Basale datastrukturer, DMA uge 4

- Hvad er en datastruktur?
- Stakke
- Køer
- Hægtede lister
- Prioritetskøer
- Hobe

Mikkel Abrahamsen

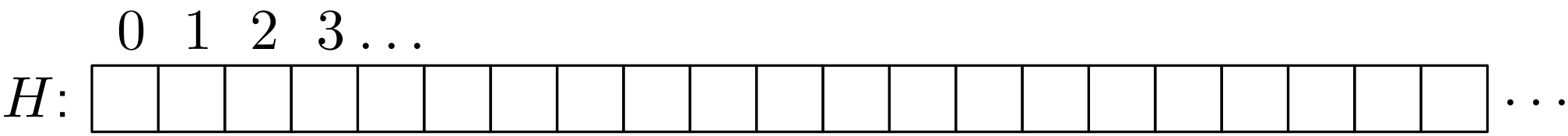
Hvad er en datastruktur?

En måde at organisere data på.
Mål: Søge/tilgå/ændre effektivt, dvs. hurtigt og pladsbesparende (kompakt).
Eksempler vi har set: Array og sorteret array.

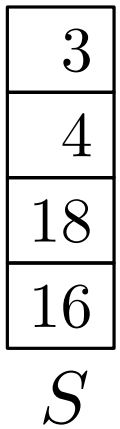
Hvad er en datastruktur?

En måde at organisere data på.
Mål: Søge/tilgå/ændre effektivt, dvs. hurtigt og pladsbesparende (kompakt).
Eksempler vi har set: Array og sorteret array.

Husk: Alle datastrukturer skal gemmes i hukommelsen, dvs. i et stort array.

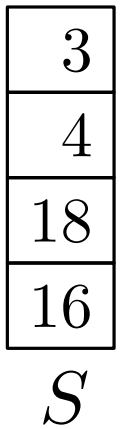


Stak

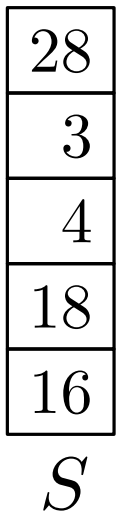


Stak

Push(S,28)

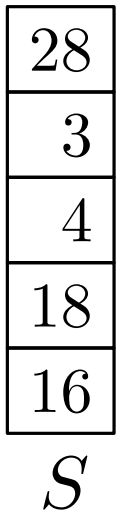


Stak



Push(S,28)

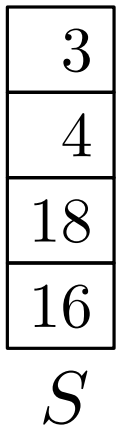
Stak



Push(S,28)

Pop(S)

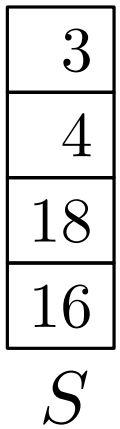
Stak



Push(S,28)

Pop(S) returnér 28

Stak

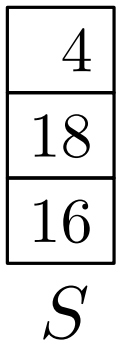


Push(S,28)

Pop(S) returnér 28

Pop(S)

Stak

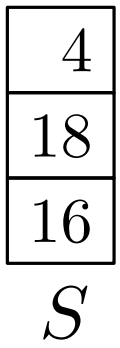


Push(S,28)

Pop(S) returnér 28

Pop(S) returnér 3

Stak



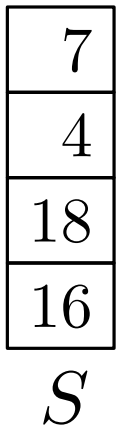
Push(S,28)

Pop(S) returnér 28

Pop(S) returnér 3

Push(S,7)

Stak



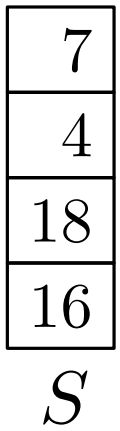
Push(S,28)

Pop(S) returnér 28

Pop(S) returnér 3

Push(S,7)

Stak



Push(S,28)

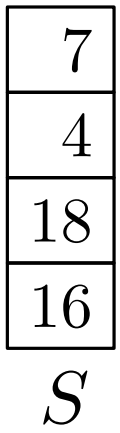
Pop(S) returnér 28

Pop(S) returnér 3

Push(S,7)

Is-Empty(S)

Stak



Push(S,28)

Pop(S) returnér 28

Pop(S) returnér 3

Push(S,7)

Is-Empty(S) returnér false

Hvordan ser stakken ud til sidst?

9
1
48
16

S

Pop(S)

Push(S,28)

Pop(S)

Is-Empty(S)

Pop(S)

Push(S,5)

socrative.com → Student login,
Room name: ABRAHAMSEN3464

9
1
48
16

A

5
48
16

B

5
1
48
16

C

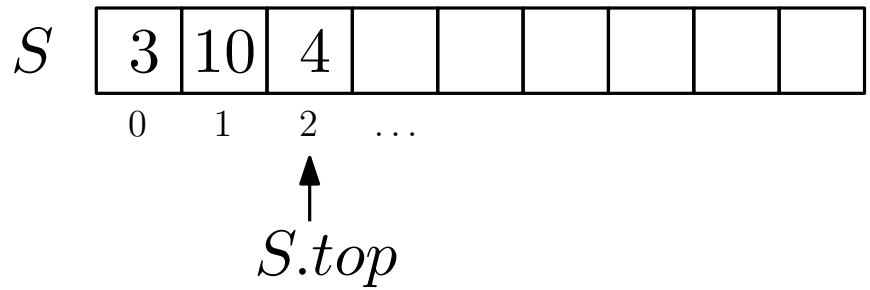
5
16

D

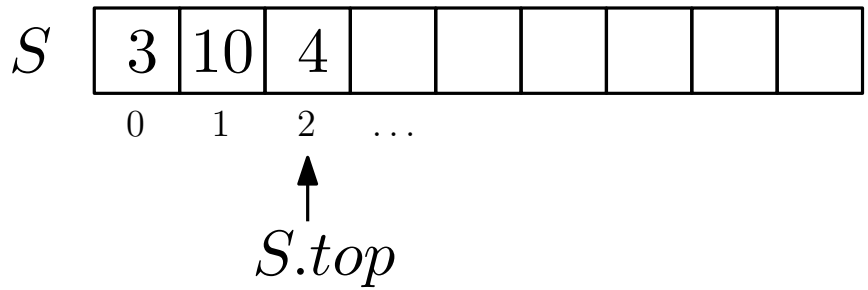
48
16
5

E

Implementation af stak med array

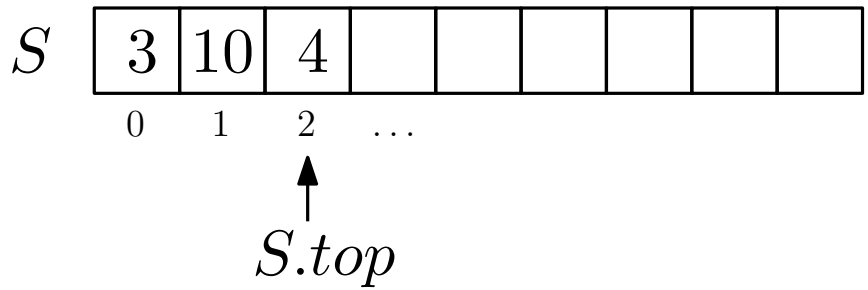


Implementation af stak med array




```
Is-Empty(S)
    return S.top == −1
```

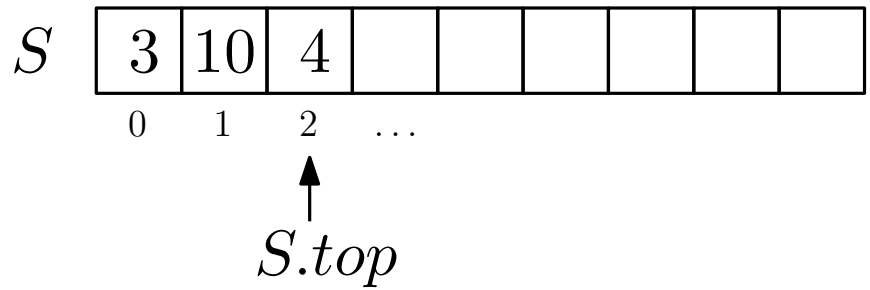
Implementation af stak med array



Is-Empty(S)
    return $S.top == -1$

Push(S, x)
    $S.top = S.top + 1$
    $S[S.top] = x$

Implementation af stak med array

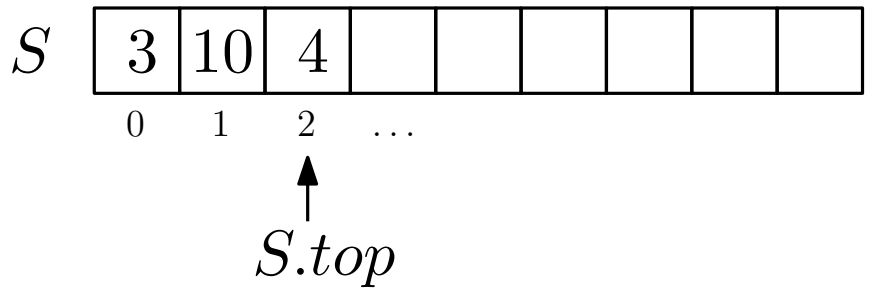


Is-Empty(S)
    return $S.top == -1$

Push(S, x)
    $S.top = S.top + 1$
    $S[S.top] = x$

Pop(S)
    $S.top = S.top - 1$
    return $S[S.top + 1]$

Implementation af stak med array



Is-Empty(S)
  return $S.top == -1$

Push(S, x)
  $S.top = S.top + 1$
  $S[S.top] = x$

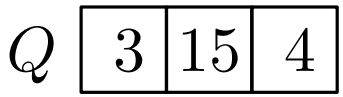
Køretid: $\Theta(1)$

Pop(S)
  $S.top = S.top - 1$
  return $S[S.top + 1]$

Kø

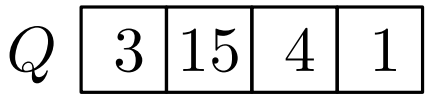
Q | 3 | 15 | 4 |

Kø



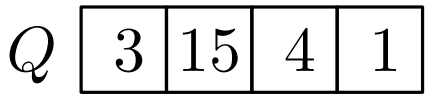
Enqueue$(Q, 1)$

Kø



Enqueue$(Q, 1)$

Kø



Enqueue$(Q, 1)$

Enqueue$(Q, 22)$

Kø

Q	3	15	4	1	22

Enqueue$(Q, 1)$

Enqueue$(Q, 22)$

Kø

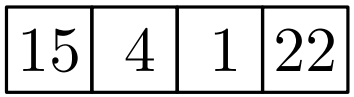
Q	3	15	4	1	22

Enqueue($Q, 1$)

Enqueue($Q, 22$)

Dequeue(Q)

Kø

Q 

Enqueue$(Q, 1)$

Enqueue$(Q, 22)$

Dequeue(Q) returnér 3

Kø

Q | 15 | 4 | 1 | 22 |

Enqueue$(Q, 1)$

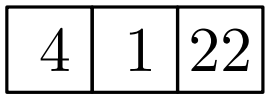
Enqueue$(Q, 22)$

Dequeue(Q) returnér 3

Dequeue(Q)

Kø

Q



Enqueue$(Q, 1)$

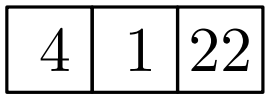
Enqueue$(Q, 22)$

Dequeue(Q) returnér 3

Dequeue(Q) returnér 15

Kø

Q



Enqueue$(Q, 1)$

Enqueue$(Q, 22)$

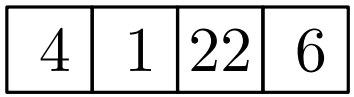
Dequeue(Q) returnér 3

Dequeue(Q) returnér 15

Enqueue$(Q, 6)$

Kø

Q



Enqueue($Q, 1$)

Enqueue($Q, 22$)

Dequeue(Q) returnér 3

Dequeue(Q) returnér 15

Enqueue($Q, 6$)

Kø

Q

4	1	22	6

Enqueue$(Q, 1)$

Enqueue$(Q, 22)$

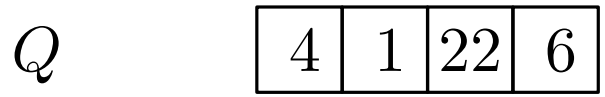
Dequeue(Q) returnér 3

Dequeue(Q) returnér 15

Enqueue$(Q, 6)$

Is-Empty(Q)

Kø



Enqueue($Q, 1$)

Enqueue($Q, 22$)

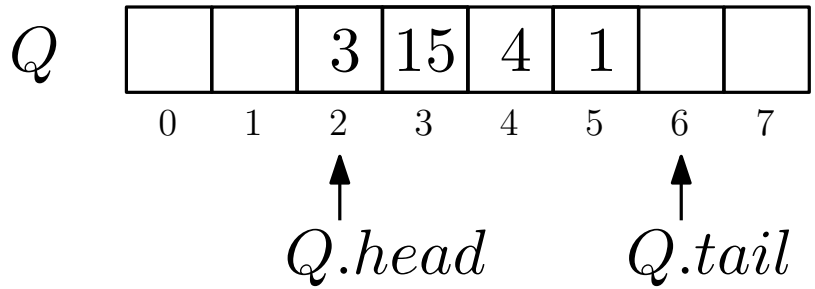
Dequeue(Q) returnér 3

Dequeue(Q) returnér 15

Enqueue($Q, 6$)

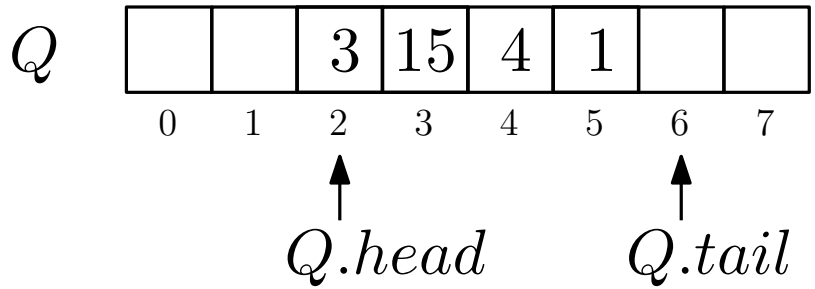
Is-Empty(Q) returnér false

Implementation af kø med array



$Q.N$: størrelse af array

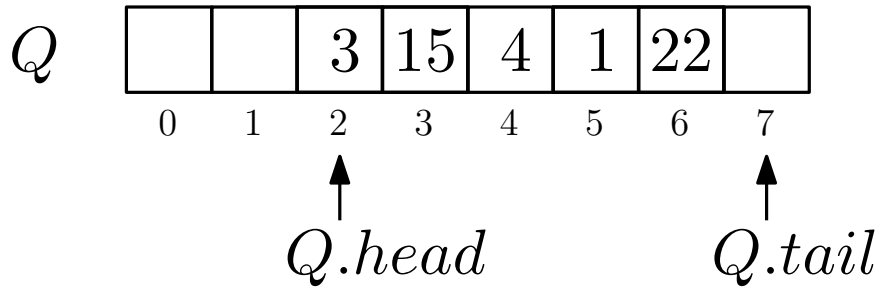
Implementation af kø med array



$Q.N$: størrelse af array

Enqueue$(Q, 22)$

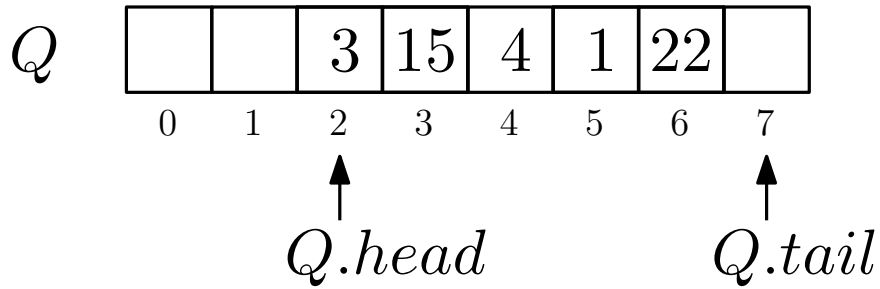
Implementation af kø med array



$Q.N$: størrelse af array

Enqueue$(Q, 22)$

Implementation af kø med array

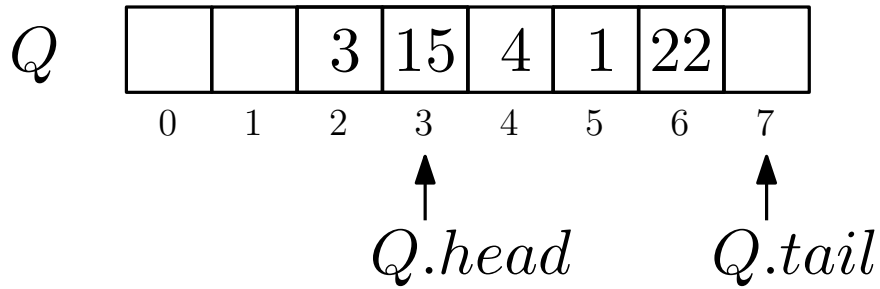


$Q.N$: størrelse af array

Enqueue($Q, 22$)

Dequeue(Q)

Implementation af kø med array

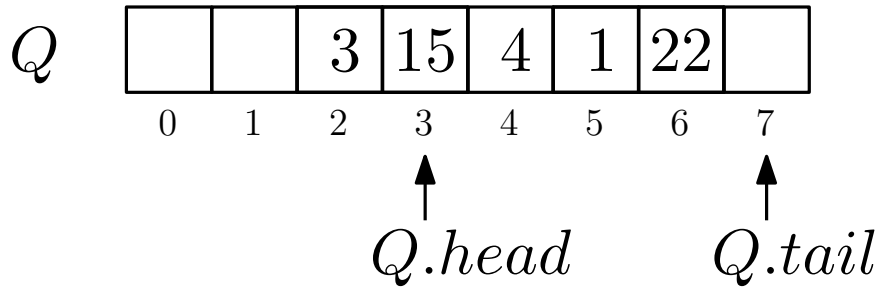


$Q.N$: størrelse af array

Enqueue$(Q, 22)$

Dequeue(Q) returnér 3

Implementation af kø med array



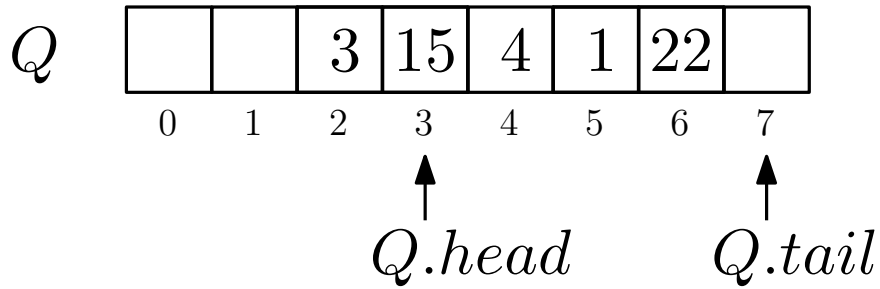
$Q.N$: størrelse af array

Enqueue$(Q, 22)$

Dequeue(Q) returnér 3

Enqueue(Q, x)
  $Q[Q.tail] = x$
  if $Q.tail < Q.N - 1$
    $Q.tail = Q.tail + 1$
  else
    $Q.tail = 0$

Implementation af kø med array



$Q.N$: størrelse af array

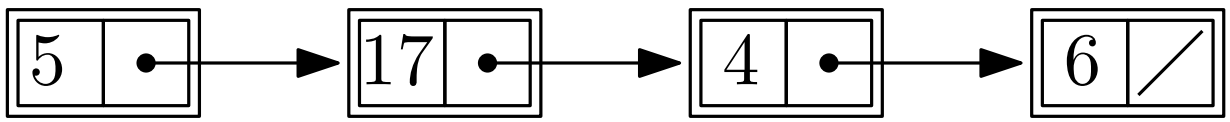
Enqueue$(Q, 22)$

Dequeue(Q) returnér 3

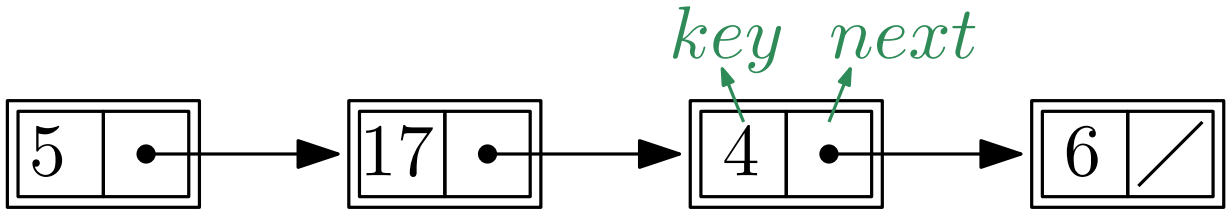
```
Enqueue(Q, x)
    Q[Q.tail] = x
    if Q.tail < Q.N − 1
        Q.tail = Q.tail + 1
    else
        Q.tail = 0
```

```
Dequeue(Q)
    x = Q[Q.head]
    if Q.head < Q.N − 1
        Q.head = Q.head + 1
    else
        Q.head = 0
    return x
```

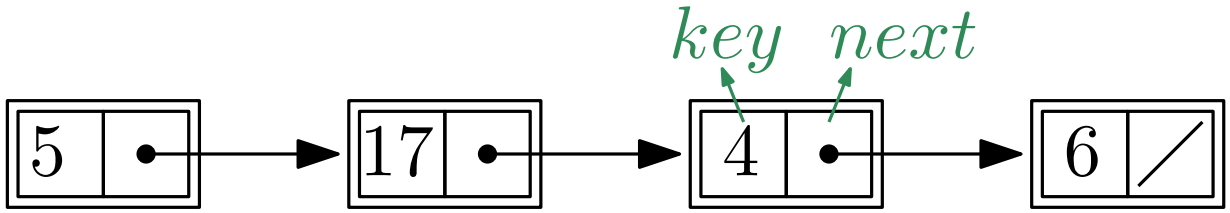
Enkelthægtet liste



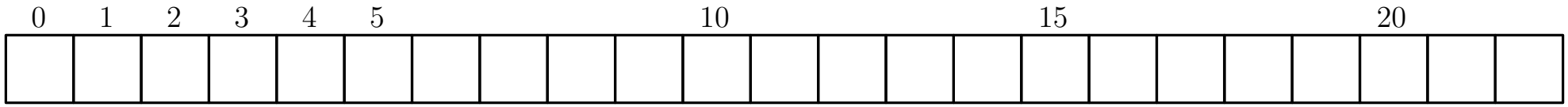
Enkelthægtet liste



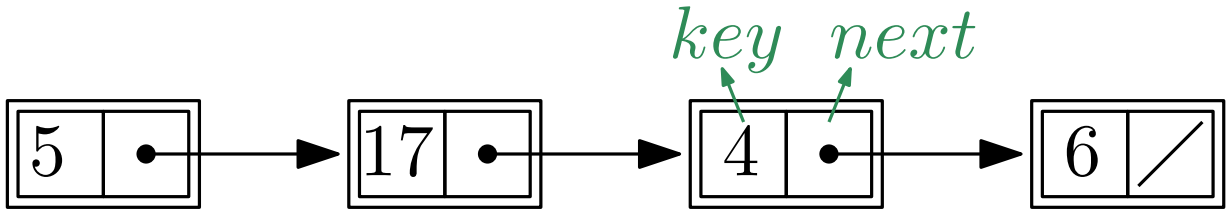
Enkelthægtet liste



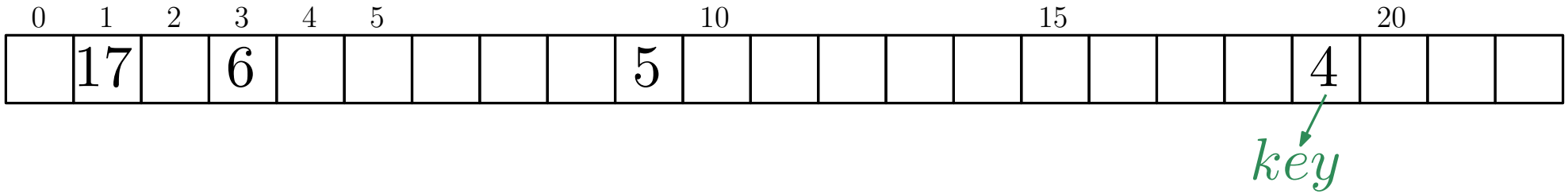
I hukommelsen:



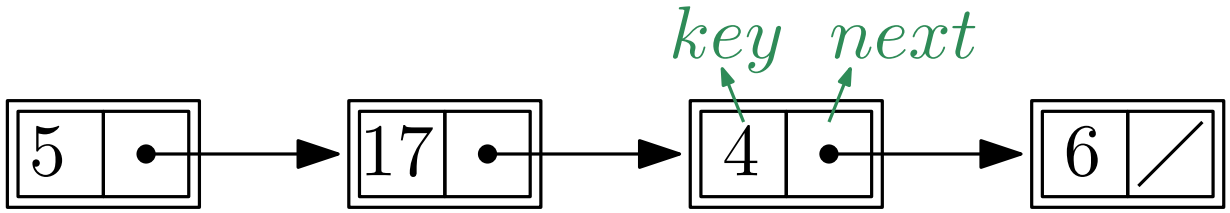
Enkelthægtet liste



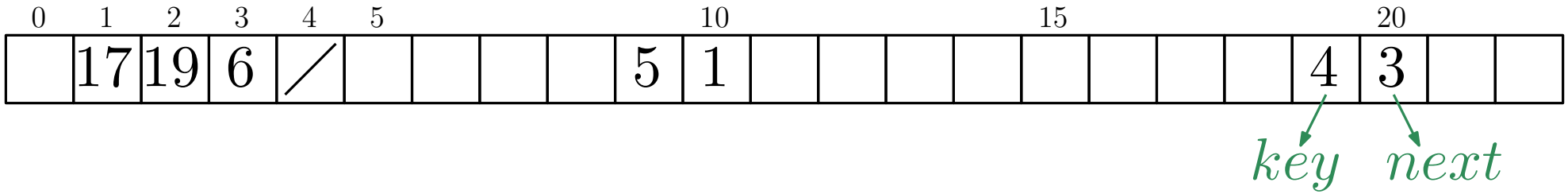
I hukommelsen:



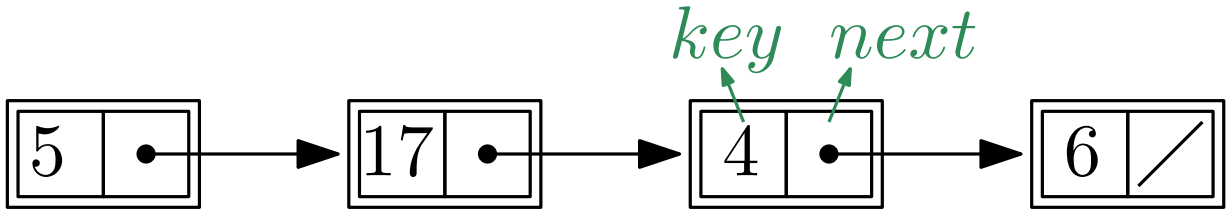
Enkelthægtet liste

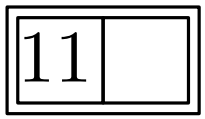


I hukommelsen:

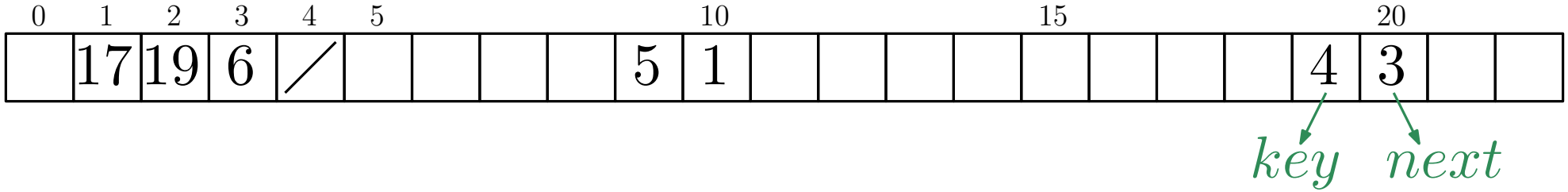


Enkelthægtet liste

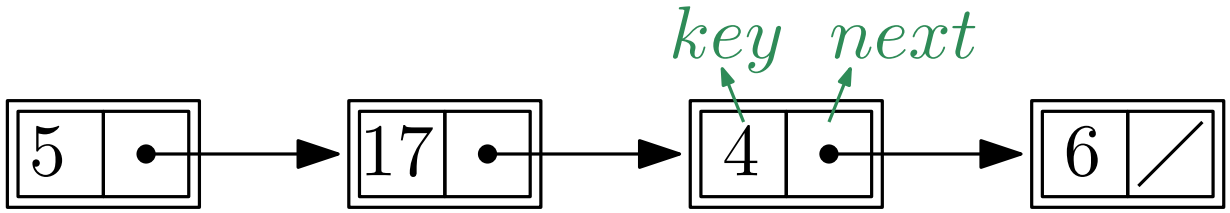


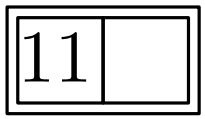


I hukommelsen:

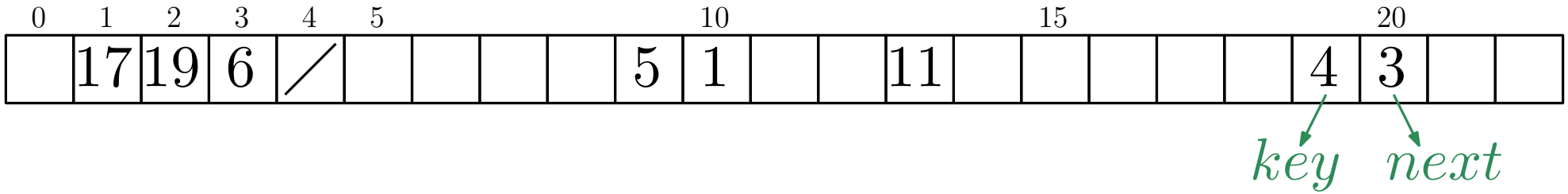


Enkelthægtet liste

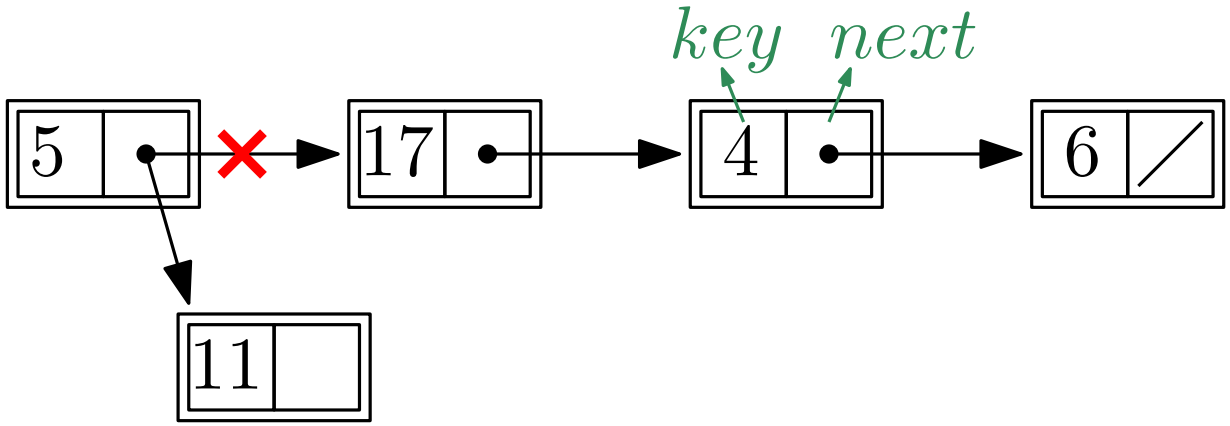




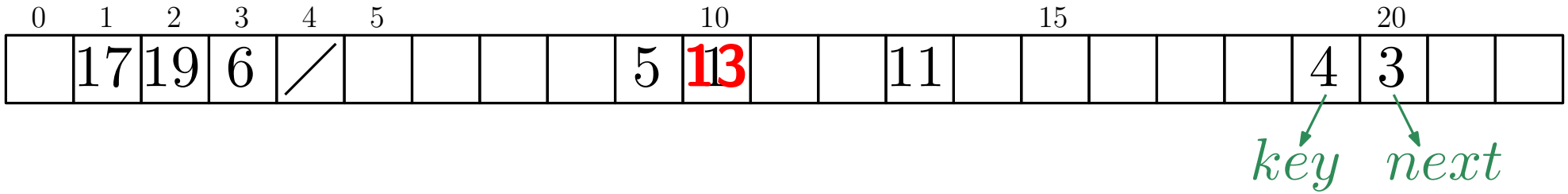
I hukommelsen:



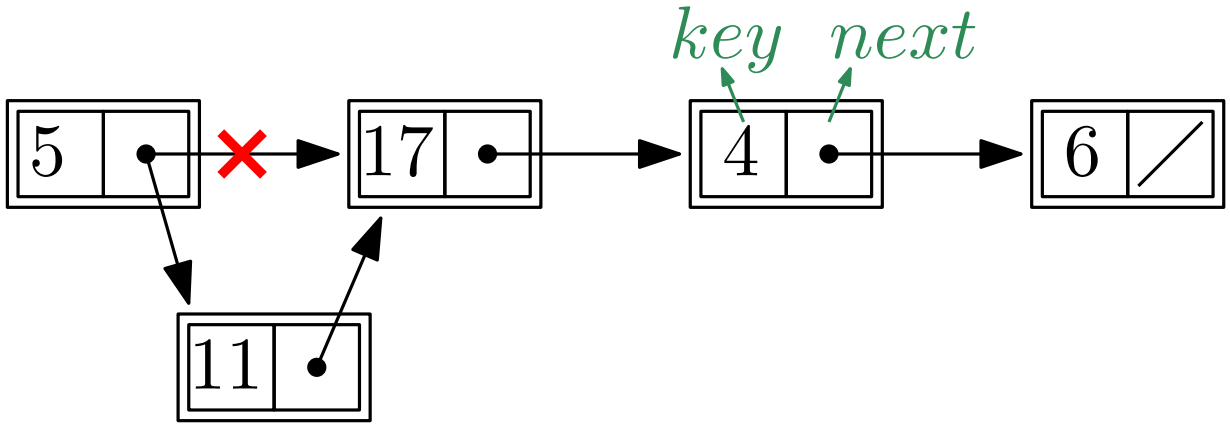
Enkelthægtet liste



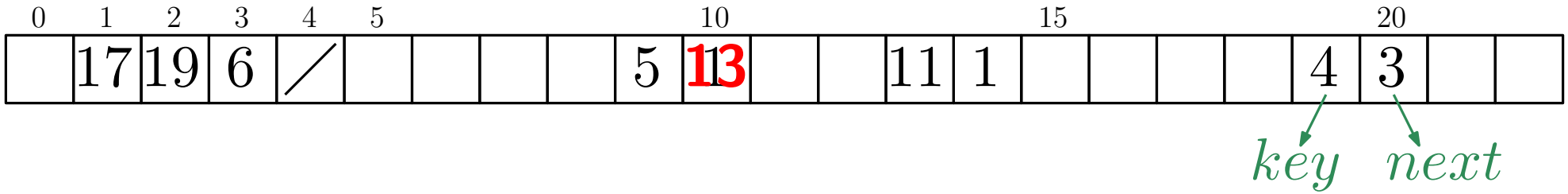
I hukommelsen:



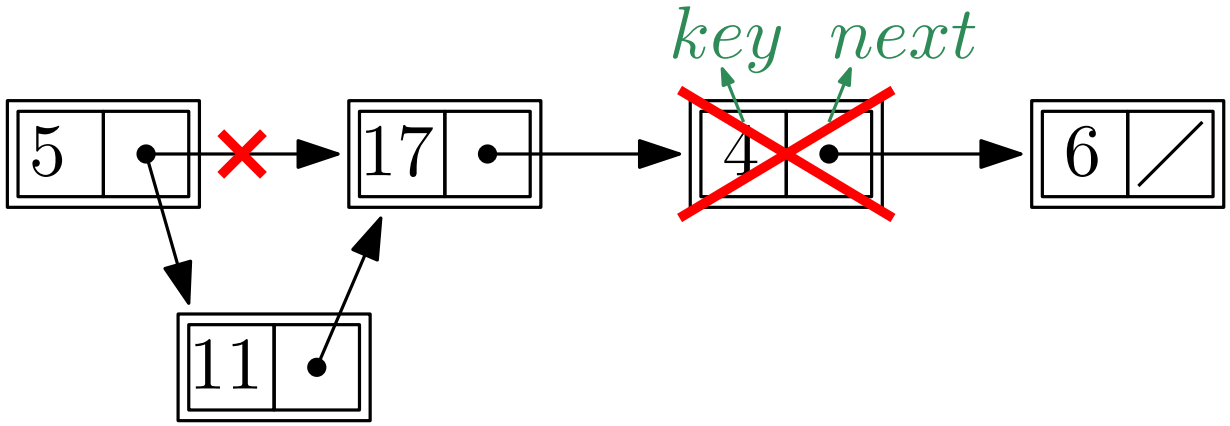
Enkelthægtet liste



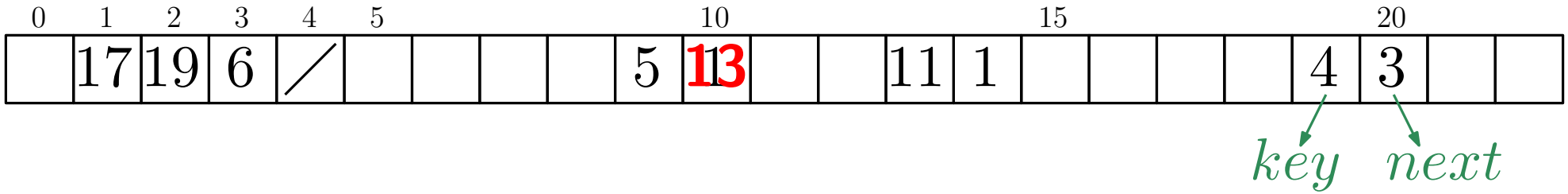
I hukommelsen:



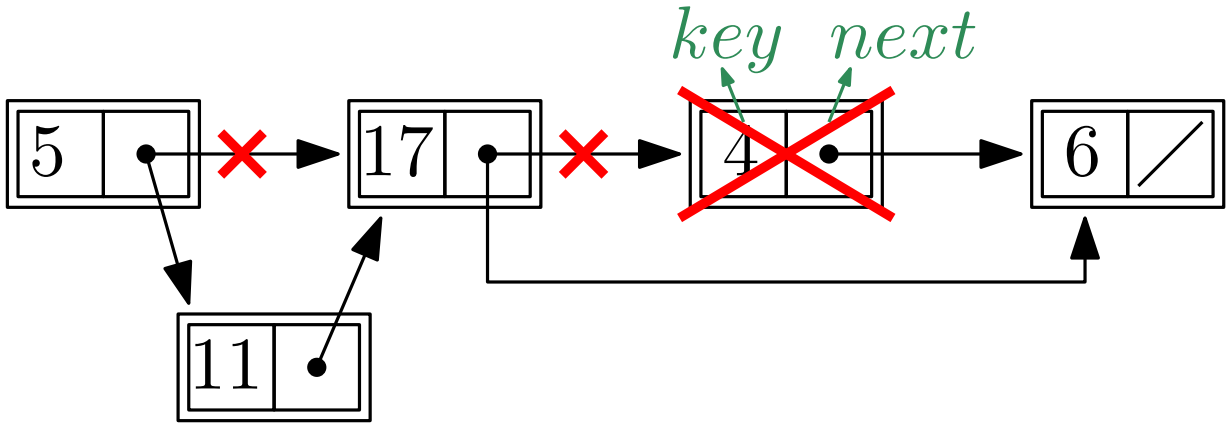
Enkelthægtet liste



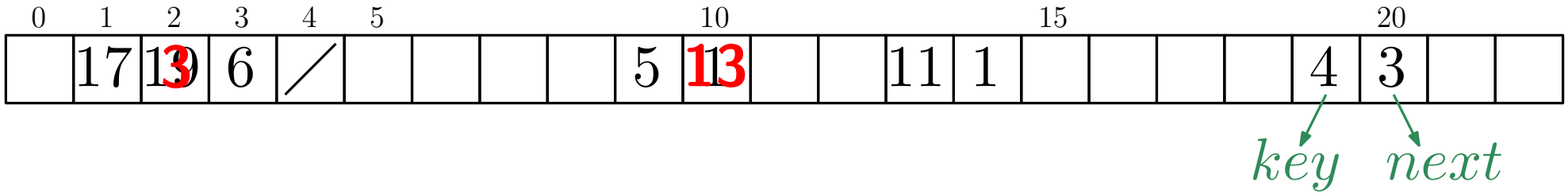
I hukommelsen:



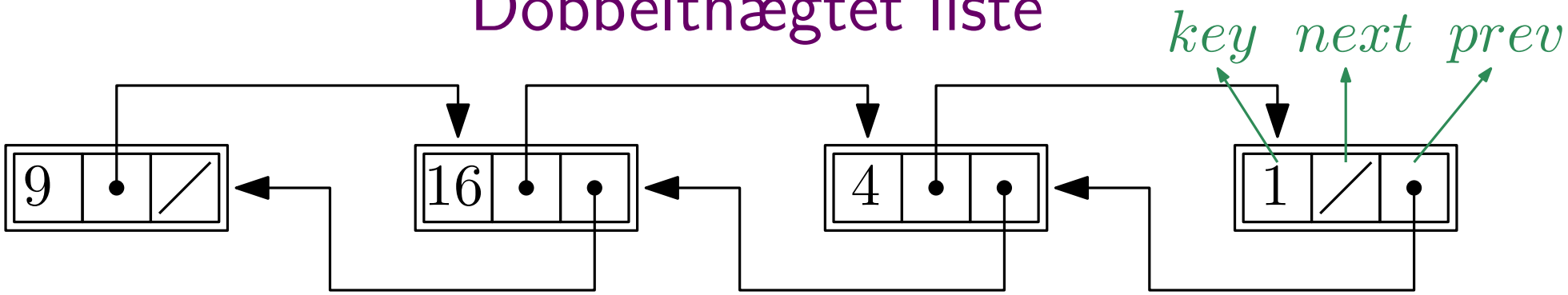
Enkelthægtet liste



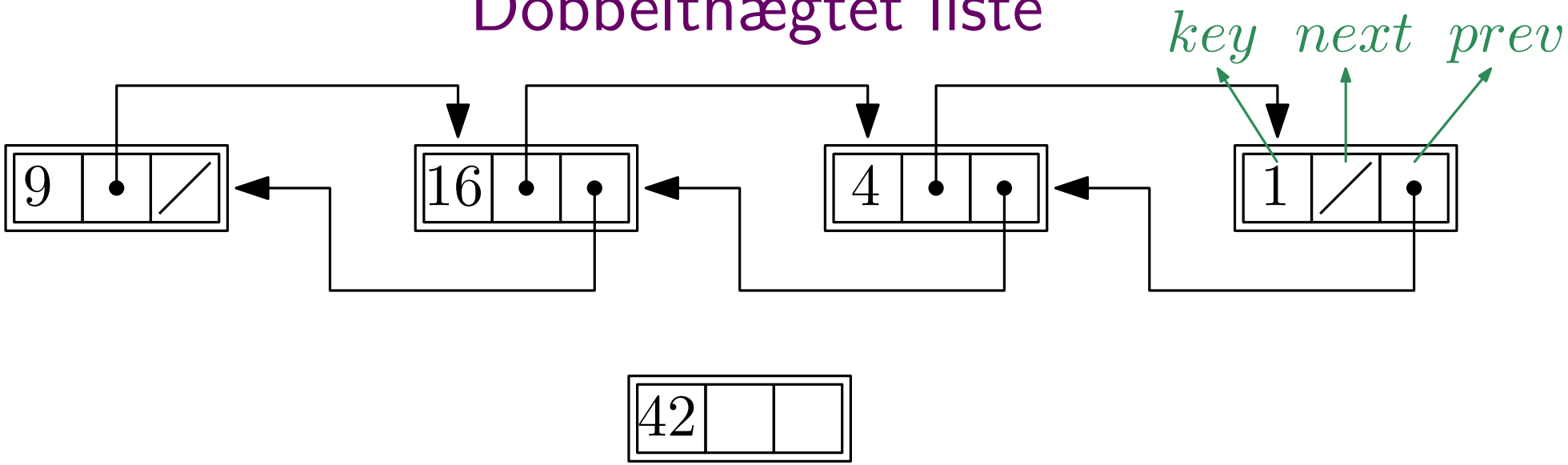
I hukommelsen:



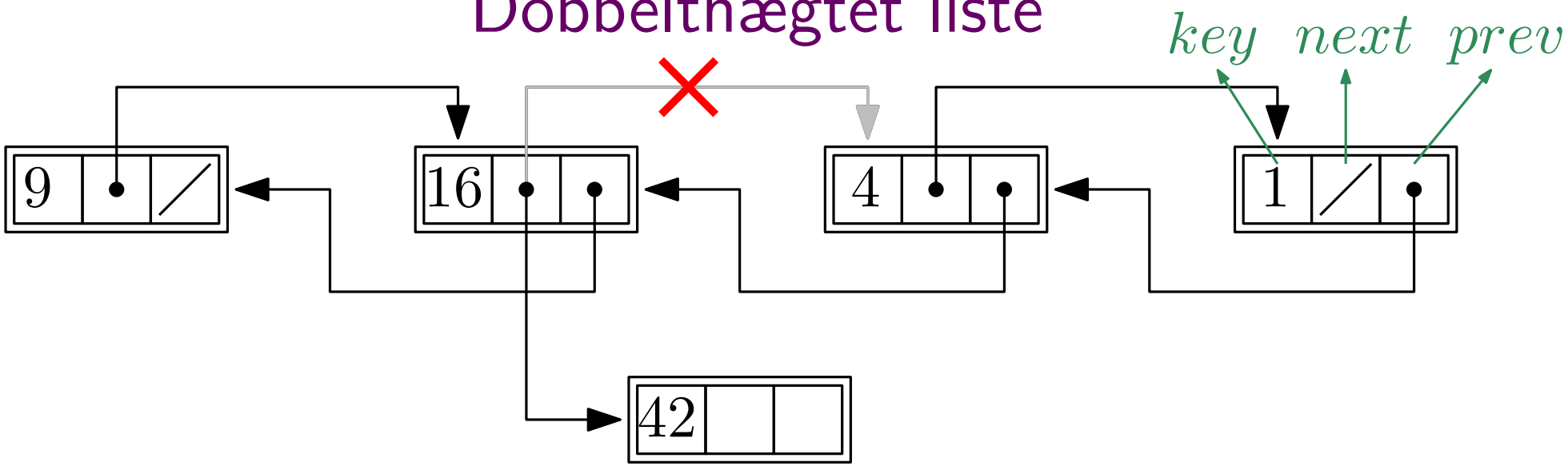
Dobbelthægtet liste



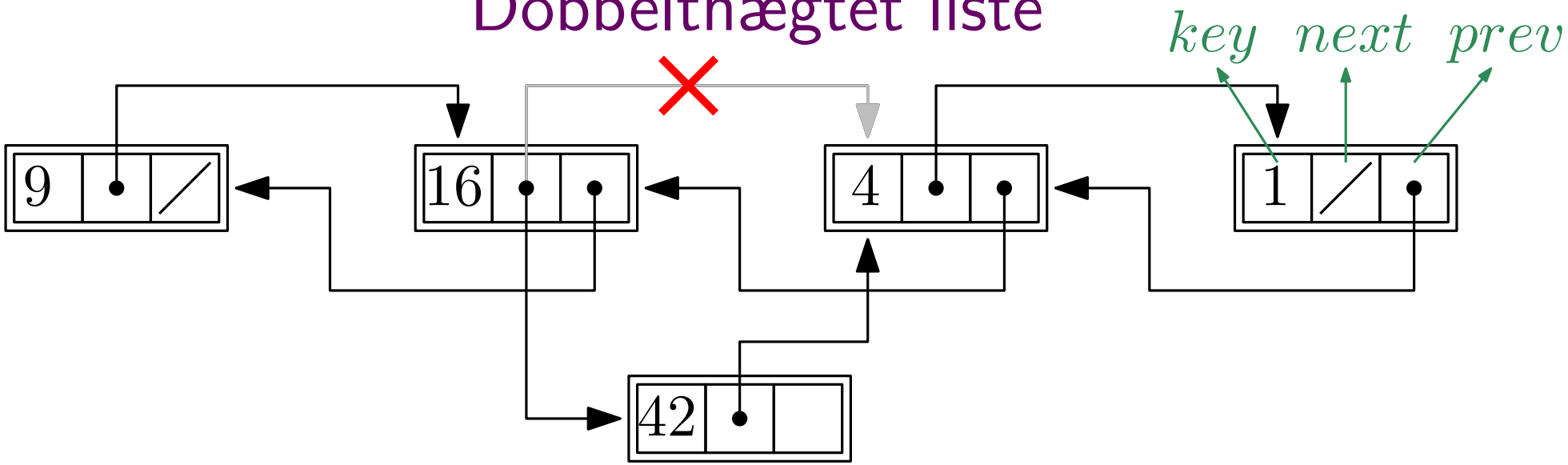
Dobbelthægtet liste



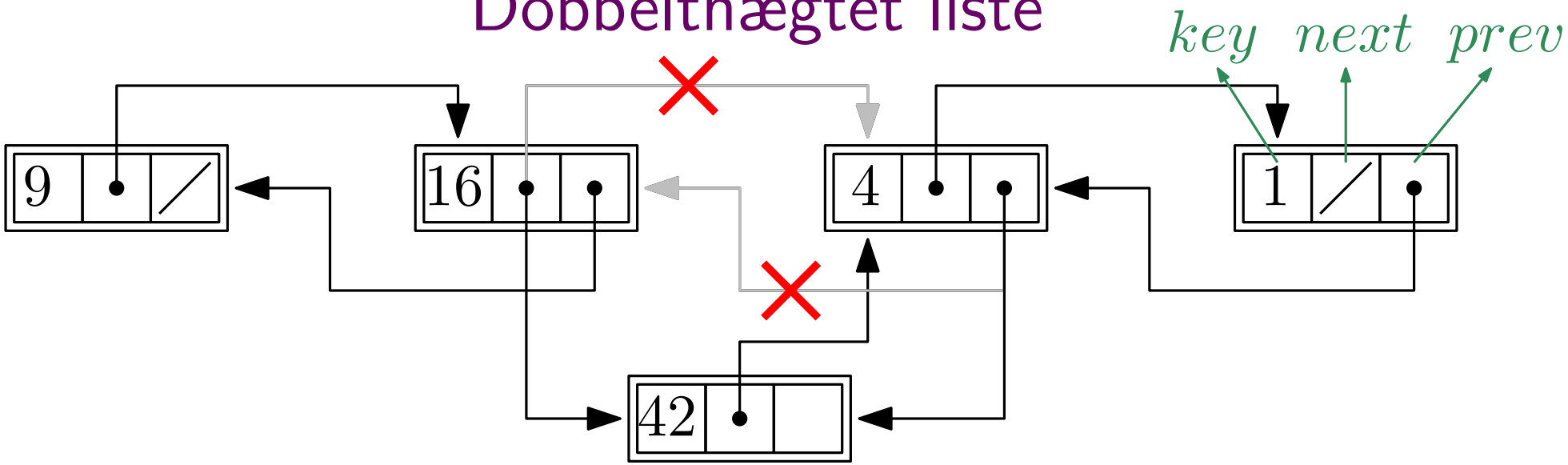
Dobbelthægtet liste



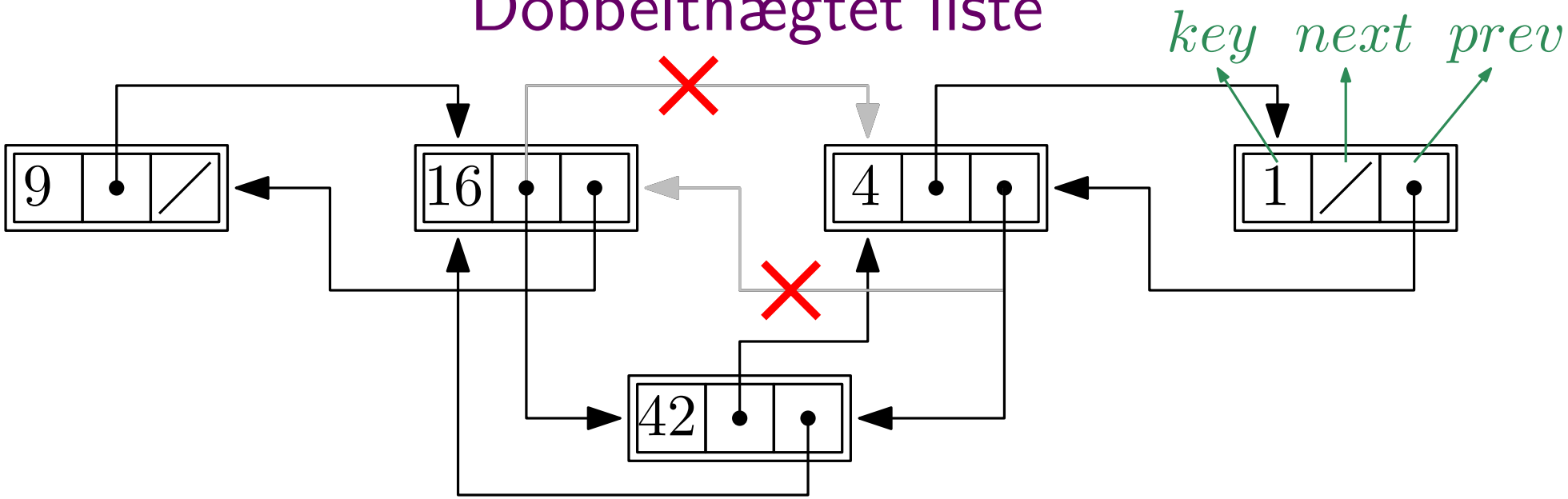
Dobbelthægtet liste



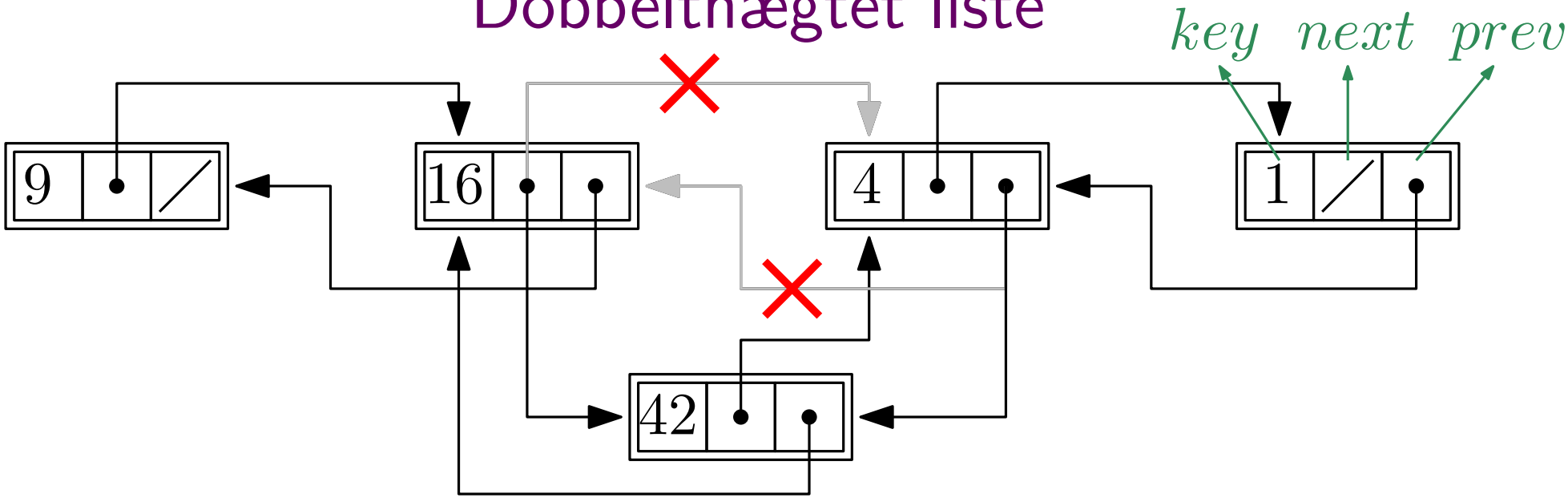
Dobbelthægtet liste

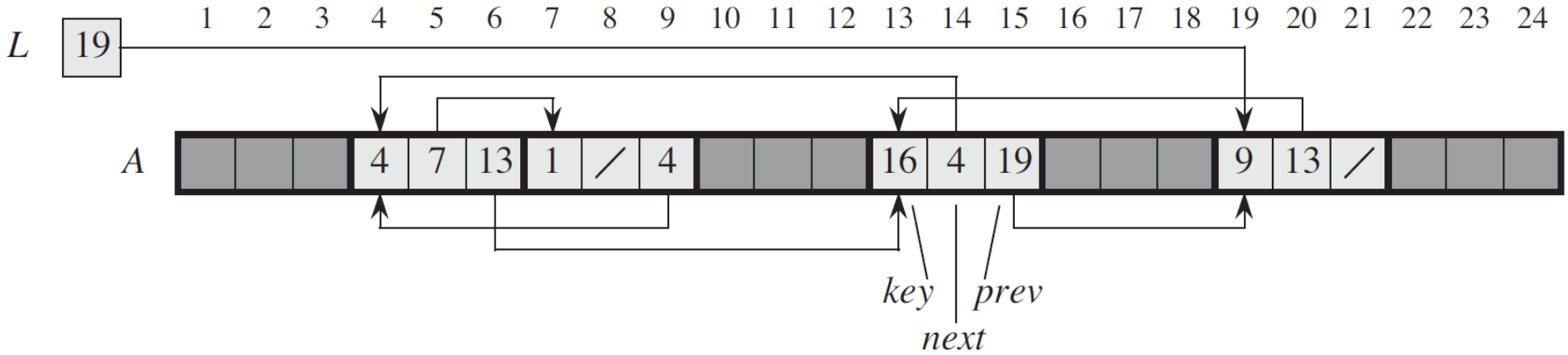


Dobbelthægtet liste

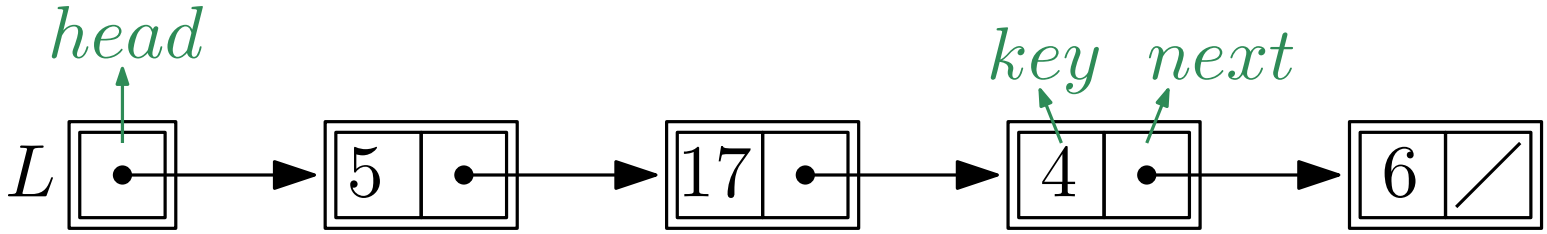


Dobbelthægtet liste

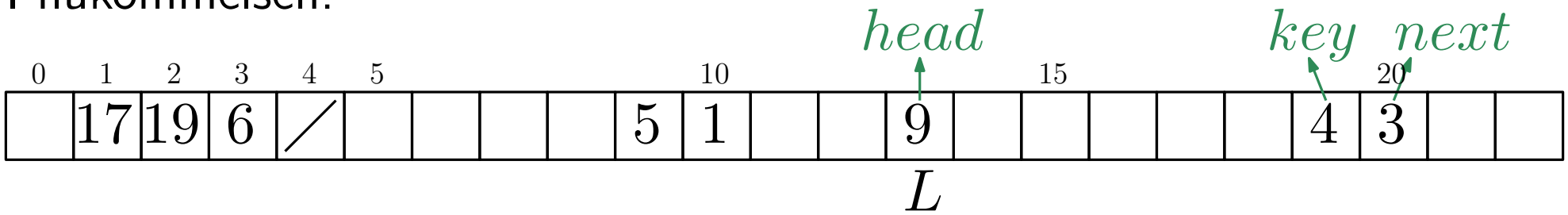




Enkelthægtet liste

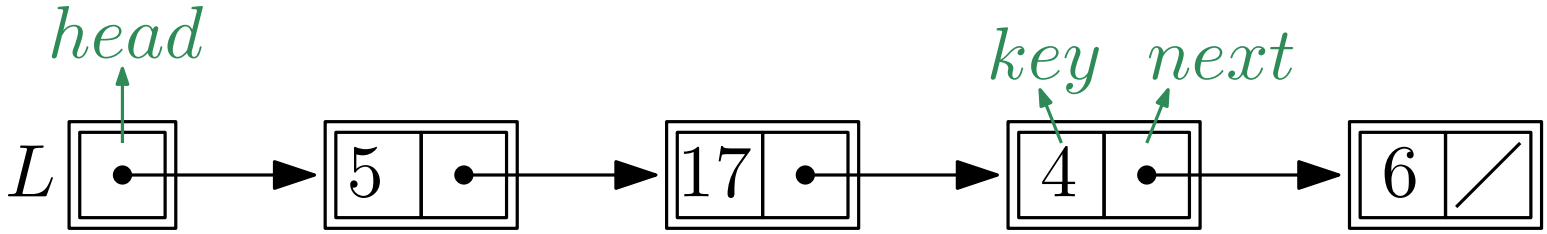


I hukommelsen:

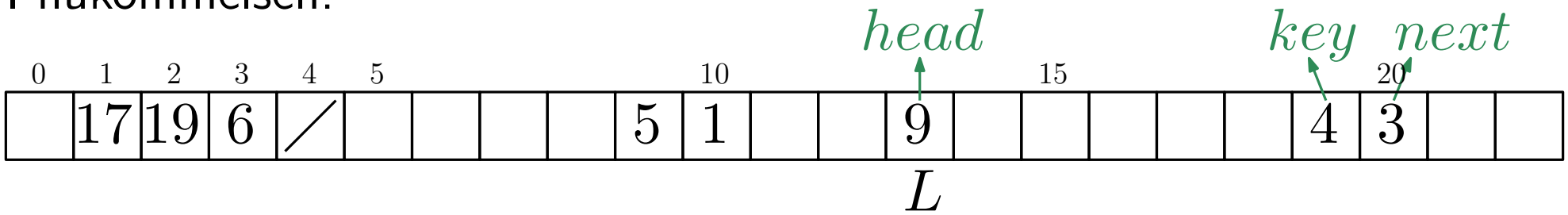


List-Search(L, k): Søg fra begyndelsen efter nøgleværdi k.

Enkelthægtet liste



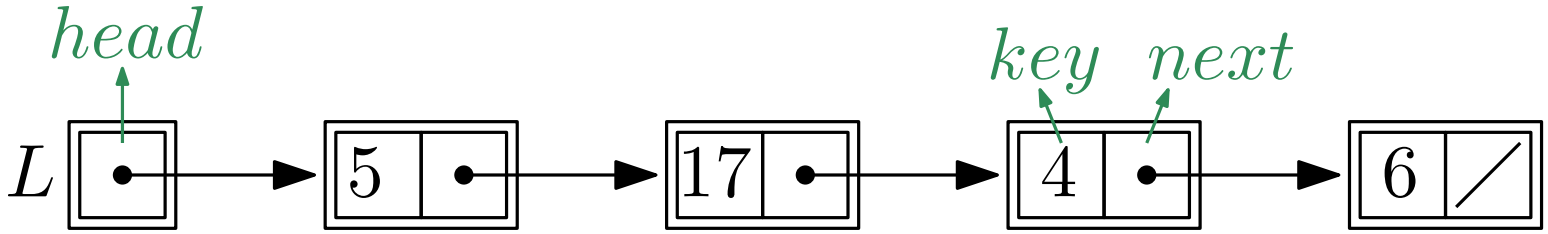
I hukommelsen:



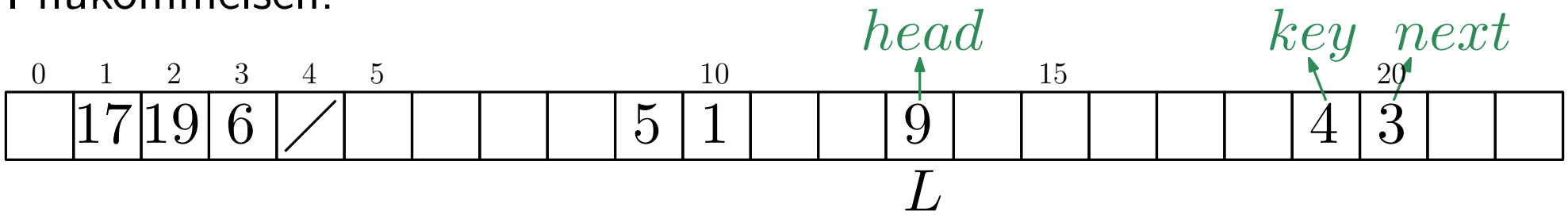
List-Search(L, k): Søg fra begyndelsen efter nøgleværdi k.

List-Search(L, k)
  $x = L.head$
  while $x \neq$ NIL and $x.key \neq k$
    $x = x.next$
  return x

Enkelthægtet liste



I hukommelsen:

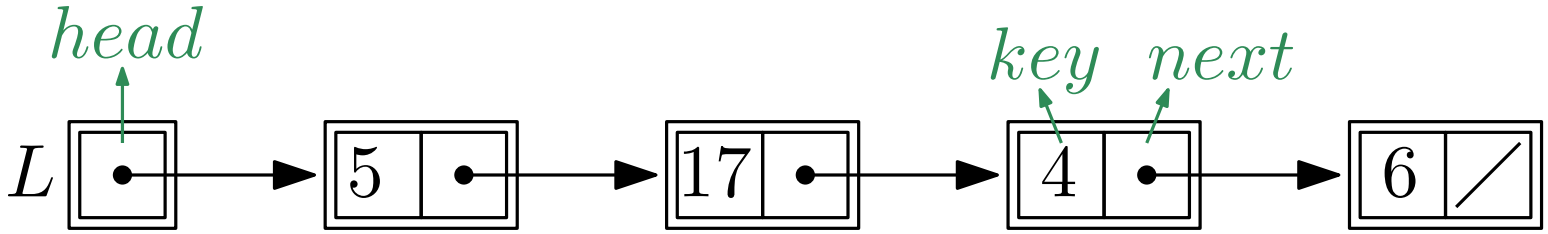


List-Search(L, k): Søg fra begyndelsen efter nøgleværdi k.

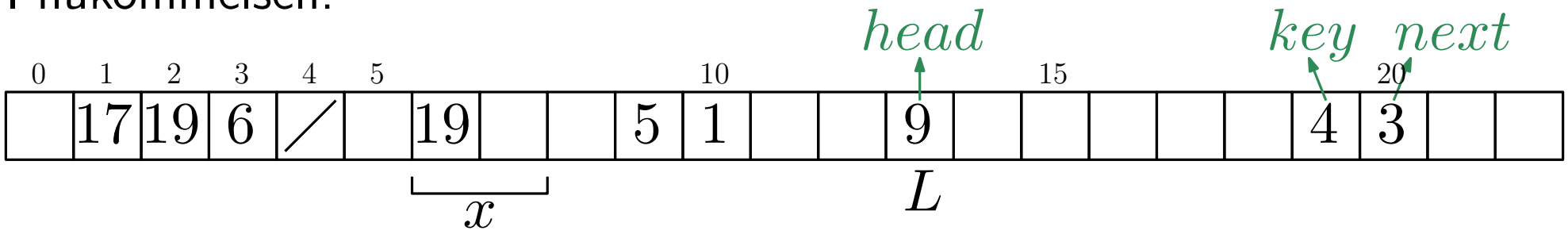
List-Search(L, k)
  $x = L.head$
  while $x \neq$ NIL and $x.key \neq k$
    $x = x.next$
  return x

$\Theta(n)$ tid.

Enkelthægtet liste

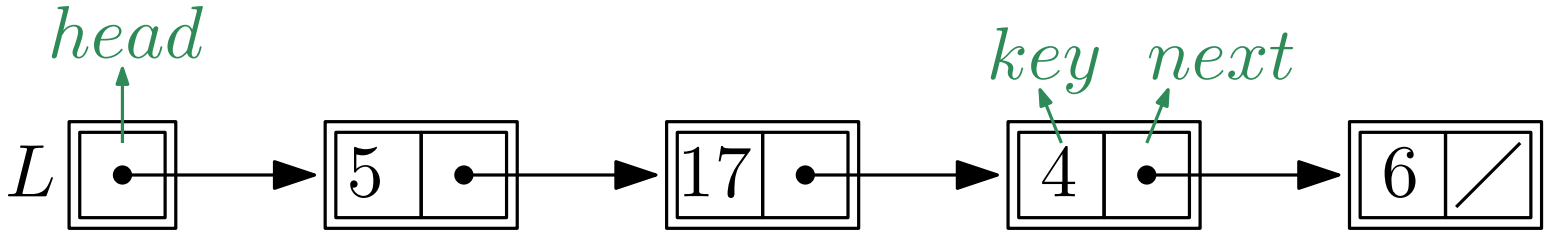


I hukommelsen:

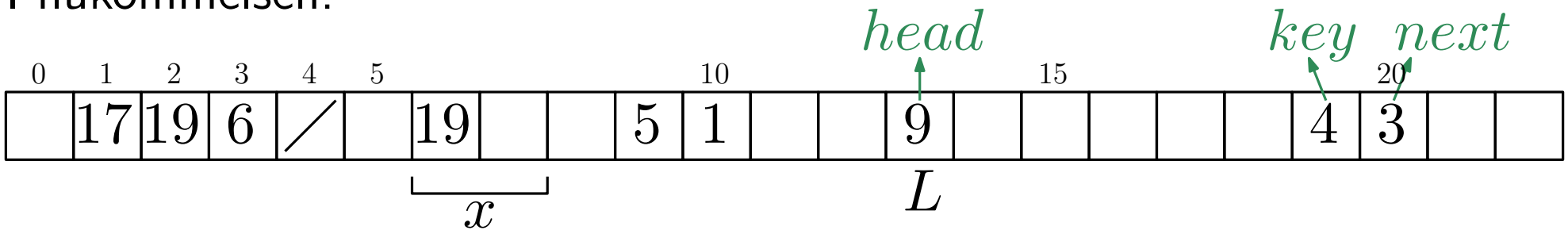


List-Insert(L, x): Indsæt listeelementet x i begyndelsen af L. $\Theta(1)$ tid.

Enkelthægtet liste



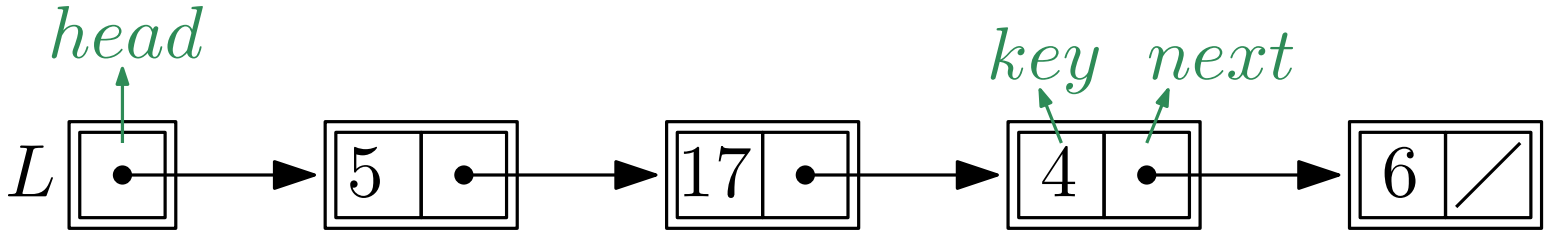
I hukommelsen:



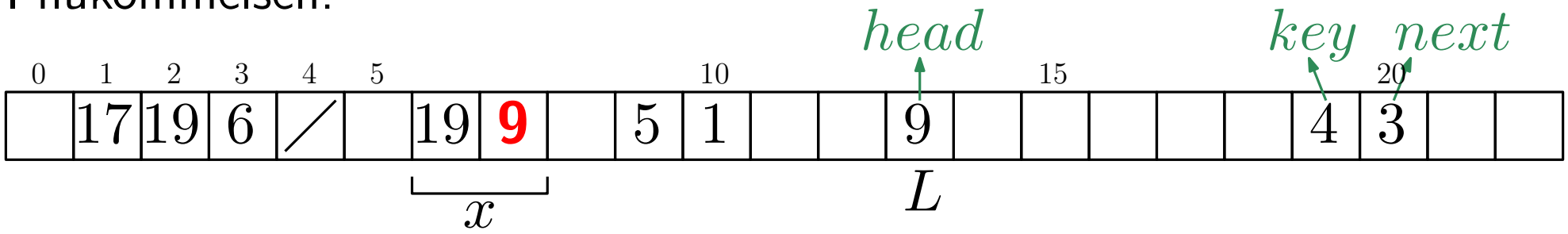
List-Insert(L, x): Indsæt listeelementet x i begyndelsen af L. $\Theta(1)$ tid.

List-Insert(L, x)
$x.next = L.head$
$L.head = x$

Enkelthægtet liste



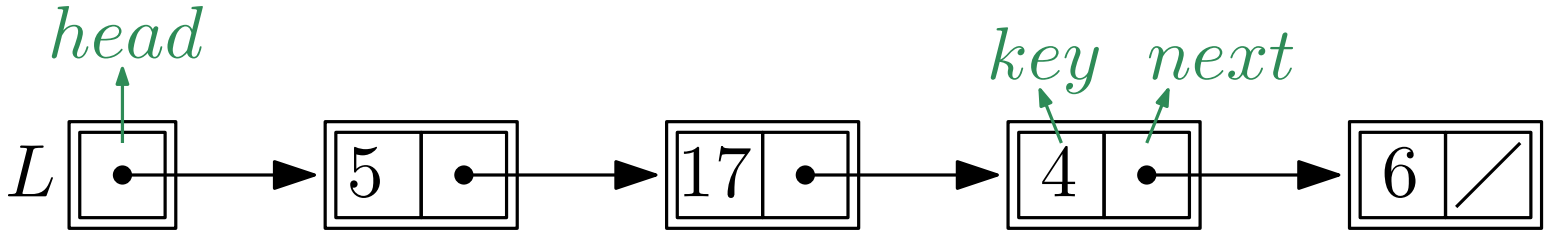
I hukommelsen:



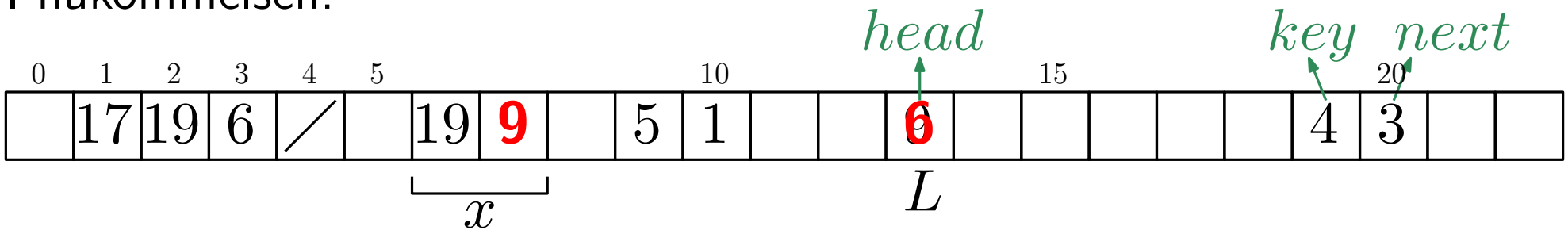
List-Insert(L, x): Indsæt listeelementet x i begyndelsen af L. $\Theta(1)$ tid.

List-Insert(L, x)
$x.next = L.head$
$L.head = x$

Enkelthægtet liste



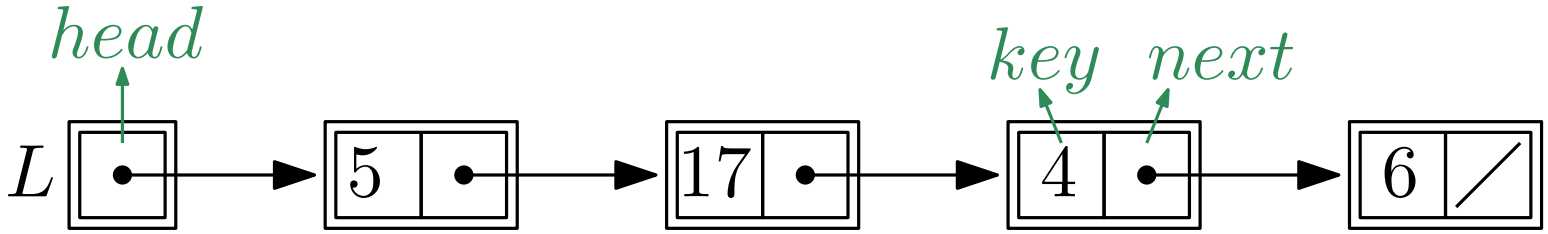
I hukommelsen:



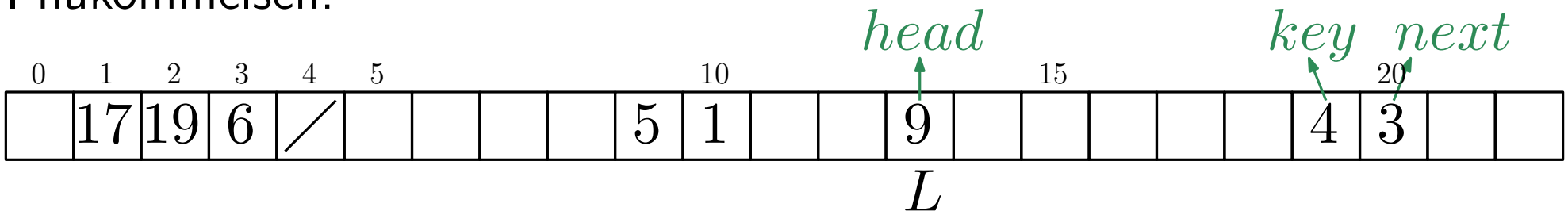
List-Insert(L, x): Indsæt listeelementet x i begyndelsen af L. $\Theta(1)$ tid.

List-Insert(L, x)
$x.next = L.head$
$L.head = x$

Operationer på hægtede lister



I hukommelsen:

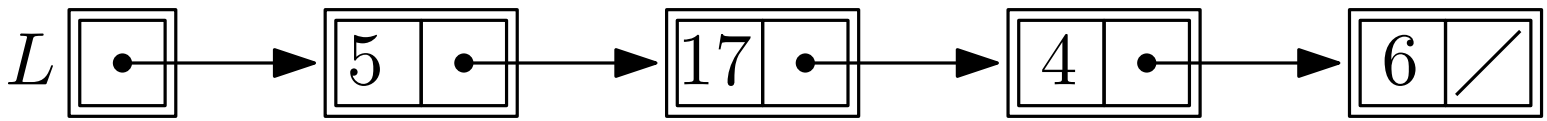


List-Search(L, k): Søg fra begyndelsen efter nøgleværdi k. $\Theta(n)$ tid.

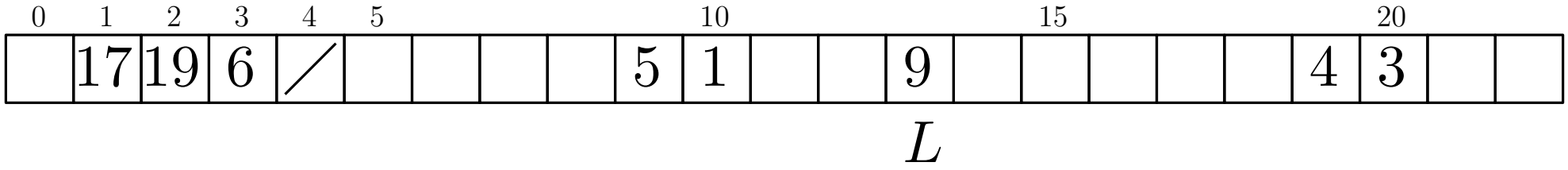
List-Insert(L, x): Indsæt listeelementet x i begyndelsen af L. $\Theta(1)$ tid.

List-Delete(L, x): Slet listeelementet x fra L. For dobbelthægtede lister: $\Theta(1)$ tid.

Enkelthægtet liste

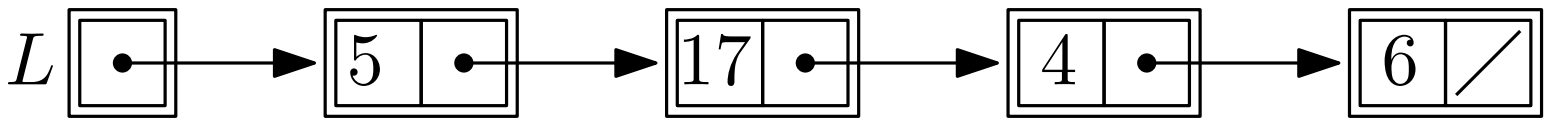


I hukommelsen:

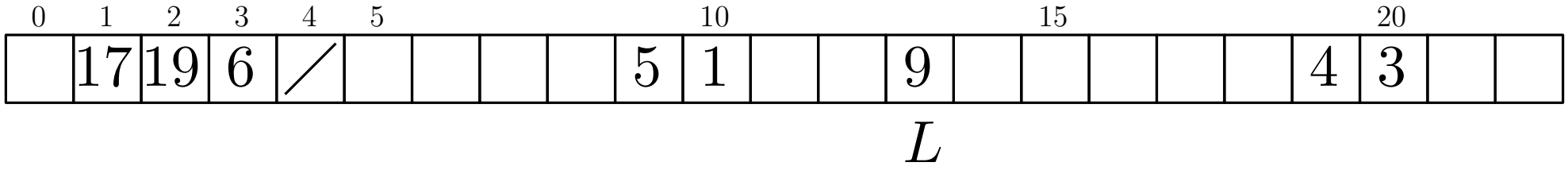


Tilgå felt nummer k i hukommelsen eller i array A tager $\Theta(1)$ tid.

Enkelthægtet liste



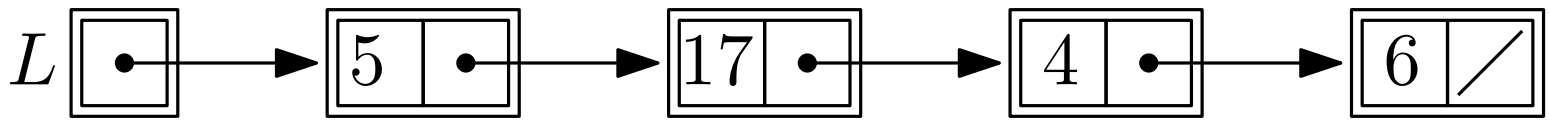
I hukommelsen:



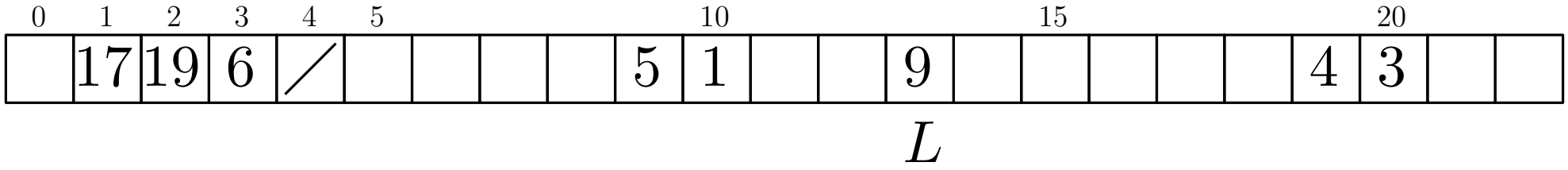
Tilgå felt nummer k i hukommelsen eller i array A tager $\Theta(1)$ tid.

Tilgå element nummer k i en liste tager $\Theta(k)$ tid.

Enkelthægtet liste



I hukommelsen:



Tilgå felt nummer k i hukommelsen eller i array A tager $\Theta(1)$ tid.

Tilgå element nummer k i en liste tager $\Theta(k)$ tid.

I F#:
`mylist.`[k] tager $\Theta(k)$ tid!

"Lists in F# are implemented as singly linked lists, which means that operations that access only the head of the list are $O(1)$, and element access is $O(n)$."