Velkommen til IN 2010 gruppe 6

Planen for i dag

- Pensum / Oppgaver
- Lab

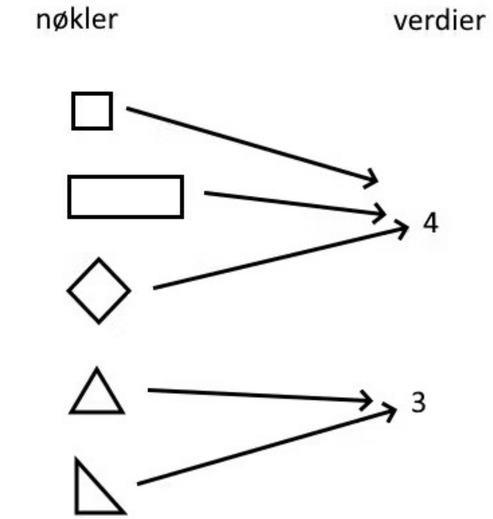
Mentimeter

Pensum gjennomgang

Map

- Abstrakt datatype som går ut på:
  - Man har nøkler
  - Hver nøkkel har kun EN verdi

- Må ha med:
  - Insetting -> Map.put(key, value)
  - Oppslag -> Map.get(key)
  - Sletting -> Map.remove(key)

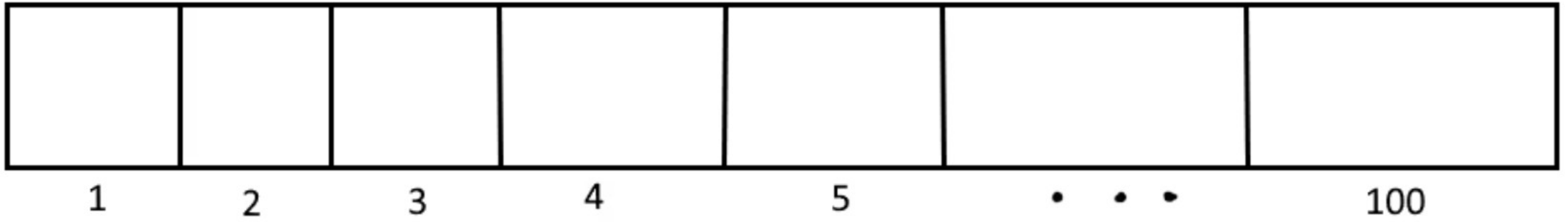


HashMap

- De-abstrahering av Map datastrukturen
- Vi bruker et Array sammen med en Hashfunksjon
  - Hashfunksjonen konverterer nøkkelen vi setter inn til en indeks i Arrayet, aka å "hashe"

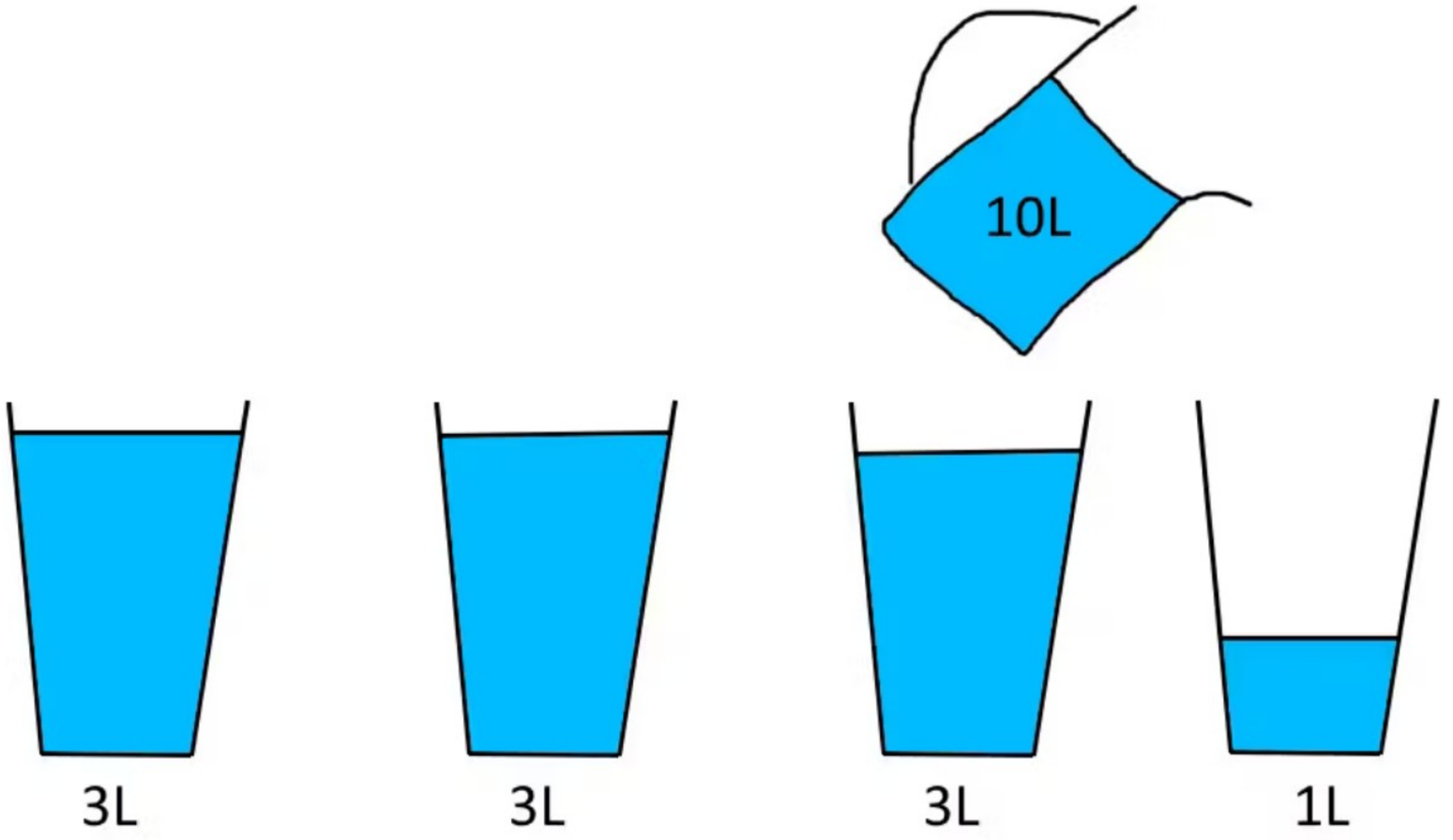
- Får vi et problem
  - Arrayet kan ikke være uendelig langt
  - Inputtene kan være uendelig mange
  - Får **kollisjoner** (to forskjellige nøkler blir hashet til samme indeks)

```
stor_liste = [*liste med 1000 tall*]
for tall in stor_liste:
    ordbok[tall] = tall * 2
```



Hva er modulo?

- Modulo i korte trekk er resten til et dele stykke
- 10 / 3 = 3 + 1 i rest
- 10 % 3 = 1



Hashing av strenger

- Går over hver bokstav av strengen som summerer opp til et spesielt tall
- Det spesielle tallet skal være unik for akkuratt den strengen

Hva er problemet her?

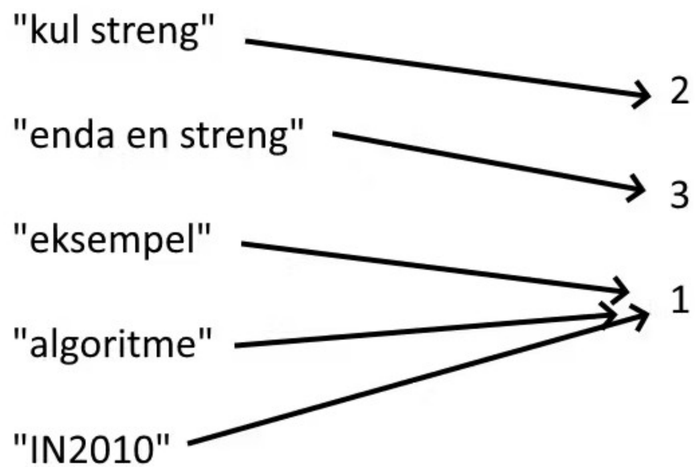
```
def kul_hash(key, size):
    # ord() er parseToInt fra Java
    return ord(key[0]) % size
```

```python
def kul_hash(key, size):
    h = 0
    for char in key:
        # ord() er parseToInt fra Java
        h = 31 * h + ord(char)
    return h % size
```

Hva er problemet her?
97

Hva er problemet her?

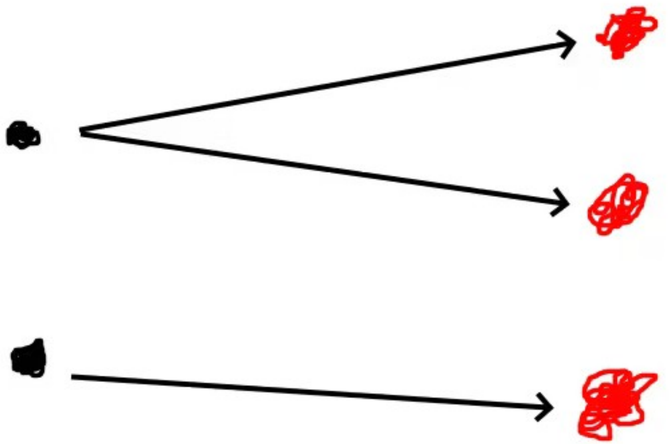
```python
def kul_hash(key, size):
    return random.randint(0, size - 1)
```



Hva er problemet her?

Mentimeter

Hva er problemet her?



Det vil alltids være kollisjoner

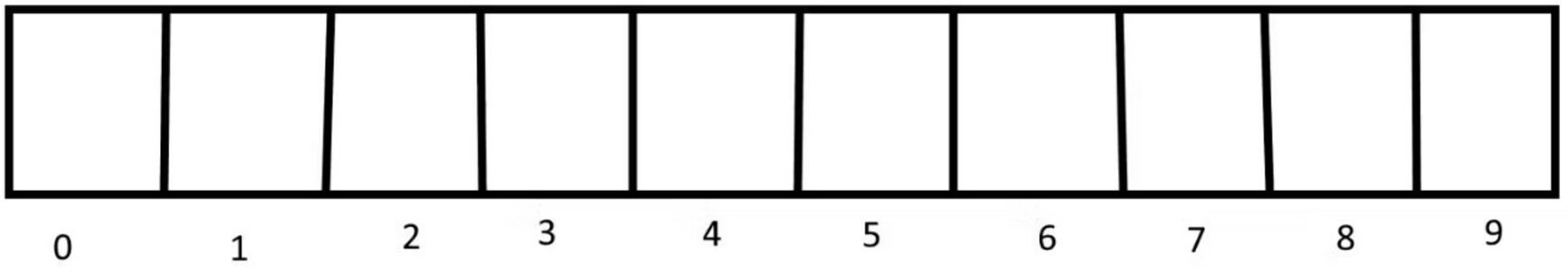
- Skal videre vise hvordan det håndteres på to forskjellige måter
  - Seperate chaining
    - Array med lenkelister
  - Linear probing
    - Setter inn i neste ledig plass i Arrayet

Seperate chaining

- Insetting av nøkkel **k** med verdi **v**
  - Sjekker Array[i], hvis **null** så sett in en lenkeliste med **k** og **v**
    hvis det er en lenkeliste der fra før av, sett **k** på slutten med **v**
  - Hvis **k** finnes i lenkelisten fra før av så erstatter vi den
    med den nye verdien **v**
- Slå opp nøkkel **k**
  - Sjekker Array[i], hvis **null**, returner **null**, ellers let igjennom
    listen og returner verdien til **k**
- Seltting av nøkkel **k** med verdi **k**
  - Sjekker Array[i], hvis **null**, return, ellers fjern **k** og **v** fra lenkelisten

verdier:

1
5
10
16
17
0
18
9



Linear probing

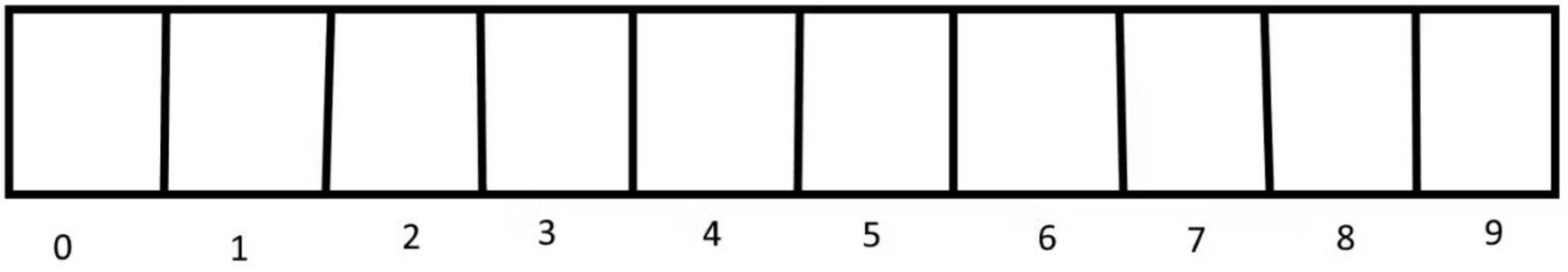
- Insetting av nøkkel **k** med verdi **v**
  - Sjekker Array[i], hvis null sett inn **k** og **v**, hvis nøkkelen er der fra før av, bytt ut verdien med **v**
  - Ellers gå til neste indeks og prøv igjen
- Slå opp nøkkel **k**
  - Sjekker Array[i], hvis **null**, return **null**, hvis nøkkelen er der så return verdien
  - Ellers gå til neste indeks
- Seltting av nøkkel **k** med verdi **v**
  - Sjekker Array[i], hvis **null**, reuturn **null**, hvis nøkkelen ikke matcher gå til neste indeks
  - Hvis nøkkelen matcher med **k** så sett den til **null**
  - Tett hullet man lagde (mer om det neste slide)

Tette hull i Linear probing

- For å tette et hull må vi finne det neste riktige elementet
- Det neste riktige elementet blir det elementet som har samme hash-indeks som det stedet vi slettet fra

verdier:

1
5
10
16
17
0
18
9



Rehashing / Kjøretid

- Når Arrayet begynner å bli fullt vil det oppstå flere kollisjoner
- Dette fikses ved å bare overføre alle elementer til et nytt og større Array
- O(n) prosess
- Hvis dere har laget grafen i Innlevering4 raskt så har dere brukt mengder og ikke lister => mengder er raskere enn lister
- Vi sier da at vi fordelere operasjonene i rehashingen til alle de tidligere operasjonene for rask innsetting
- Insetting får O(1 + 1) :)

Linear probing

10 poeng

(a) Vi starter med et tomt array på størrelse 10.

0	1	2	3	4	5	6	7	8	9

Hashfunksjonen du skal bruke er $h(k, N) = k \mod N$, som for dette eksempelet blir det samme som $h(k, 10) = k \mod 10$. Altså hasher et tall til sitt siste siffer.

Bruk *linear probing* til å sette inn disse tallene i den gitte rekkefølgen:

$$21, 54, 82, 10, 20, 44$$

Fyll ut tabellen slik den ser ut etter alle tallene er satt inn med linear probing. Skriv svaret som en kommaseparert liste, der _ kan brukes for å indikere en tom plass.

(b) Forklar kort hvordan algoritmen for innsetting ved linear probing fungerer, og skisser algoritmen med pseudokode. Du kan anta at arrayet i input ikke er fullt, og at du har en hashfunksjon h slik at $h(k, N)$ gir et tall mellom 0 og $N - 1$ for en vilkårlig nøkkel k. Ledig plass i arrayet er indikert ved **`null`**.

Input: Et array A av størrelse N, en nøkkel k og verdi v
Output: Et array som inneholder (k, v)

```
1 Procedure LinearProbingInsert(A, k, v)
  |  // ...
```

Mentimeter

Lab

mafredri@ifi.uio.no