První nápady

První, co si můžeme uvědomit je, že můžeme jednoduše mít 2 pointery - jeden pro levou a druhý pro pravou půlku stromu. Když budeme pohybovat oběma pointery najednou, nemusíme si pamatovat jaké pohyby jsme udělali, protože ty pohyby budeme dělat v obou půlkách stromu (samozřejmě zrcadlově).

Takhle jsme si zjednodušili problém a jednoduchý průchod stromem s podmínkou konstantní paměti.

Jak to uděláme?

Teď musíme určit nějaká pravidla jak bude fungovat náš program, abychom mohli zaručit, že jsme opravdu prošli všechny vrcholy stromu. Využijeme toho, že nemusíme mít zadaný strom vcelku na konci algoritmu a stačí, abychom jen určili jestli je symetrický, či ne.

Začneme u vrcholu hned pod kořenem, bez újmy na obecnosti, v levém podstromu a řekneme si, že každý náš krok v levém podstromu uděláme zrcadlově v pravém. Dál popíšu instrukce, které použijeme u každého vrcholu, bez jakékoliv změny.

Pokud vrchol má levého syna, půjdeme k levému synovi. Pokud nemá levého syna, ale má pravého syna, půjdeme k pravému synovi. Pokud nemá žádného syna, půjdeme zpátky nahoru k otci.

Ale jak by tohle mohlo fungovat? Přece bychom se takhle zacyklili mezi vrcholem bez synů a jeho otcem. Aby teda mohl náš algoritmus pokračovat dál, začeneme odstraňovat odkazy na syny vrcholů, které jsme už zcela prohlédli. Jak ale poznáme, kterého syna vymazat? Toto můžeme určit pomocí těch instrukcí, které jsem popsal o odstavec dříve, tedy:

Víme, že jsme se vrátili k otci: Pokud má vrchol levého syna, vymažeme odkaz na něj (zrovna jsme tam byli). Pokud nemá levého syna, ale má pravého syna, vymažene odkaz na něj (zřejmě jsme odtamtud vyšli). Pokud vrchol už nemá žádné syny, půjdeme opět nahoru k otci.

Pomocí těchto dvou sad instrukcí, nevymažeme žádný vrchol, u kterého jsme ještě nebyli a postupně projdeme celým (pod)stromem. Samozřejmě také při tom všem, když návštívíme vrchol, zkontrolujeme, že opačný podstrom je stále zrcadlový. Konec algoritmu poznáme tím, že naše procházka narazí na vrchol, který už nemá otce.

Na další stránce je toto napsané v pseudokódu:

Algorithm 1 $Main(leftNode, rightNode)$

Input: node $leftNode$, equivalent node $rightNode$ of the opposite subtree

```
function compareNodes(leftNode, rightNode)
    if leftNode.value == rightNode.value then
        if leftNode has leftChild == rightNode has rightChild then
            if leftNode has rightChild == rightNode has leftChild then
                return True
            end if
        end if
    end if
    return False
end function

function goUp(leftNode as reference, rightNode as reference)
    leftNode ← leftNode.parent
    rightNode ← rightNode.parent
    if leftNode has leftChild then
        leftNode.leftChild ← null
        rightNode.rightChild ← null
    else if leftNode has rightChild then
        left.rightChild ← null
        rightNode.leftChild ← null
    end if
end function

while leftNode has parent do
    if not compareNodes(leftNode, rightNode) then
        return False
    end if.
    if leftNode has leftChild then
        leftNode ← leftNode.leftChild
        rightNode ← rightNode.rightChild
    else if leftNode has rightChild then
        leftNode ← leftNode.rightChild
        rightNode ← rightNode.leftChild
    else
        call goUp(leftNode)
    end if
end while
return True
```

Output: bool $result$

Tento algoritmus projde celý levý a pravý podstrom najednou v lineárním čase a při tom ukládá pouze 2 vrcholy, které současně porovnáváme. Tedy paměťová složitost je konstantní a časová složitost lineární.