برای اینکه تشخیص دهیم که روی یک کامپیوتر در این شبکه نرم افزار نظارتی را نصب کنیم یا نه می‌توانیم از این نکات استفاده کنیم که اگر تمام فرزندان یک گره نظارت شوند و حداقل روی یکی از آنها این نرم افزار نصب شده باشد، احتیاجی به نصب نرم افزار روی این کامپیوتر نیست یا به عبارتی اگر حداقل یکی از فرزندان این گره تحت نظارت نباشد باید روی خود گره این نرم افزار را نصب کنیم تا این فرزند یا فرزندان را بتوانیم نظارت کنیم یا اگر همه‌ی فرزندان تحت نظارت باشند اما نرم‌افزار روی هیچ‌یک از آنها نصب نشده باشد، در اینصورت برای اینکه خود گره را تحت نظارت داشته باشیم، باید نرم افزار را روی خودش نصب کنیم.

با توجه به این دو شرط تعداد رایانه‌هایی که لازم است روی آنها نرم افزار را نصب کنیم به صورت زیر است:

$$f(T) = f(T_0) + f(T_1) \ldots + f(T_n) + max\{x, y\} \quad (۱)$$

$$x = 1 \; if \; \exists a \in children(T) : monitor(a) = 0 \; else \; 0 \quad (۲)$$

$$y = 1 \; if \; \forall a \in children(T) : software(a) = 0 \; else \; 0 \quad (۳)$$

$$(۴)$$

البته باید به این نکته توجه داشته باشیم که روی هیچ‌یک از برگ‌ها این نرم افزار نصب نخواهد شد زیرا می‌توان آن‌را روی گره پدرشان نصب کرد. می‌توانیم با برچسب زدن هر گره و فرزندانش در هر مرحله از پیمایش پس‌ترتیب درخت، اطلاعاتی که احتیاج داریم را بدست آوریم. ابتدا هر گره را به صورت زیر تعریف می‌کنیم تا اطلاعات موردنیاز را ذخیره کند:

```
class Node:
    def __init__(self, name=None, parent=None) -> None:
        self.parent = parent
        self.name = name
        self.children: list["Node"] = []
        self.is_installed = None
        self.is_monitored = None

    def add_child(self, *nodes: "Node") -> None:
        for node in nodes:
            node.parent = self
        self.children.extend(nodes)

    def remove_child(self, node: "Node") -> None:
        self.children.remove(node)
```

حال با اجرای تابع زیر می‌توانیم حداقل تعداد نرم‌افزار های موردنیاز و گره‌هایی که این نرم افزار باید روی آنها نصب شود را پیدا می‌کنیم، تابعی که از روی تابع بازگشتی بالا درست شده است و شرط‌ها را برای هر گره بررسی می‌کند.

```
def process_network_tree(root: Node):
    if len(root.children) == 0:
        if root.parent is None:
            return 1, [root.name]
        else:
            root.is_installed = False
            root.is_monitored = False
            return 0, []

    sofware_nodes = []
    softwares = 0
```

```
for child in root.children:
    count, nodes = process_network_tree(child)
    sofware_nodes += nodes
    softwares += count

x = any([not child.is_monitored for child in root.children])
y = all([not child.is_installed for child in root.children])
if x or y:
    root.is_installed = True
    root.is_monitored = True
    sofware_nodes = [root.name] + sofware_nodes
    softwares += 1
    root.parent.is_monitored = True
    for child in root.children:
        child.is_monitored = True

return softwares, sofware_nodes
```