برای پیدا کردن تعداد گره های پر درخت می توانیم به صورت بازگشتی تعداد گره های فرزندان چپ و راست گره را پیدا کرده و در صورت پر بودن گره ریشه آن را یک واحد اضافه می کنیم تا تعداد همه ی گره های پر را به دست آوریم. (طبق تعریف گره پر یا دو فرزند دارد یا فرزند ندارد)

CountFullNodes(P)

Require: Pointer p to tree T
Ensure: Count of full nodes in the tree
  get tree T pointed by P
  count = 0
  if T is empty **then**
    return 1
  else
    $countLeft = CountFullNodes(T_{left})$
    $countRight = CountFullNodes(T_{right})$
    if T has left and right child **then**
      count += 1
    count = count + countLeft + countRight
    return count

Listing 1: Python Implementation

```
def count_full_nodes(root):
    count = 0

    if not root.left and not root.right:
        return 1

    if root.left:
        count += count_full_nodes(root.left)
    if root.right:
        count += count_full_nodes(root.right)

    if root.right and root.right:
        count += 1

    return count
```

الگوریتم، مسئله را به دو مسئله کوچکتر تبدیل می کند و پس از حل آنها با انجام یک مقایسه و جمع آنها را باهم ترکیب می کند بنابراین اگر $T(n)$ نشان دهنده زمان اجرای الگوریتم باشد داریم :

$$T(n) = 2\,T(n/2) + \theta(1)$$

طبق قضیه اصلی، زمان اجرای الگوریتم $\theta(n)$ خواهد بود.