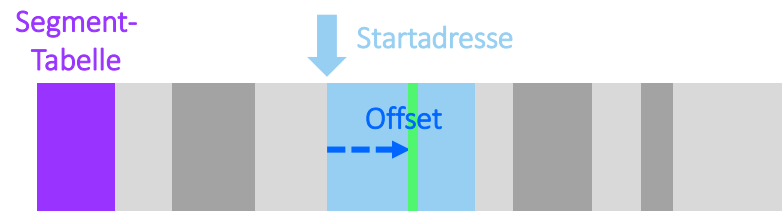


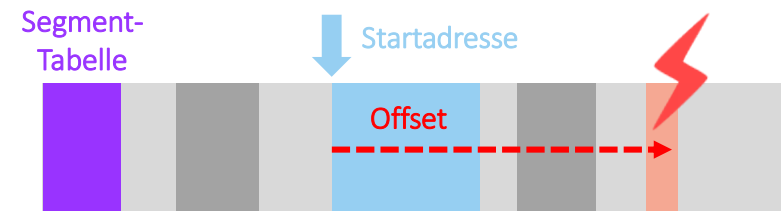
# Lösung der Übungsabgabe zur Speicherverwaltung

# Segmentbasierte Adressberechnung

- Die physikalische Adresse erhält man bei der **segmentbasierten Adressberechnung** durch die Addition von Startadresse mit dem in der virtuellen Adresse enthaltenen Offset (Byteposition in der Seite)
- Ist der Offset größer als die **Segmentlänge** entsteht eine **Speicherschutzverletzung** (engl. **Segmentation-Fault**):

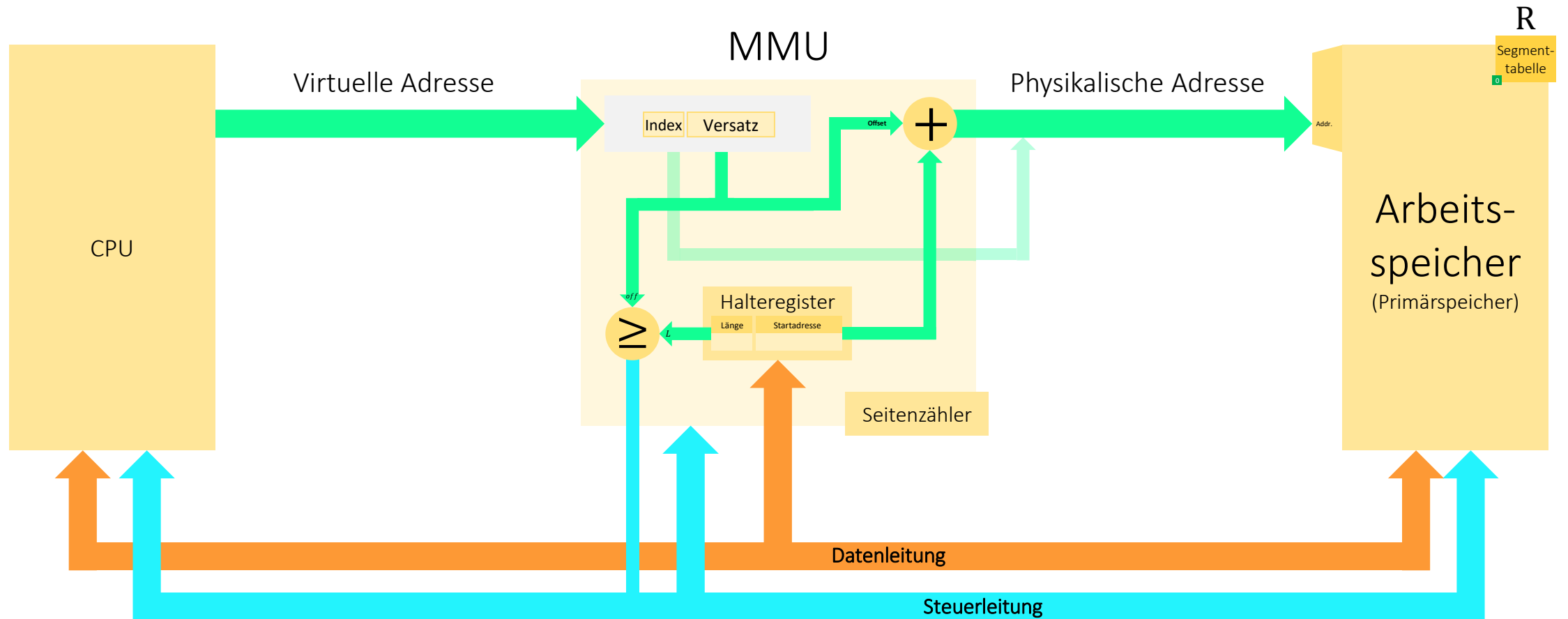


Speicherschutzverletzung:  
 $\text{Offset} < \text{Länge}$

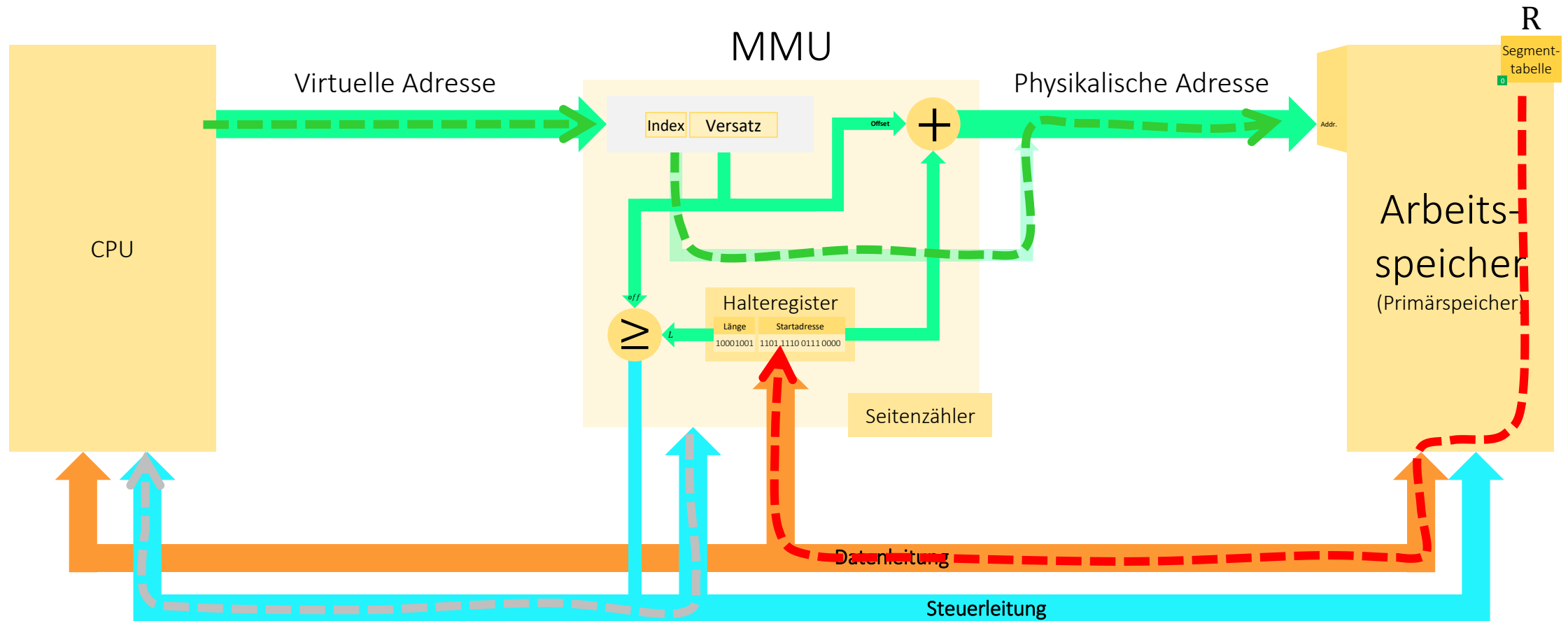


Speicherschutzverletzung:  
 $\text{Offset} \geq \text{Länge}$

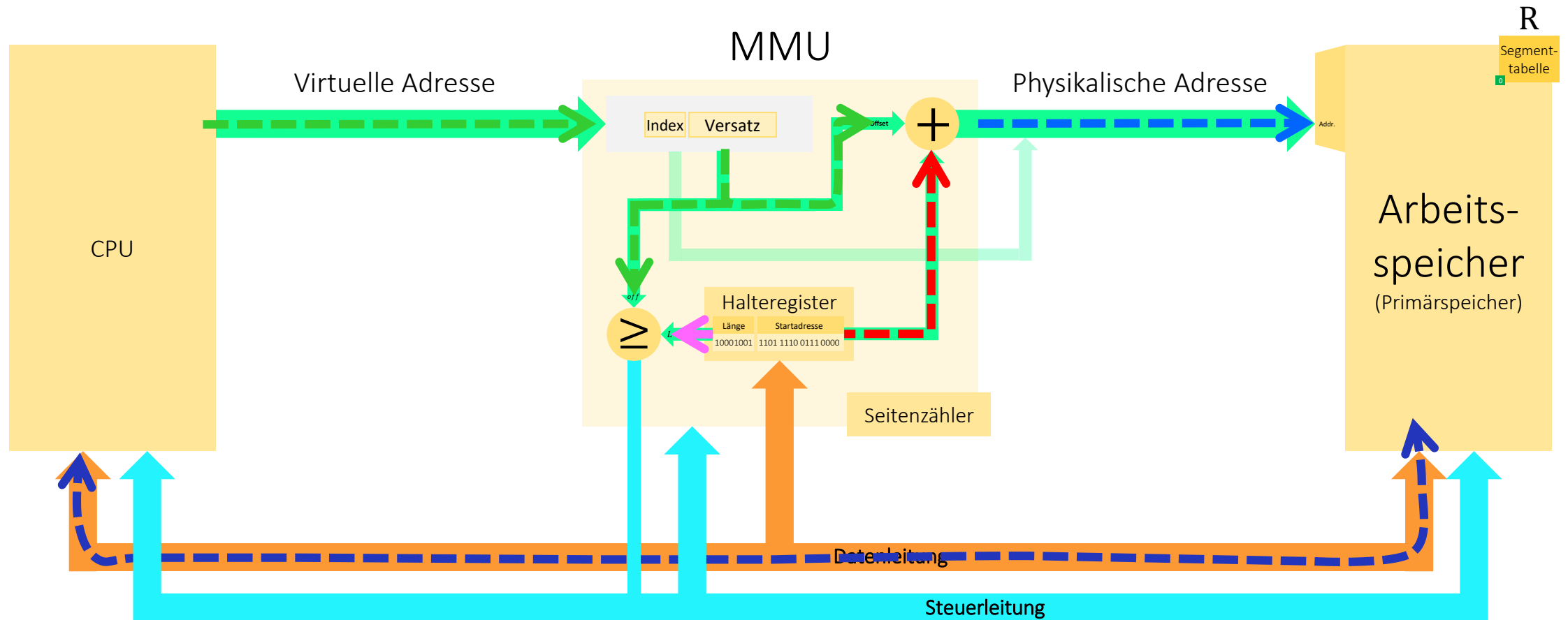
# Segmentierung



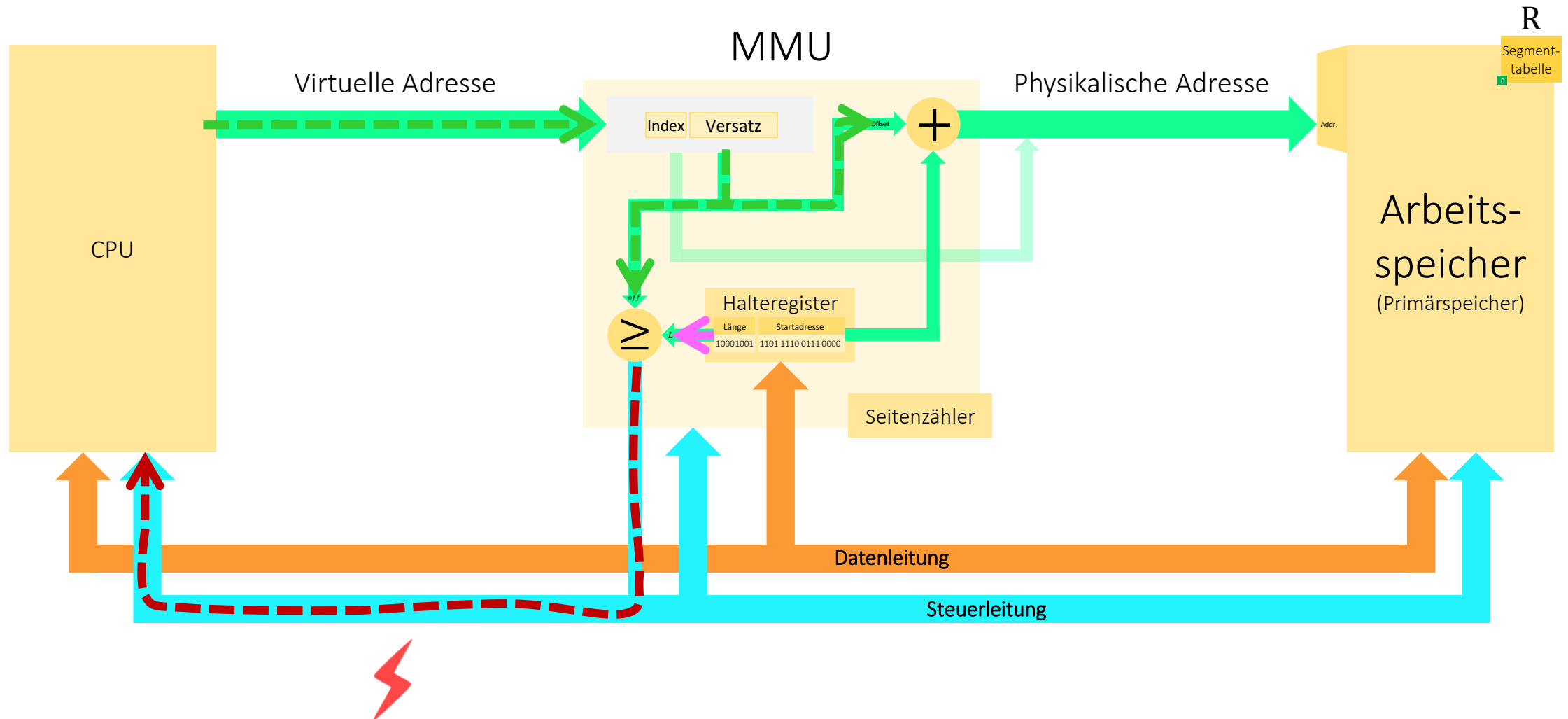
# Holen der Segmentnummer (Falls vorhanden)



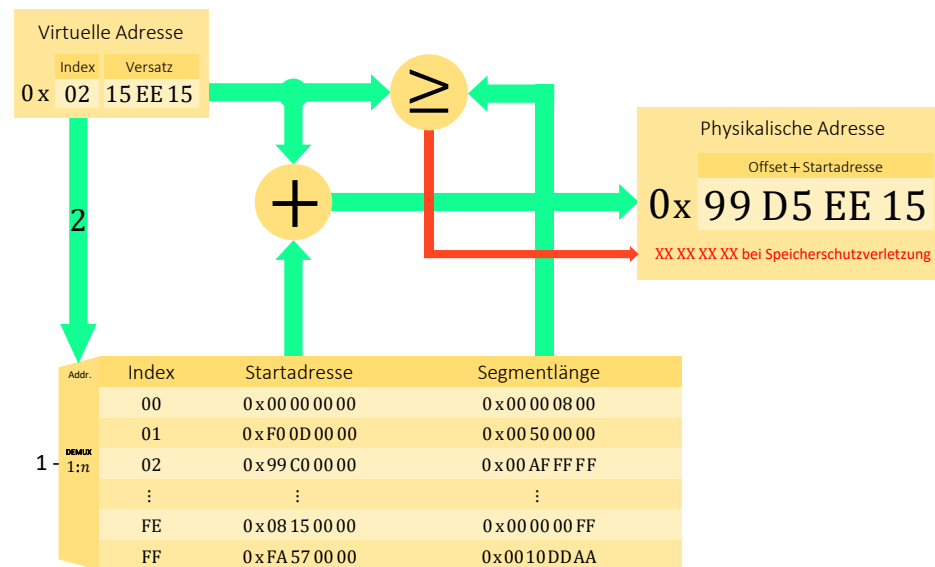
# Zusammensetzen der physikalischen Adresse



# Im Falle einer Speicherschutzverletzung



# Segmentbasierte Adressberechnung (Beispiel)



# LRU-Algorithmus

## Beispiel:

Registertabelle

Zugriff:	1	5	3	3	4	1	7	1	4	3	3	5	4	1
Kachel 0:	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Kachel 1:	-	5	5	5	5	5	7	7	7	7	7	5	5	5
Kachel 2:	-	-	3	3	3	3	3	3	3	3	3	3	3	3
Kachel 3:	-	-	-	-	4	4	4	4	4	4	4	4	4	4

Zugriffstabelle (Zahl der Zyklen seit des letzten Aufrufs)

Kachel 0:	0	1	2	3	4	0	1	0	1	2	3	4	5	0
Kachel 1:	INT_MAX	0	1	2	3	4	0	1	2	3	4	0	1	2
Kachel 2:	INT_MAX	INT_MAX	0	0	1	2	3	4	5	0	0	1	2	3
Kachel 3:	INT_MAX	INT_MAX	INT_MAX	INT_MAX	0	1	2	3	0	1	2	3	0	1



# Adjazenztafel der 'Älter als'-Relation $\preccurlyeq$

- $\preccurlyeq$  kann als Adjazenzmatrix  $\text{Adj}_{\preccurlyeq}$  dargestellt werden
- $j \preccurlyeq i \equiv \text{Adj}_{\preccurlyeq}[i, j] = 1$ , ansonsten gilt  $\text{Adj}_{\preccurlyeq}[i, j] = 0$
- Offensichtlich muss  $\preccurlyeq$  reflexiv sein: Für die Diagonale  $d_0 \cdots d_{n-1} = 1$
- Außerdem ist  $\preccurlyeq$  antisymmetrisch:  $\text{Adj}_{\preccurlyeq}[i, j] = 0 \Leftrightarrow \text{Adj}_{\preccurlyeq}[j, i] = 1$
- Statt  $\forall j \in \{k \in K \mid j > k\}$  wird für ein festes  $k$  hier  $\forall j > k$  geschrieben

# Adjazenztafel der 'Älter als'-Relation $\preceq$

Verwendung im LRU-Algorithmus (Ähnliches Vorgehen bei LFU):

- $\preceq$  kann sich nach jedem Kachelaufruf verändern
- $\preceq_t$  ist die 'Älter als'-Relation nach dem Zeitschritt  $t$



# LRU-Algorithmus mit Adjazenzmatrix

- Wurde unmittelbar vor dem  $t$ -ten Zeitschritt die Kachel  $k \in K$  aufgerufen, so muss  $k$  unter  $\preccurlyeq_t$  als  $\preccurlyeq$  bzw.  $<$  die folgenden Eigenschaften erfüllen:
  - $\forall j > k. \text{Adj}_{\preccurlyeq}[k, j] = 0$  bzw.  $\forall j > k. k < j$ : Zeile auf Null setzen
  - $\forall i < k. \text{Adj}_{\preccurlyeq}[i, k] = 1$  bzw.  $\forall i < k. k \preccurlyeq i$ : Spalte auf Eins setzen
- Der Älteste Eintrag zum Zeitpunkt  $t$  wird  $x := \max_{\preccurlyeq_t} K$  genannt:
  - $\forall i < x. \text{Adj}_{\preccurlyeq}[i, x] = 0$  bzw.  $\forall i < x. i < x$ : Spalte ist Null
  - $\forall j > x. \text{Adj}_{\preccurlyeq}[x, j] = 1$  bzw.  $\forall j > x. j \preccurlyeq x$ : Zeile ist Eins

}  $\forall j \in K. j \preccurlyeq x$

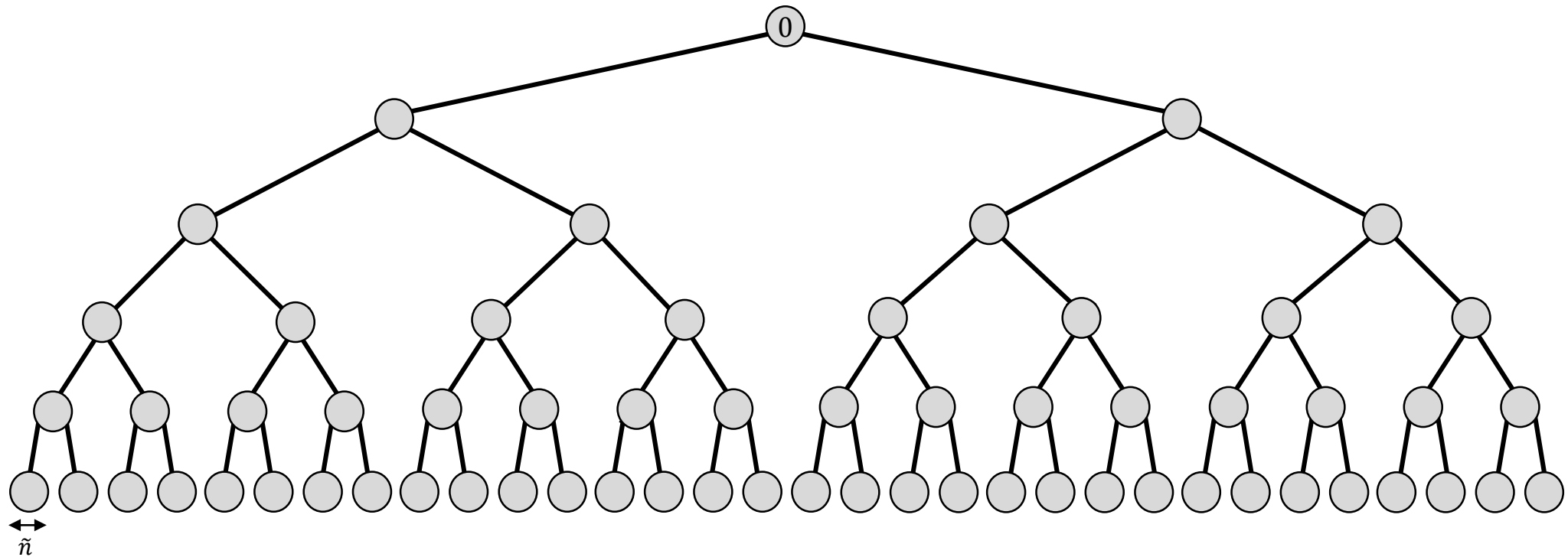
# Aufgabe zu Kachelaufrufen

$i \setminus j$	0	1	2	3	4	5	6	7
0	$d_0$	0	0	0	0	0	0	0
1		$d_1$	0	0	1	0	0	1
2			$d_2$	1	1	1	1	1
3				$d_3$	1	1	1	1
4					$d_4$	0	0	0
5						$d_5$	1	1
6							$d_6$	1
7								$d_7$

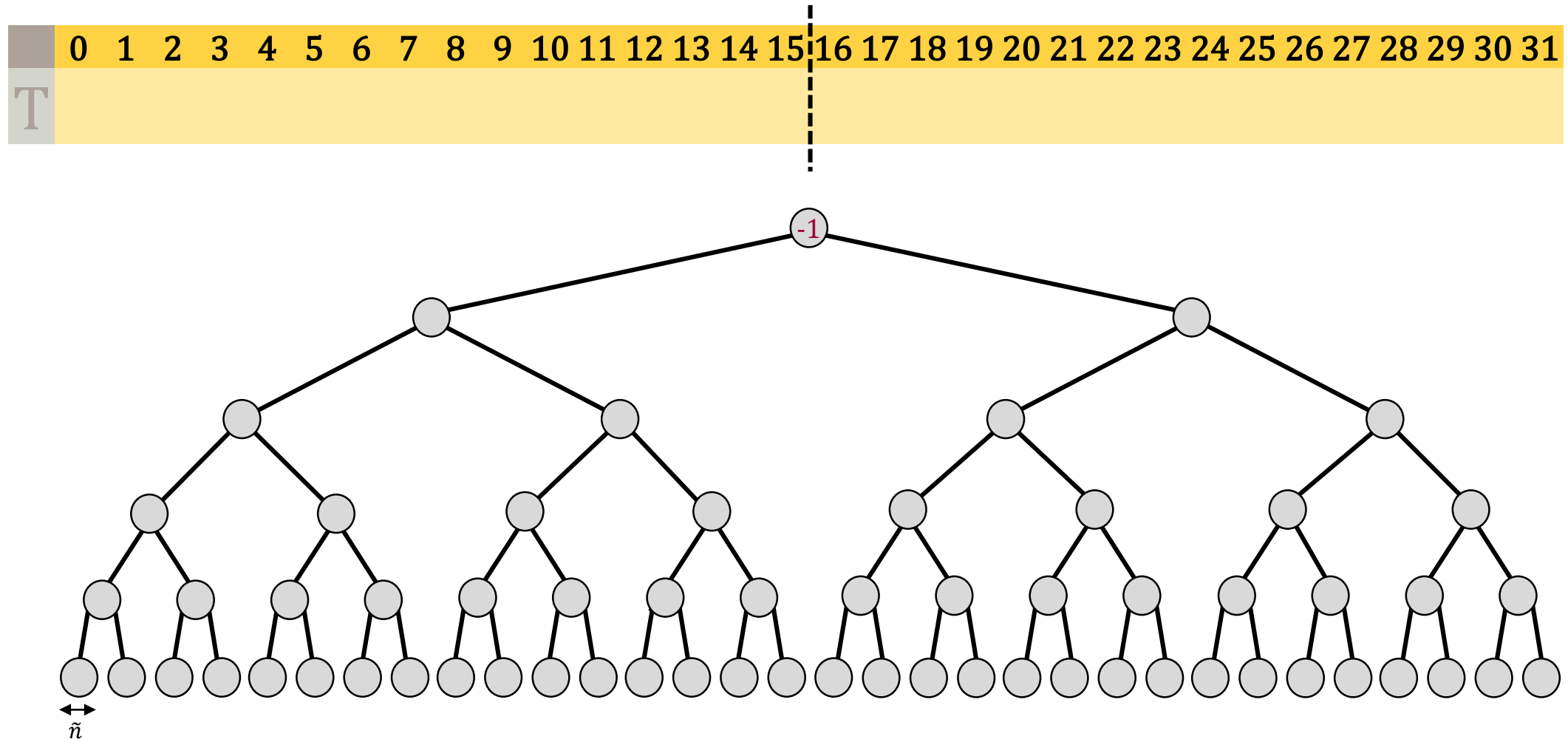
$$k = 4$$
[illegible]
$$k = 5$$
[illegible]

# Buddy-Verfahren (Ausgangszustand)

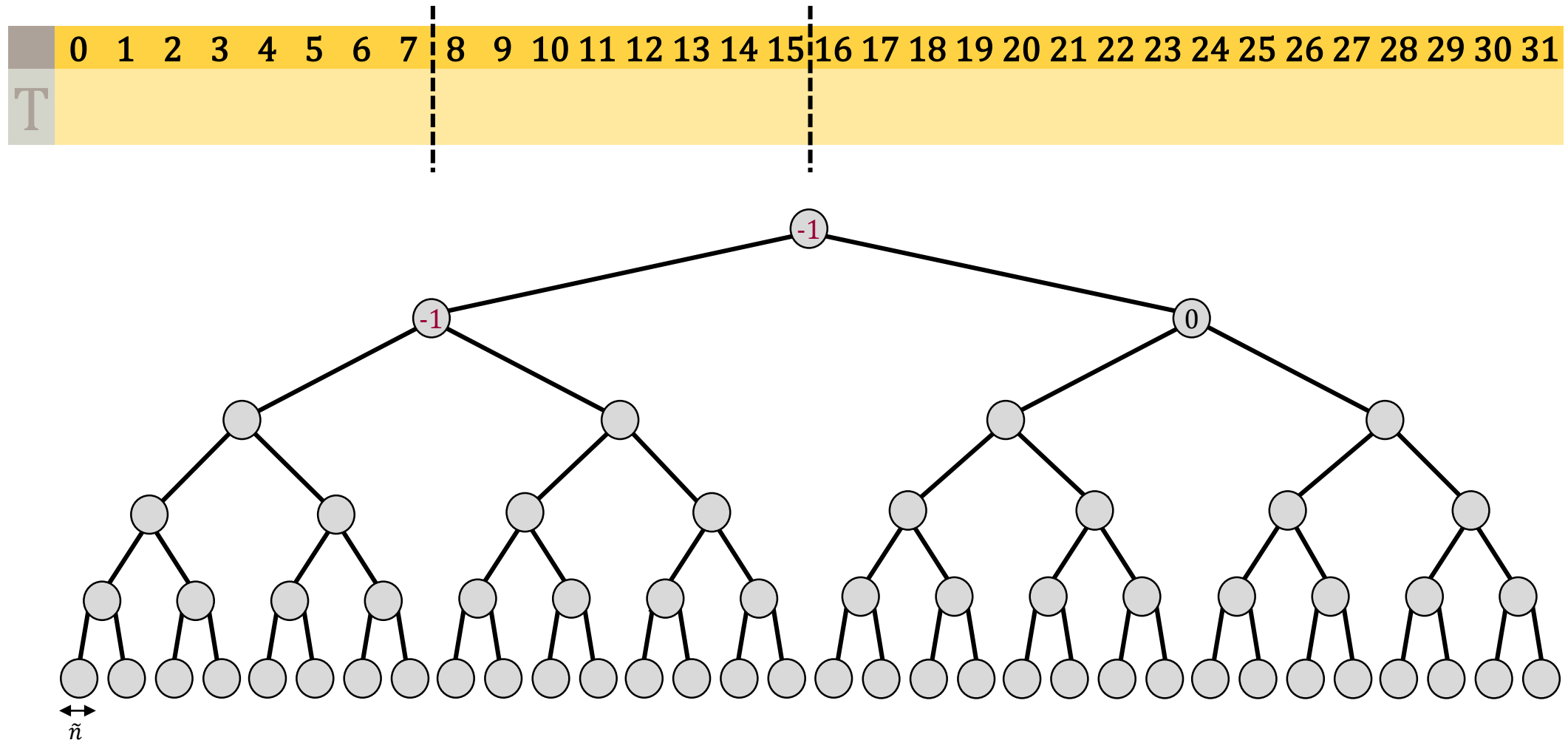
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
T																																



# Buddy-Verfahren (A Reservieren I)

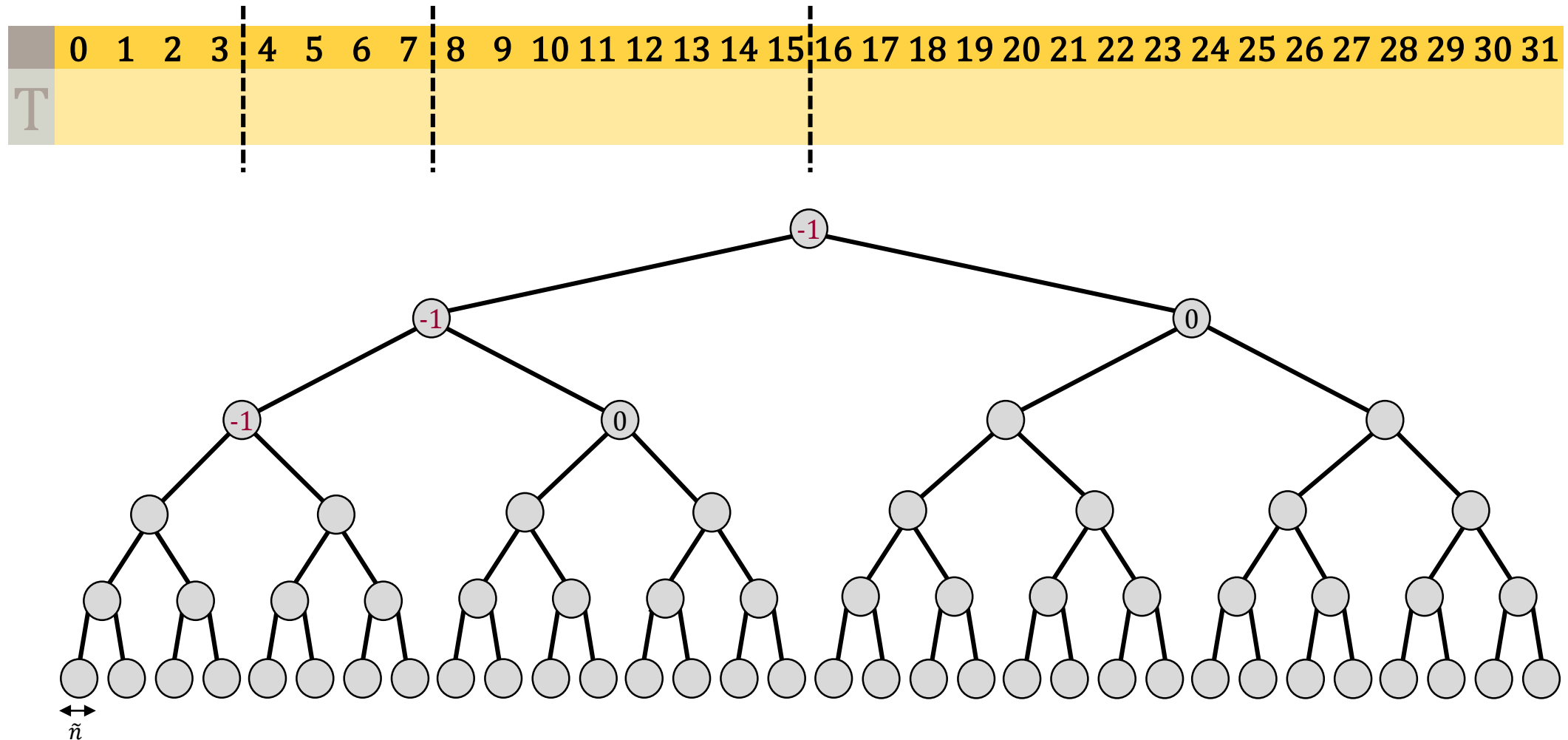


# Buddy-Verfahren (A Reservieren II)

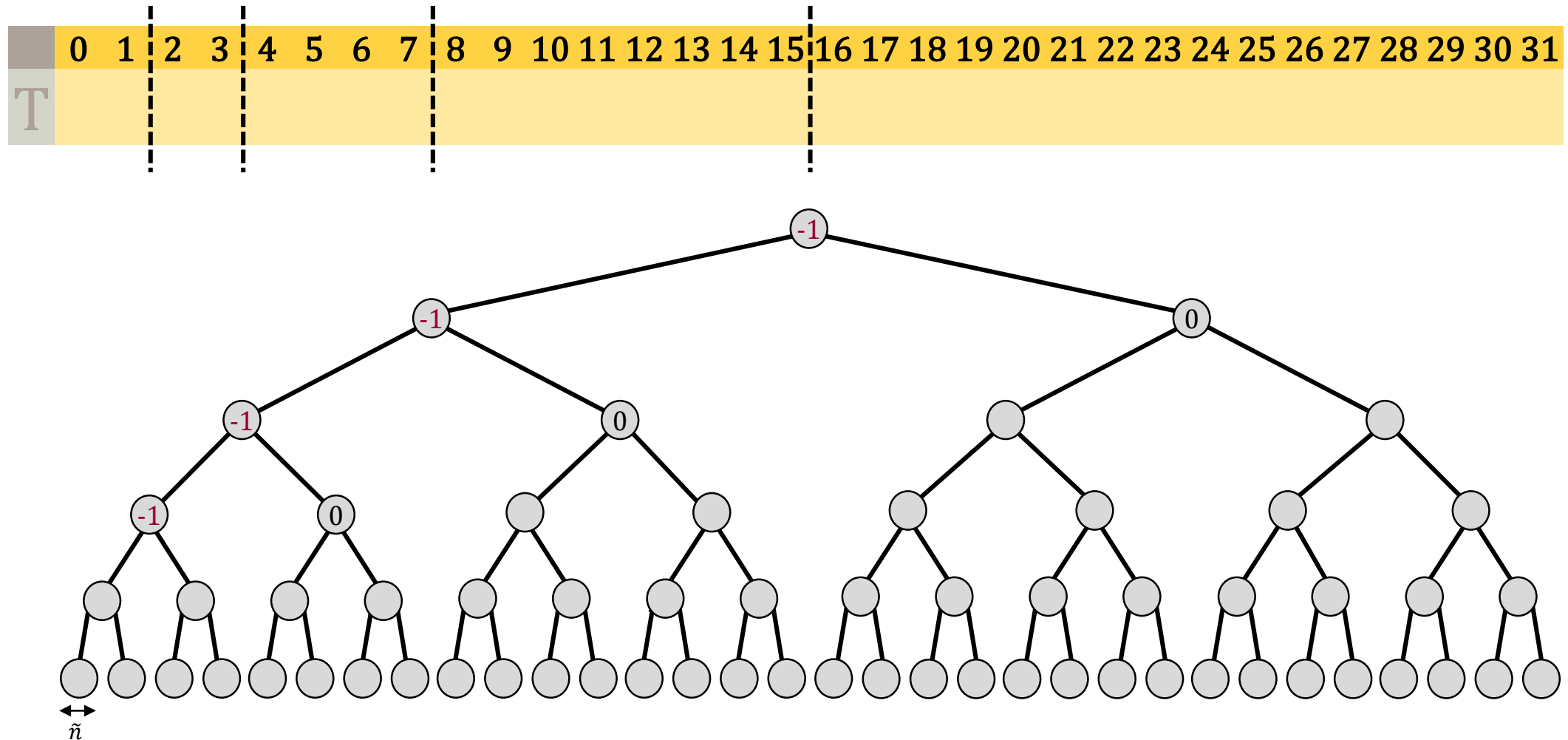




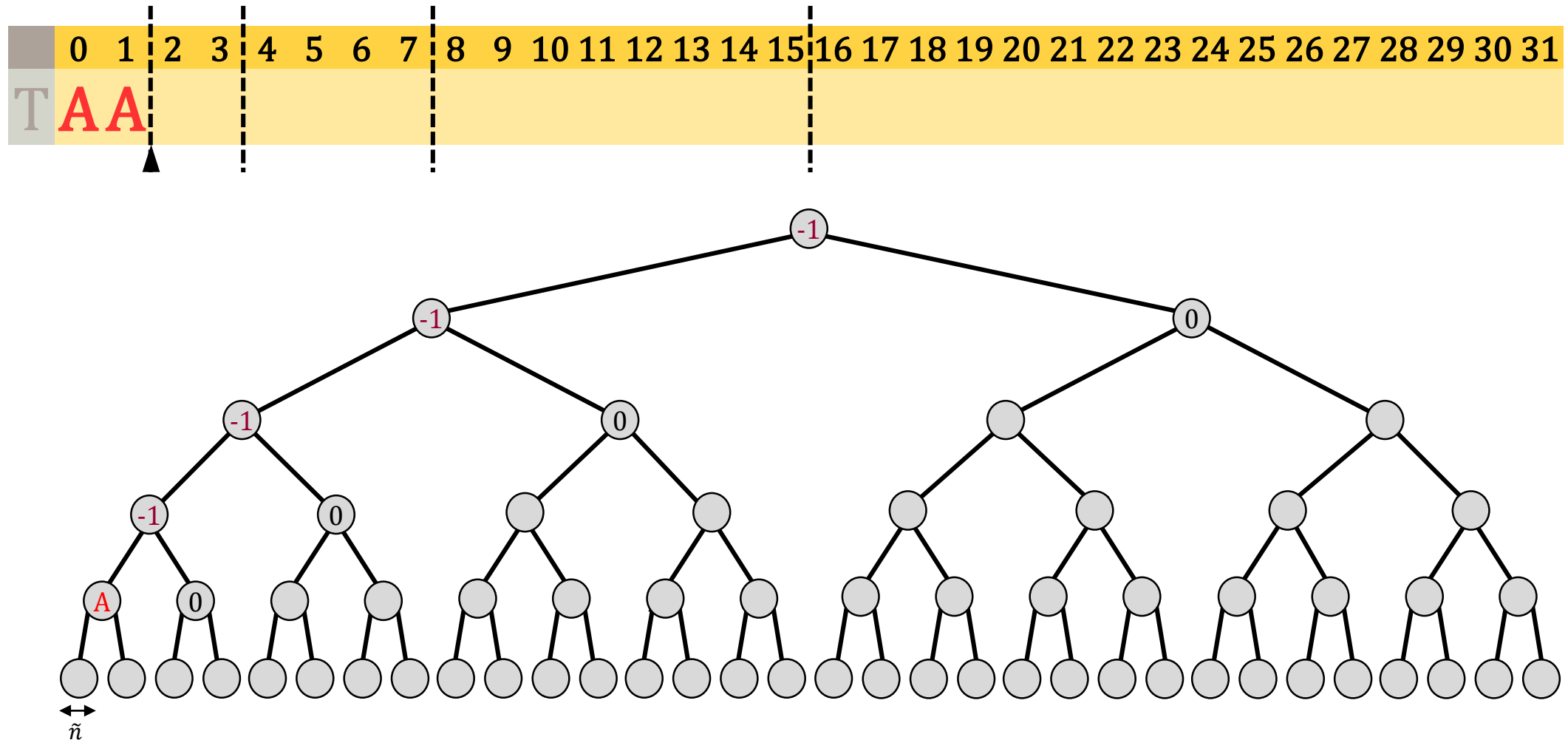
# Buddy-Verfahren (A Reservieren III)



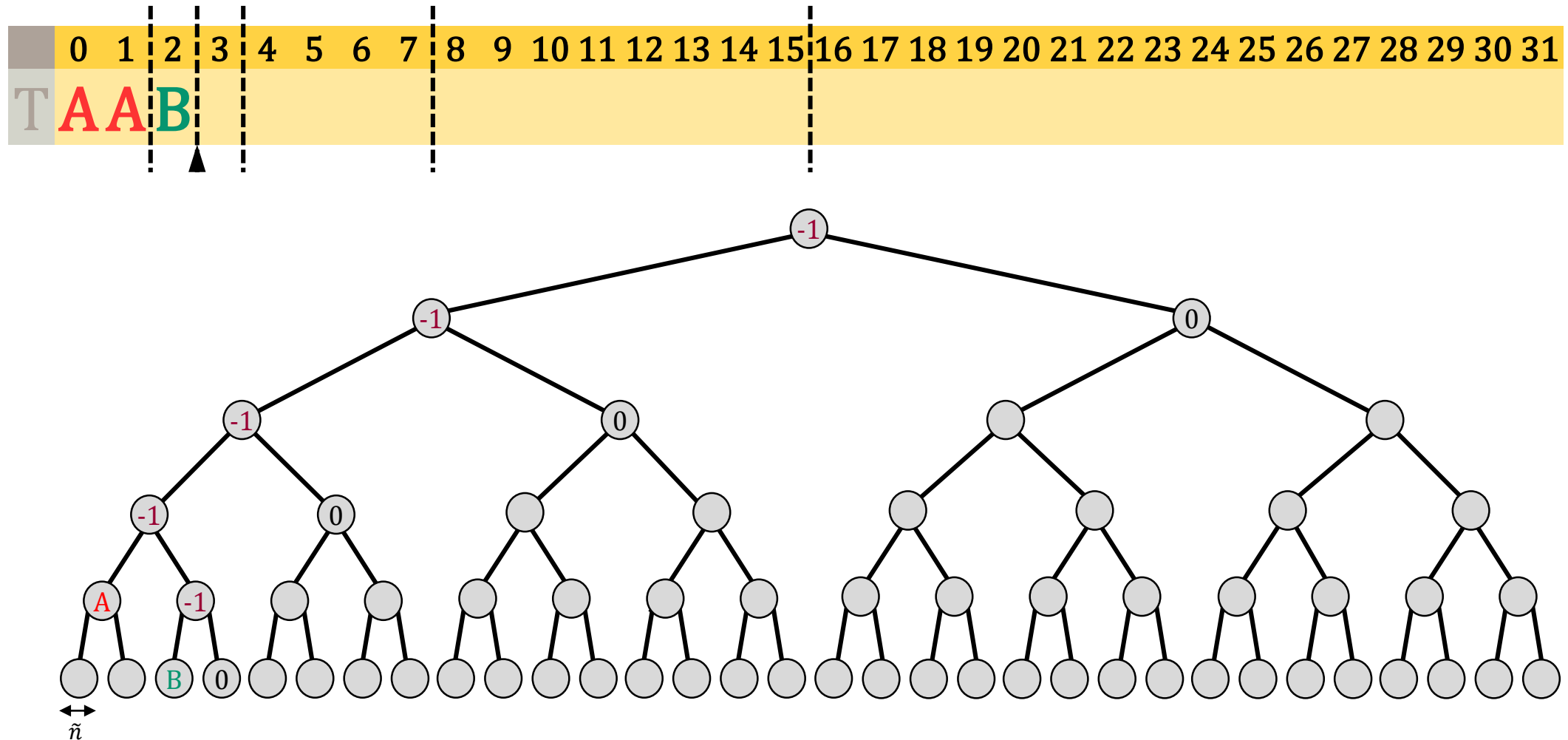
# Buddy-Verfahren (A Reservieren IV)



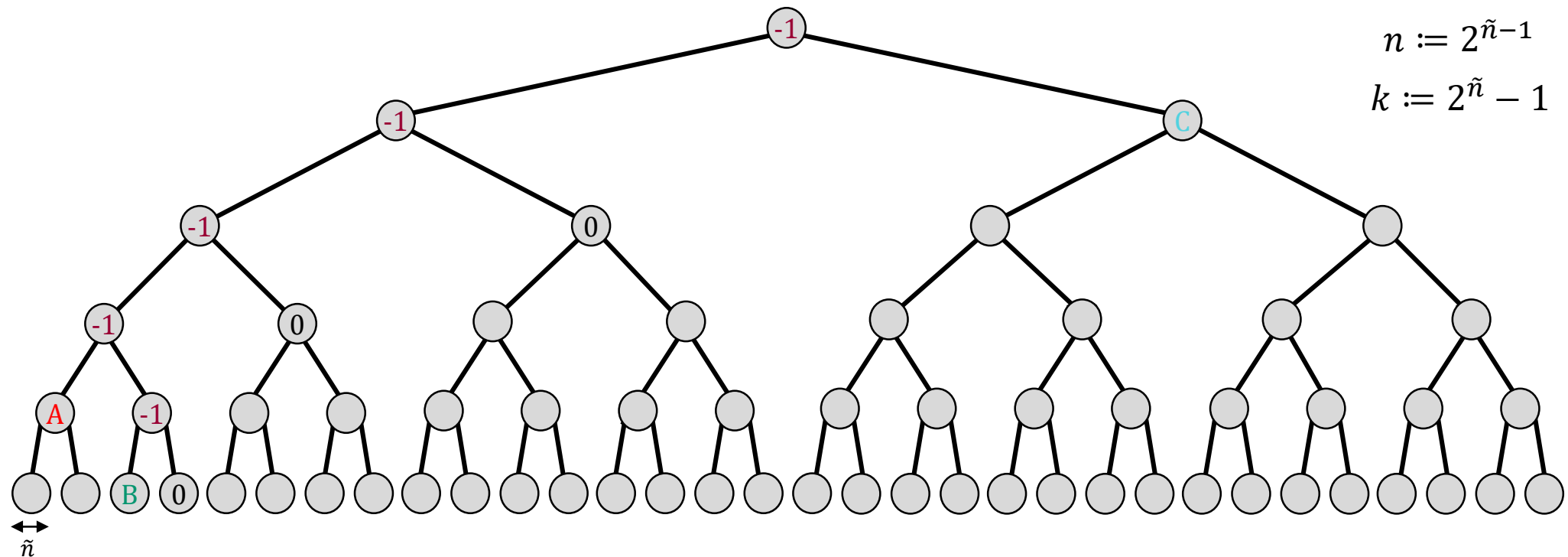
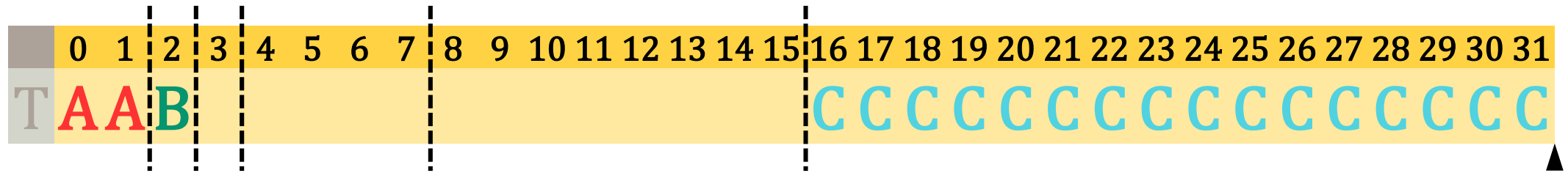
# Buddy-Verfahren (A Reservieren)



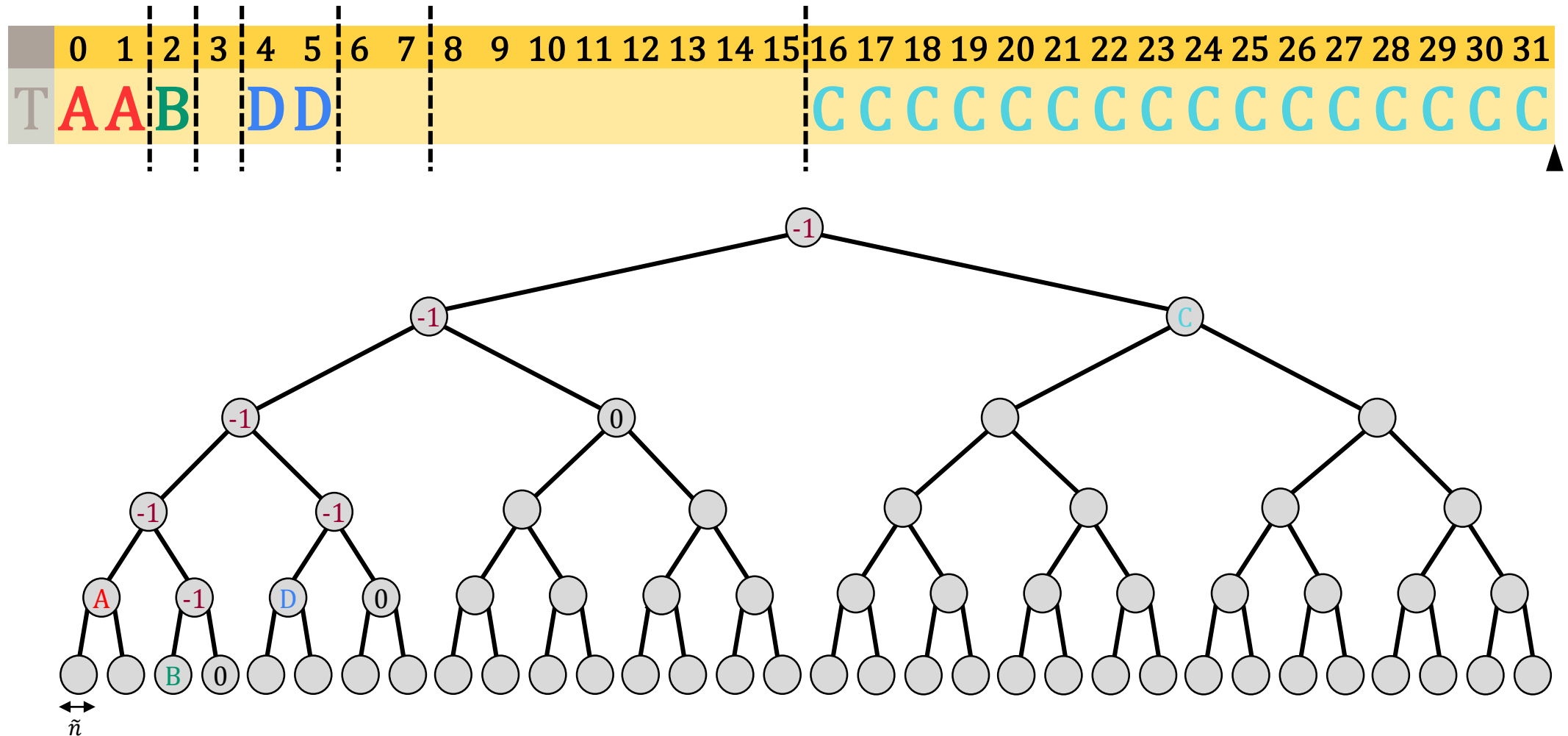
# Buddy-Verfahren (B Reservieren)



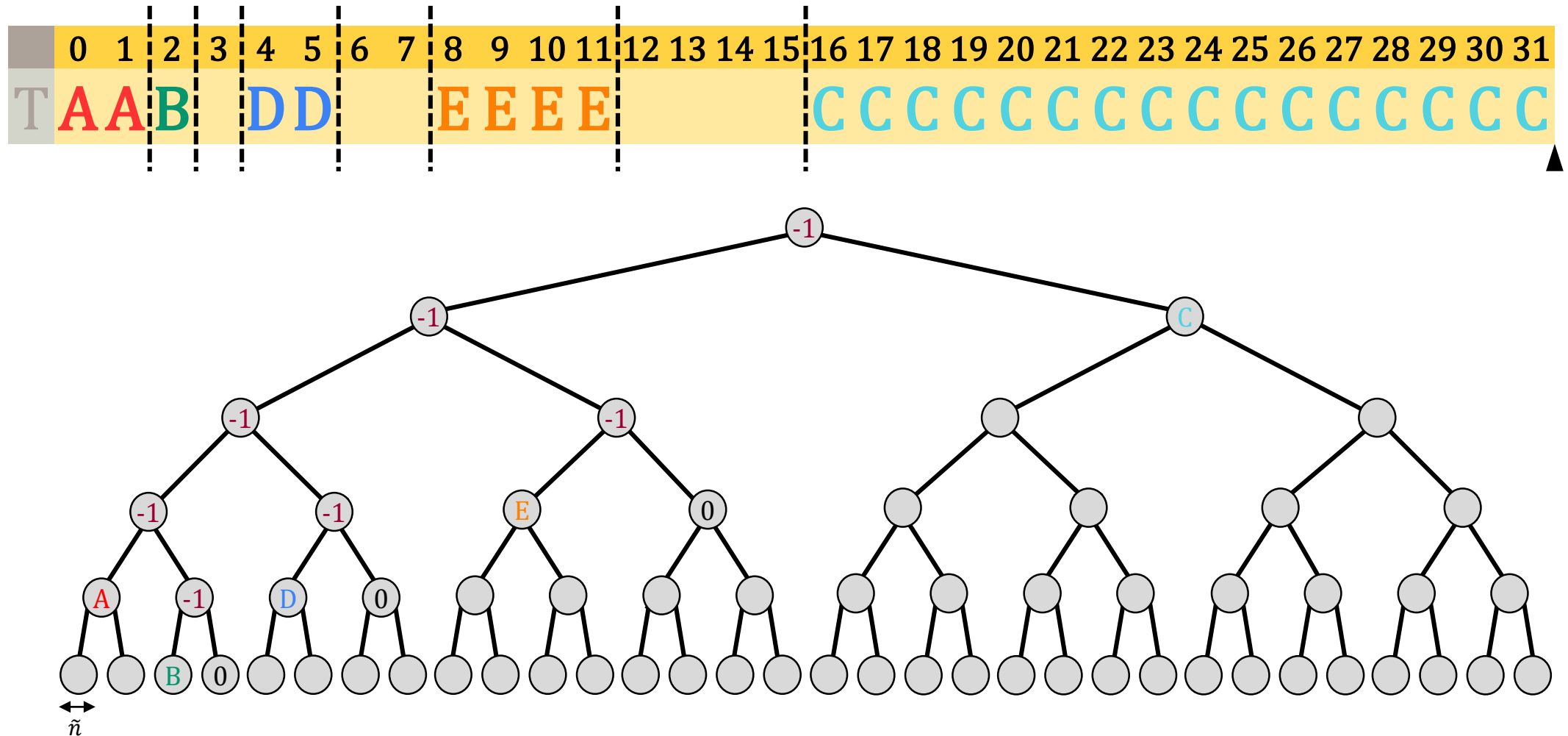
# Buddy-Verfahren (C Reservieren)



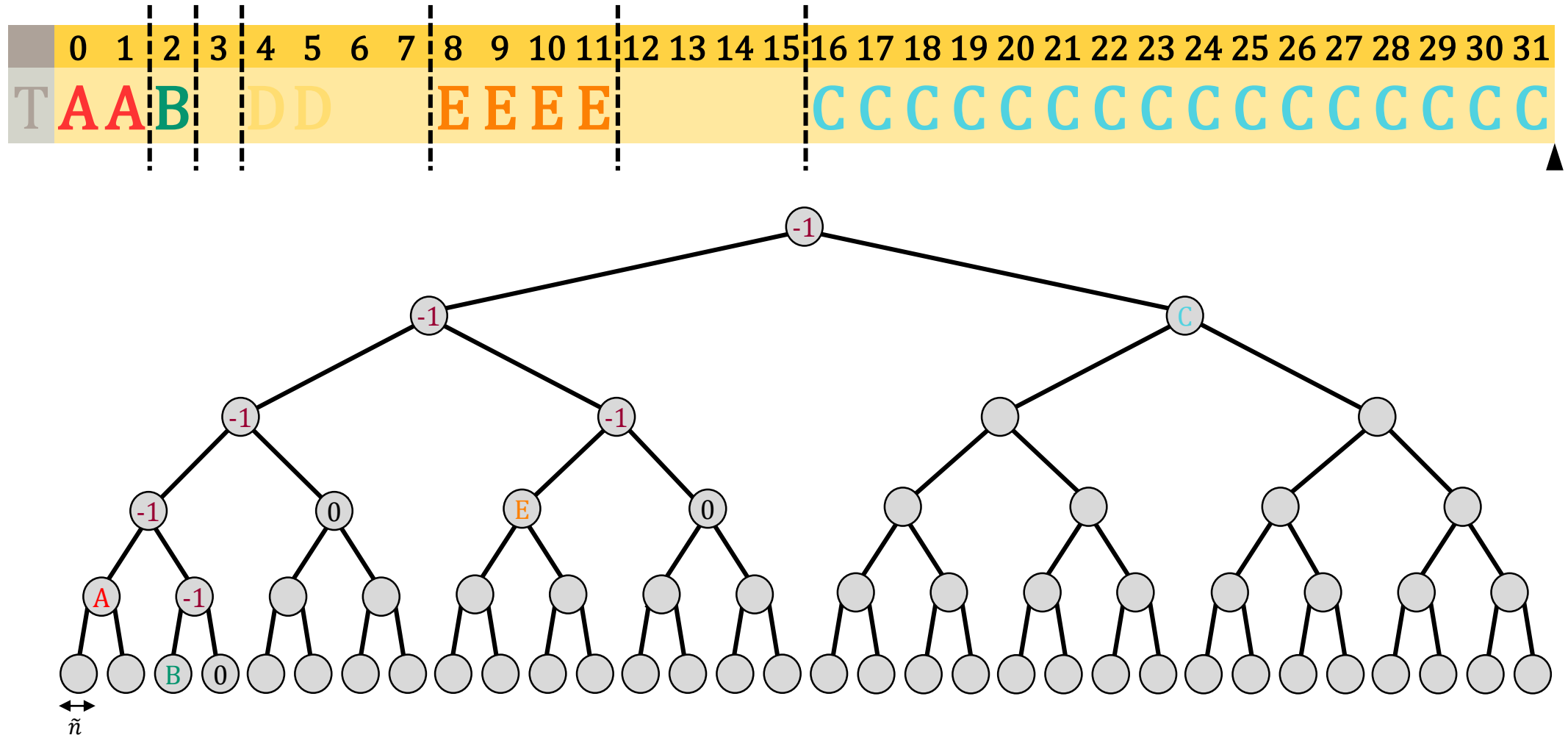
# Buddy-Verfahren (D Reservieren)



# Buddy-Verfahren (E Reservieren)



# Buddy-Verfahren (D Freigeben)





# Buddy-Verfahren (A Freigeben)

