4.1 Stakke og køer - lister noter CLRS 10

- Datastrukturer
- Stakke og køer
- Hægtede lister

Disse noter er stærkt inspireret af noter af Philip Bille og Inge Li Gørtz til kurset Algoritmer og Datastrukturer, på DTU,
http://www2.compute.dtu.dk/courses/02105+02326/2015/#generelinfo

Introduktion til datastrukturer

- Datastrukturer
- Stakke og køer
- Hægtede lister

Datastrukturer

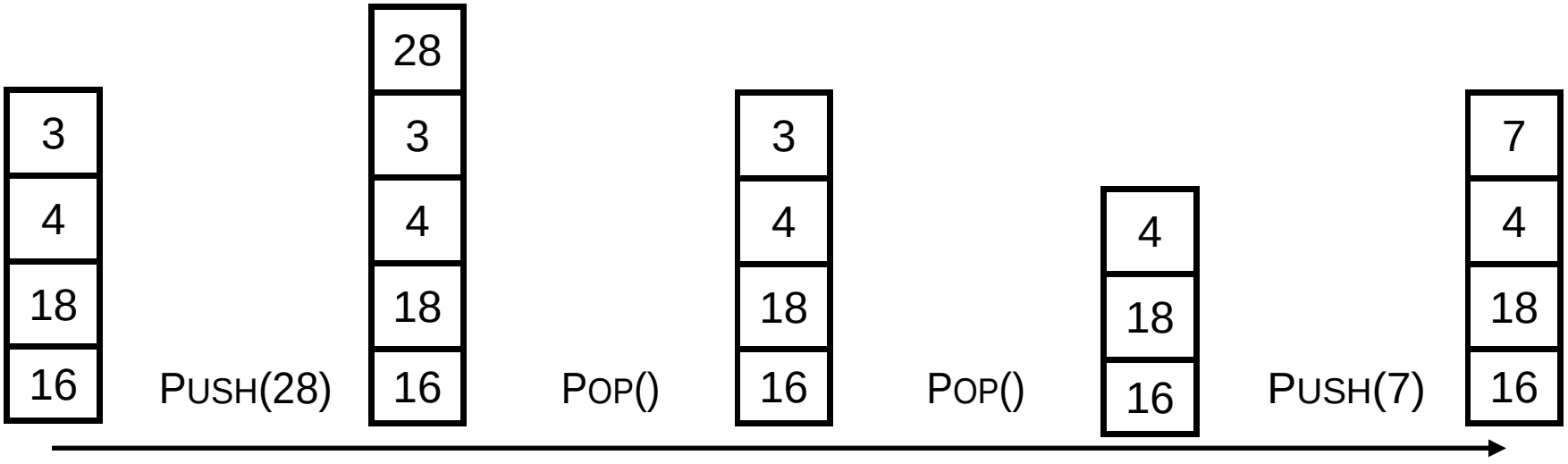
- Datastruktur. Metode til at organise data så det kan søges i/tilgås/manipuleres effektivt.
- Mål. Hurtig og kompakt
- Terminologi. Dynamisk vs. statisk datastruktur.

Introduktion til datastrukturer

- Datastrukturer
- Stakke og køer
- Hægtede lister

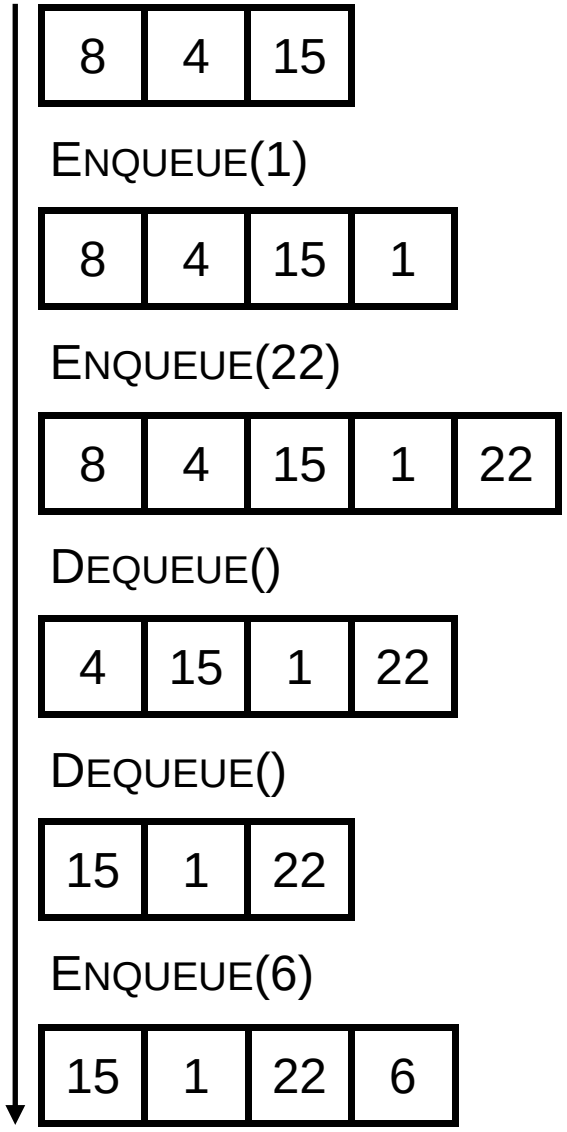
Stak

- Stak. Vedligehold en dynamisk sekvens (stakken) S af elementer under følgende operationer.
  - PUSH(x): tilføj et nyt element x til S.
  - POP(): fjern og returner det seneste tilføjede element i S.
  - ISEMPTY(): returner sand hvis S ikke indeholder nogle elementer.



Kø

- Kø. Vedligehold en dynamisk sekvens (køen) K af elementer under følgende operationer.
  - ENQUEUE(x): tilføj et nyt element x til K
  - DEQUEUE(): fjern og returner det tidligst tilføjede element i K.
  - ISEMPTY(): returner sand hvis K ikke indeholder nogle elementer.

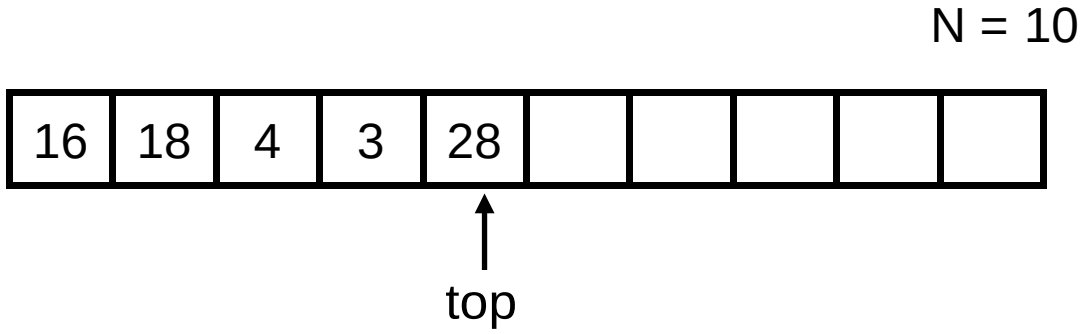


Anvendelser

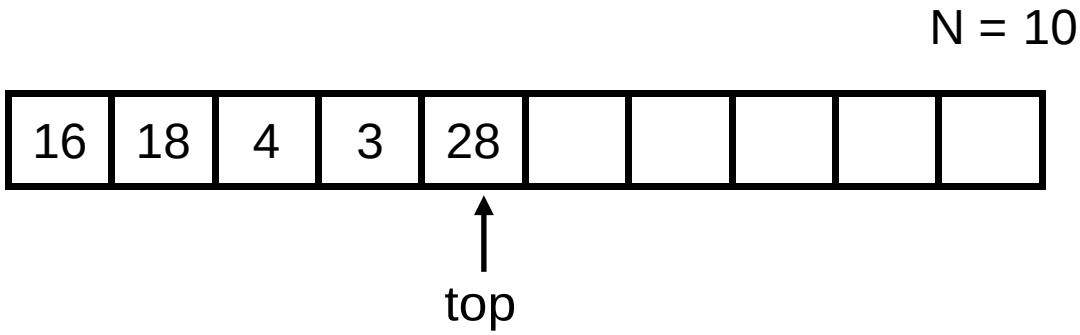
- Stakke.
  - Virtuelle maskiner
  - Parsing
  - Funktionskald
  - Backtracking
- Køer.
  - Skedulering af processer
  - Buffering
  - Breddeførst søgning

Implementation af stak med tabel

- Stak. Stak med kapacitet N vha. tabel.
- Datastruktur.
  - Tabel S[1..N]
  - Index top i S.
- Operationer.
  - PUSH(x): Tilføj x på S[top+1], sæt top = top + 1
  - POP(): returner S[top], sæt top = top - 1
  - ISEMPTY(): returner sand hvis og kun hvis top = 0.
  - Tjek for overløb og underløb i PUSH og POP.



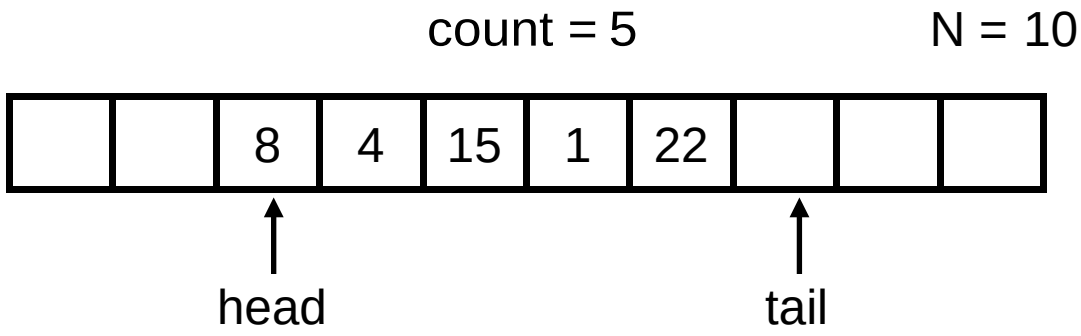
Implementation af stak med tabel



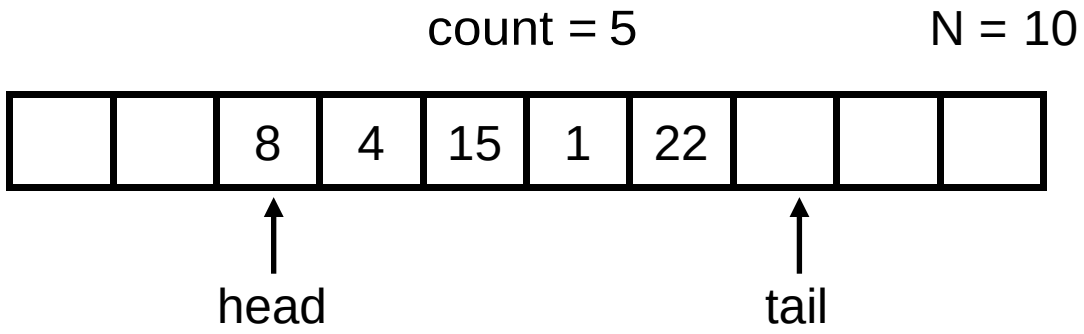
- Tid.
  - PUSH i Θ(1) tid.
  - POP i Θ(1) tid.
  - ISEMPTY i Θ(1) tid.
- Plads.
  - Θ(N) plads.
- Mangler.
  - Vi skal kende kapacitet N fra start.
  - Vi spilder plads når antal elementer er << N.

Implementation af kø med tabel

- Kø. Kø med kapacitet N vha. tabel.
- Datastruktur.
  - Tabel S[1..N]
  - Indeks head (tidligst indsatte element) og tail (næste ledige element) i S og tæller count (antal elementer i kø).
- Operationer.
  - ENQUEUE(x): Tilføj x på S[tail], opdater count og tail cyklisk.
  - DEQUEUE(): returner S[head], opdater count og head cyklisk.
  - ISEMPTY(): returner sand hvis og kun hvis count = 0.
  - Tjek for overløb og underløb i DEQUEUE og ENQUEUE.



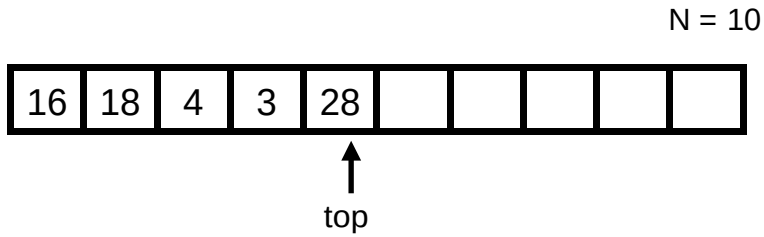
Implementation af kø med tabel



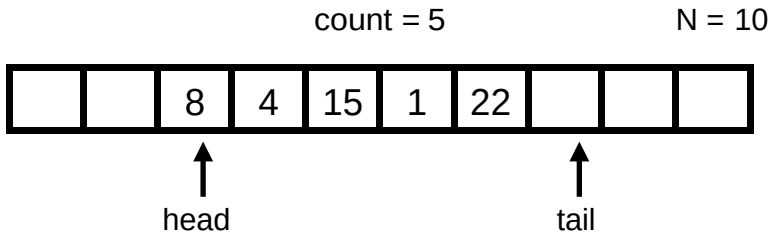
- Tid.
  - ENQUEUE i Θ(1) tid.
  - DEQUEUE i Θ(1) tid.
  - ISEMPTY i Θ(1) tid.
- Plads.
  - Θ(N) plads.
- Mangler.
  - Vi skal kende kapacitet N fra start.
  - Vi spilder plads når antal elementer er << N.

Stakke og køer (opsamling)

- Stak.
  - Tid. PUSH, POP, ISEMPTY i Θ(1) tid.
  - Plads. Θ(N)



- Kø.
  - Tid. ENQUEUE, Dequeue, ISEMPTY i Θ(1) tid.
  - Plads. Θ(N)



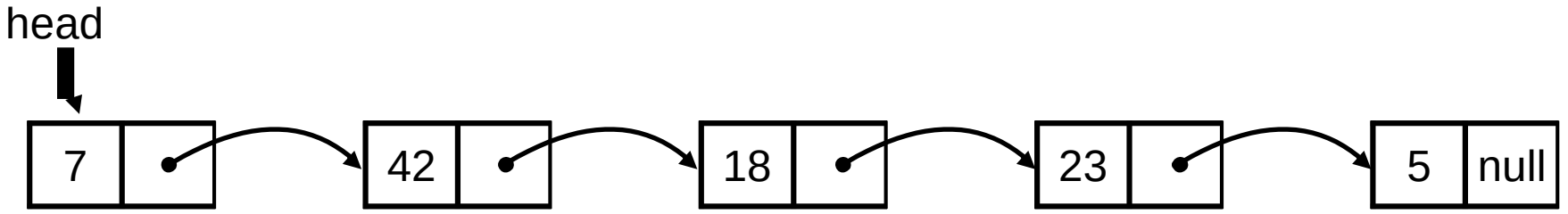
- Udfordring. Kan vi komme ned på lineær plads med samme tid? (uden at kende N på forhånd)

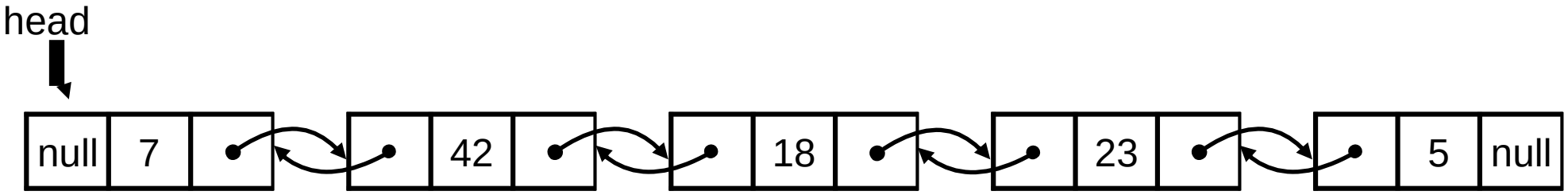
Introduktion til datastrukturer

- Datastrukturer
- Stakke og køer
- Hægtede lister

Hægtede lister

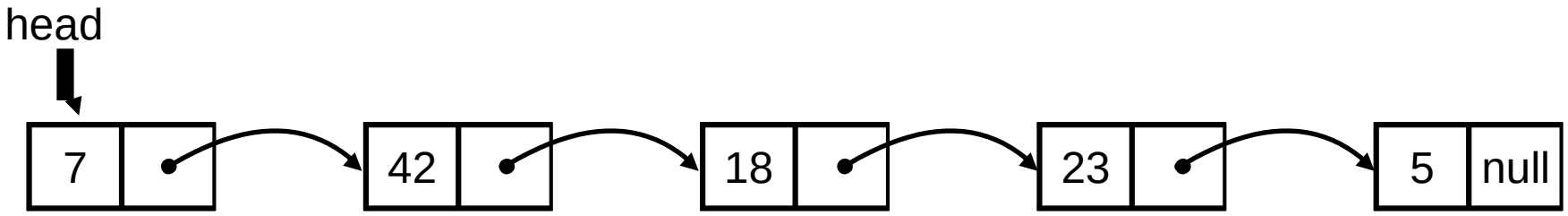
- Hægtede lister.
  - Datastruktur til at vedligeholde en dynamisk sekvens af elementer i lineær plads.
  - Rækkefølge af elementer bestemt af referencer/pegere kaldet hægter.
  - Effetiv at indsætte og fjerne elementer eller sammenhængende dele af elementer.
  - Dobbelt-hægtede vs enkelt-hægtede.

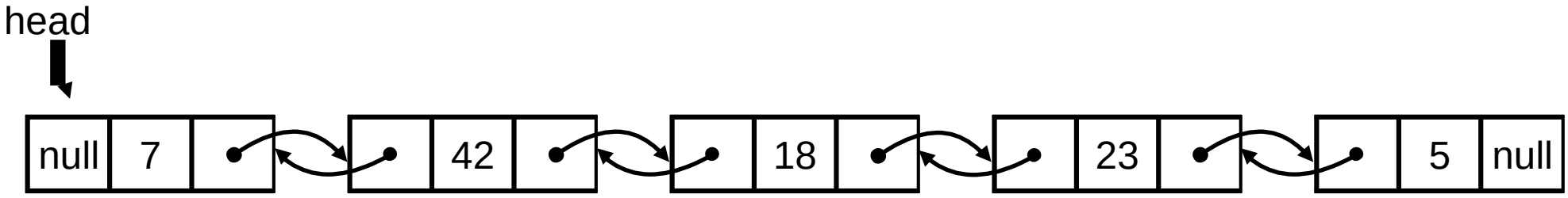




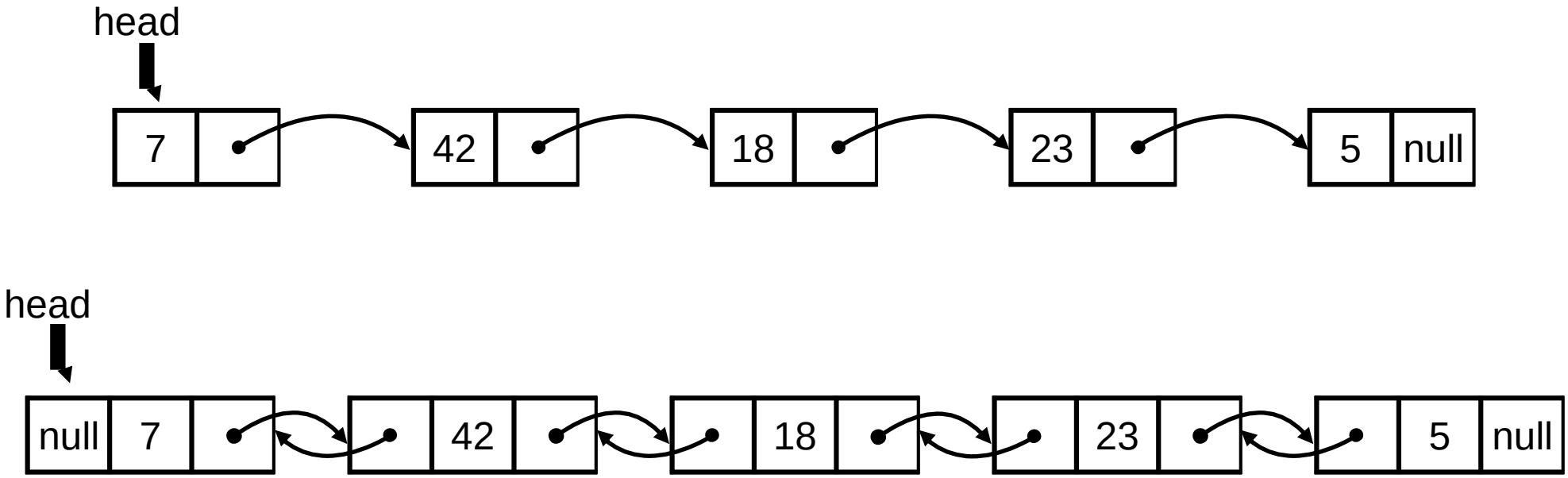
Hægtede lister

- Simple operationer.
  - SEARCH(head, k): returner knude med værdi k i listen. Returner null hvis den ikke findes.
  - INSERT(head, x): indsæt knude x i starten af listen. Returner ny head.
  - INSERT(y, x): indsæt knude x efter knuden y (kræver at vi har y på forhånd).
  - DELETE(head, x): fjern knude x i listen.





Hægtede lister

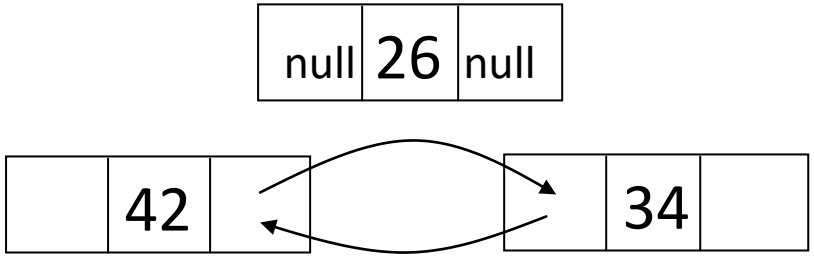


- Tid. Hvor hurtigt kører operationerne?
  - SEARCH i $\Theta(n)$ tid
  - INSERT og DELETE i $\Theta(1)$ tid.
- Plads.
  - $\Theta(n)$

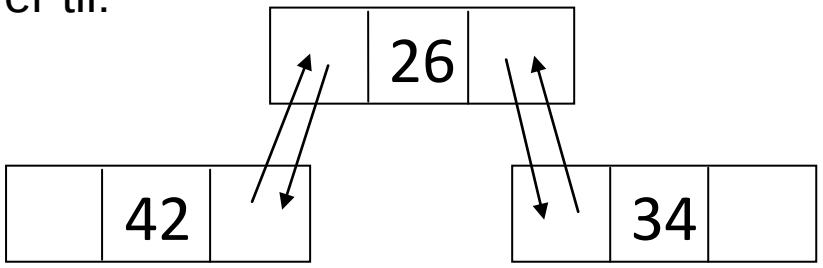
Hægtede lister

Hvordan fungerer insert?

Simpelt: Flyt pegerne så den nye knude er mellem to eksisterende!



Bliver til:



Delete: Gør det modsatte.

Hægtede lister

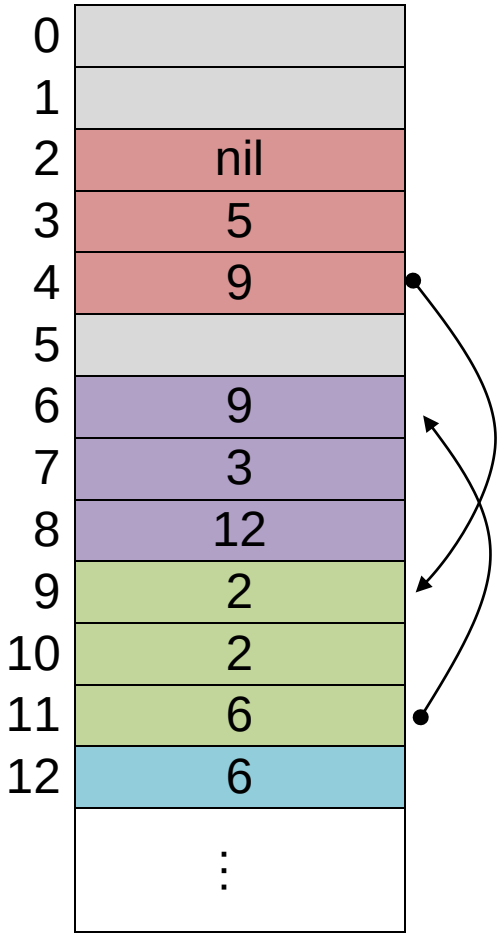
I RAM modellen?

- Hver knude identificeres med første plads i RAM'en.
- Hver knude fylder tre ord i RAM'en.

Se eksempel herunder: Første knude er placeret på plads 2 (til 4) I RAM'en.

Farverne til højre svarer til farverne i knuderne herunder. Knuderne der indeholder 1 og 8 kan ikke ses helt i uddraget.

Bemærk hvordan knuderne ikke ligger i rækkefølge i rammen (kun internt for en knude). Der kan også være ubrugt plads imellem knuderne.

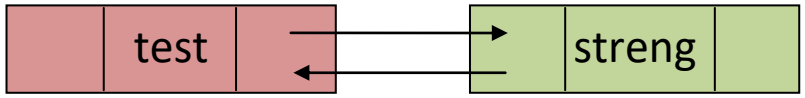


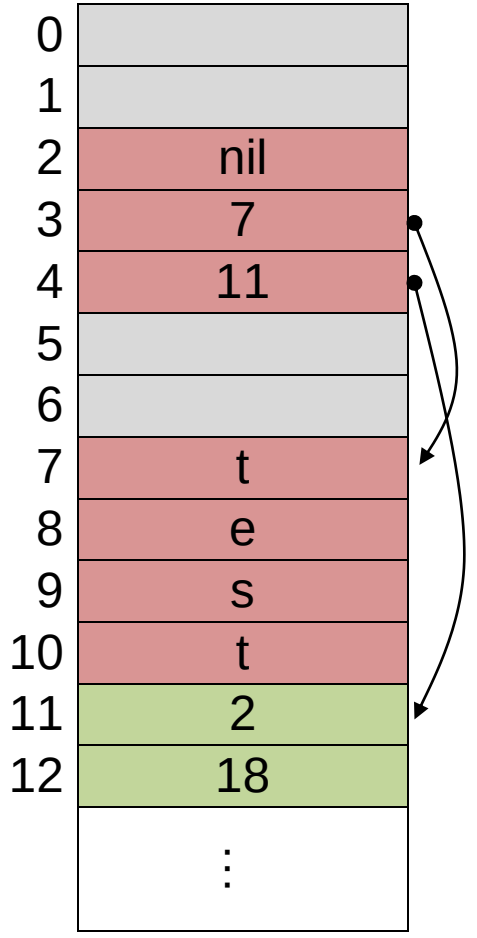
Hægtede lister

Hvad hvis en knude indeholder mere end et tal?

- Samme idé. Vi gemmer bare adressen til det knuden indeholder!

Se eksempel herunder:





Bemærk, at hvert bogstav fylder en hel plads i RAM'en.
I et praksis ville der nok være en plads efter strengen "test" der indikerede at vi havde nået slutningen af strengen, men det abstraherer vi fra her.

Med disse få idéer kan vi bygge meget komplicerede data strukturer!

Implementation af stak og kø med hægtede lister

- Opgave. Overvej hvordan man kan implementere stakke og køer med hægtede lister effektivt.

- Stak. Vedligehold en dynamisk sekvens (stakken) S af elementer under følgende operationer.
  - PUSH(x): tilføj et nyt element x til S.
  - POP(): fjern og returner det seneste tilføjede element i S.
  - ISEMPTY(): returner sand hvis S ikke indeholder nogle elementer.

- Kø. Vedligehold en dynamisk sekvens (køen) K af elementer under følgende operationer.
  - ENQUEUE(x): tilføj et nyt element x til K
  - DEQUEUE(): fjern og returner det tidligst tilføjede element i K.
  - ISEMPTY(): returner sand hvis K ikke indeholder nogle elementer.

Stakke og køer

- Stak og kø implementeret med hægtet liste
- Stak.
  - Tid. PUSH, POP, ISEMPTY i Θ(1) tid.
  - Plads. Θ(n)

- Kø.
  - Tid. ENQUEUE, Dequeue, ISEMPTY i Θ(1) tid.
  - Plads. Θ(n)

Hægtede lister

- Hægtet liste. Fleksibel datastruktur til at vedligeholde en sekvens af elementer i lineær plads.
- Andre hægtede datastrukturer. Cykliske lister, træer, grafer, …