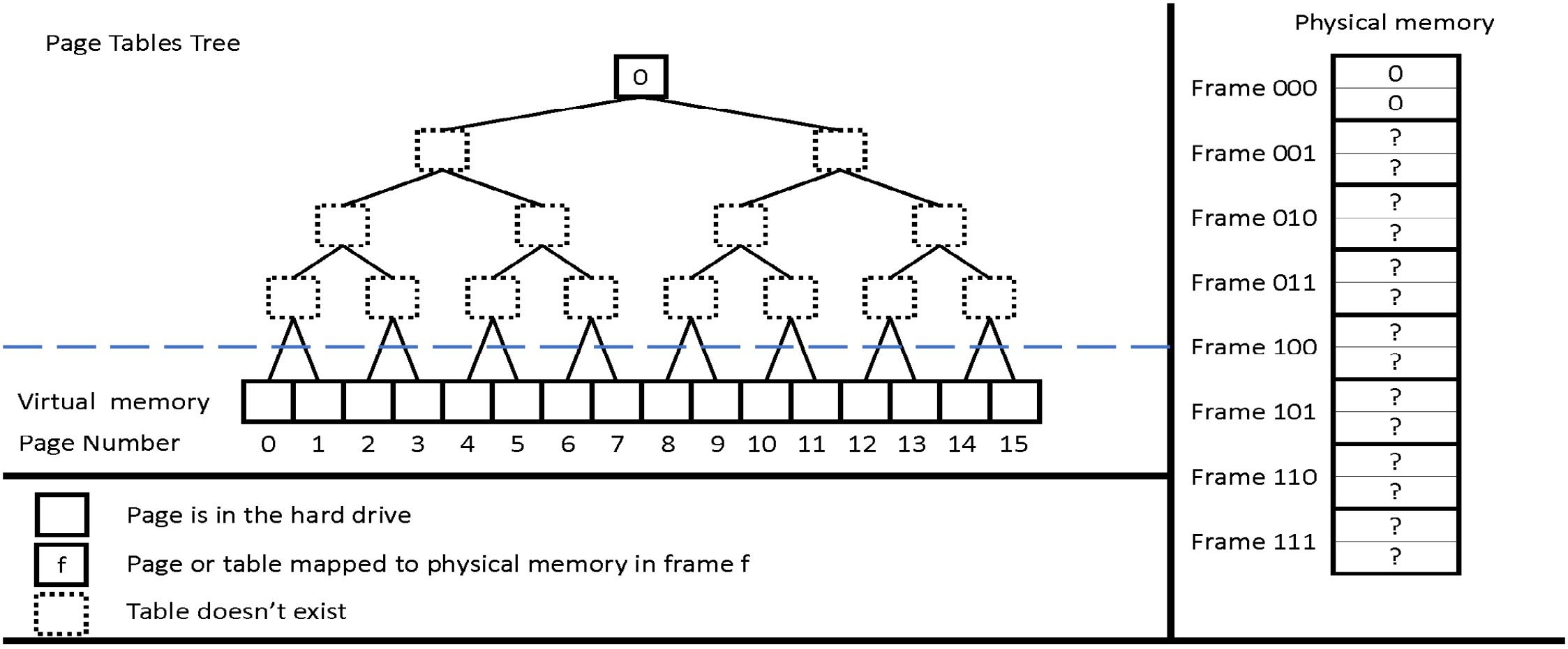
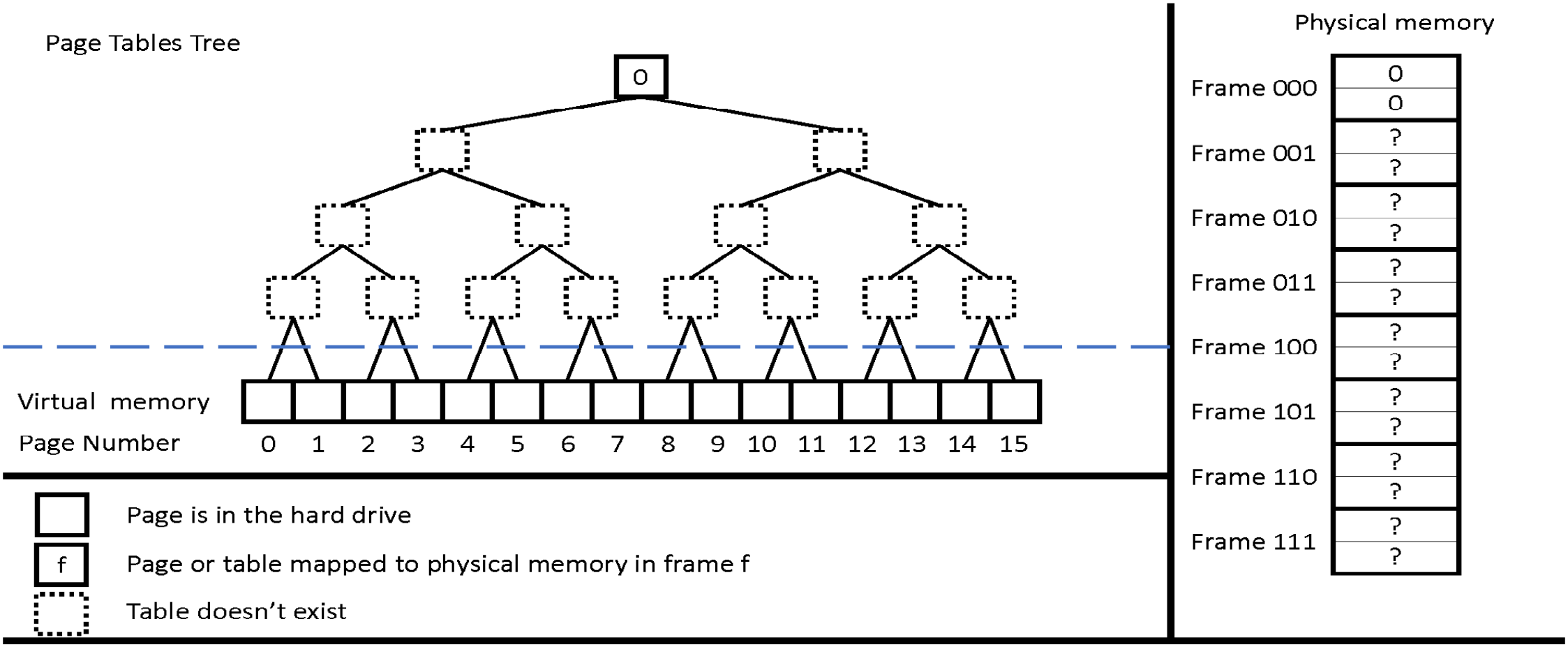
WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



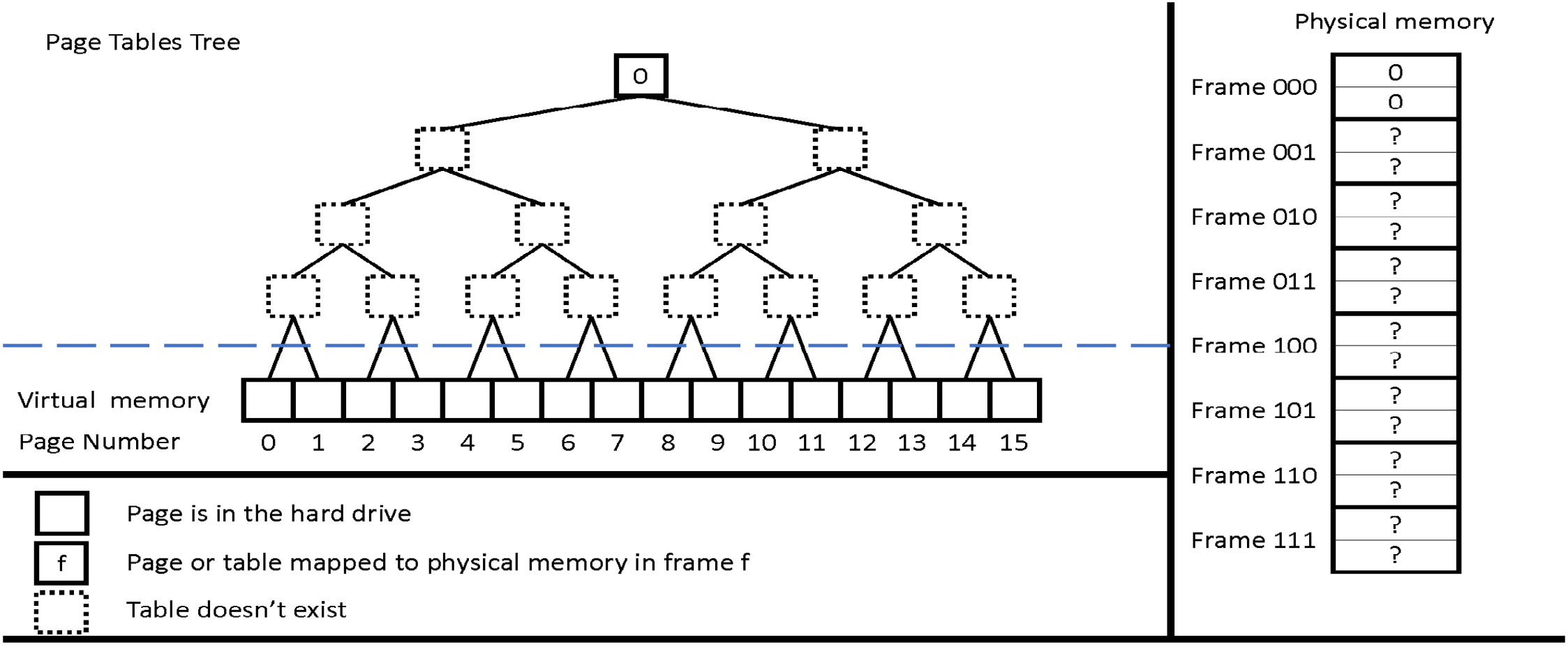
Virtual memory size is $32=2^5$, page/frame/table size is $2=2^1$, therefore we need ceil((5-1)/1) = 4 layers in the tree.
A single page table for all pages would require 16 rows, but we only have 16 words in our entire physical memory!
That's why we must use a hierarchical page table.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



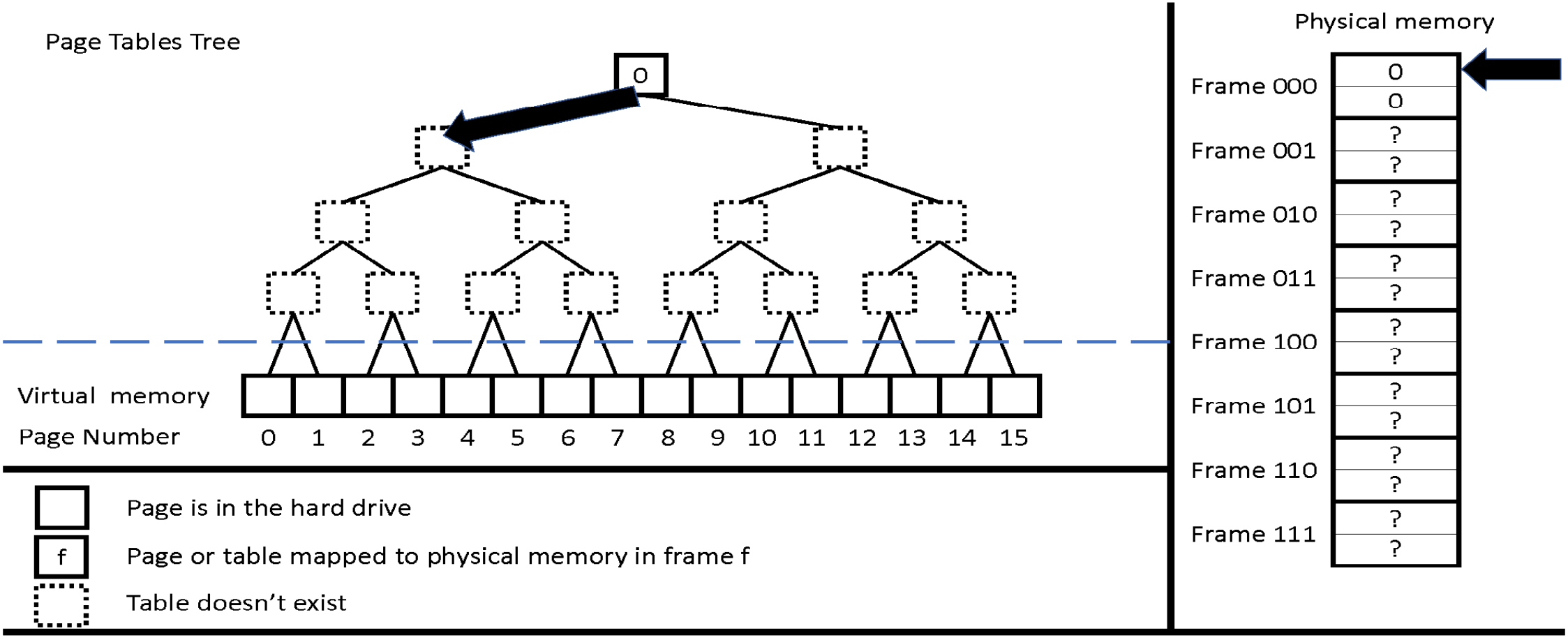
After initialization, the physical memory only contains the root table. It will always be in frame 0.
Both its rows are now 0, as no other table exists yet.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



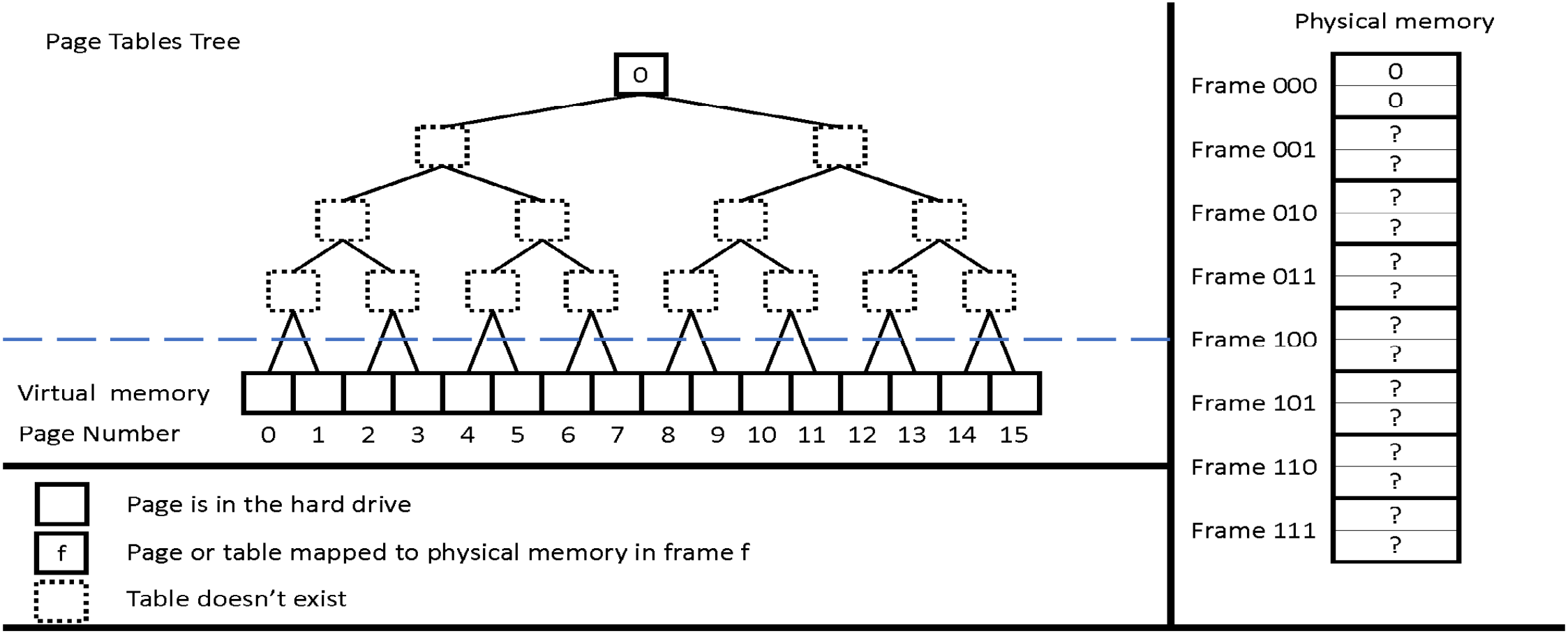
Let's now run VMwrite(13, 3).
13 is 01101, offset width is 1, so we want to write 3 into word 1 of page 6.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



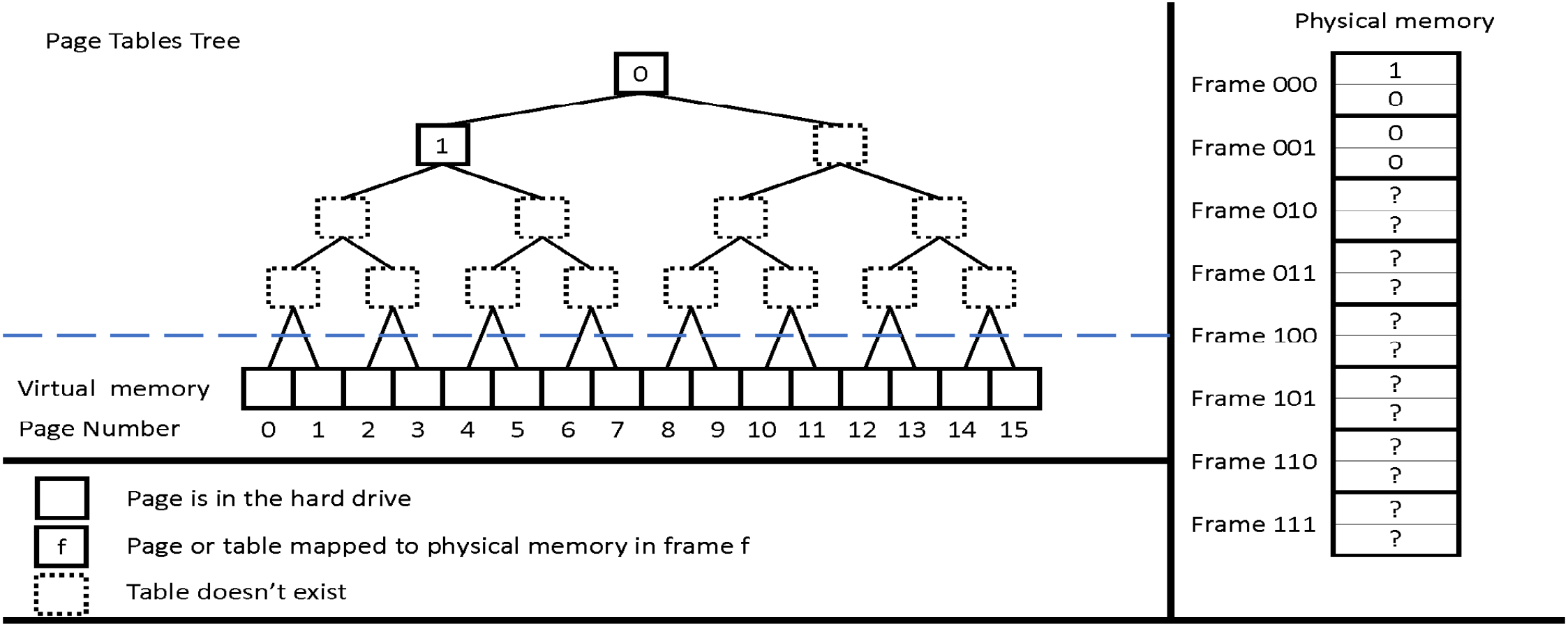
We have to first find page 6, we do it by traversing the tree according to the address 0110.
The root is in frame 0 and the first part of the address is 0, we read the RAM in address 0 + 0 = 0.
Since its content is zero, we know that the table below it is not in the RAM and we must create it.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



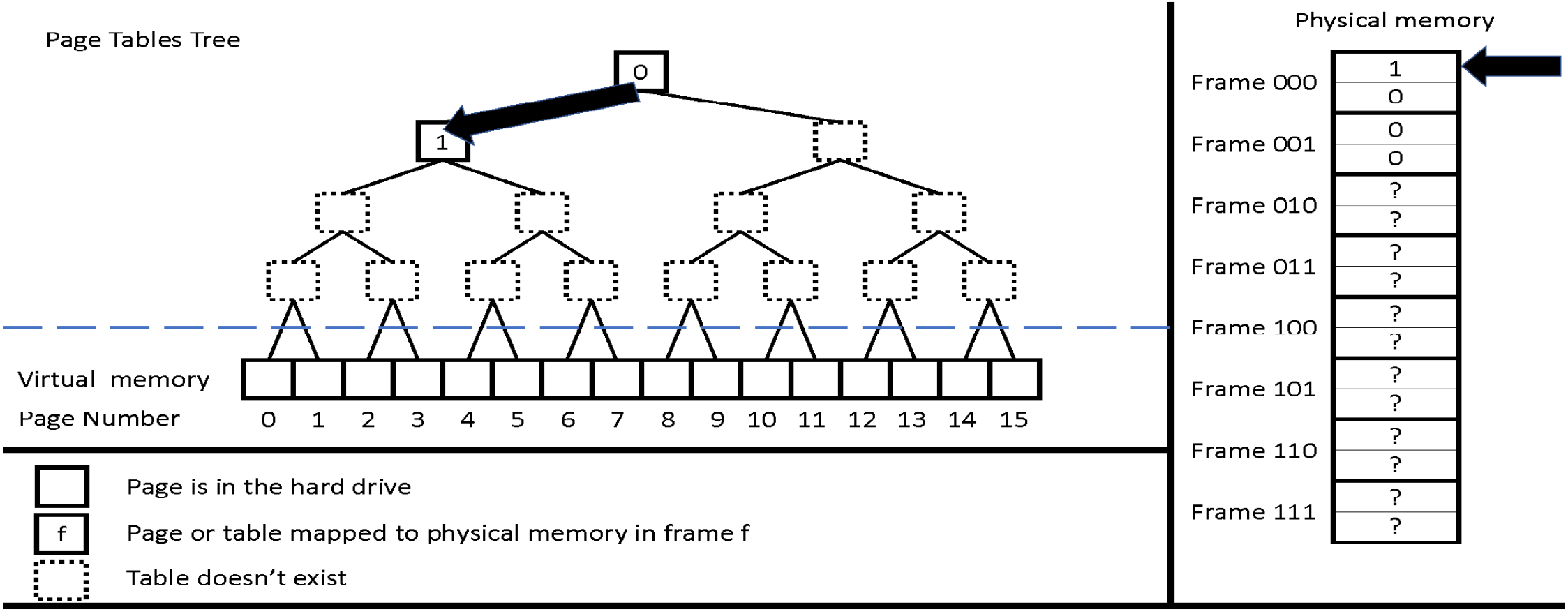
In order to create a new table, we need a frame. We find one by traversing the entire tree in DFS.
This is done by going to the root table (always in frame 0), iterating over its rows, and recursively entering every entry that isn't 0.
This time the traversal we only saw one frame during the traversal – 0, therefore we know that frame 1 is unused.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



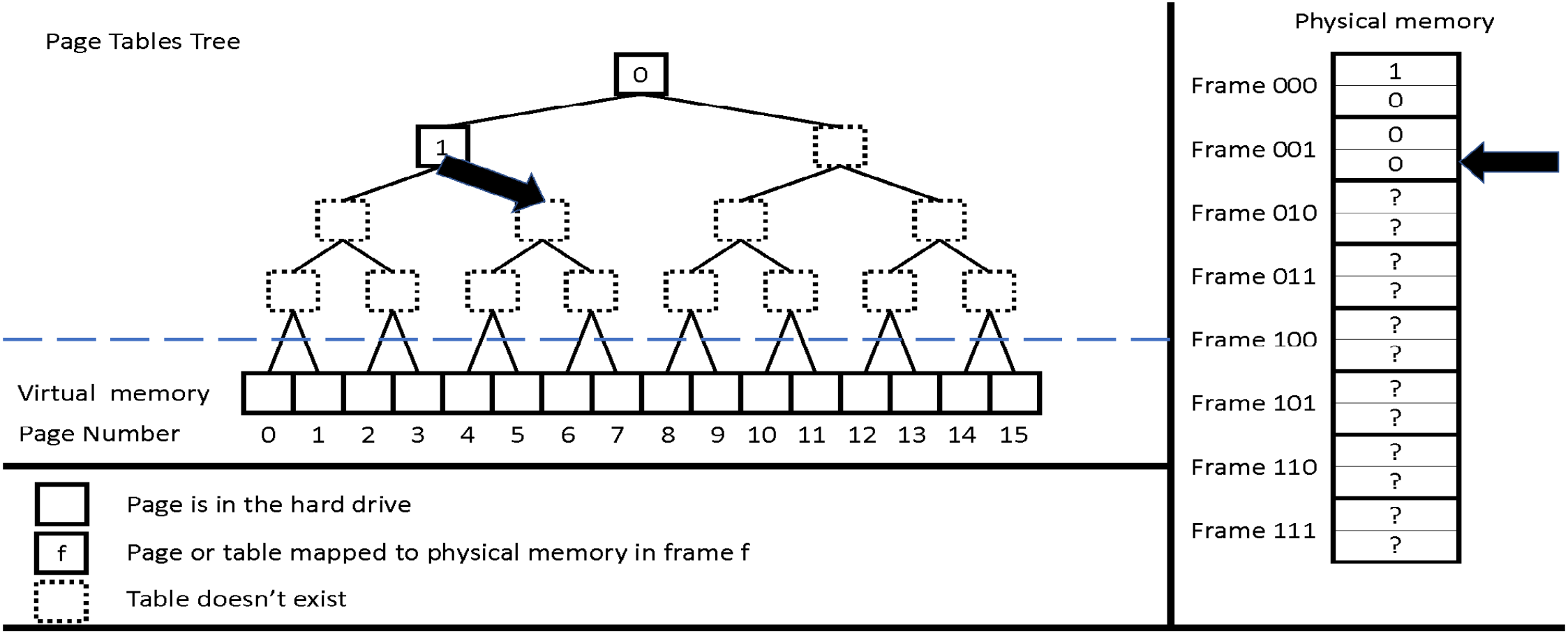
We map our new table to frame 1 and fill it with zeros. We also link it from the root table.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



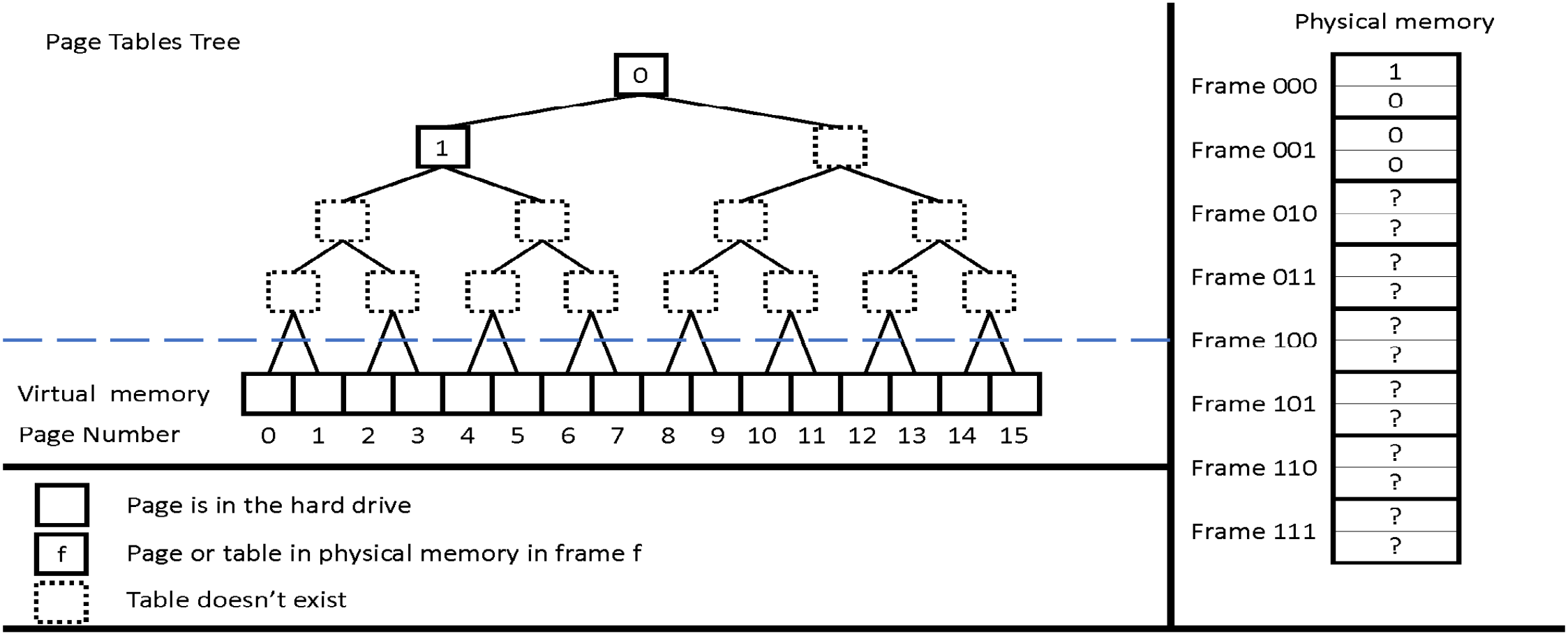
Now that the table exists, we continue down to address 0110.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



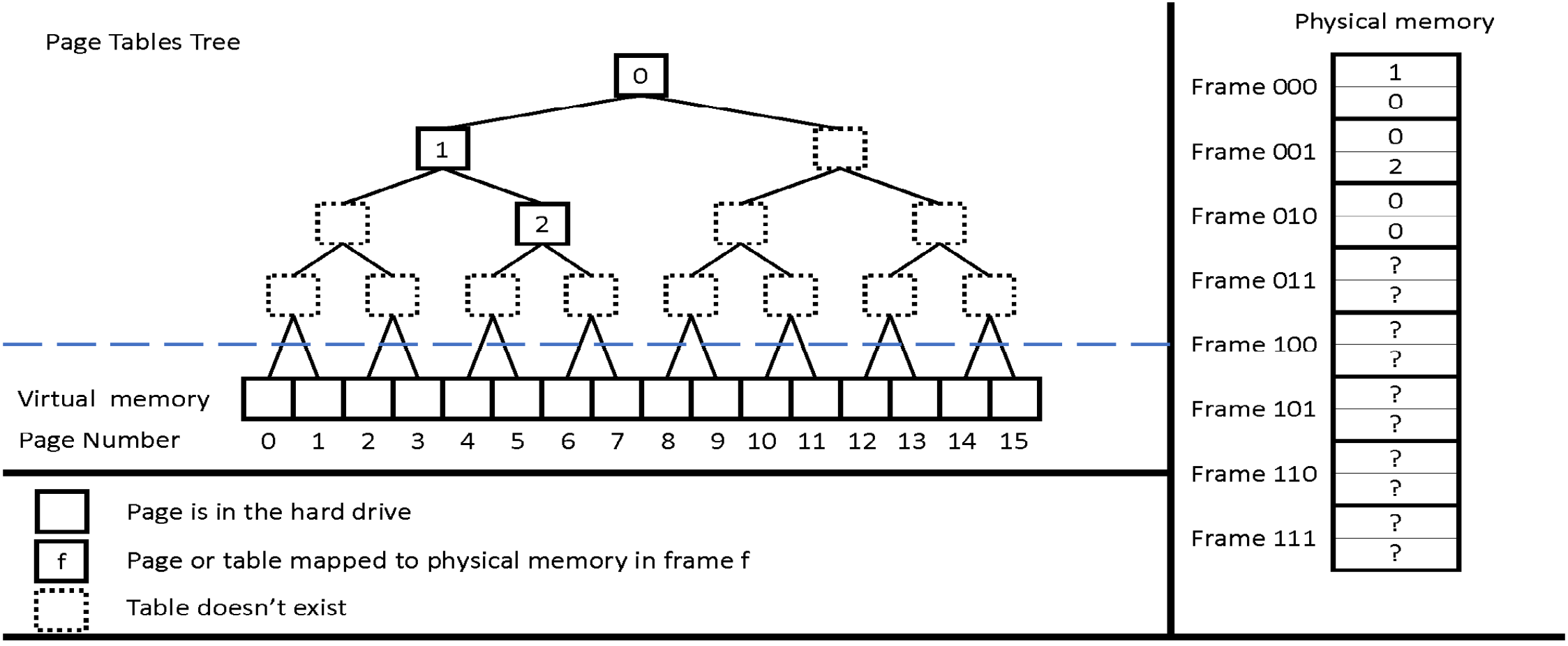
Only to hit the same issue again.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



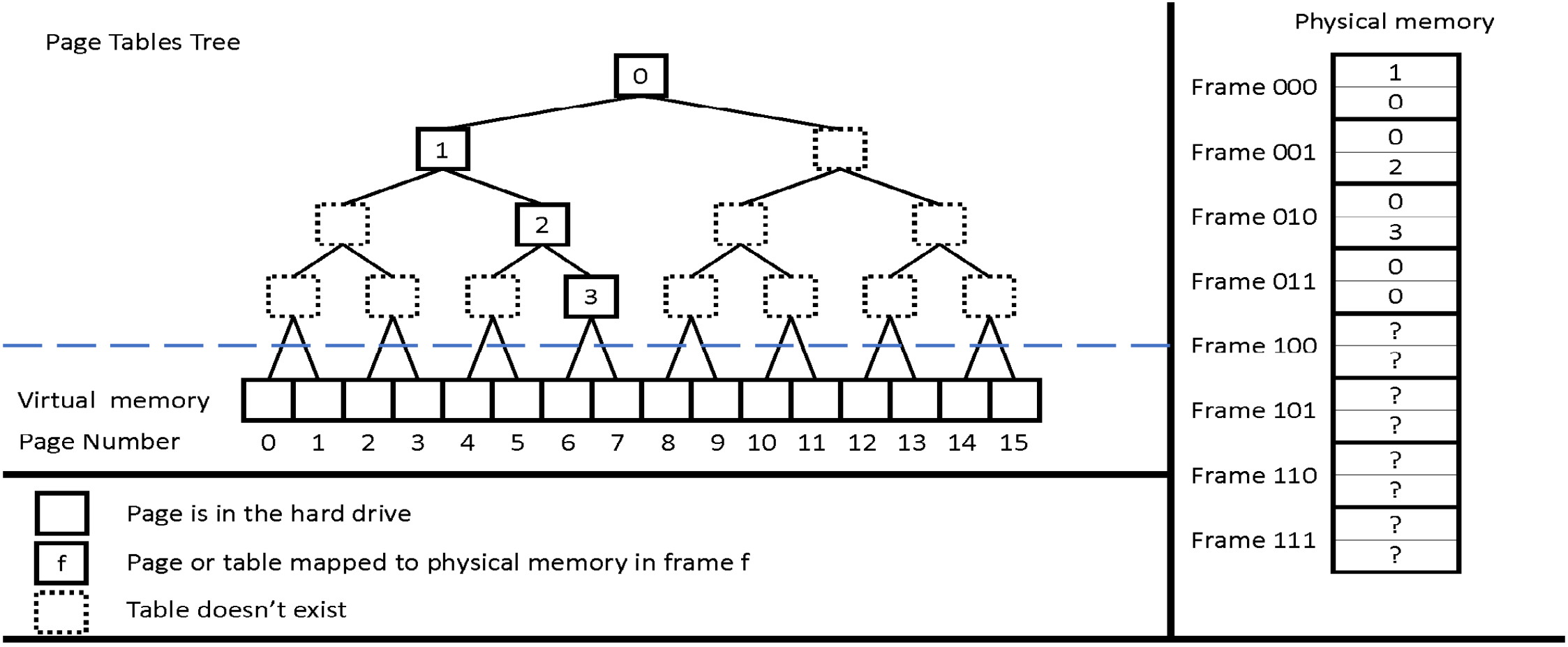
Now when traversing the tree we will enter frame 1 from the entry in frame 0.
Again, we keep track of the maximal frame visited during the traversal and we see that this time it is 1, so frame 2 must be unused.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



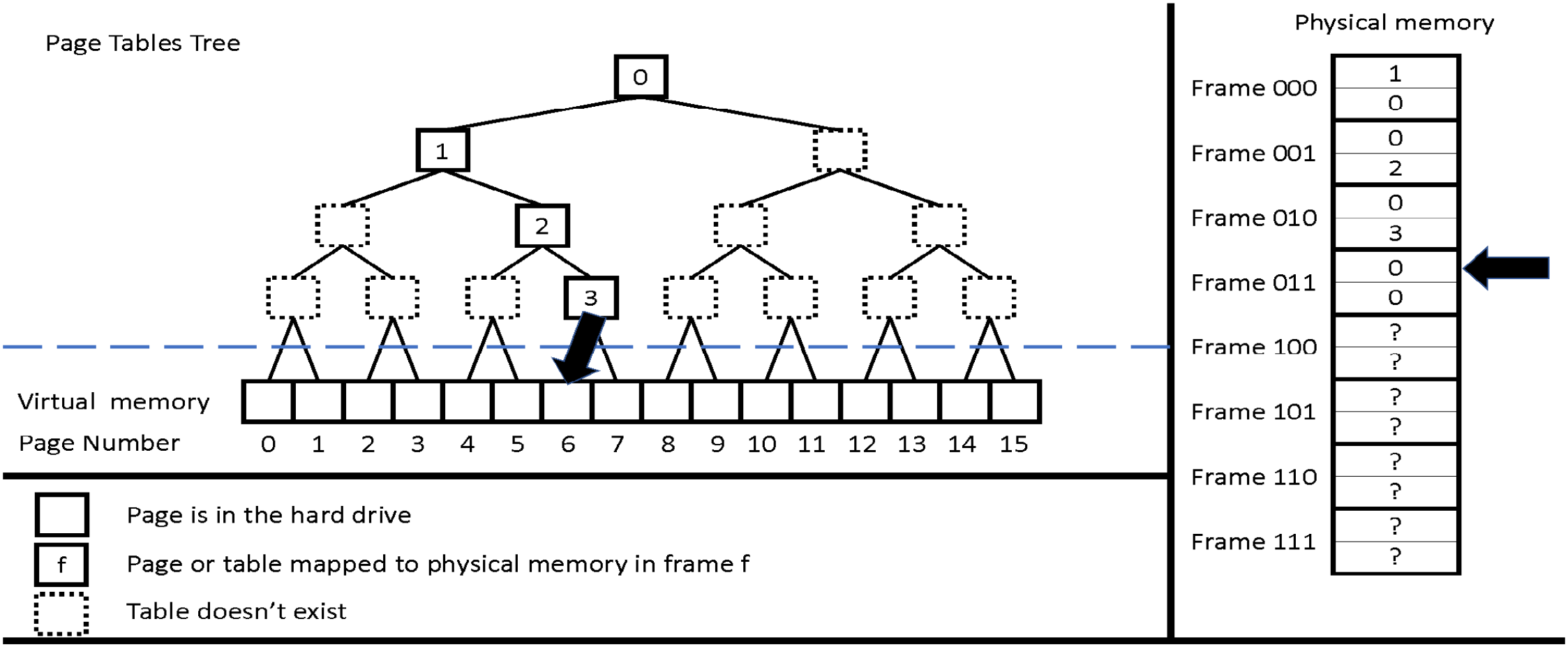
We map the table to frame 2 and fill it with zeros. We also link it from its parent table.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



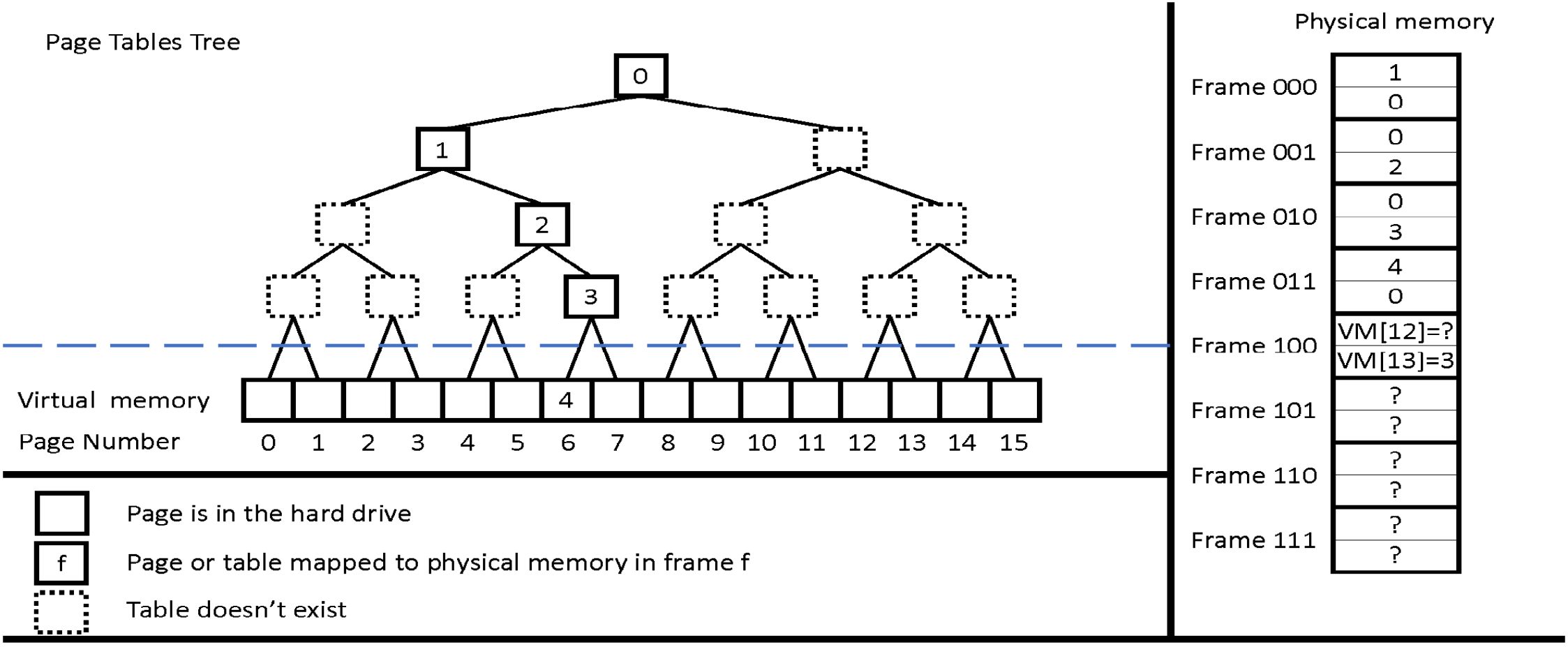
We continue in the same way and create another table for the next layer.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



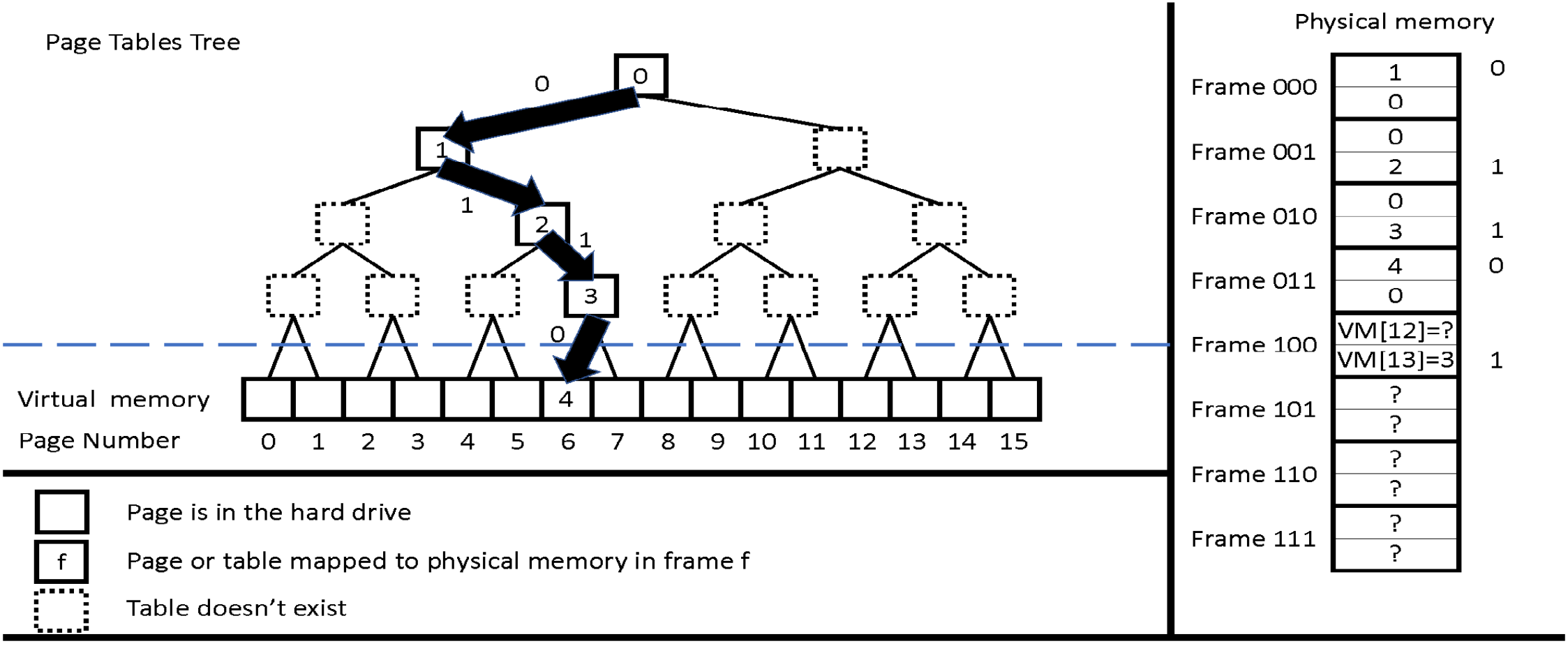
Arriving at the leaf we now want to bring page 6 into the physical memory.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



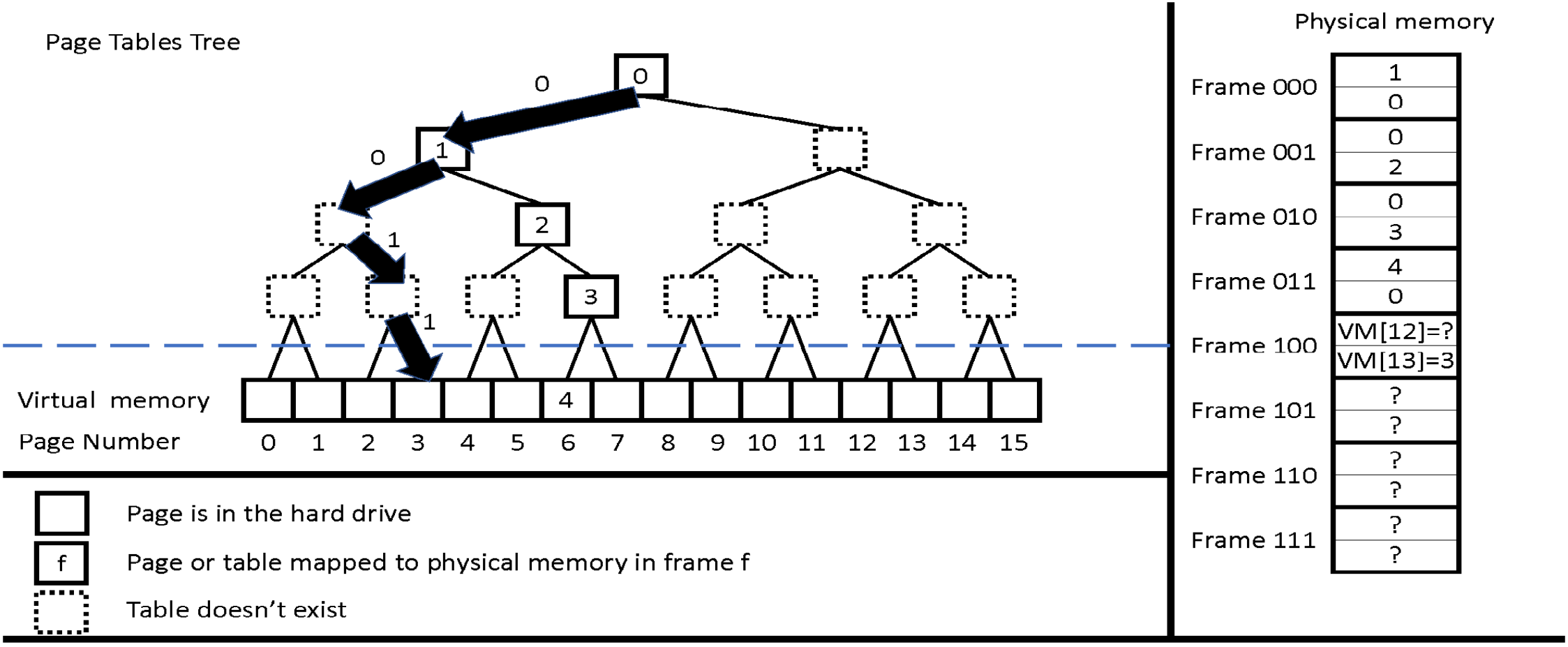
Similar to before, we find that frame 4 is unused and call PMrestore(4, 6).
We now combine our physical address 4, or 100 with the offset 1 to get 1001, or 9, and call PMwrite(9, 3)

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



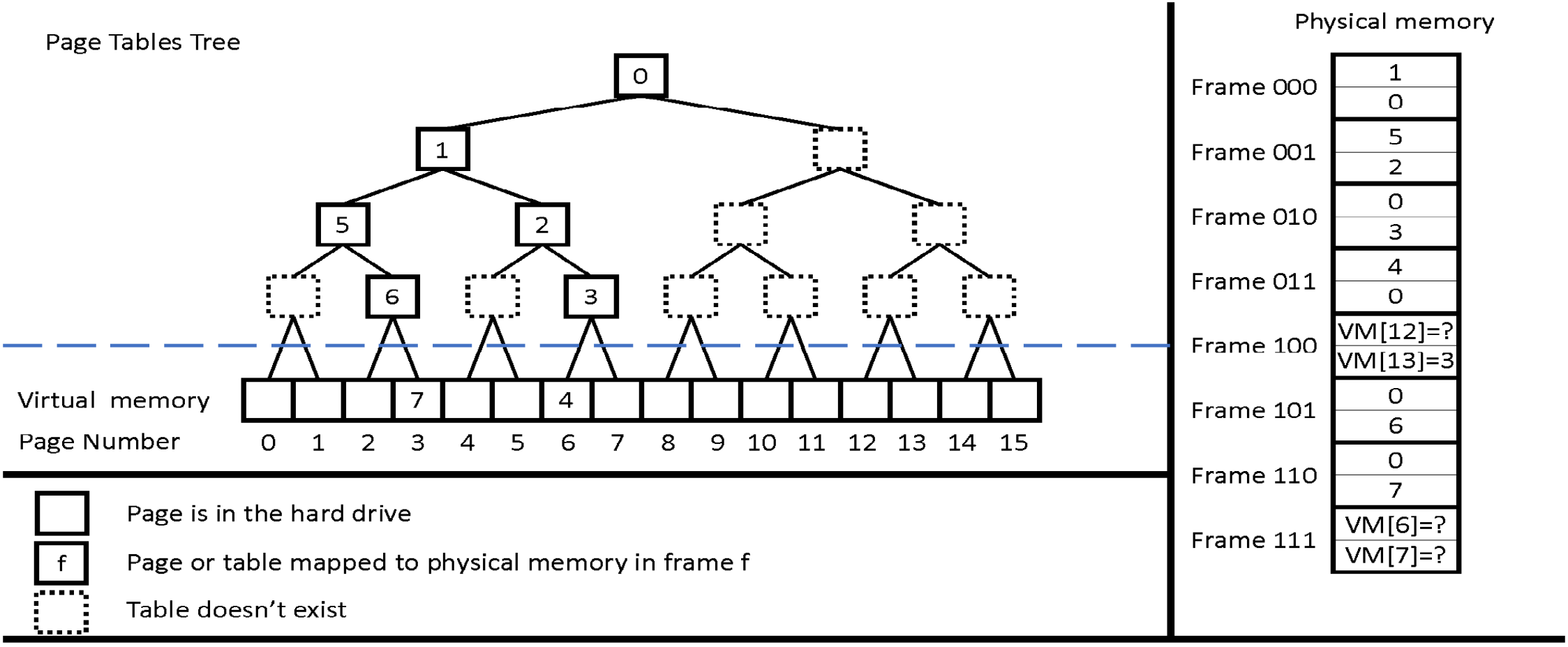
Now if we run VMread(13, &value), going down the tree will go smoothly and we can call PMread(9, &value) to get the appropriate value.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



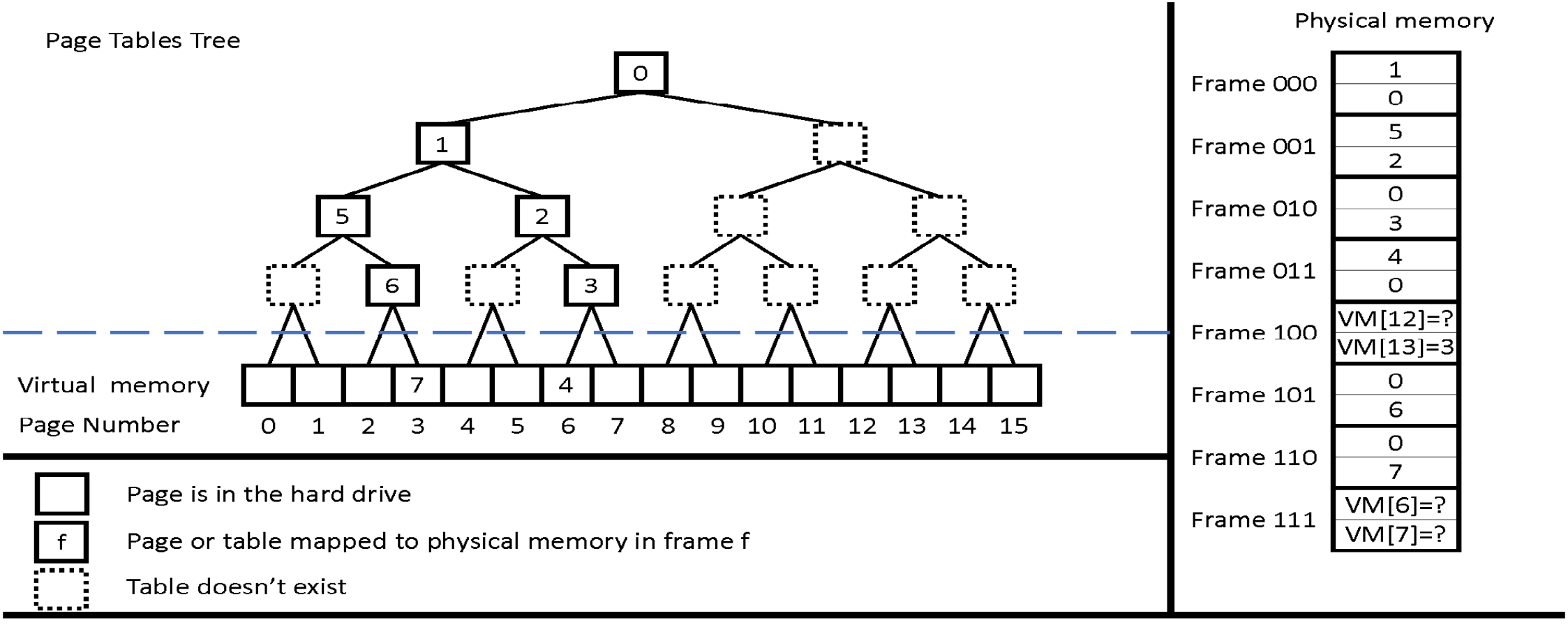
But if we run VMread(6, &value), we're looking for address 00110, and we have to do the same process we did before to find frames for the missing tables and the missing page.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



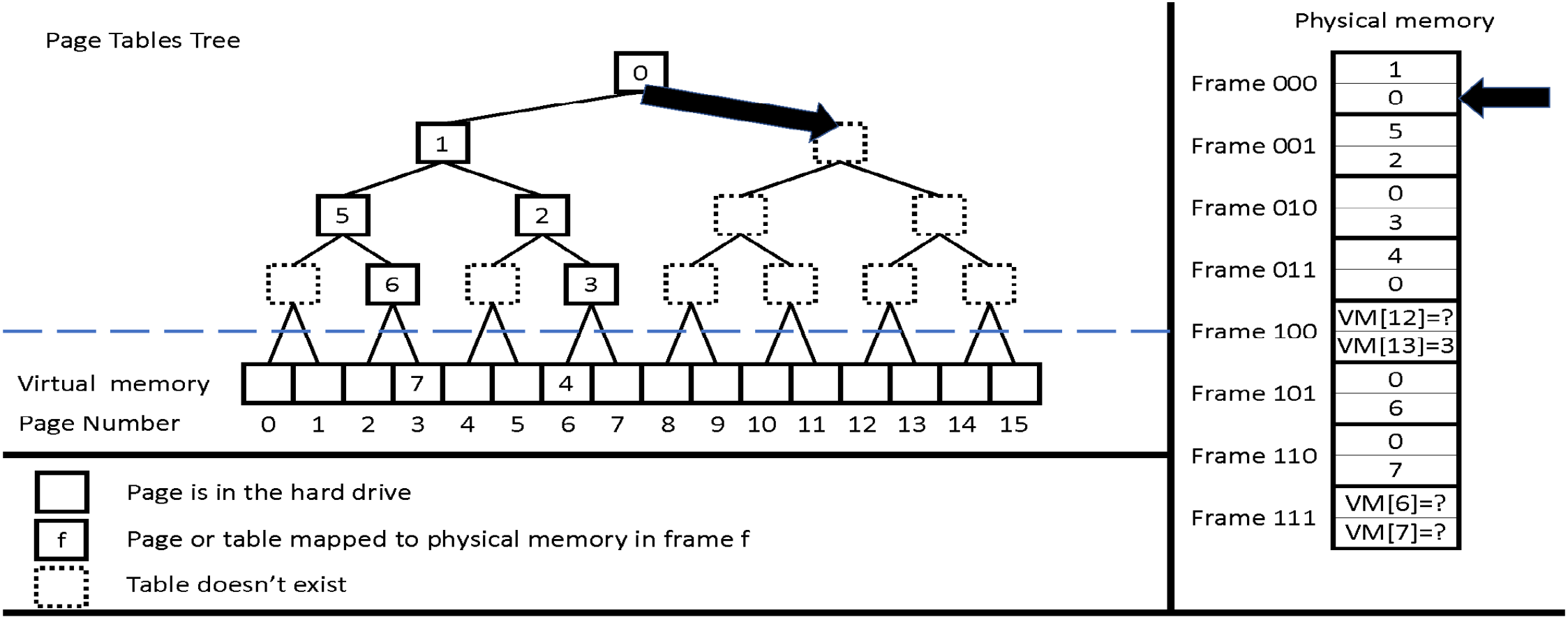
We end up with the situation above. Where our entire memory is full.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



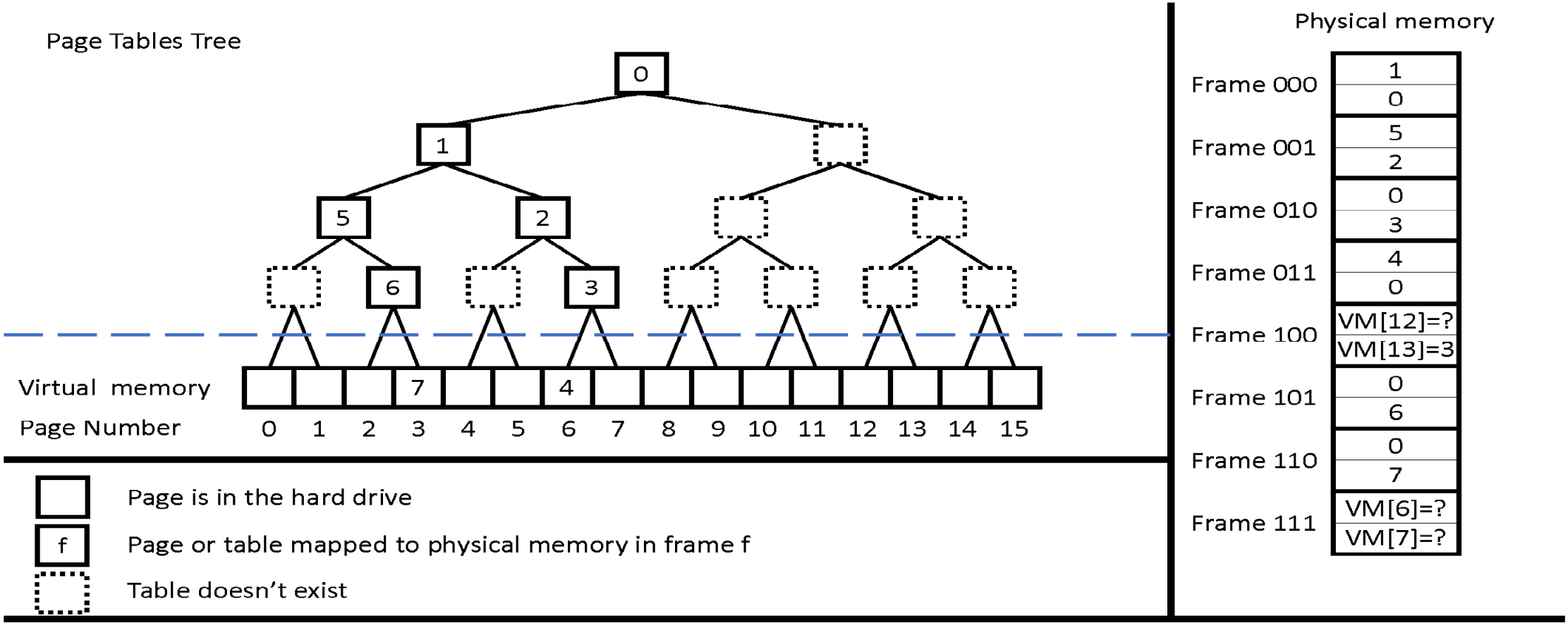
Let's now call VMread(31, &value).

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



The address is 11111, so the page number is 1111 and the offset is 1
We immediately hit a missing table directly from the root, so we will have to find a frame to create it in.

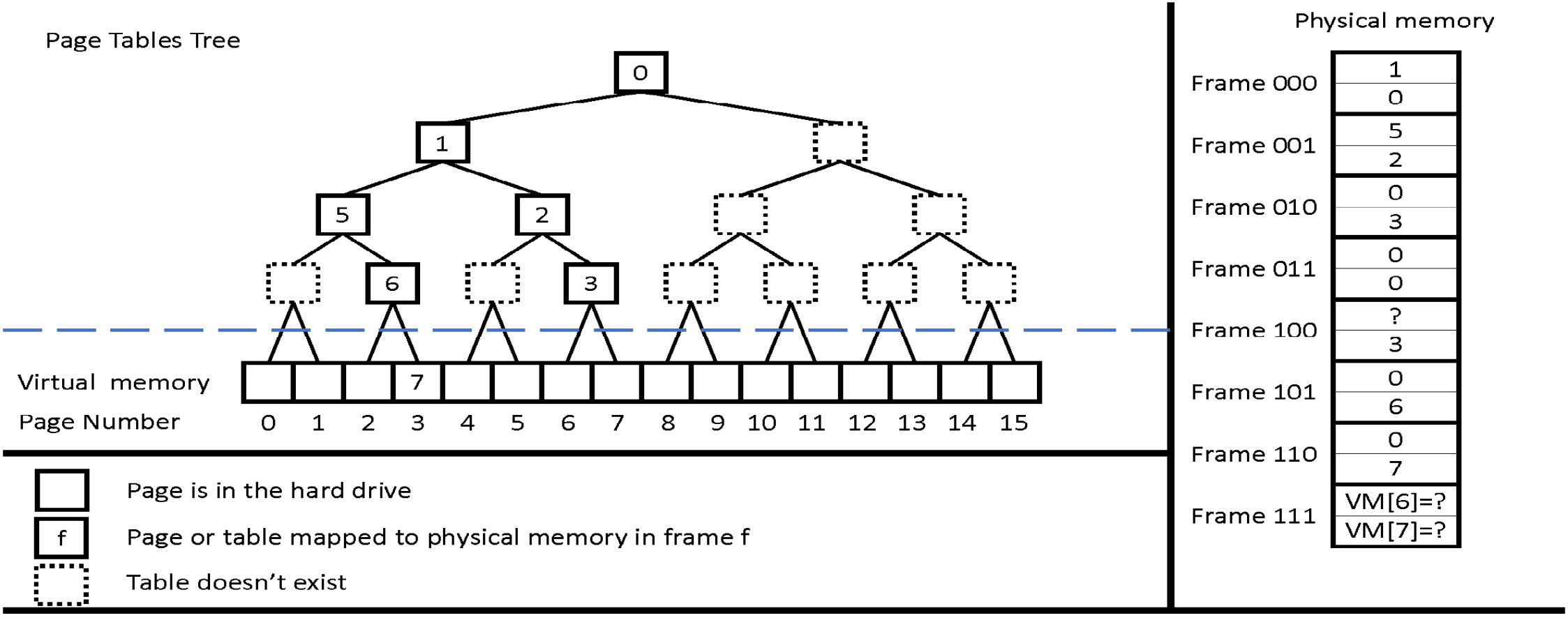
WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



This time, traversing through the tree we see that frame 7 is in use, which means there are no unused frames.
We also haven't seen any tables along the way with only zeros, so we have to evict a page.

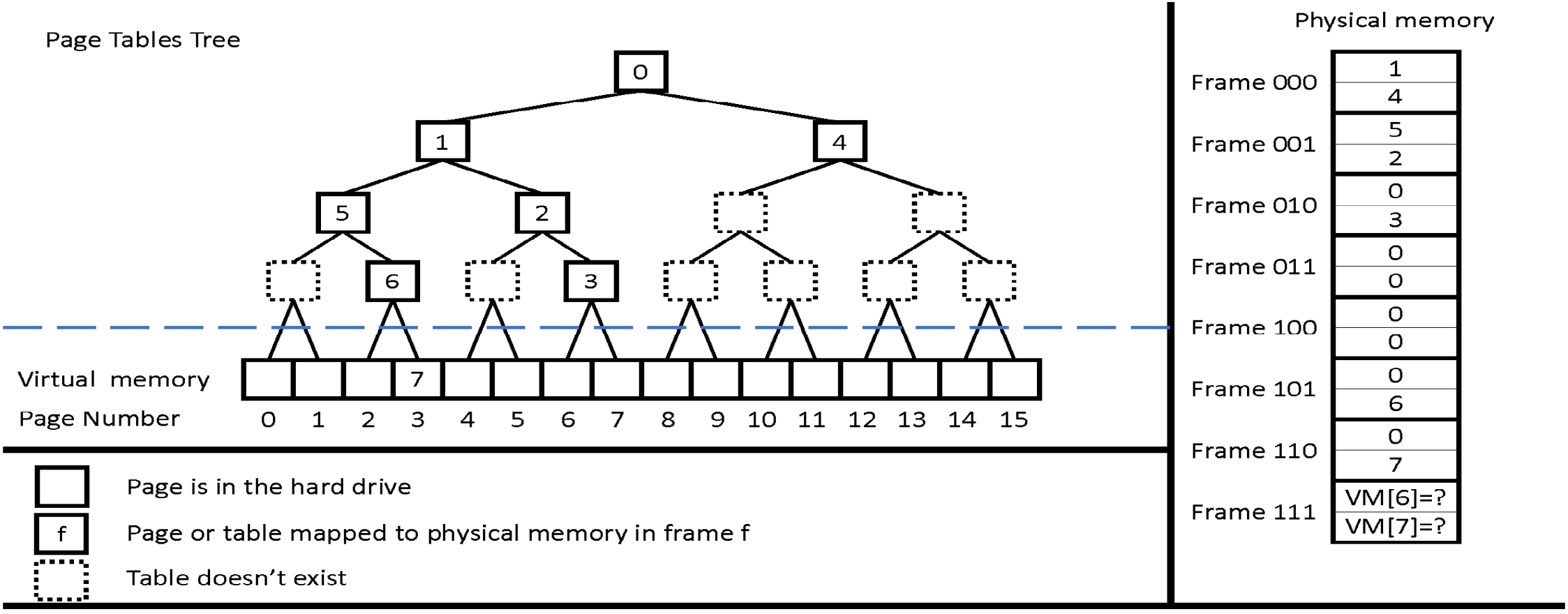
Now we will look at our evacuation algorithm, we have two paths, first one contains the following frames 0->1->2->3->4 and page number 6, the second one contains the following frames 0->1->5->6->7 and page number 3, so now we want to calculate the weight function for the first path, we have 4 even numbers [including the page number] and 2 odd, so the weight is (4*4 + 2*2) = 20, for the second path we have 2 even numbers and 4 odd [including the page number], so the weight is (2*4 + 4*2) = 16, the maximum between 16 and 20 is 20 which mean page 6 will be evacuated.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



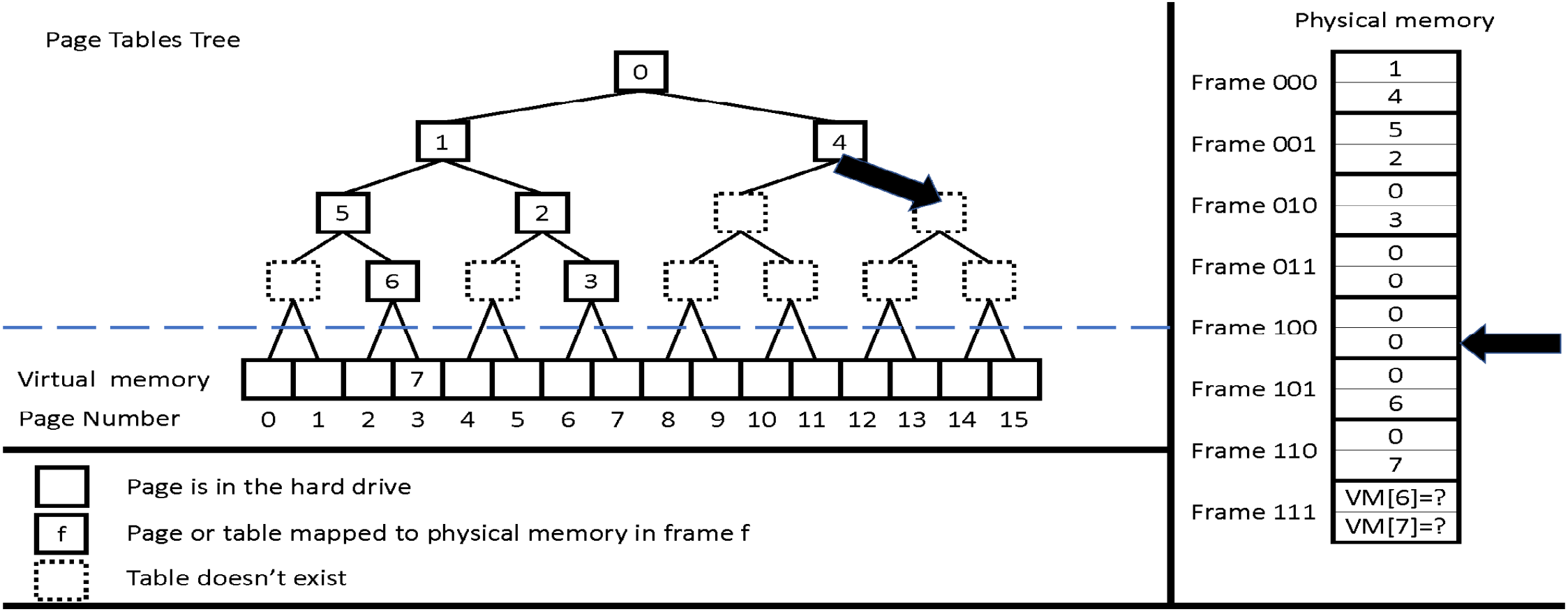
We call PMevict(4,6) and remove the link to it from its parent table which is in frame 3.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



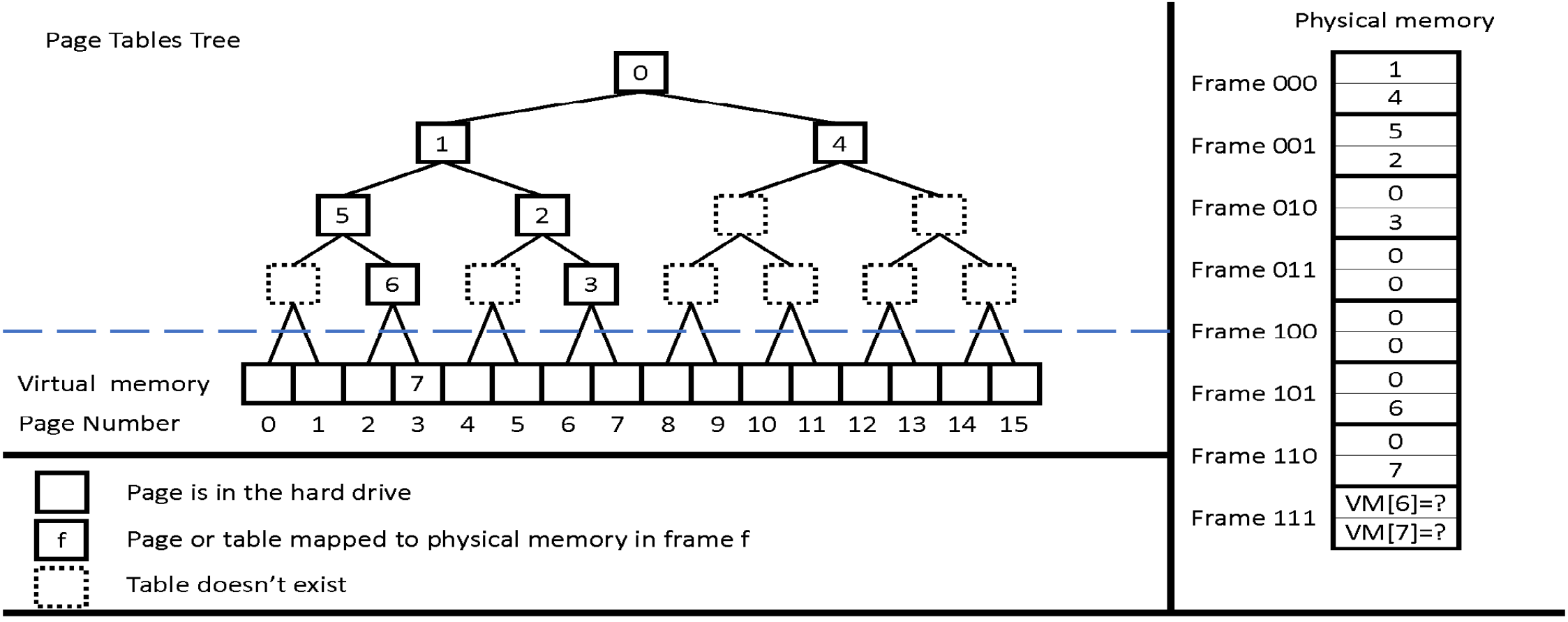
We create our new table in frame 4, fill it with zeros and link it from its parent – the root.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



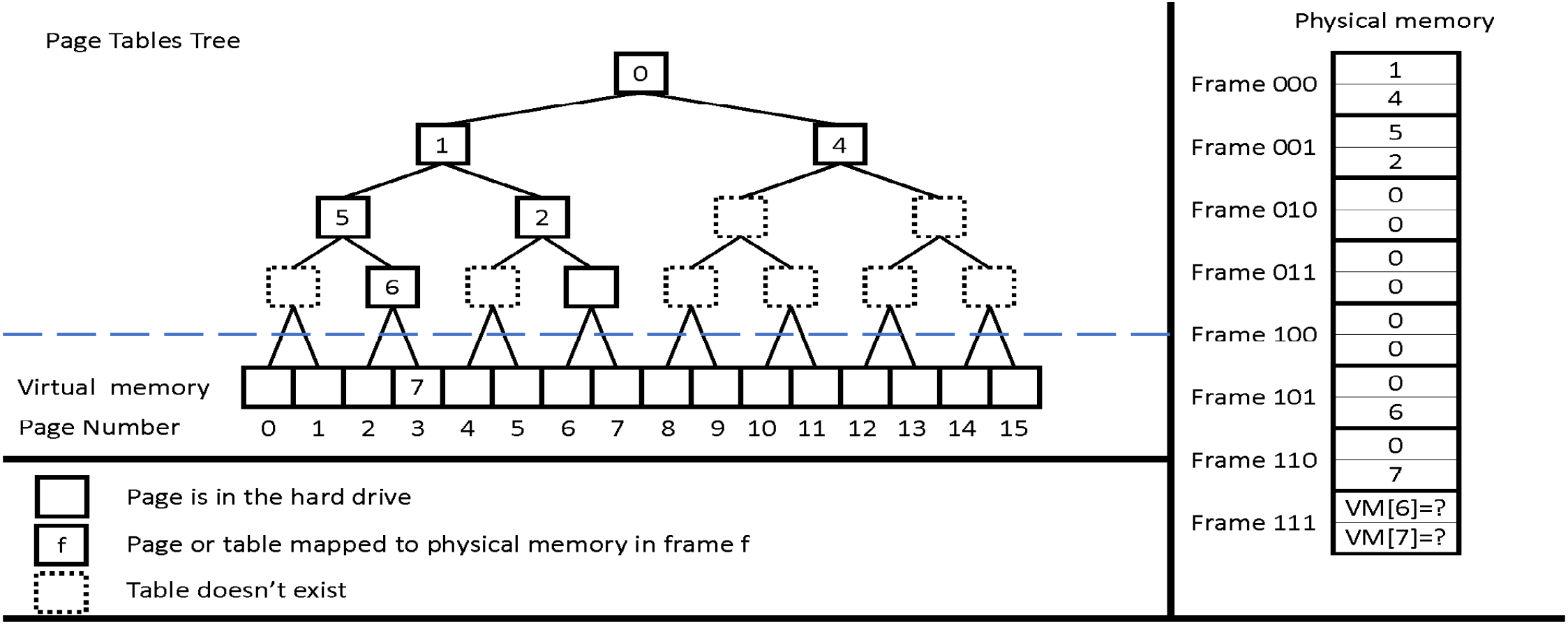
In the next level we hit another non-existing table, so again we must find a frame.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



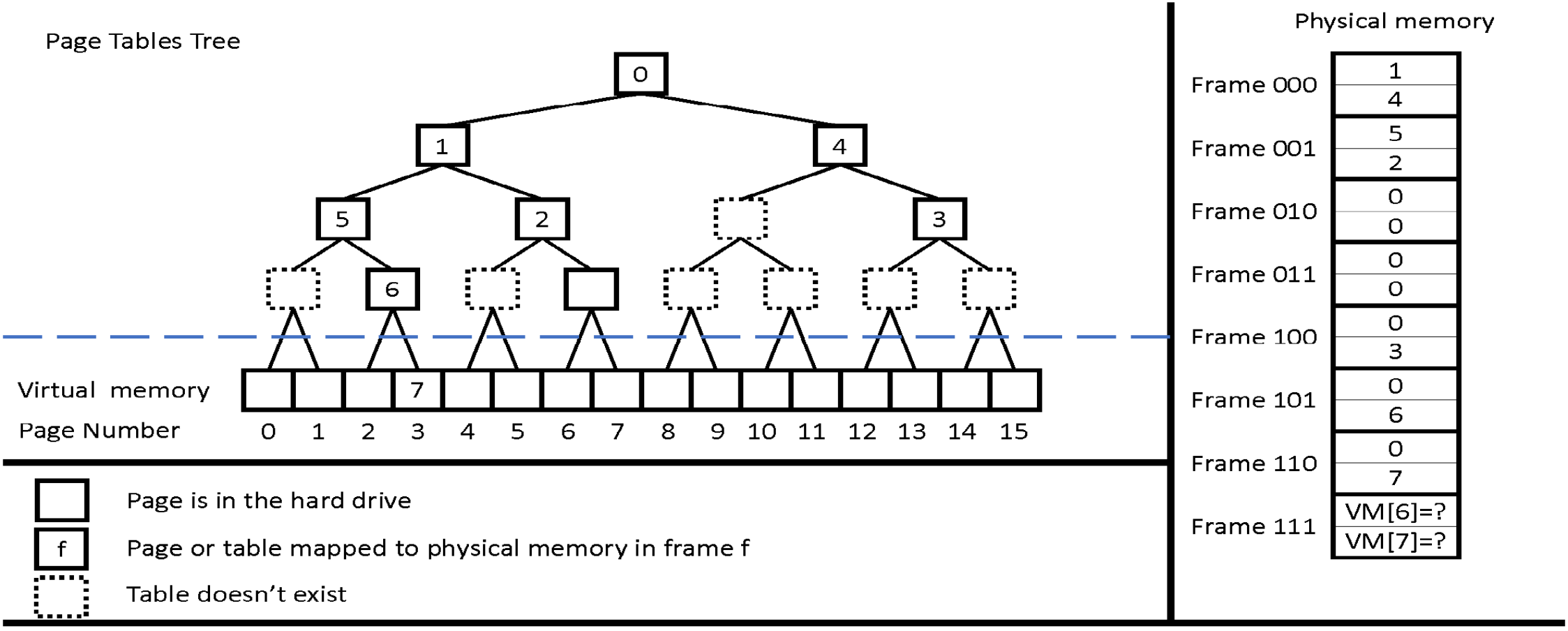
This time when traversing the tree we see that frame 3 only contains zeros, which means there are no pages below it and it can be safely removed.
Note that while frame 4 also only contains zeros, we don't want to remove it as we need it for page 15 (remember?).

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



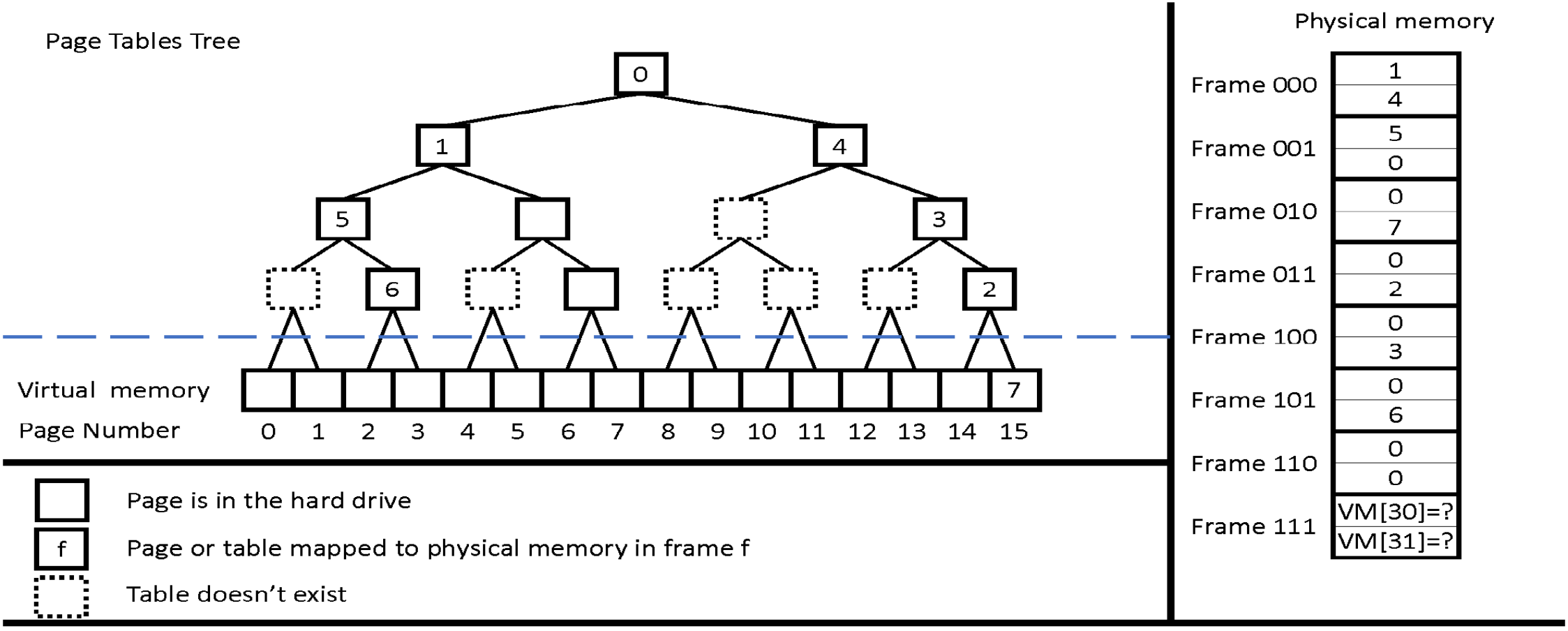
So we unlink frame 3 from the table above it which is in frame 2.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



And we create our new table.

WEIGHT_EVEN = 4
WEIGHT_ODD = 2
NUM_FRAMES = 8
NUM_PAGES = 16
TABLES_DEPTH = 4



We continue this process until finally page 15 is brought in.
Make sure you understand the steps leading to the state above.