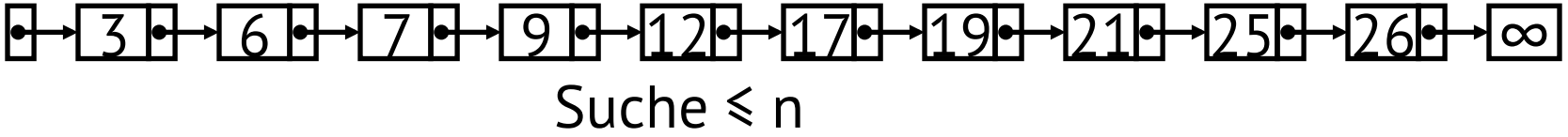
2.5 Bäume

Idee

- Worst-Case Aufwand für Search(), Insert(), Delete() bei Rot-Schwarz-Bäumen ist O(log n)
- Schwer zu implementieren

Idee

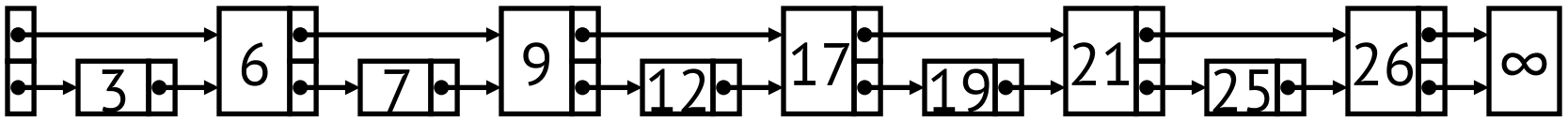
- Idee: Halte eine einfach verkettete sortierte Liste



Suche $\leqslant n$

Idee

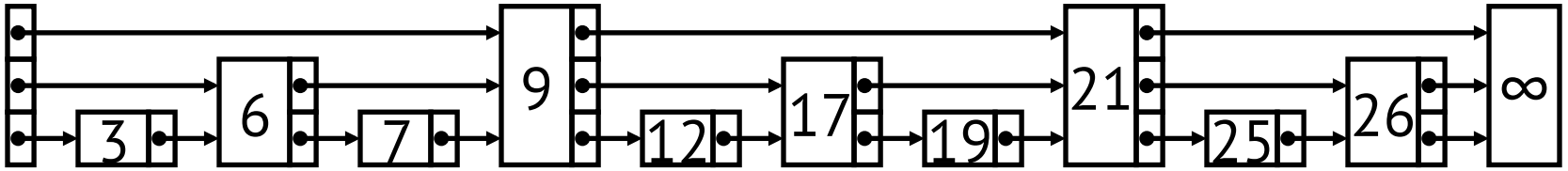
- Idee: Halte eine einfach verkettete sortierte Liste
- Füge Abkürzungen zwischen den Elementen ein



Suche $\leqslant \lceil n/2 \rceil + 1$

Idee

- Idee: Halte eine einfach verkettete sortierte Liste
- Füge Abkürzungen zwischen den Elementen ein
- Füge Abkürzungen zwischen den Abkürzungen ein



Suche $\leqslant \lceil n/4 \rceil + 2$

Definition

- Skip-Listen sind sortierte, vorwärts-verkettete Listen
- Jeder Knoten k enthält level(k) ⩾ 1 Vorwärtszeiger
- Der i-te Vorwärtszeiger eines Knotens k (1 ⩽ i ⩽ level(k)) zeigt auf den nächsten Knoten k' mit level(k') ⩾ i oder auf NIL
- Der Level eines Knotens wir bei der Erzeugung des Knotens zufällig festgelegt und danach nicht mehr geändert!

Definition

- Erzeugung eines zufälligen Levels
- RandomLevel()

```
v ← 1
while random() < p and v < MaxLevel do
    v ← v + 1
return v
```

- Level-Verteilung für p = 1 / 2
  - ≈ 50% Level 1
  - ≈ 25% Level 2
  - …

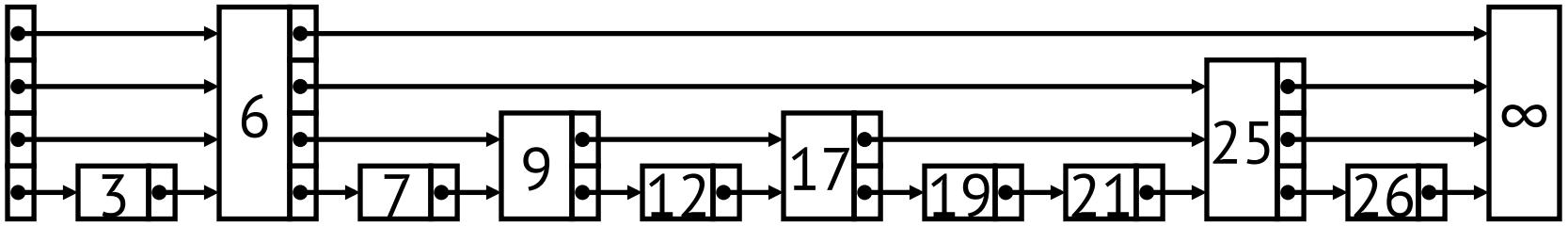
Search

- Search(list, search_key)

```
    x ← list.head
    for i ← list.level downto 1 do
        while x.next( i ).key < search_key do
            x ← x.next( i )
    x ← x.next( i )
    if x.key = search_key then
        return x
    else
        return not found;
```

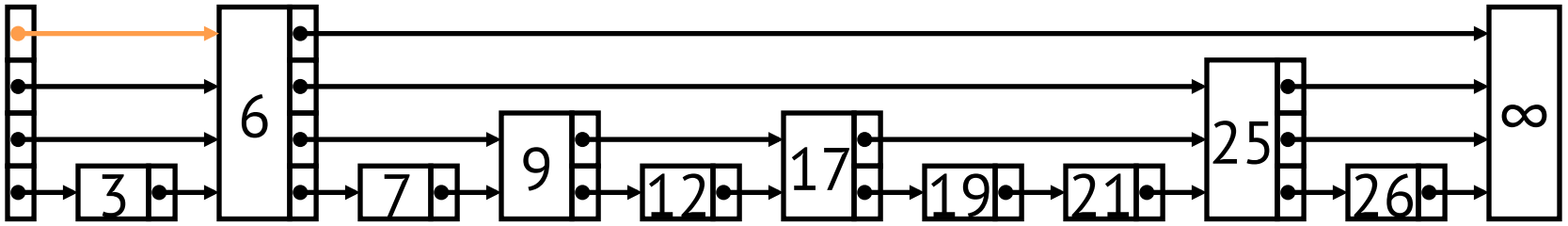
Search

Search(19)



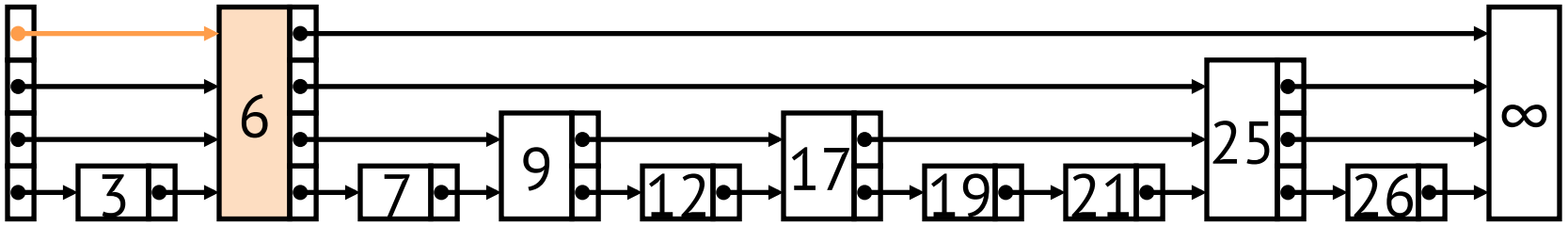
Search

Search(19)



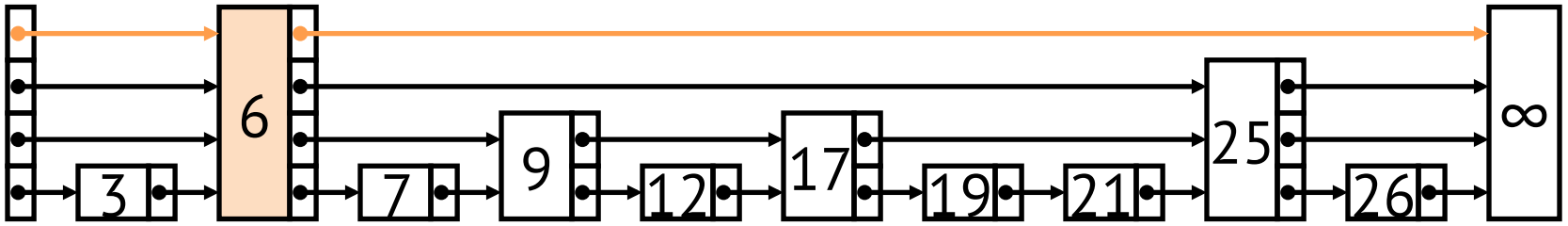
Search

Search(19)



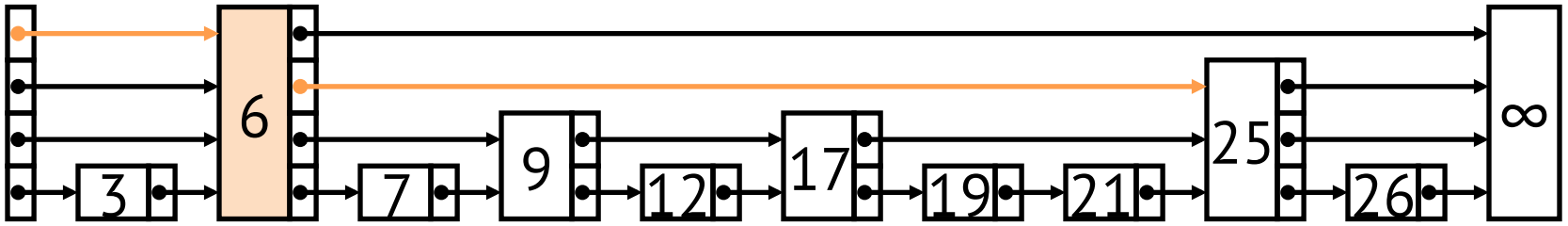
Search

Search(19)



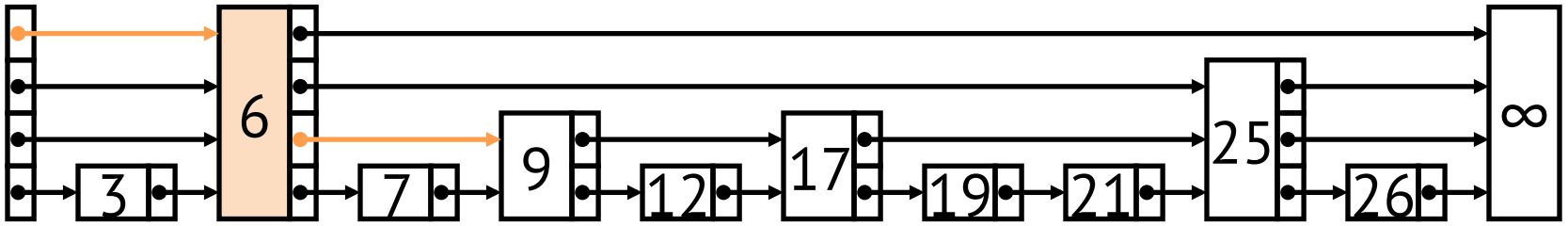
Search

Search(19)



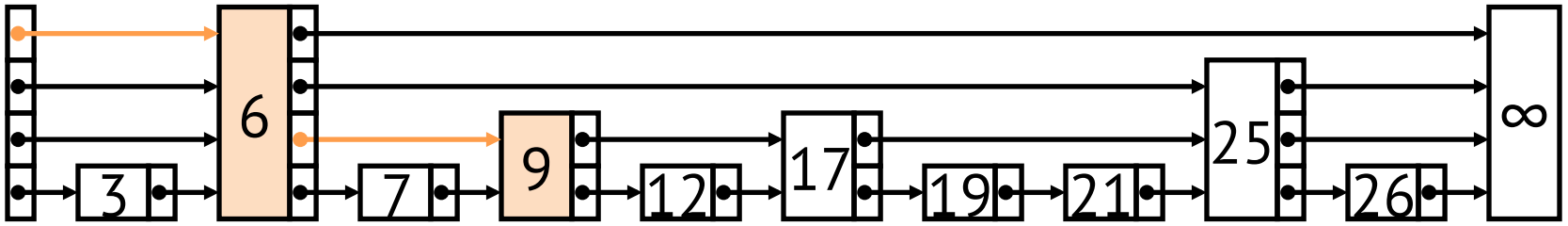
Search

Search(19)



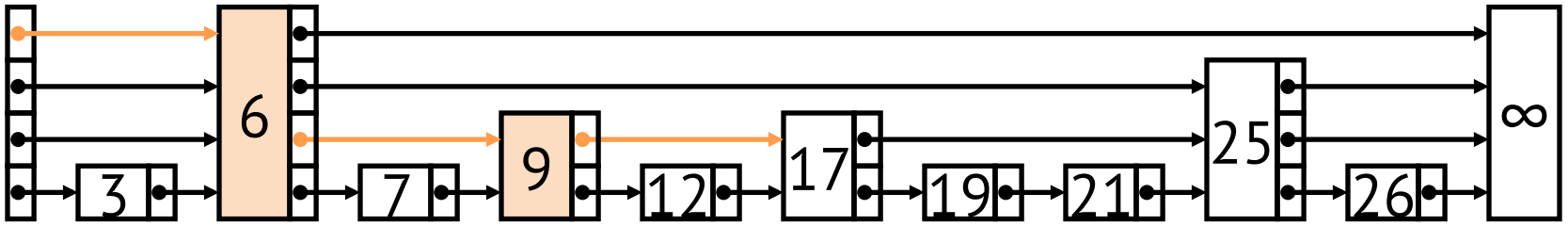
Search

Search(19)



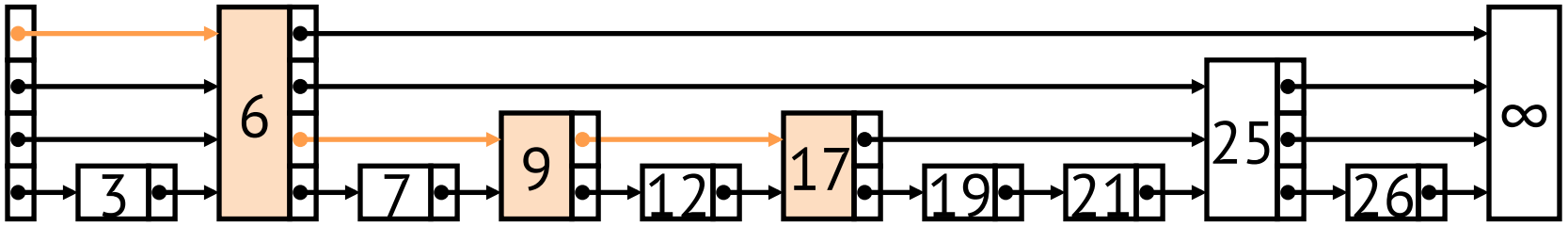
Search

Search(19)



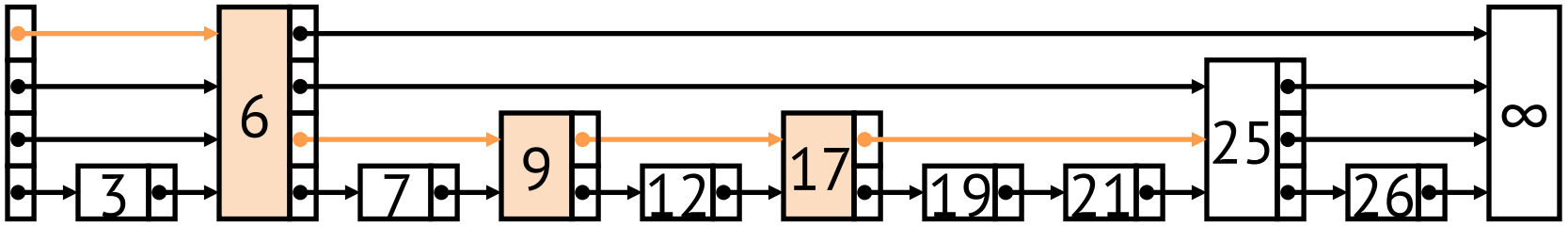
Search

Search(19)



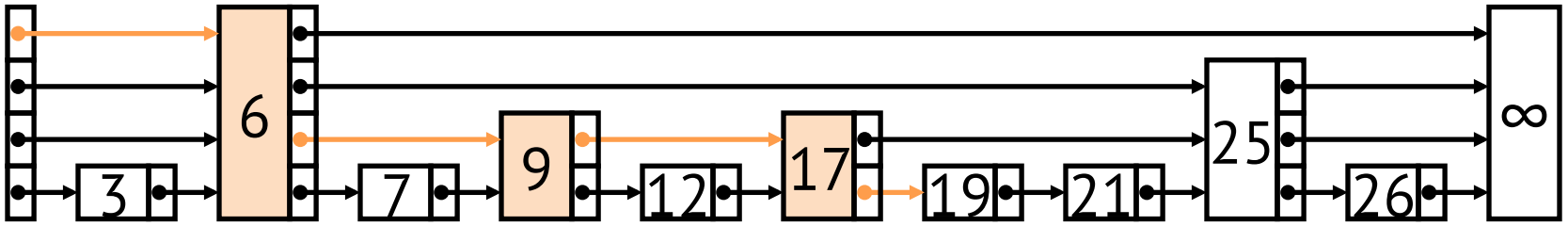
Search

Search(19)



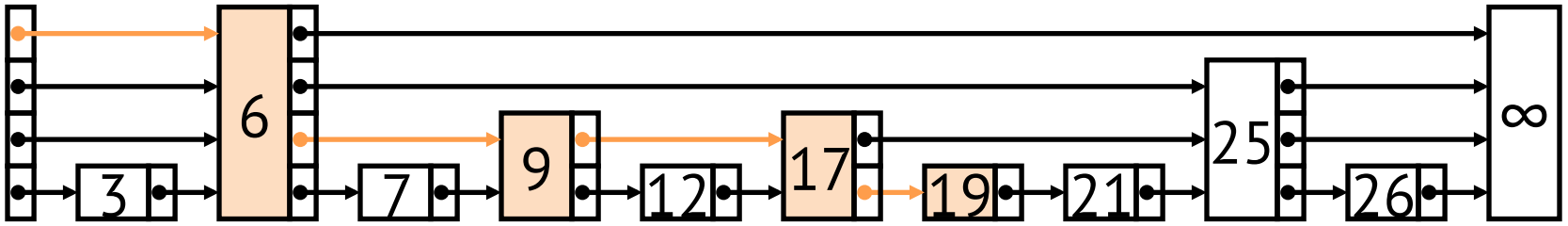
Search

Search(19)



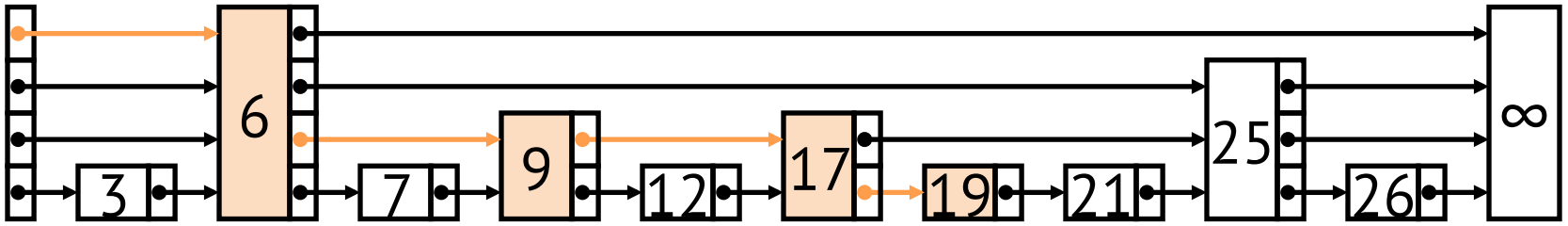
Search

Search(19)



Search

Search(19)



gefunden!

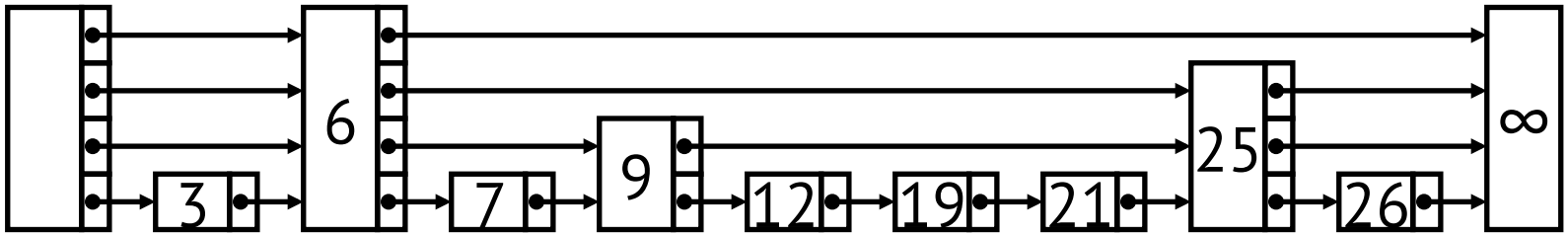
Insert

- Insert(list, search_key)

```
update[1..list.level] ← list.head
x ← list.head
for i ← list.level downto 1 do
    while x.next( i ).key < search_key do
        x ← x.next( i )
    update[ i ] ← x
x ← x.next( i )
if x.key ≠ search_key then
    new_level ← RandomLevel()
    if new_level > list.level then
        list.level ← new_level
    x = MakeNode( new_level, search_key )
    for i ← 1 to new_level do
        x.next( i ) ← update[ i ].next( i )
        update[ i ].next( i ) ← x
```

Insert

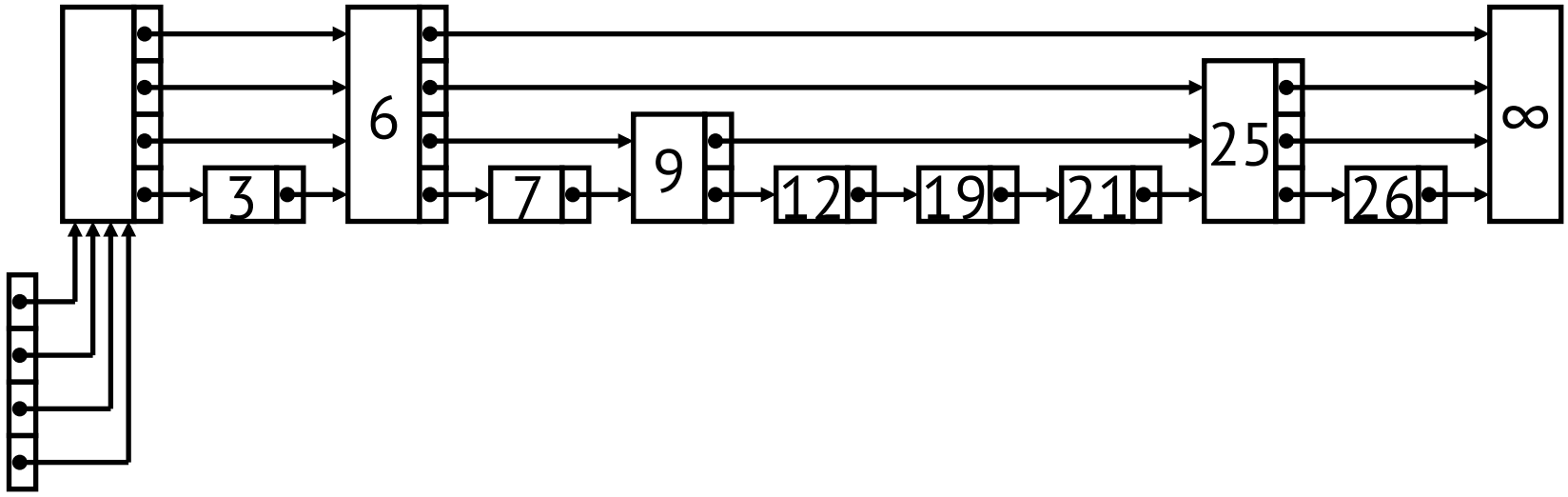
Insert(17)



update[]

Insert

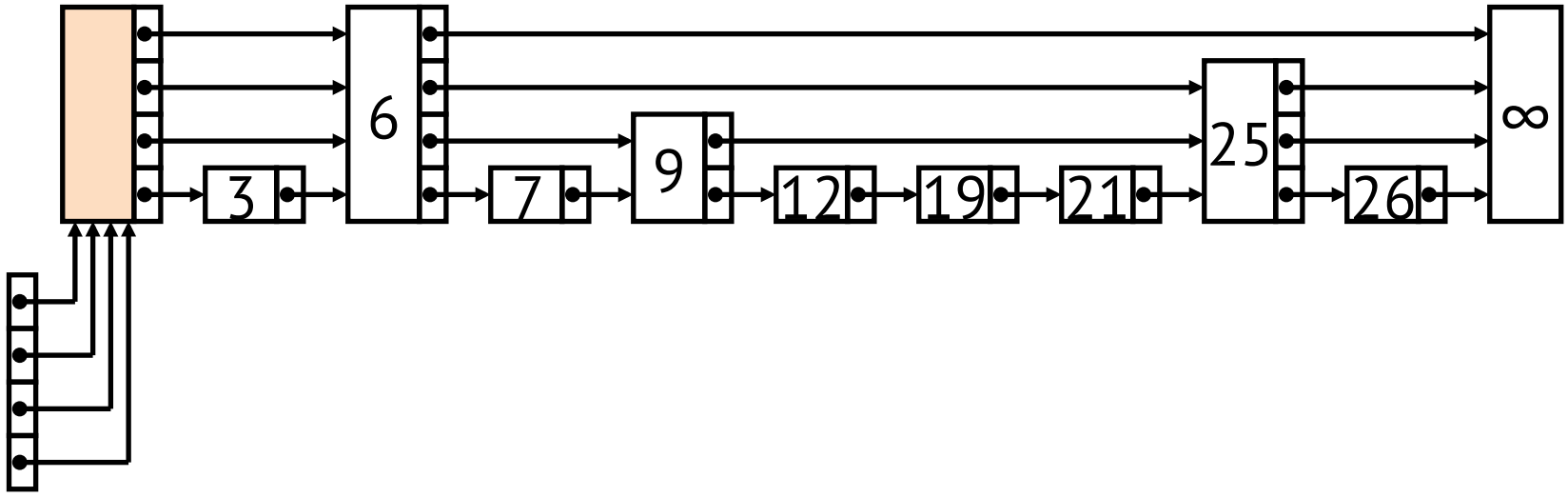
Insert(17)



update[]

Insert

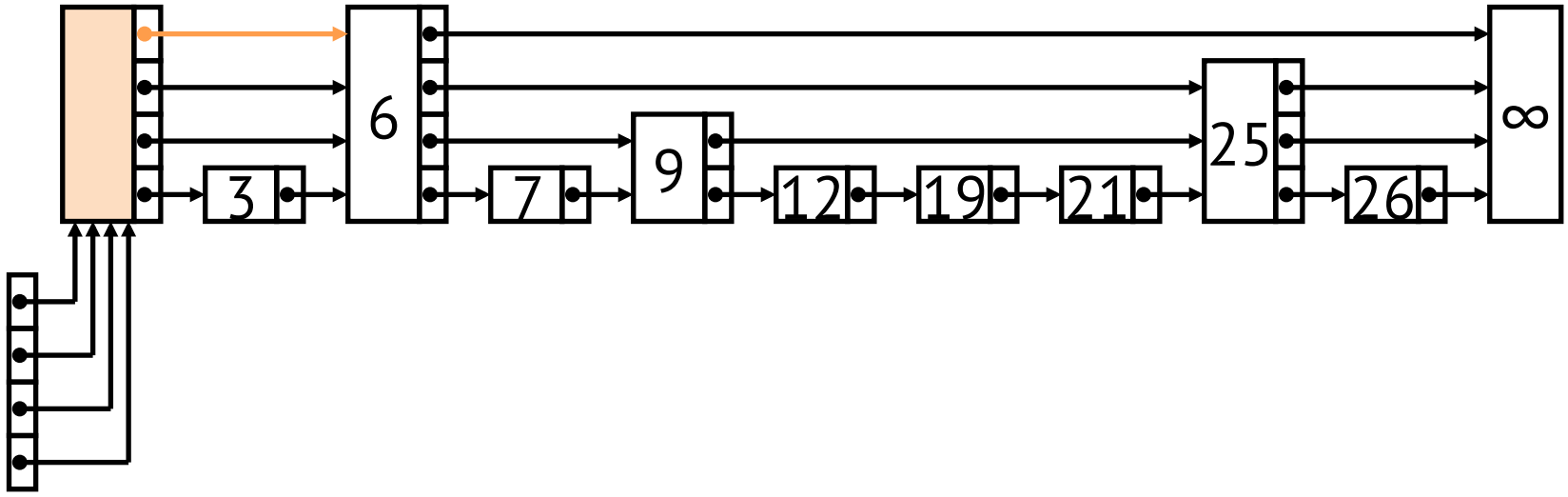
Insert(17)



update[]

Insert

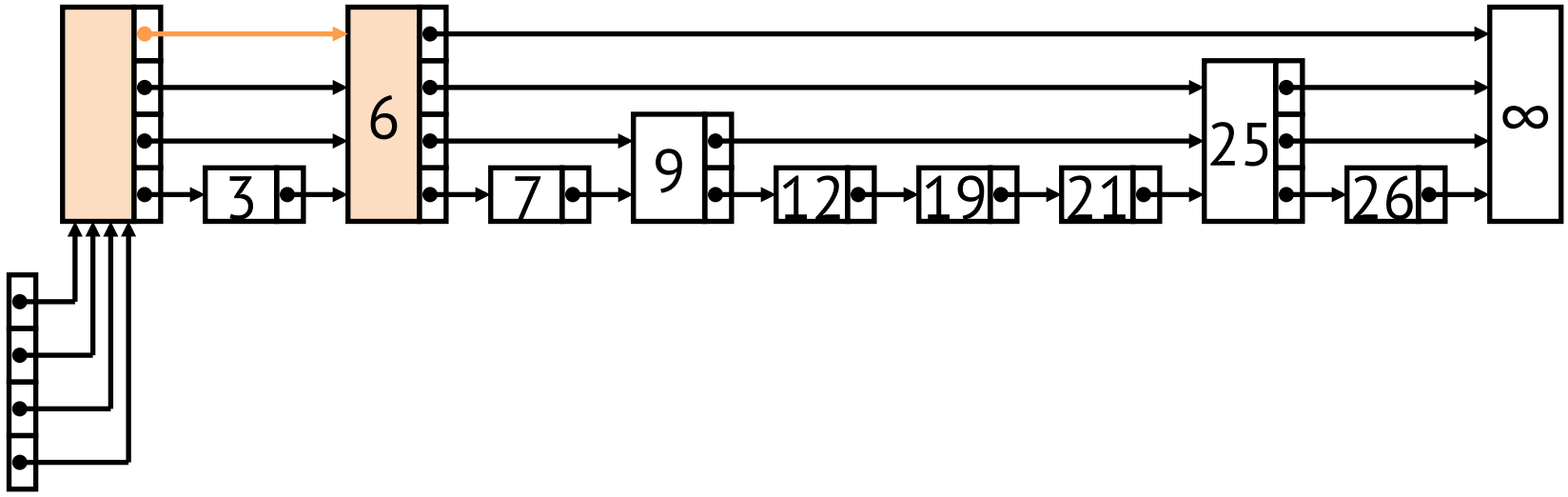
Insert(17)



update[]

Insert

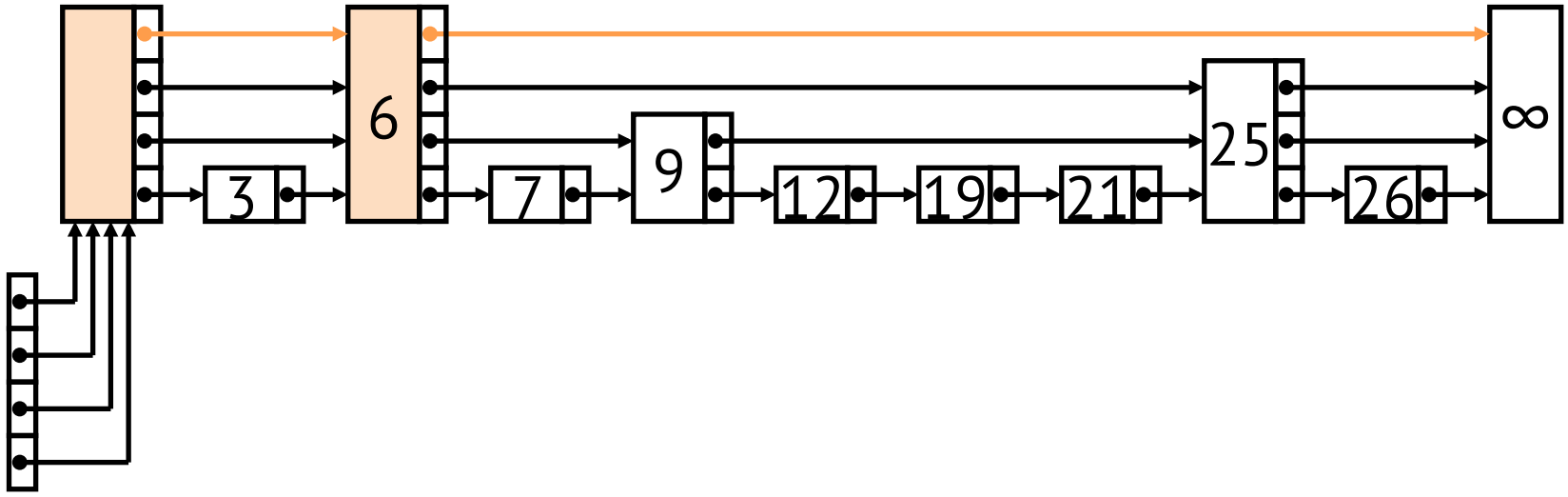
Insert(17)



update[]

Insert

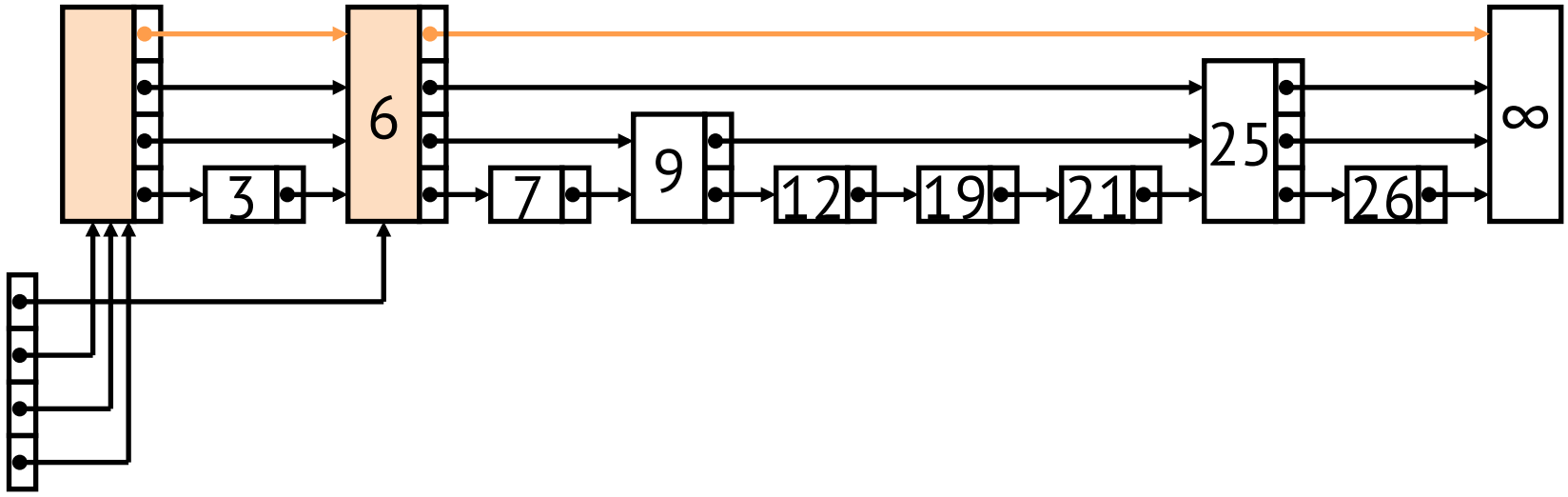
Insert(17)



update[]

Insert

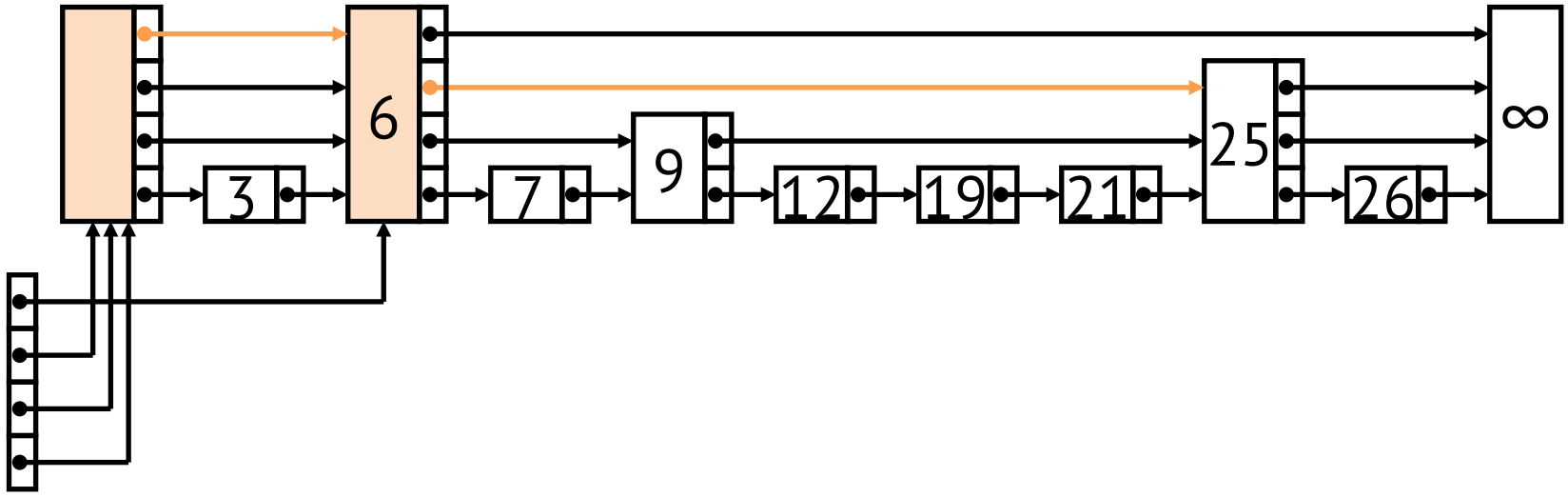
Insert(17)



update[]

Insert

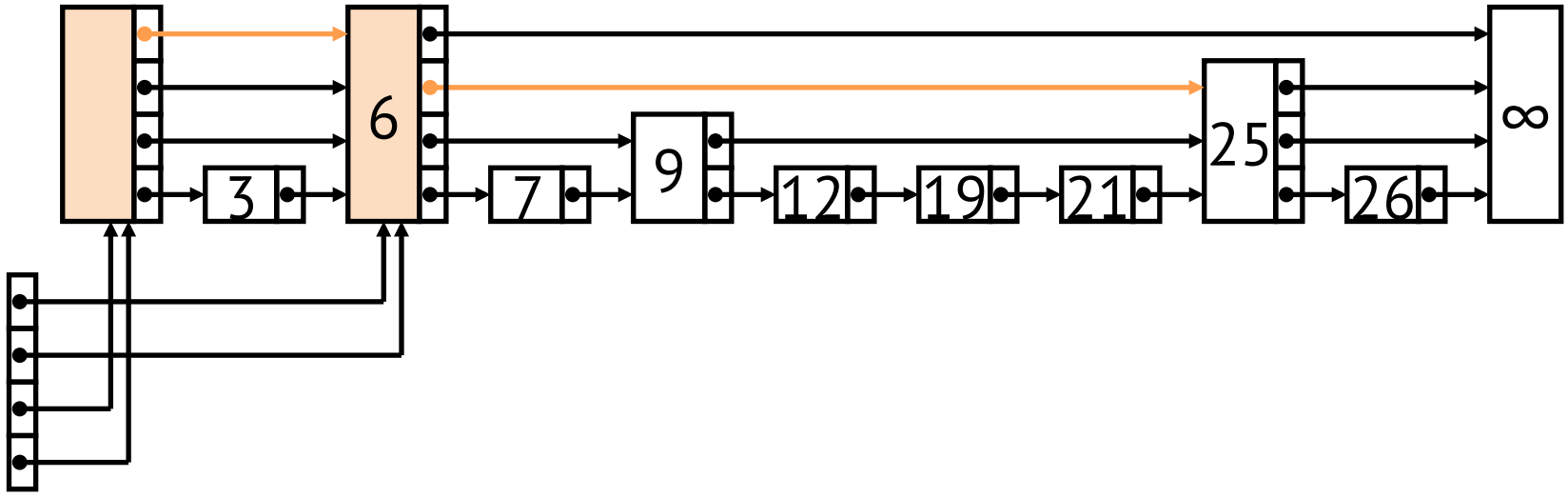
Insert(17)



update[]

Insert

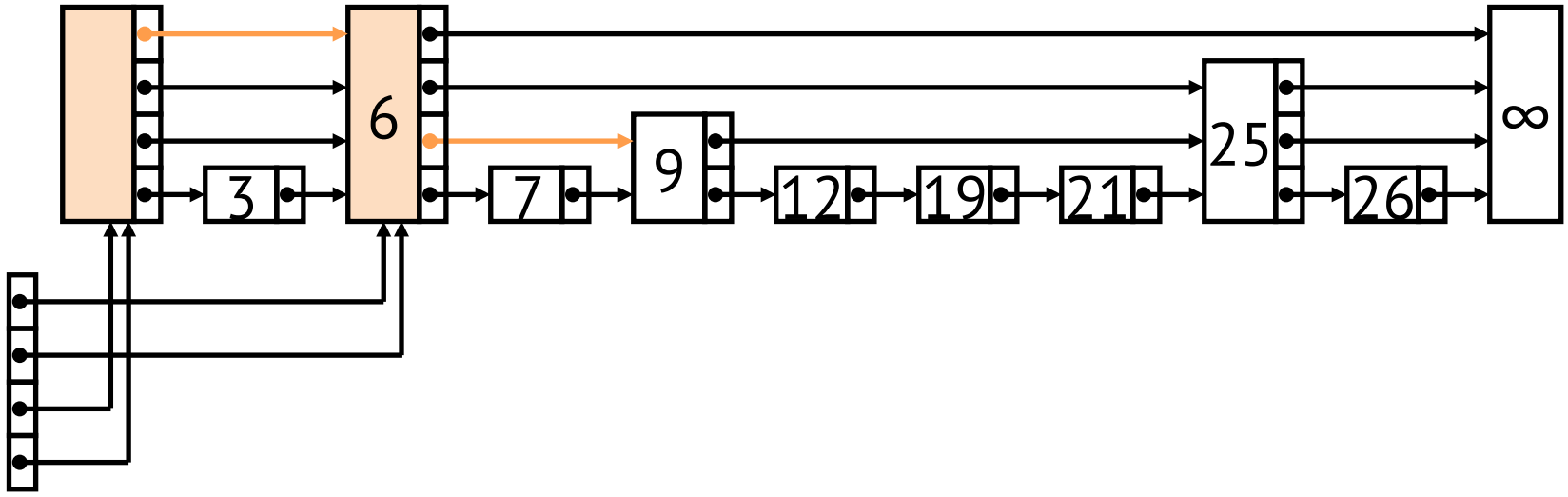
Insert(17)



update[]

Insert

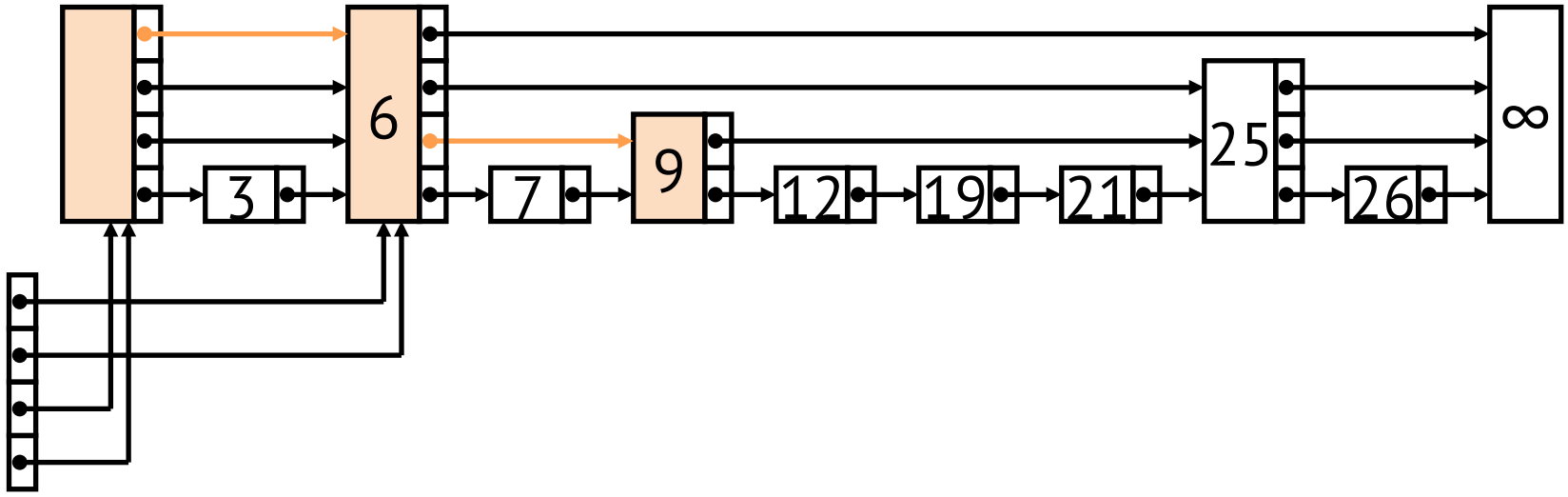
Insert(17)



update[]

Insert

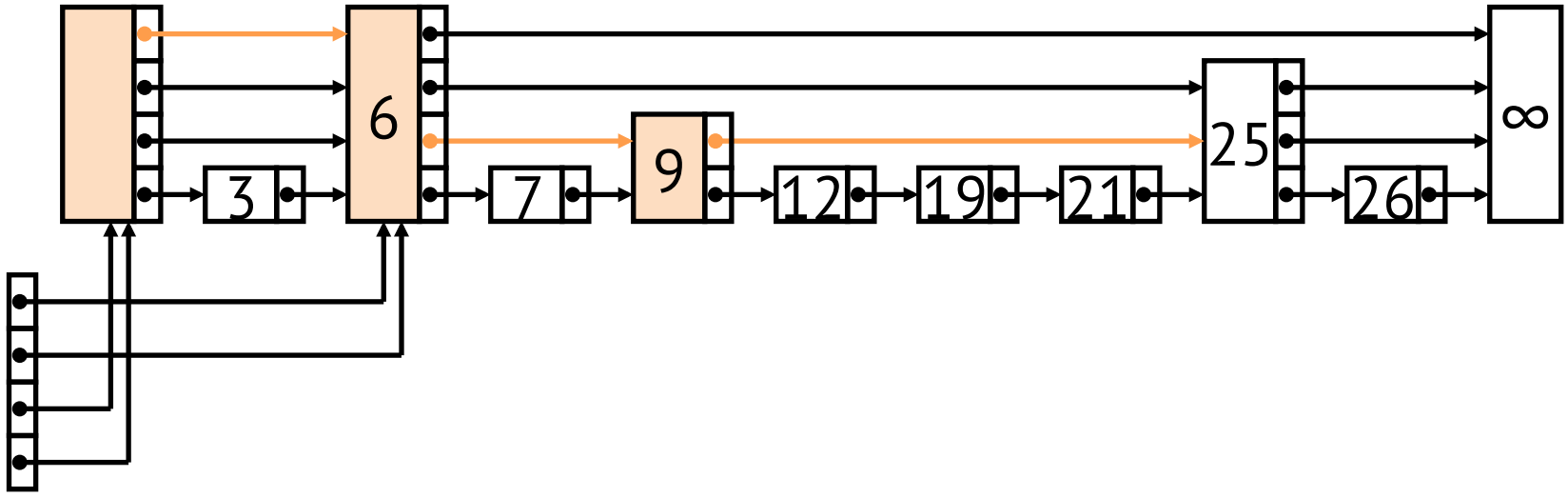
Insert(17)



update[]

Insert

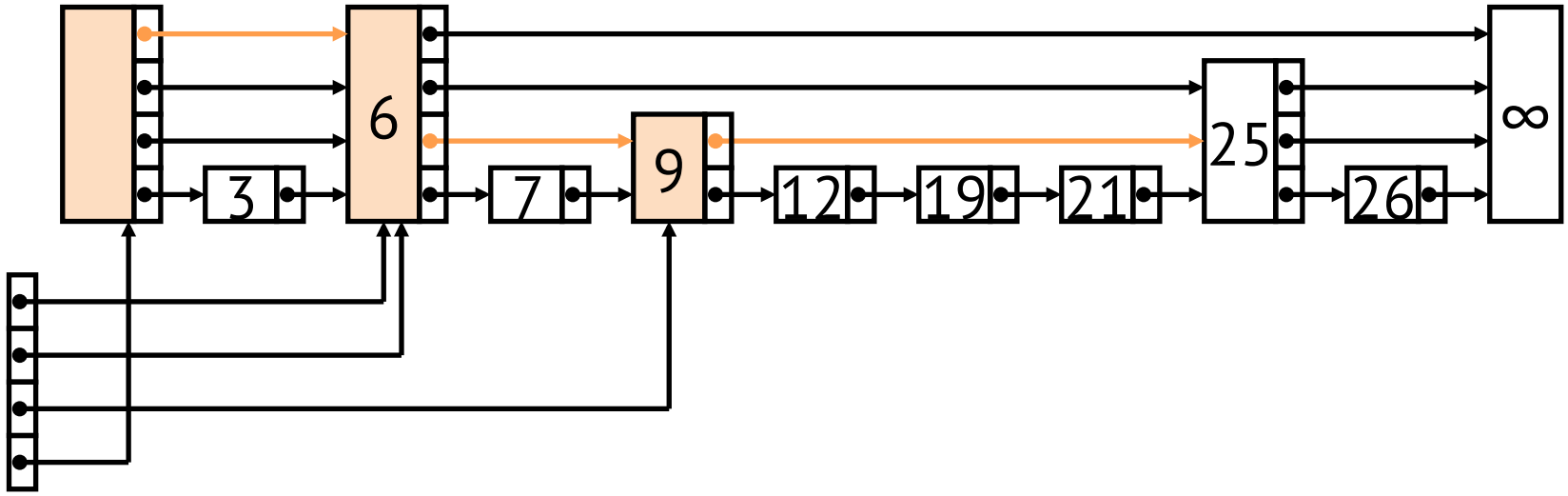
Insert(17)



update[]

Insert

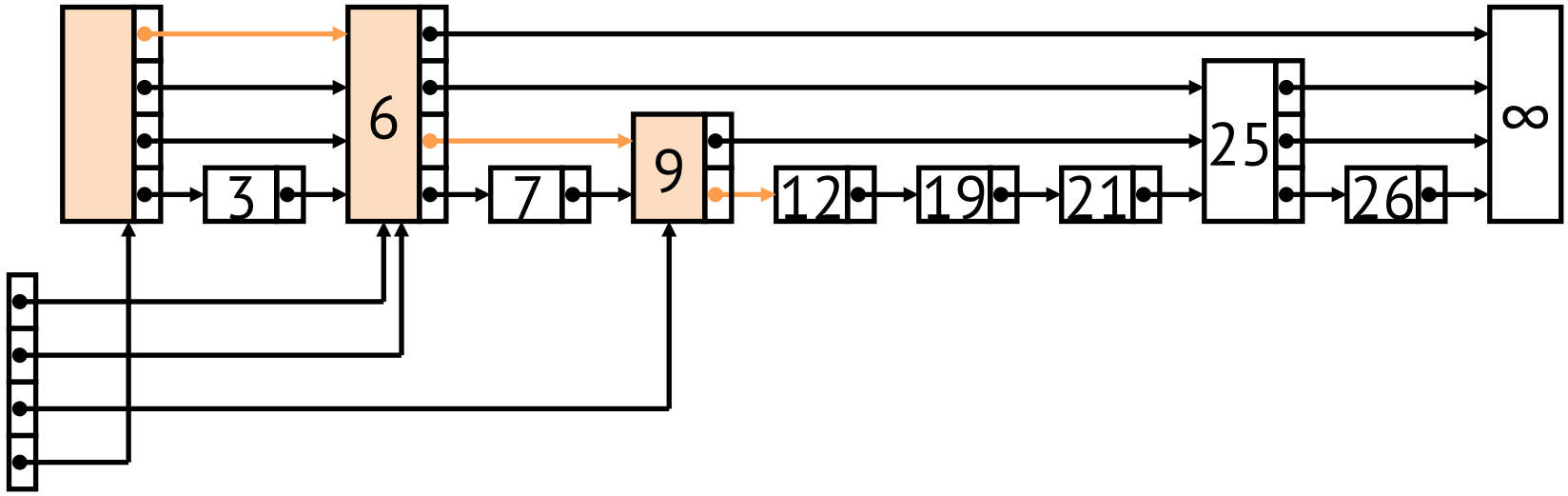
Insert(17)



update[]

Insert

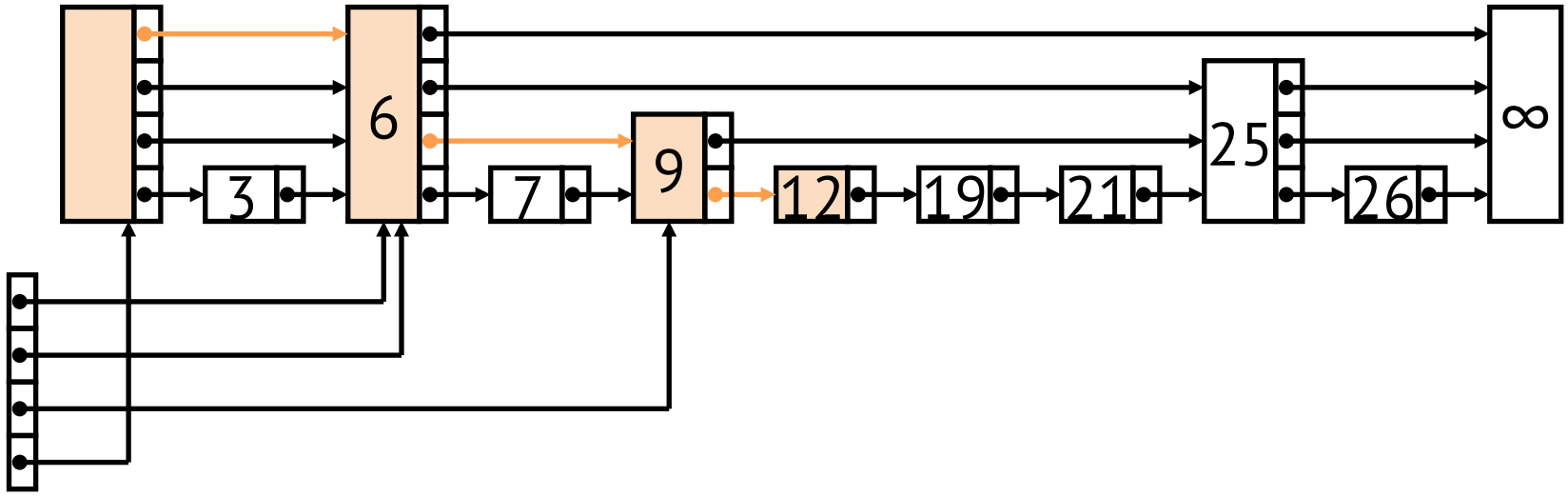
Insert(17)



update[]

Insert

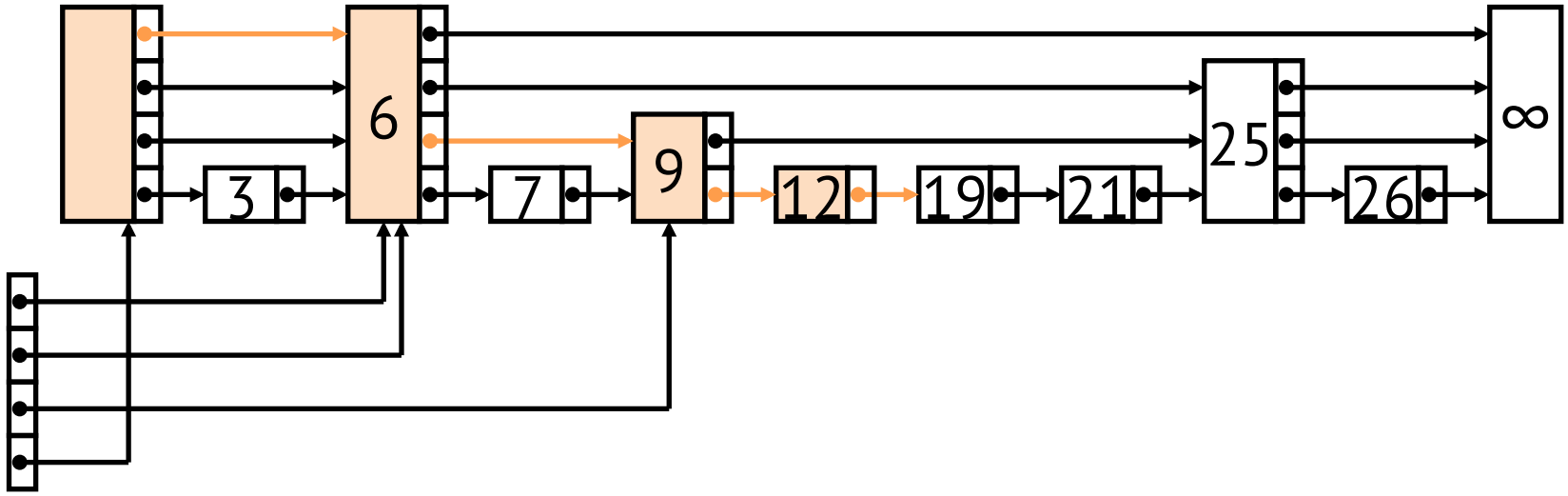
Insert(17)



update[]

Insert

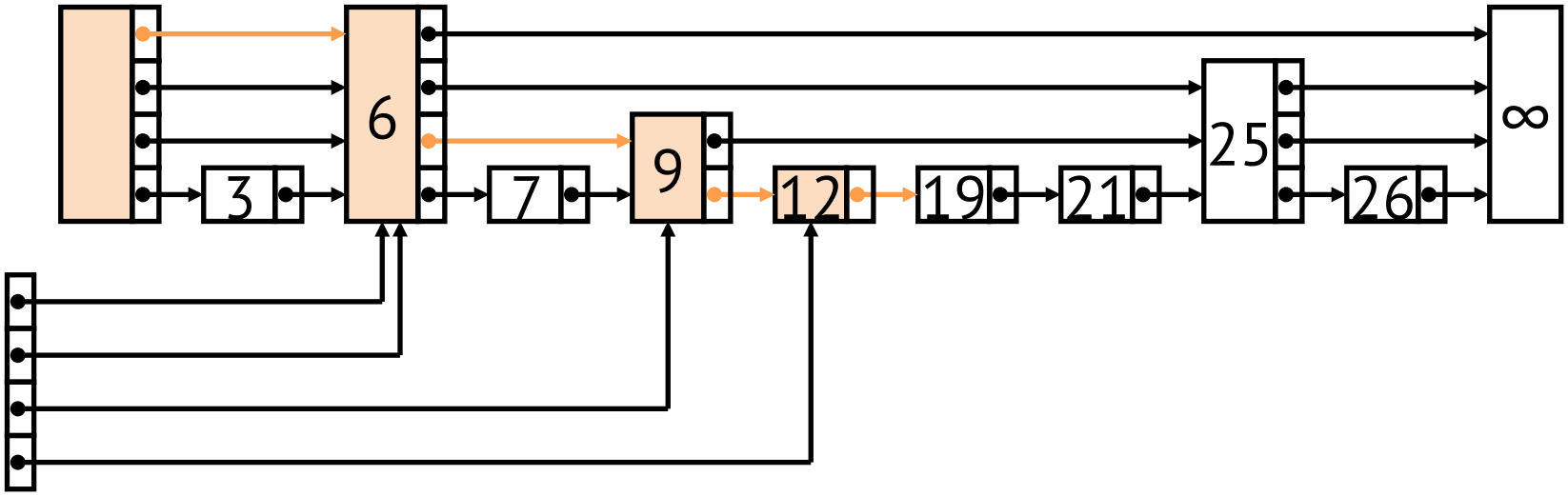
Insert(17)



update[]

Insert

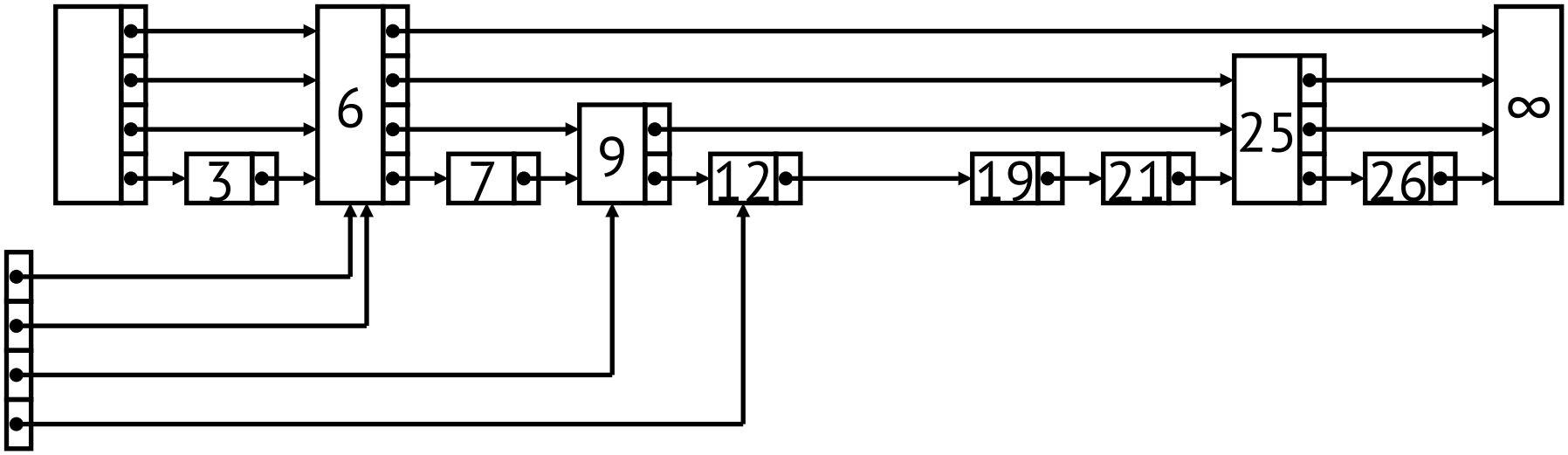
Insert(17)



update[]

Insert

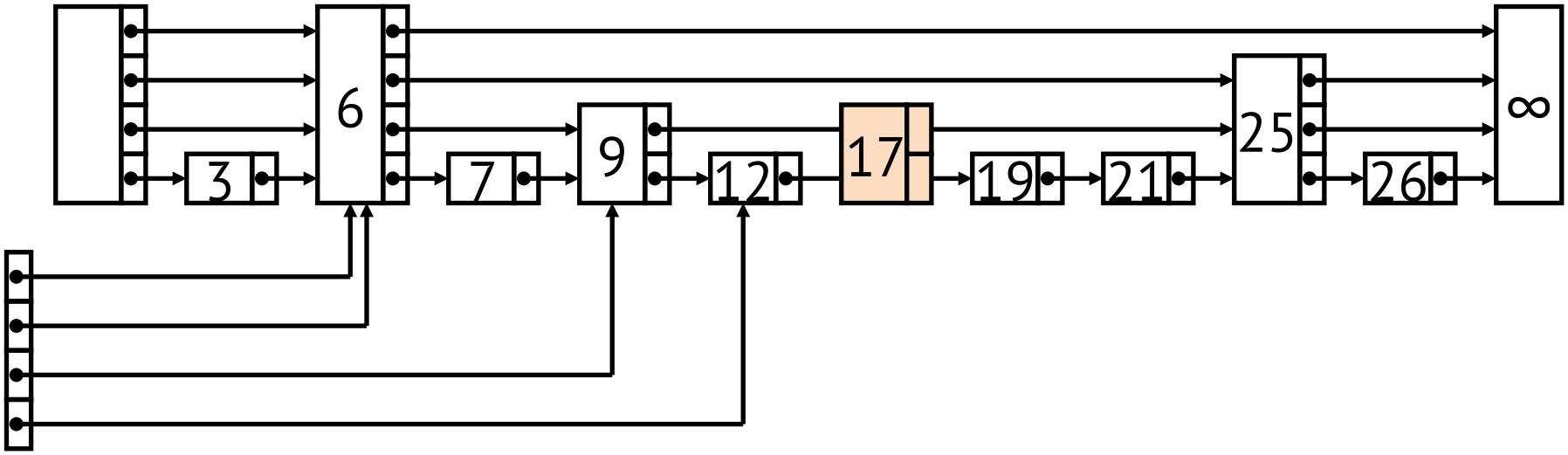
Insert(17)



update[]

Insert

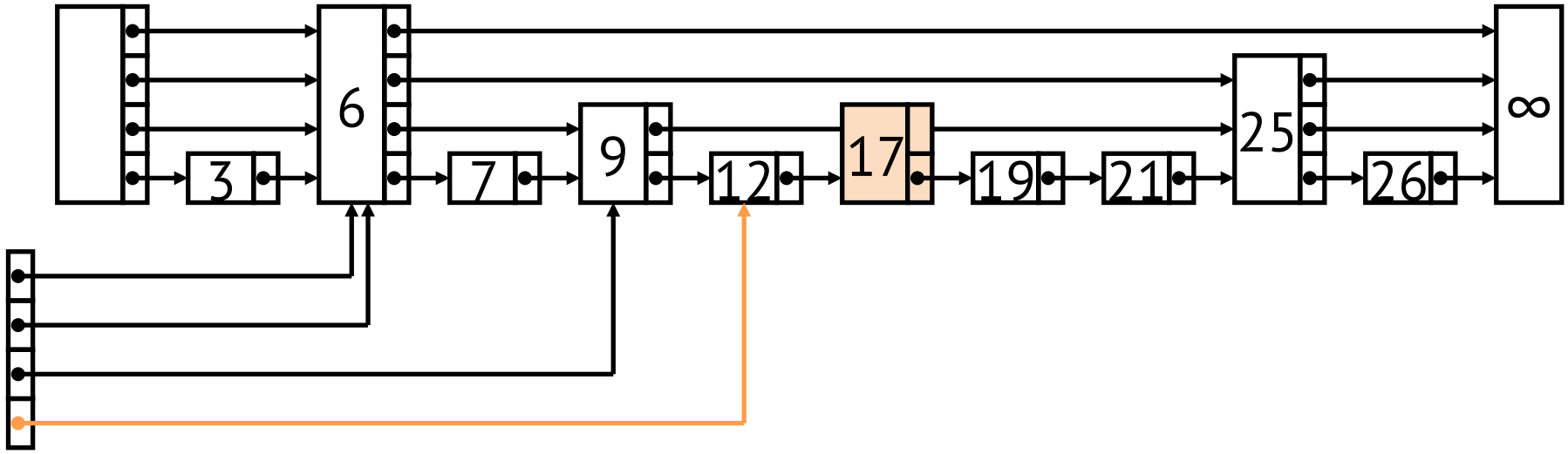
Insert(17)



update[]

Insert

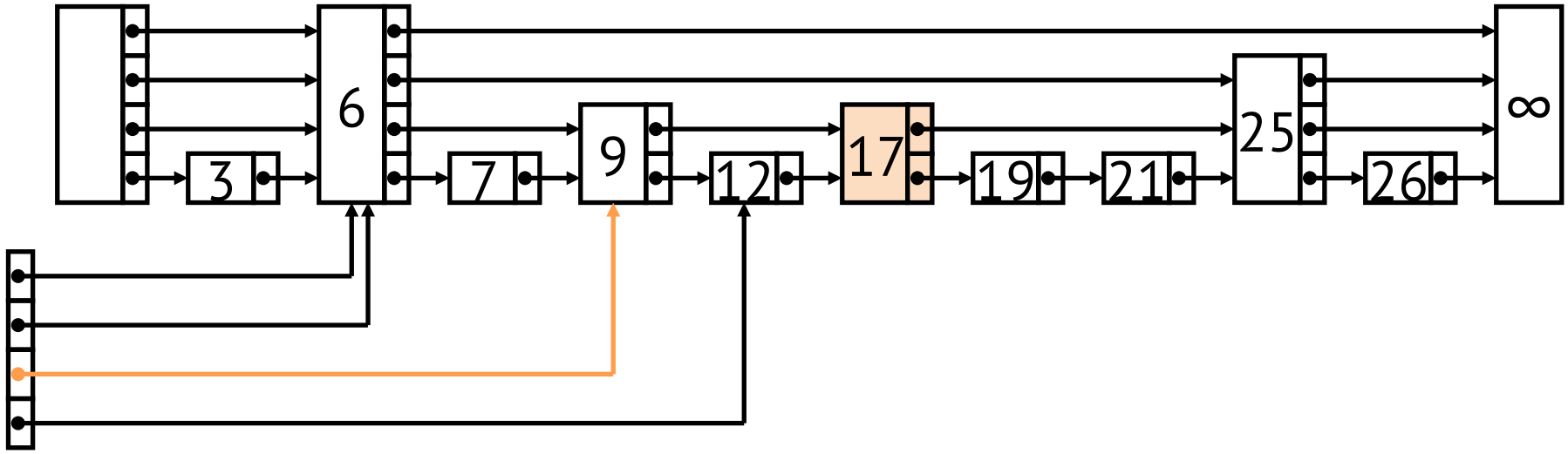
Insert(17)



update[]

Insert

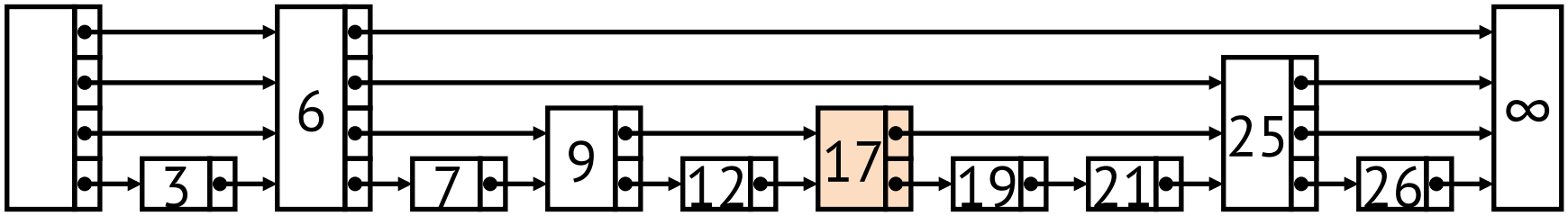
Insert(17)



update[]

Insert

Insert(17)



Delete

- Delete(list, search_key)

```
update[1..list.level] ← list.header
x ← list.header
for i = list.level downto 1 do
    while x.next( i ).key < search_key do
        x ← x.next( i )
    update[ i ] ← x
x ← x.next( i )
if x.key = search_key then
    for i ← 1 to x.level do
        update[ i ].next[ i ] ← x.next[ i ]
    free( x )
    while list.level > 1 and list.header.next( list.level ) = NIL do
        list.level ← list.level − 1
```

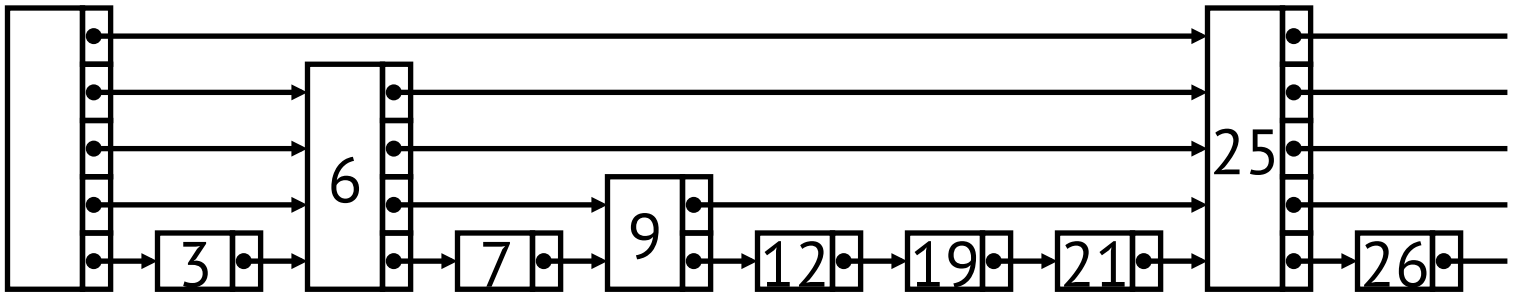
Analyse

- Was ist die erwartete Länge des Suchpfads?
- Was ist die erwartete Höhe der Knoten?
- Es sei $L(n) = \log_{1/p} n$, insbesondere $p^{L(n)}=1/n$
- Analysiere die Zahl der horizontalen und vertikalen Schritte getrennt!

Analyse

- Was ist die erwartete Länge des Suchpfads?
- Idee: Analysiere den Suchpfad rückwärts.

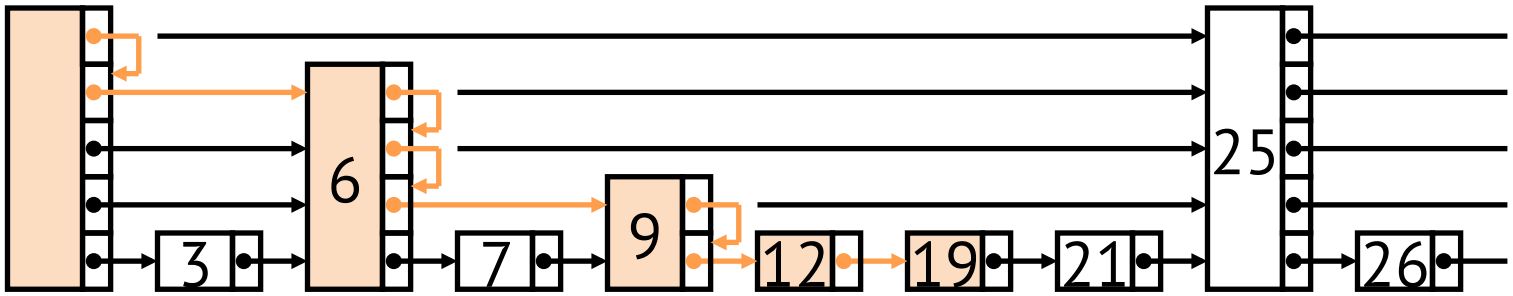
Search(19)



Analyse

- Was ist die erwartete Länge des Suchpfads?
- Idee: Analysiere den Suchpfad rückwärts.

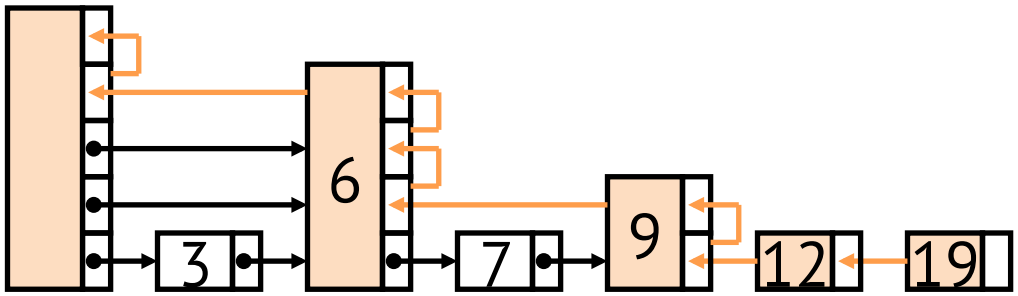
Search(19)



Analyse

- Was ist die erwartete Länge des Suchpfads?
- Idee: Analysiere den Suchpfad rückwärts.

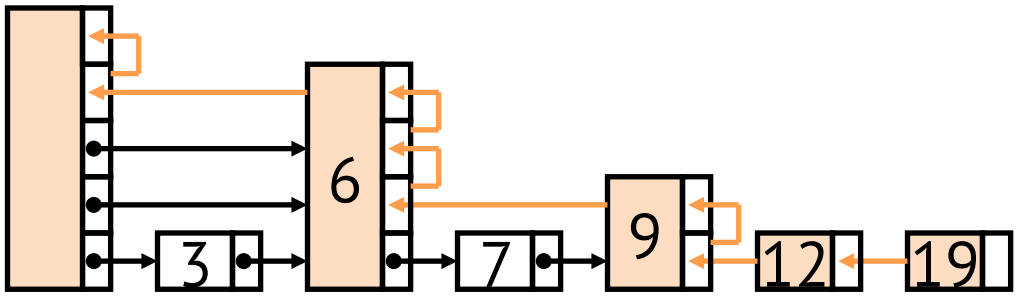
Search(19)



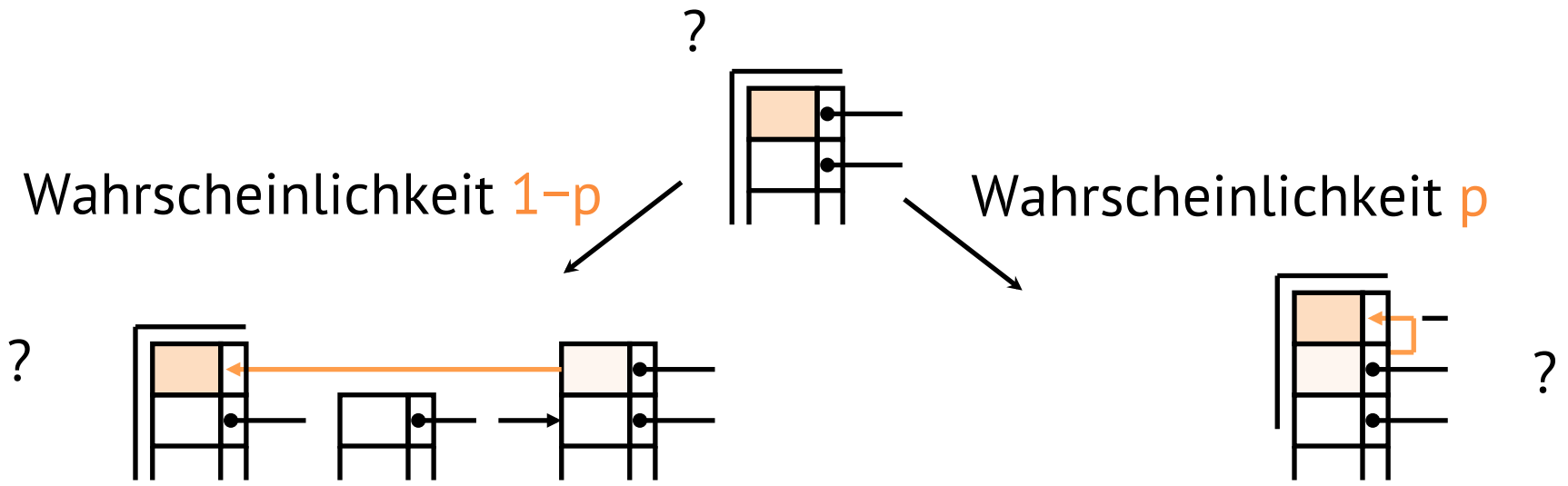
Analyse

- Es sei C(k) die erwartete Anzahl der Schritte, um k Level nach oben zu steigen.
- C(0) = 0

Search(19)



Analyse



$$C_k = (1-p)\times(1+C_k) + p\times(1+C_{k-1})$$

$$C_k = 1 + C_k - p - p\times C_k + p + p\times C_{k-1}$$

$$C_k = 1 \,/\, p + C_{k-1}$$

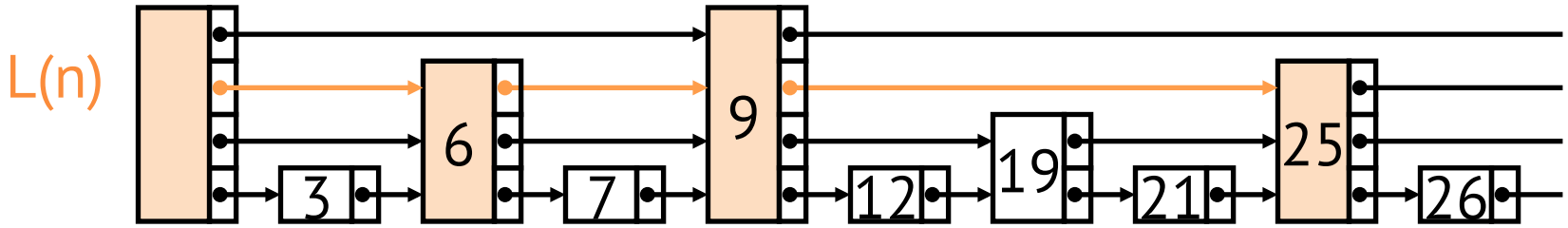
$$C_k = k \,/\, p$$

Analyse

1. die erwartete Anzahl der Schritte, um L(n)-1 Level nach oben zu steigen, ist $C_{L(n)-1} = (L(n)-1) / p$
2. …

Analyse

- Die erwartete Zahl der verbleibenden horizontalen Schritte bis zum Anfang der Liste ist durch die Zahl der Knoten mit Level $\geqslant L(n)$ beschränkt.
- Es gibt $np^{L(n)-1} = 1/p$ solcher Knoten



Analyse

- Es sei w_i die Wahrscheinlichkeit, dass ein Knoten q mit level(q) ⩾ i erzeugt wird

$$
\begin{aligned}
w_i &= \sum_{j=i}^{\infty} p^{j-1}(1-p) \\
&= \frac{1-p}{p} \sum_{j=i}^{\infty} p^j \\
&= \frac{1-p}{p} \left(\sum_{j=0}^{\infty} p^j - \sum_{j=0}^{i-1} p^j \right) \\
&= \frac{1-p}{p} \left(\frac{1}{1-p} - \frac{1-p^i}{1-p} \right) \\
&= p^{i-1}
\end{aligned}
$$

Analyse

- Die Wahrscheinlichkeit, dass ein Knoten q mit level(q) ⩾ L(n) erzeugt wird, ist also

$$w_{L(n)} = p^{L(n)-1} = \frac{1}{p} p^{log_{1/p} n} = \frac{1}{pn}$$

und die erwartete Zahl von Knoten mit Level ⩾ L(n) ist somit 1 / p

Analyse

1. die erwartete Anzahl der Schritte, um L(n)-1 Level nach oben zu steigen, ist $C_{L(n)-1} = (L(n)-1) / p$
2. Die erwartete verbleibende horizontale Länge bis zum Startknoten ist $\leqslant 1 / p$
3. …

Analyse

- Im Startknoten müssen wir von Level L(n) zum maximalen Level kommen
- Die Wahrscheinlichkeit, dass der Level eines Elements > k ist, ist p^k
- Die Wahrscheinlichkeit, dass kein Knoten mit Level > k existiert, ist $(1-p^k)^n$
- Die Wahrscheinlichkeit, dass der maximale Level der n Elemente > k ist, ist $1-(1-p^k)^n \leqslant np^k$
- Ist M der maximale Level, so ist $P(M > k) \leqslant np^k$

Analyse

$$
\begin{aligned}
E(M) &= \sum_{k=0}^{L(n)} kP(M=k) + \sum_{k=L(n)+1}^{\infty} kP(M=k) \\
&\leq L(n) + \sum_{k=L(n)+1}^{\infty} kP(M>k-1) \\
&\leq L(n) + \sum_{k=L(n)+1}^{\infty} knp^{k-1} \\
&= L(n) + \sum_{k=0}^{\infty} (k+L(n)+1)np^{k+L(n)}
\end{aligned}
$$

Analyse

$$
\begin{aligned}
E(M) &= \dots \\
&= L(n) + \sum_{k=0}^{\infty} (k + L(n) + 1) n p^{k+L(n)} \\
&= L(n) + \sum_{k=0}^{\infty} (k + L(n) + 1) p^{k} \\
&= L(n) + \sum_{k=0}^{\infty} k p^{k} + \sum_{k=0}^{\infty} (L(n) + 1) p^{k} \\
&= L(n) + \frac{p}{(1-p)^2} + (L(n) + 1) \frac{1}{1-p} \\
&= O(L(n))
\end{aligned}
$$

Analyse

1. die erwartete Anzahl der Schritte, um L(n)-1 Level nach oben zu steigen, ist $C_{L(n)-1} = (L(n)-1) / p$
2. Die erwartete verbleibende horizontale Länge bis zum Startknoten ist $\leqslant 1 / p$
3. Der erwartete maximale Level ist O(L(n))

Analyse

1. die erwartete Anzahl der Schritte, um L(n)-1 Level nach oben zu steigen, ist $C_{L(n)-1} = (L(n)-1) / p = O(L(n))$
2. Die erwartete verbleibende horizontale Länge bis zum Startknoten ist $\leqslant 1 / p = O(1)$
3. Der erwartete maximale Level ist O(L(n))

Analyse

- Die erwartete Länge des Suchpfades bei einer Skip-Liste mit n Elementen ist

$$O(L(n)) = O(\log_{1/p} n) = O(\log n)$$