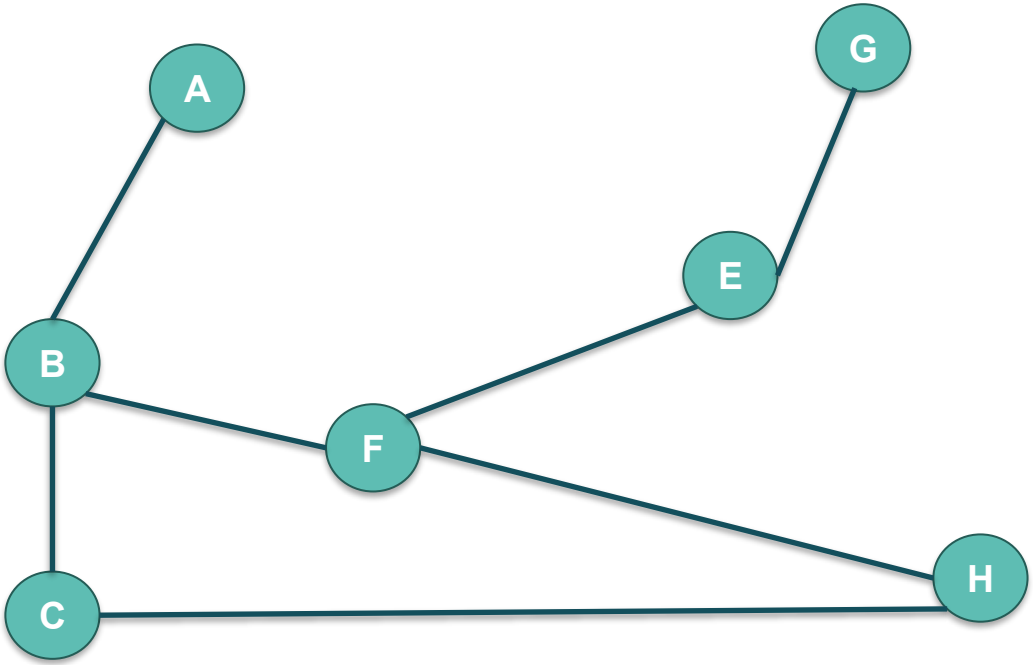
…Dette tar veldig lang tid

Spesifikt $O(|V^3|)$

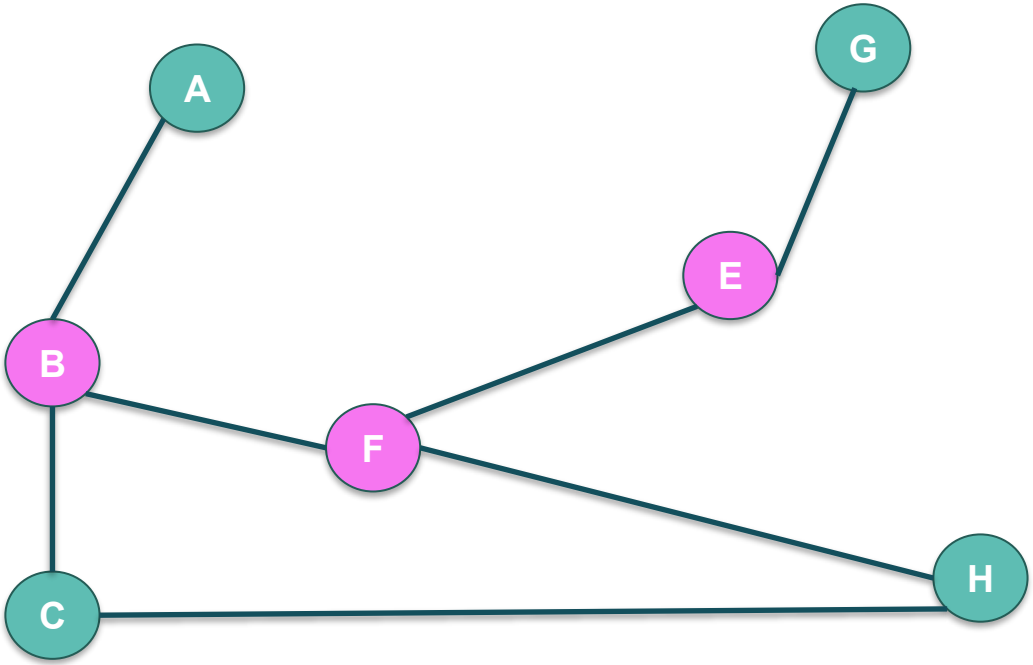
Separasjonsnoder

Bedre løsning?
Vi sjekker om en graf har 1 eller flere **separasjonsnoder**

Separasjonsnoder



Separasjonsnoder



Separasjonsnoder

- En node u er en separasjonsnode hvis:
  - u er rot i T og har mer enn én utgående kant i T, eller
  - u ikke er roten, og har et barn v, slik at ingen etterkommere av v har en tilbakekant til en forgjenger av u (ikke inkludert u selv)

Separasjonsnoder

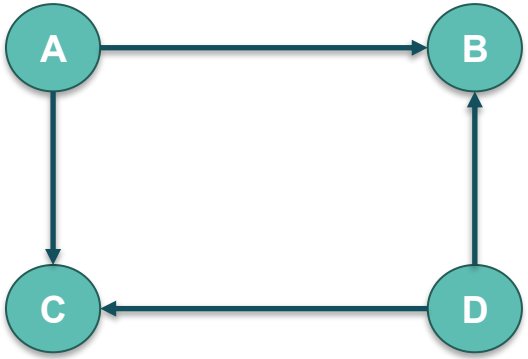
Sterkt sammenhengende komponenter

Sterkt sammenhengende grafer

En rettet graf kan være både sterkt sammenhengende og svakt sammenhengende

Sterkt sammenhengende grafer

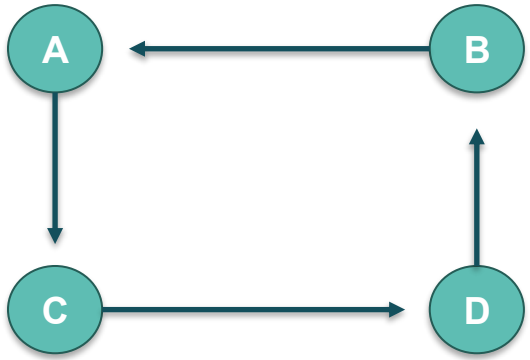
En rettet graf kan være både sterkt sammenhengende og svakt sammenhengende



Sterkt sammenhengende grafer

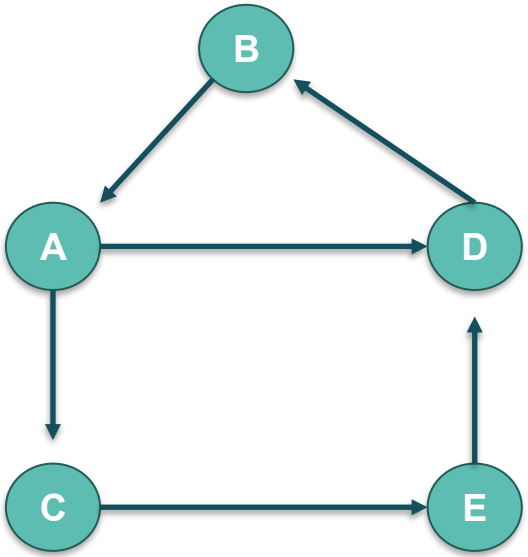
En rettet graf kan være både sterkt sammenhengende og svakt sammenhengende

Sterkt sammenhengende grafer er som sykler



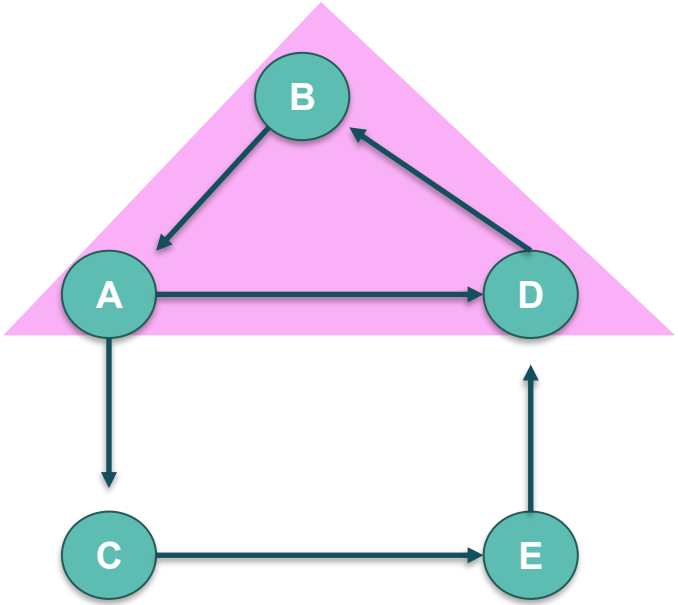
Sterkt sammenhengende komponent

SUBDEL av en graf som er sterkt sammenhengende



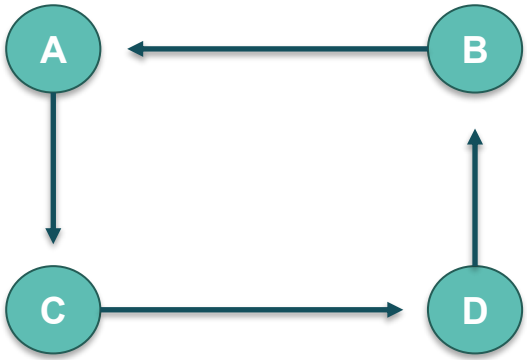
Sterkt sammenhengende komponent

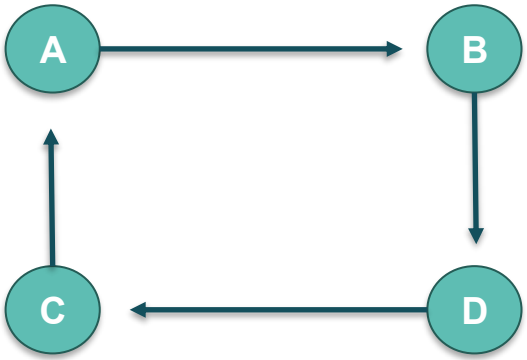
SUBDEL av en graf som er sterkt sammenhengende



Sterkt sammenhengende komponent

Trikset for å finne sterkt sammenhengende komponenter er å reversere grafer

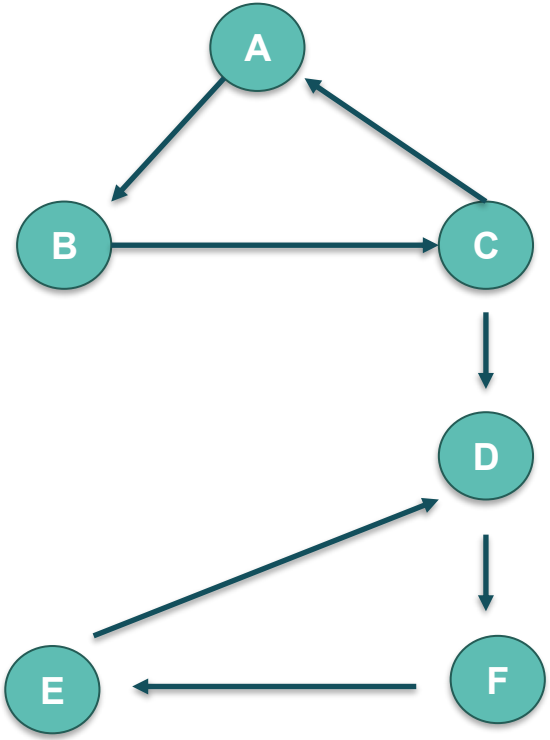




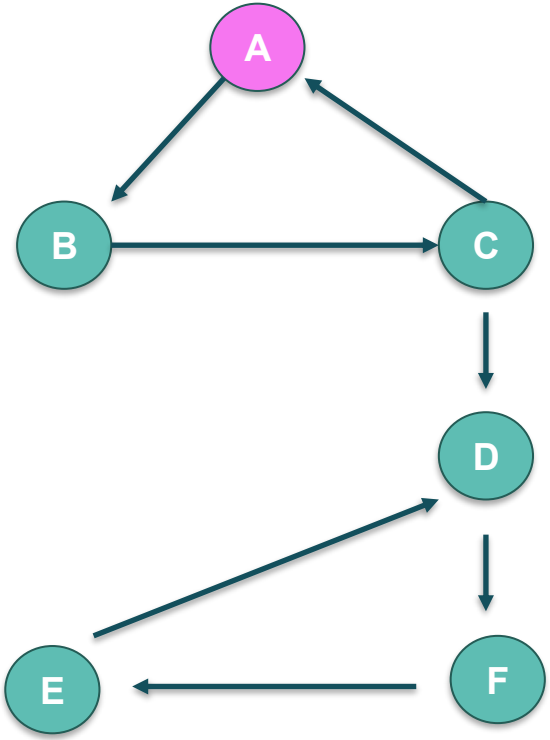
Kosaraju

1) DFS søk: Når du kommer til en node med null flere utgående kanter, legger du den til på en stack
2) Reverser grafen
3) DFS søk på den reverserte grafen: Start med noden på toppen av stacken.
4) Har DFS søket tatt deg tilbake til en node du allerede har besøkt? Da er det en komponent

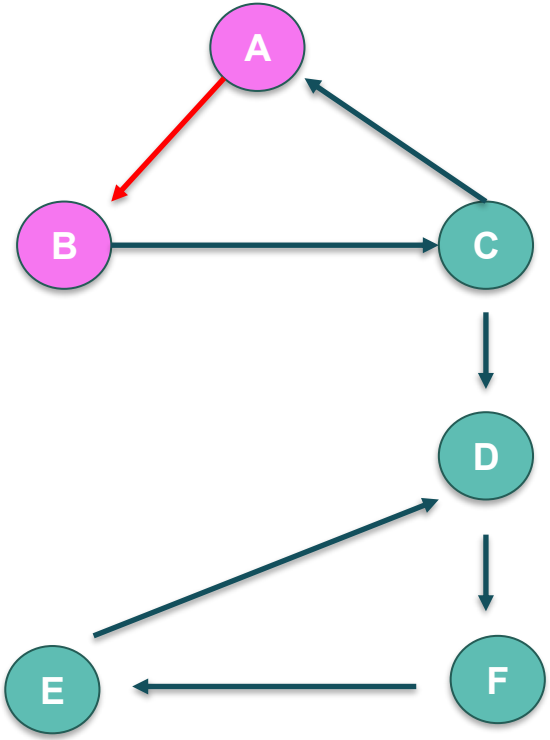
Stack



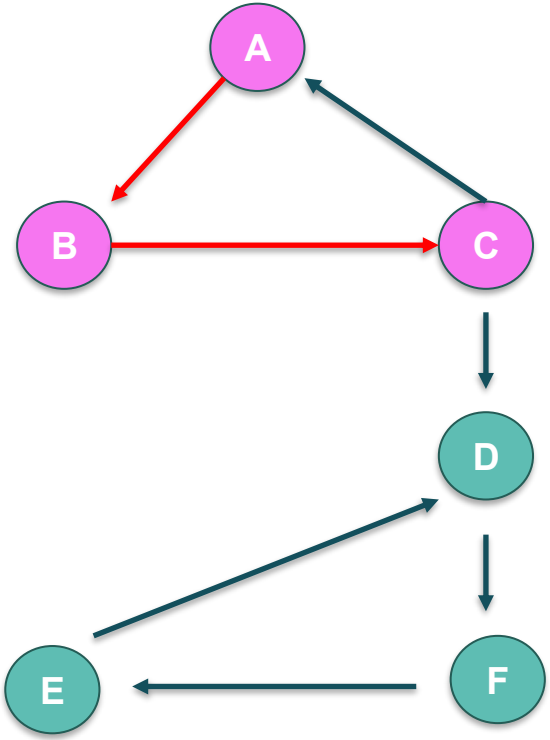
Stack



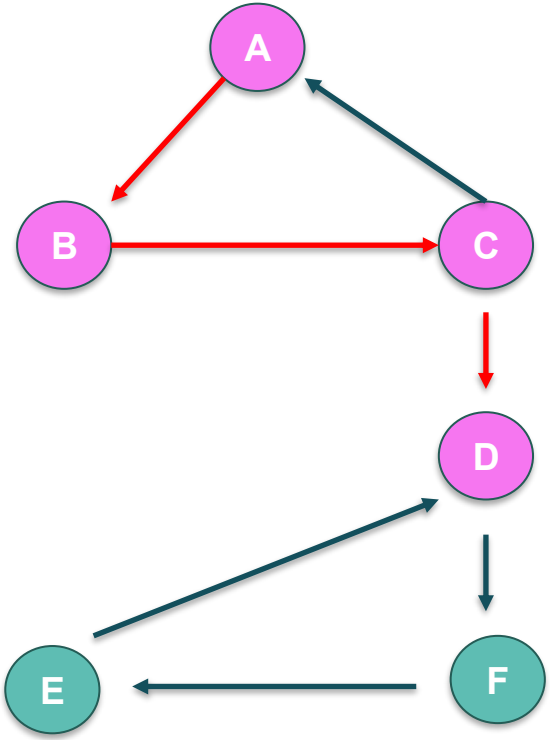
Stack



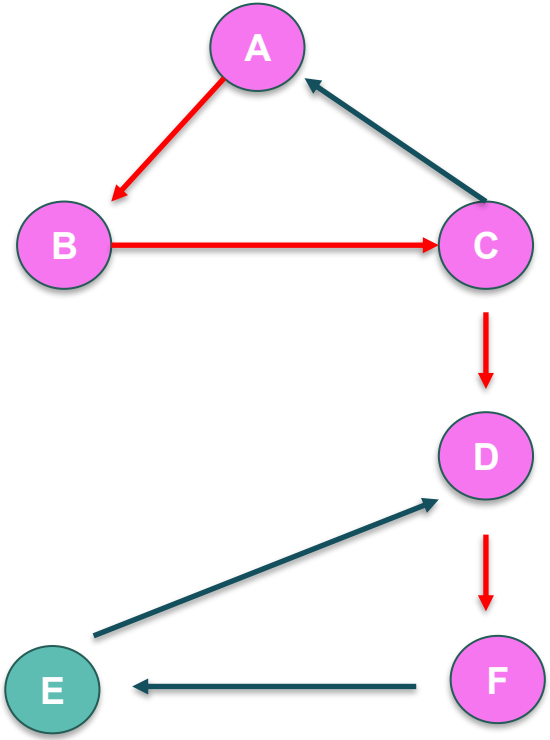
Stack



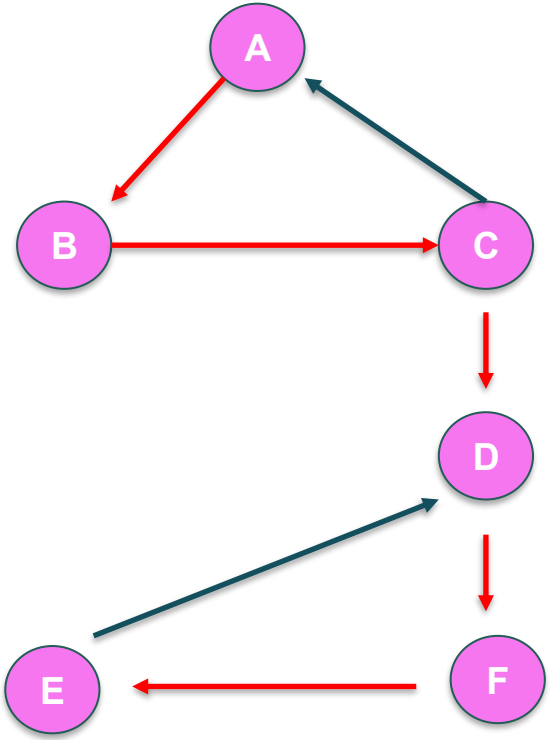
Stack



Stack

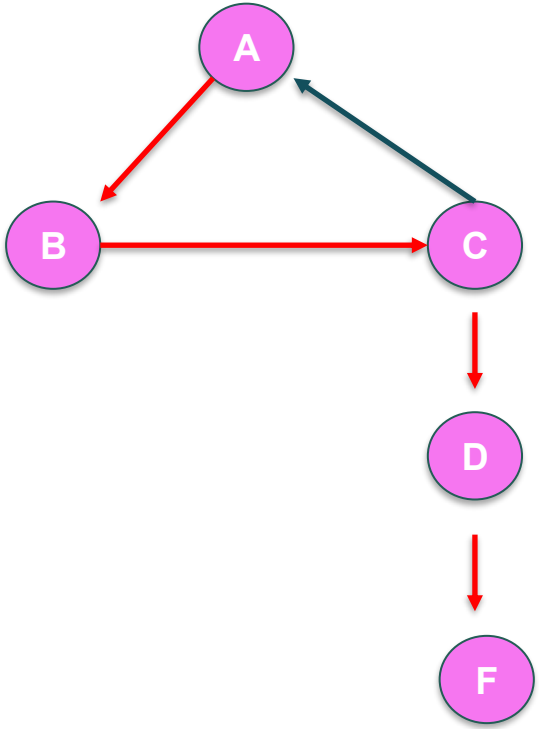


Stack



Stack

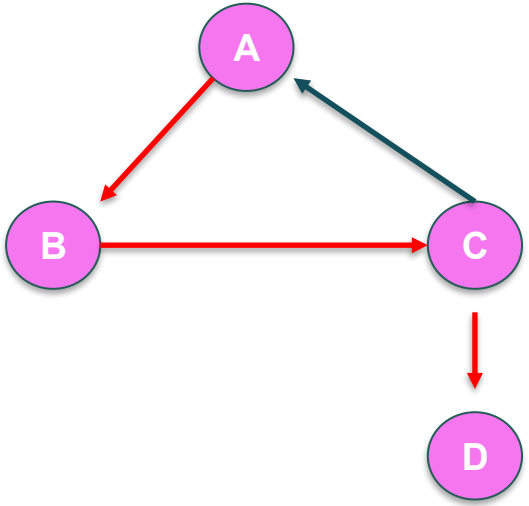
E



Stack

F

E

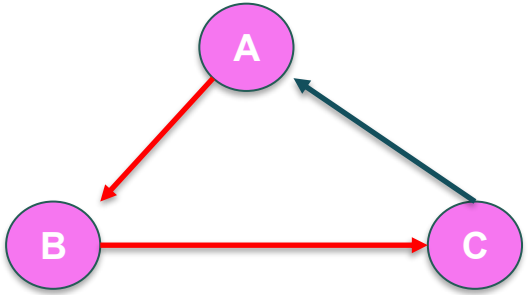


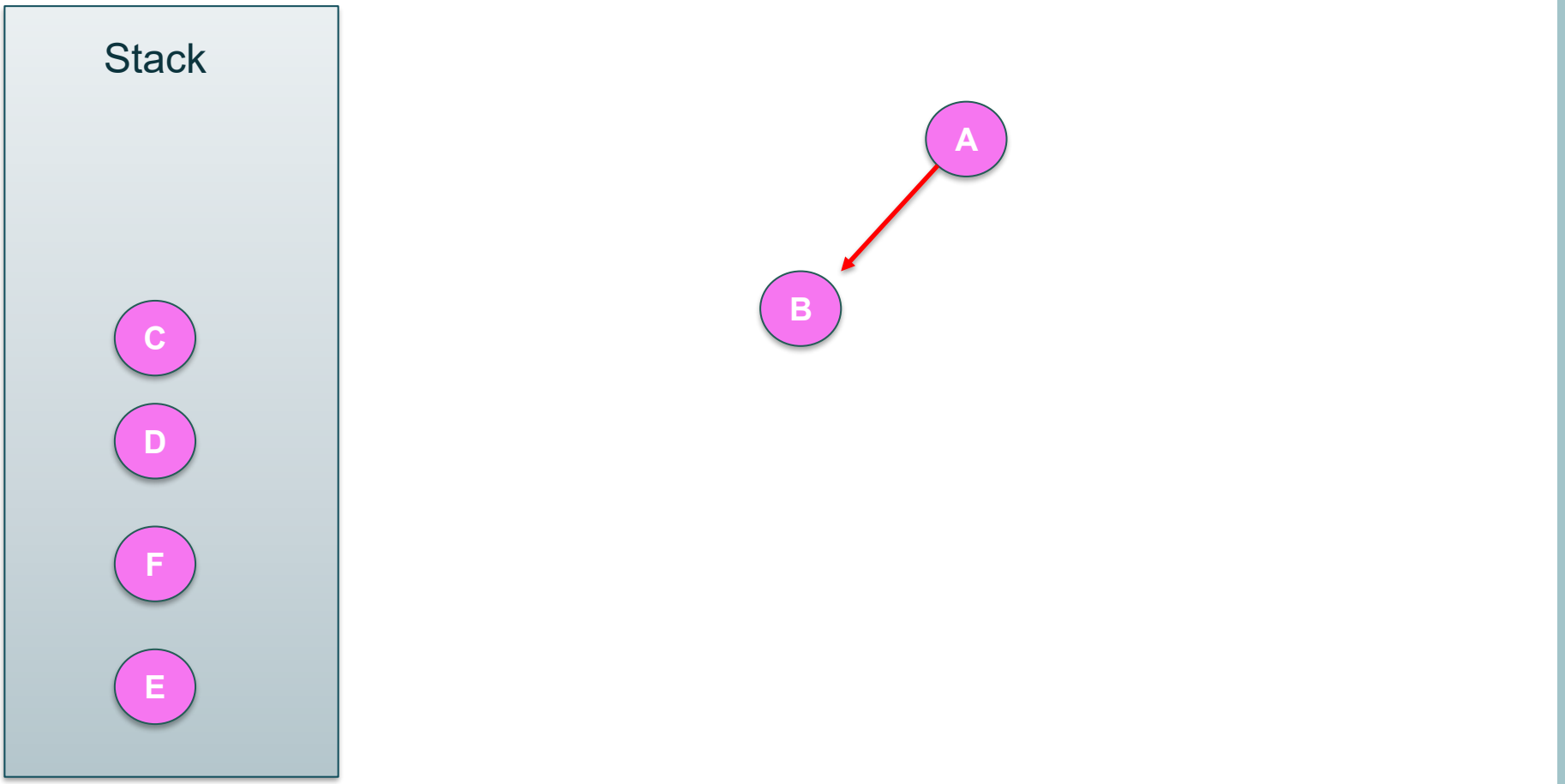
Stack

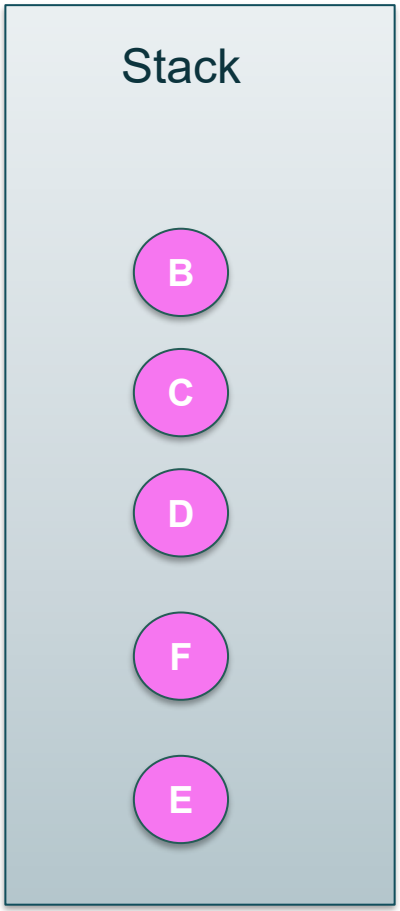
D

F

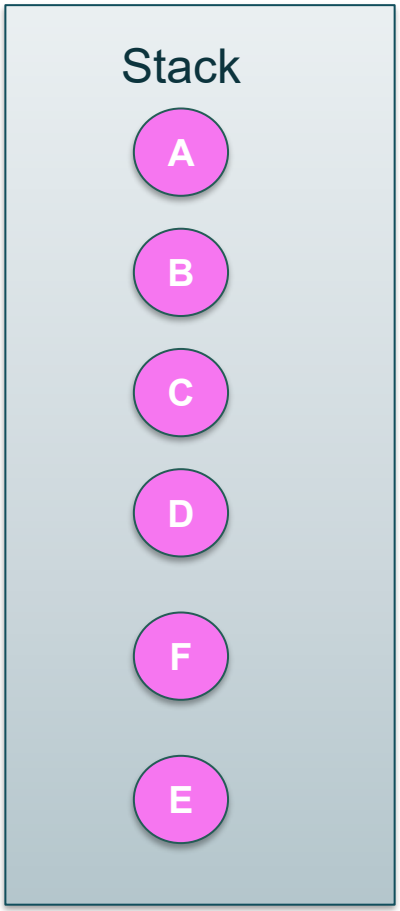
E

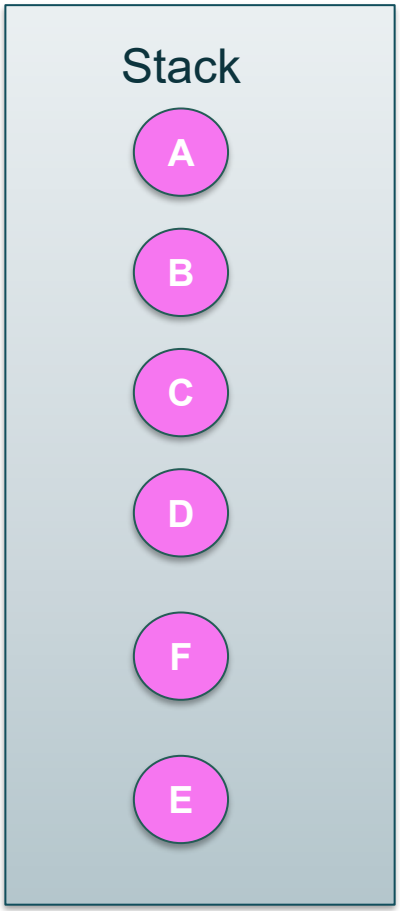


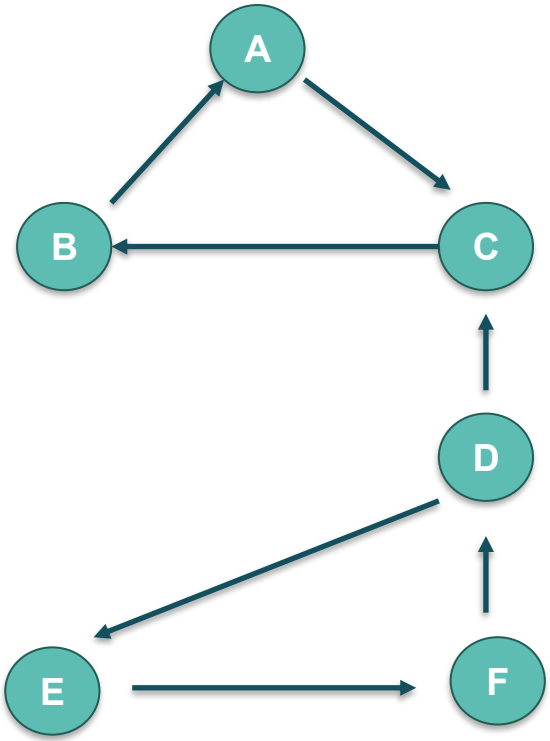

Stack
C
D
F
E
A
B



A


Stack
A
B
C
D
F
E


Stack
A
B
C
D
F
E


A
B
C
D
E
F

Stack

B

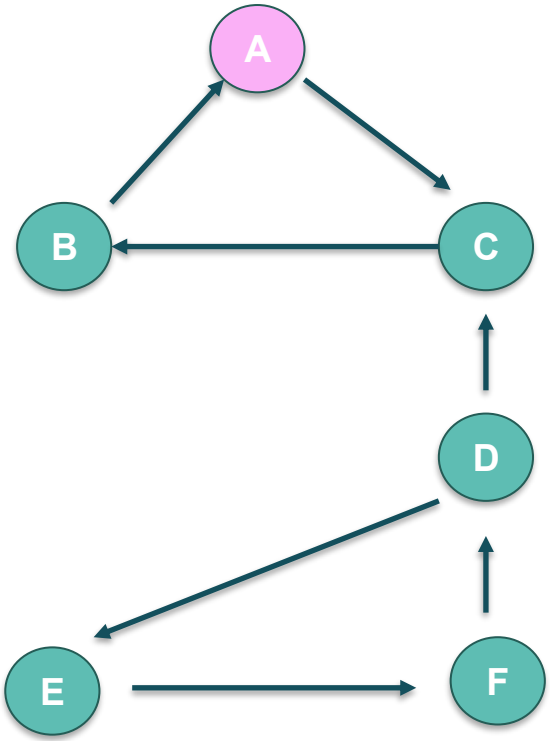
C

D

F

E

A



Stack

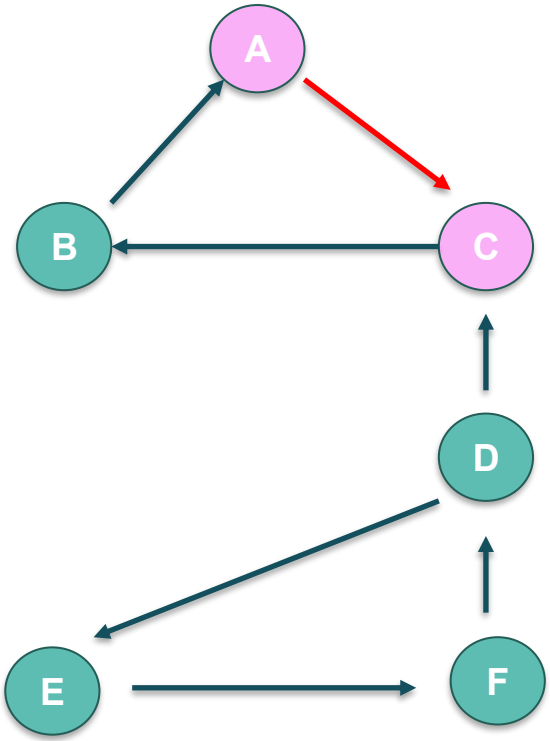
B

C

D

F

E



Stack

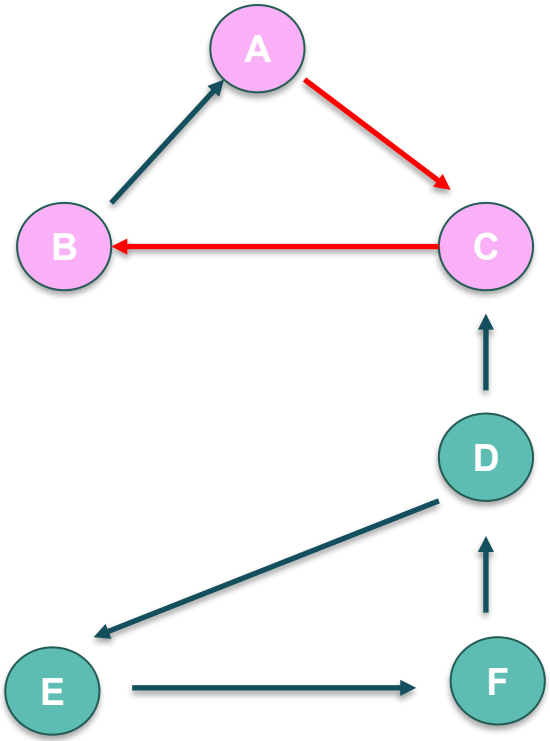
B

C

D

F

E



Stack

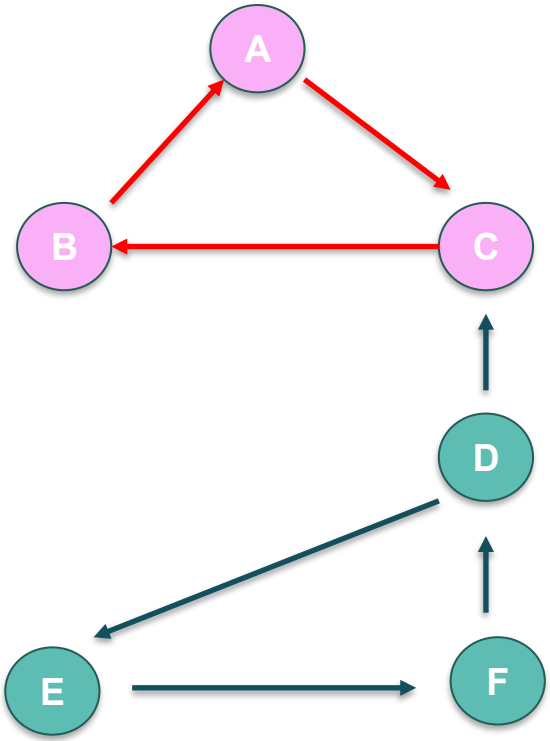
B

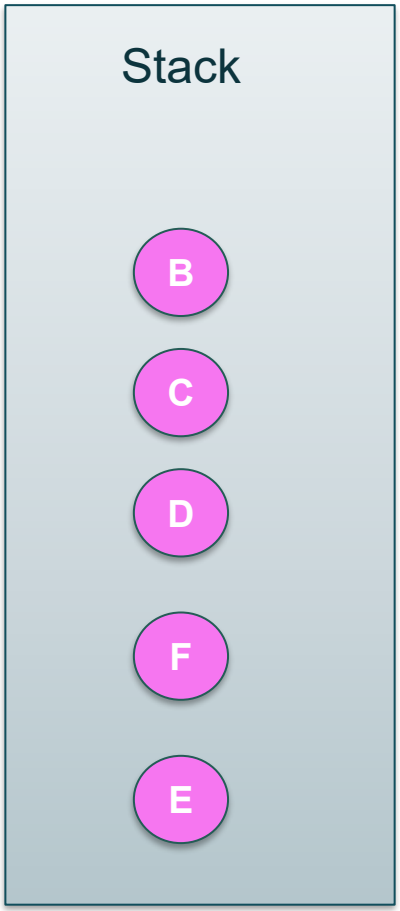
C

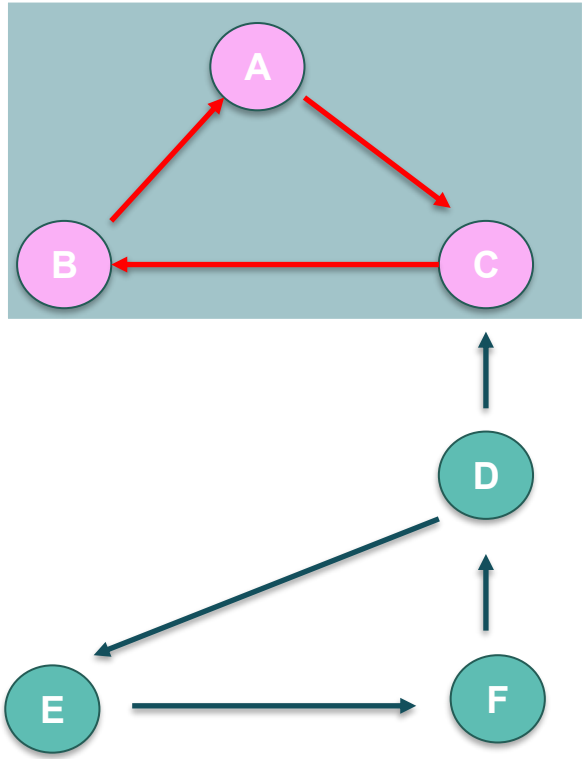
D

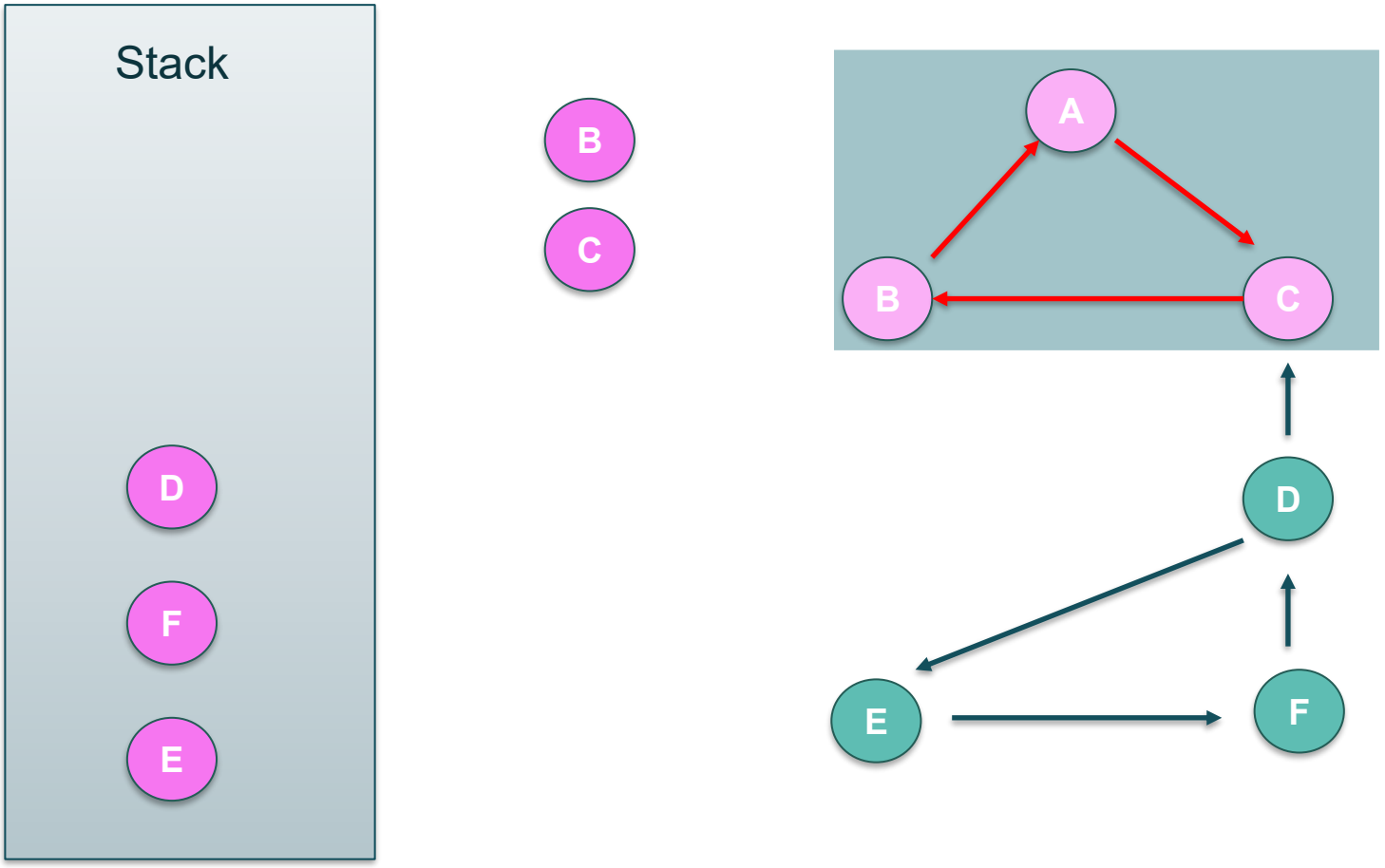
F

E




Stack
B
C
D
F
E


A
B
C
D
E
F

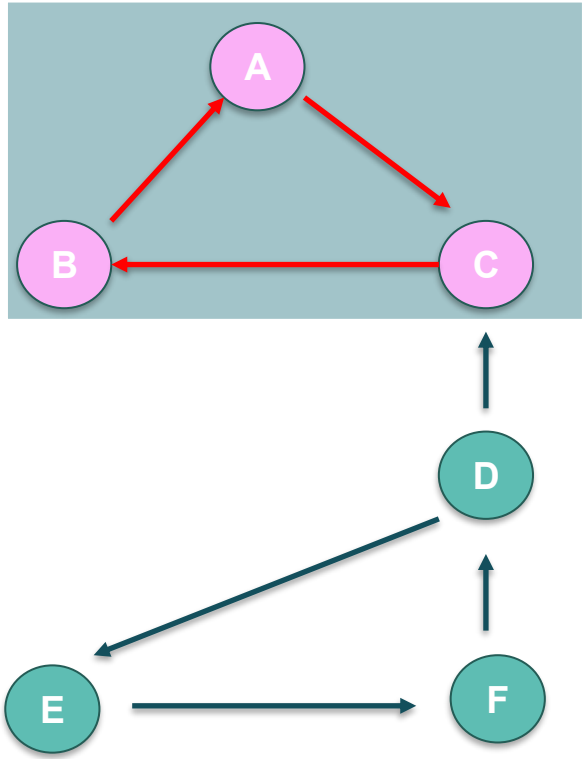

Stack
B
C
D
F
E
A
B
C
D
E
F

Stack

F

E

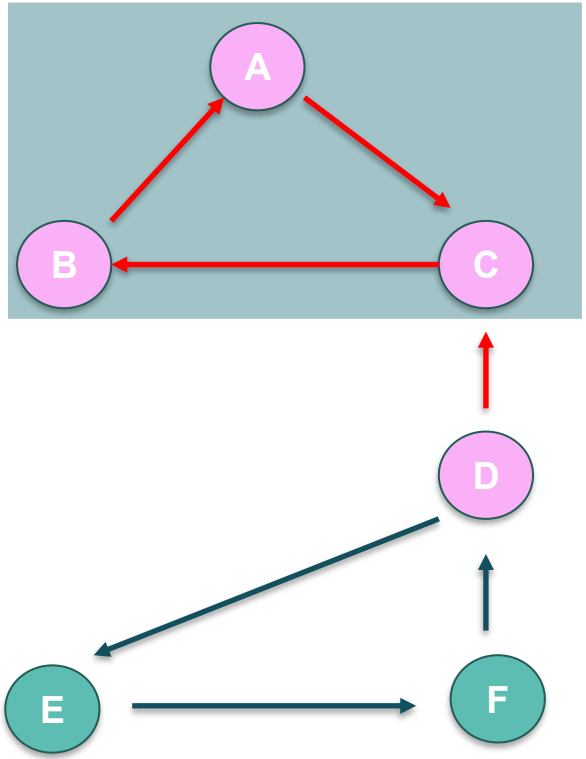
D



Stack

F

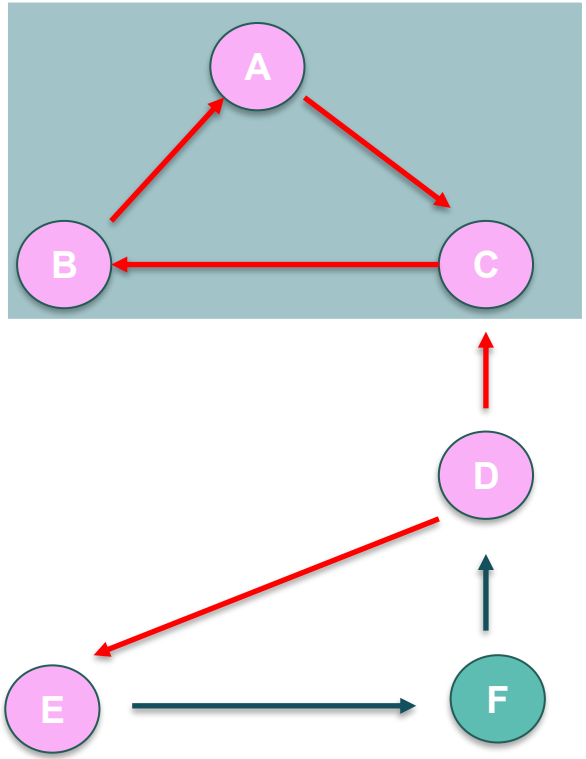
E



Stack

F

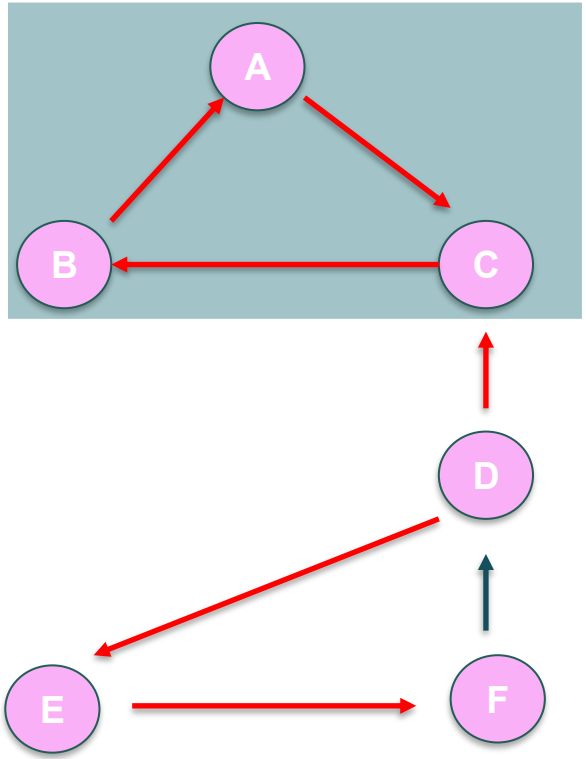
E



Stack

F

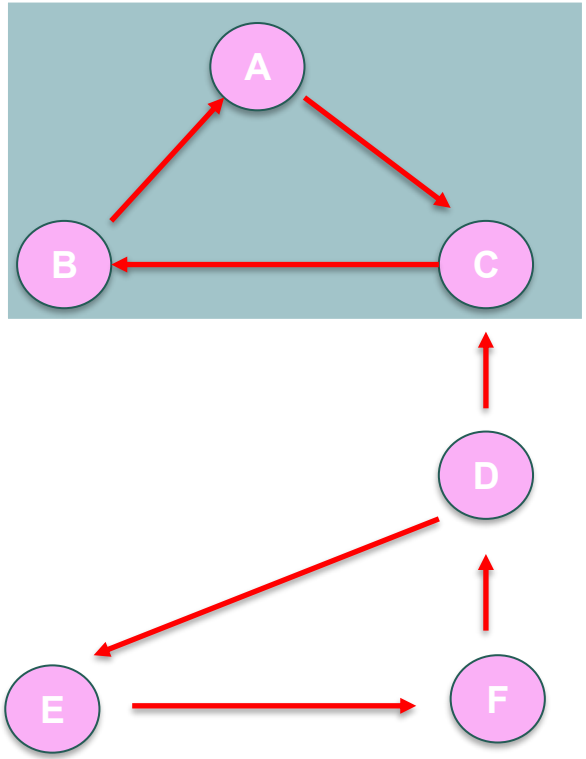
E

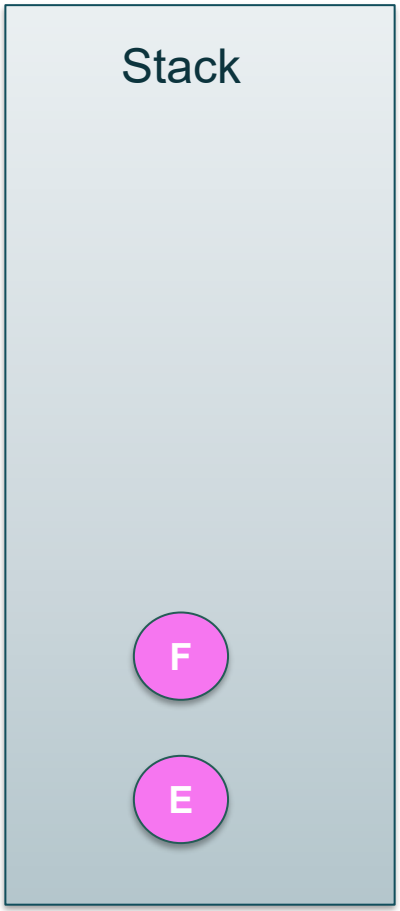


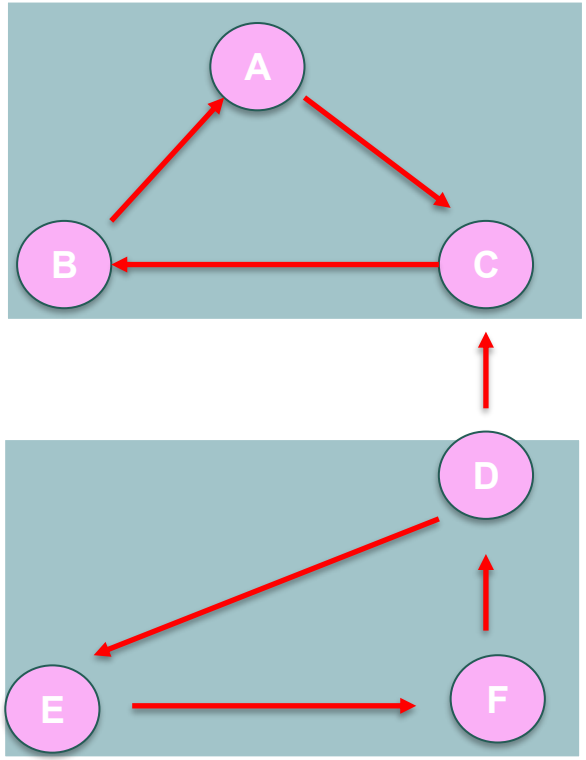
Stack

F

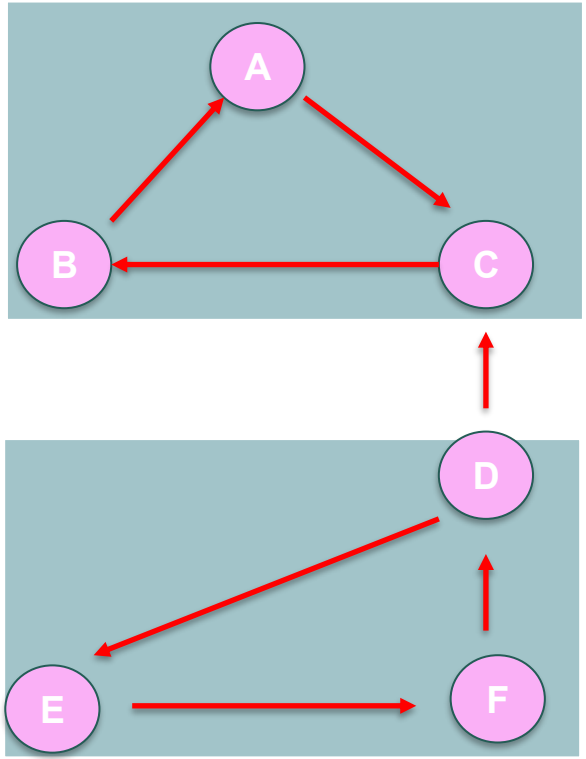
E




Stack
F
E


A
B
C
D
E
F

Stack



Oppgaver + oblighjelp