

Grafer og hashing

Benjamin Bjørnseth

Informasjon

- Studasser
- algdat@idi.ntnu.no

Program

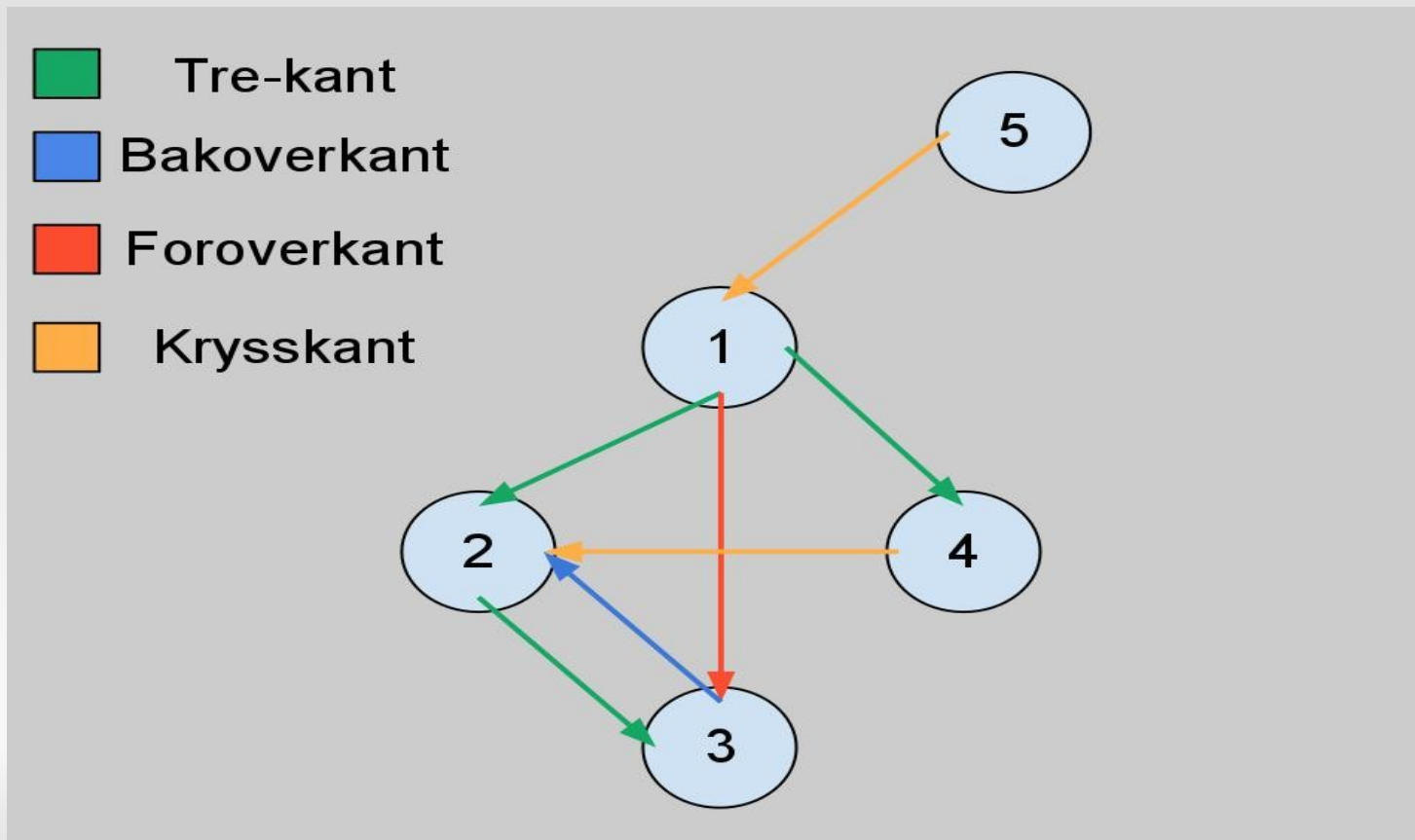
- Presentasjon av øving 2
- Grafer og traverseringsalgoritmer
 - BFS, DFS
- Hashing
- Gjennomgang av øving 1

Øving 2 - Teori

- Omhandler grafer, trær og traversering
- Oppgave 1: Diverse
- Oppgave 2: Pre- in- og postorder
- Oppgave 3: Ytelse for binære (søke)trær

Øving 2 - Teori

- Oppgave 4: DFS kant-typer



Øving 2 - Praksis

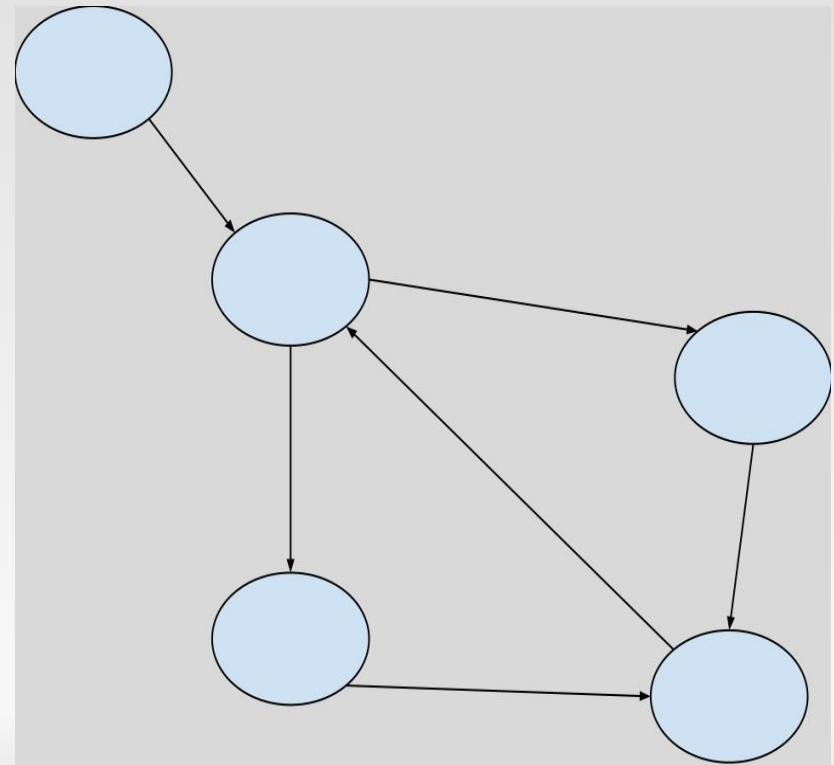
- Finn Ratatosk!
- Søk i tre

Øving 2 - Praksis

```
class Node:
    barn = None
    ratatosk = None
    nesteBarn = None # bare til bruk i DFS
    def __init__(self):
        self.barn = []
        self.ratatosk = False
        self.nesteBarn = 0
```

Grafer

- En mengde noder og en mengde kanter.
 - $G = (V, E)$
- Kan ha mer informasjon
 - Vekt
 - Flyt
- Forskjellige typer
- Terminologi
 - Naboer, barn, sti etc.

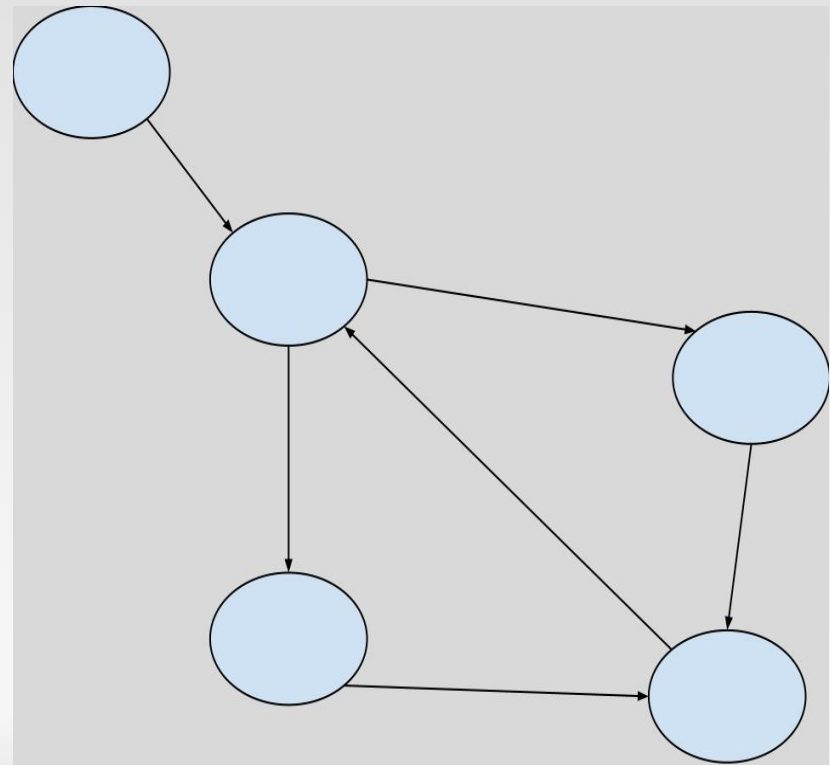


Hvorfor grafer?

- Problemer kan naturlig repræsenteres som grafer

```
for i in range(10):  
    if i < 5:  
        print 'x'  
    else:  
        print 'y'
```

=>



Hvorfor grafer?

- Grundig studert datastruktur
- Klarer du modellere problemet ditt rett, finnes kanskje løsningen allerede

Grafbruk - eksempler

- Optimale ruter for maksimering av godstransport
 - Beste måten å route pakker i internett
 - Billigste måten å legge vei mellom byer
 - Avhengighetsløsning
 - +++
-
- Kommer til å lære om alt dette i løpet av faget.

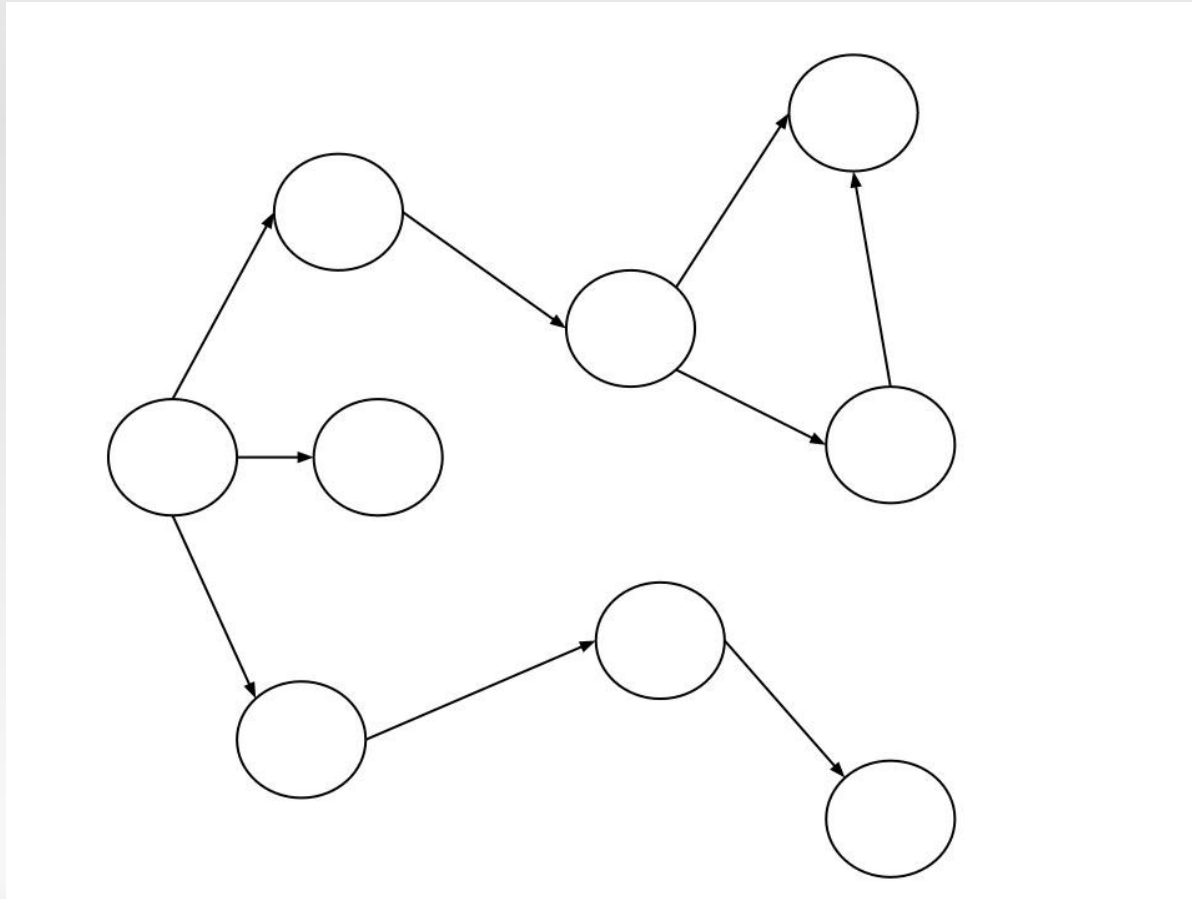
I dag: Graftraversering

- Søk i grafen
 - Øving 2
- Oppdag struktur av grafen
- Byggestein i andre algoritmer
 - Tofarging av graf

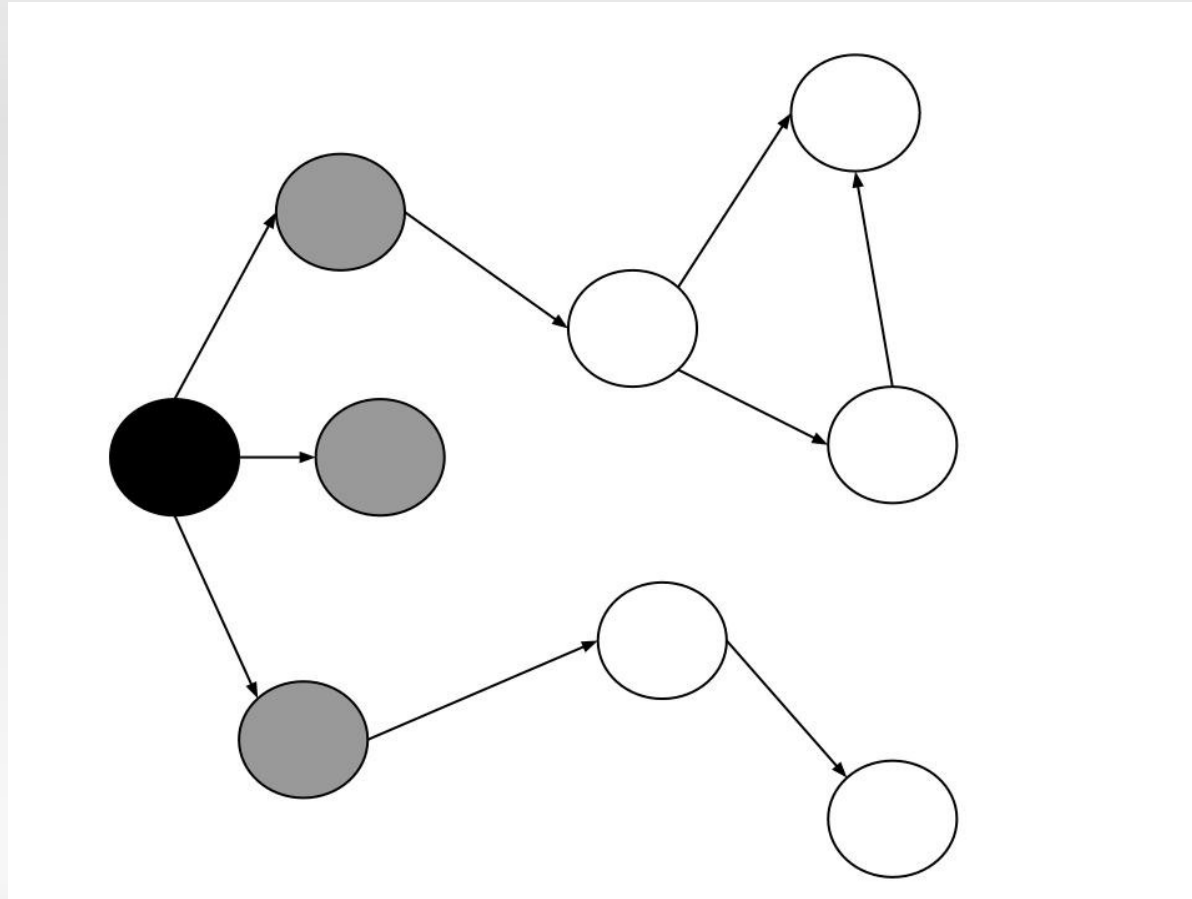
Graftraversering

- Vi lærer to algoritmer:
 - Bredde-først søk
 - Dybde-først søk

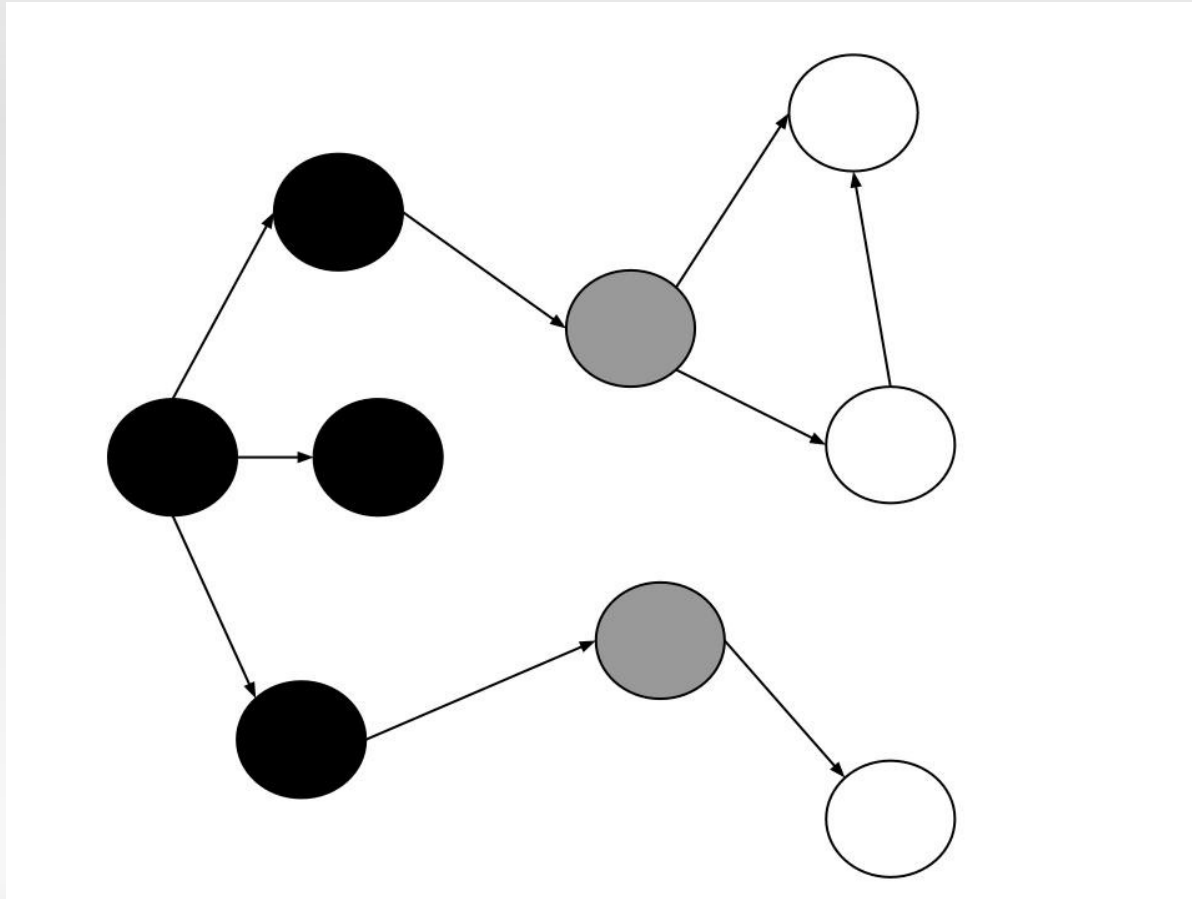
Fremgangsmåte: Bredde først



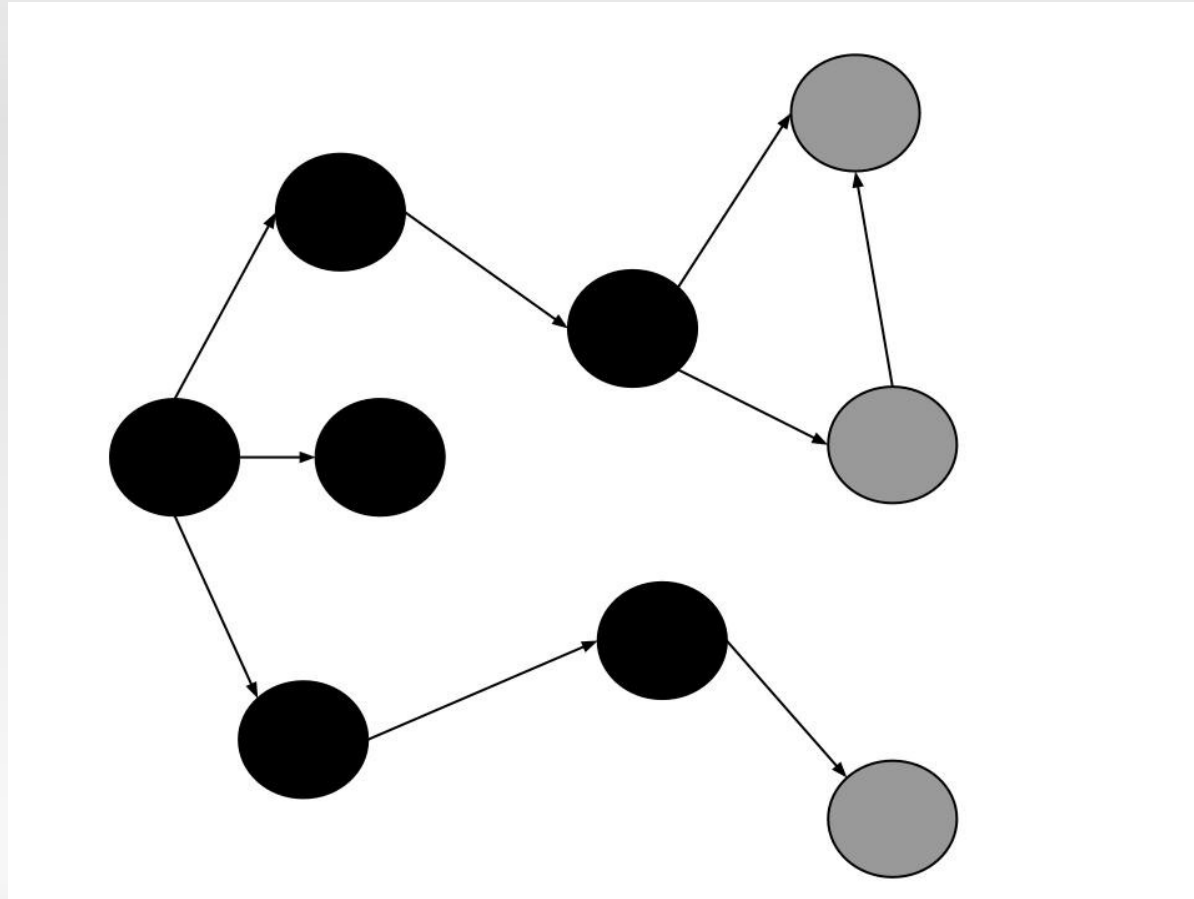
Fremgangsmåte: Bredde først



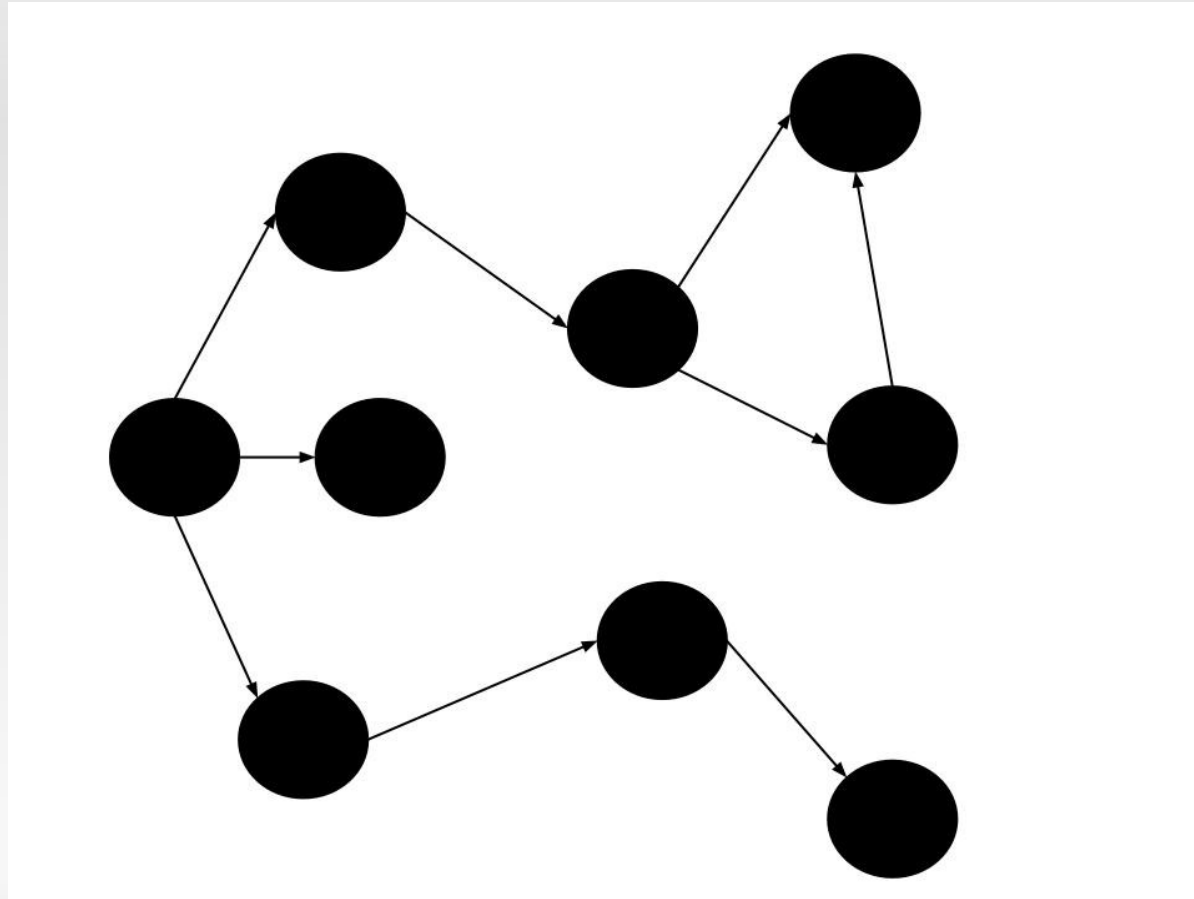
Fremgangsmåte: Bredde først



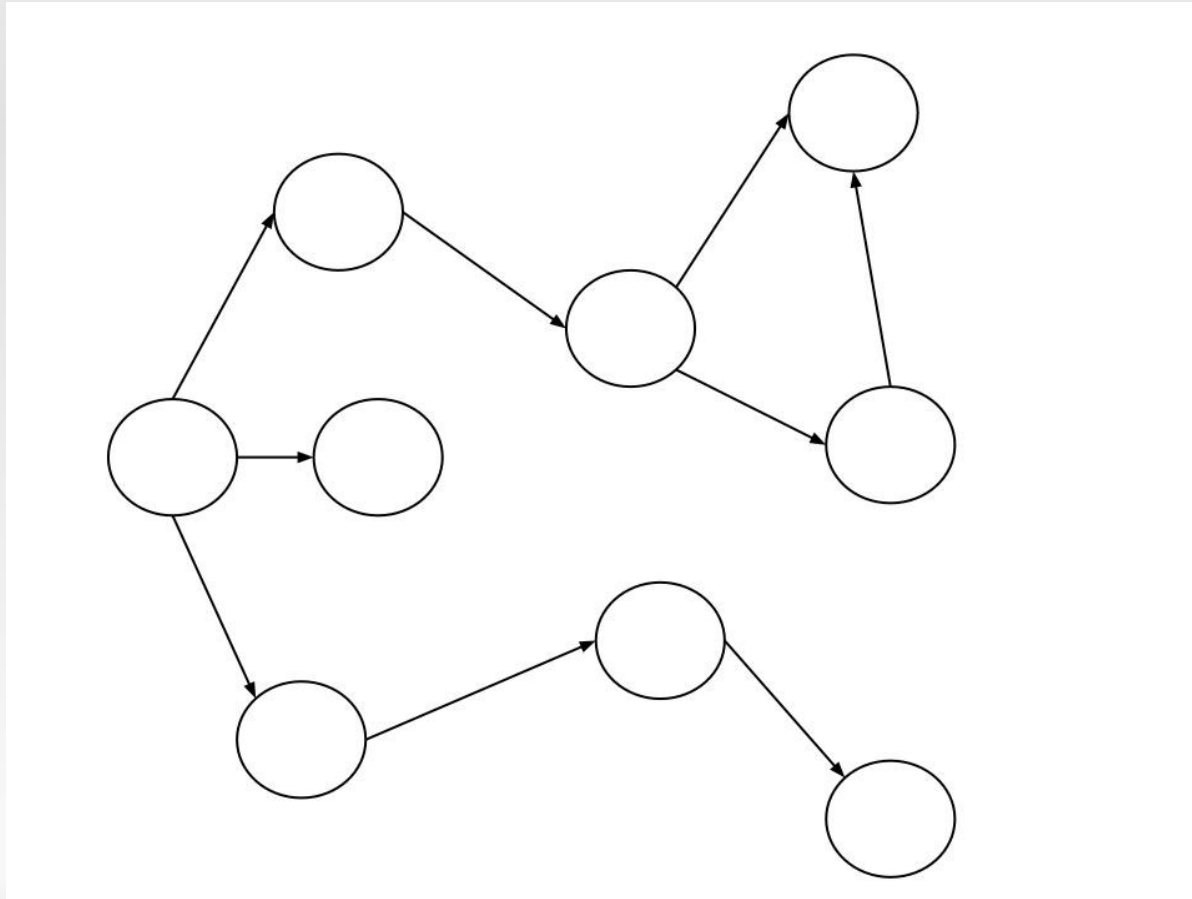
Fremgangsmåte: Bredde først



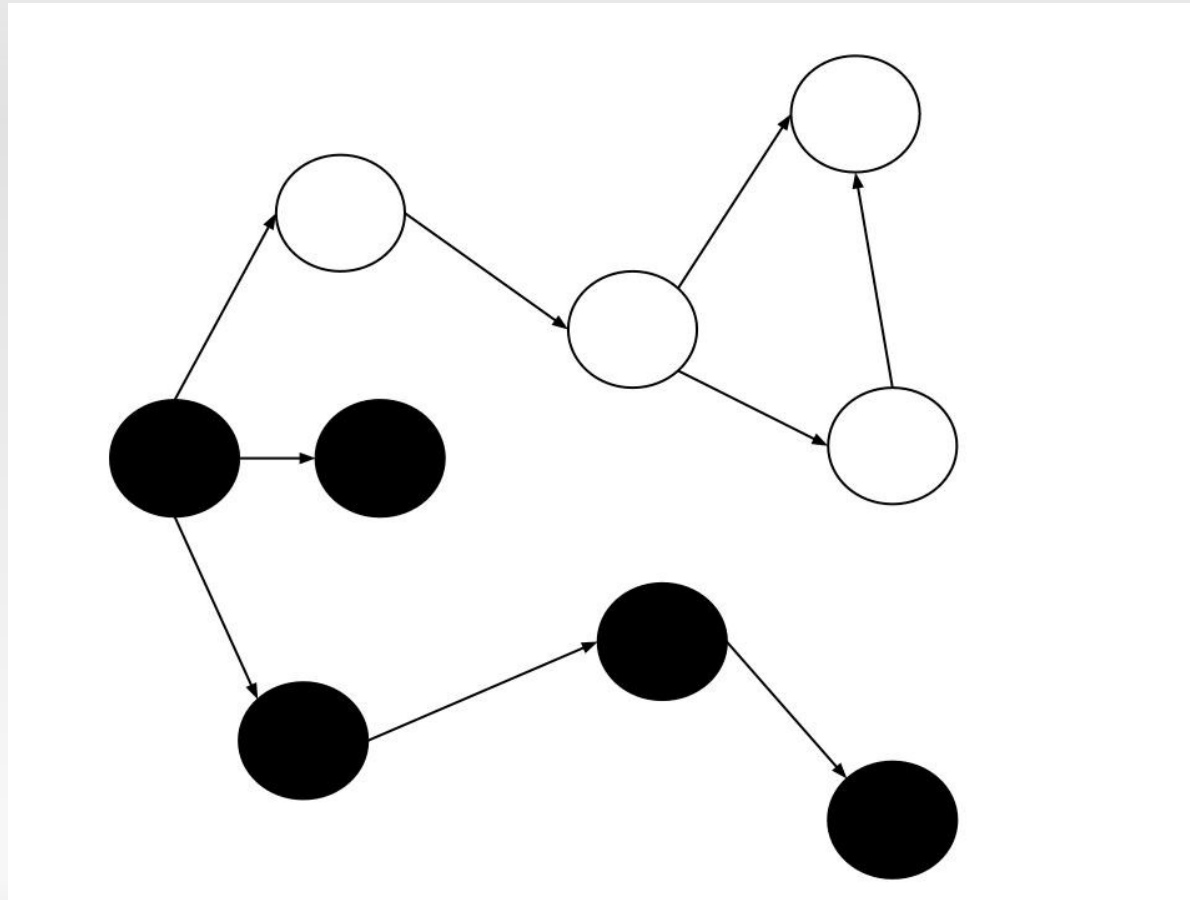
Fremgangsmåte: Bredde først



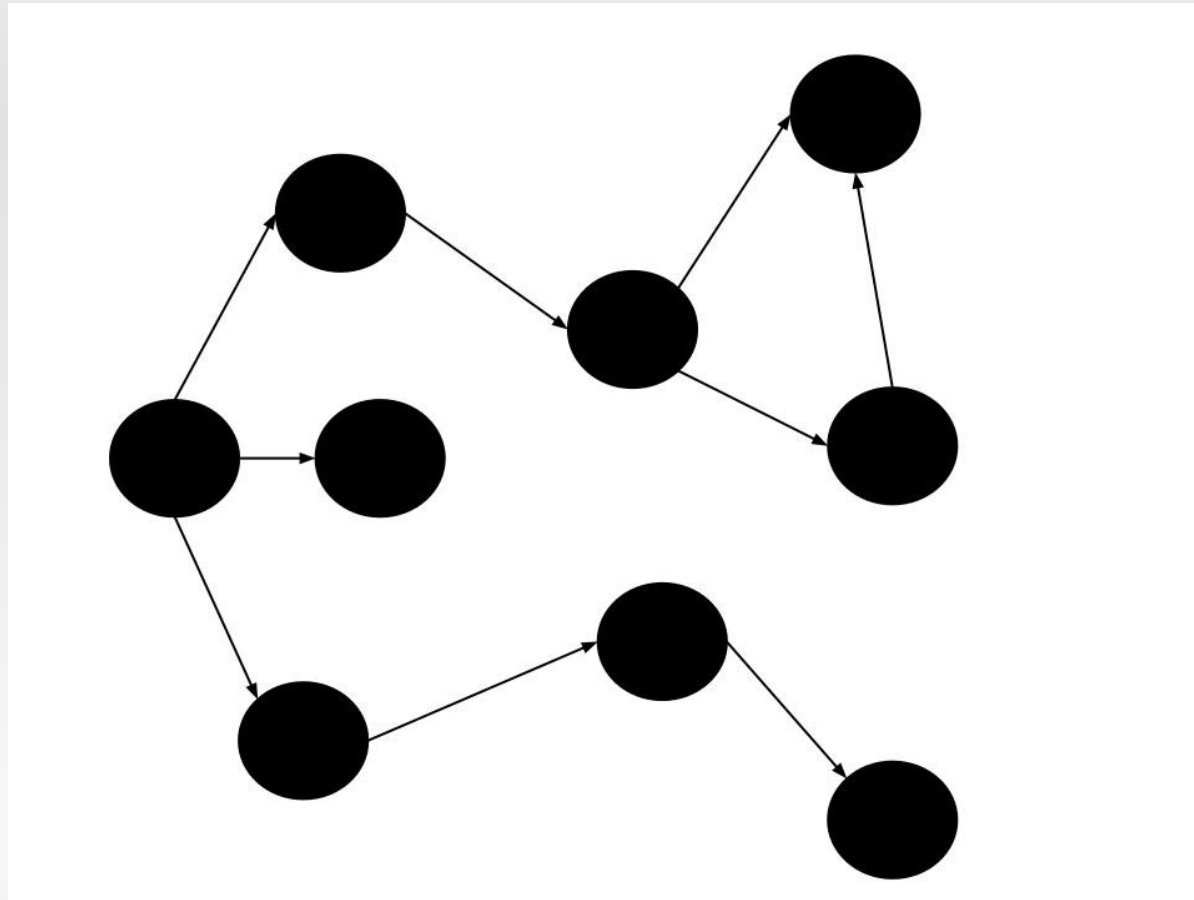
Fremgangsmåte: Dybde først



Fremgangsmåte: Dybde først

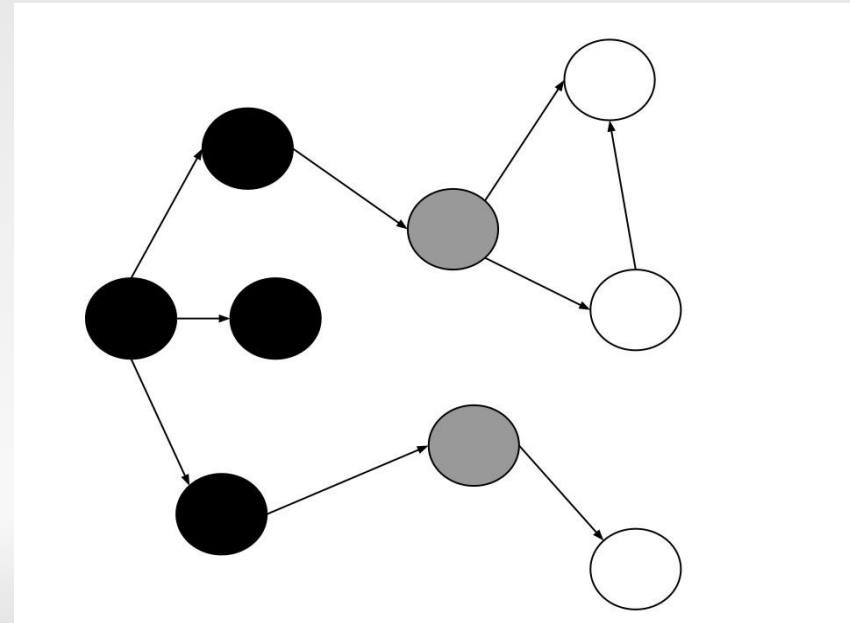


Fremgangsmåte: Dybde først



Intermission: Felles terminologi

- Konseptuell fargelegging av noder
 - Oppdaget node: Hvit
 - Oppdaget, ikke ferdig prosessert node: Grå
 - Ferdig prosessert node: Svart



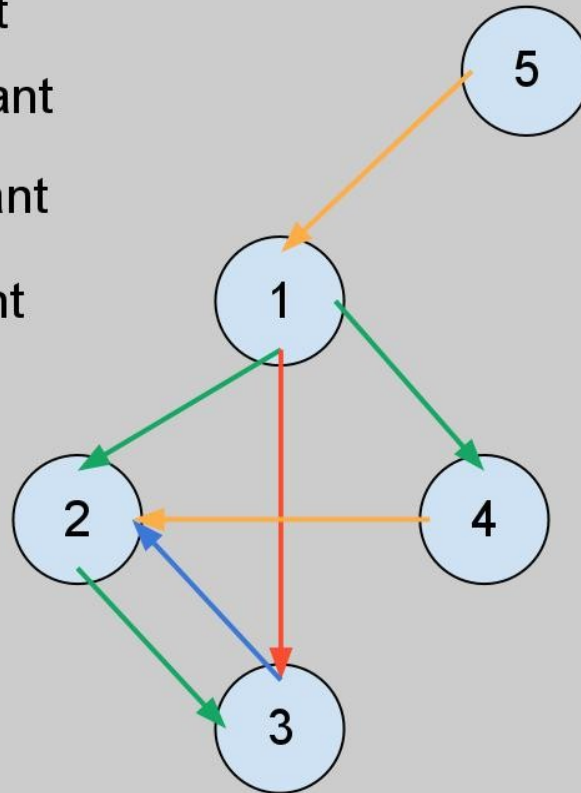
Intermission: Felles terminologi

- Fire kantkategorier
 - Tre-kant (Tree edge)
 - Bakoverkant (Backwards edge)
 - Foroverkant (Forward edge)
 - Krysskant (Cross edge)

- (Er ikke så viktig)

Kantkategorier

-  Tre-kant
-  Bakoverkant
-  Foroverkant
-  Krysskant



Bredde-først søk

```
def BFS(s):  
    Q = deque([s])  
    while Q:  
        u = Q[0] #Første element i køen  
        if u.naboer: #Har u flere naboer?  
            v = u.naboer.pop() #Fjern neste nabo  
            if v.uoppdaget:  
                Q.append(v)  
                v.uoppdaget = False  
        else:  
            Q.popleft()
```

Bredde-først søk

```
def BFS(s):  
    Q = deque([s])  
    while Q:  
        u = Q.popleft()           #Første element i køen  
        for v in u.naboer:  
            if v.uoppdaget:  
                Q.append(v)  
                v.uoppdaget = False
```

- Kjøretid: $O(V + E)$

Eksempel!

Egenskaper ved BFS

- Beregner korteste vei!

Bredde-først søk som byggesten

- Kan benyttes for å løse maks-flyt problemet
- Kan benyttes i søppeltømmingsalgoritmer for å finne referanser.

Bredde-først søk?

```
def BFS(s):
    D = deque([s])
    while D:
        u = D[0]                #Første element i køen
        if u.naboer:           #Har u flere naboer?
            v = u.naboer.pop() #Fjern neste nabo
            if v.uoppdaget:
                D.append(v)
                v.uoppdaget = False
        else:
            D.popleft()
```

Dybde-først søk!

```
def DFS(s):  
    D = deque([s])  
    while D:  
        u = D[-1]           #Øverste element i stakken  
        if u.naboer:       #Har u flere naboer?  
            v = u.naboer.pop() #Fjern neste nabo  
            if v.uoppdaget:  
                D.append(v)  
                v.uoppdaget = False  
        else:  
            D.pop()
```

- Kjøretid: $O(V + E)$

Eksempel!

Dybde-først søk

```
def DFS_whole(G):  
    for v in G.V:  
        if v.uoppdaget:  
            DFS(v)
```

Egenskaper ved DFS

- Parentesstruktur
- Noder langs hvite stier oppdages

Bruk av DFS

- Topologisk sortering
 - Avslør sykler
- Finn (sterkt) sammenknyttede komponenter
- Undersøk om grafen er enkeltvis sammenknyttet

and now for something



completely different...

Arrays

- RAM gjør arrays veldig raske
- Oppslag i konstant tid

Eksempel

- Server som opererer med sessions
- Hver bruker får en session id ved pålogging
- Server bruker session id til å slå opp i tabell
- Maksimalt 100 brukere pålogget samtidig

Eksempel

```
def login(u, pw):  
    authenticate(u, pw)  
    assert_valid_user_count()  
    max_session_id += 1  
    session_id = max_session_id  
    session_info[session_id] = SessionInfo()  
    return max_session_id
```

```
def get_session_info(session_id):  
    return session_info[session_id]
```

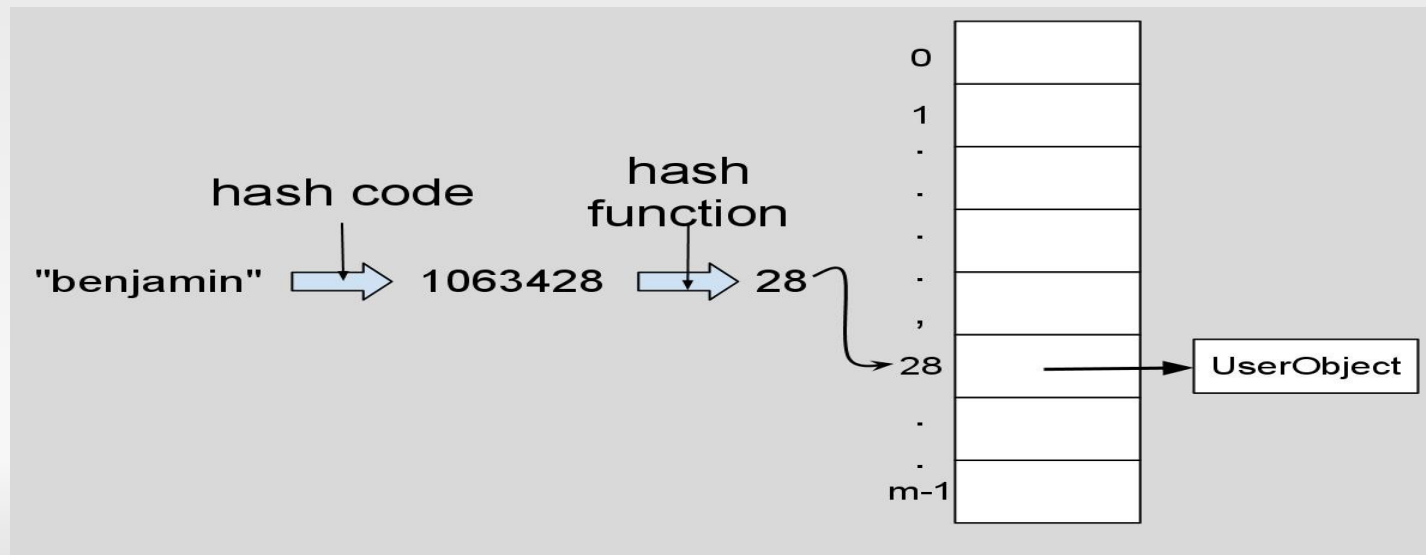
```
def store_session_info(session_id):  
    session_info[session_id].store()
```

Hva om...

- Vi bytter ut session id med bruker id?
 - Dårlig plassutnyttelse
 - Plasskrevende

Enter hashtables!

- Arrayprinsipp: slå opp vha indeks
- Forskjell: indeks beregnes ut fra nøkkel
 - Hashfunksjon
- Nøkkel trenger ikke være tall

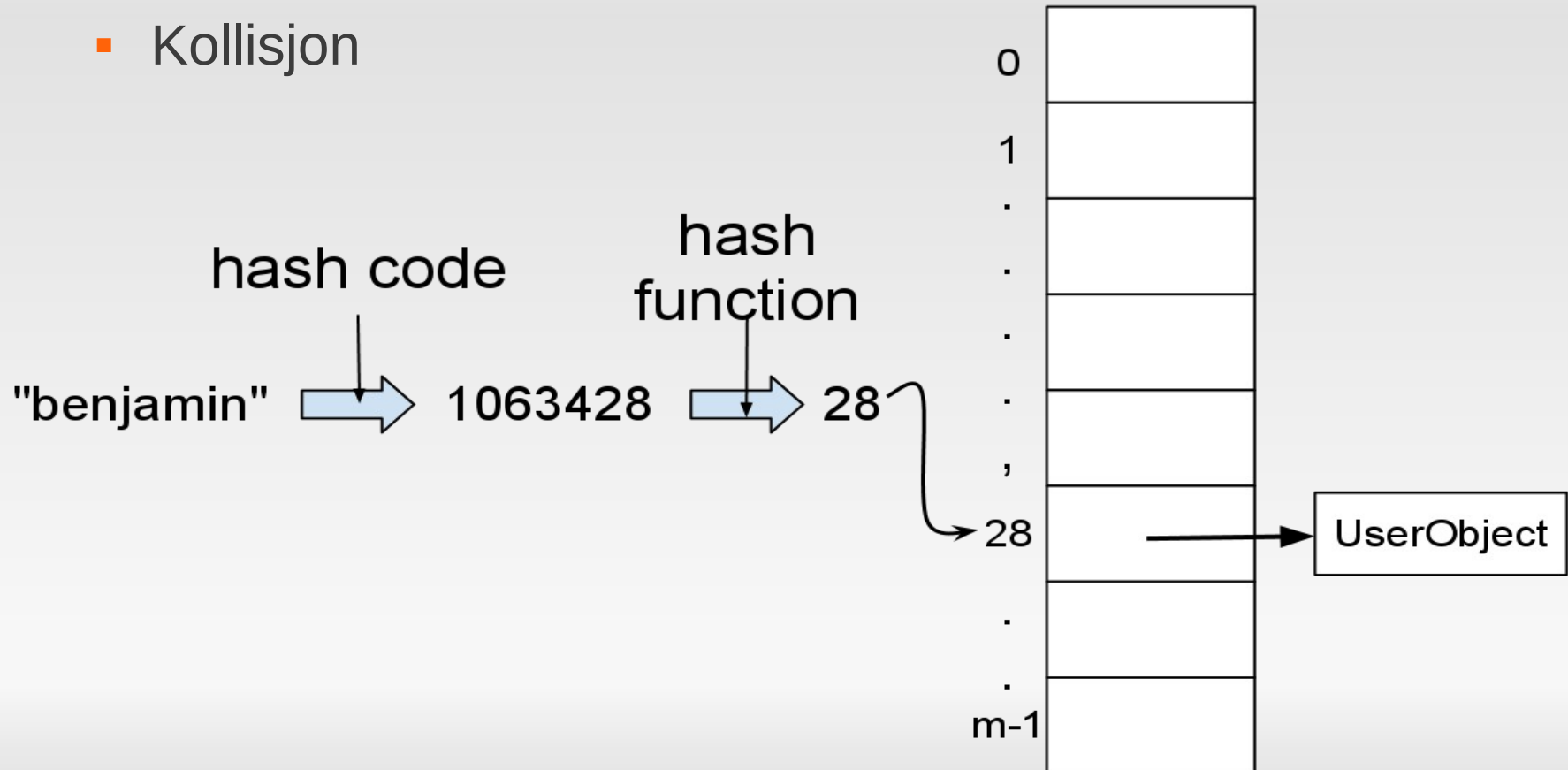


Eksempler

- Symboltabell
- Ordbok
- Brukerregister

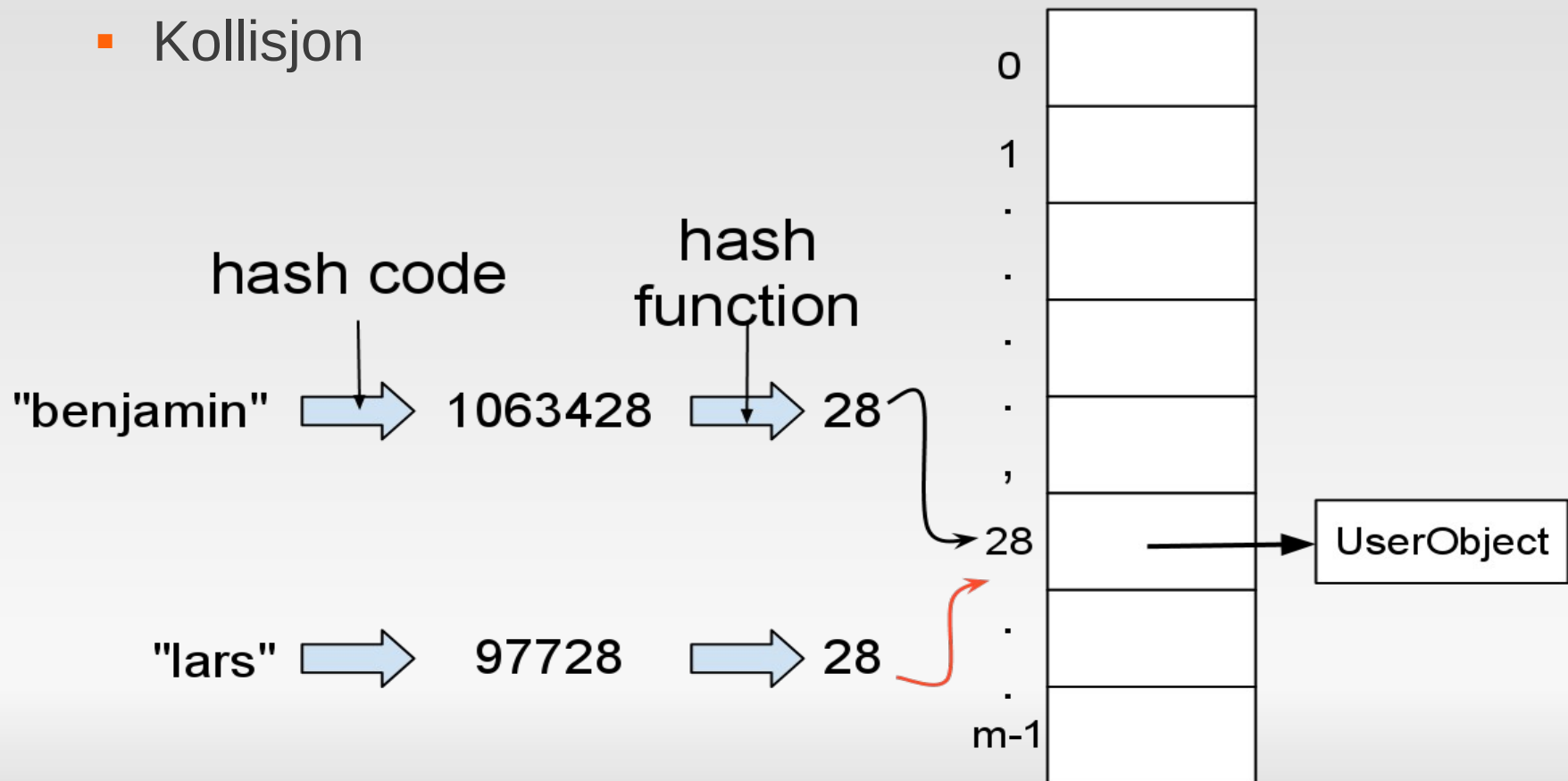
Problemer

- Nøkler kan gi samme hashverdi
 - Kollisjon



Problemer

- Nøkler kan gi samme hashverdi
 - Kollisjon



Løsninger

- Reaktive
 - Lenking (chaining)
 - Åpen adressering (open addressing)
- Preventive
 - God hashfunksjon

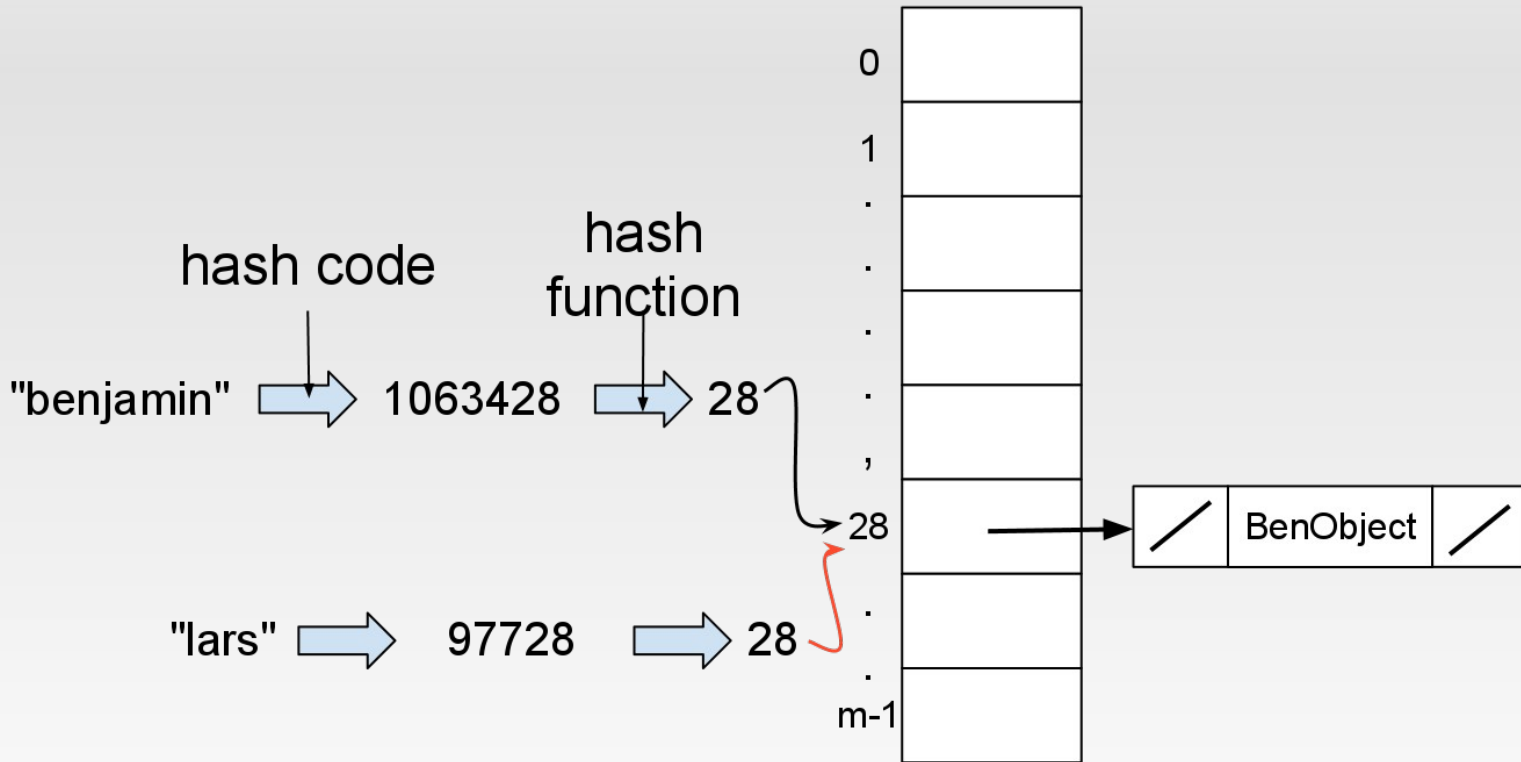
Terminologi

- Lastfaktor: $\alpha = n/m$
 - n : antall elementer i hashtabellen
 - m : antall plasser i hashtabellen
- Mål på fyllgrad i prosent
 - $m = 10, n = 5 \Rightarrow \alpha = 0.50$
 - "Tabellen er halvfull"
 - $m = 100, n = 200 \Rightarrow \alpha = 2$
 - "Tabellen er 200% full"

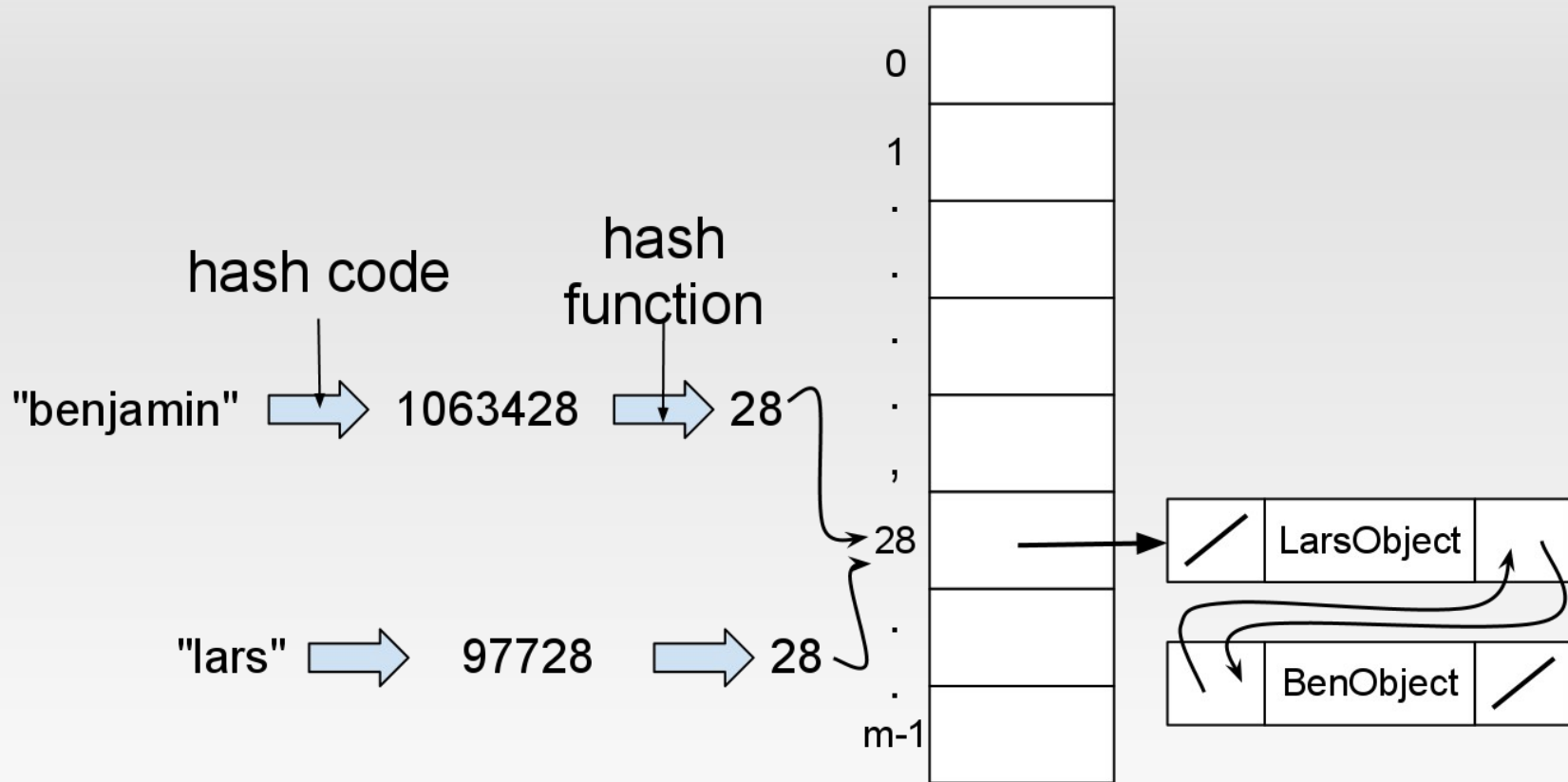
Lenking

- Hver plass i tabellen er en lenket liste
- Ved kollisjon: Legg ny verdi først i lista

Lenking - eksempel



Lenking - eksempel



Kjøretid

- Innsetting: Worst case $O(1)$
- Sletting: Worst case (med diverse antagelser) $O(1)$
- Søk: Average case $O(1 + \alpha)$
 - Antall bøtter må være proporsjonalt i forhold til antall elementer
 - I praksis konstant søketid

Åpen adressering

- Elementer lagres direkte i tabellen
- Ved kollisjon leter vi etter en ny plass

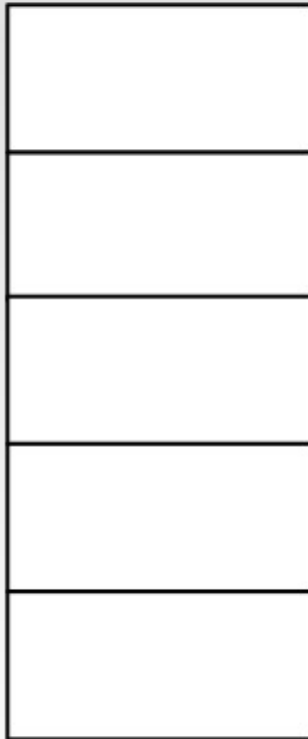
Åpen adressering - eksempel

1

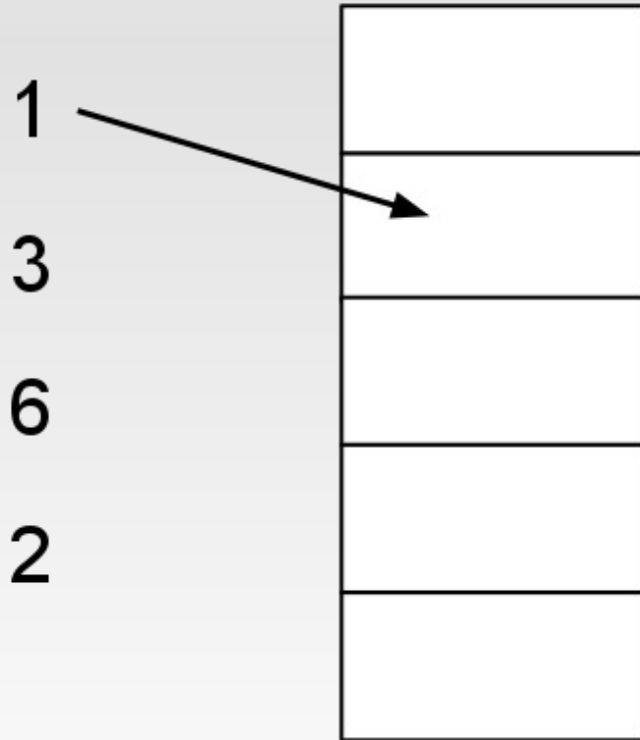
3

6

2



Åpen adressering - eksempel

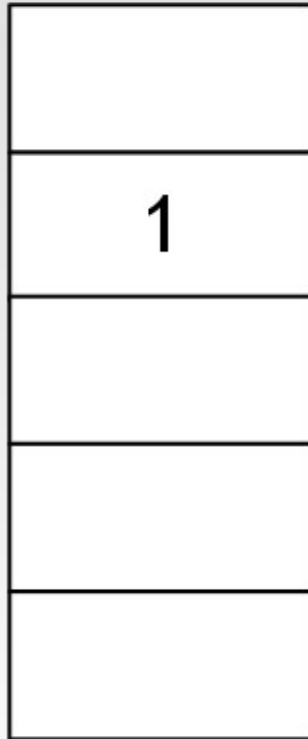


Åpen adressering - eksempel

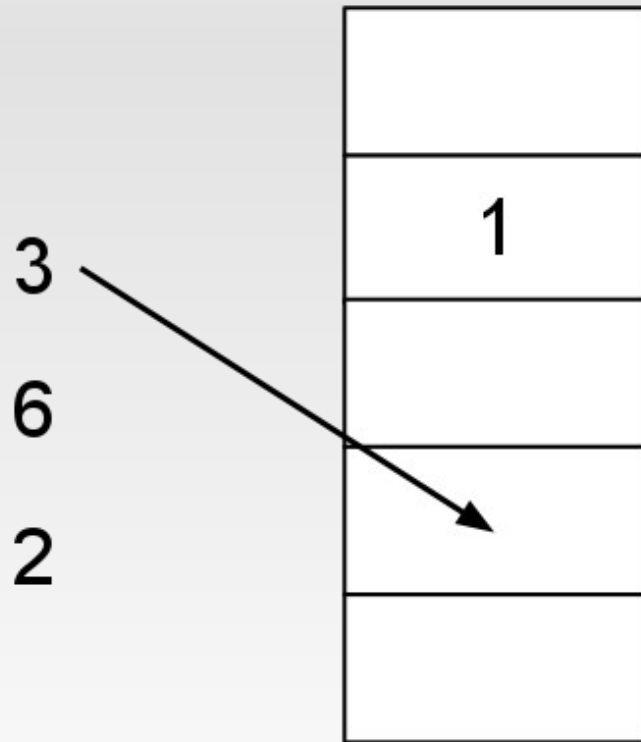
3

6

2



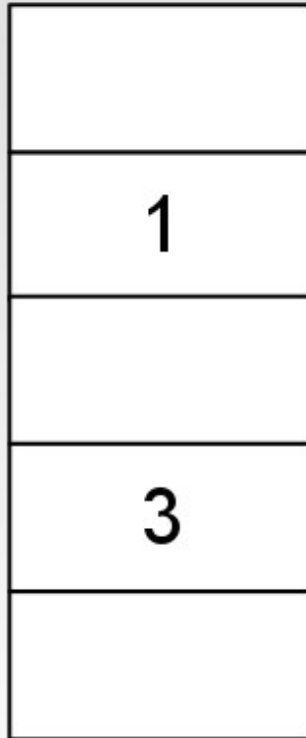
Åpen adressering - eksempel



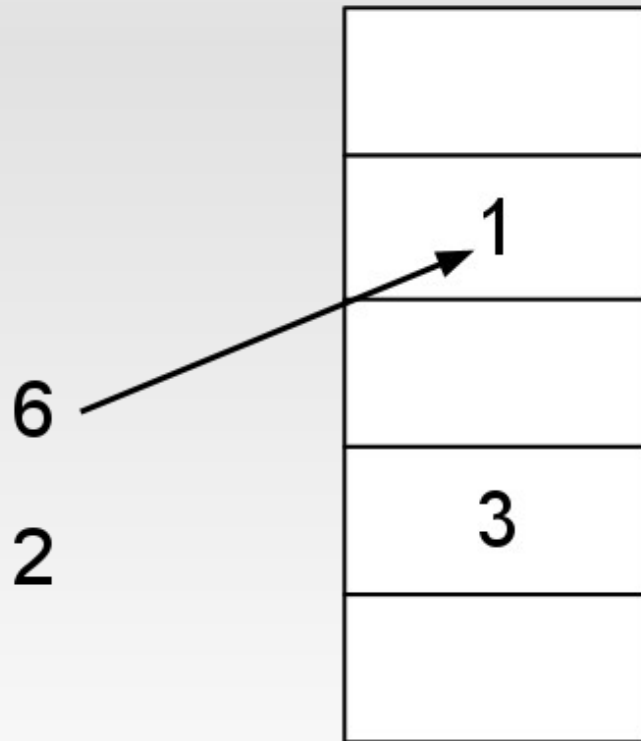
Åpen adressering - eksempel

6

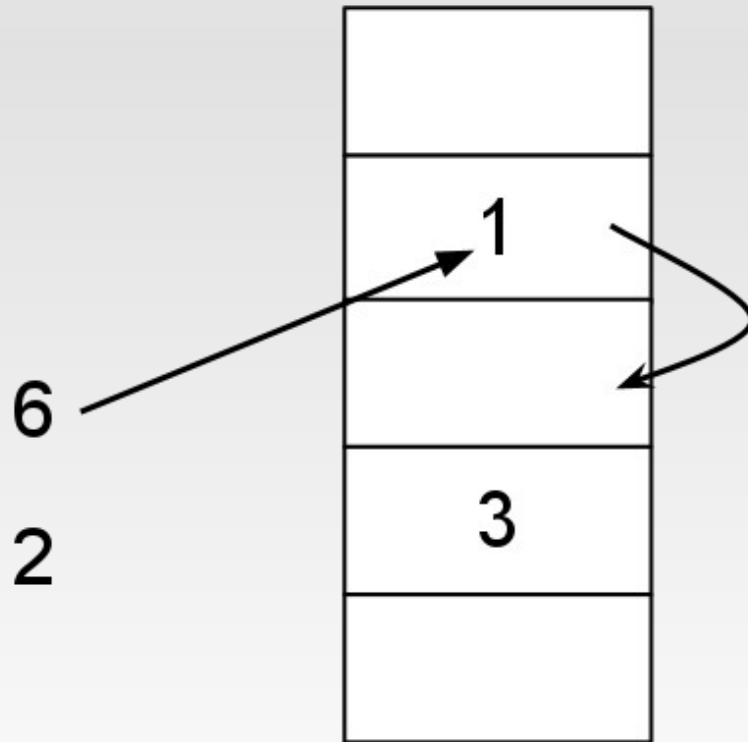
2



Åpen adressering - eksempel



Åpen adressering - eksempel

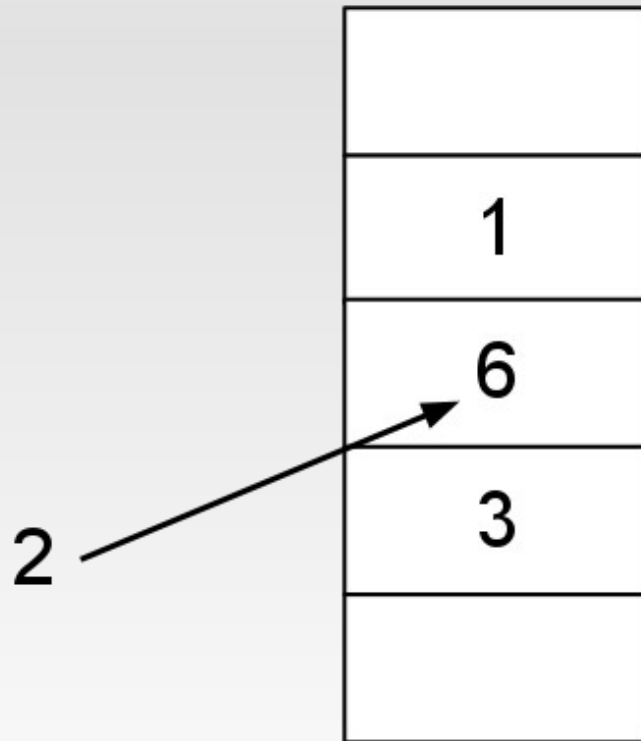


Åpen adressering - eksempel

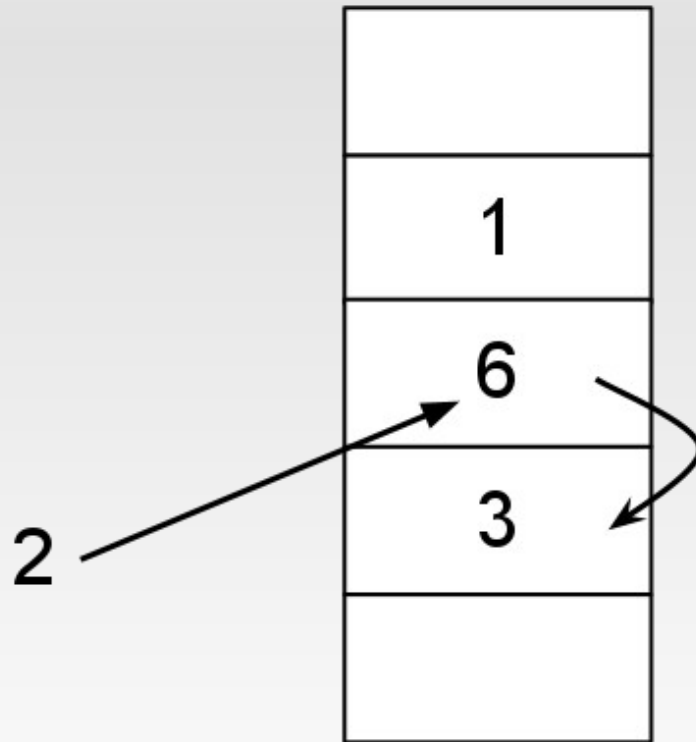
2

1
6
3

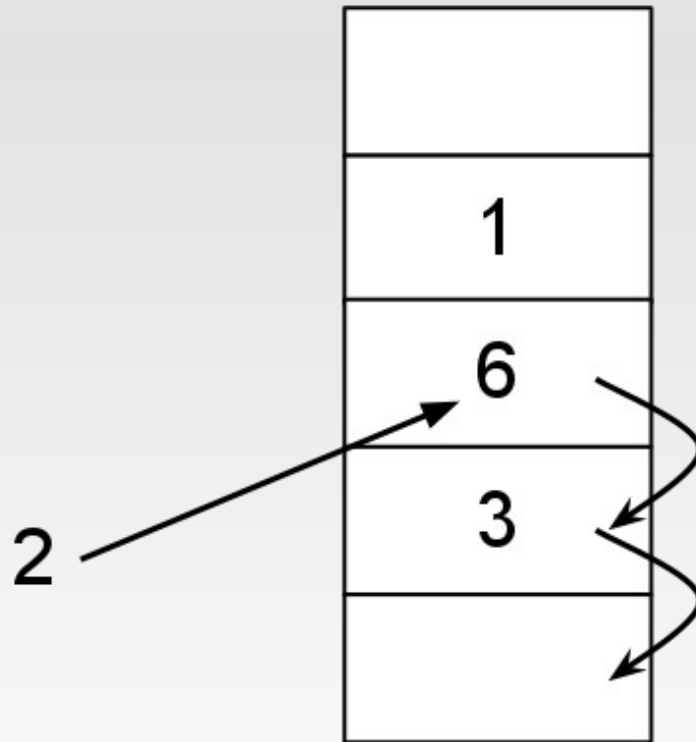
Åpen adressering - eksempel



Åpen adressering - eksempel



Åpen adressering - eksempel



Åpen adressering - eksempel

1
6
3
2

Letemetodikk

- Linear probing
 - $h(k, i) = (h'(k) + i) \bmod m$

Letemetodikk

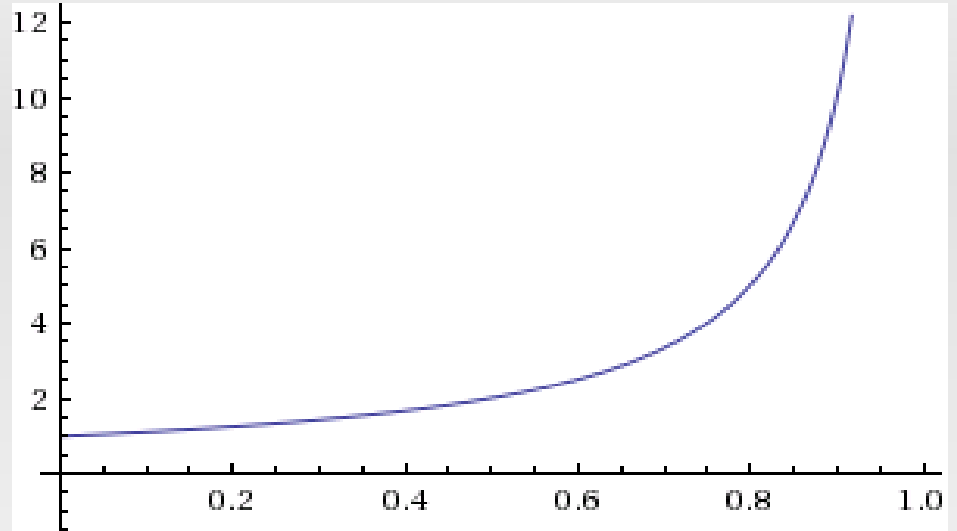
- Linear probing
 - $h(k, i) = (h'(k) + i) \bmod m$
- Quadratic probing
 - $h(k, i) = (h'(k) + c \cdot i + d \cdot i^2) \bmod m$

Letemetodikk

- Linear probing
 - $h(k, i) = (h'(k) + i) \bmod m$
- Quadratic probing
 - $h(k, i) = (h'(k) + c \cdot i + d \cdot i^2) \bmod m$
- Double hashing
 - $h(k, i) = (h_1(k) + i \cdot h_2(k)) \bmod m$

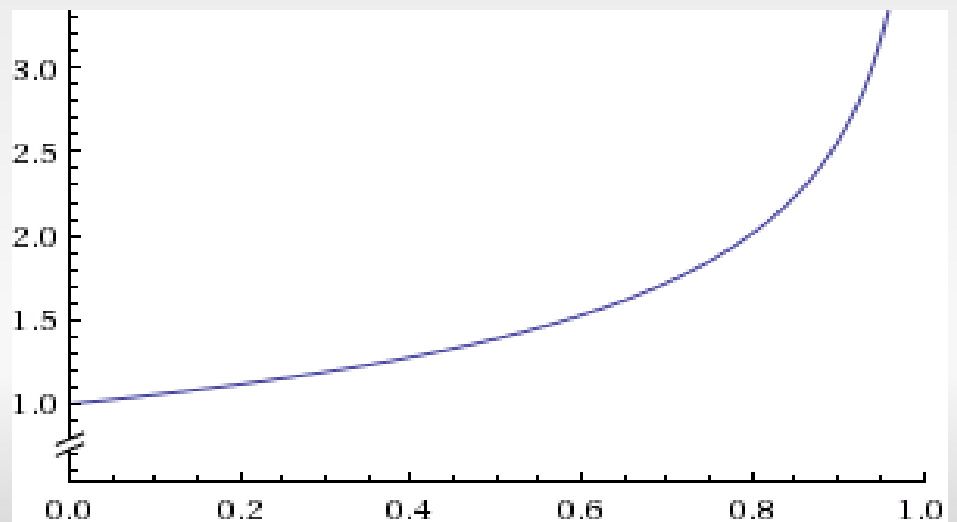
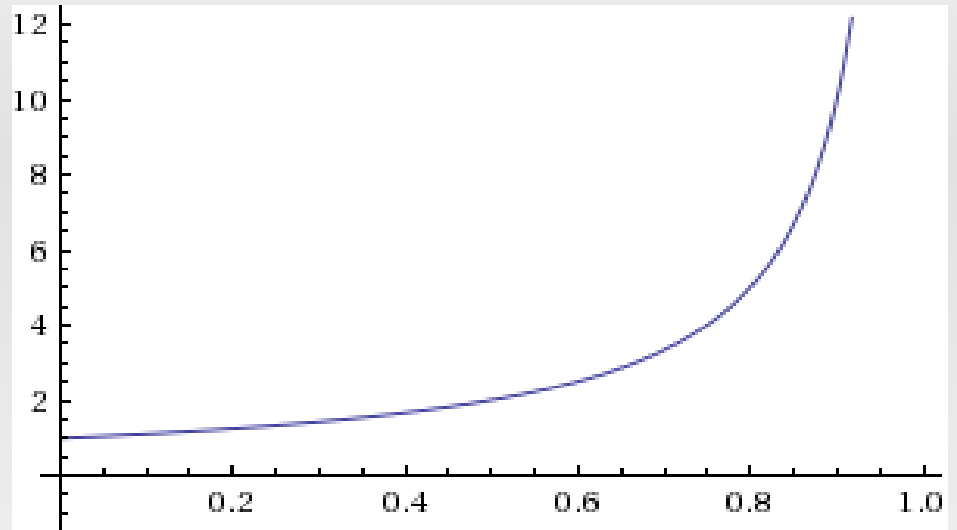
Kjøretid (med forbehold)

- Innsetning, bomstøk:
 - $O(1/(1 - \alpha)) \Rightarrow$



Kjøretid (med forbehold)

- Innsetning, bomstøk:
 - $O(1/(1 - \alpha)) \Rightarrow$
- Treffstøk:
 - $O(1/\alpha * \ln(1/(1 - \alpha))) \Rightarrow$
- Konklusjon:
Høy α gir dårlig ytelse



Merk!

Open Hashing == Closed Addressing

≠

Closed Hashing == Open Addressing

Hashfunksjoner

- Påkrevde egenskaper:
 - Deterministisk
- Ønskede egenskaper:
 - Jevn fordeling - enkel uniform hashing
 - Rask

Hashfunksjoner

- Divisjonsmetoden
 - $h(k) = k \bmod m$
- Multiplikasjonsmetoden
 - $h(k) = \text{floor}(m * (kA \bmod 1))$

Øving 1 - Teori

Øving 1 - Praksis

Spørsmål?