10–1 木構造

前章で学習したリストは、順序付けられたデータの並びを表現するデータ構造でした。本章では、データ間の階層的な関係を表現するデータ構造である《木構造》を学習します。

木とは

本章で学習するのは**木構造**です。まずは、木（*tree*）とは何かを理解するとともに、木に関する用語を **Fig.10-1** を見ながら覚えていきましょう。

木に関する用語

木を構成するのは**ノード／節**（*node*）と**枝**（*edge*）です。各ノードは枝を通じて他のノードと結びつきます。図中の○がノードで、── が枝です。

なお、図の上側を**上流**と呼び、下側を**下流**と呼びます。

根 …… 最も上流のノードが**根**（*root*）です。一つの木に対して、根は一つだけ存在します。
植物の木に根があるのと同じです。図の上下を逆にすると、木のイメージが分かりやすくなります。

葉 …… 最下流のノードが**葉**（*leaf*）です。**終端節**（*terminal node*）あるいは**外部節**（*external node*）とも呼ばれます。

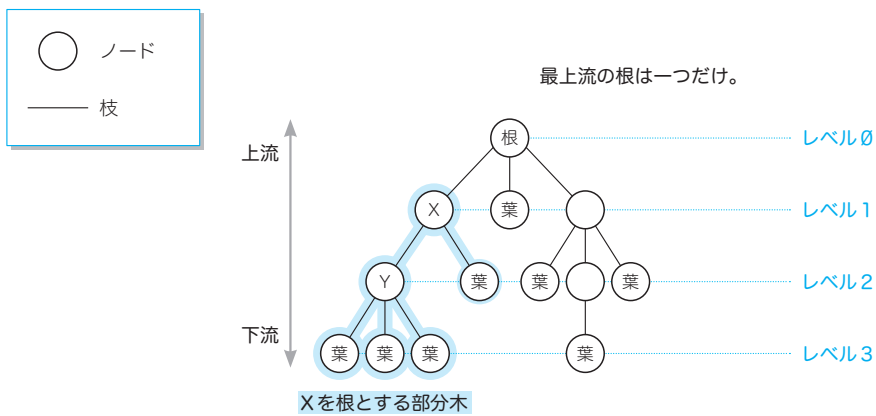

Fig.10-1 木

非終端節 … 葉以外のノード（根を含みます）が**非終端節**（*non-terminal node*）です。**内部節**（*internal node*）とも呼ばれます。

子 …… あるノードと枝で結ばれた下流側のノードが**子**（*child*）です。各ノードは何個でも子をもつことができます。
たとえば、Xは2個の子を、Yは3個の子をもっています。
なお、最下流の**葉は子をもちません**。

親 …… あるノードと枝で結ばれた上流側のノードが**親**（*parent*）です。各ノードにとって親は1個だけです。たとえば、Yにとっての親はXです。
なお、**根だけは親をもちません**。

兄弟 …… 共通の親をもつノードが**兄弟**（*sibling*）です。

先祖 …… あるノードから上流側にたどれるすべてのノードが**先祖**（*ancestor*）です。

子孫 …… あるノードから下流側にたどれるすべてのノードが**子孫**（*descendant*）です。

レベル …… 根からどれくらい離れているかを示すのが**レベル**（*level*）です。最上流である根のレベルは0であり、枝を一つ下流へとたぐっていくたびに、レベルは一つずつ増加します。

度数 …… 各ノードがもつ子の数が**度数**（どすう）（*degree*）です。たとえば、Xの度数は2で、Yの度数は3です。
なお、すべてのノードの度数がn以下である木を**n進木**（しんぎ）と呼びます。ここに示す木は、すべてのノードの子が3個以下ですから、3進木です。
すべてのノードの子の数が2個以下であれば、その木は2進木です。

高さ …… 根から最も遠い葉までの距離、すなわち葉のレベルの最大値が、**高さ**（*height*）です。ここに示す木の高さは3です。

部分木 …… あるノードを根とし、その子孫から構成される木が**部分木**（ぶぶんぎ）（*subtree*）です。青色で囲んだ部分は、Xを根とする部分木です。

空木 …… ノードや枝がまったく存在しない木が**空木**（くうぎ）（*null tree*）です。

順序木と無順序木

兄弟ノードの順序関係を区別するかどうかで、木は２種類に分類されます。

兄弟関係にあるノードの順序関係を区別する木が**順序木**（*ordered tree*）で、区別しない木が**無順序木**（*unordered tree*）です。

たとえば、**Fig.10-2** に示す図**a**と図**b**は、順序木としてみれば別の木ですが、無順序木としてみれば同じ木です。

二つの木は、異なる順序木であって、同一の無順序木である。

Fig.10-2　順序木と無順序木

順序木の探索

順序木のノードを走査する方法には、大きく二つの手法があります。ここでは、２進木を例に考えていきます。

横型探索（breadth–first search）

幅優先探索とも呼ばれる**横型探索**は、レベルの低い点から始めて、**左側から右側へとなぞり**、それが終わると次のレベルにくだる方法です。

Fig.10-3 に示すのが、横型探索によってノードを走査する例です。ノードをなぞる順は、以下のようになります。

A ➡ B ➡ C ➡ D ➡ E ➡ F ➡ G
➡ H ➡ I ➡ J ➡ K ➡ L

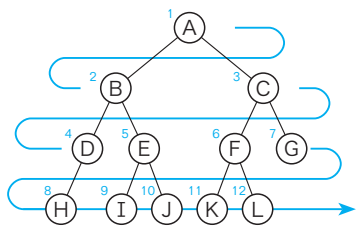

Fig.10-3　横型探索

縦型探索（depth–first search）

深さ優先探索とも呼ばれる**縦型探索**は、葉に到達するまで**下流にくだっていく**のを優先する方法です。

葉に到達して行き止まりとなった場合は、いったん親に戻って、それから次のノードへとたどっていきます。

Fig.10-4 に示すのが、縦型探索の走査の概略です。

ここで、ノードAに着目します。**Fig.10-5** に示すように、走査の過程でAを通過するのは全部で３回です。

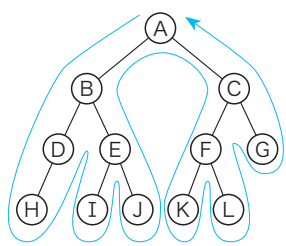

Fig.10-4　縦型探索

- AからBにくだる直前。
- BからCに行く途中。
- CからAに戻ってきたとき。

他のノードでも同様です。二つの子の一方あるいは両方がなければ回数は少なくなるものの、各ノードを最大3回通過します。

3回の通過のうちの、どの時点で実際に“**立ち寄る**”のかによって、縦型探索は以下に示す3種類の走査法に分けられます。

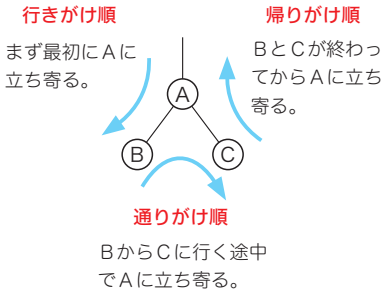

Fig.10-5　縦型探索と走査

▪ 行きがけ順（preorder：前順／先行順）

次の手順で走査します。

ノードに立ち寄る ➡ 左の子にくだる ➡ 右の子にくだる

Fig.10-4 の木を考えましょう。たとえばノードAを通過するタイミングに着目すると、{ Aに立ち寄る➡Bにくだる➡Cにくだる }という手順です。

そのため、木全体の走査は次のようになります。

A ➡ B ➡ D ➡ H ➡ E ➡ I ➡ J ➡ C ➡ F ➡ K ➡ L ➡ G

▪ 通りがけ順（inorder：間順／中間順）

次の手順で走査します。

左の子にくだる ➡ **ノードに立ち寄る** ➡ 右の子にくだる

たとえばノードAを通過するタイミングに着目すると、{ Bにくだる➡Aに立ち寄る➡Cにくだる }という手順です。

そのため、木全体の走査は次のようになります。

H ➡ D ➡ B ➡ I ➡ E ➡ J ➡ A ➡ K ➡ F ➡ L ➡ C ➡ G

▪ 帰りがけ順（postorder：後順／後行順）

次の手順で走査します。

左の子にくだる ➡ 右の子にくだる ➡ **ノードに立ち寄る**

木全体の走査は、次のようになります。

H ➡ D ➡ I ➡ J ➡ E ➡ B ➡ K ➡ L ➡ F ➡ G ➡ C ➡ A