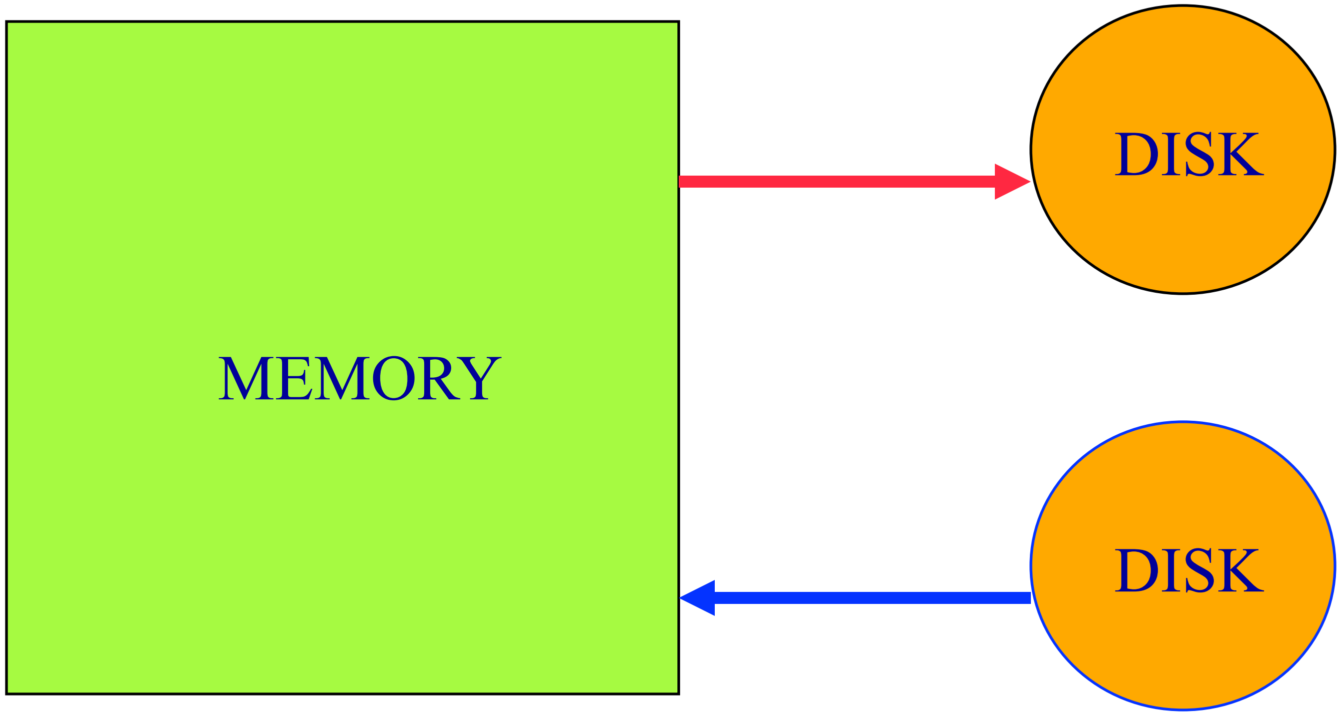
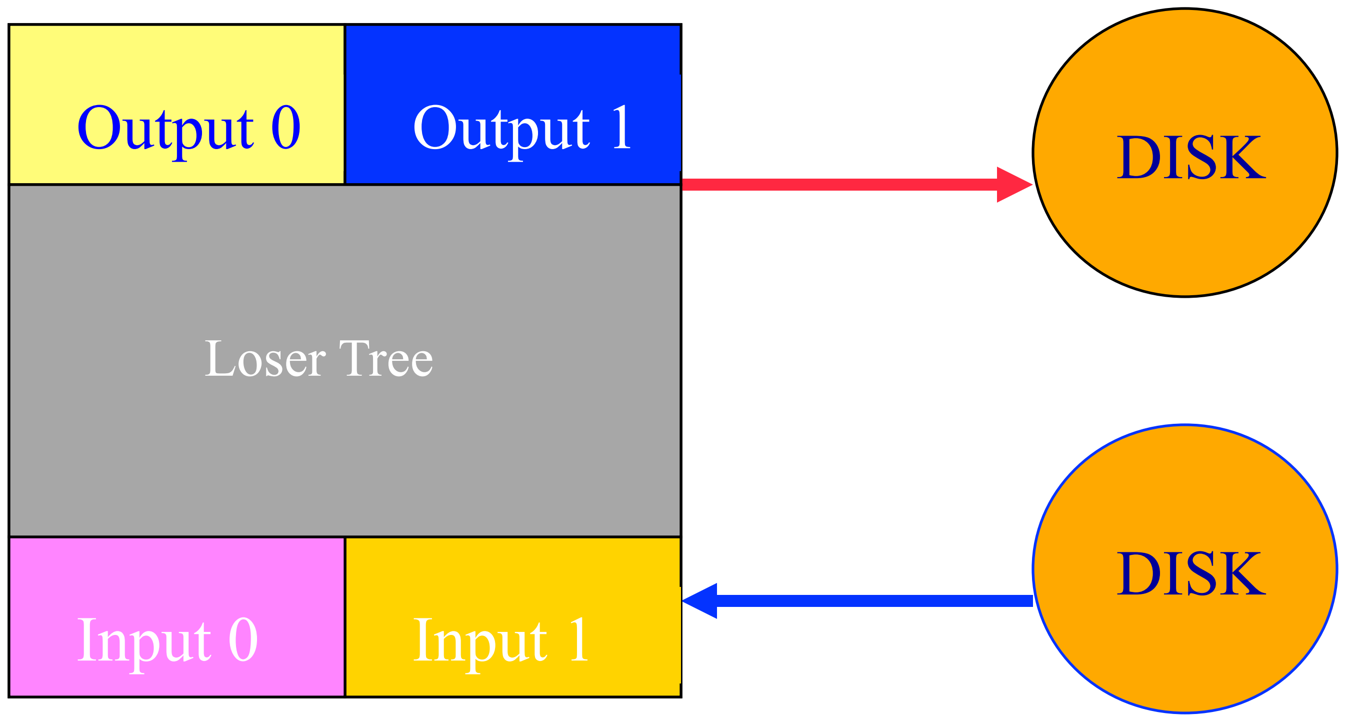
Improve Run Generation

- Overlap input,output, and internal CPU work.
- Reduce the number of runs (equivalently, increase average run length).

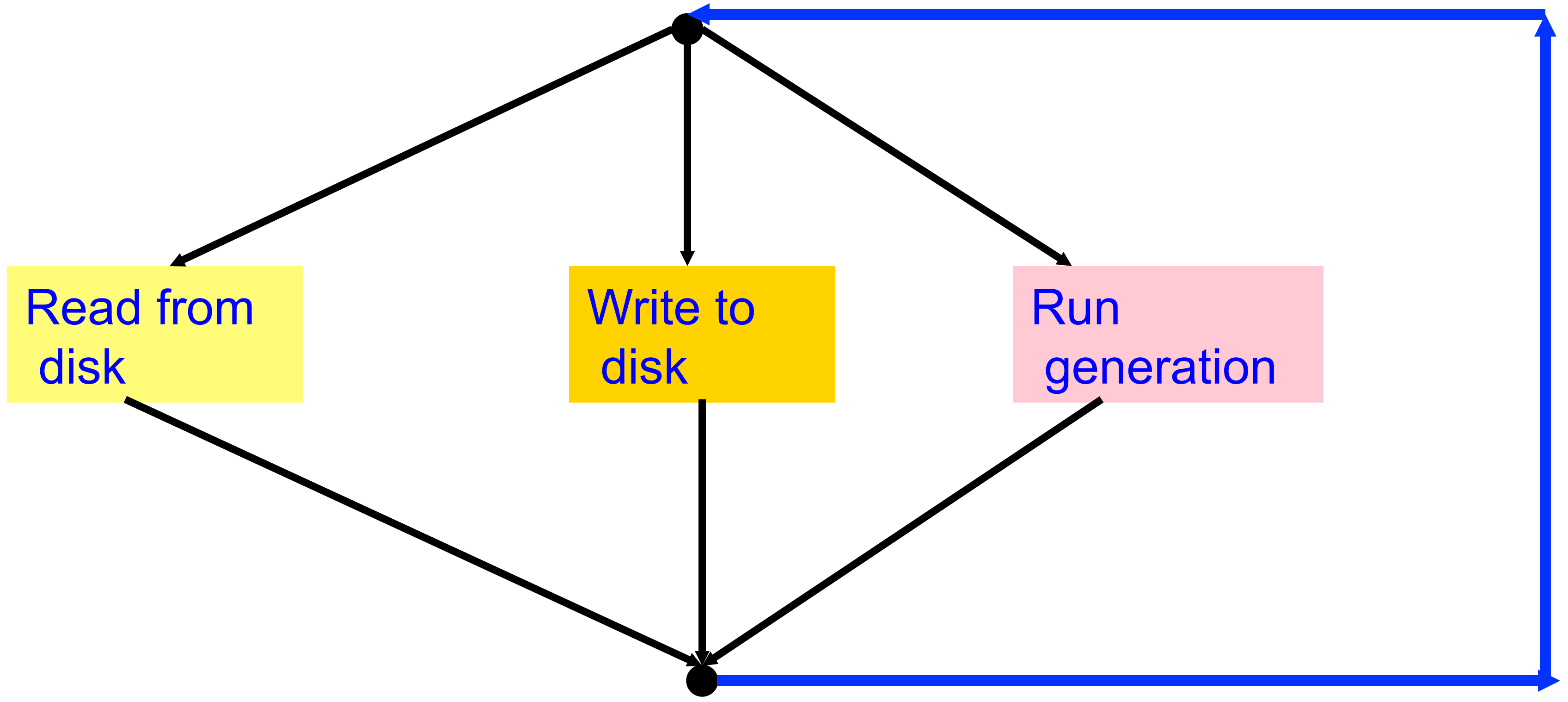


New Strategy



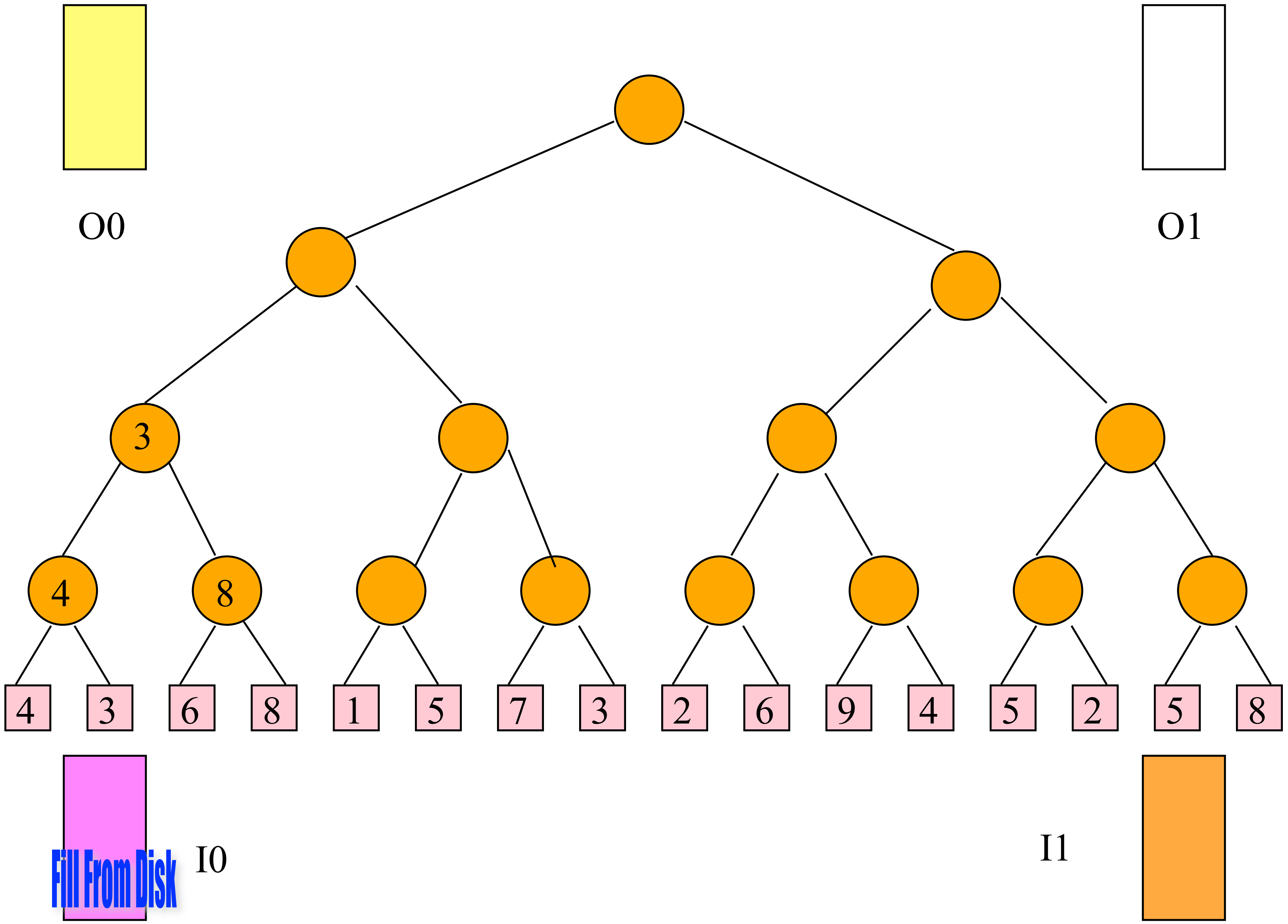
- Use 2 input and 2 output buffers.
- Rest of memory is used for a min loser tree.
- Actually, 3 buffers adequate.

Steady State Operation

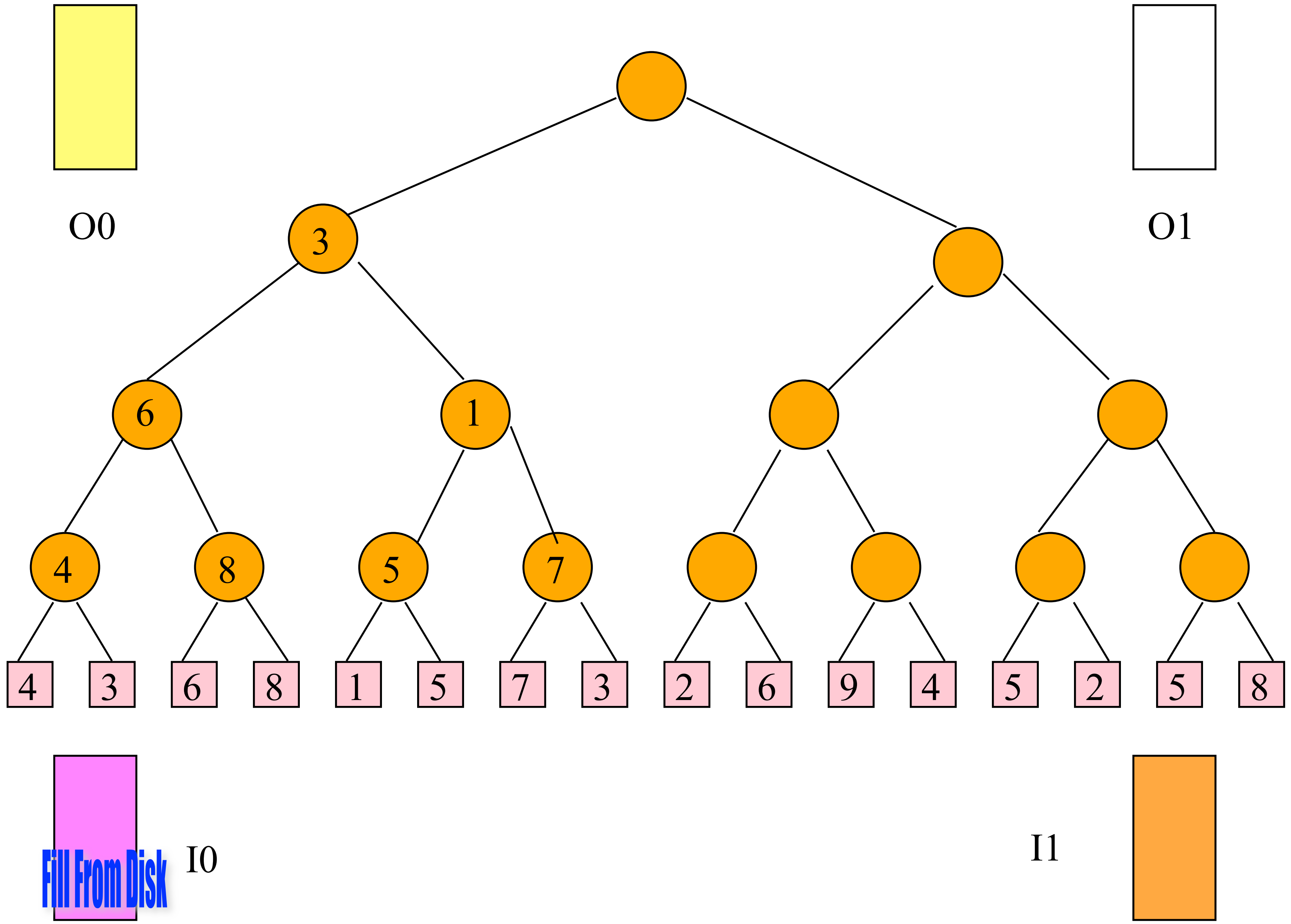


- Synchronization is done when the active input buffer gets empty (the active output buffer will be full at this time).

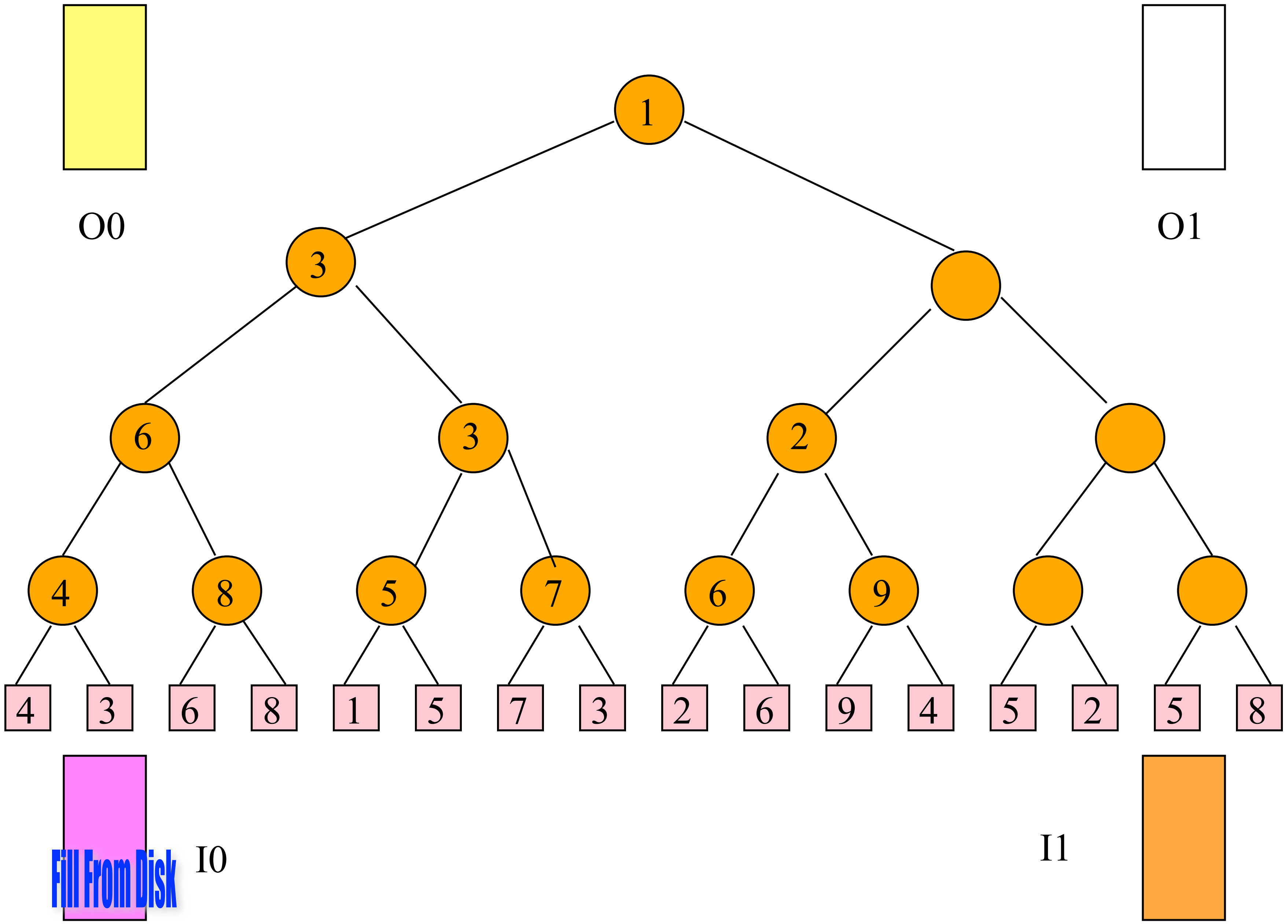
Initialize



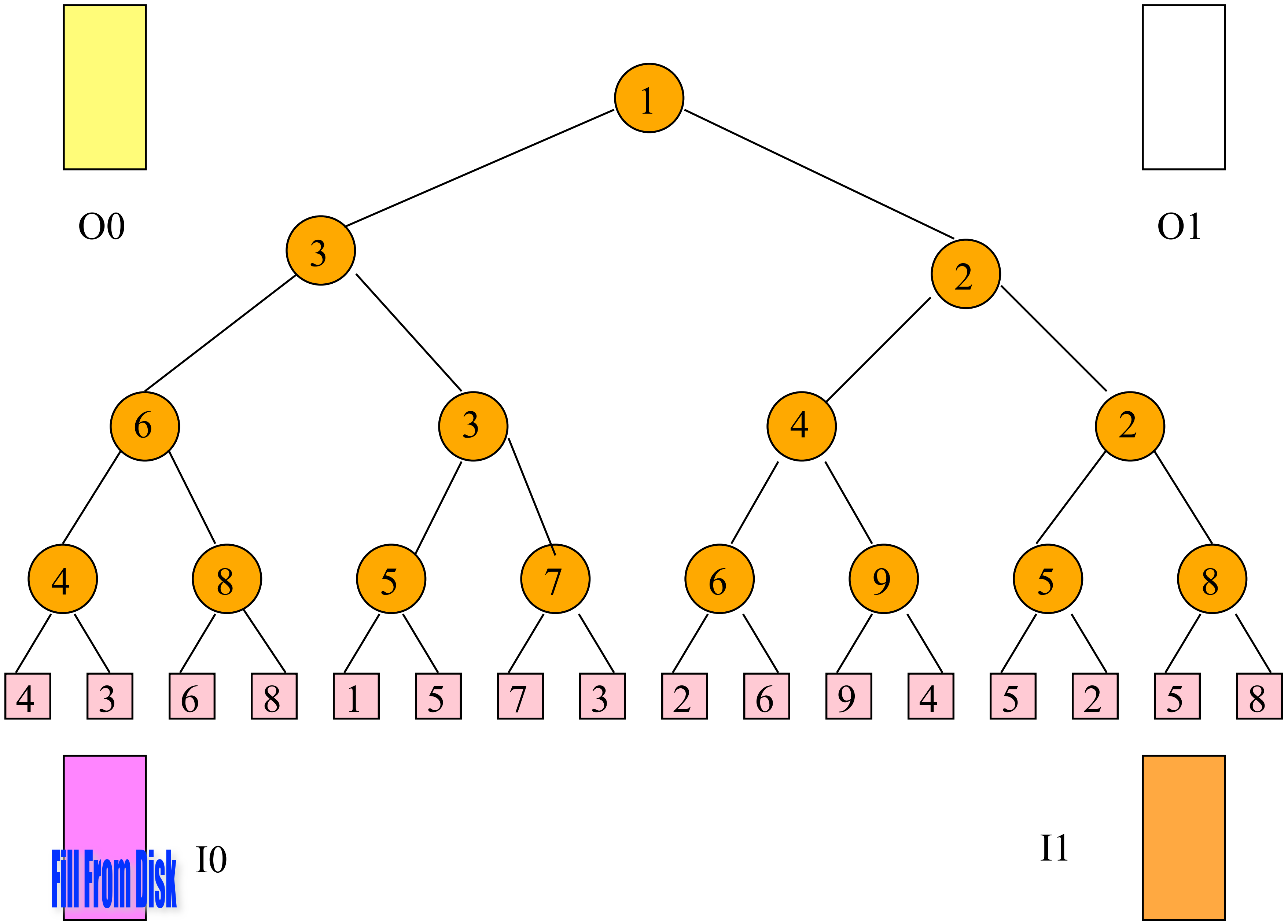
Initialize



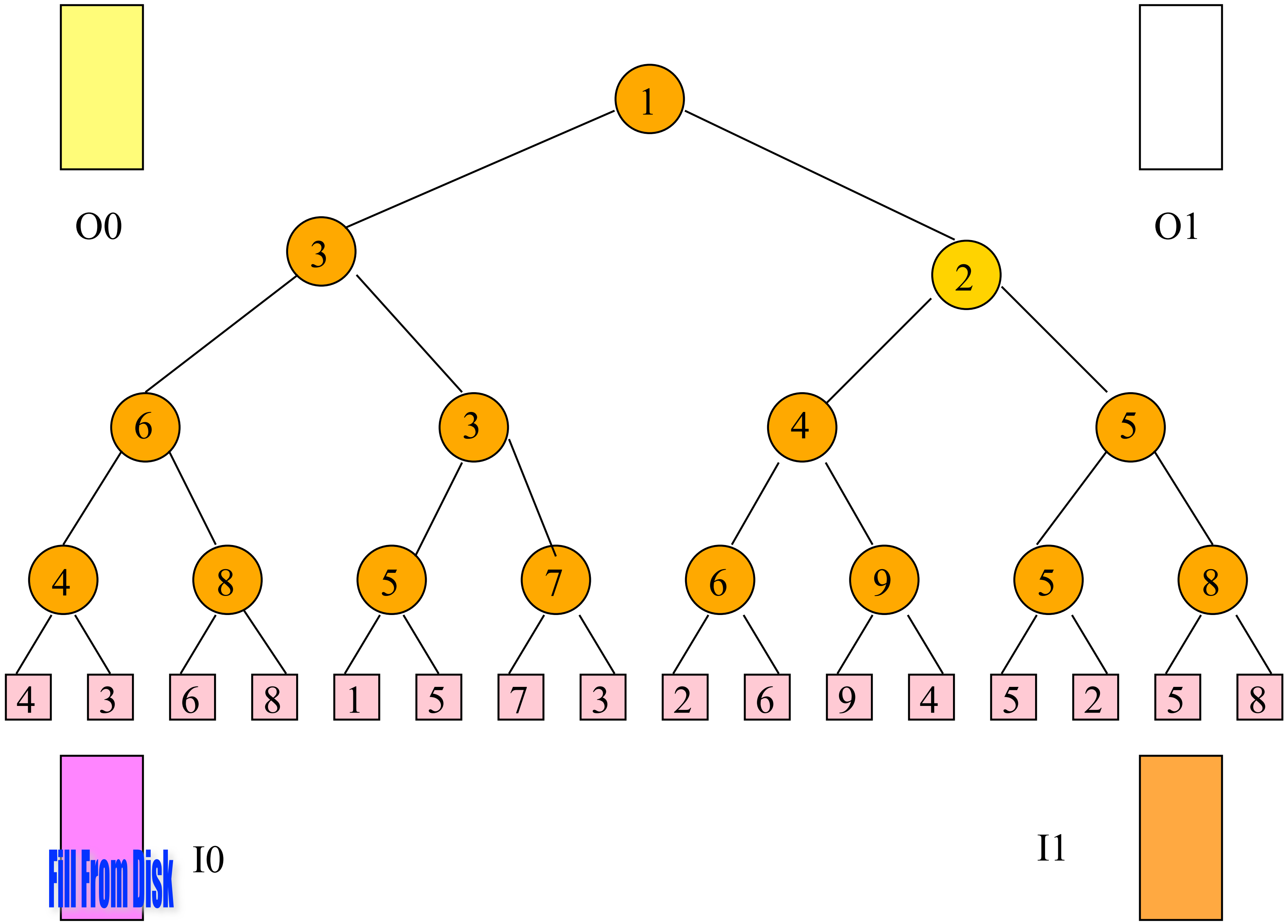
Initialize



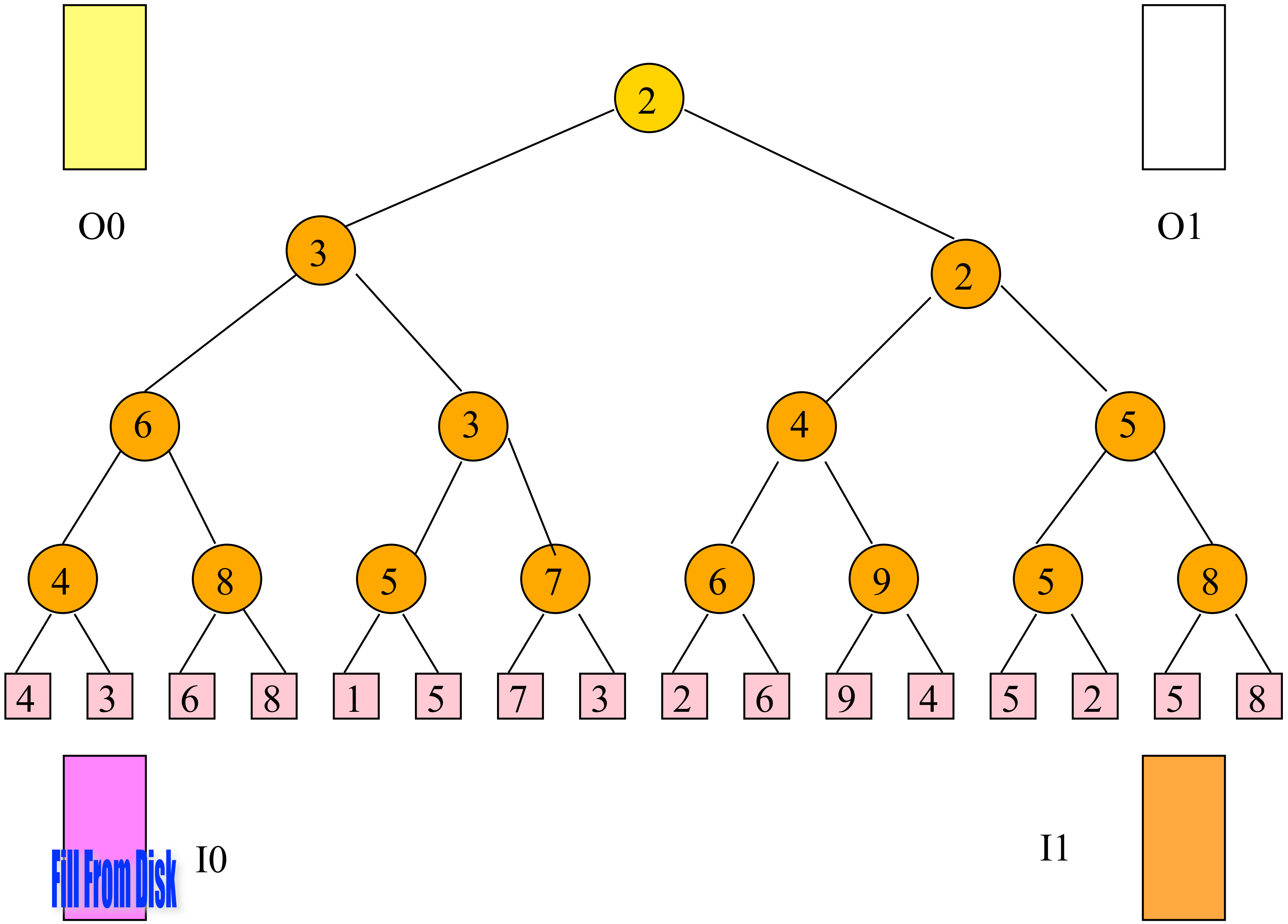
Initialize


O0
O1
1
3
2
6
3
4
2
4
8
5
7
6
9
5
8
4
3
6
8
1
5
7
3
2
6
9
4
5
2
5
8
Fill From Disk
I0
I1

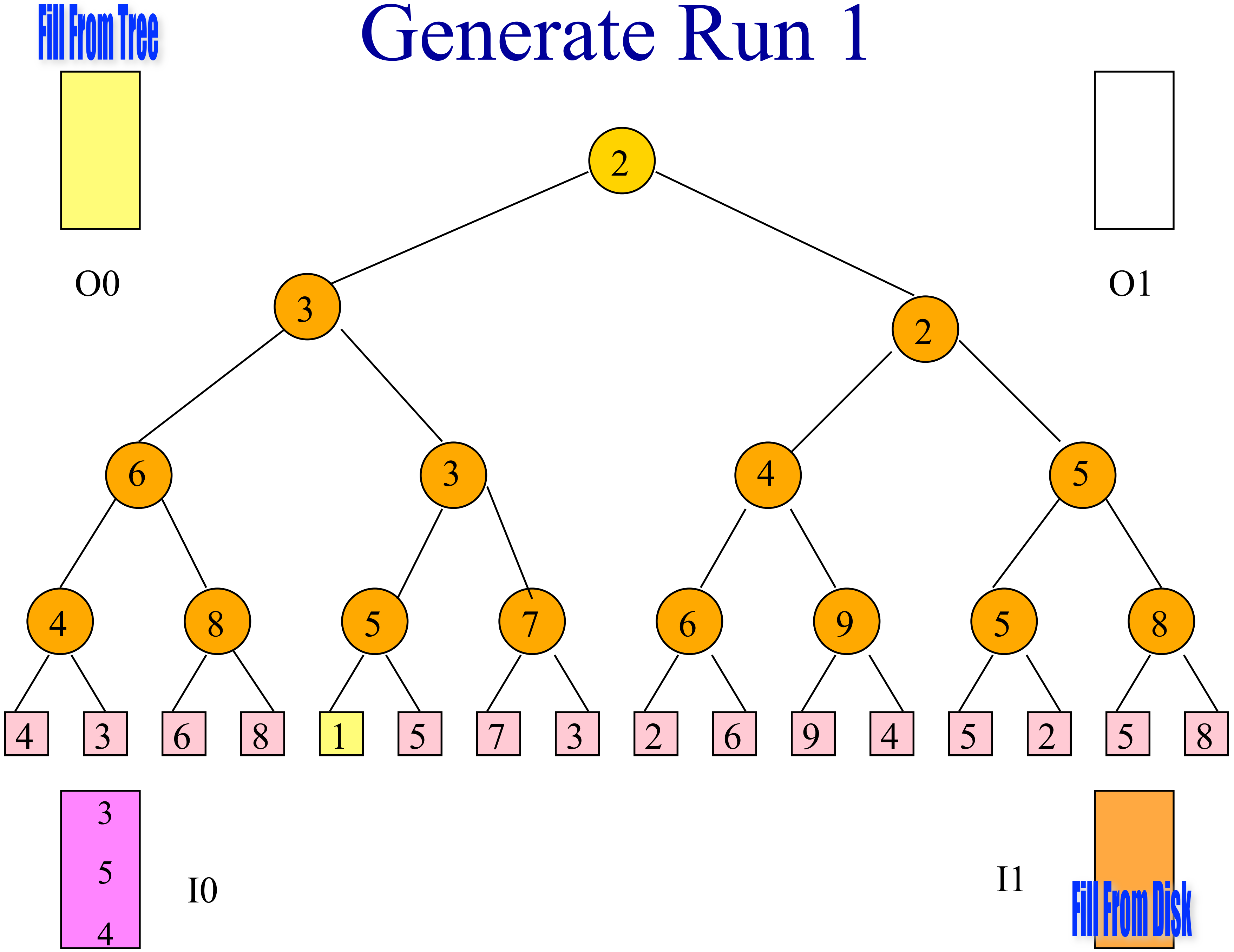
Initialize


O0
O1
1
3
2
6
3
4
5
4
8
5
7
6
9
5
8
4
3
6
8
1
5
7
3
2
6
9
4
5
2
5
8
Fill From Disk
I0
I1

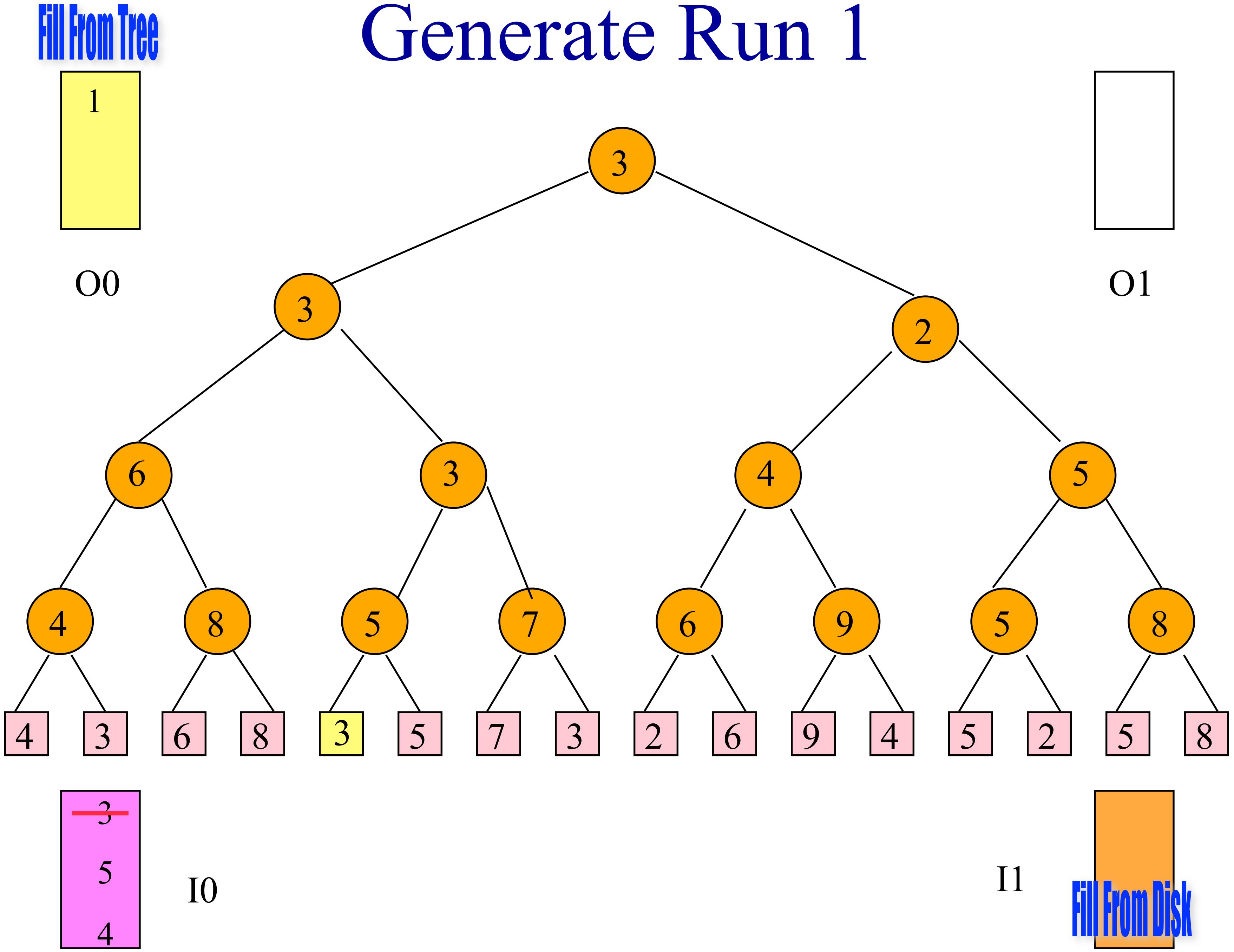
Initialize


O0
O1
2
3
2
6
3
4
5
4
8
5
7
6
9
5
8
4
3
6
8
1
5
7
3
2
6
9
4
5
2
5
8
Fill From Disk
I0
I1

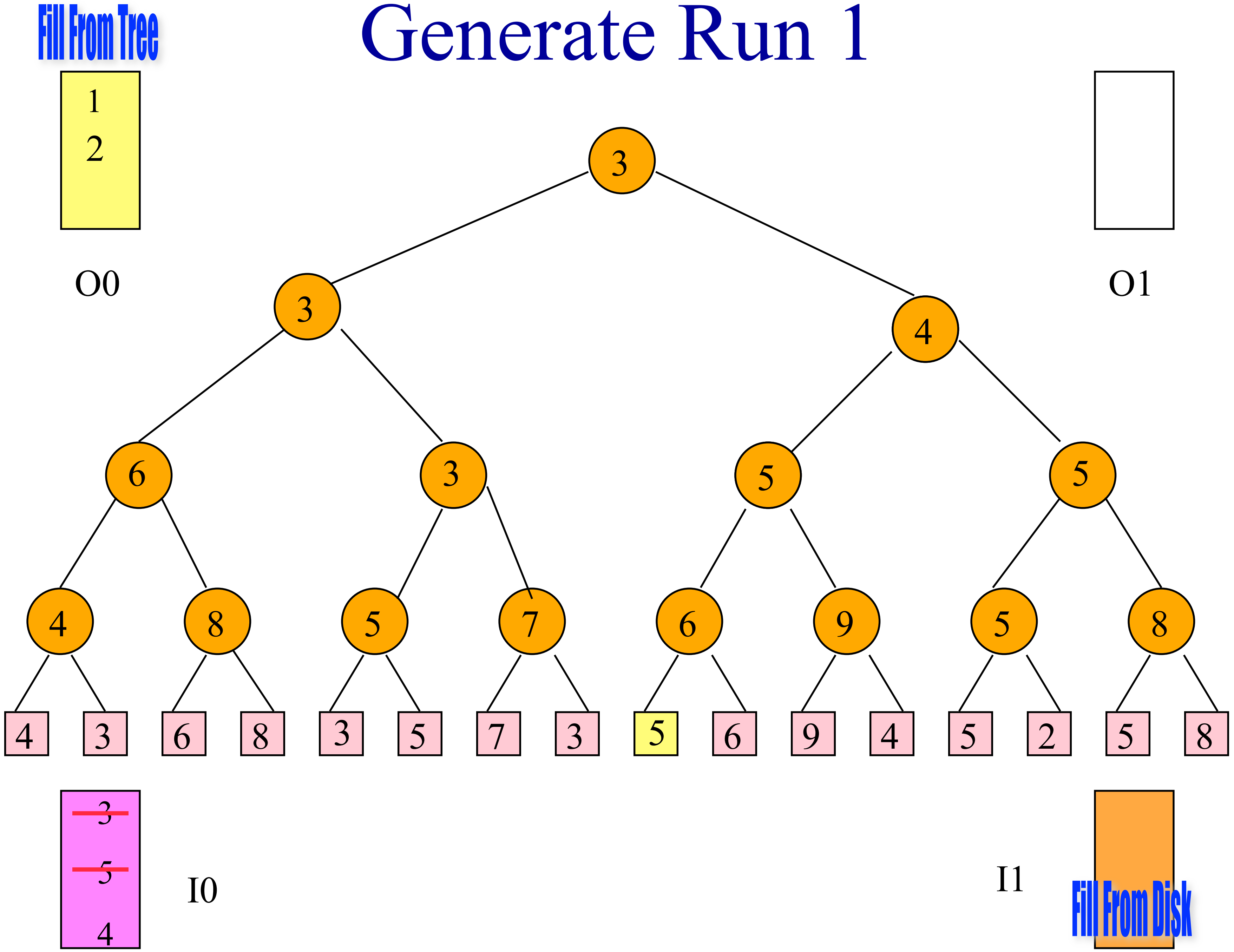
Generate Run 1



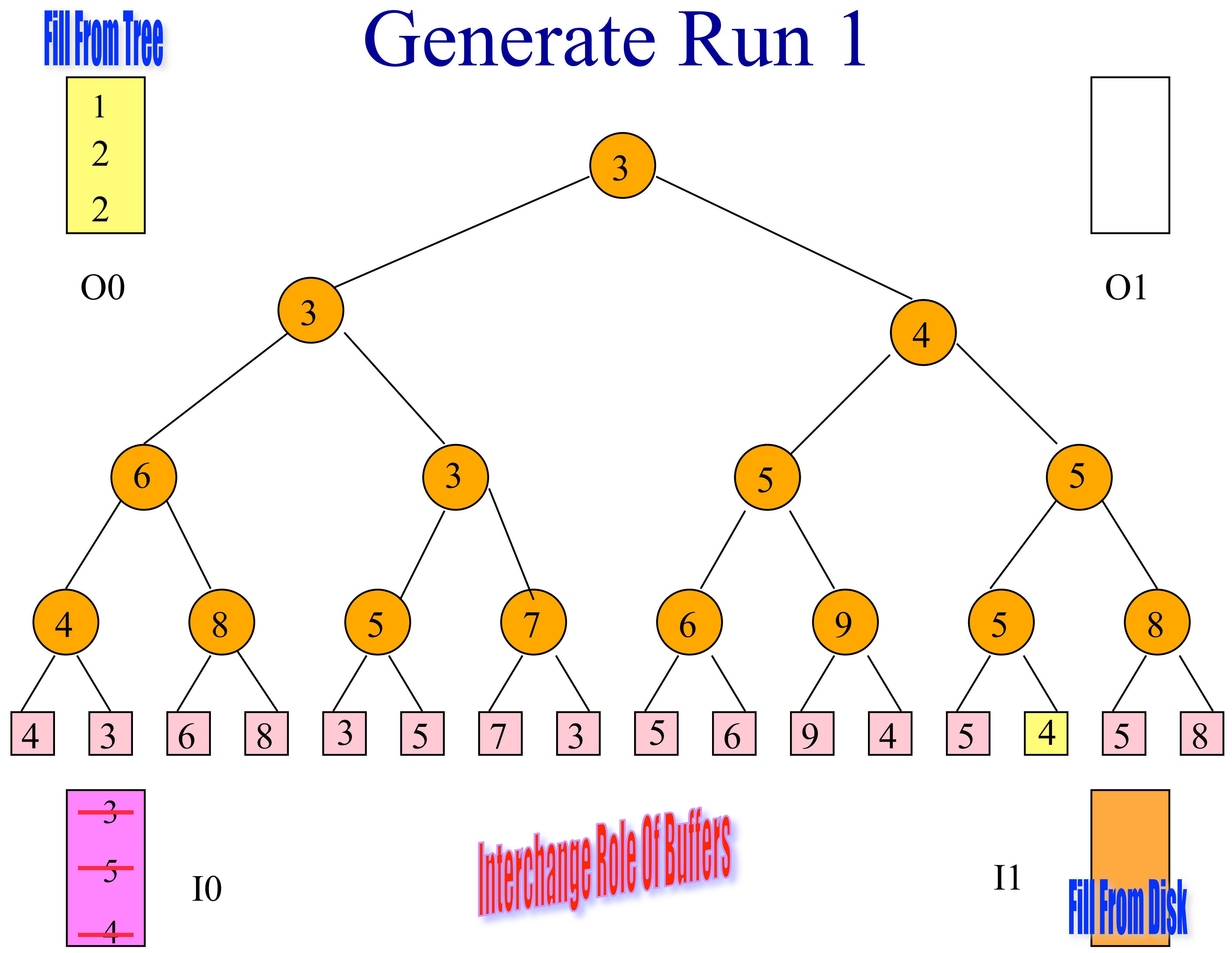
Generate Run 1



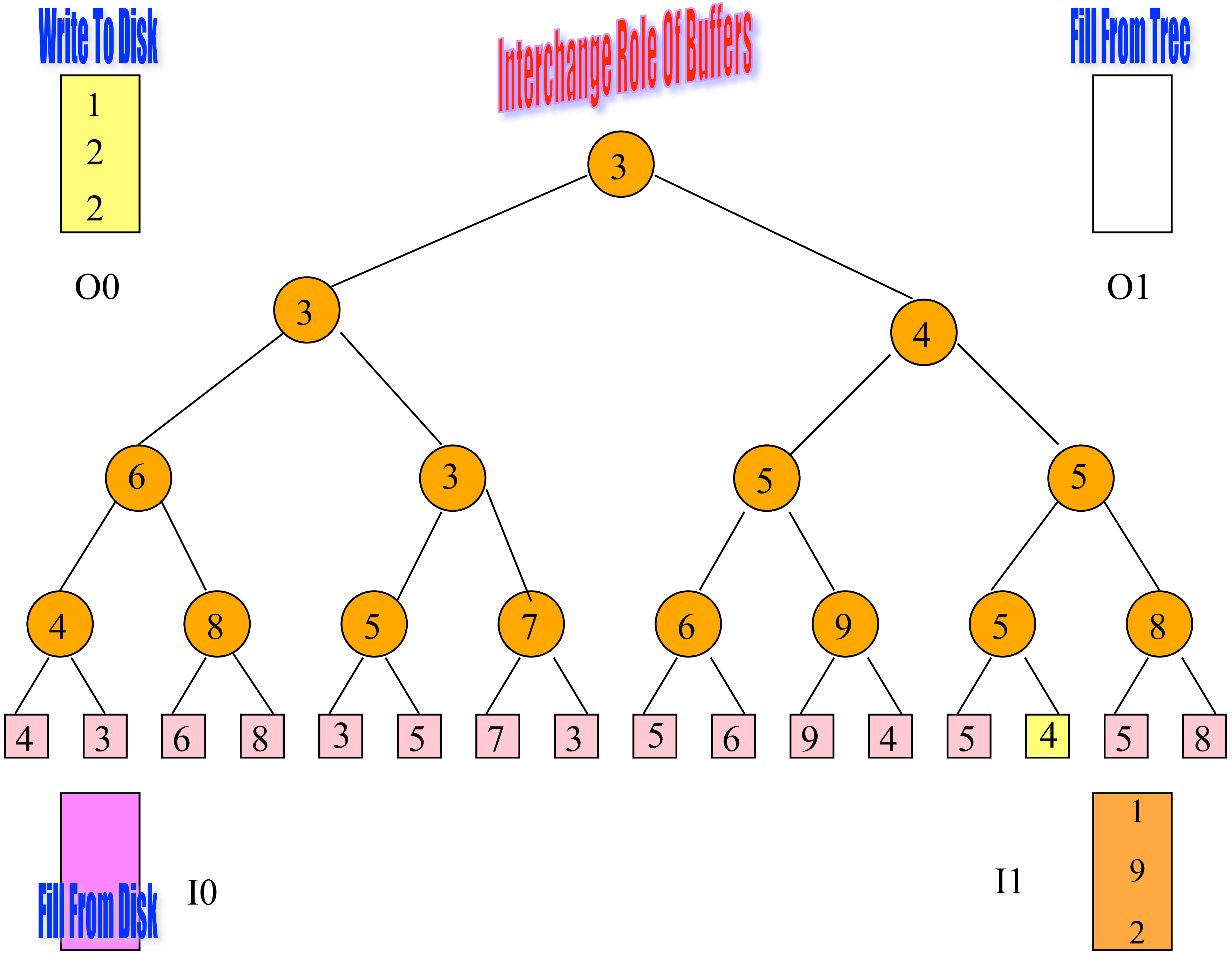
Generate Run 1



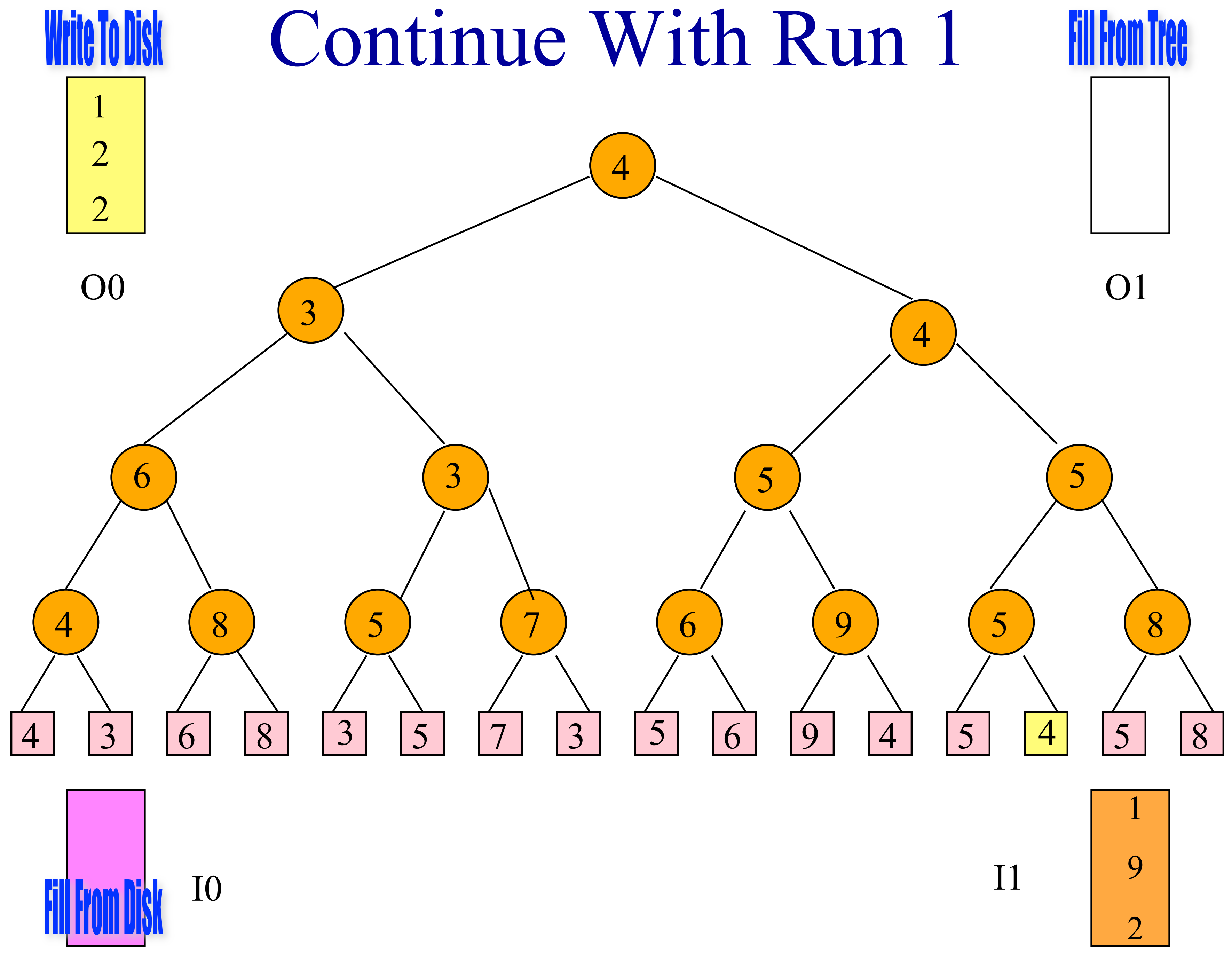
Generate Run 1



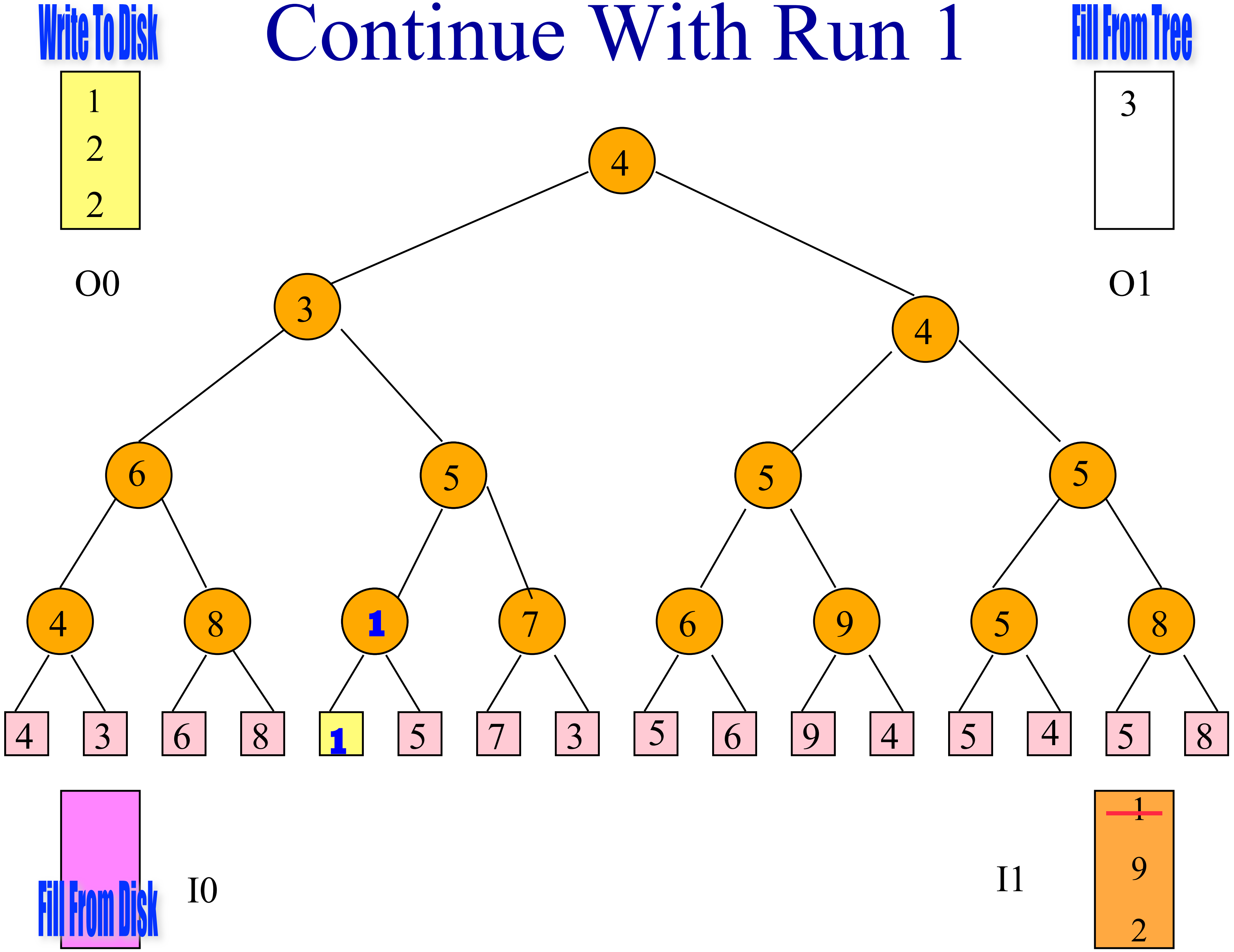
Interchange Role Of Buffers



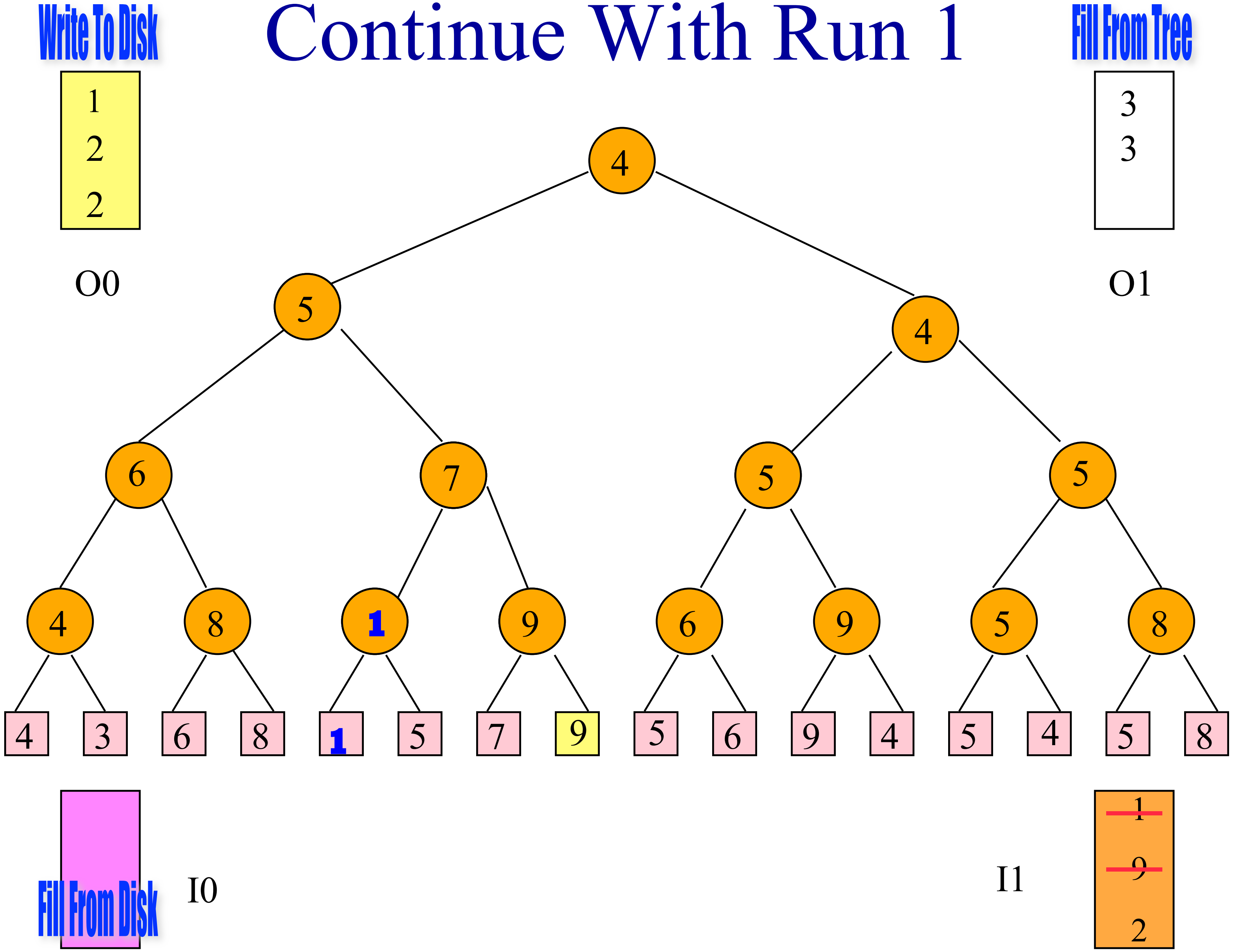
Continue With Run 1



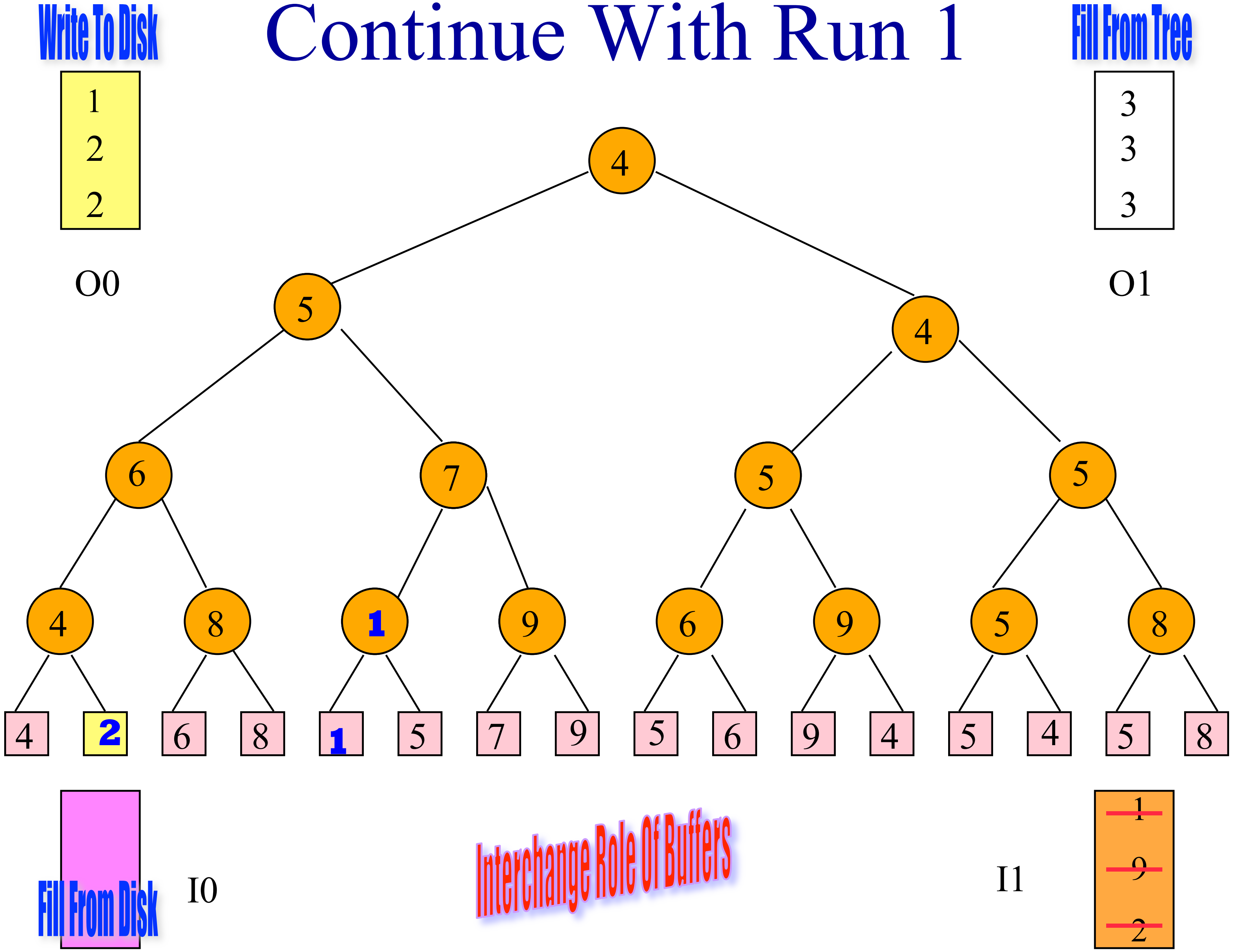
Continue With Run 1



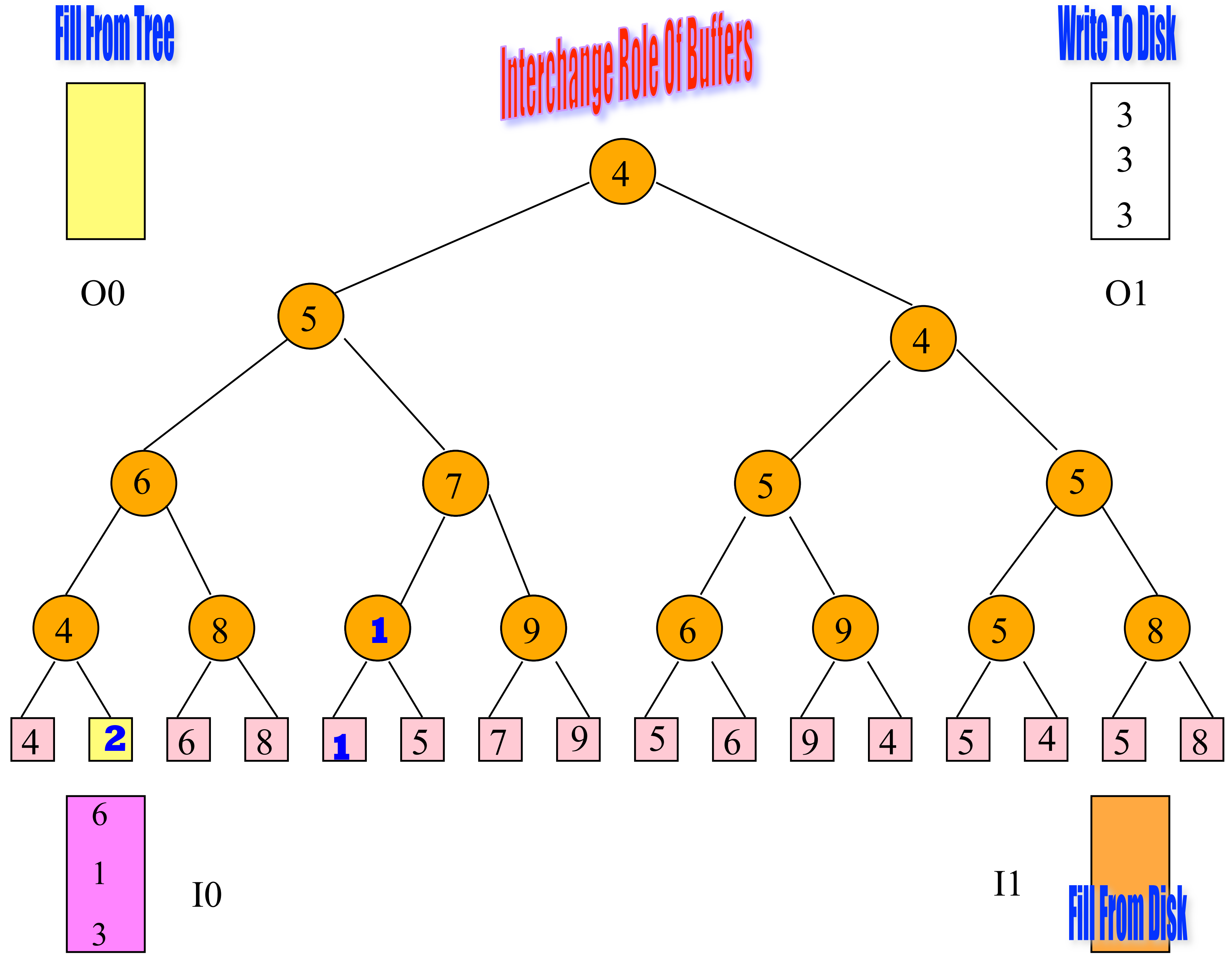
Continue With Run 1



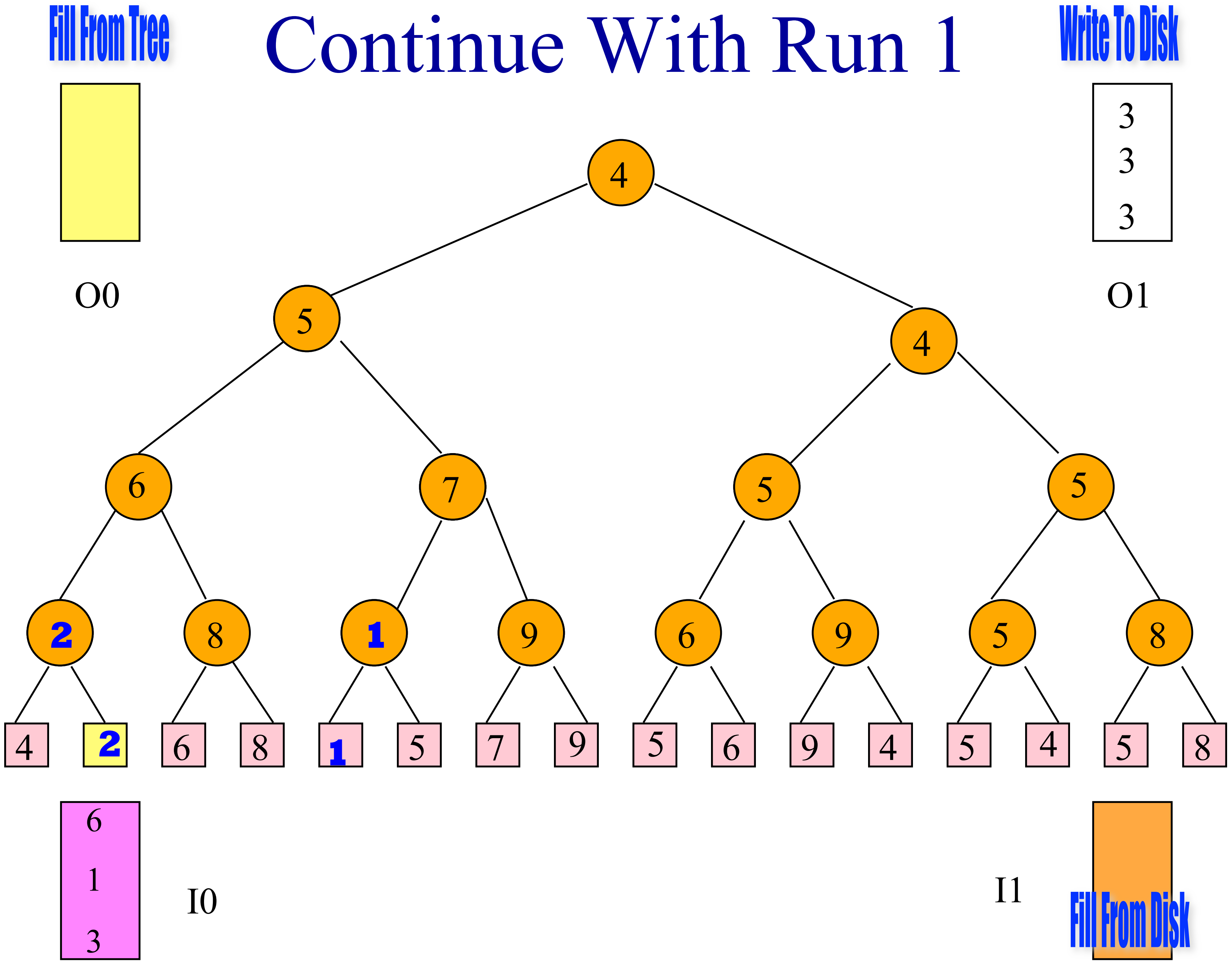
Continue With Run 1



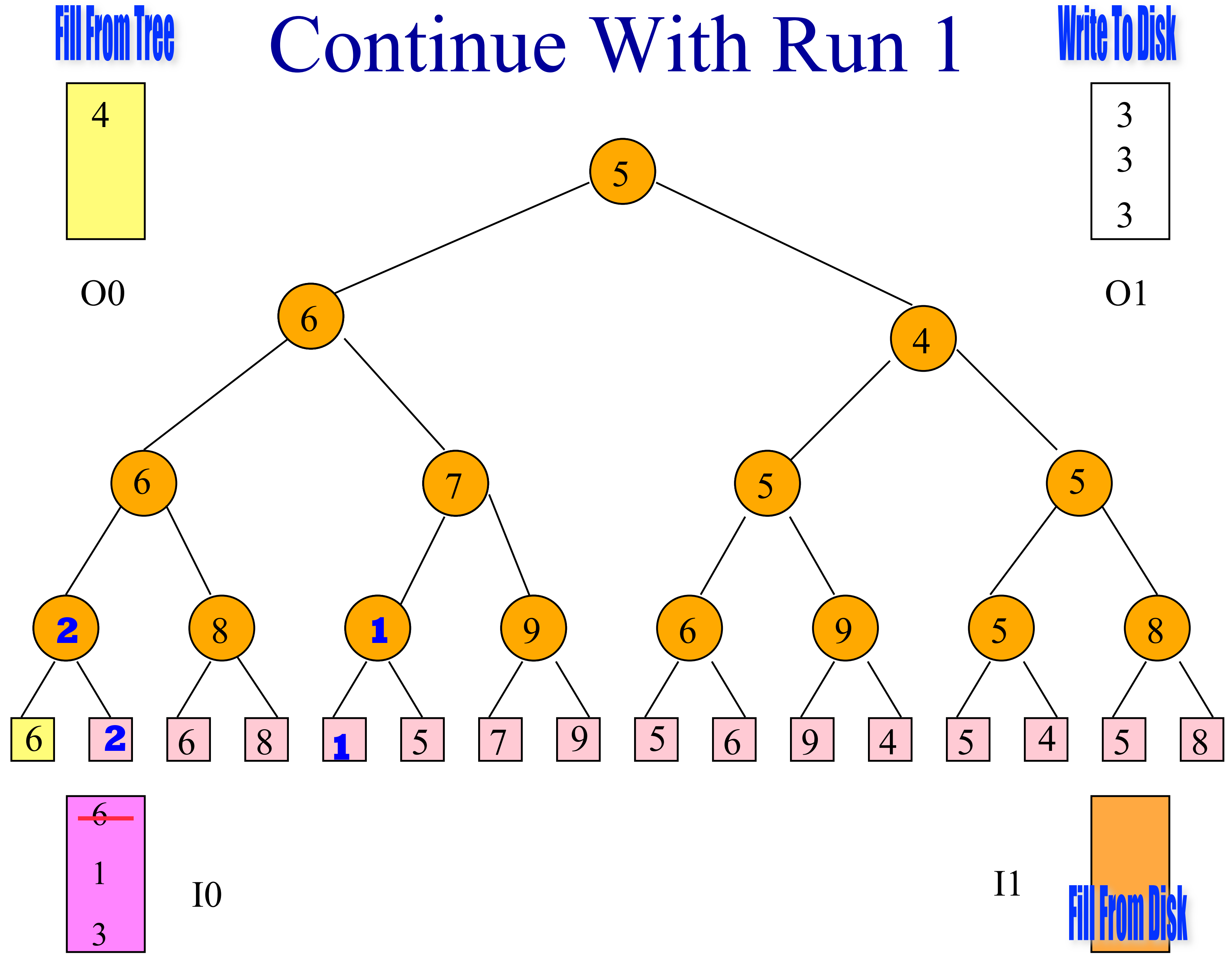
Interchange Role Of Buffers



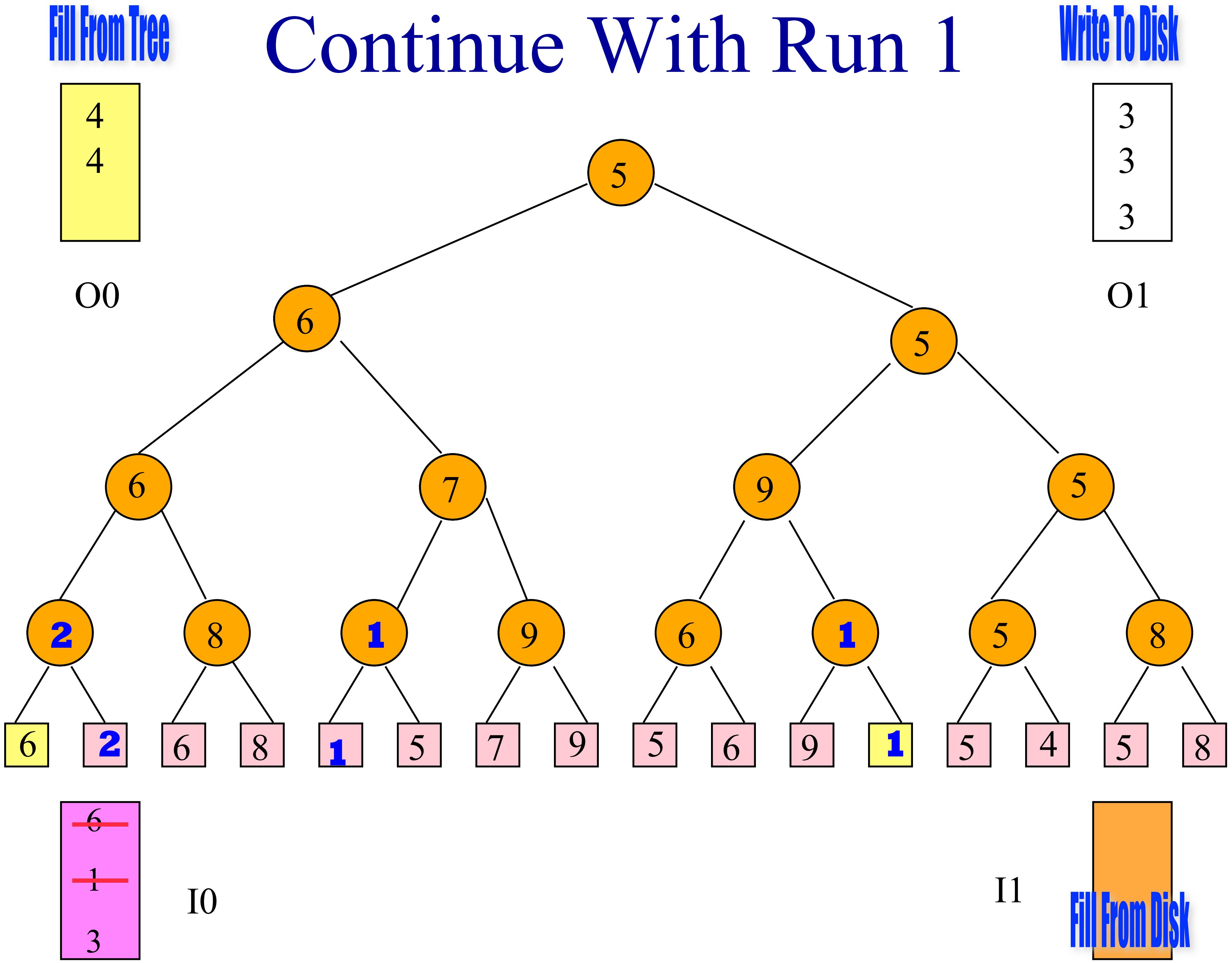
Continue With Run 1



Continue With Run 1



Continue With Run 1



RUN SIZE

- Let k be number of external nodes in loser tree.
- Run size >= k.
- Sorted input => 1 run.
- Reverse of sorted input => n/k runs.
- Average run size is ~2k.

Comparison

- Memory capacity $= m$ records.
- Run size using fill memory, sort, and output run scheme $= m$.
- Use loser tree scheme.
  - Assume block size is b records.
  - Need memory for 4 buffers ($4b$ records).
  - Loser tree $k = m - 4b$.
  - Average run size $= 2k = 2(m - 4b)$.
  - $2k >= m$ when $m >= 8b$.

Comparison

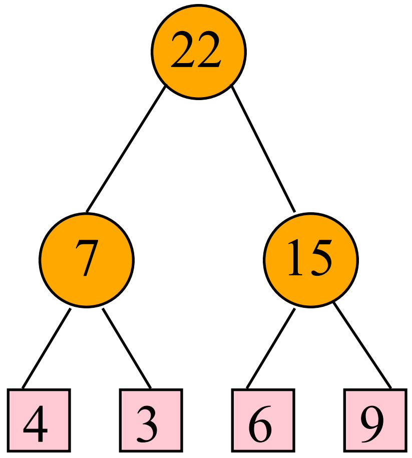
- Assume b = 100.

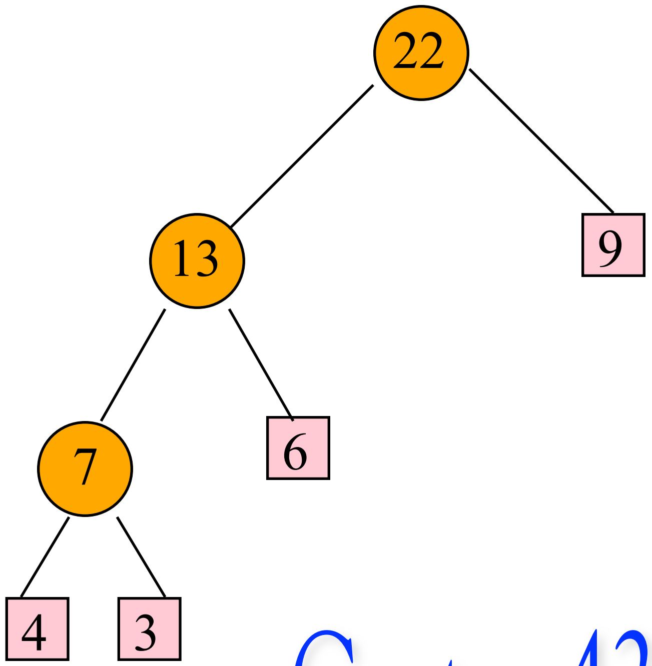
m	600	1000	5000	10000
k	200	600	4600	9600
2k	400	1200	9200	19200

Comparison

- Total internal processing time using fill memory, sort, and output run scheme = O((n/m) m log m) = O(n log m).
- Total internal processing time using loser tree = O(n log k).
- Loser tree scheme generates runs that differ in their lengths.

Merging Runs Of Different Length





Cost = 44

Cost = 42

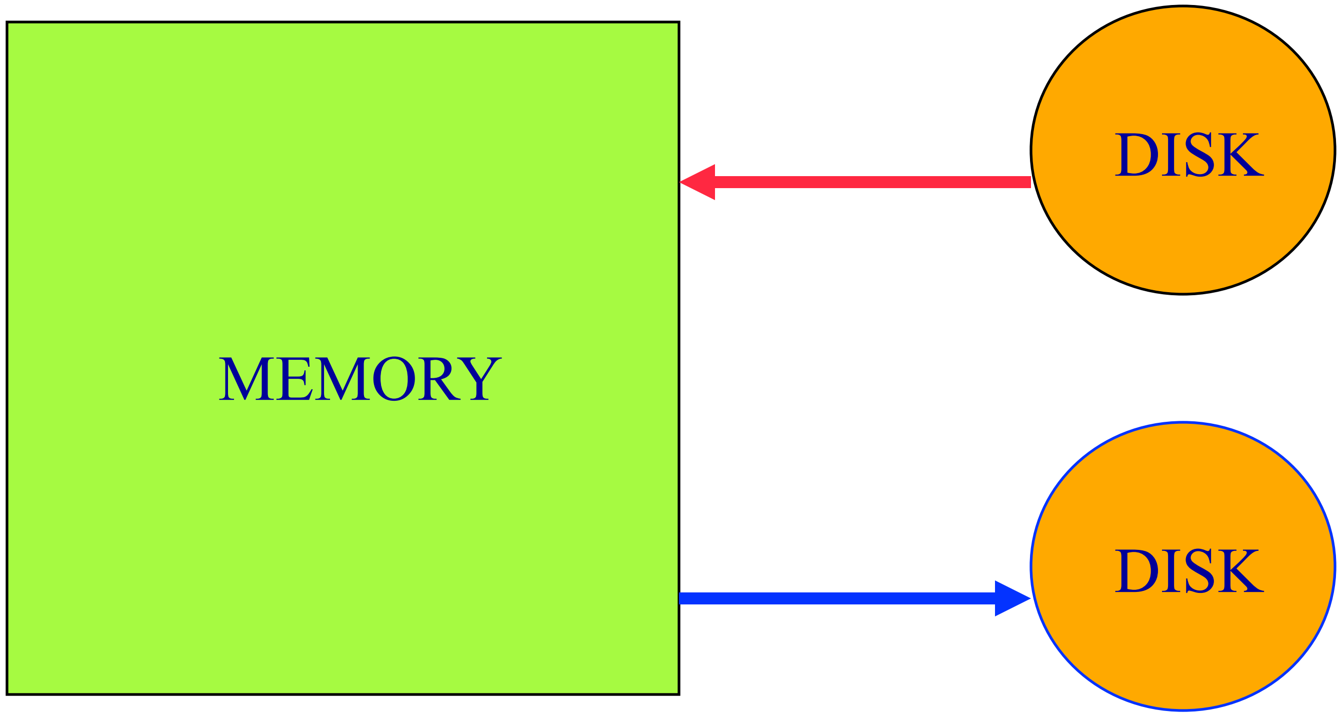
Best merge sequence?

Improve Run Merging

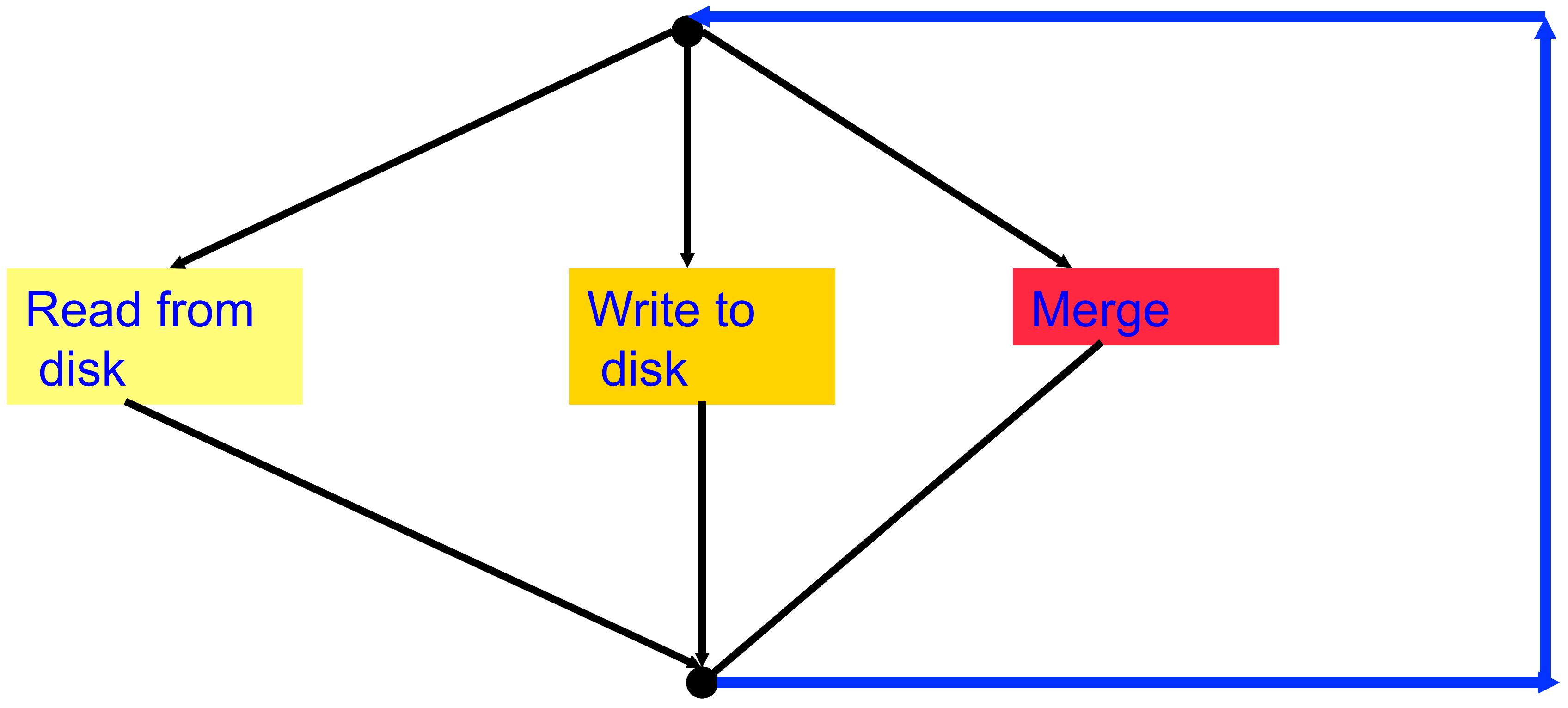
- Reduce number of merge passes.
  - Use higher order merge.
  - Number of passes $= \text{ceil}(\log_k(\text{number of initial runs}))$ where k is the merge order.
- More generally, a higher-order merge reduces the cost of the optimal merge tree.

Improve Run Merging

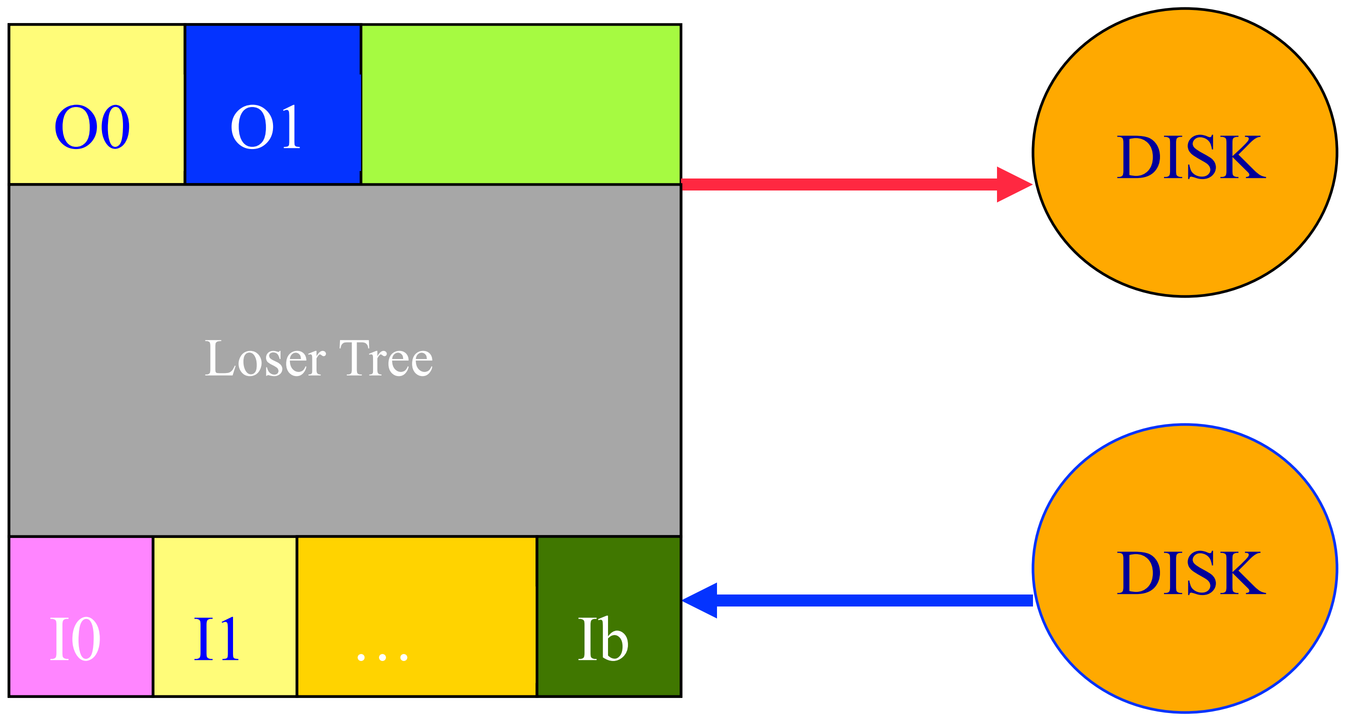
- Overlap input, output, and internal merging.



Steady State Operation



Partitioning Of Memory



- Need exactly 2 output buffers.
- Need at least k+1 (k is merge order) input buffers.
- 2k input buffers suffice.

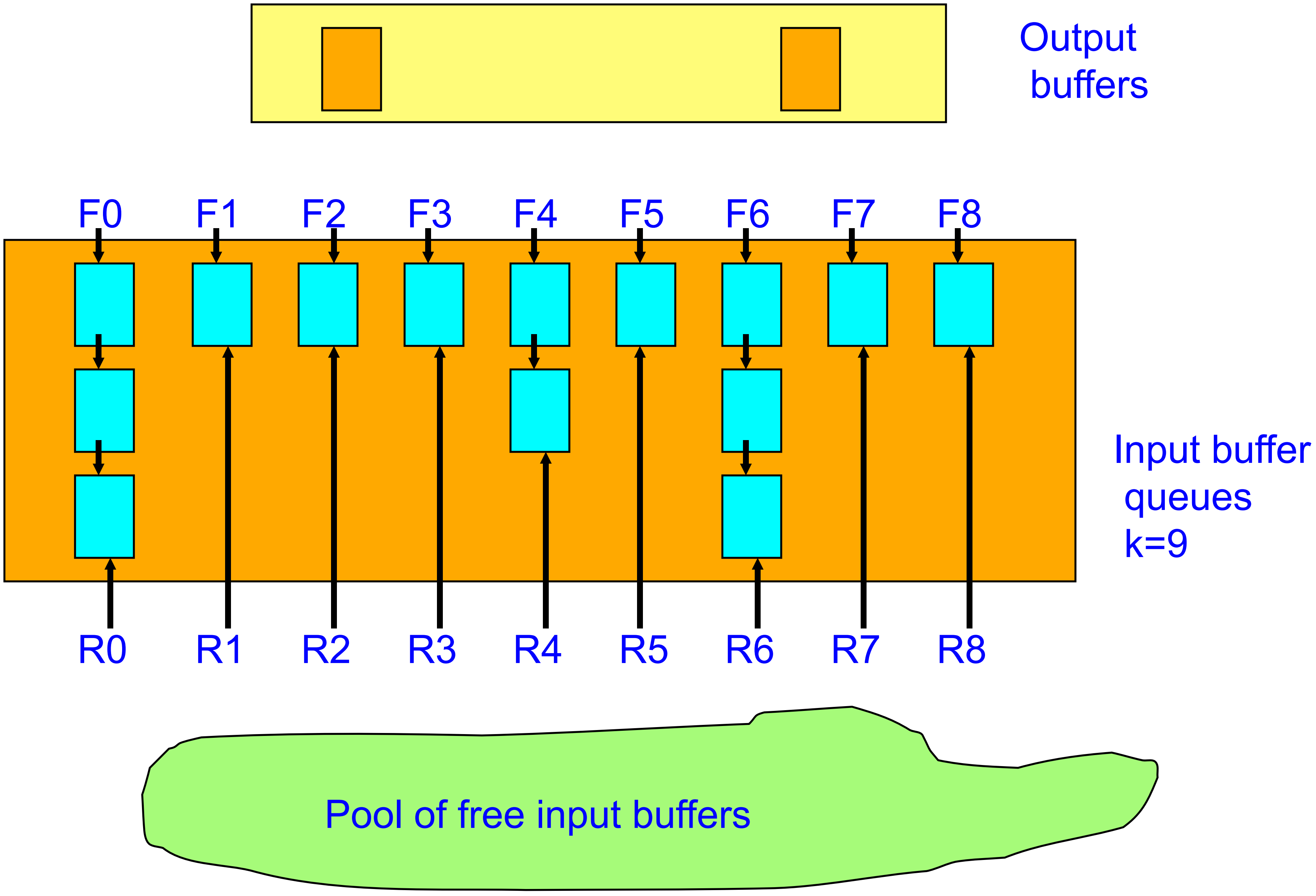
Number Of Input Buffers

- When 2 input buffers are dedicated to each of the k runs being merged, 2k buffers are not enough!
- Input buffers must be allocated to runs on an as needed basis.

Buffer Allocation

- When ready to read a buffer load, determine which run will exhaust first.
  - Examine key of the last record read from each of the k runs.
  - Run with smallest last key read will exhaust first.
- Next buffer load of input is to come from run that will exhaust first, allocate an input buffer to this run.

Buffer Layout



Initialize To Merge k Runs

- Initialize k queues of input buffers, 1 queue per run, 1 buffer per run.
- Input one buffer load from each of the k runs.
- Put k – 1 unused input buffers into pool of free buffers.
- Set activeOutputBuffer = 0.
- Initiate input of next buffer load from first run to exhaust. Use remaining unused input buffer for this input.

The Method kWayMerge

- k-way merge from input queues to the active output buffer.
- Merge stops when either the output buffer gets full or when an end-of-run key is merged into the output buffer.
- If merge hasn’t stopped and an input buffer gets empty, advance to next buffer in queue and free empty buffer.

Merge k Runs

```
repeat
    kWayMerge;
    wait for input/output to complete;
    add new input buffer (if any) to queue for its run;
    determine run that will exhaust first;
    if (there is more input from this run)
        initiate read of next block for this run;
    initiate write of active output buffer;
    activeOutputBuffer = 1 – activeOutputBuffer;
until end-of-run key merged;
```

What Can Go Wrong?

kWayMerge

- k-way merge from input queues to the active output buffer.
- Merge stops when either the output buffer gets full or when an end-of-run key is merged into the output buffer.
- If merge hasn't stopped and an input buffer gets empty, advance to next buffer in queue and free empty buffer. There may be no next buffer in the queue.

What Can Go Wrong?

Merge k Runs

```
repeat
  kWayMerge;
  wait for input/output to complete;
  add new input buffer (if any) to queue for its run;
  determine run that will exhaust first;
  if (there is more input from this run)
    initiate read of next block for this run;
  initiate write of active output buffer;
  activeOutputBuffer = 1 – activeOutputBuffer;
until end of run key merged;
```

There may be no free input buffer to read into.

Initializing For Next k-way Merge

Change

```
if (there is more input from this run)
        initiate read of next block for this run;
```

to

```
if (there is more input from this run)
        initiate read of next block for this run;
else
        initiate read of a block for the next k-way merge;
```