Funnel Sort

Srinivas Eswar, Georgia Institute of Technology

November 2016

External Sorting

- $T(N; Z, L)$ – Total number of memory transfers
  - N is the problem size
  - Z is the size of the cache
  - L is the cache line size
- $T(N; Z, L) = \theta(\frac{N}{L} \log_{\frac{Z}{L}} \frac{N}{L})$
- K- way Merge Sort
  - Fit one block from each of the sorted lists in Cache
  - $T(N; Z, L) = K \times T\left(\frac{N}{K}; Z, L\right) + O(\frac{N}{L})$
  - $K = \frac{Z}{L}$

Cache Oblivious Sorting

- Cache parameters are unknown
- Divide and conquer!
- Require a special data structure
  - Recursively merges different number of lists
  - Dependent only on the size of the input
  - K – Funnels!

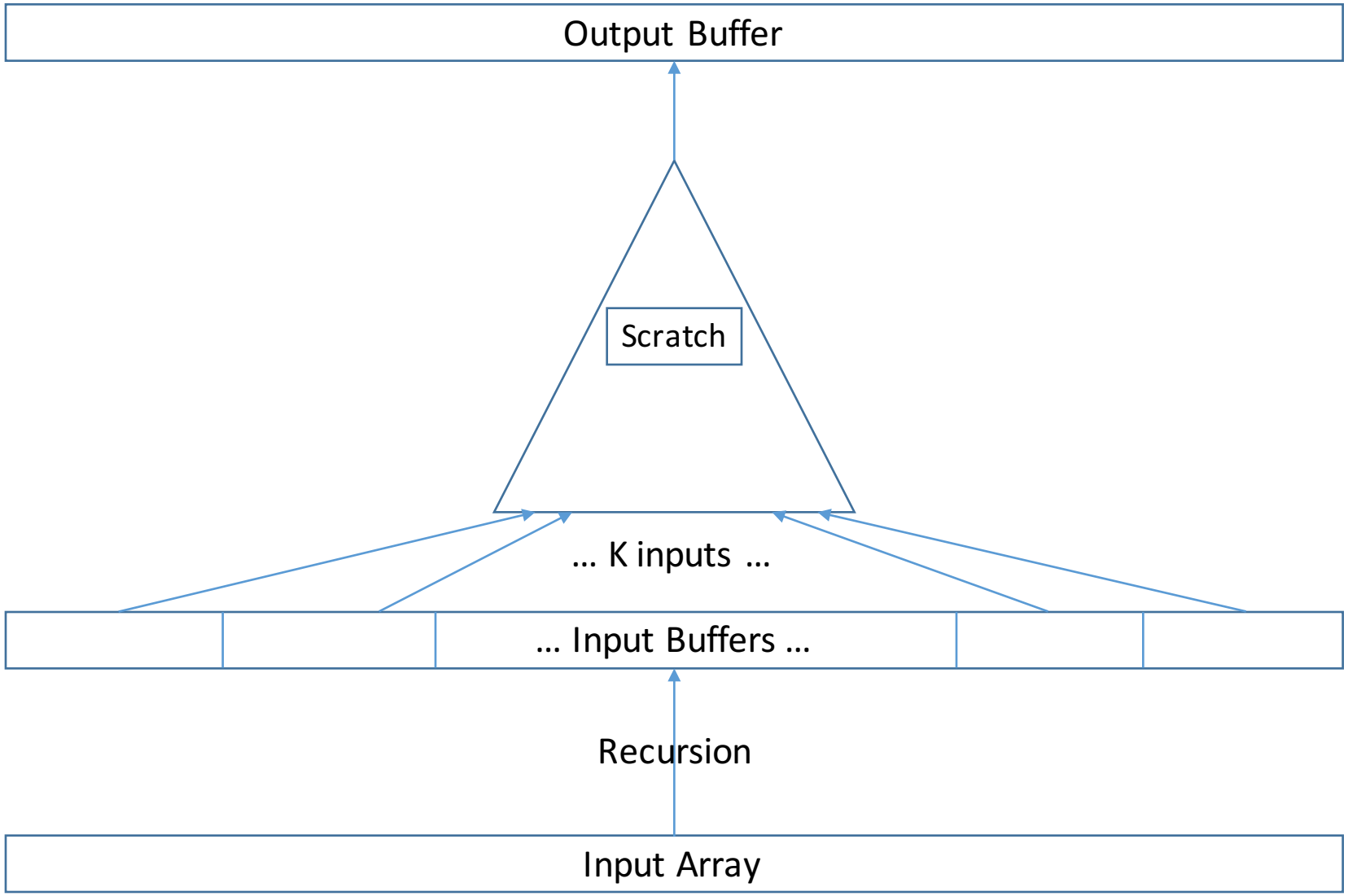
Funnels

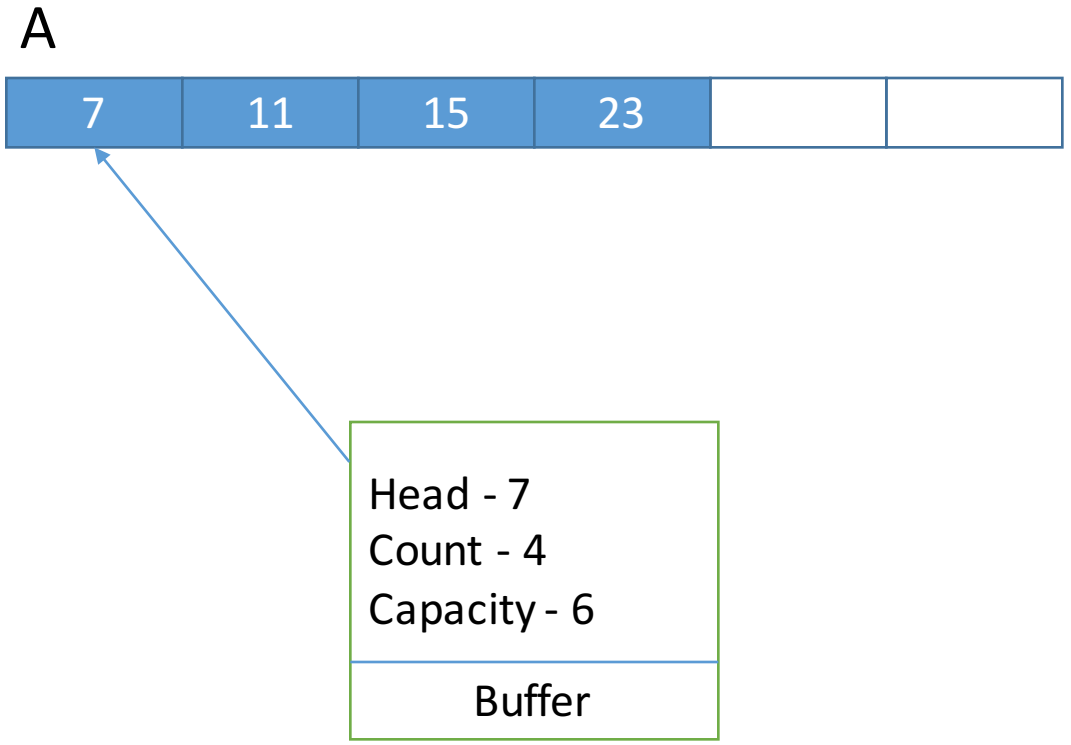
- K − Funnel is a recursive data structure
  - K sorted input lists represented as a “buffer”
  - 1 output buffer
  - Additional scratch space

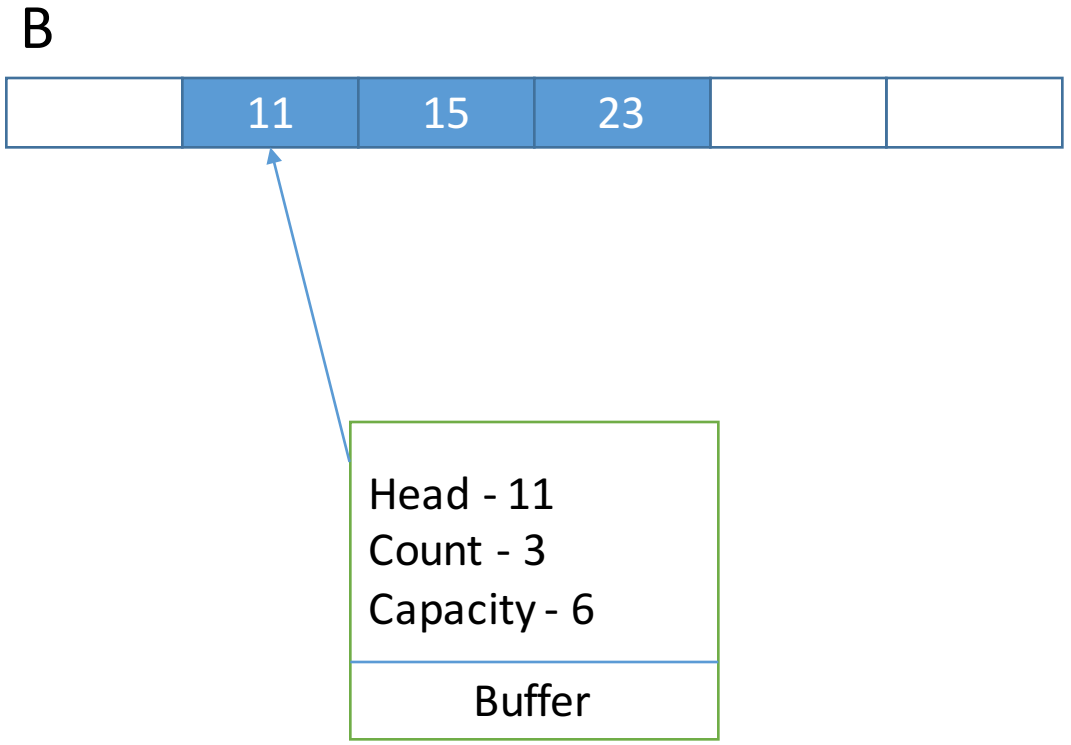
- Buffer
  - Always know the head of the list
  - Can check if it is empty in constant time
  - Circular queue

Funnelsort

- Funnelsort algorithm
  - Split the array into $K = N^{\frac{1}{3}}$ contiguous segments of size $N/K = N^{\frac{2}{3}}$.
  - Recursively sort each segment.
  - Apply the $K-$ Funnel to merge the sorted segments.

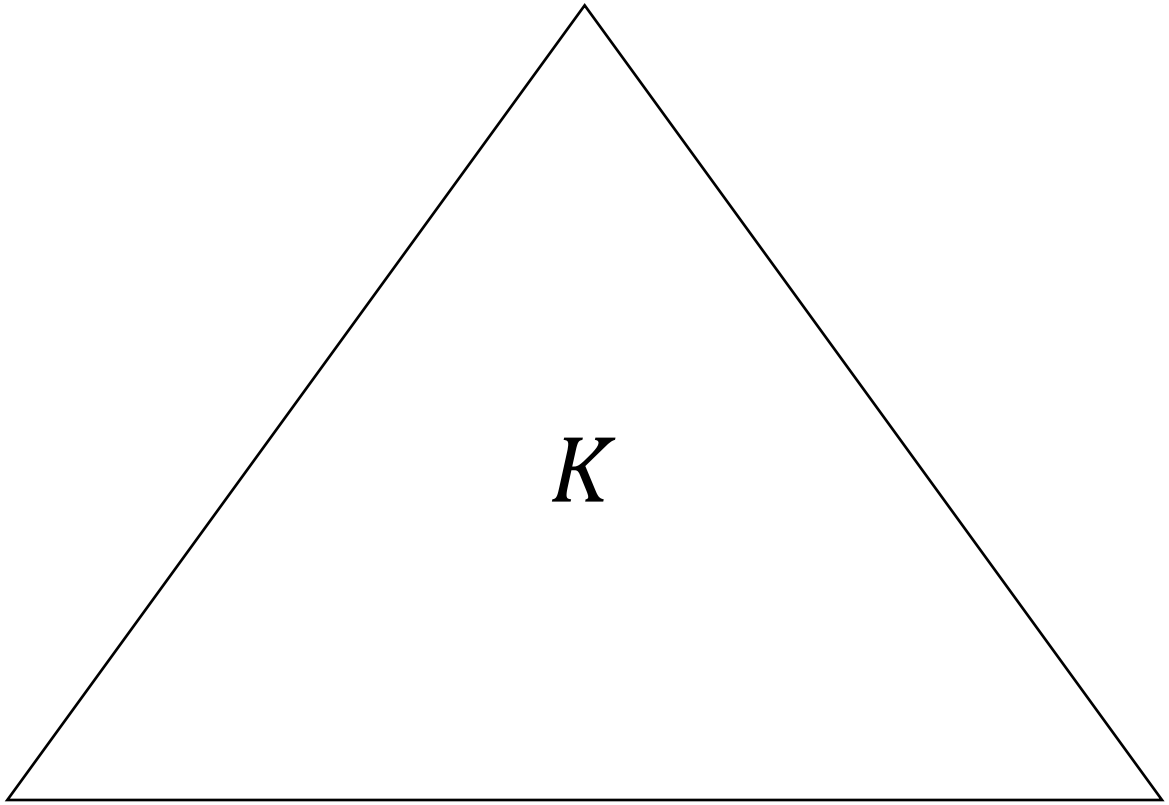

Output Buffer
Scratch
... K inputs ...
... Input Buffers ...
Recursion
Input Array

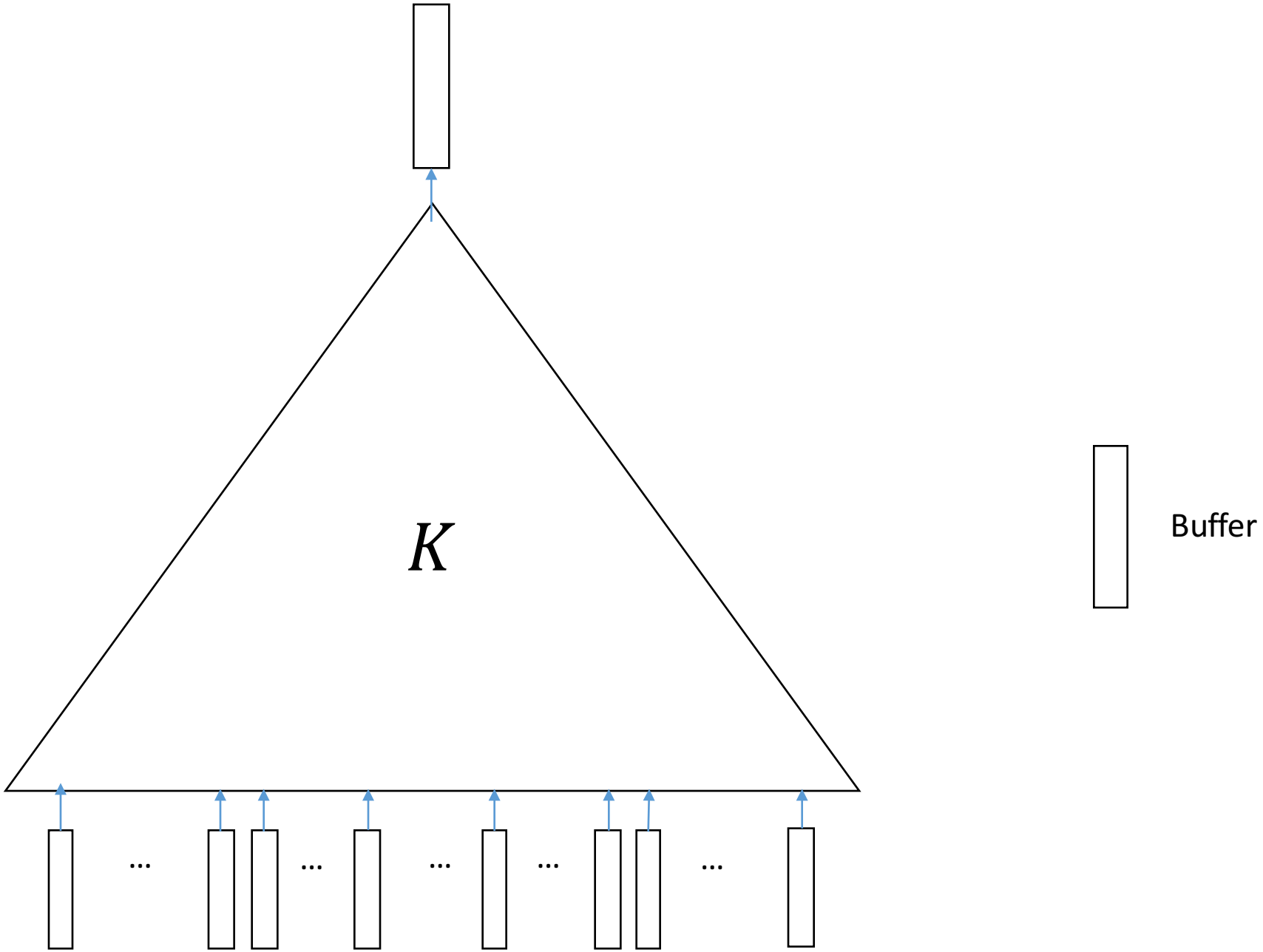

A
7
11
15
23
Head - 7
Count - 4
Capacity - 6
Buffer

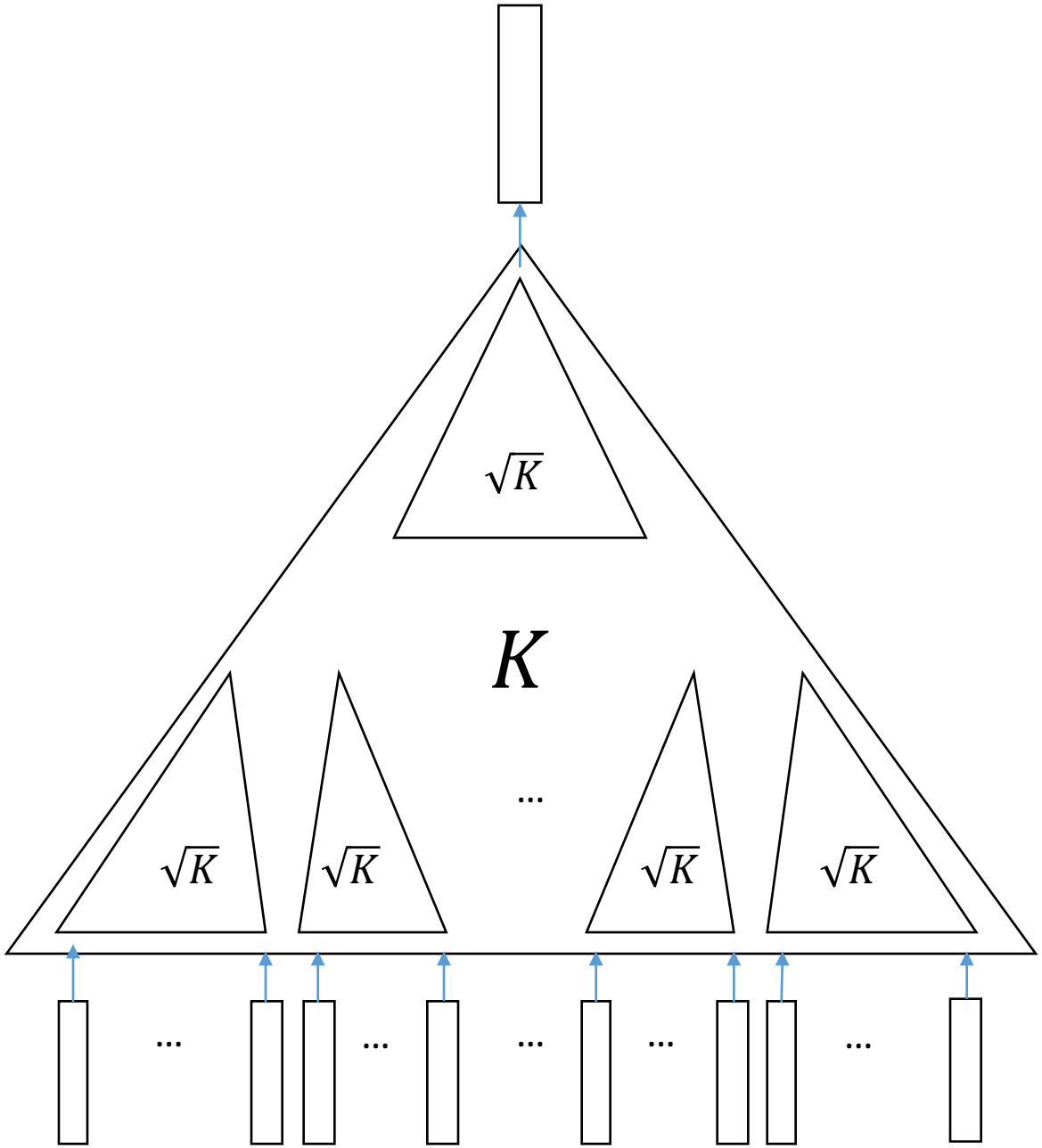

B
11
15
23
Head - 11
Count - 3
Capacity - 6
Buffer

K − Funnel details

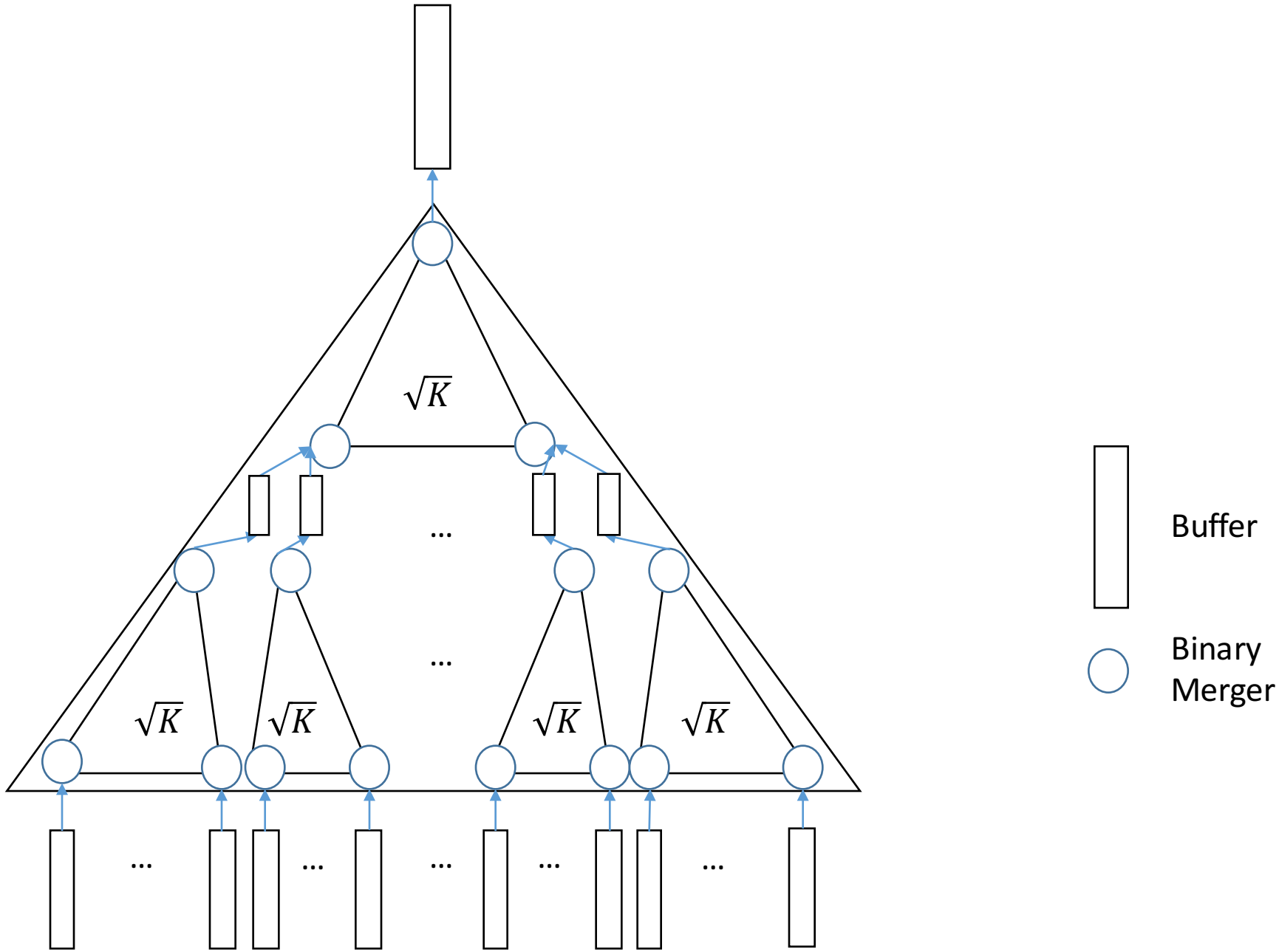
- K − Funnel is a recursive data structure
  - Recursively split into 1 top $\sqrt{K}$ − Funnel and $\sqrt{K}$ bottom $\sqrt{K}$ − Funnels.
  - Output buffers of the bottom Funnels serve as input buffers to the Top Funnel.
  - K is of the form 2^{2^x}.
  - Can be viewed as a complete binary tree with nodes as Funnels and edges as buffers.
  - Leaf/Smallest Funnels are 2 – Funnels aka Binary Mergers


K


K
Buffer



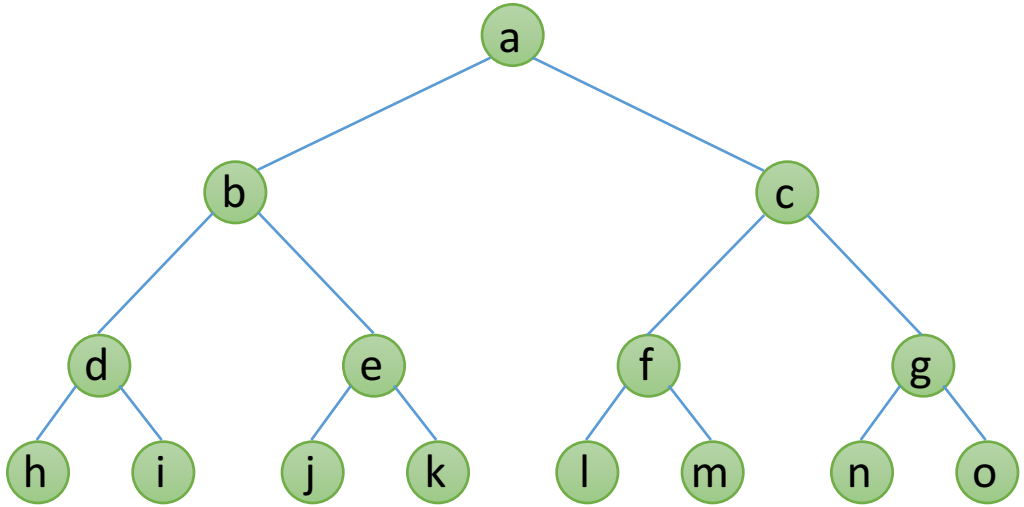
Buffer

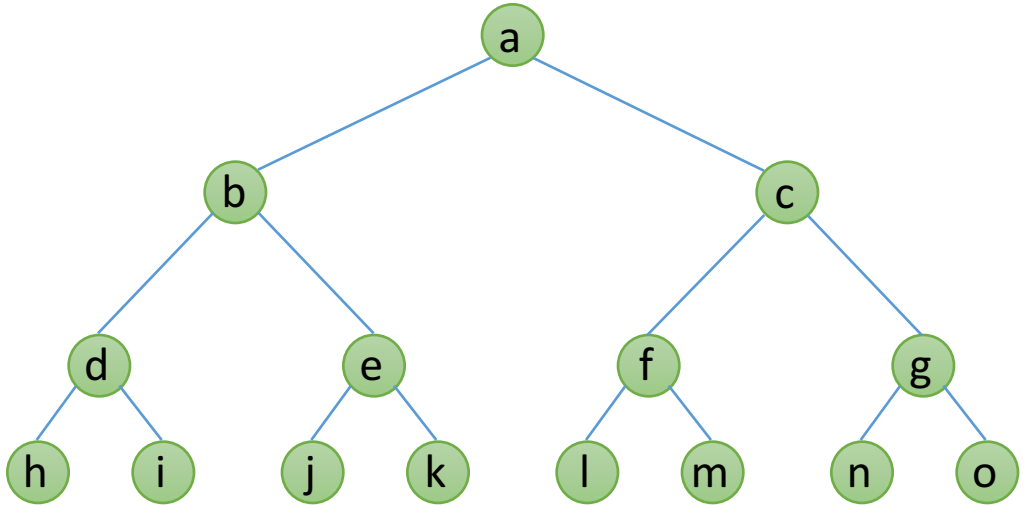

$\sqrt{K}$
$\sqrt{K}$
$\sqrt{K}$
$\sqrt{K}$
$\sqrt{K}$
Buffer
Binary
Merger

K — Funnel Buffer Layout

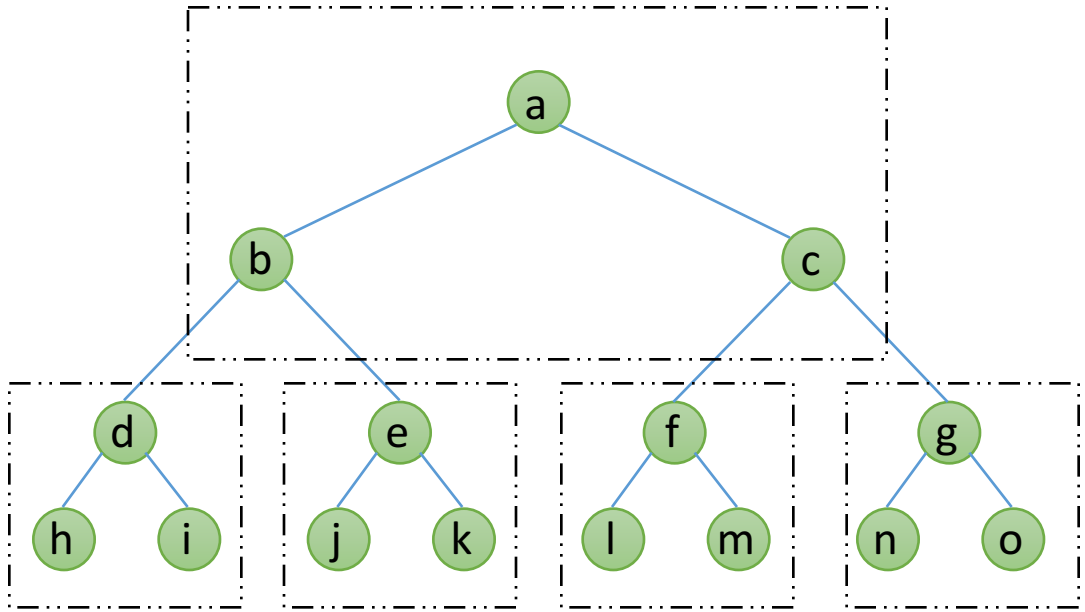
- K — Funnel are essentially a bunch of buffers.
  - Capacity of the output buffer $\geq K^3$.
  - Capacity of the input buffers $\sim K^2$ each.
  - Capacity of the intermediate buffers $K^{\frac{3}{2}}$ each.

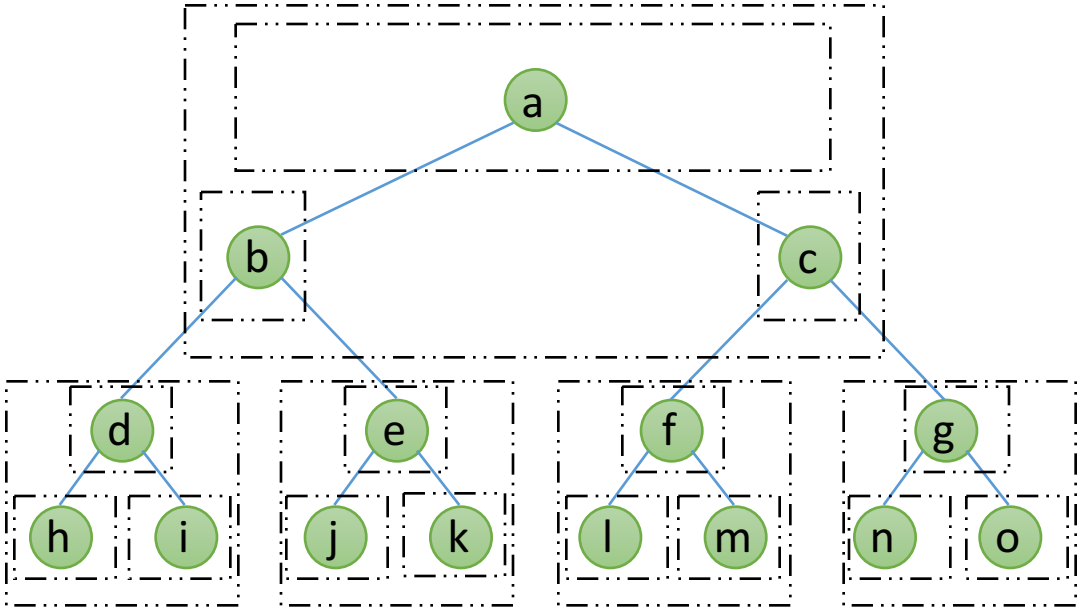
- Van Emde Boas layout – “Splitting the middle level of a complete binary tree”.

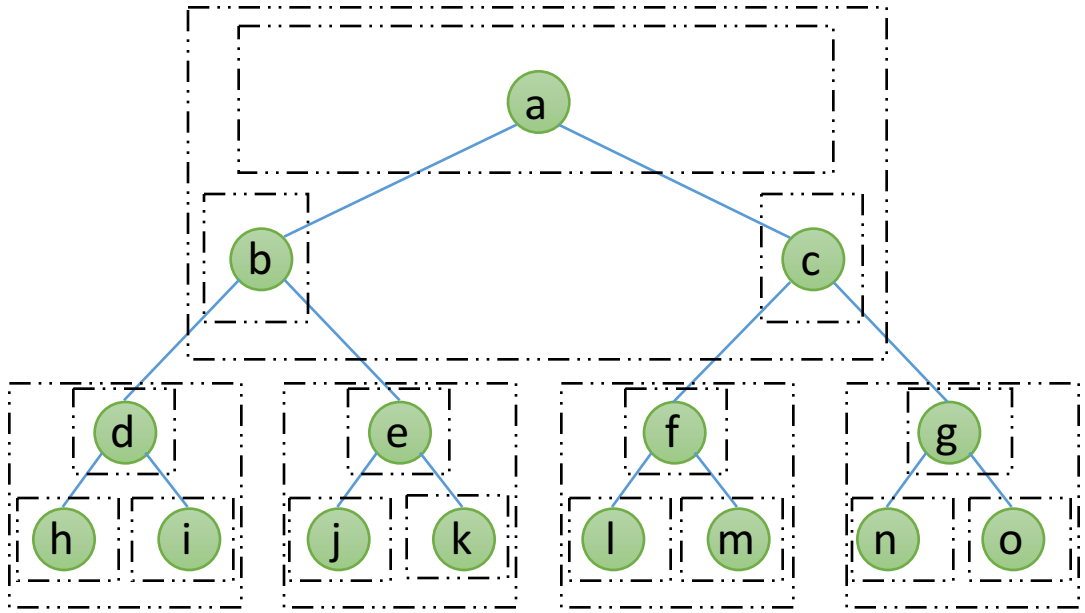

a
b
c
d
e
f
g
h
i
j
k
l
m
n
o



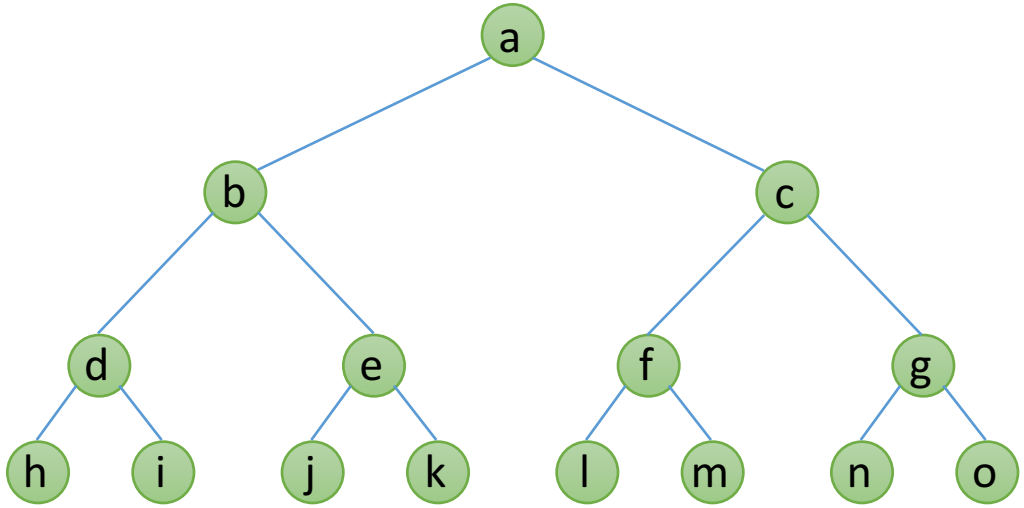
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o


a
b
c
d
e
f
g
h
i
j
k
l
m
n
o


a
b
c
d
e
f
g
h
i
j
k
l
m
n
o

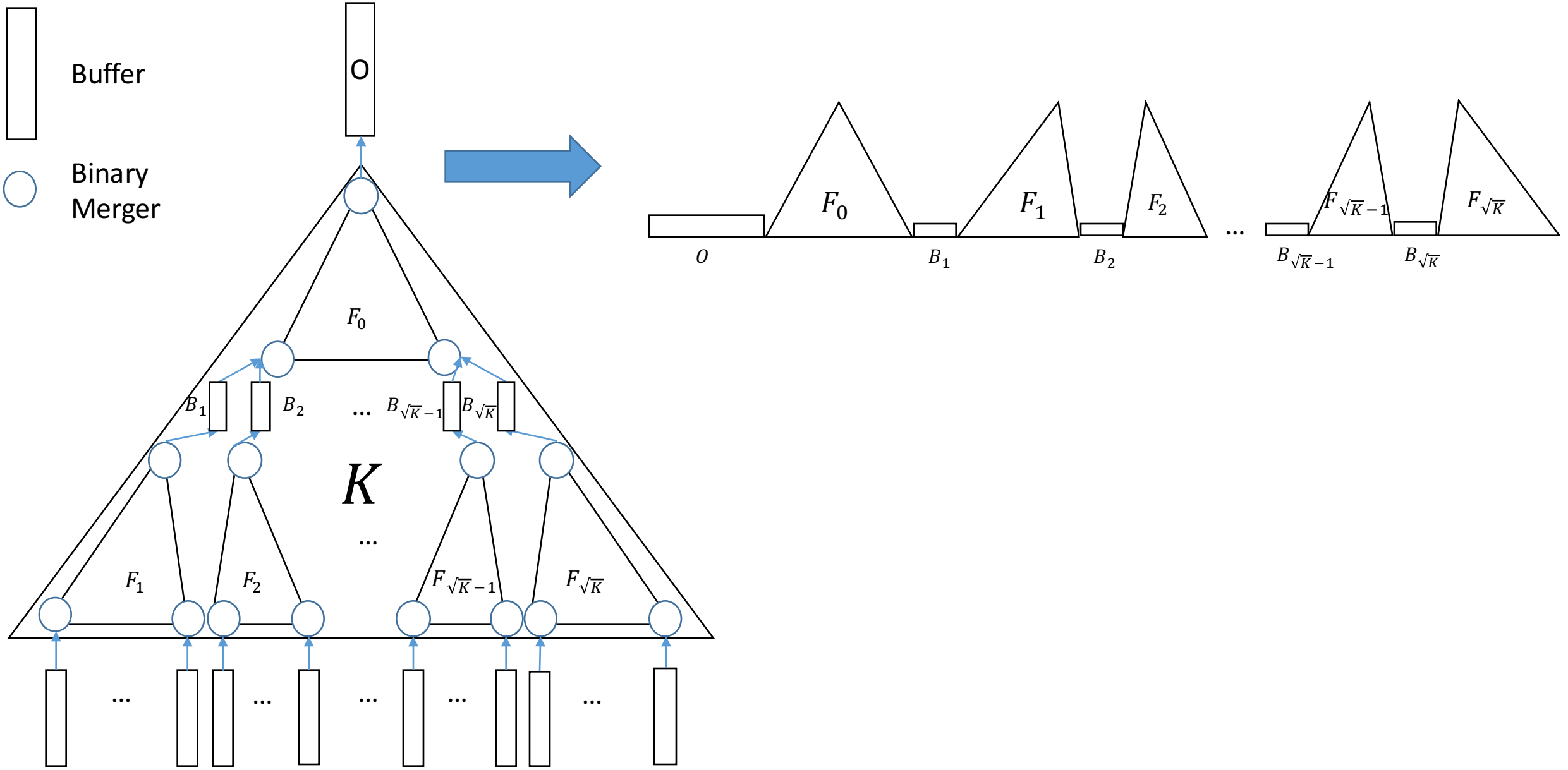


a	b	c	d	h	i	e	j	k	f	l	m	g	n	o



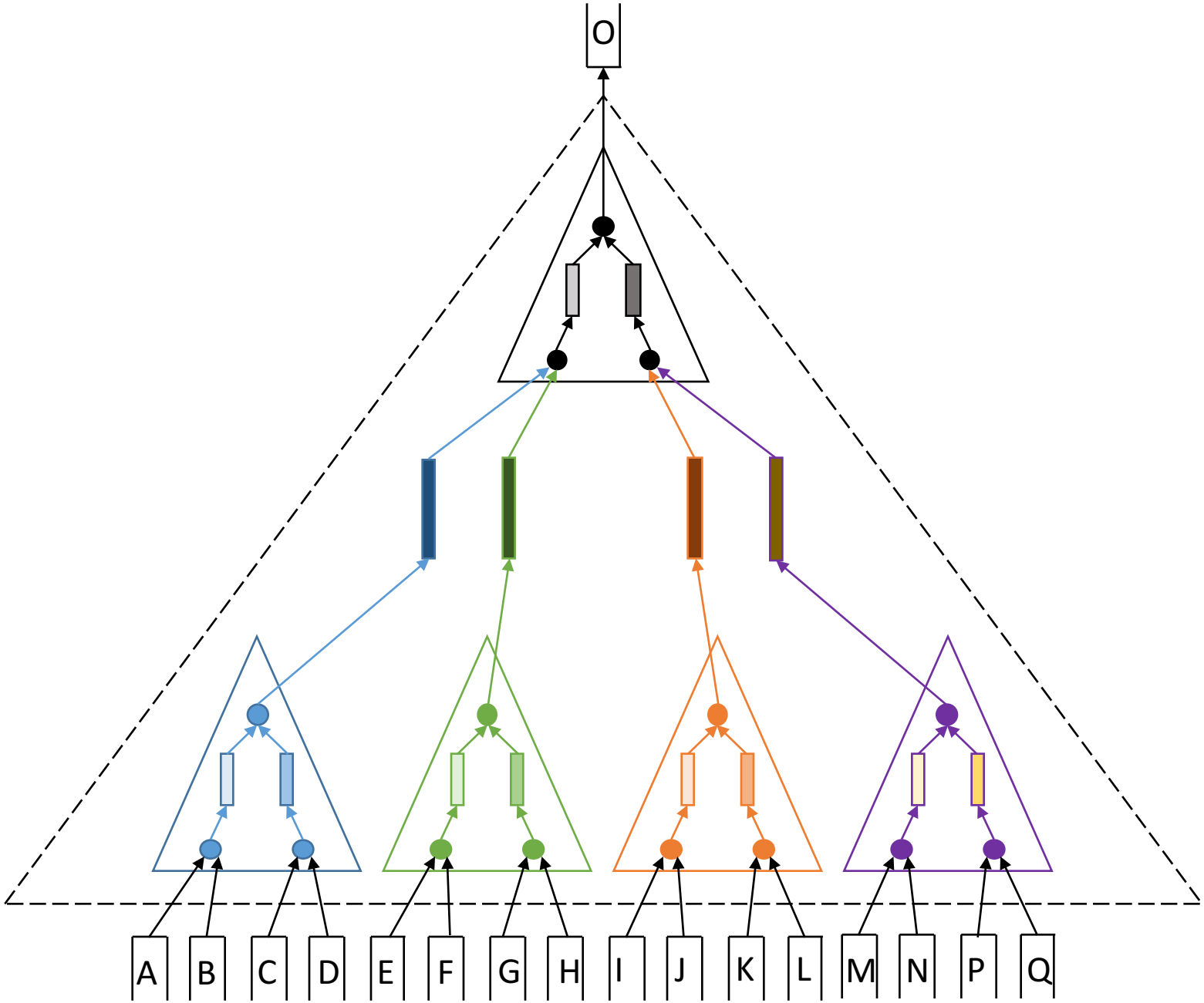
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o

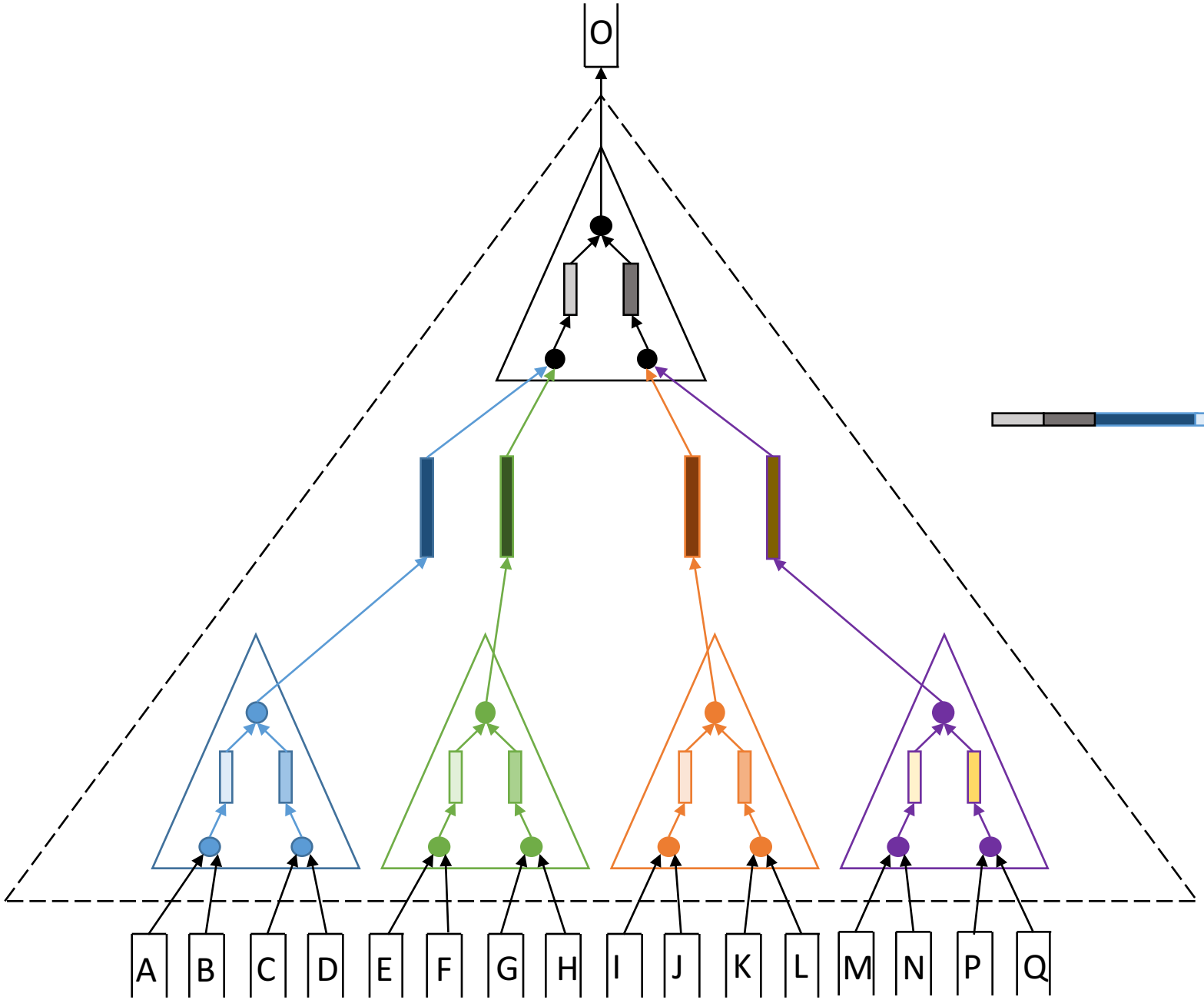
a	b	c	d	h	i	e	j	k	f	l	m	g	n	o


Buffer
Binary Merger
O
F_0
B_1
B_2
...
$B_{\sqrt{K}-1}$
$B_{\sqrt{K}}$
K
F_1
F_2
$F_{\sqrt{K}-1}$
$F_{\sqrt{K}}$
O

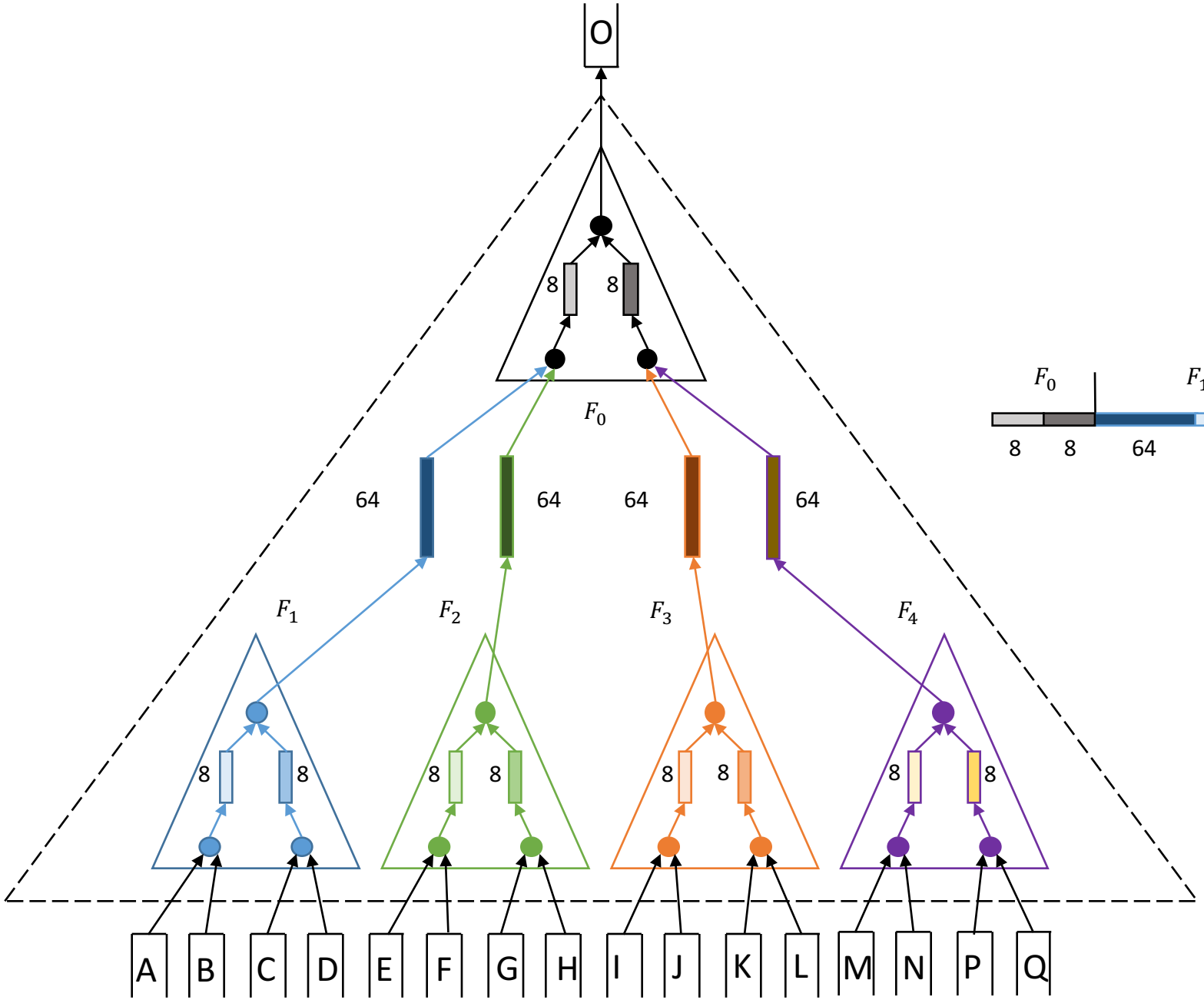
16 – Funnel

- 16 Input Buffers
- Merges 4096 elements
- Recursively split into 4 –Funnels and 2 –Funnels

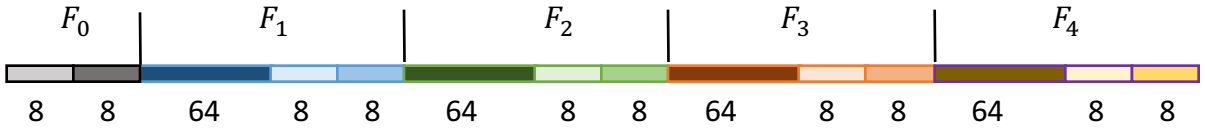

O
A
B
C
D
E
F
G
H
I
J
K
L
M
N
P
Q

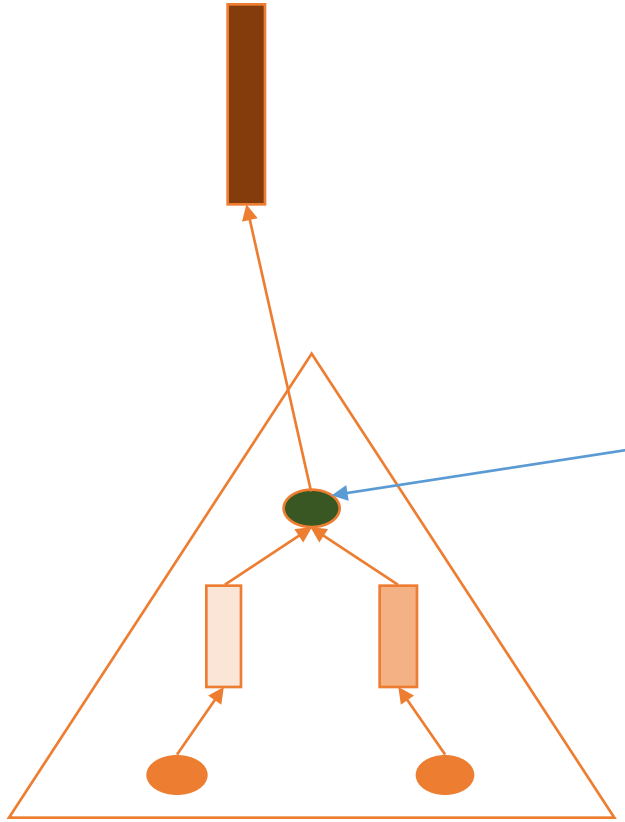


Memory Layout

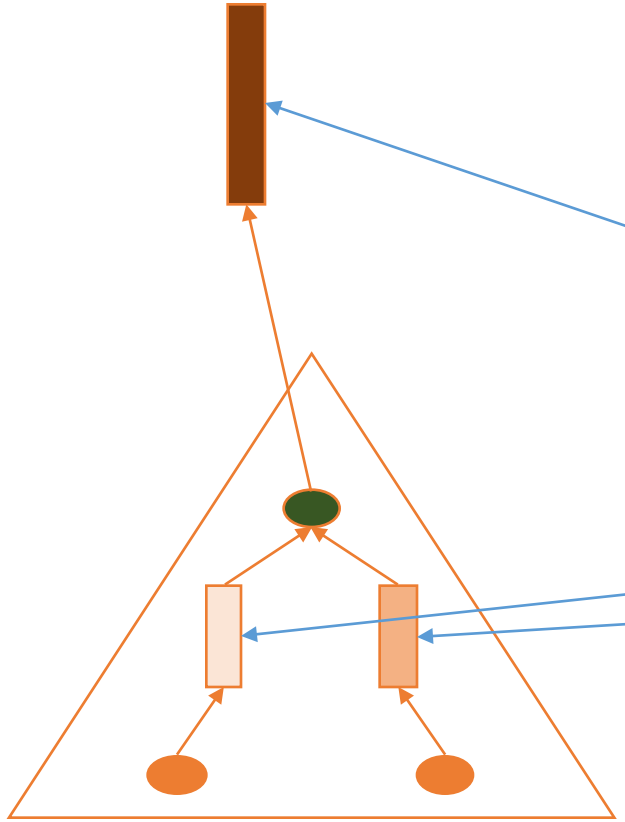


Memory Layout

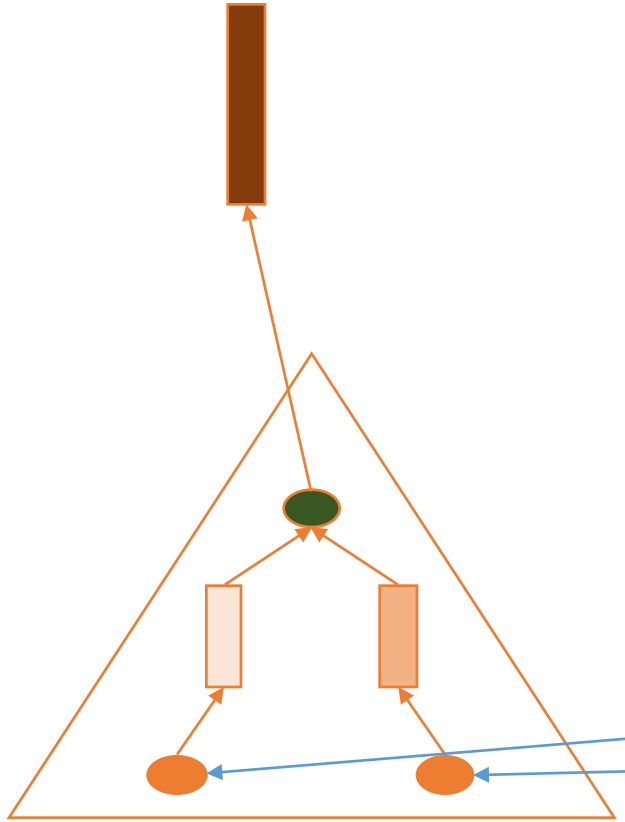




Binary Merger
Output Buffer
Left Input Buffer
Right Input Buffer
Left Child
Right Child



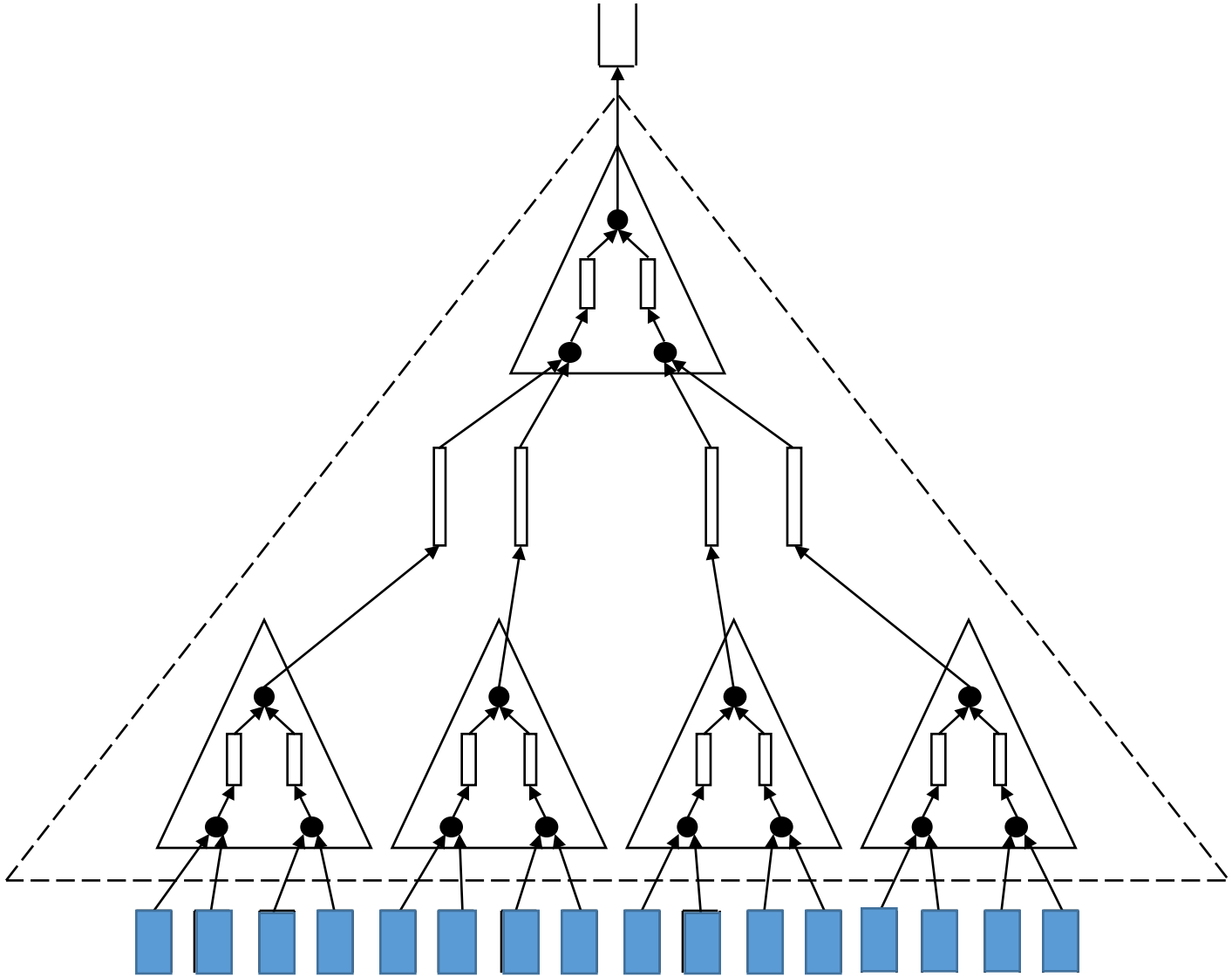
Binary Merger
Output Buffer
Left Input Buffer
Right Input Buffer
Left Child
Right Child

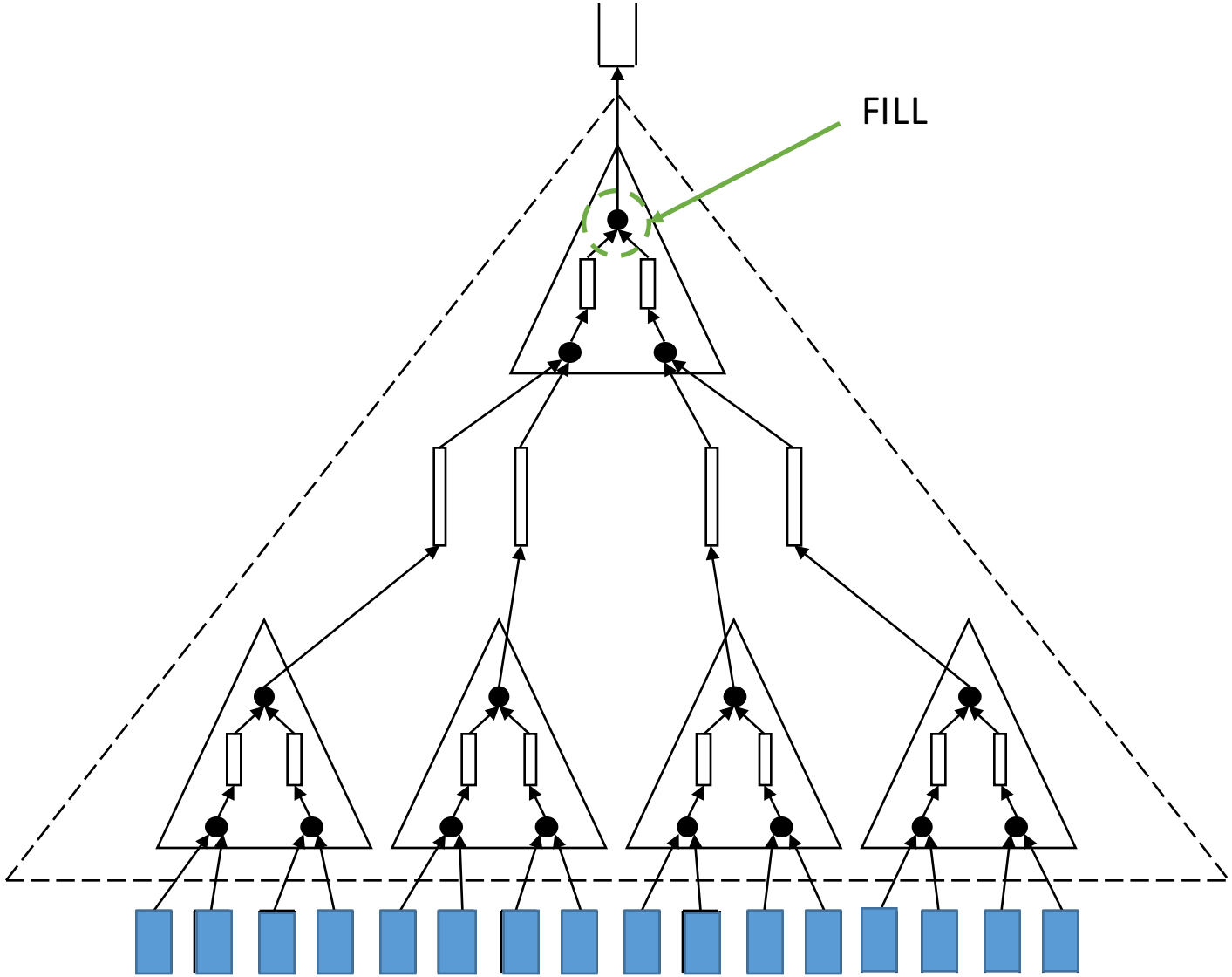


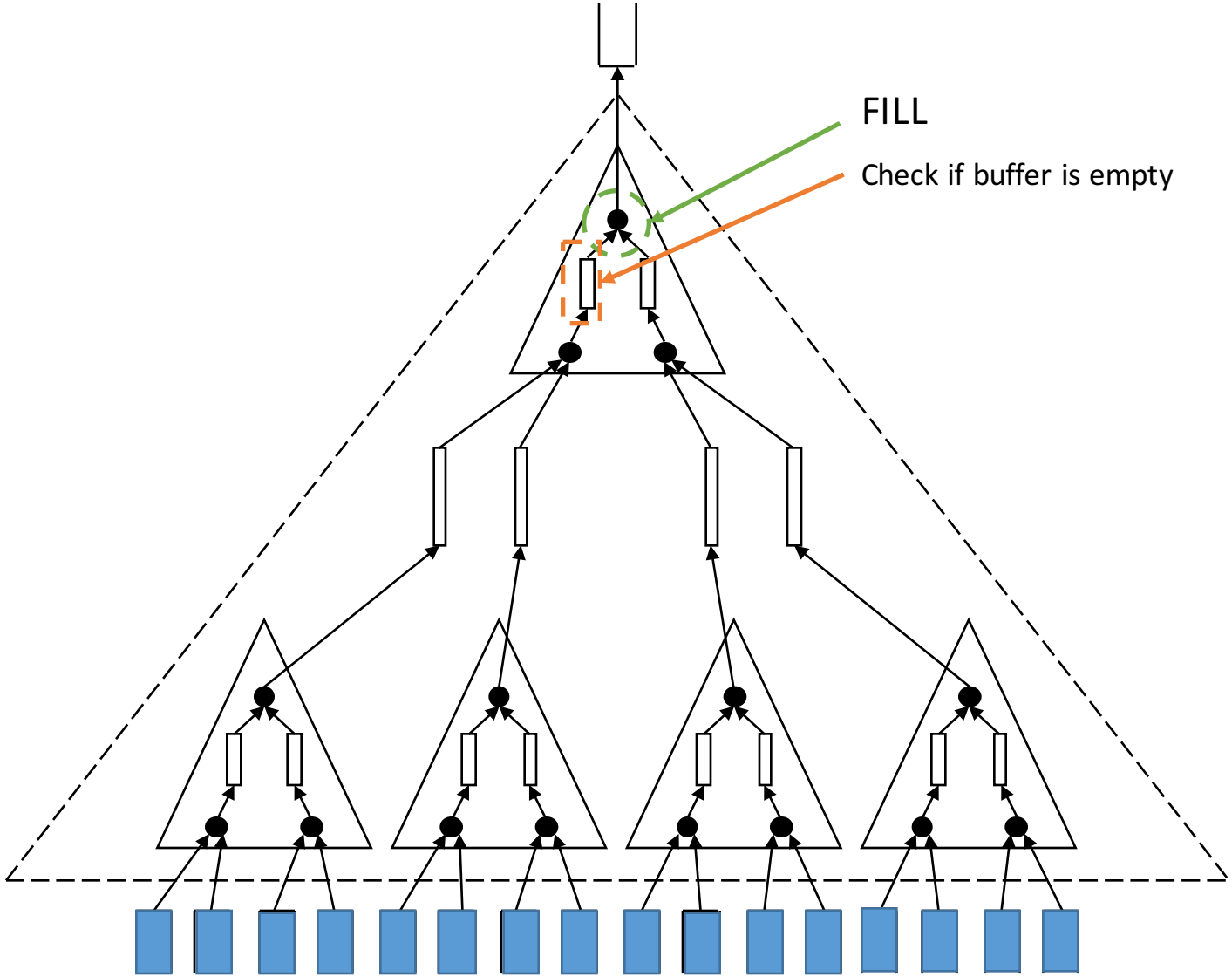
Binary Merger
Output Buffer
Left Input Buffer
Right Input Buffer
Left Child
Right Child

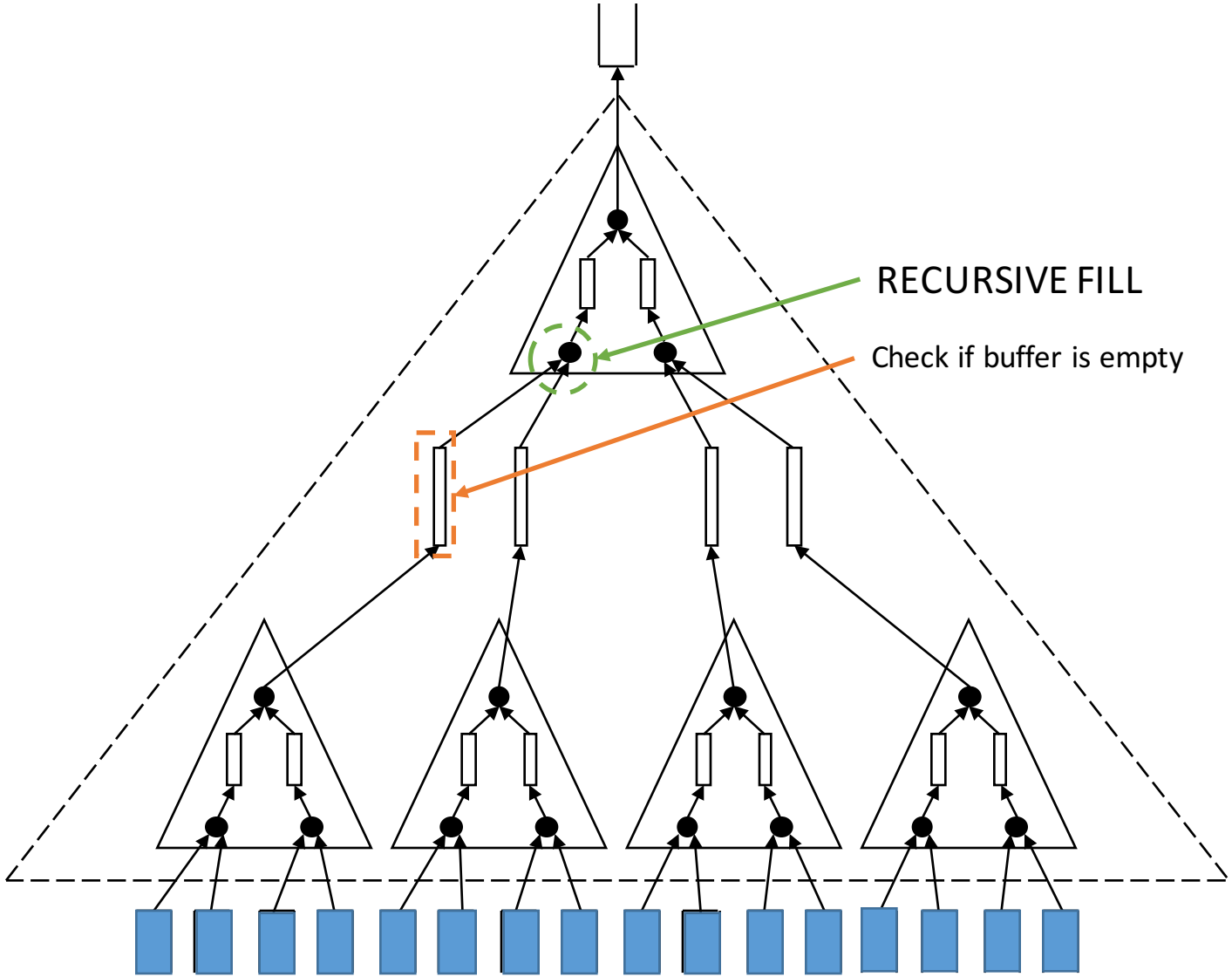
Merging via a Binary Merger

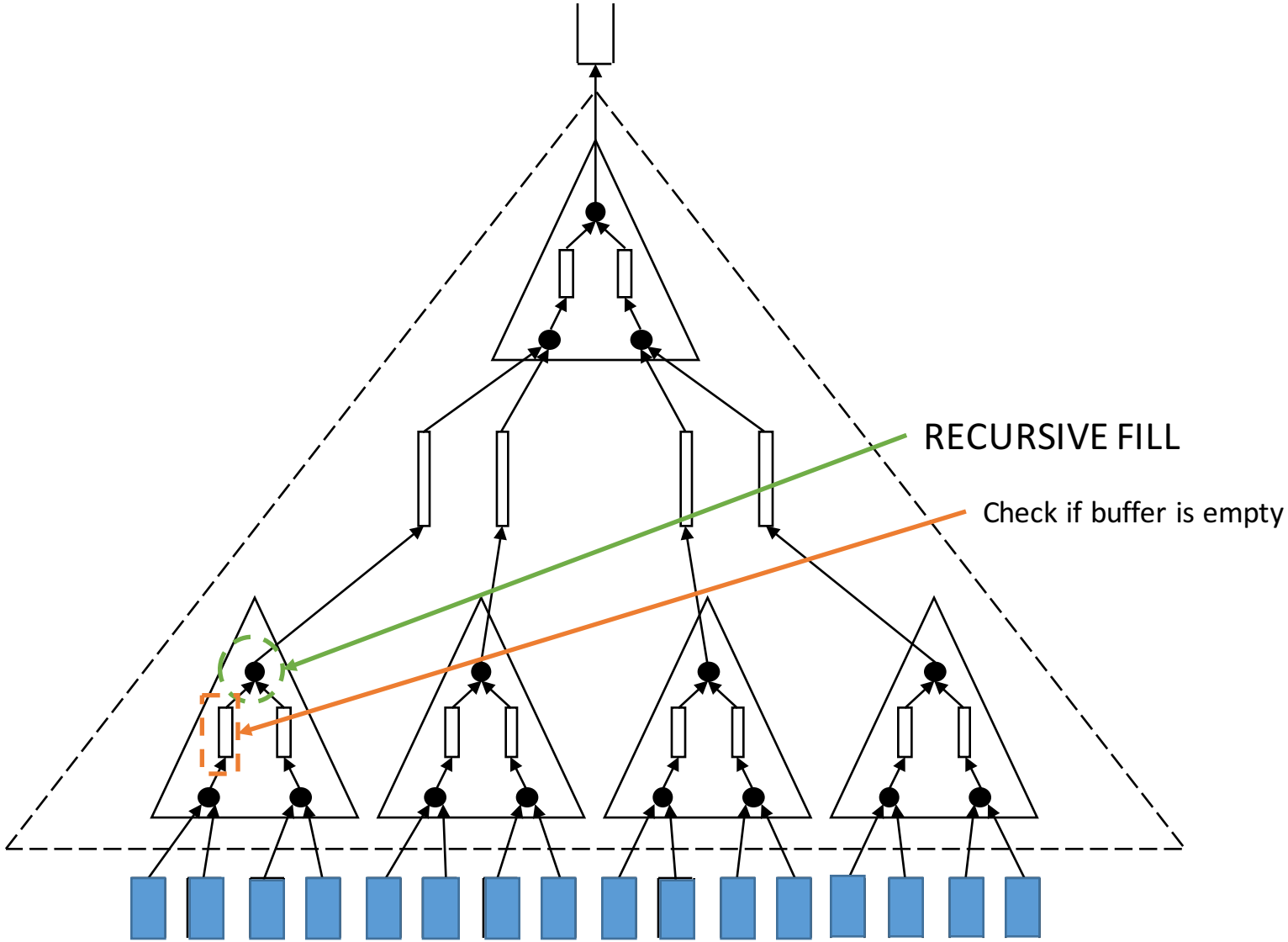
- Fill algorithm
- Repeat till output buffer is full
  - Check if input buffers are empty.
  - Recursively fill empty buffers.
  - Merge input buffers using standard merging.

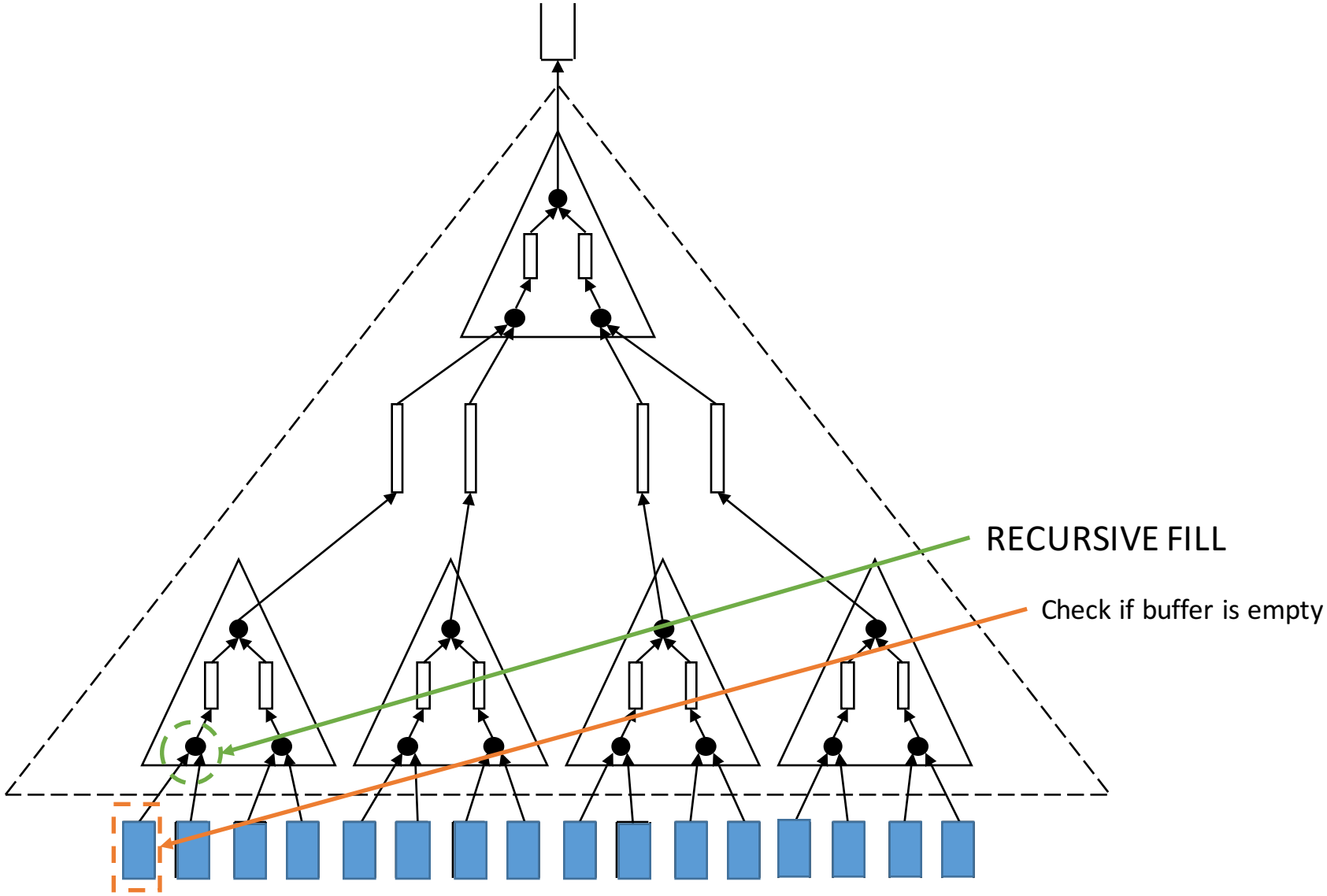


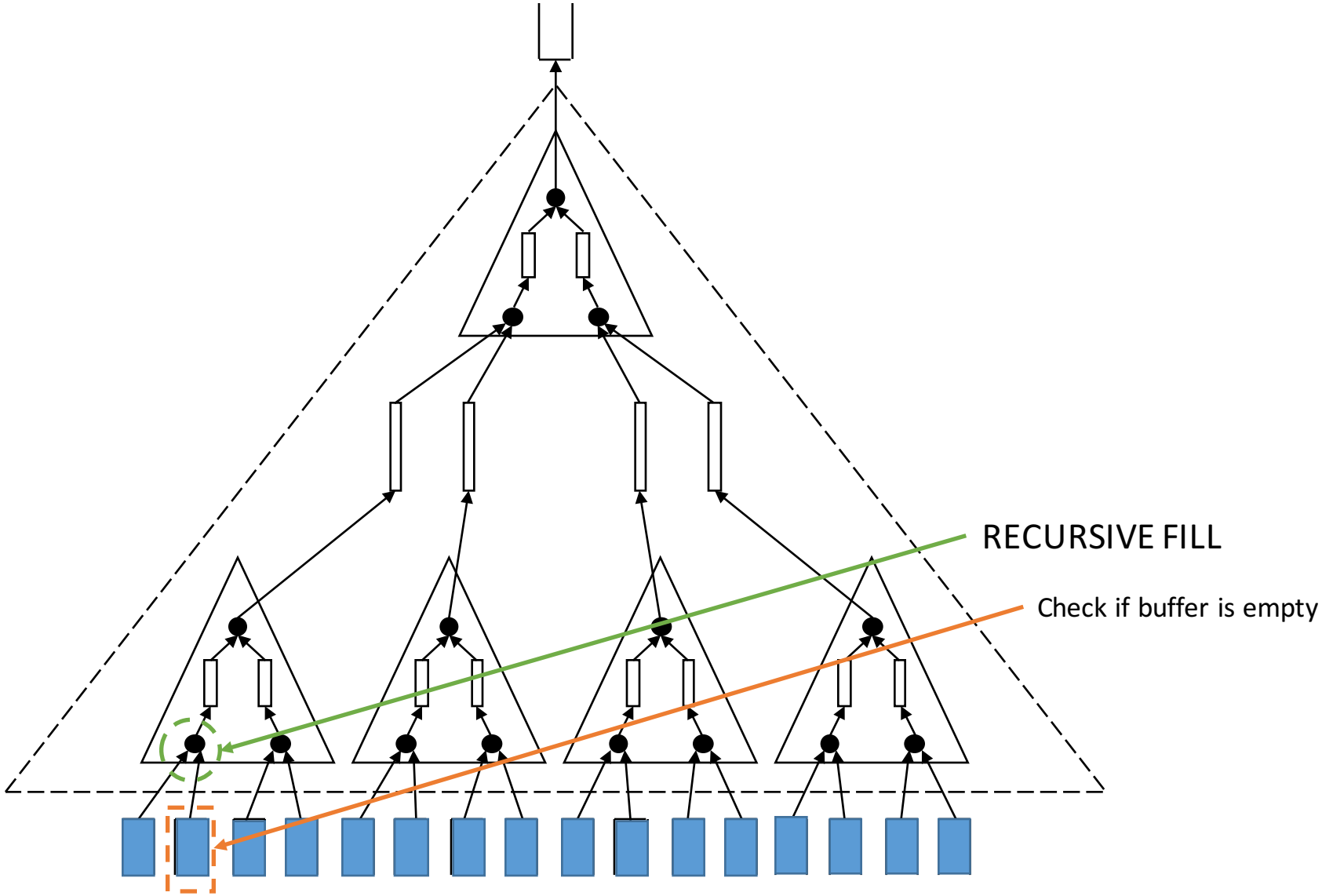

FILL

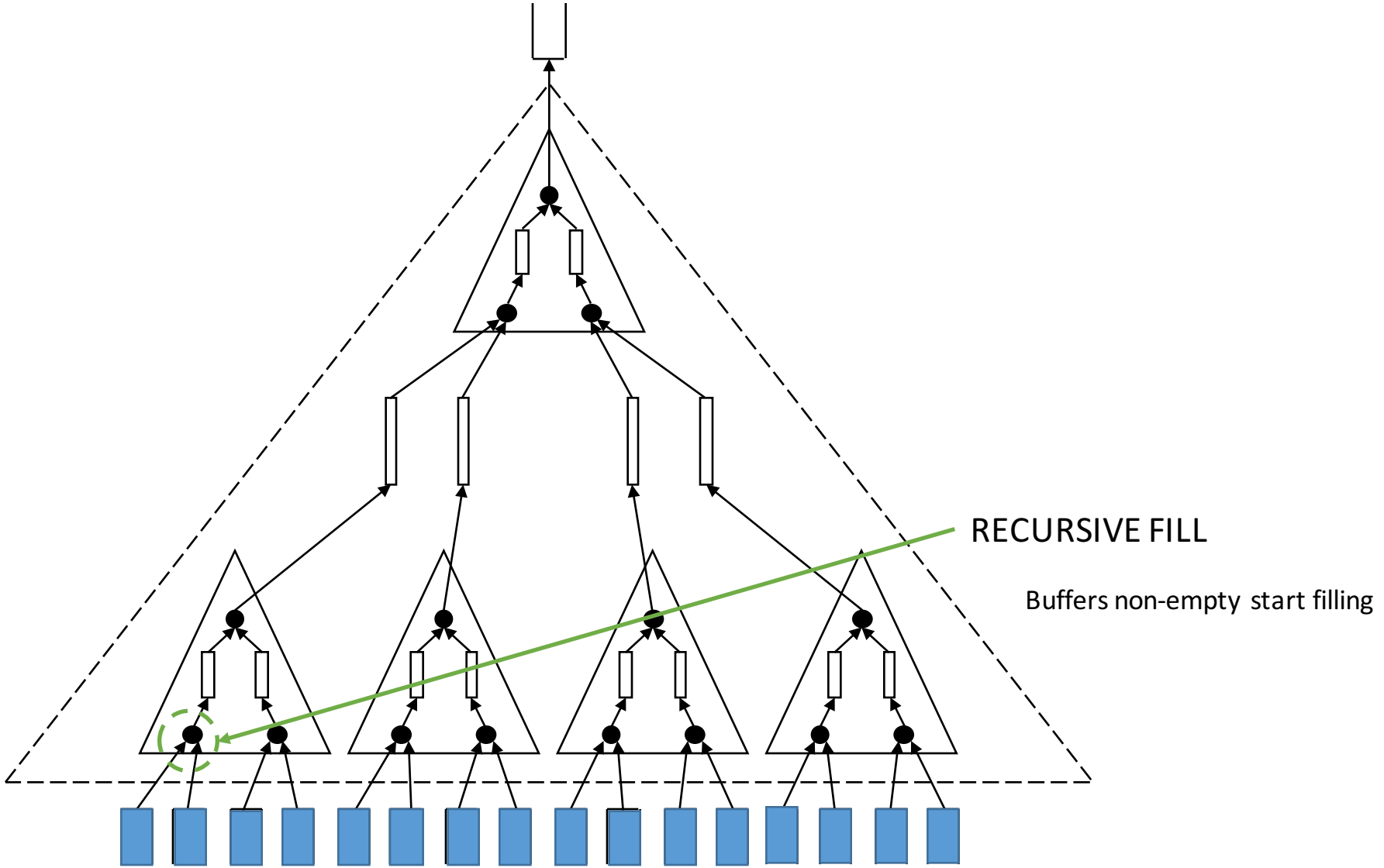

FILL
Check if buffer is empty

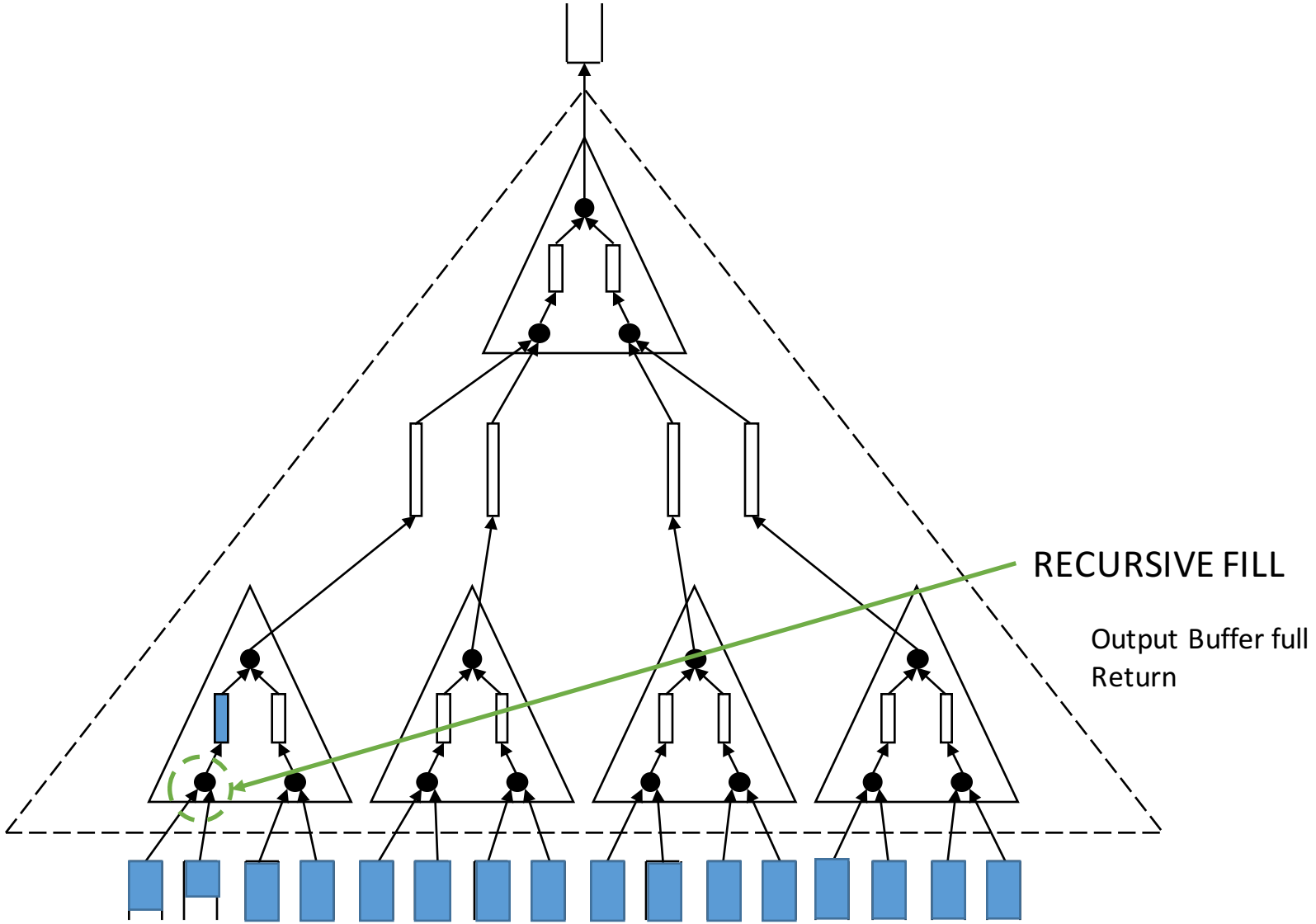

RECURSIVE FILL
Check if buffer is empty

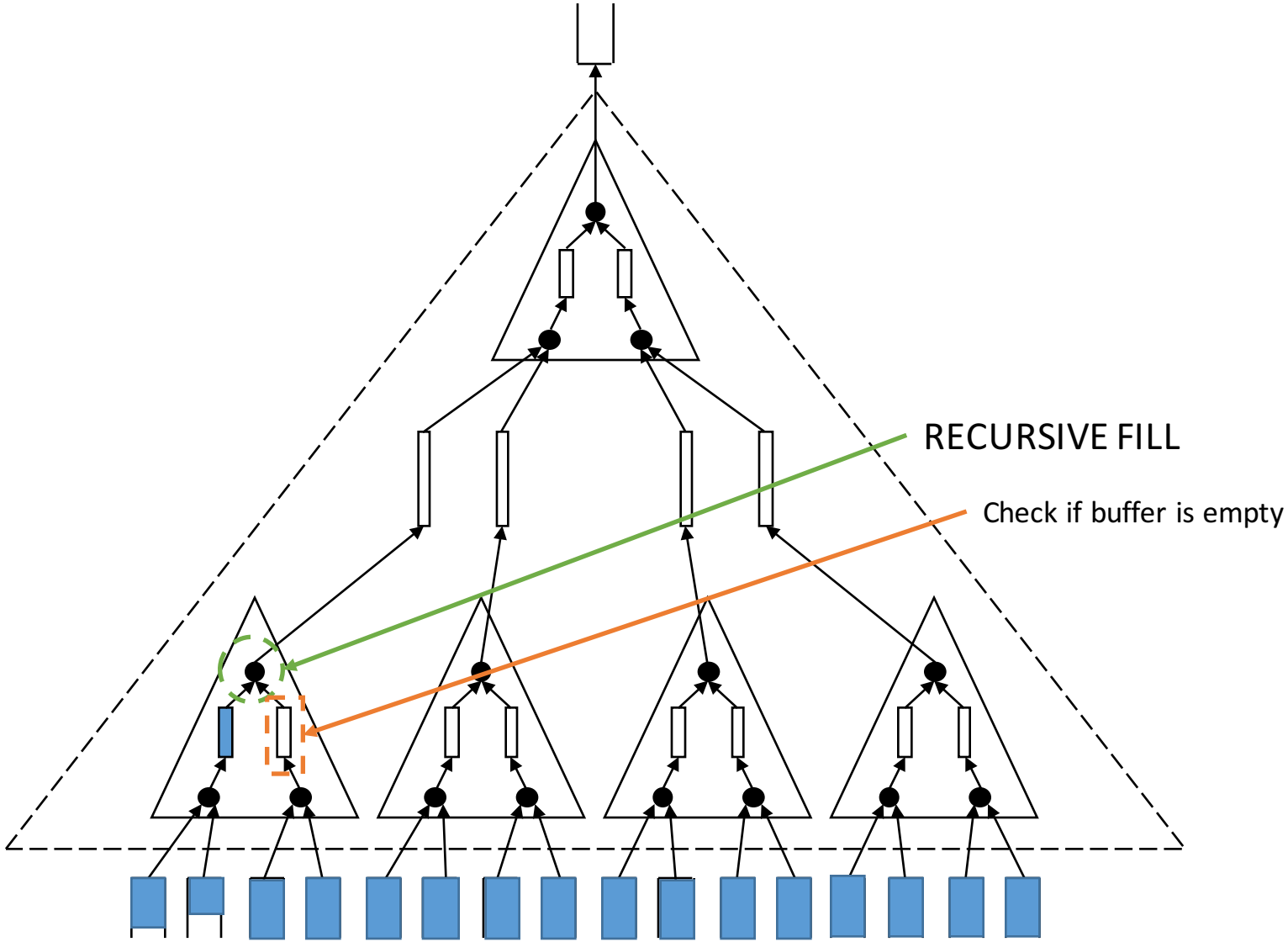

RECURSIVE FILL
Check if buffer is empty

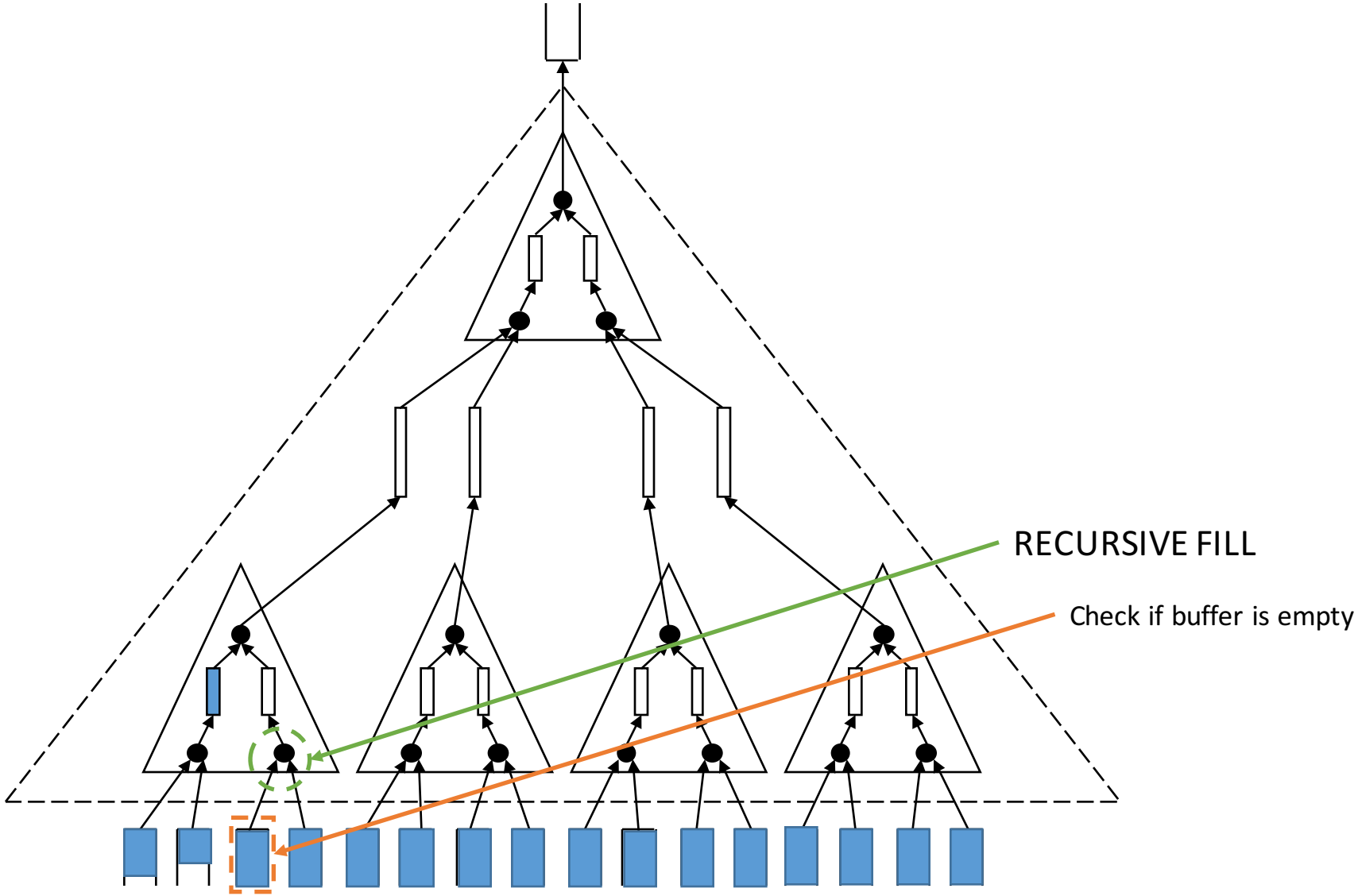

RECURSIVE FILL
Check if buffer is empty

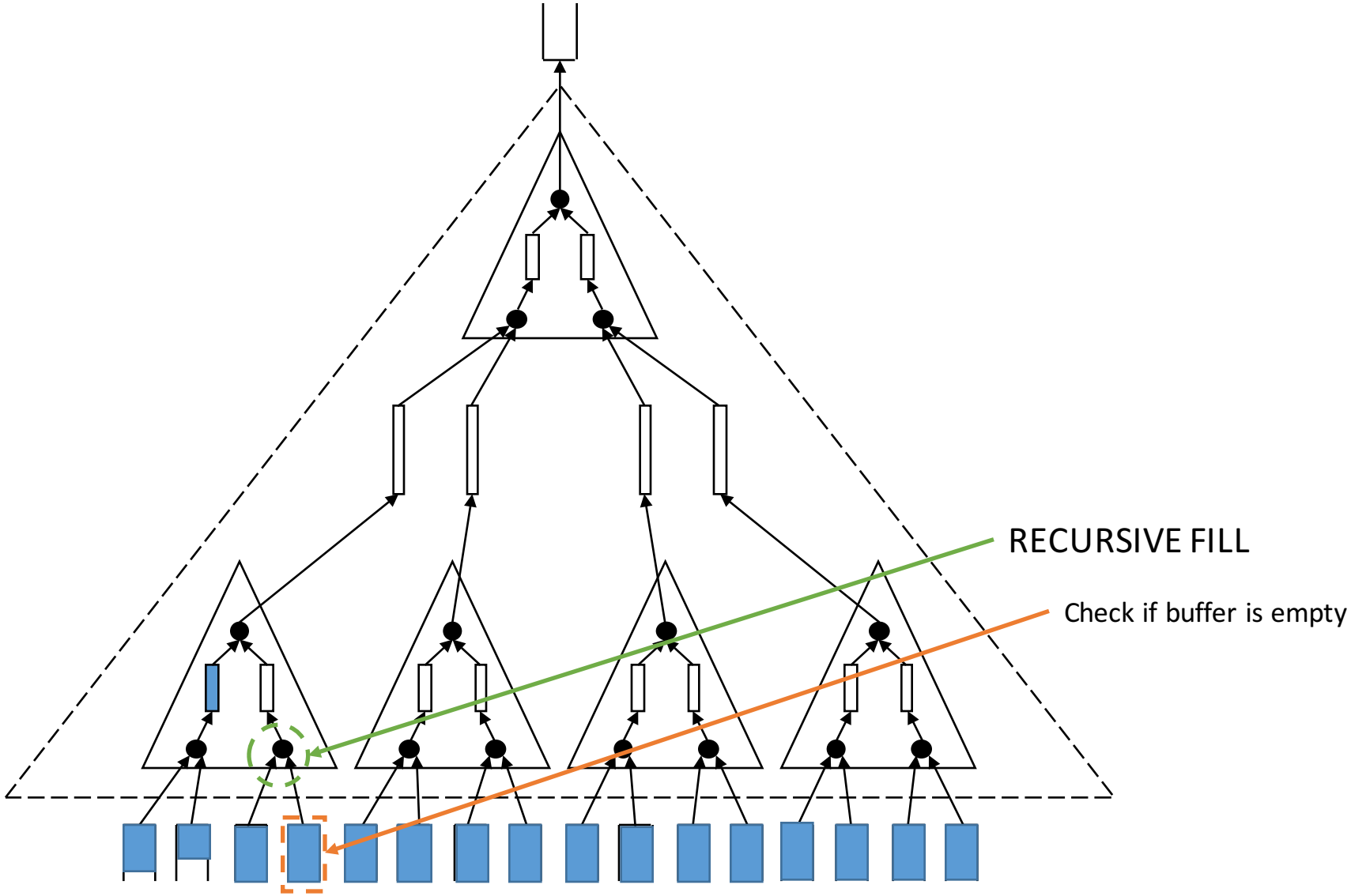

RECURSIVE FILL
Check if buffer is empty

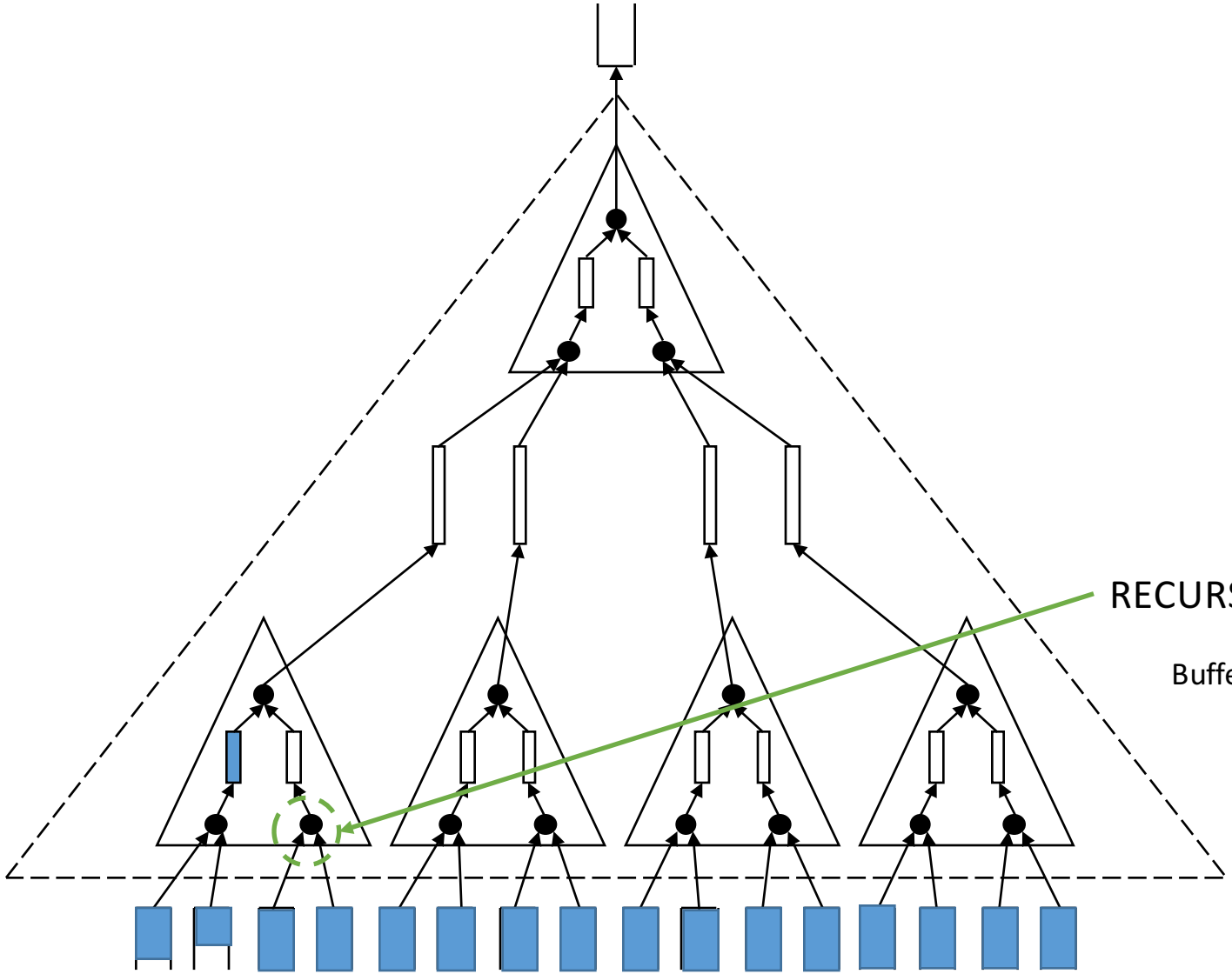

RECURSIVE FILL
Buffers non-empty start filling

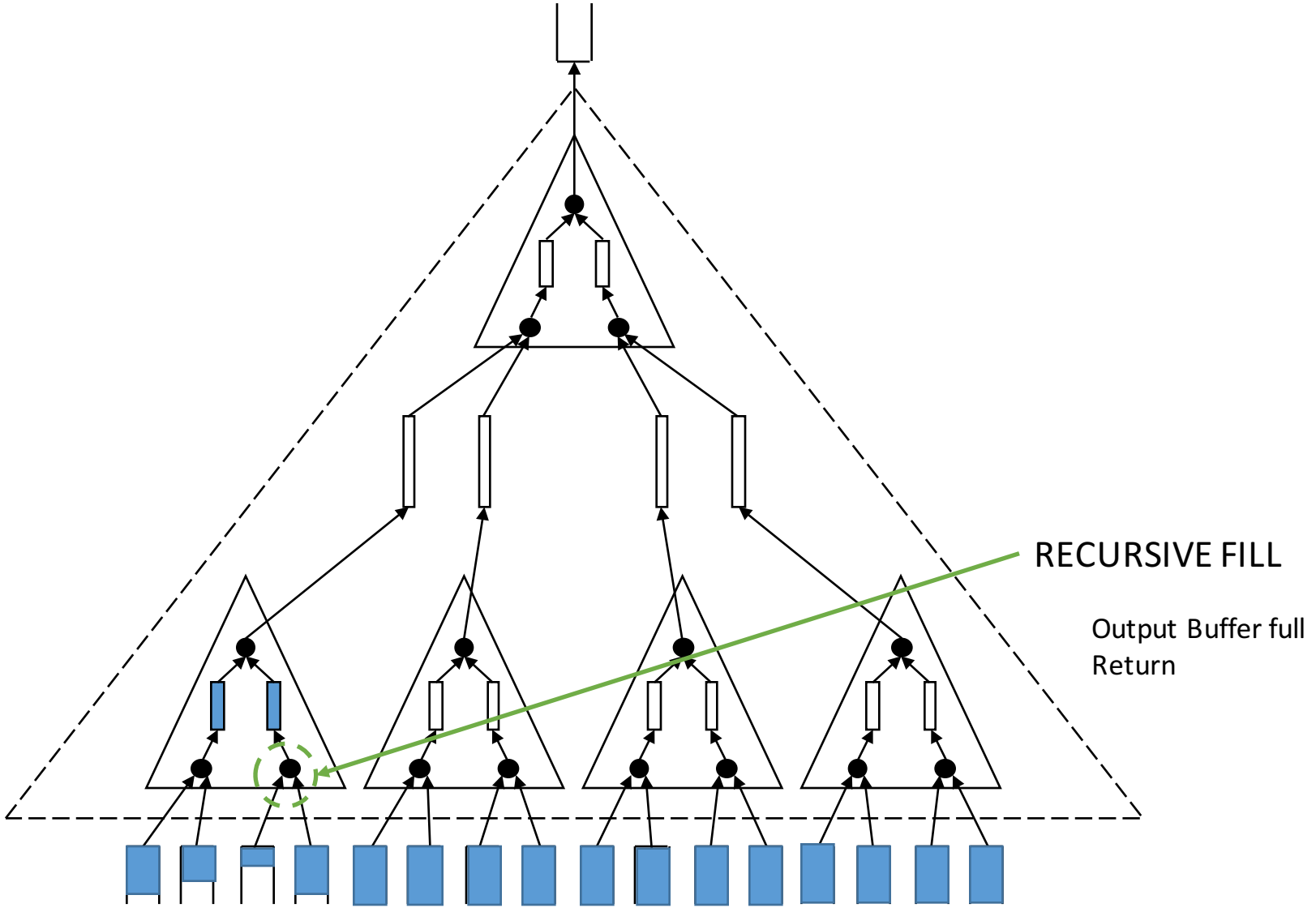

RECURSIVE FILL
Output Buffer full
Return

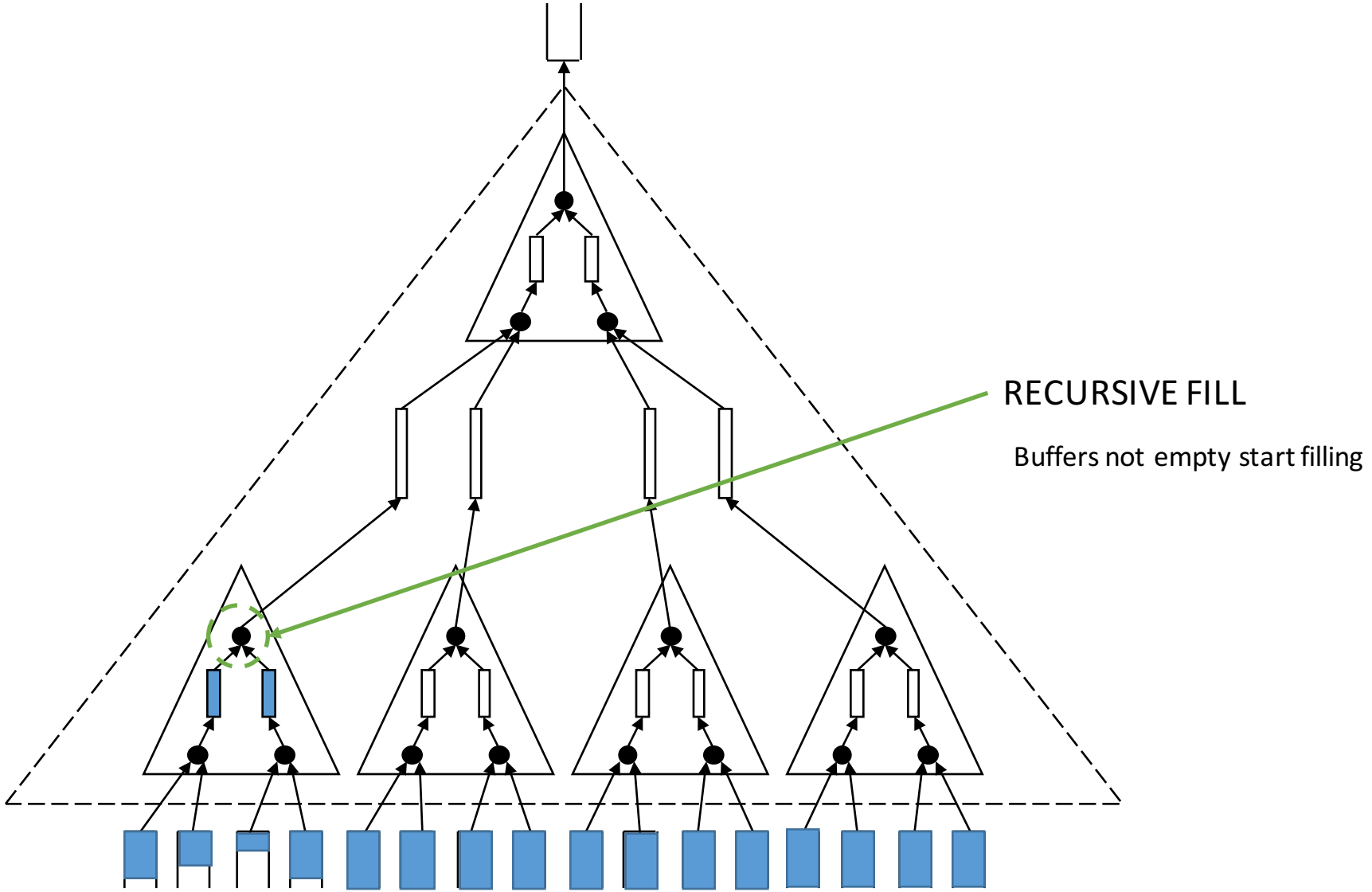

RECURSIVE FILL
Check if buffer is empty

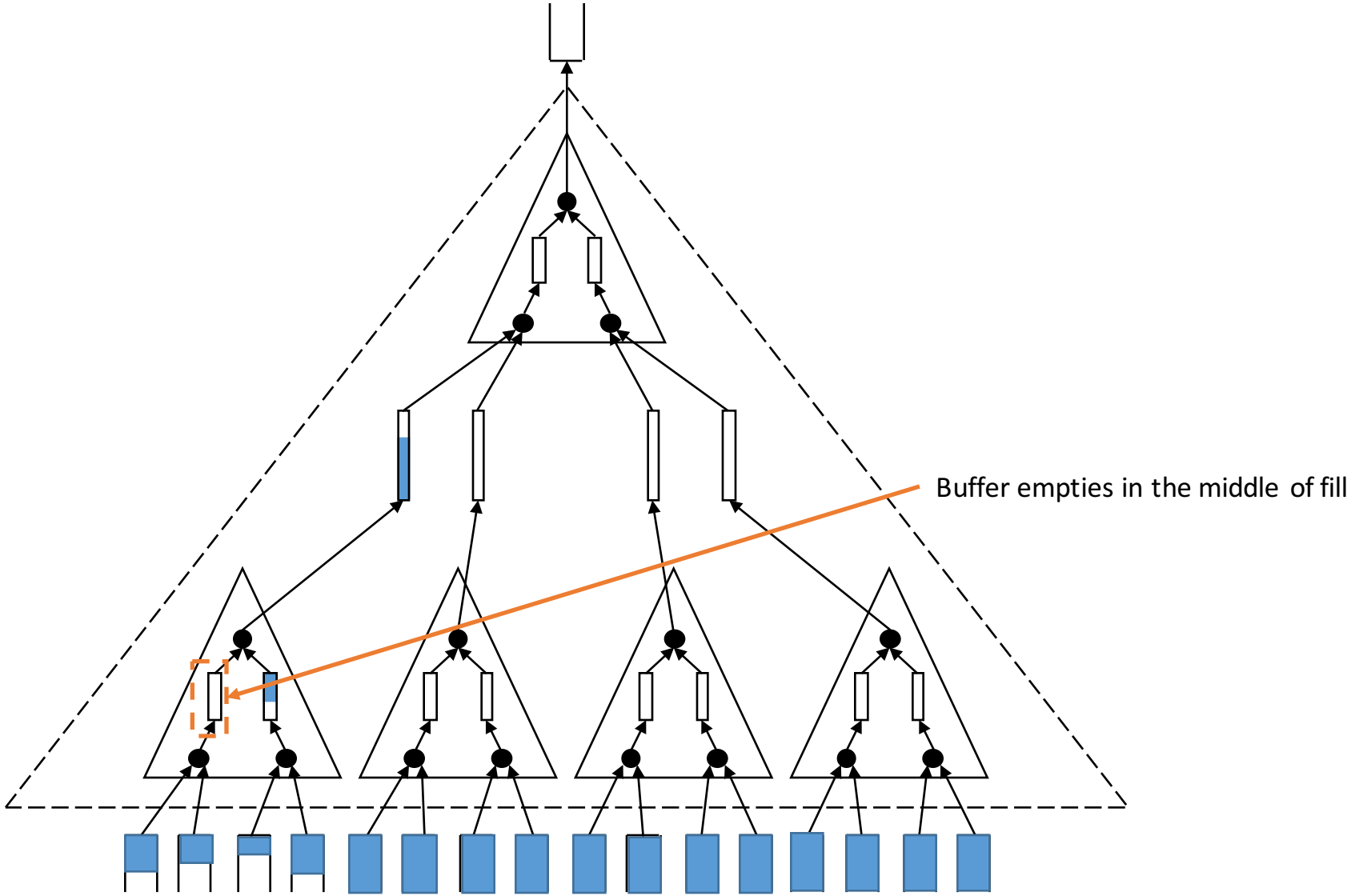

RECURSIVE FILL
Check if buffer is empty

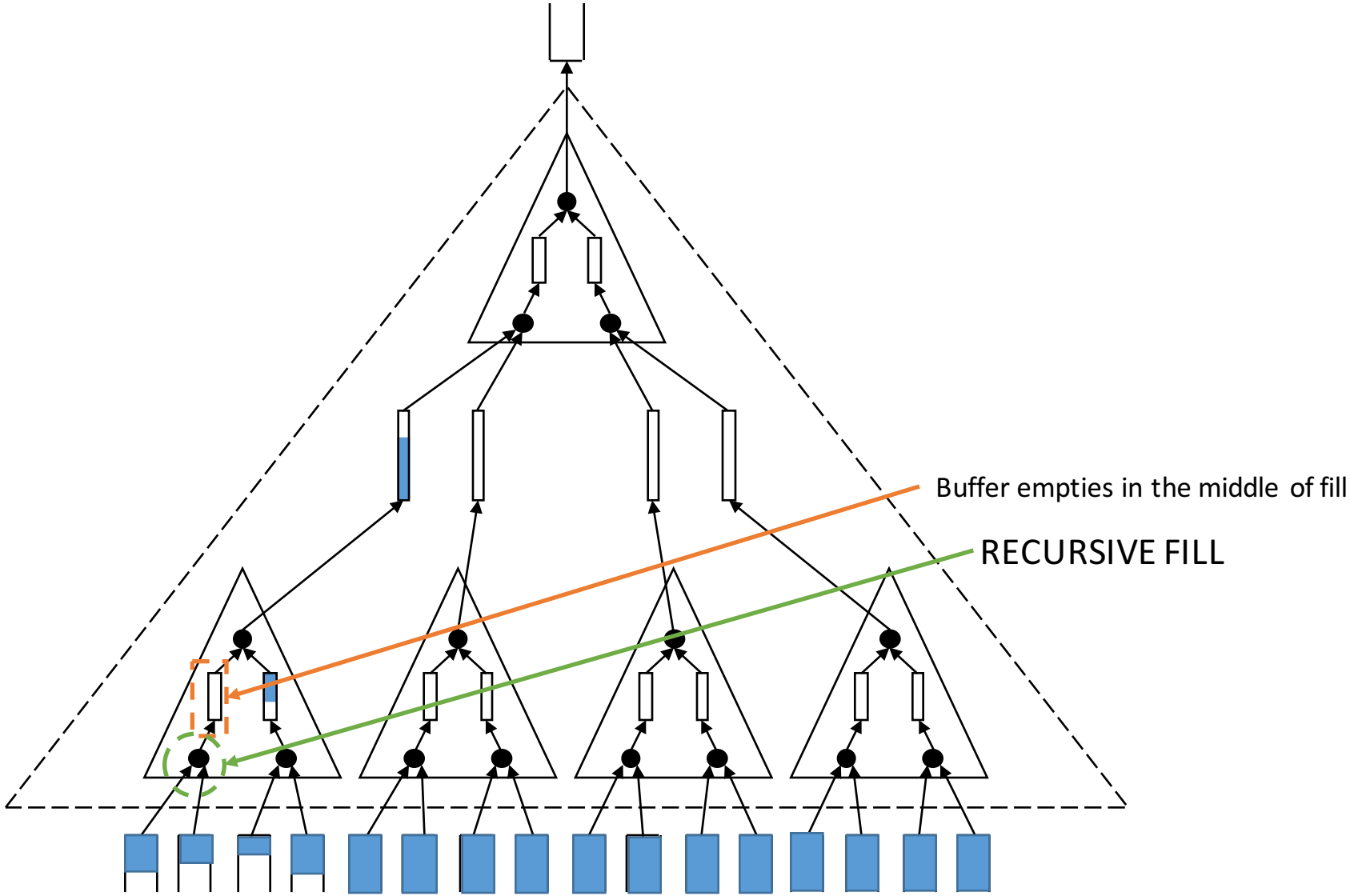

RECURSIVE FILL
Check if buffer is empty

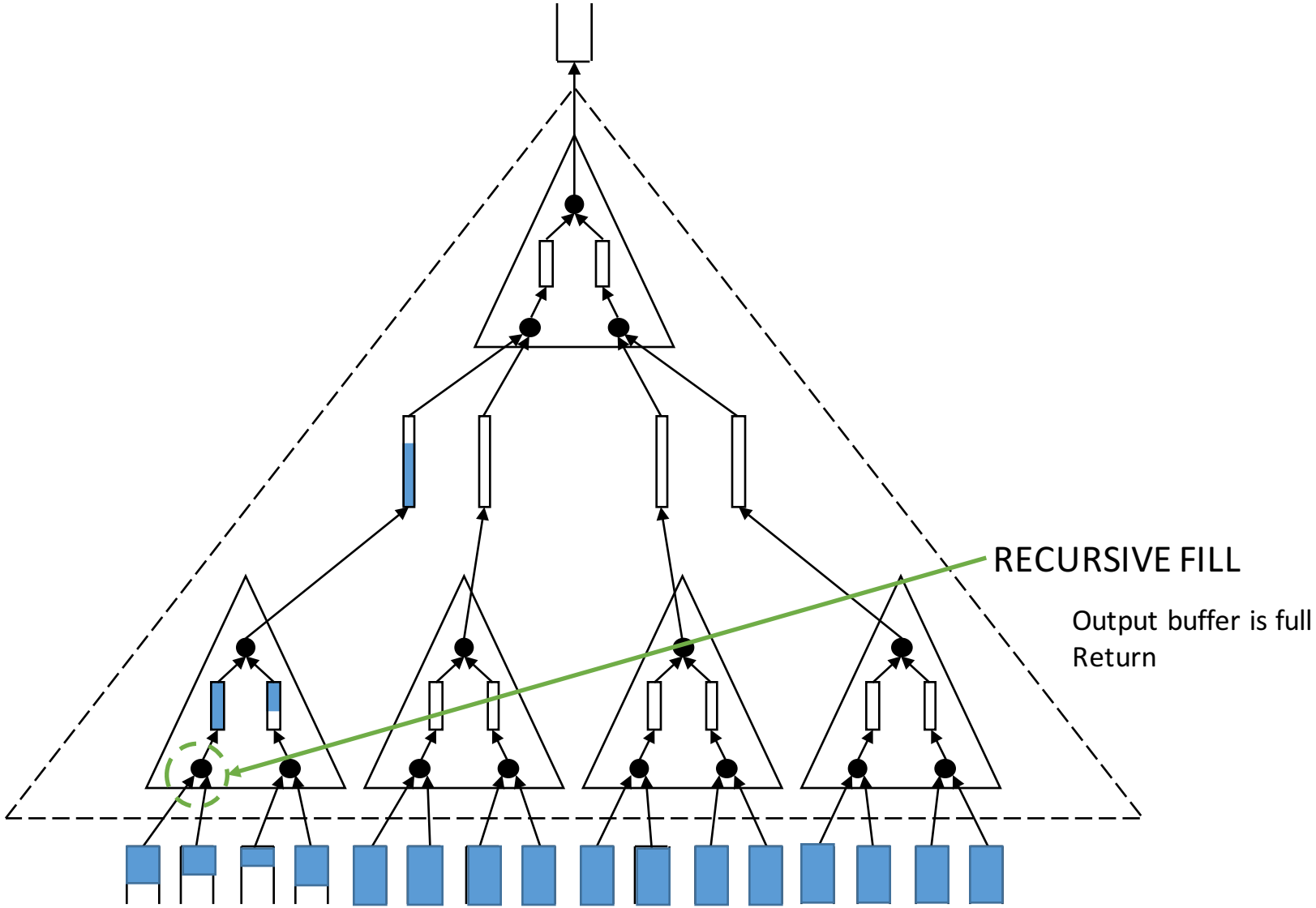

RECURSIVE FILL
Buffers not empty start filling.

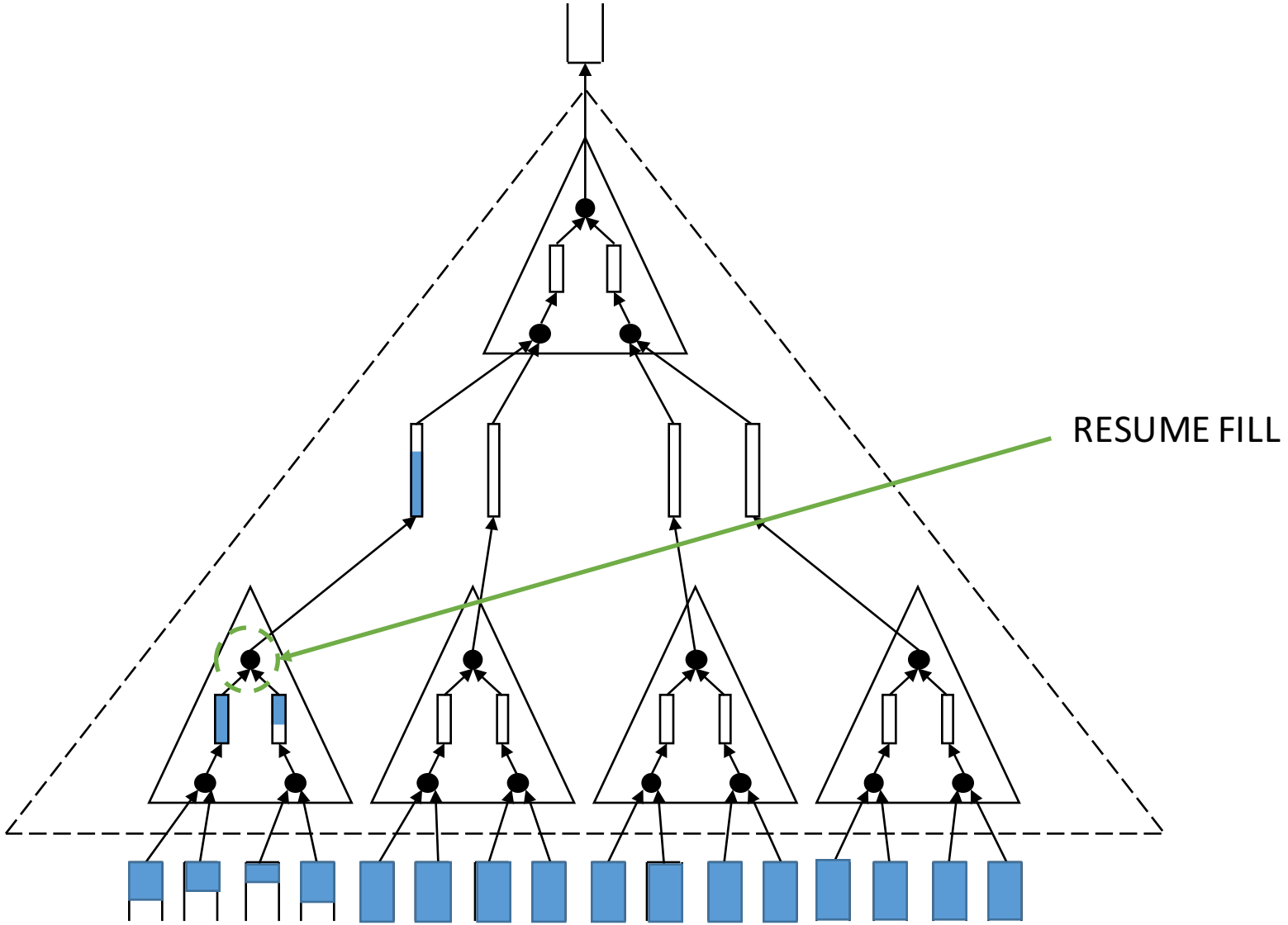

RECURSIVE FILL
Output Buffer full
Return

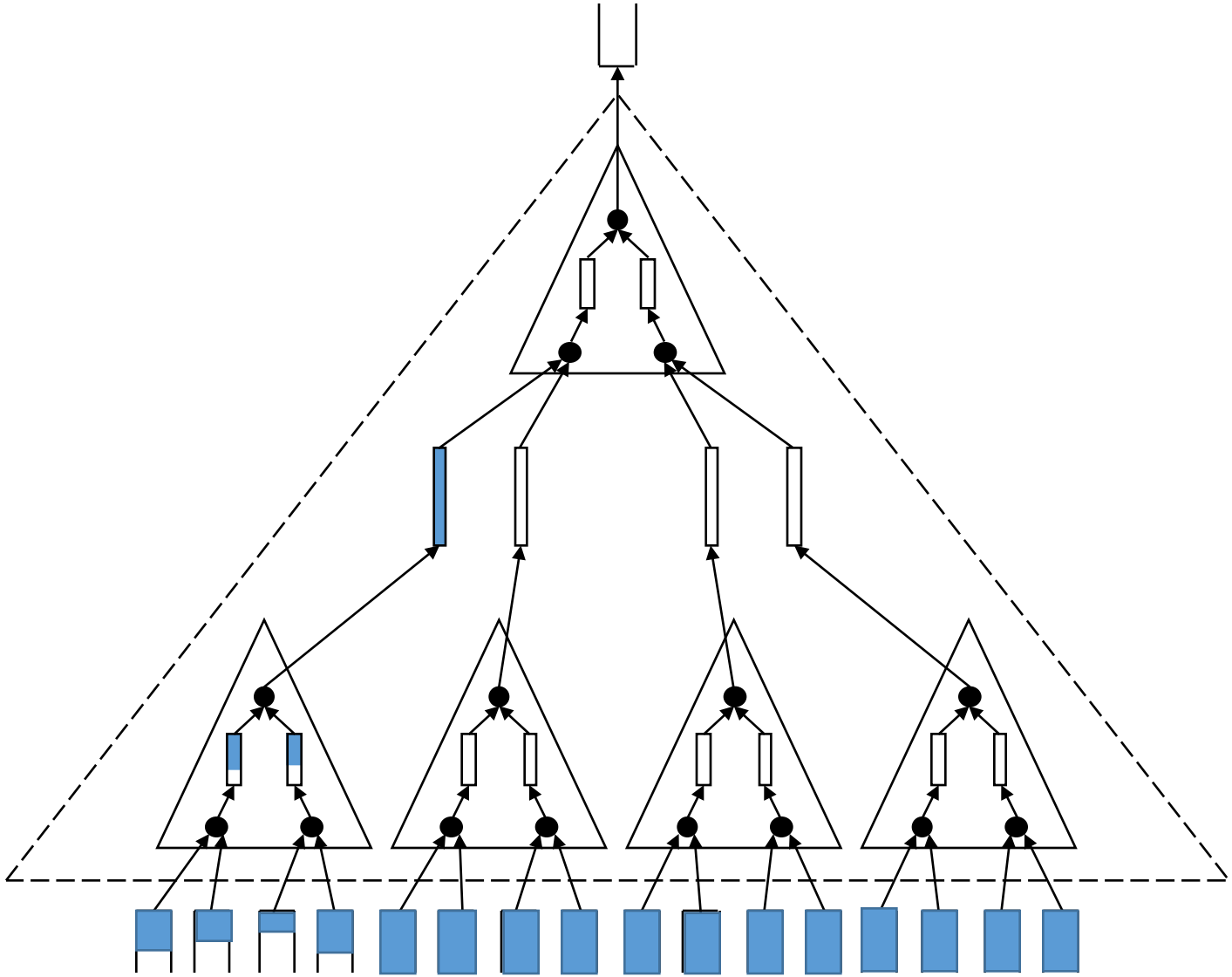

RECURSIVE FILL
Buffers not empty start filling

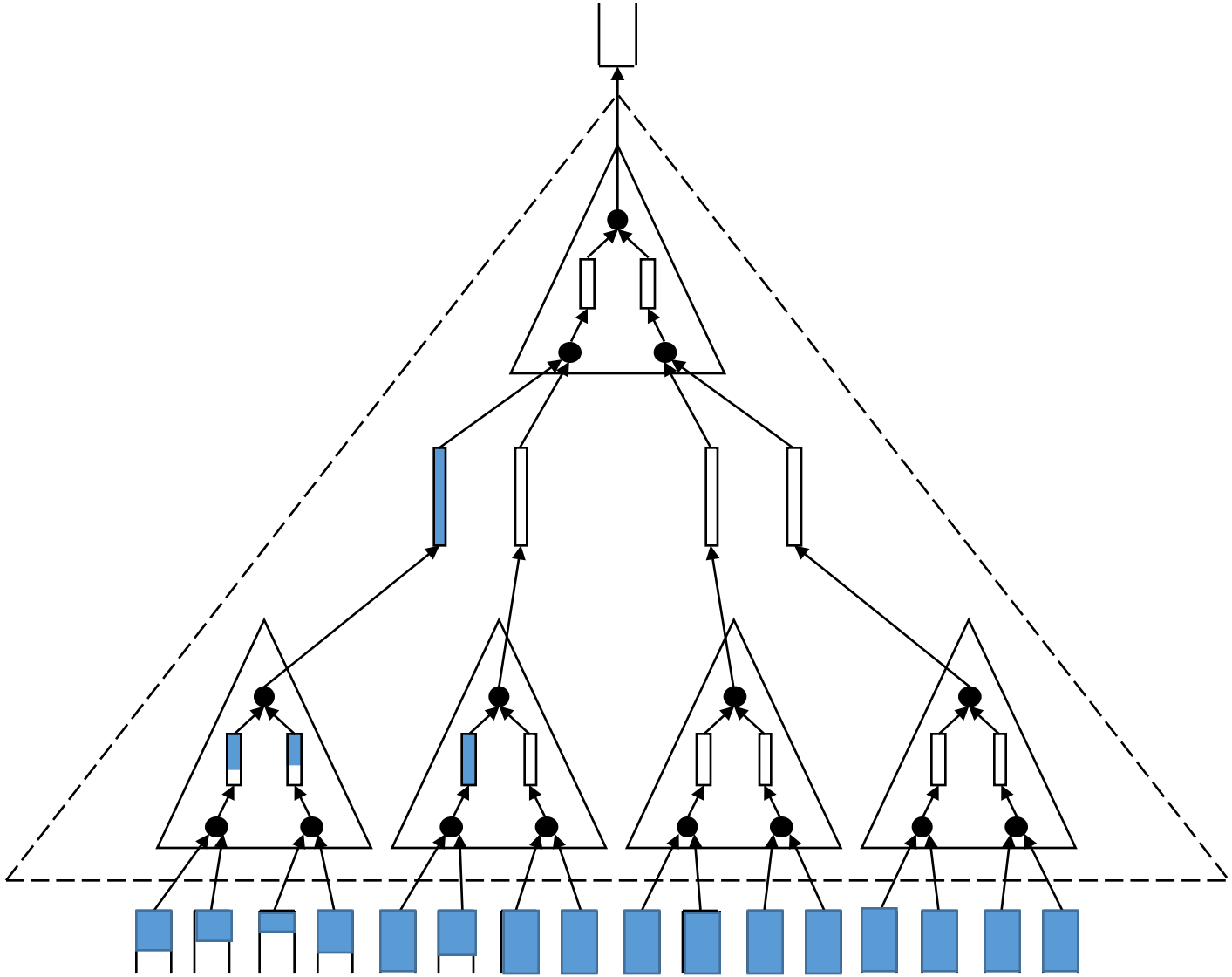

Buffer empties in the middle of fill

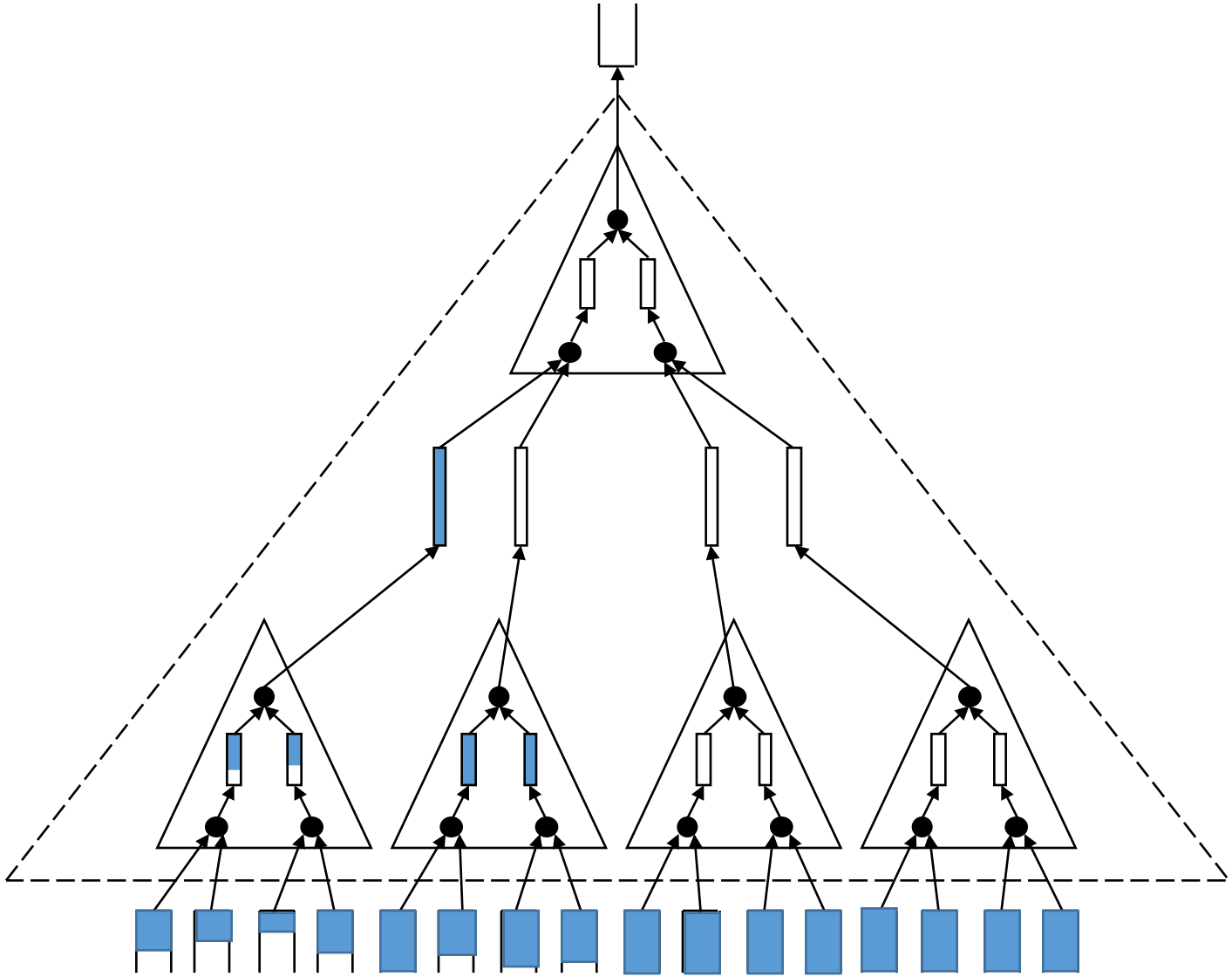

Buffer empties in the middle of fill
RECURSIVE FILL

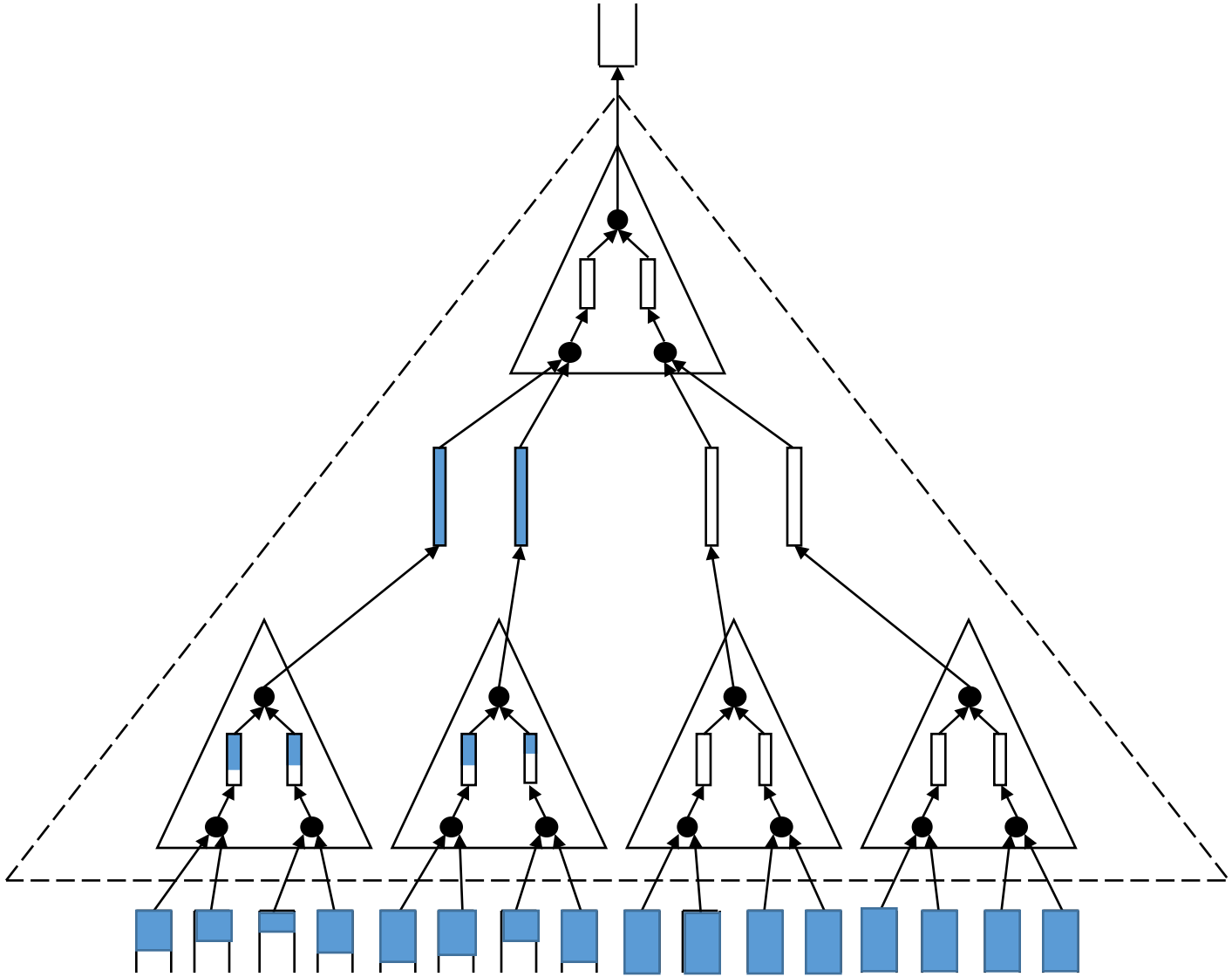

RECURSIVE FILL
Output buffer is full
Return


RESUME FILL





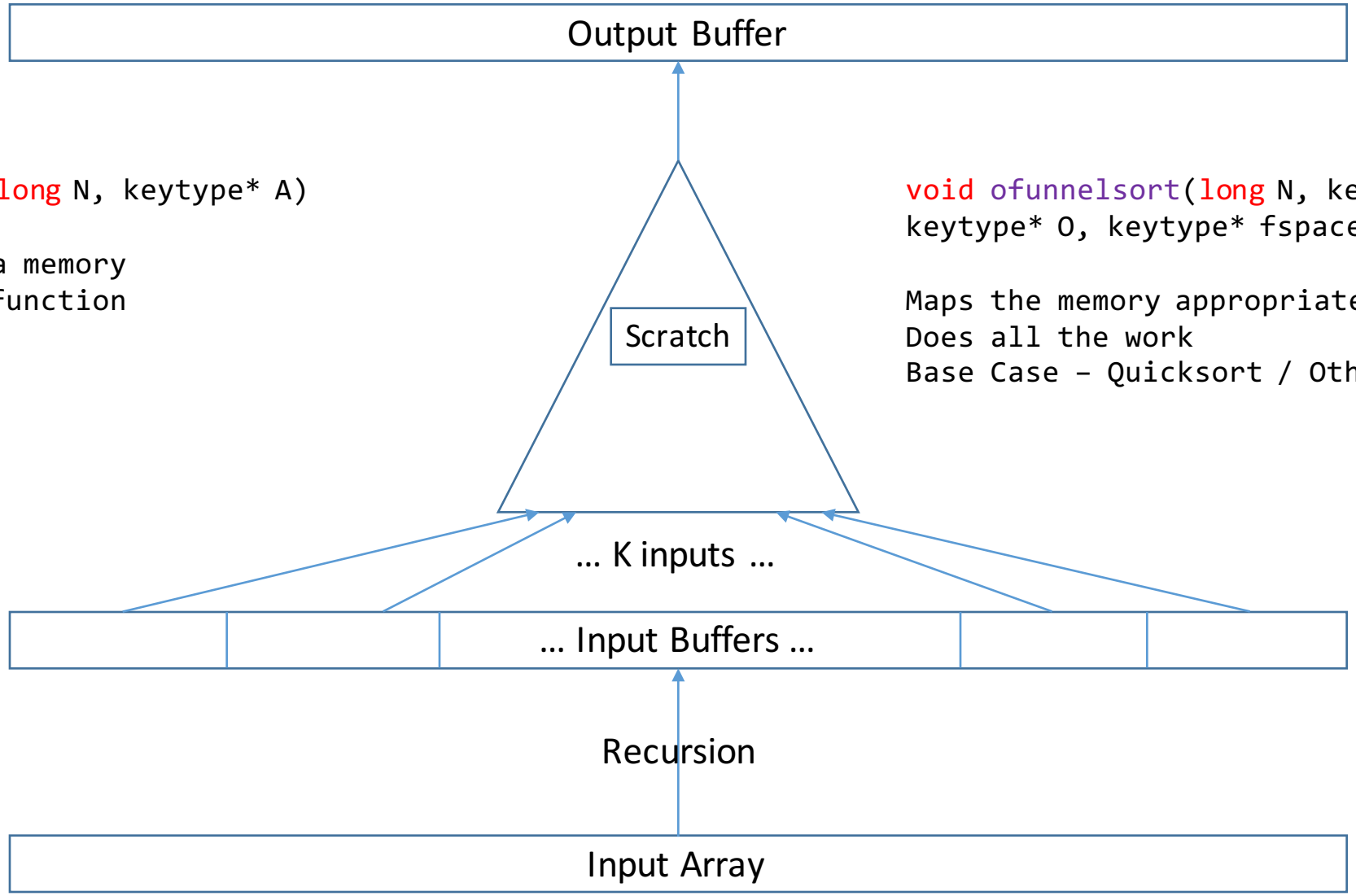




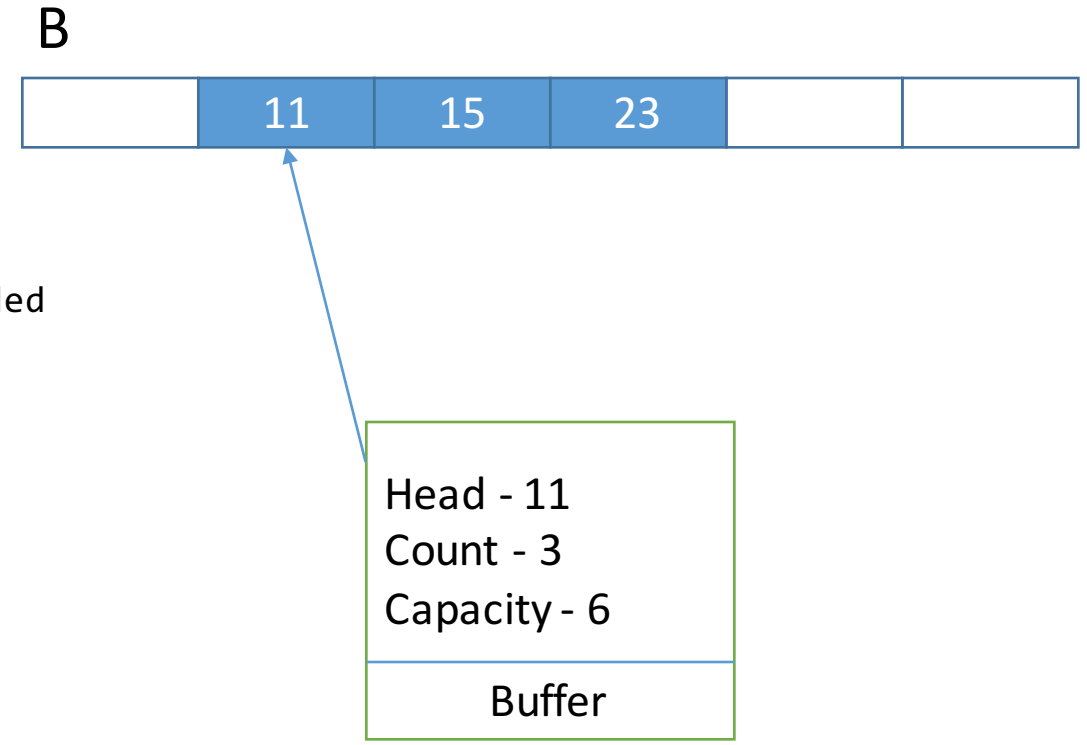
Tips

- Ensure K is correctly calculated.
- Only internal buffers need to be contiguous. Scaffolding elements like pointers to binary mergers, buffers and other temporary variables can be regularly “malloced”.
- What is the return type of Funnel Initialization?

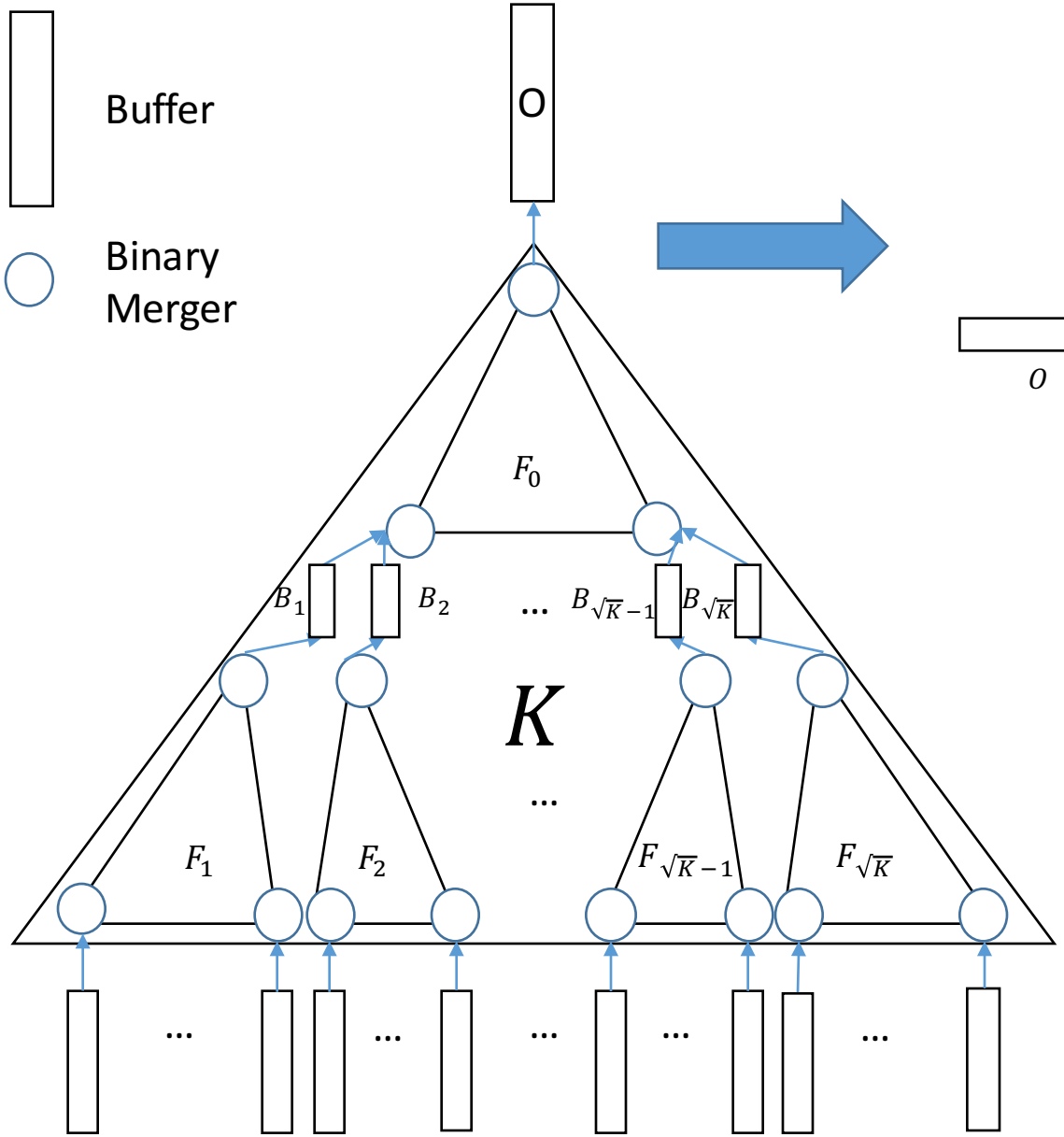
```
void funnelSort(long N, keytype* A)

Initializes extra memory
Calls recursive function
```

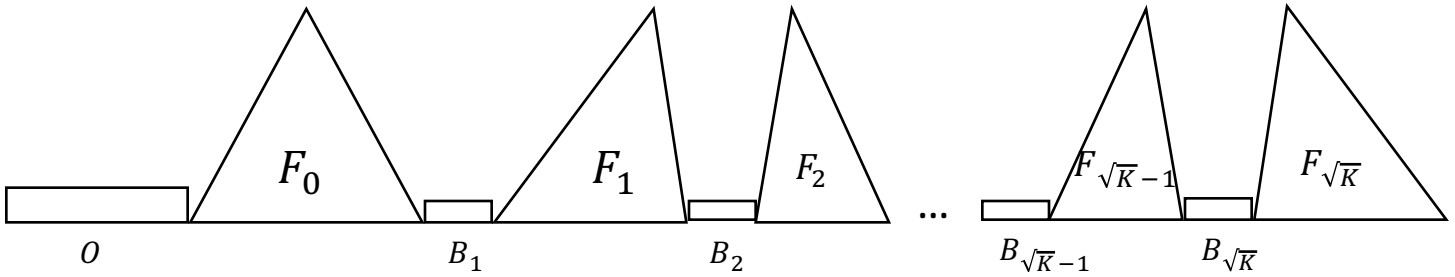



```
void ofunnelsort(long N, keytype* A, int d,
keytype* O, keytype* fspace)

Maps the memory appropriately
Does all the work
Base Case – Quicksort / Other Sort
```



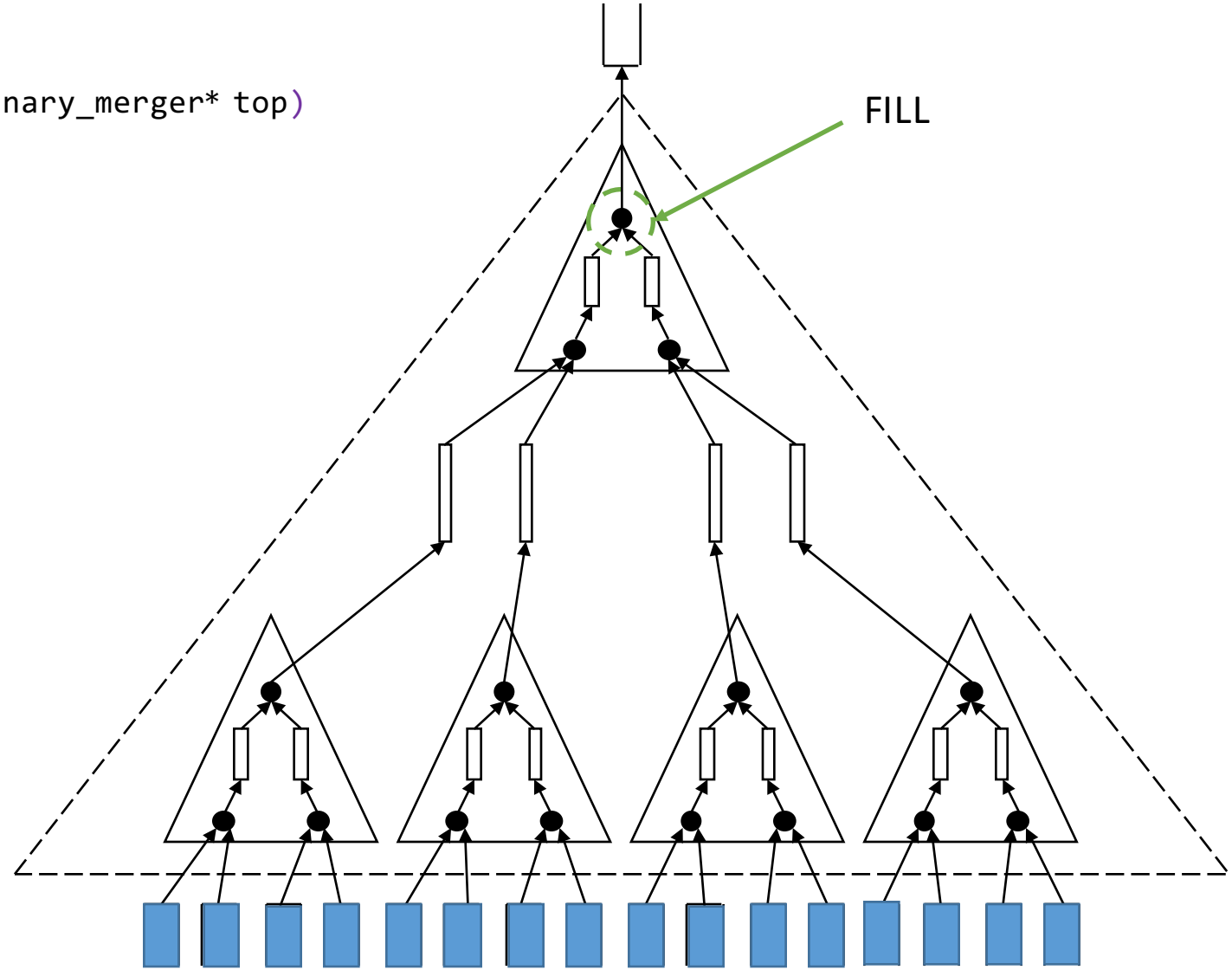

```
Buffer Interface is provided

struct buffer

void buffer_init_mem
void buffer_init
void increment
void buffer_destroy
void buffer_insert
keytype buffer_extract
keytype buffer_peep
```






```
struct binary_merger

void binary_merger_init

struct binary_merger* funnel_init
struct binary_merger* funnel_init_rec
```

```
void fill (struct binary_merger* top)
```



References

- Demaine, Erik D. "Cache-oblivious algorithms and data structures." *Lecture Notes from the EEF Summer School on Massive Data Sets* 8, no. 4 (2002): 1-249.
- Rønn, Frederik. "Cache-oblivious searching and sorting." PhD diss., Master's thesis, University of Copenhagen, 2003.
- Demaine lecture

Thank You

- Questions