plcpcomp

Patrick Dinklage

Jonas Ellert

Johannes Fischer

Dominik Köppl

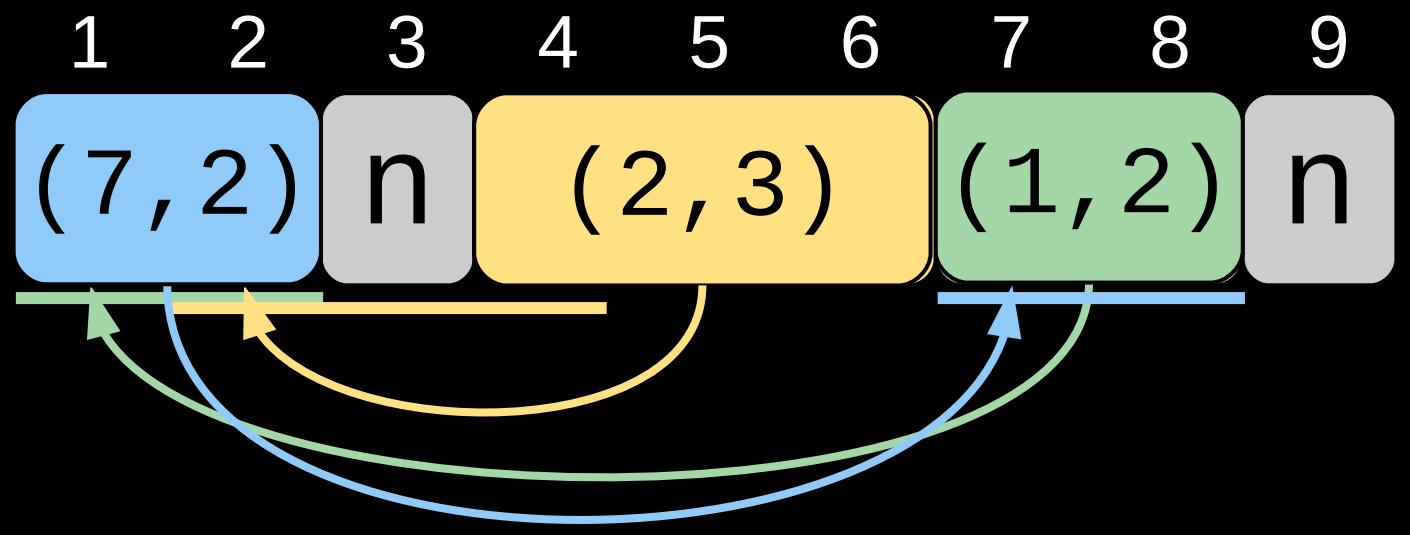
Manuel Penschuck

greedy compression

scheme	method	ref
LZ76	unidirectional	Lempel,Ziv '76
longest first	grammar	中村 + '09
lcpcomp	bidirectional	Dinklage+ '17
LZ-LFS	hybrid	Mauer+ '17 西 + '18
自己参照なし	without self-ref	藤原 + '19
unicomp	unidirectional	峰松 + '19

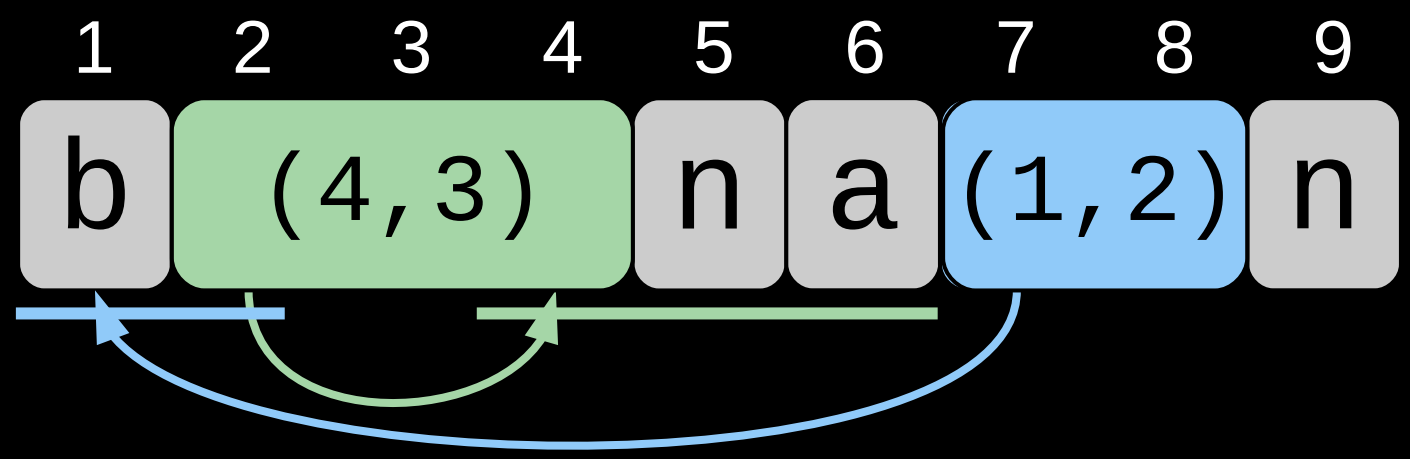
bidirectional compression

- replace substrings by references
- in any direction
- allow self-references
- no cycles



lcpcomp

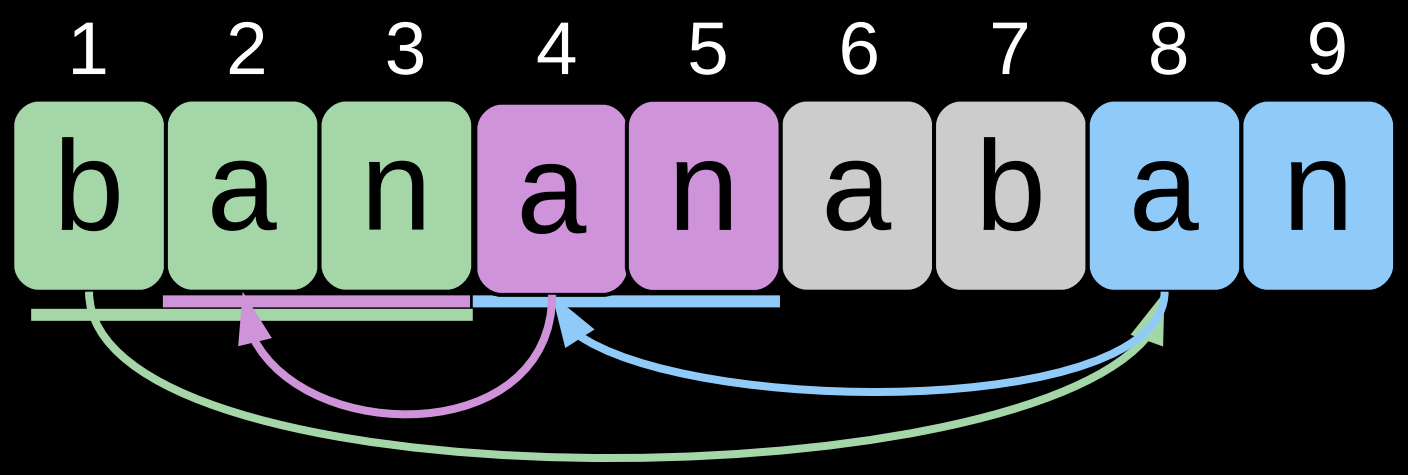
- replace longest reoccurring substring
- recurse



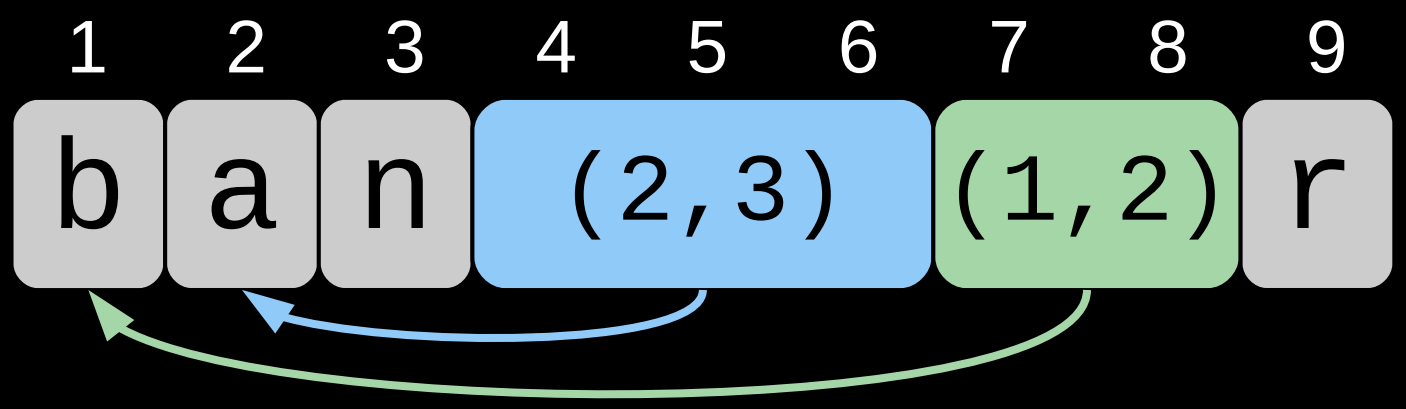
*Starting from position **4**, copy **3** symbols.*
*Starting from position **1**, copy **2** symbols.*

cycles

- need restriction to prevent cycles
- rule: reference to *lexicographically smaller* suffix



~ LZ77

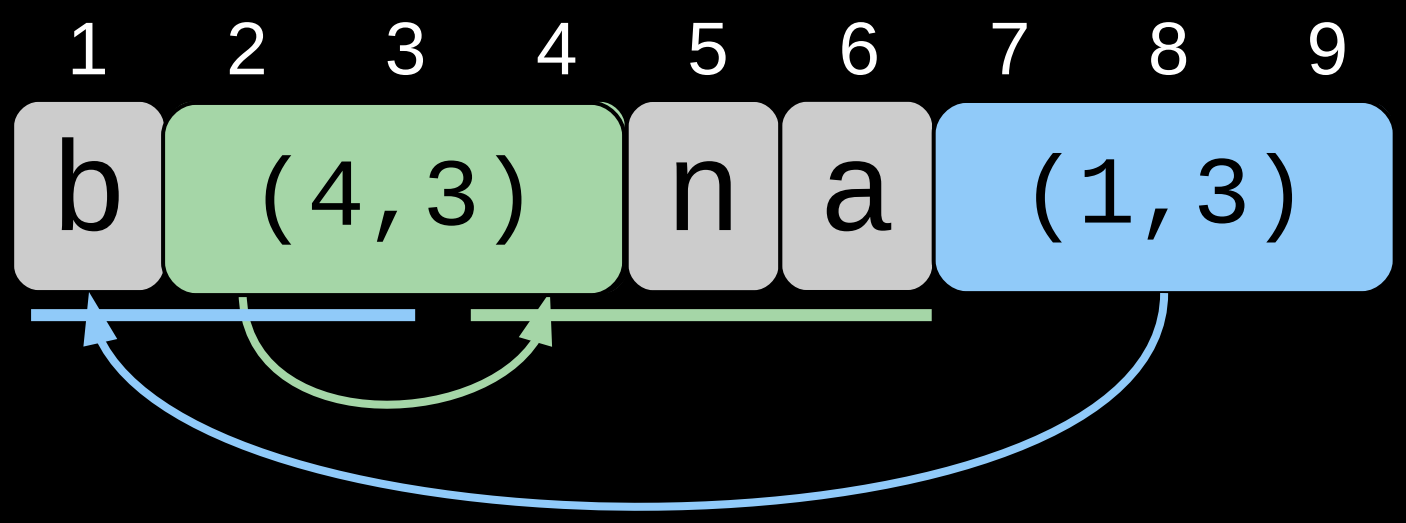


- LZ77
  - reference to *previous text position*
- lcpcomp
  - reference to *lexicographically smaller* suffix

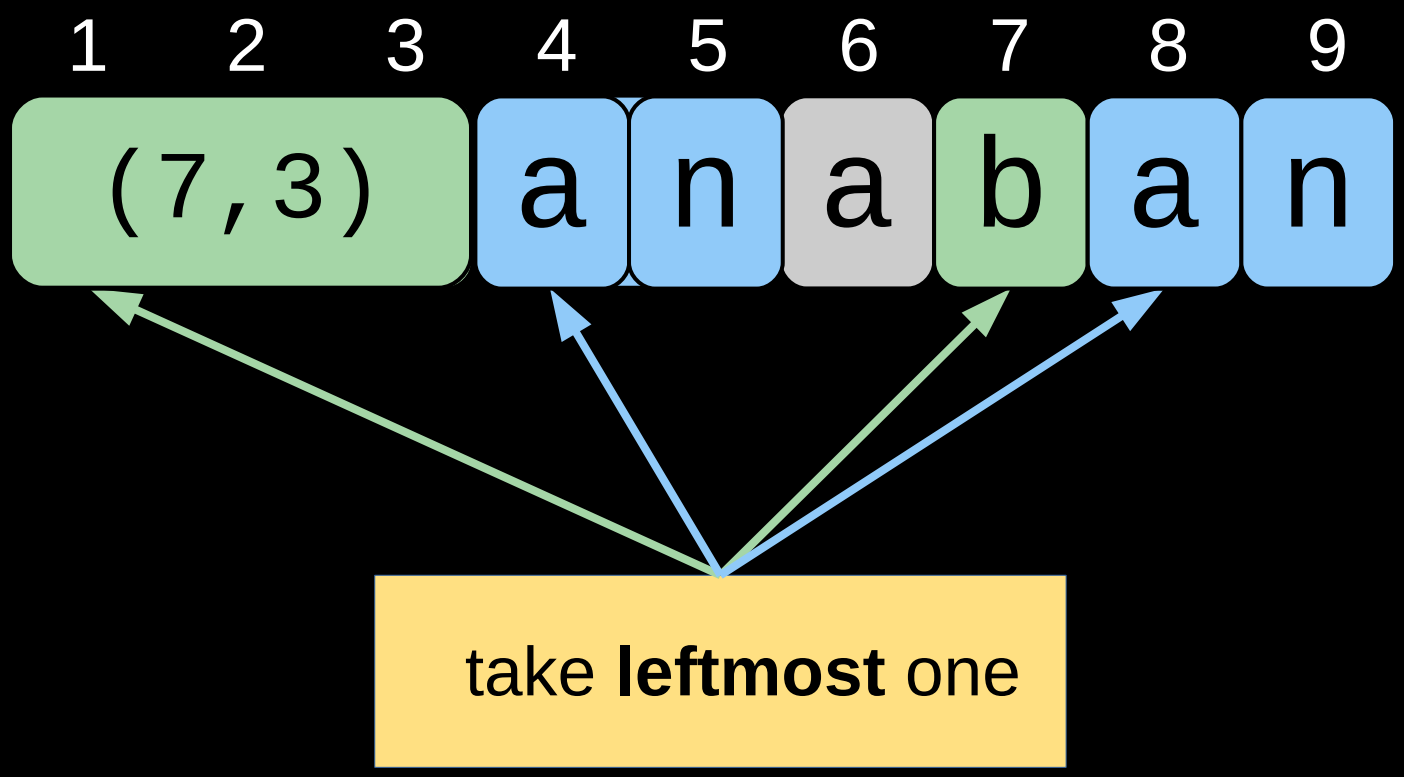
=> both greedy + reference order constraint

lcpcomp

- multiple choices
- order not clear



plcpcomp: tie break



how?

find

- longest re-occuring substring S
- = longest common prefix (LCP) of two suffixes

by

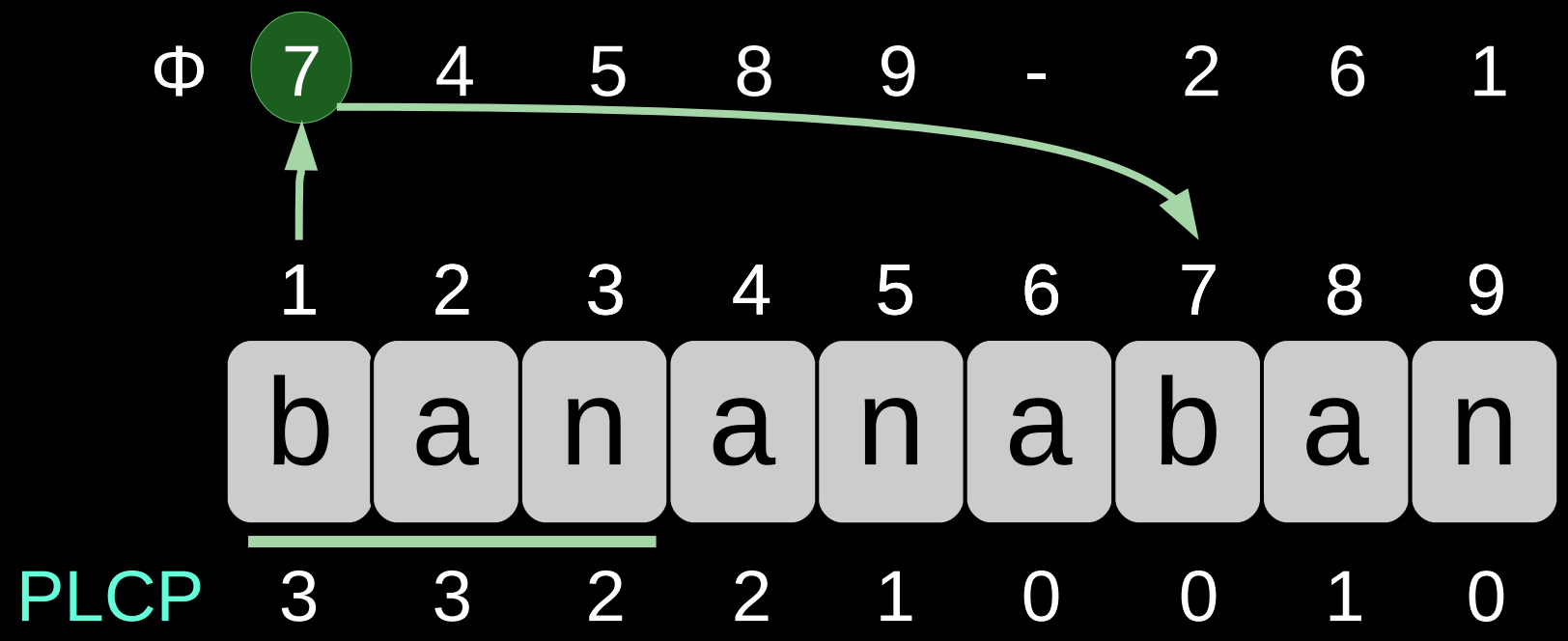
- |S| = PLCP[i]
- S starts at argmax PLCP[i]
- PLCP[i] = LCP of i-th suffix and Φ[i]-th suffix

Φ array

$\Phi[i] := SA[ISA[i] - 1]$

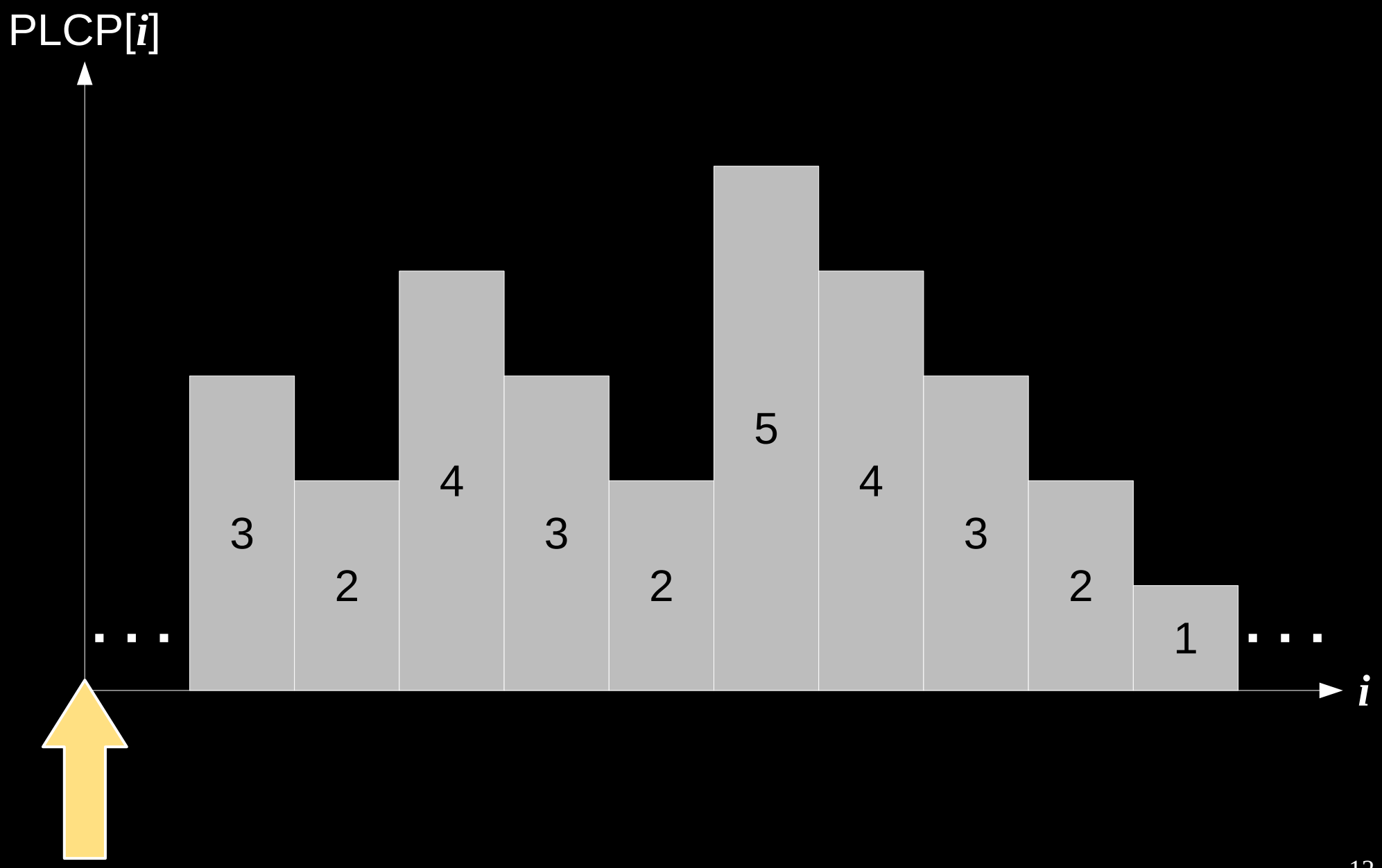
Φ	7	4	5	8	9	-	2	6	1
SA	6	8	4	2	7	1	9	5	3
ISA	6	4	9	3	8	1	5	2	7
	1	2	3	4	5	6	7	8	9
	b	a	n	a	n	a	b	a	n

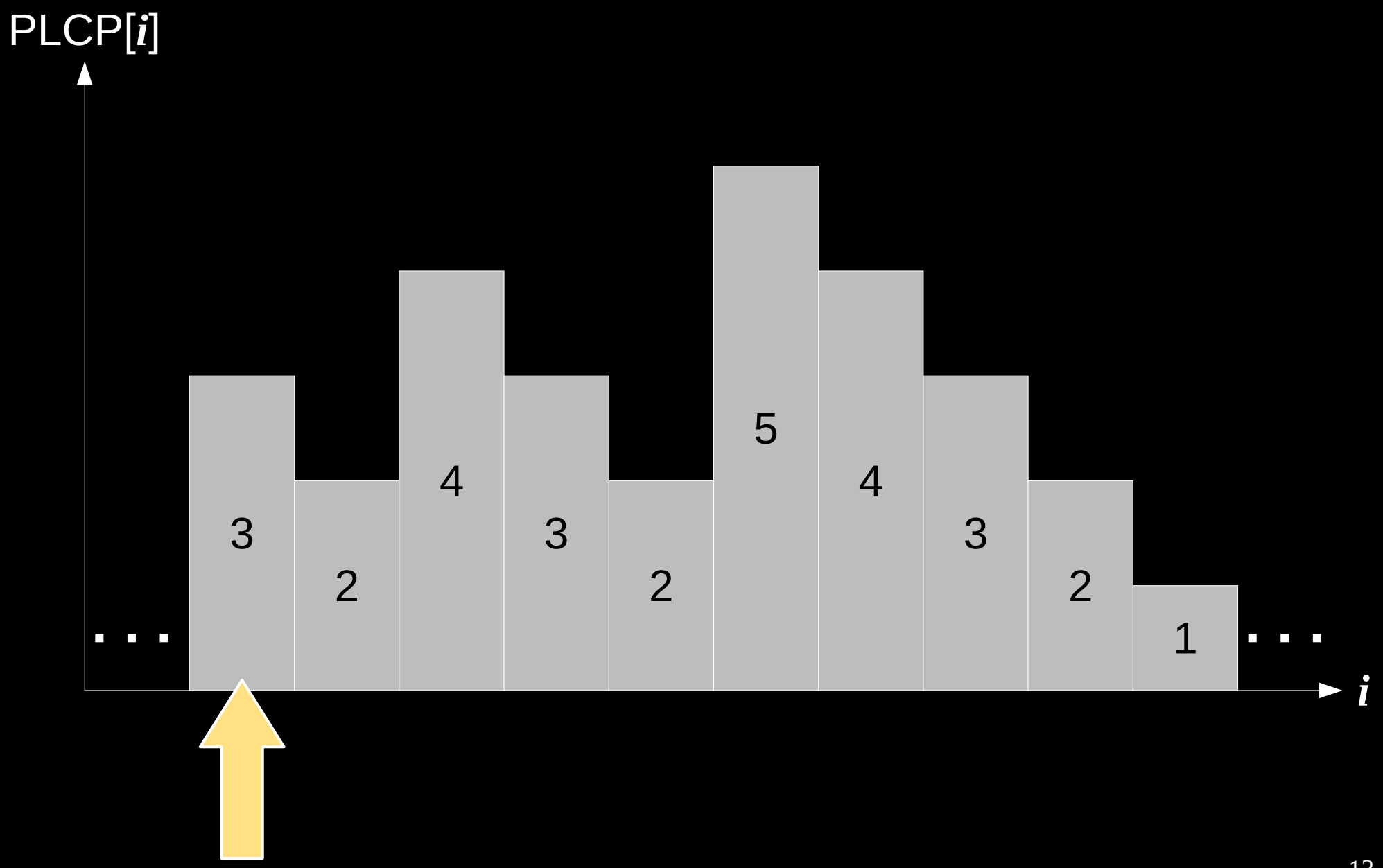
main idea

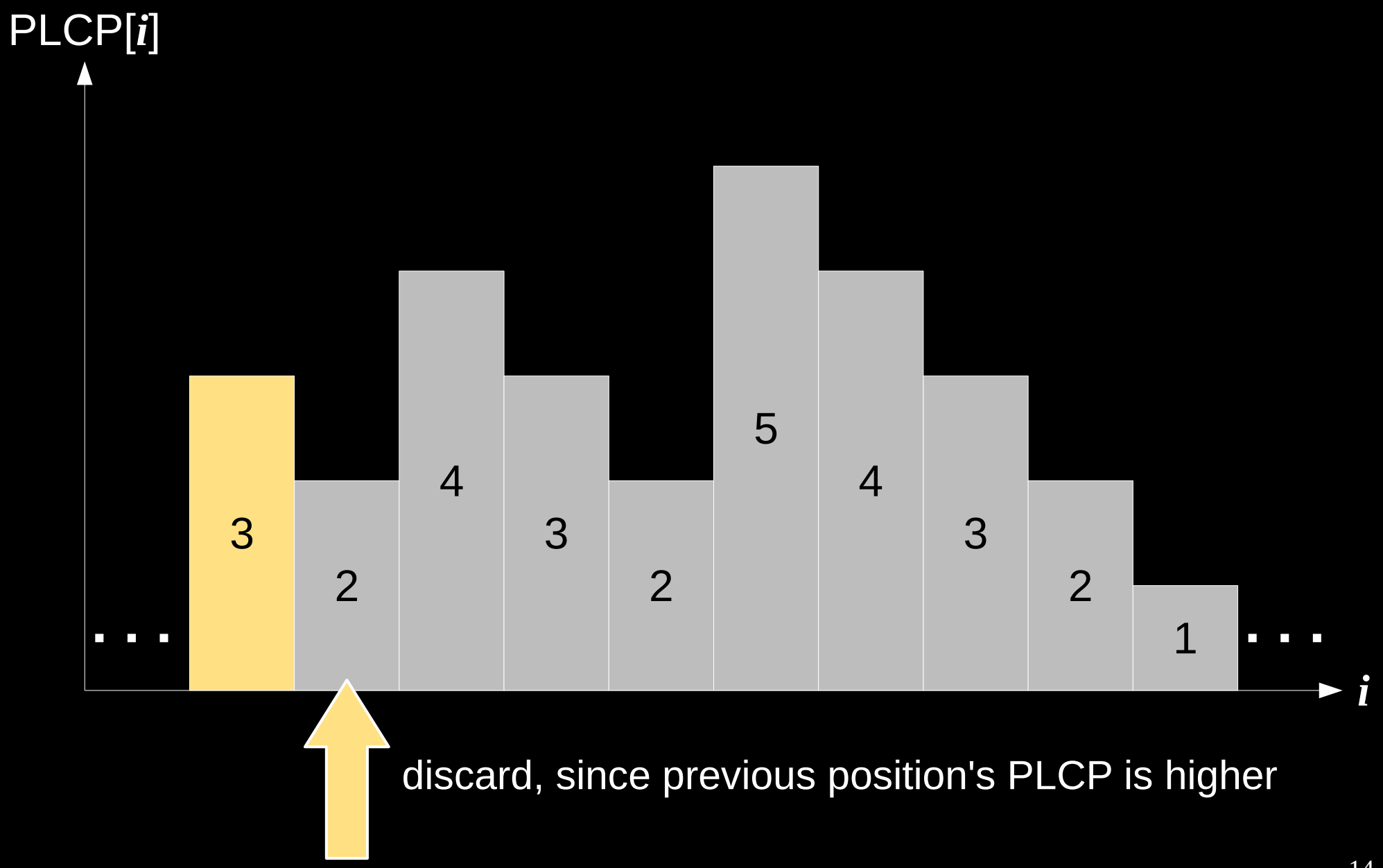


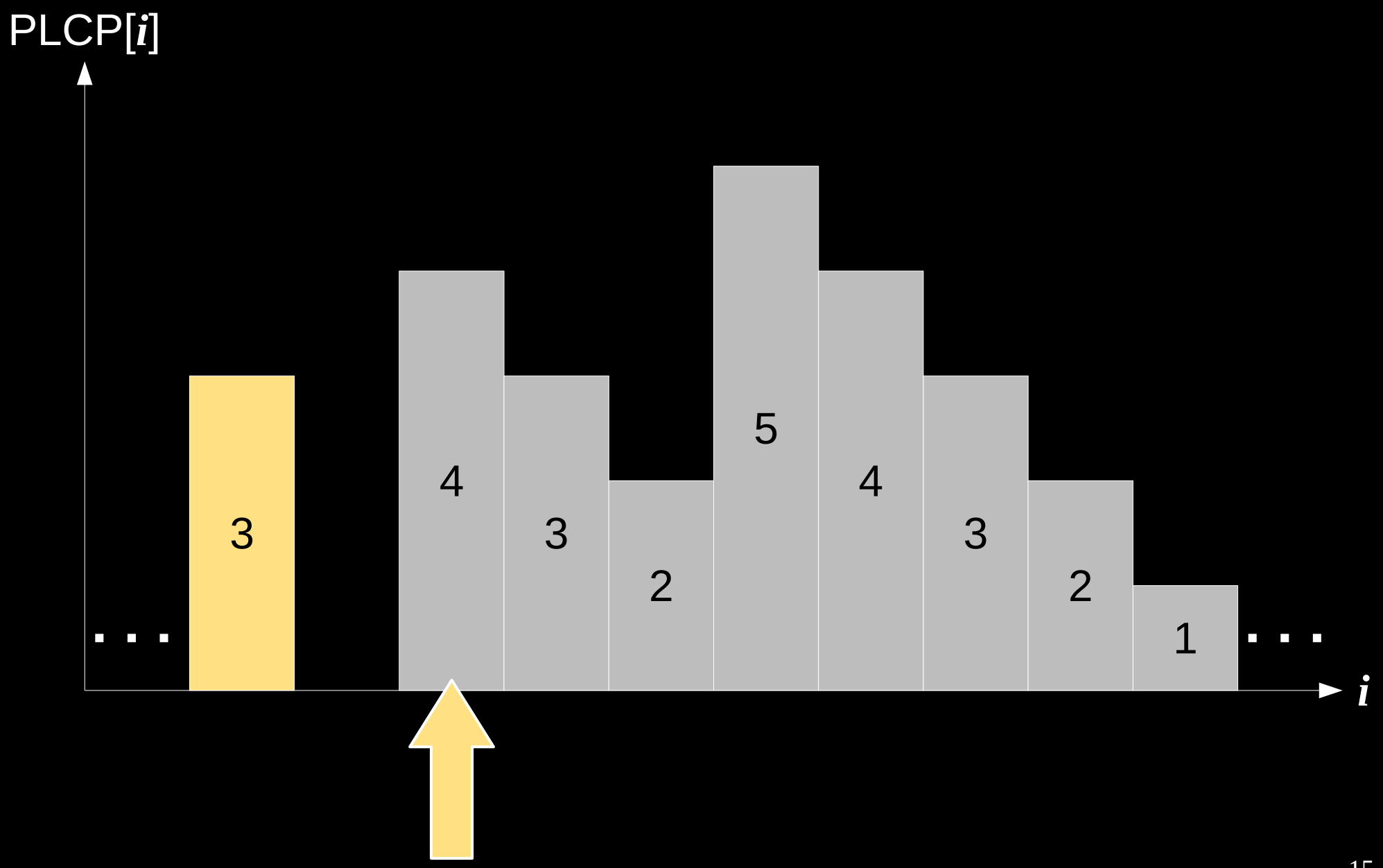
- PLCP: length of factor
- Φ: reference

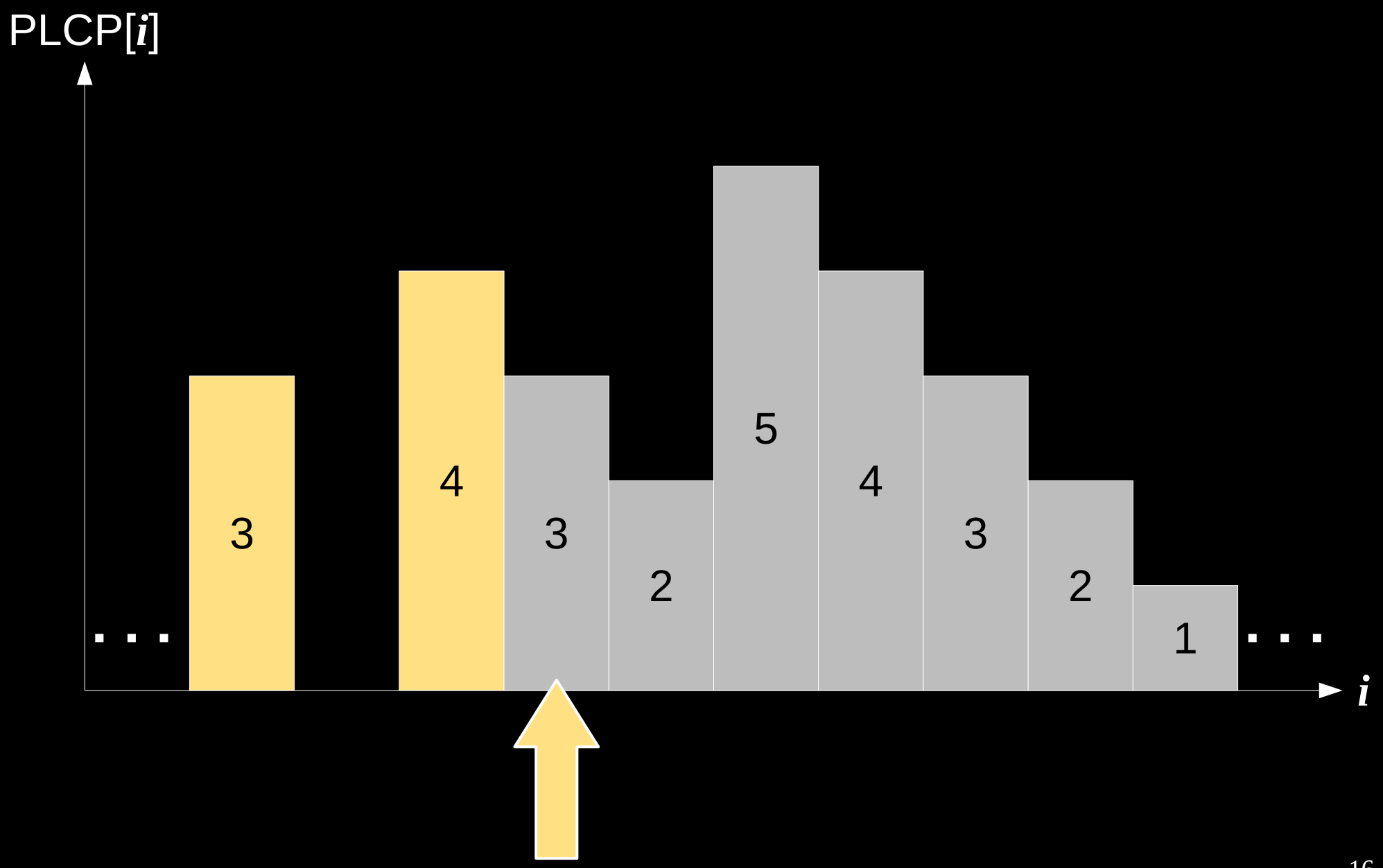
computation

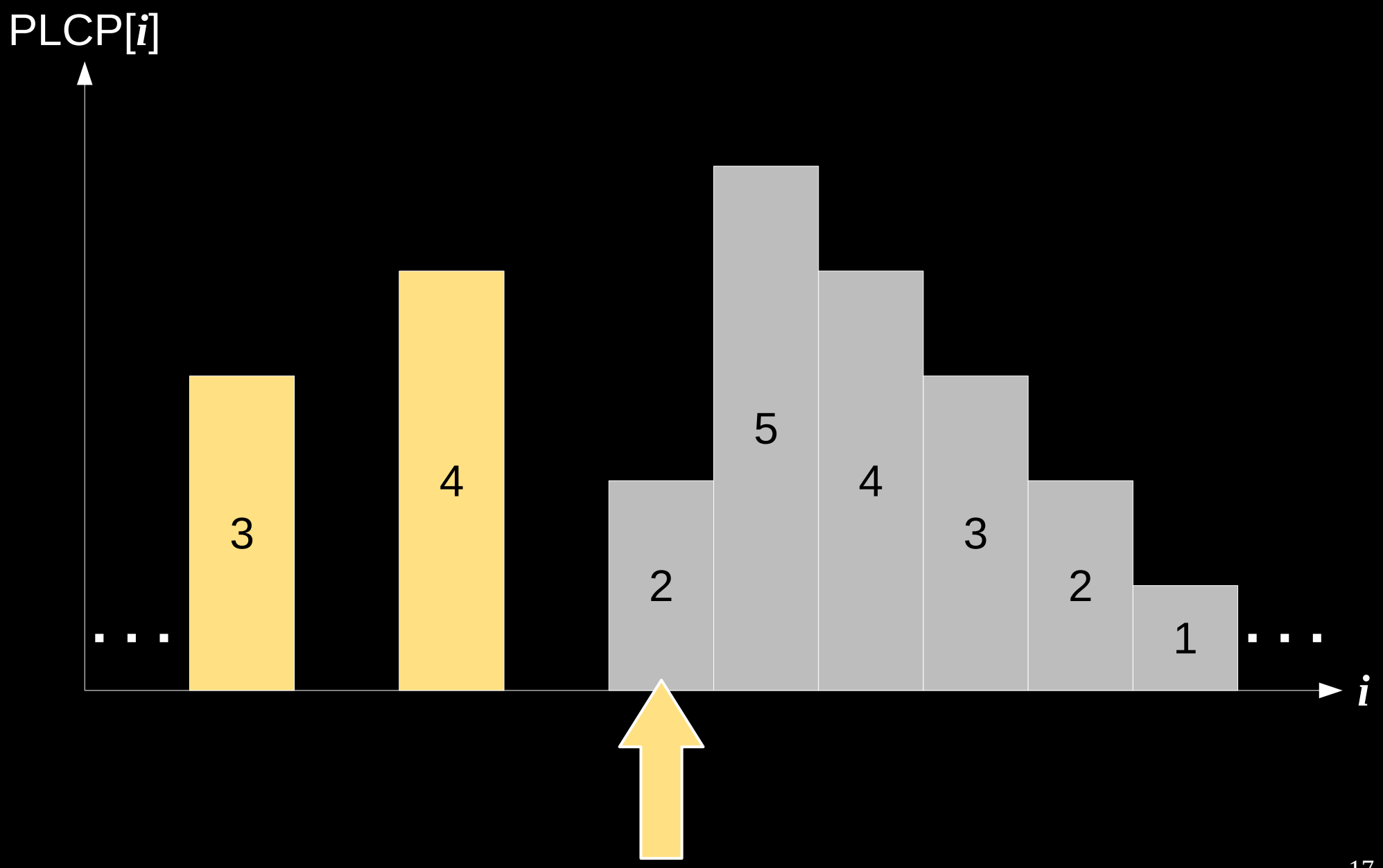


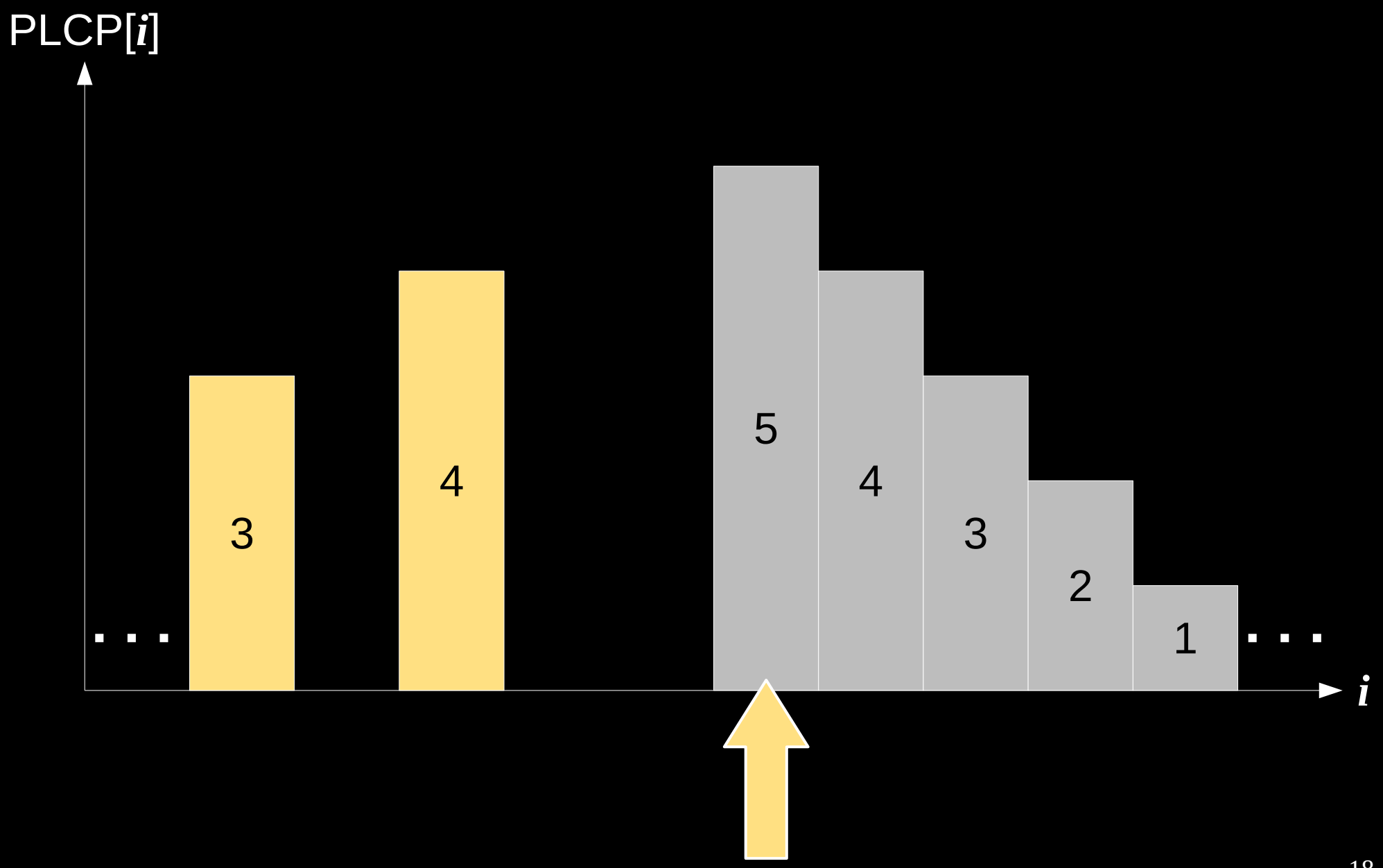

PLCP[i]
3
2
4
3
2
5
4
3
2
1
i

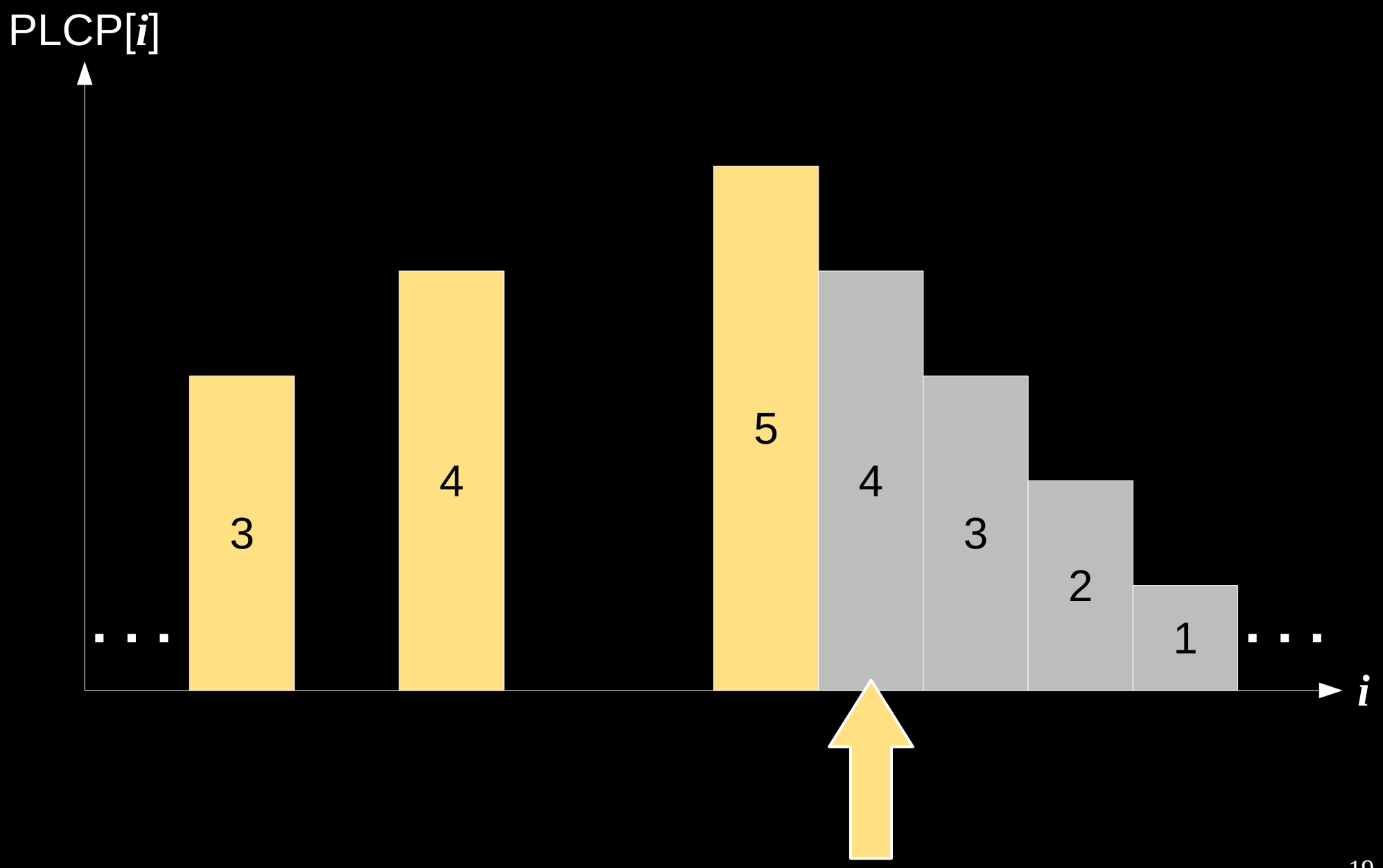

PLCP[i]
3
2
4
3
2
5
4
3
2
1
i
discard, since previous position's PLCP is higher

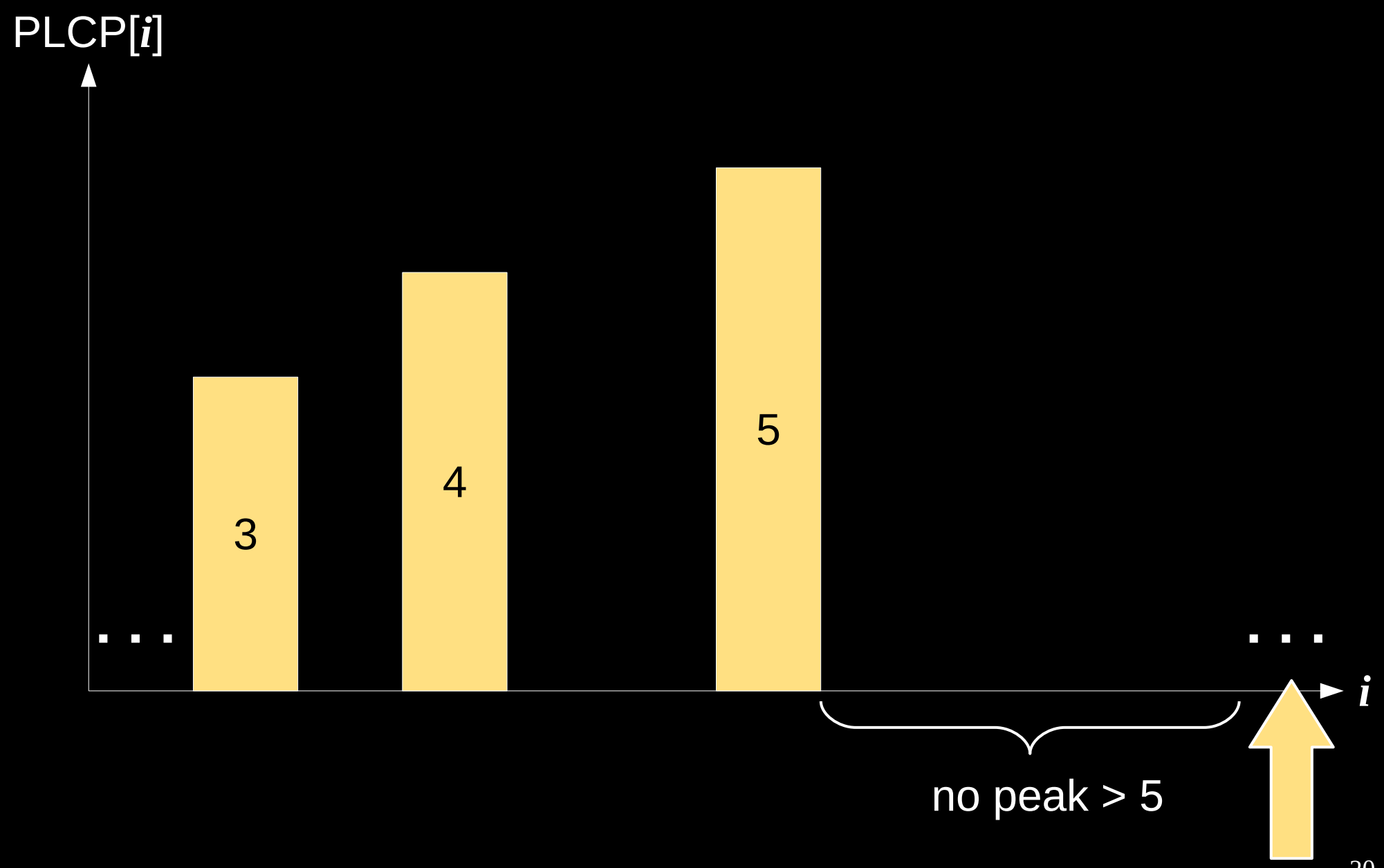

PLCP[i]
3
4
3
2
5
4
3
2
1
i

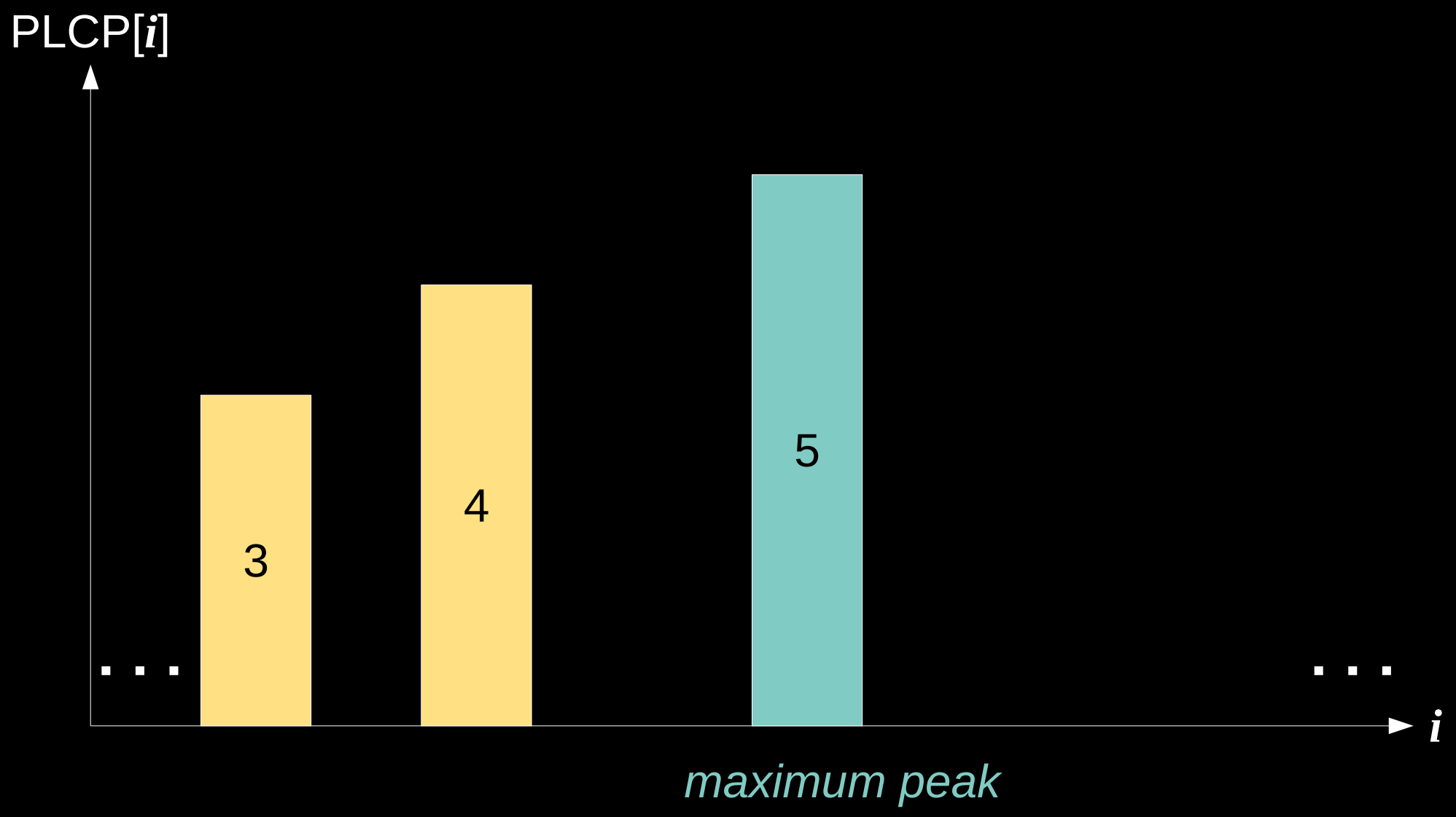

PLCP[i]
3
4
3
2
5
4
3
2
1
i

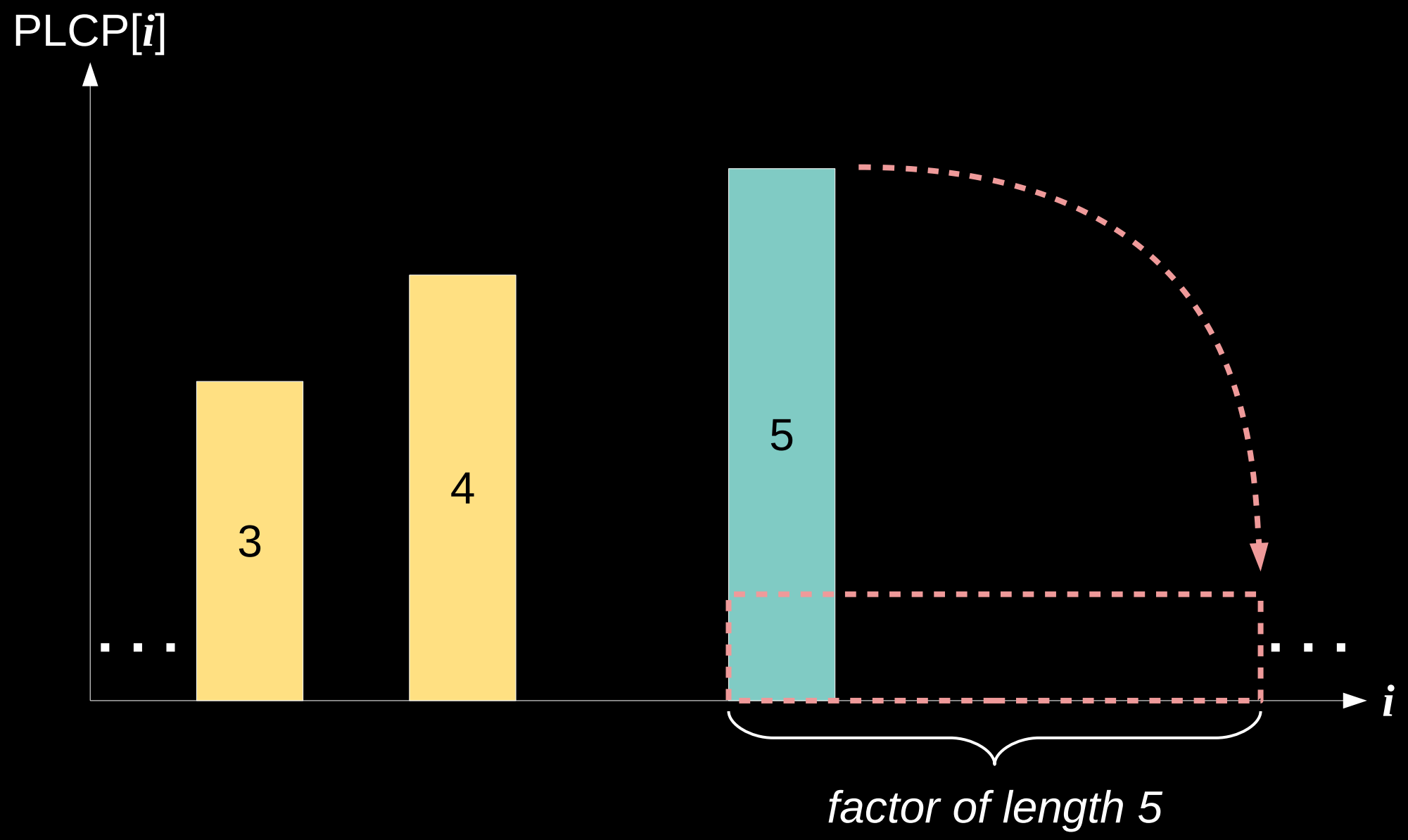

PLCP[i]
3
4
2
5
4
3
2
1
i

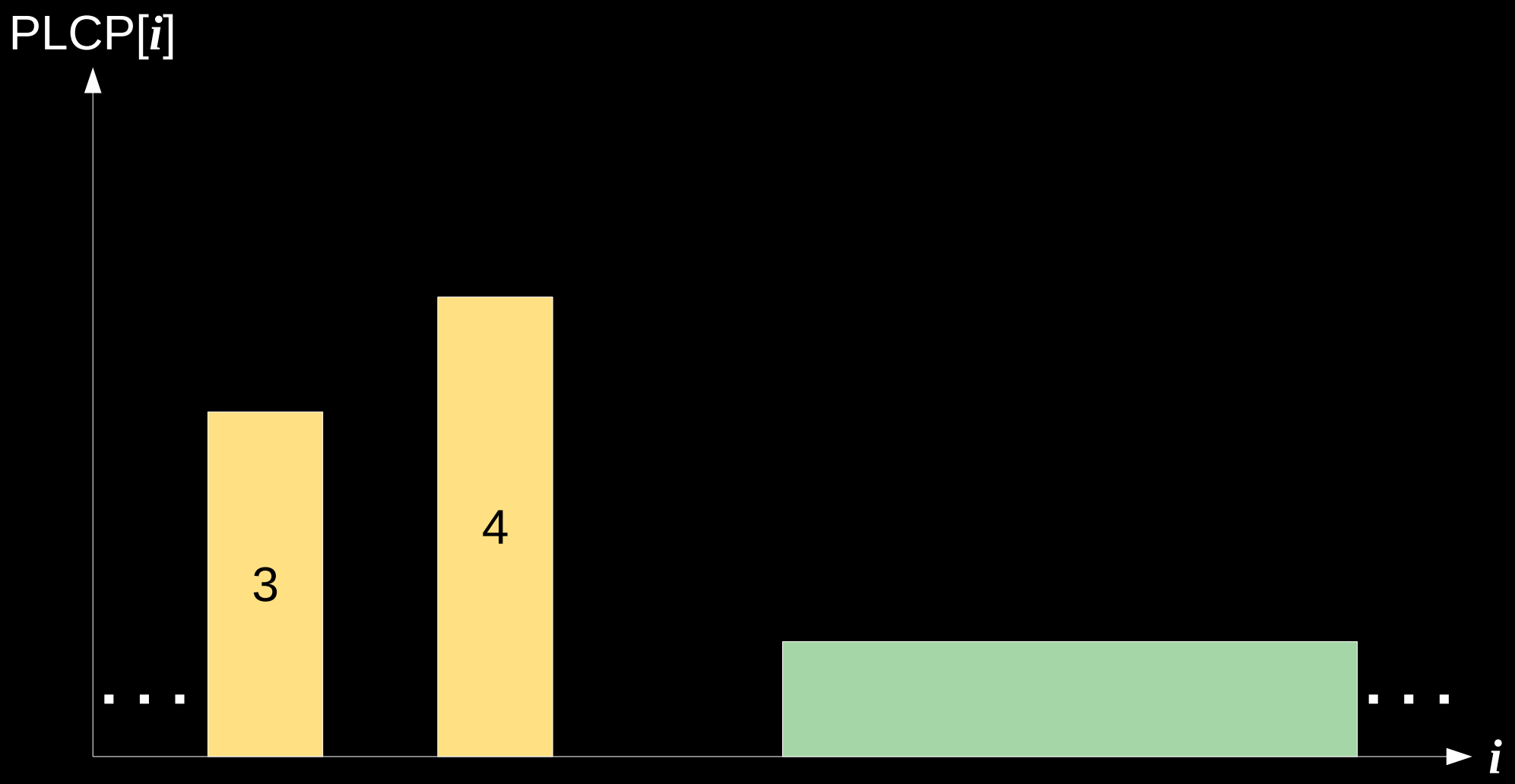

PLCP[*i*]
3
4
5
4
3
2
1
i


PLCP[*i*]
3
4
5
4
3
2
1
i

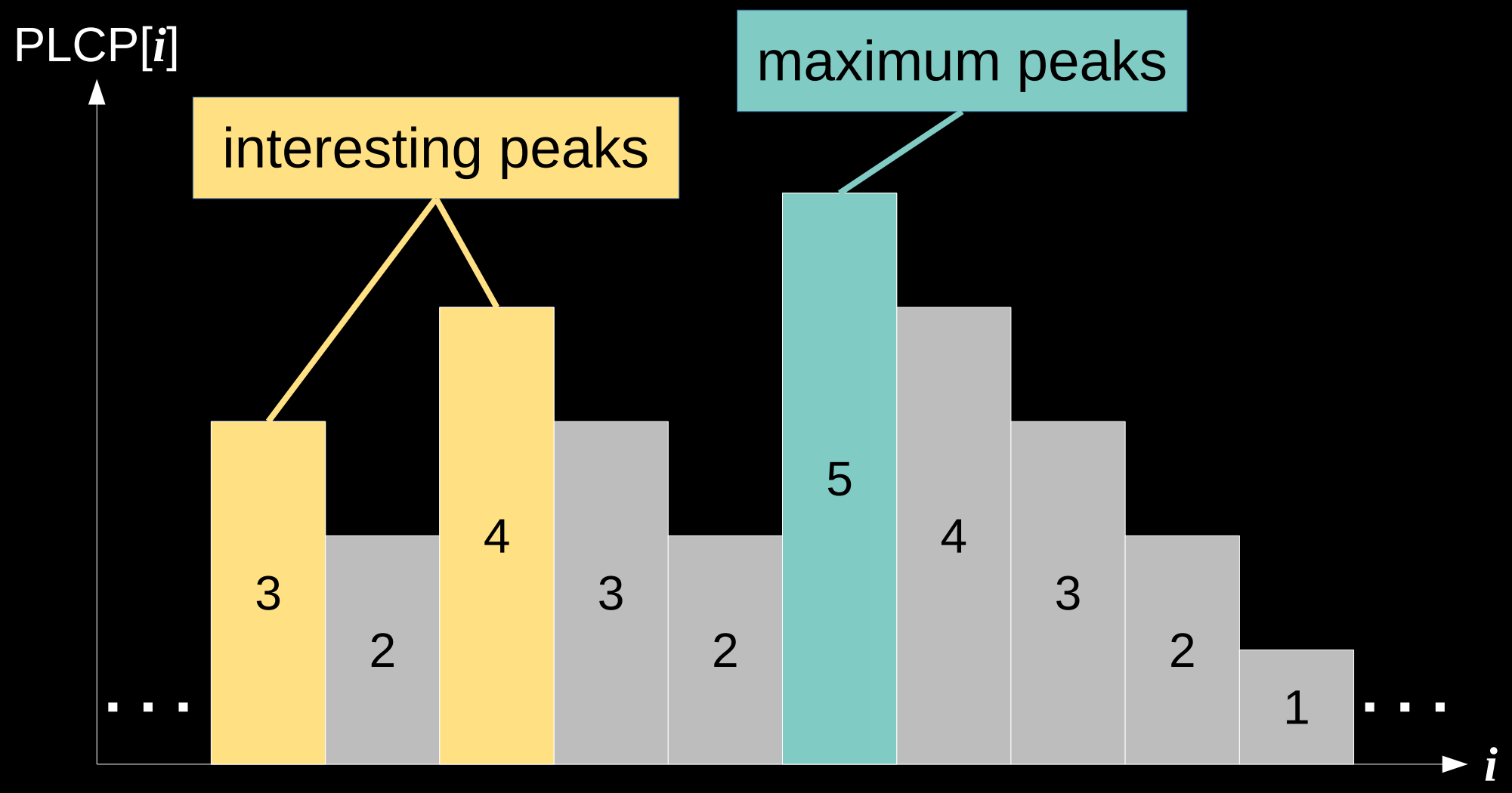

PLCP[i]
3
4
5
. . .
. . .
i
no peak > 5


PLCP[i]
3
4
5
. . .
. . .
i
maximum peak

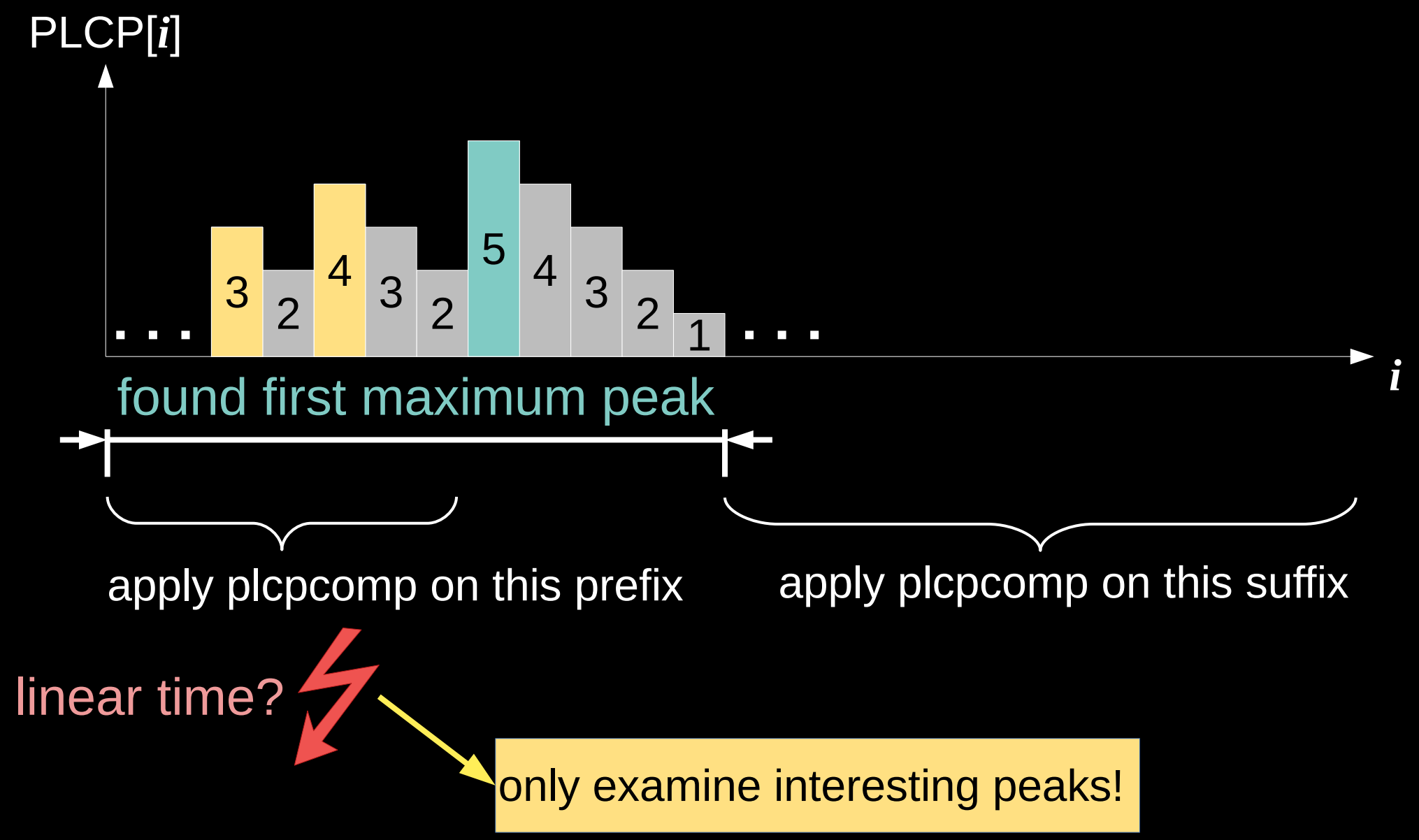

PLCP[*i*]
3
4
5
i
factor of length 5


PLCP[i]
3
4
i

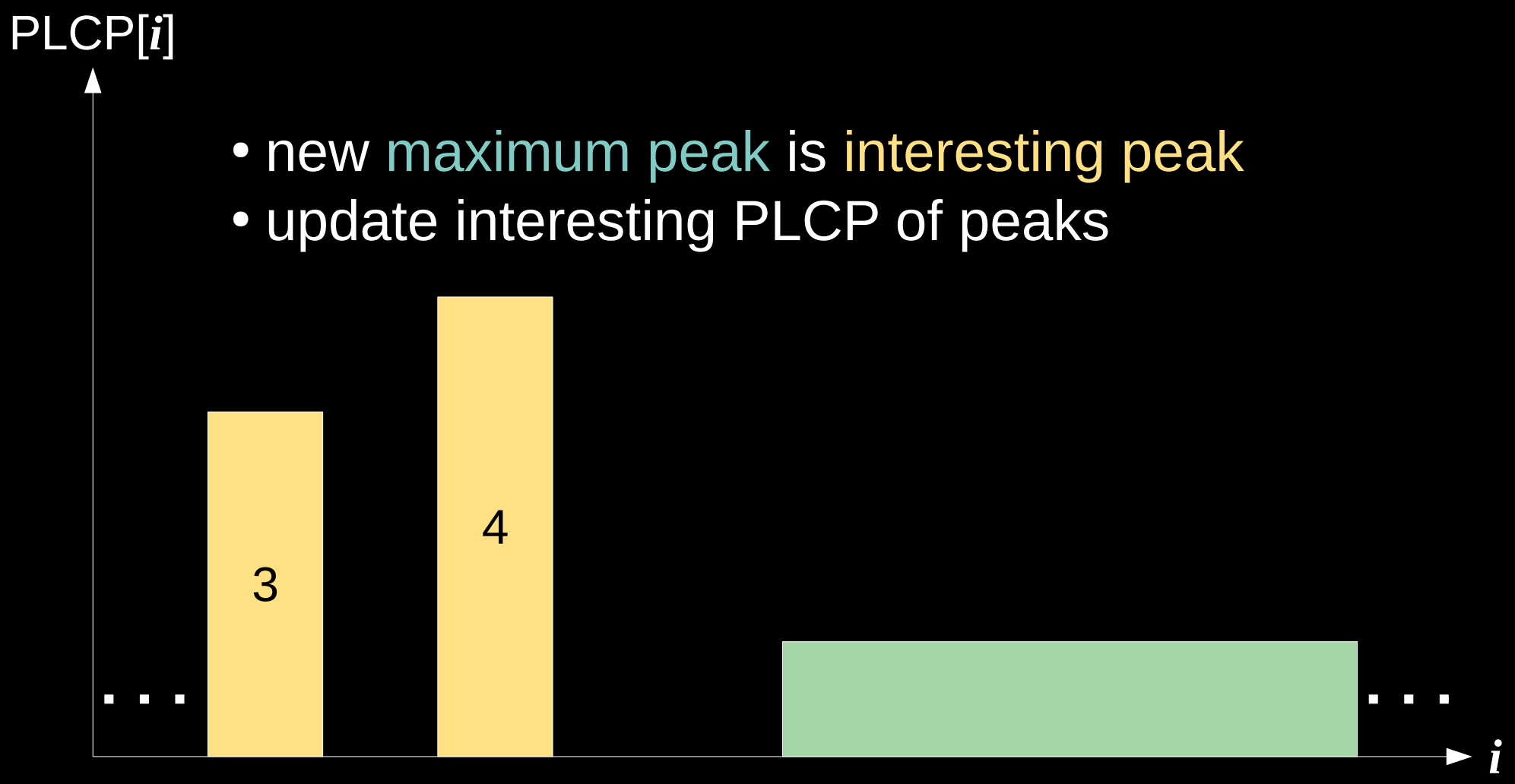
2 definitions

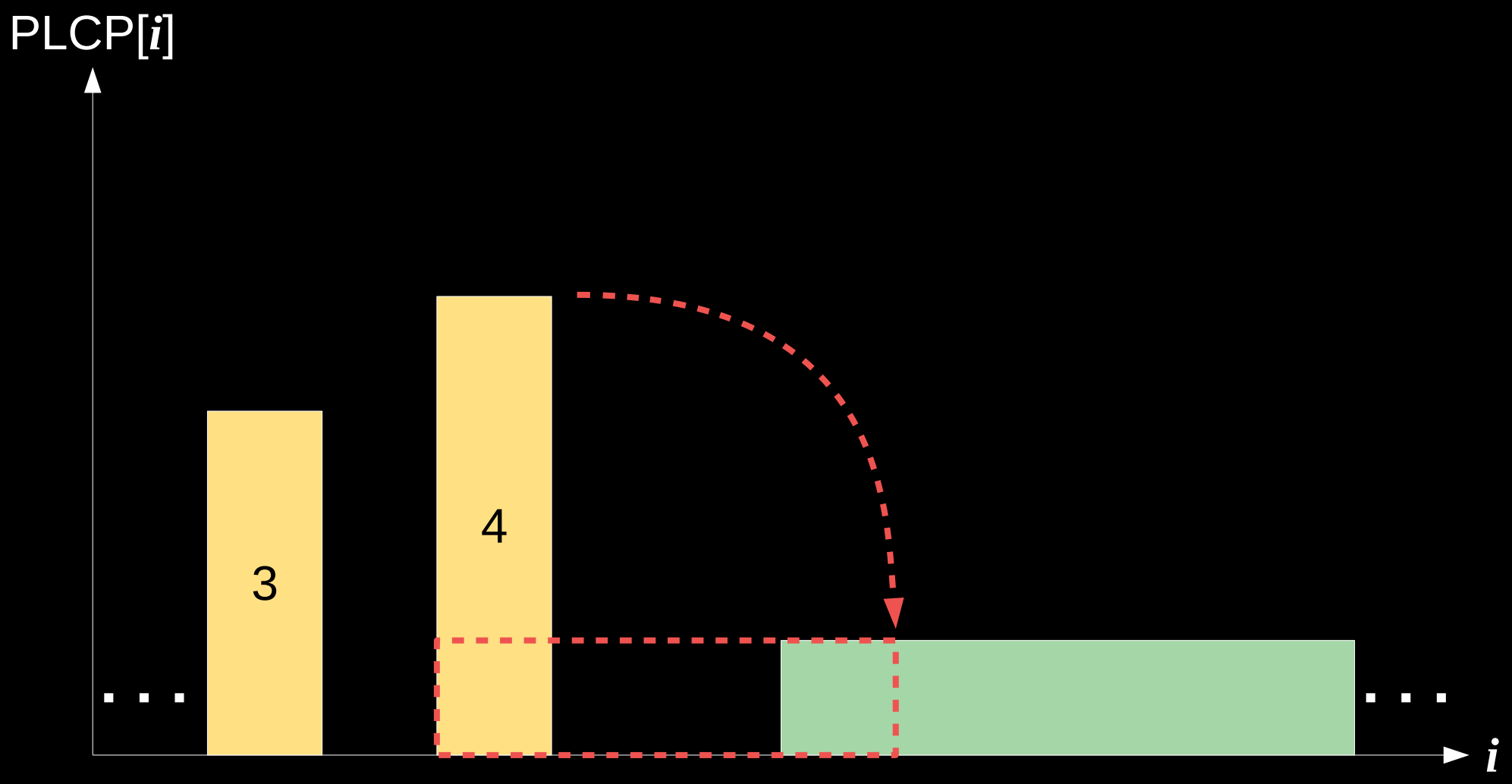


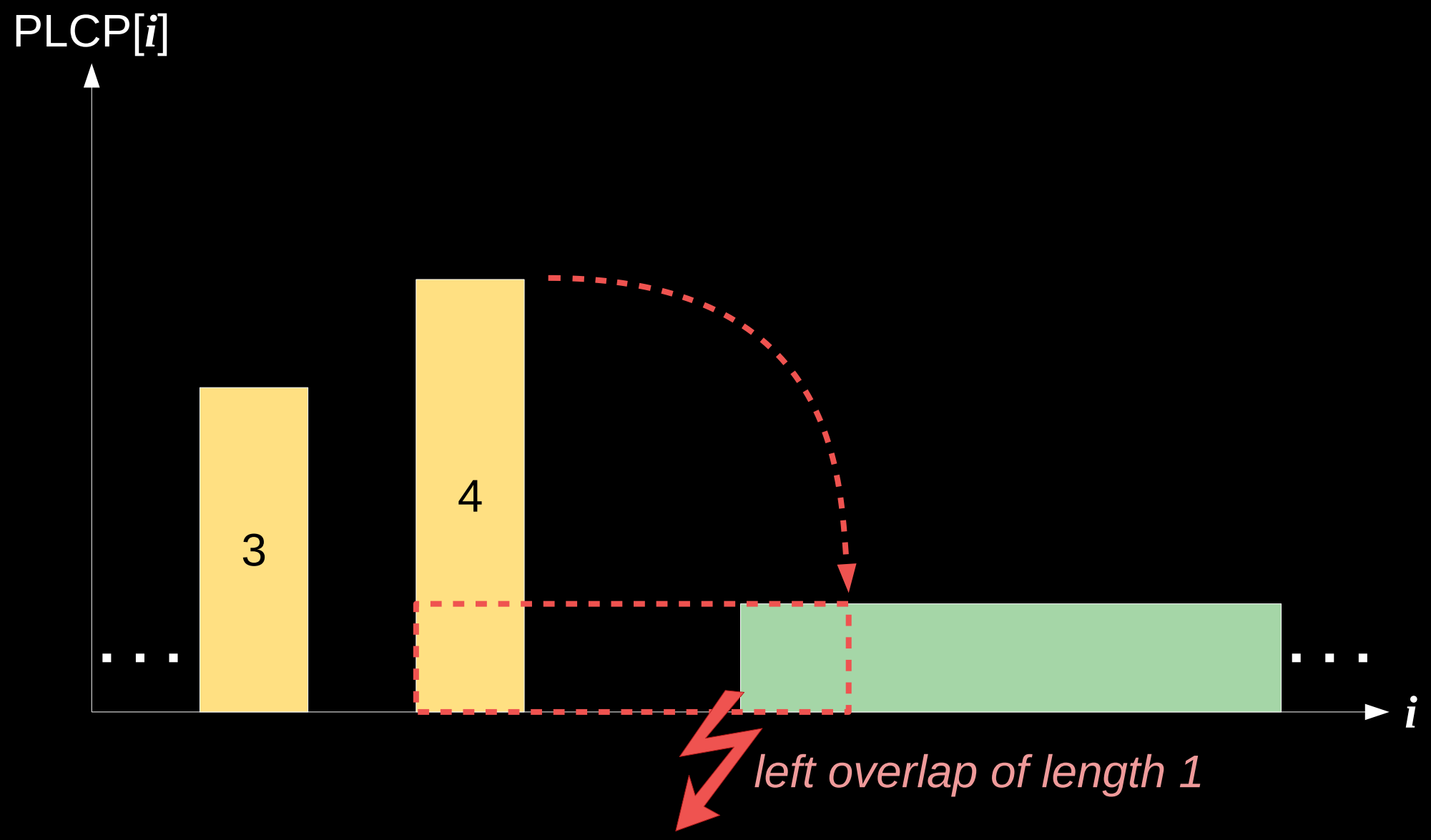
recurse

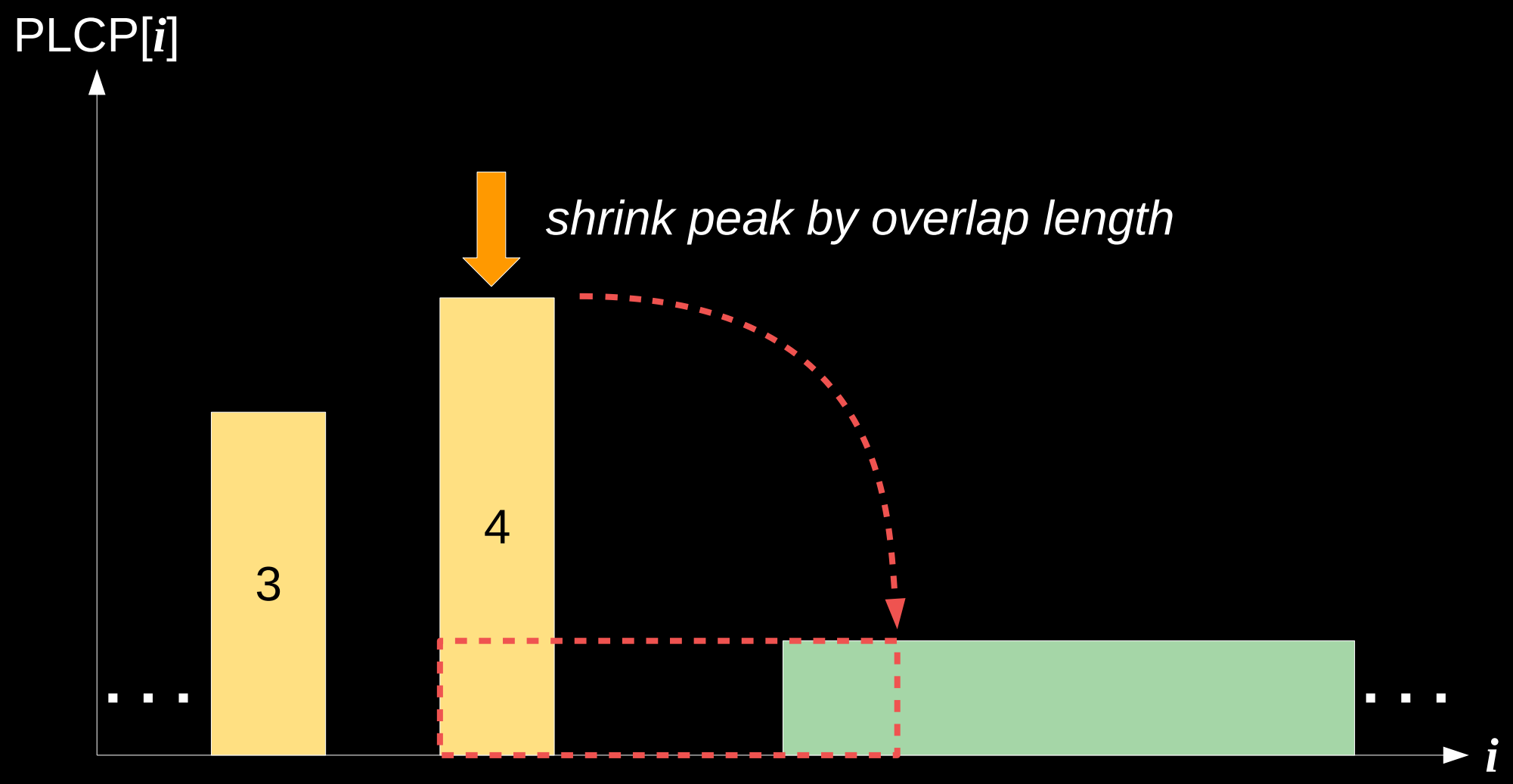

PLCP[i]
3
2
4
3
2
5
4
3
2
1
i
found first maximum peak
apply plcpcomp on this prefix
apply plcpcomp on this suffix
linear time?
only examine interesting peaks!

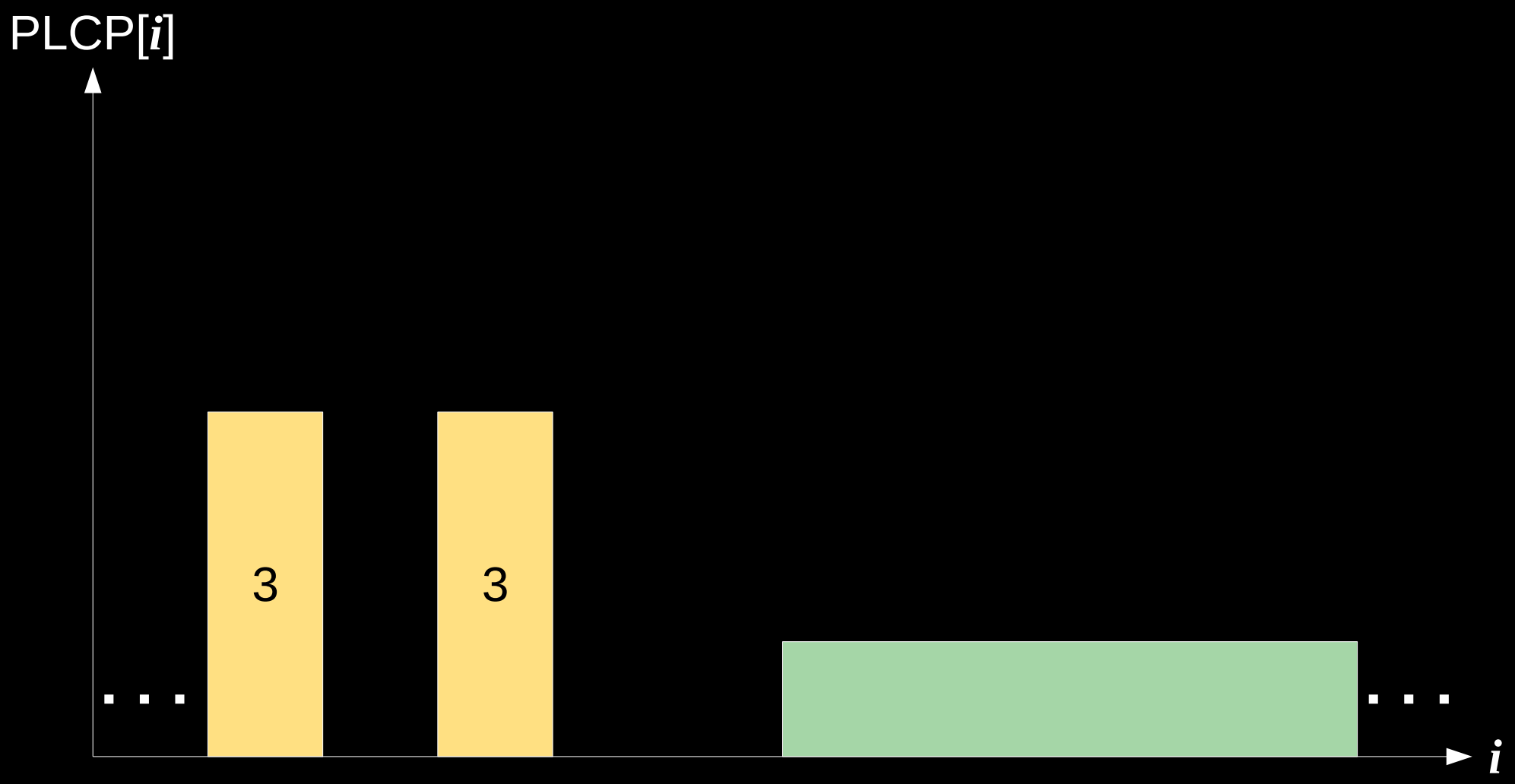
- new maximum peak is interesting peak
- update interesting PLCP of peaks

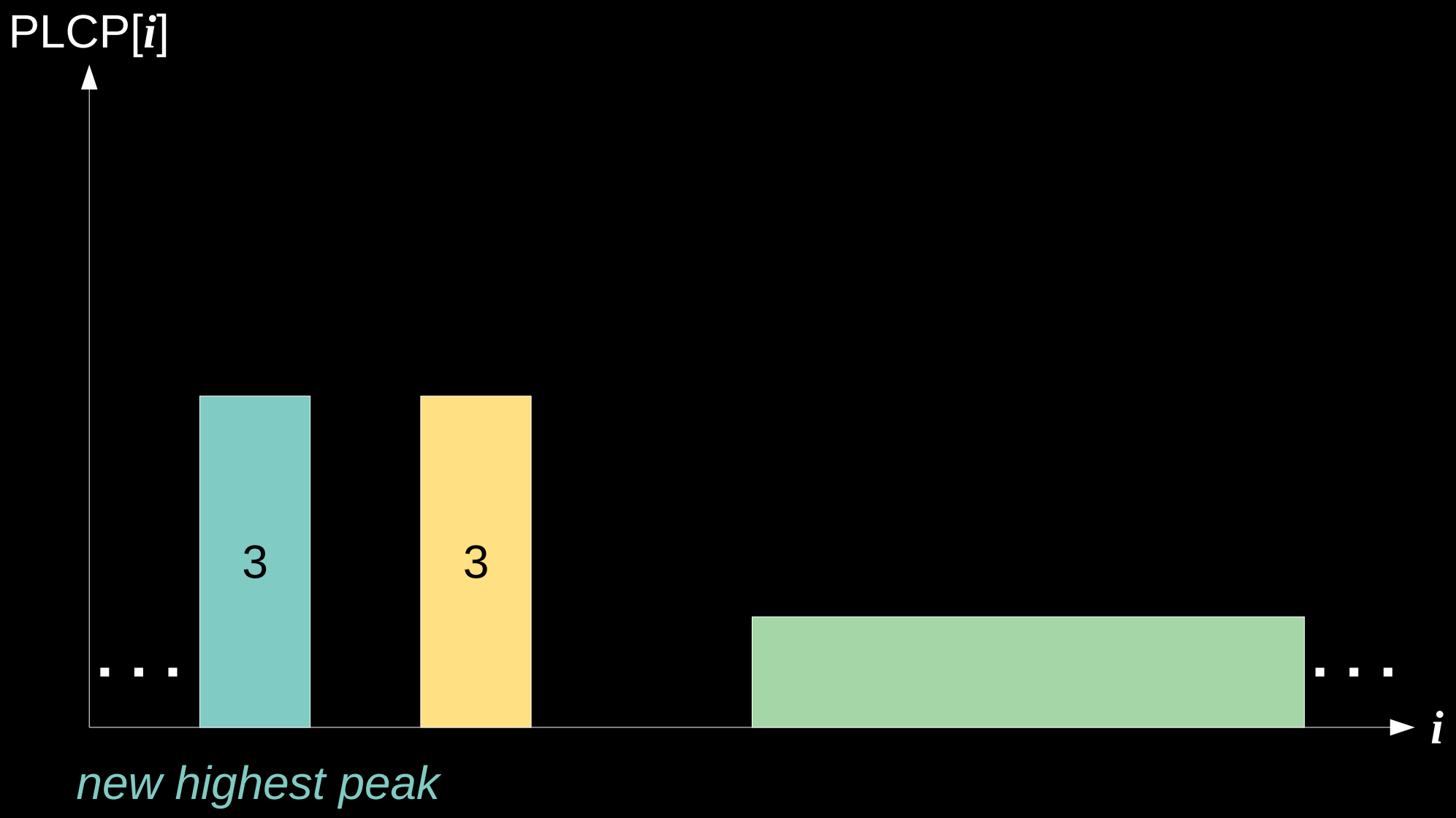


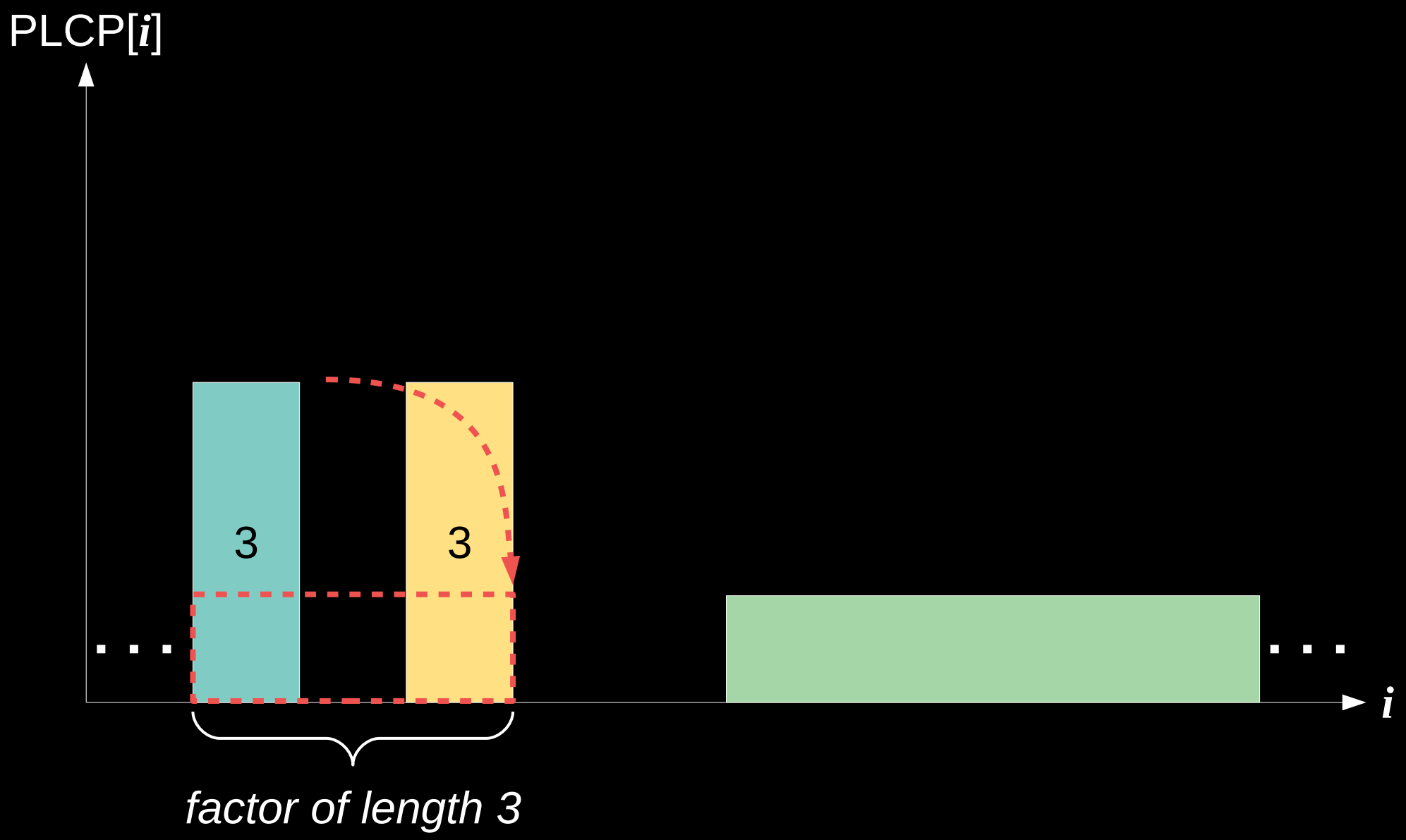

PLCP[i]
3
4
i

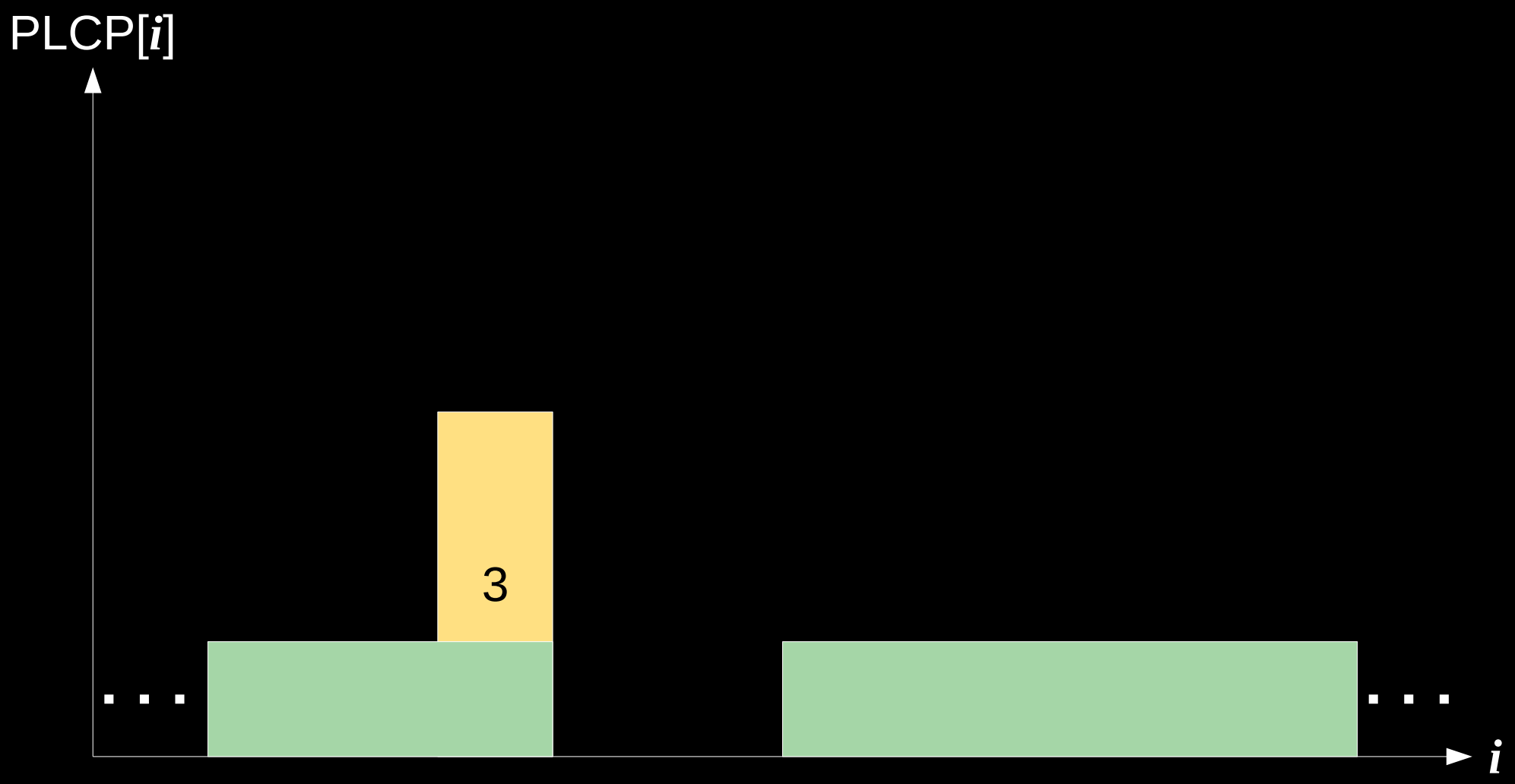

PLCP[i]
3
4
i
left overlap of length 1

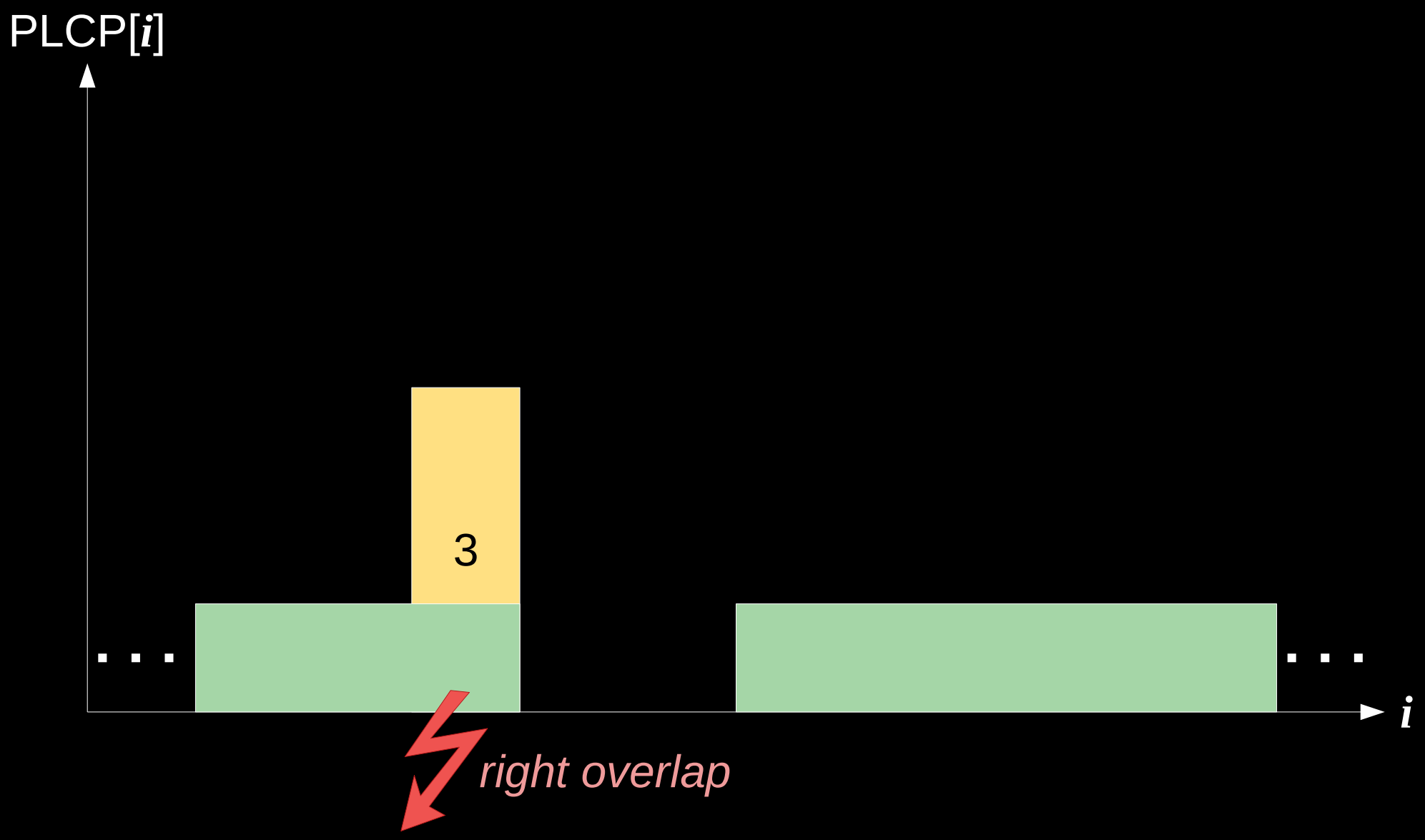

PLCP[*i*]
shrink peak by overlap length
3
4
i

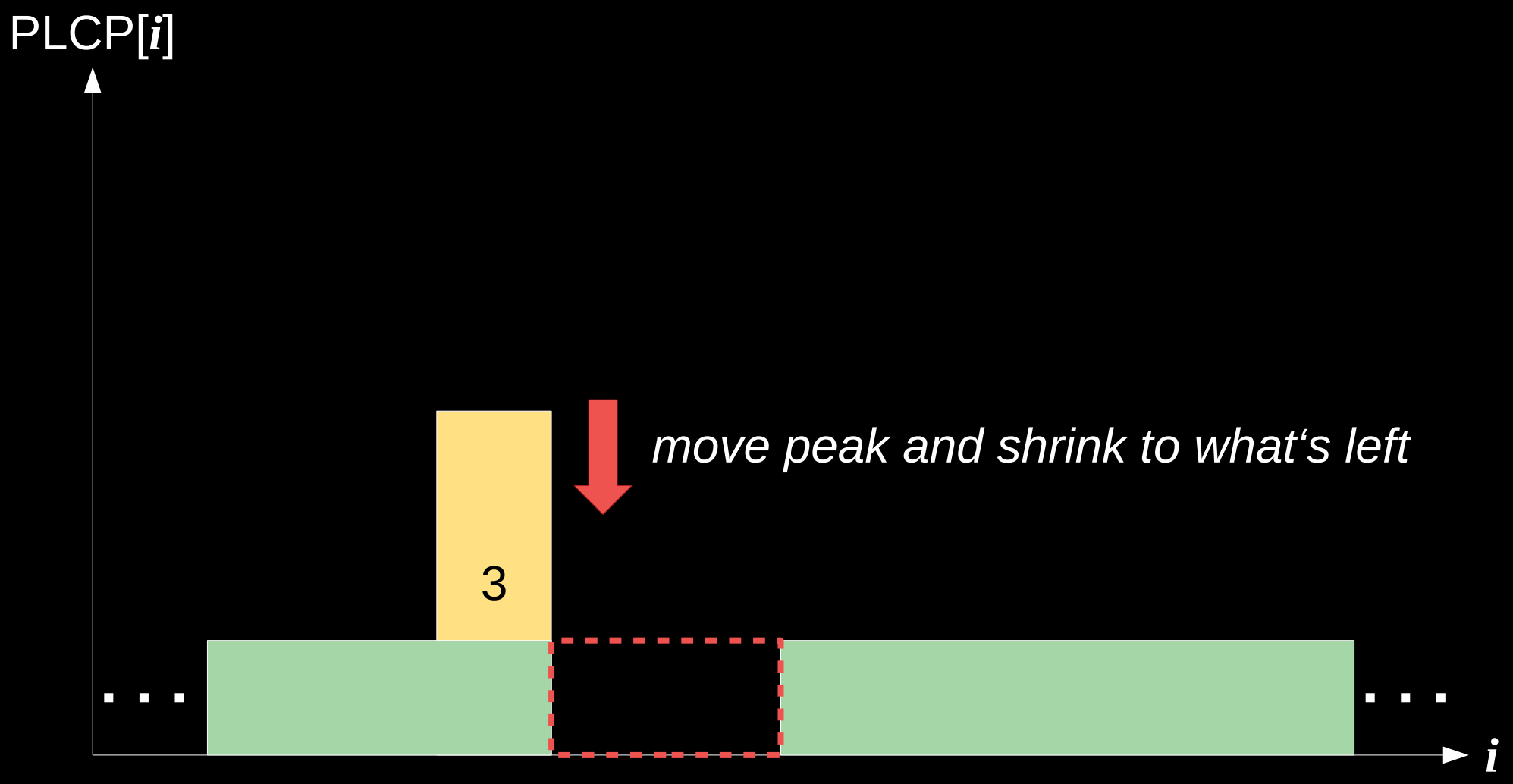

PLCP[i]
3
3
i

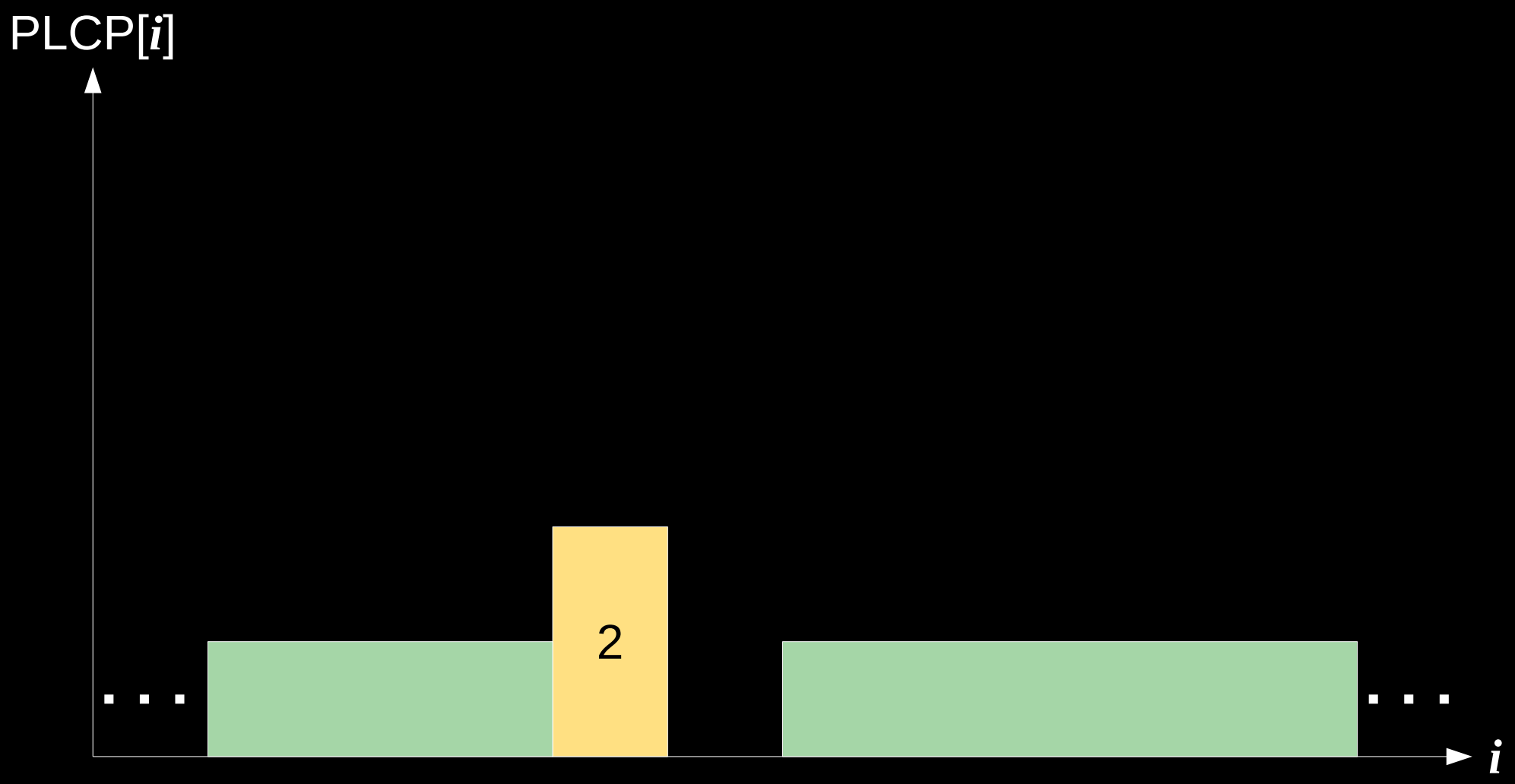

PLCP[*i*]
3
3
i
new highest peak

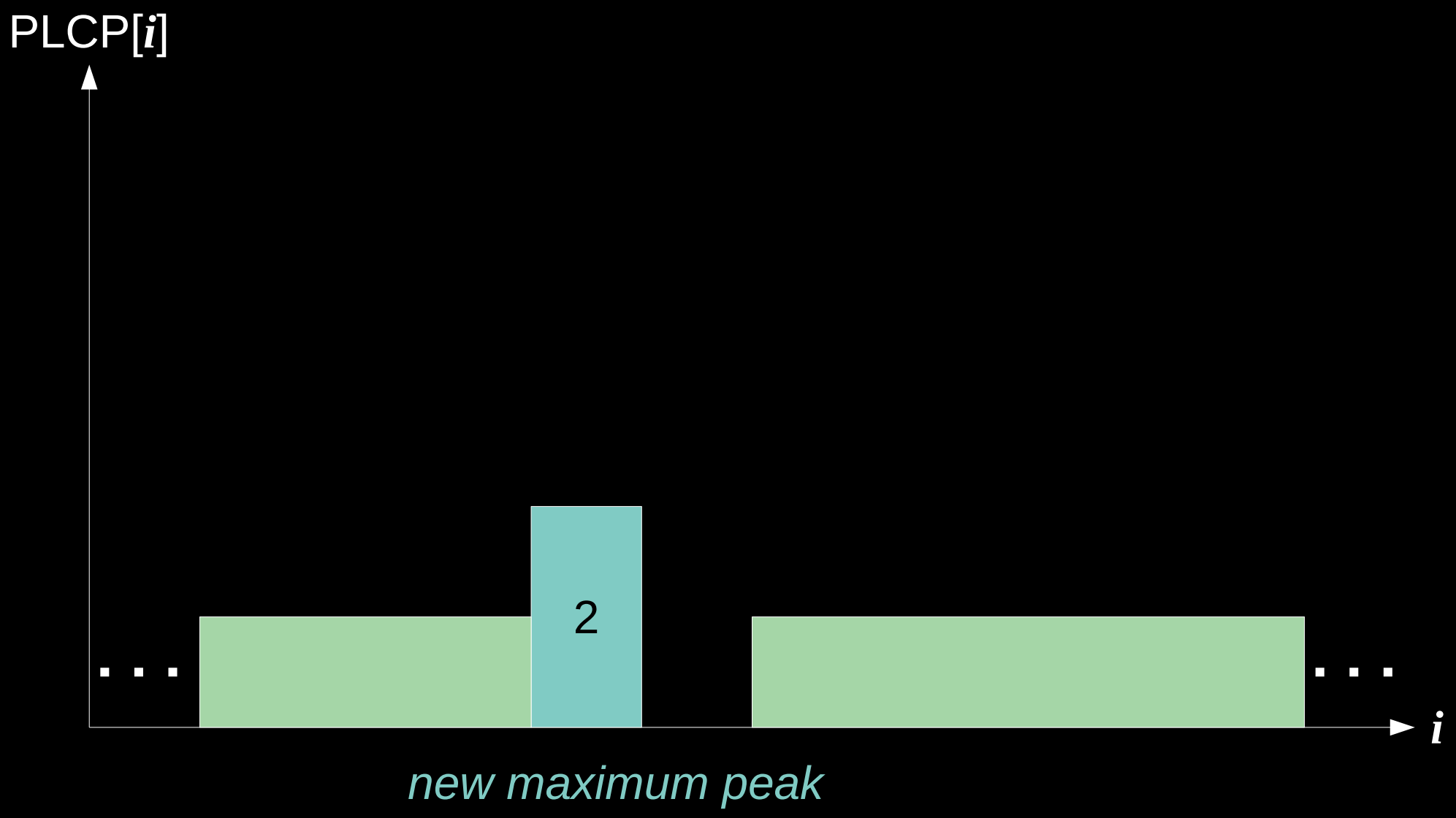

PLCP[*i*]
3
3
i
factor of length 3

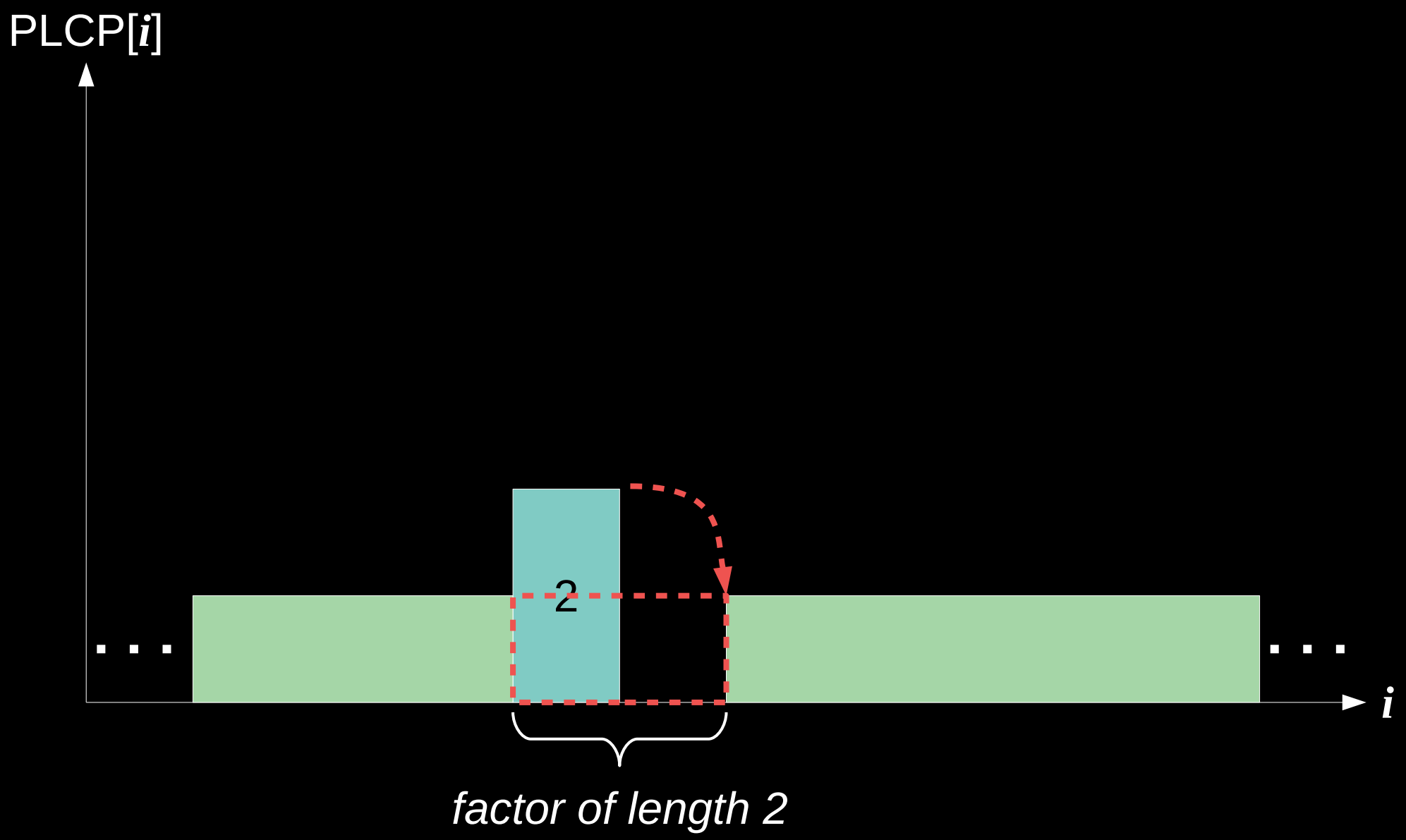

PLCP[*i*]
3
i


PLCP[i]
3
i
right overlap


PLCP[*i*]
move peak and shrink to what‘s left
3
i

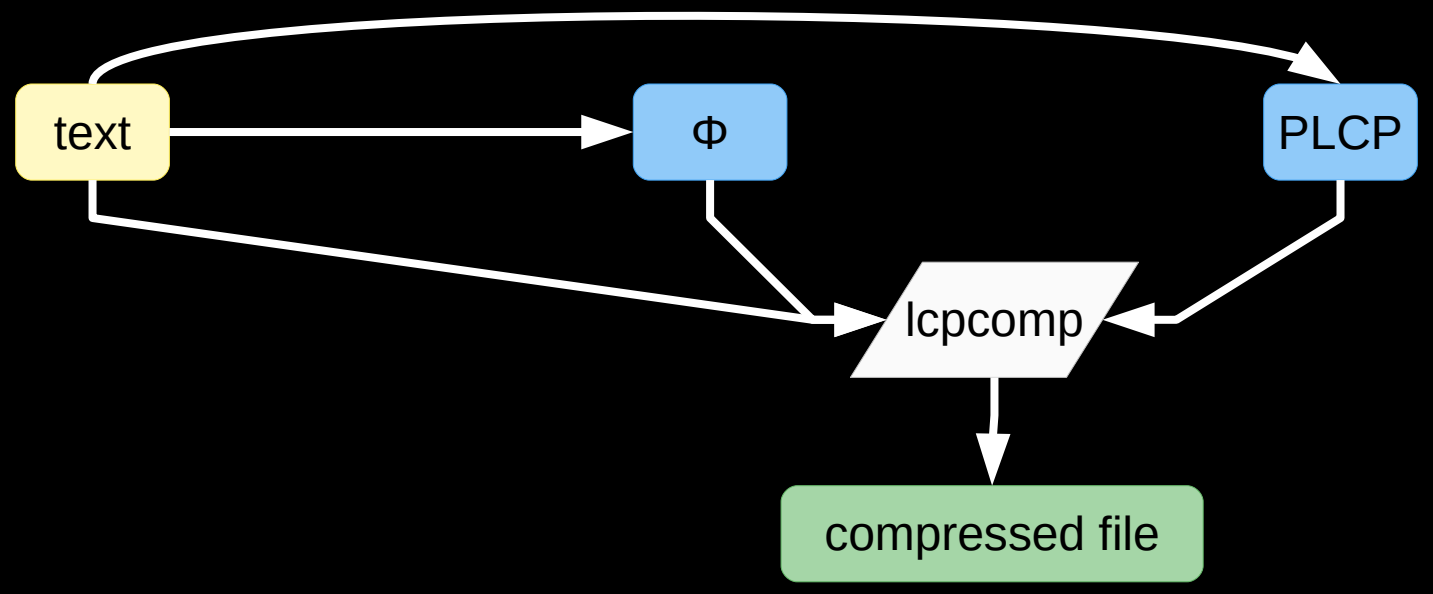

PLCP[*i*]
2
i


PLCP[*i*]
2
i
new maximum peak

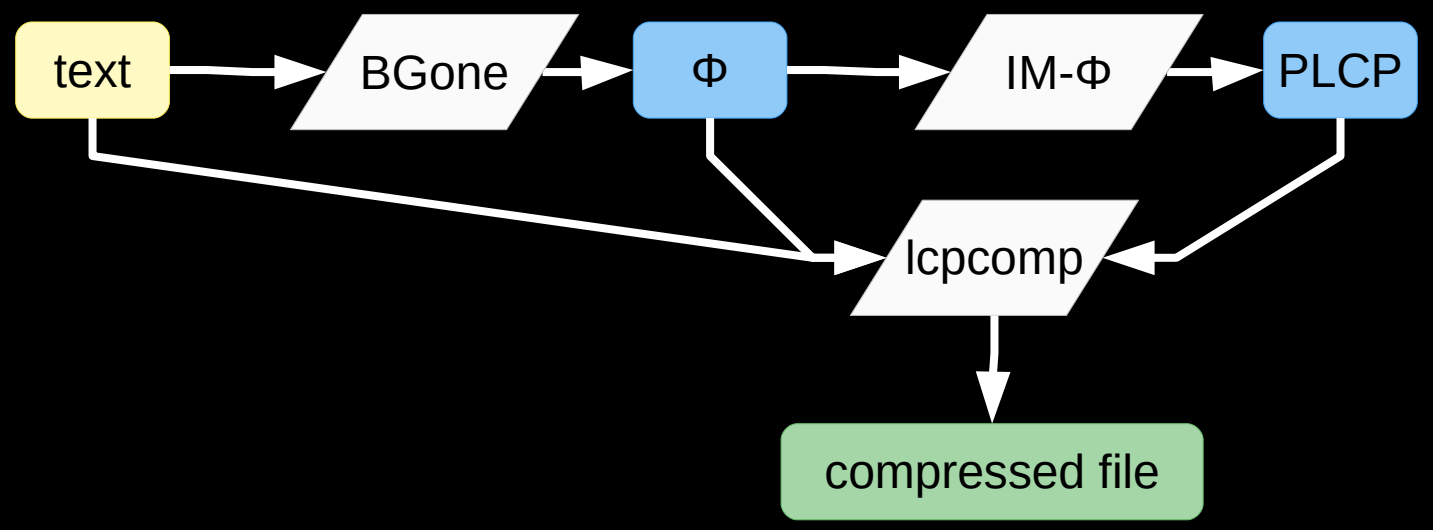

PLCP[*i*]
2
i
factor of length 2


PLCP[*i*]
finished!
i

data structures



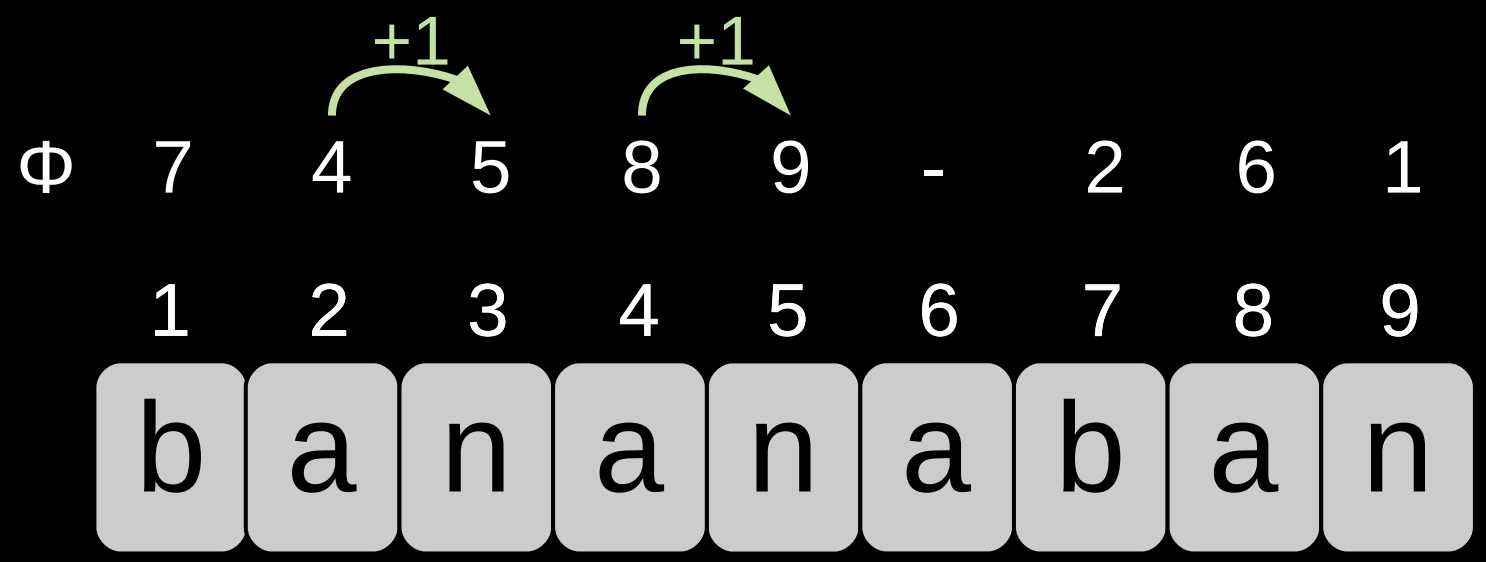
data structures & algorithms



- BGone: 後藤, 坂内 '14
- IM-Φ: Kärkkäinen+'09

O(n) time

sparse Φ/PLCP



sparse Φ/PLCP

|sparse Φ| = #runs in BWT =: *r*

[Kärkkäinen+ '16]

+1 +1

sparse Φ	7	4		8		-	2	6	1
	1	2	3	4	5	6	7	8	9
	b	a	n	a	n	a	b	a	n

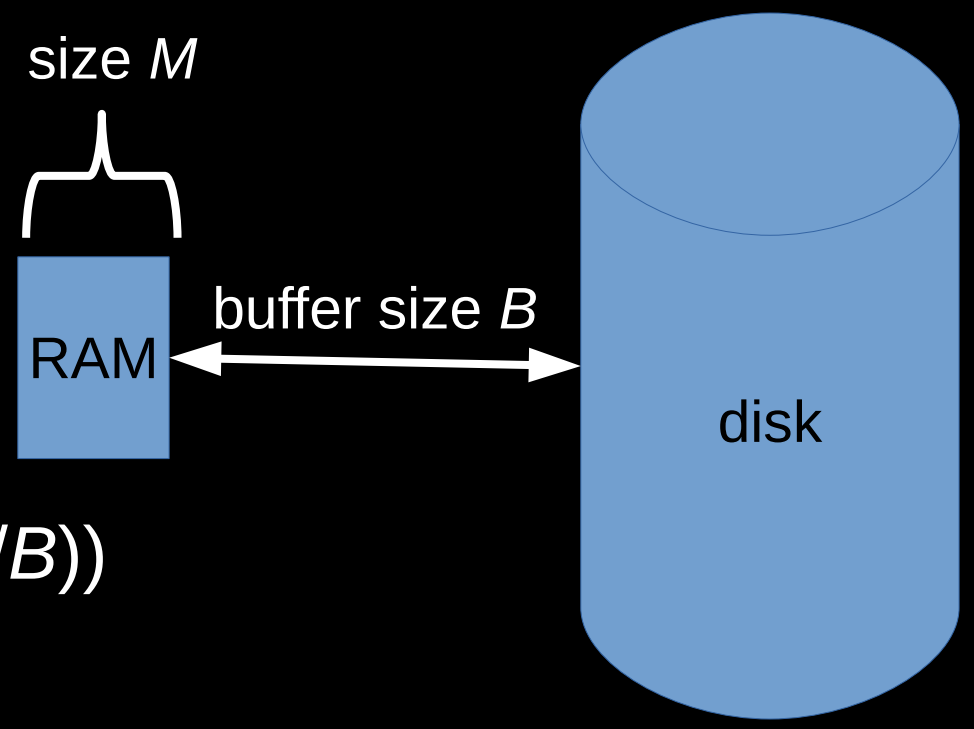
complexity

- time : $O(n)$
- space:

$$\max(\underbrace{r \lg n}_{\text{sparse } \Phi} + \overbrace{3n + o(n)}^{\text{PLCP + bit vector}} + \underbrace{(O(n \lg n) + 1)^{0.5} \lg n}_{\text{maintain interesting peaks}}, \overbrace{n \lg n}^{\text{construct } \Phi})$$

external memory

- # of I/Os
  - scan(n) = $\Theta(n/B)$
  - sort(n) = $\Theta((n/B) \log_{M/B} (n/B))$



execution

given text, PLCP, Φ

- scan PLCP
  - create tuples <j, PLCP[j]>
  - sort
- scan Φ
  - create tuples <j, PLCP[j], Φ[j]>
- scan text
  - create output

3 scan + 1 sort

summary

plcpcomp

- bidirectional compression scheme
- algorithmic improvement to lcpcomp
- linear scan of text, PLCP, Φ
- => works in EM
- **fastest** solution to compress file in EM with ratio ~ LZ77

Outlook

- bounds for #factors?
- bounds for maintaining interesting peaks
- algorithmic optimization
- more variants
  - using Φ^{-1}
  - unidirectional