簡潔データ構造 第 3 回
順序木の LOUDS 表現

marimo

2023 年 4 月 25 日

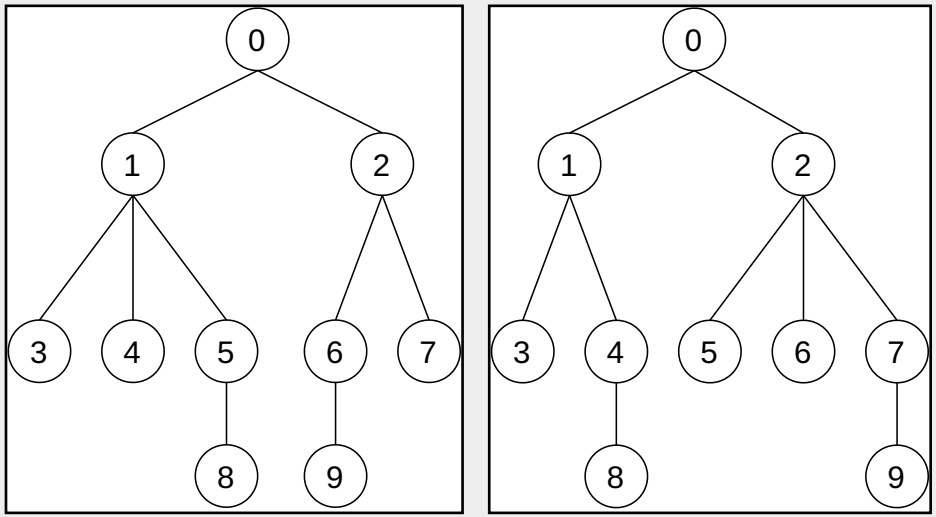
準備 : 順序木

定義

順序木とは根付き木であって各頂点の子に順序がついているもの

n 頂点の順序木の総数はカタラン数 C_{n-1} になるので，表現の情報理論的下限は

$$\left\lceil \lg \frac{1}{n^{2(n-1)}\mathrm{C}_{n-1}} \right\rceil = 2n - O(\lg n)\,\mathrm{bits}$$



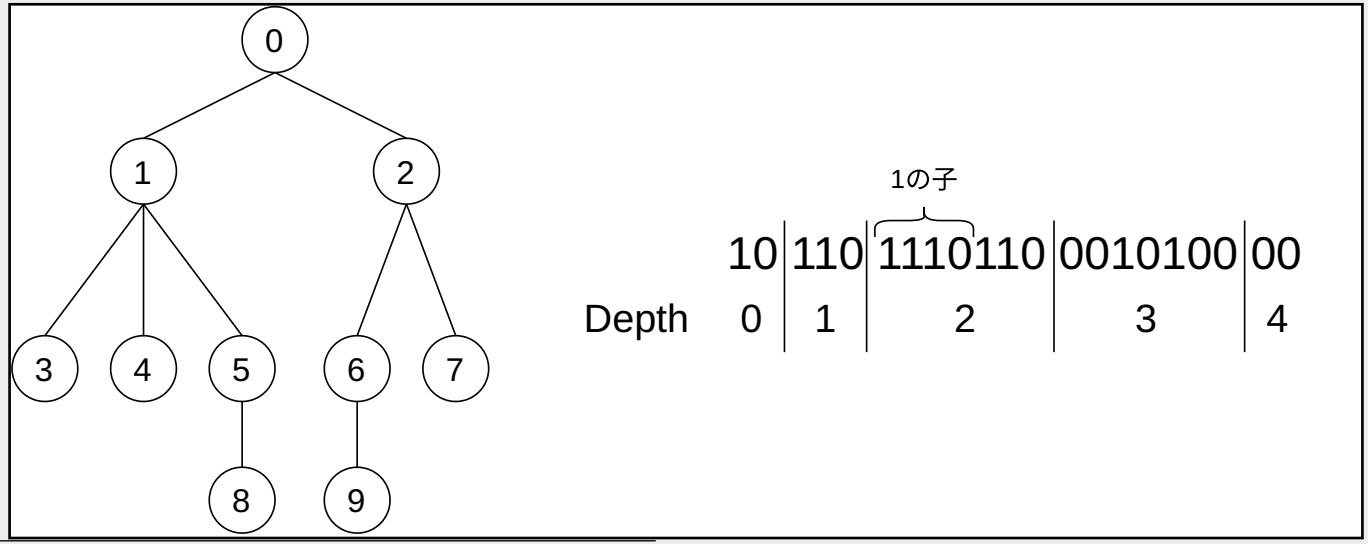
順序木の例 2 つ。 これらは単なる根付き木としては同型だが順序木としては同型でない。

LOUDS 表現

定義 (LOUDS 表現 level-order unary degree sequence representation).

順序木の **LOUDS 表現**とは次数が d のノードを d 個の 1 と続く 1 個の 0 で表し，幅探索順で並べたビット列 L のこと。ただし仮想的な根の親を付けて 10 で表す。*1

深さ d のノードたちに対応するビット列を深さ $d+1$ のビット列と呼ぶことにする。



1 これは簡単のために付けているので 2 bits ケチりたいなら消せる

LOUDS 表現

補題

任意の LOUDS 表現 L について，$d \geq 0$ に対して， 深さ d のビット列の中の 1 の数と深さ $d+1$ のビット列の中の 0 の数，および深さ d のノード数は全て等しい。

深さ $d \geq -1$[*2]のノードの次数は L の深さ $d+1$ に 1 進数符号で書いてある。この次数の和は深さ $d+1$ のノードの総数を表していたから，1 の総数はこれに等しい。

深さ d のノードに対して L の深さ $d+1$ の列に 1 つの符号が存在する。1 つの符号には 1 つの 0 が存在する。 つまり深さ $d+1$ の列の 0 の数と深さ d のノードの数は等しい。□

2 便宜的に仮想的な根の親の深さを -1 とする

LOUDS 表現

補題から幅優先順で i 番目のノードと i 番目の 1 が対応付けられるので，対応する 1 の位置をノードの**位置**として定める。

$\mathrm{bfs\_rank}(x)$ で位置 x のノードの幅優先順，$\mathrm{bfs\_select}(i)$ で幅優先順で i 番目のノードの位置を表すことにすると

$$i = \mathrm{bfs\_rank}(x) = \mathrm{rank}_1(L, x)$$
$$x = \mathrm{bfs\_select}(i) = \mathrm{select}_1(L, i)$$

LOUDS による演算

次の 2 つの演算を新たに定義する。

$$\begin{aligned}\mathrm{parent\_rank}(x) &= \text{位置 } x\text{のノードの親の幅優先順} \\ \mathrm{first\_child\_select}(i) &= \text{幅優先順で } i\text{番目のノードの最初の子の位置}\end{aligned}$$

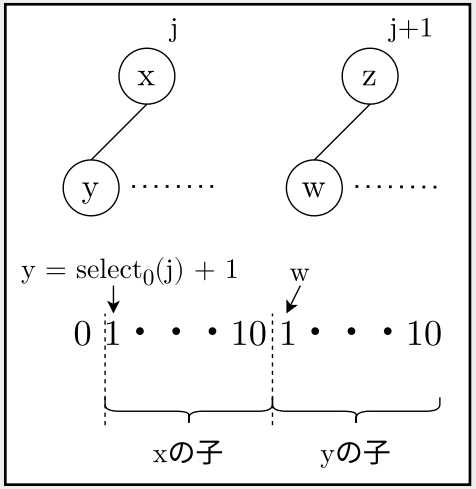
ただし，葉については対応する符号 0 を最初の子の位置として定める。

LOUDS による演算

補題

任意の LOUDS 表現 L について次が成立する。i をあるノードの幅優先順，x をそのノードの最初の子の位置とする。

$$i = \mathrm{parent\_rank}(x) = \mathrm{rank}_0(L, x-1)$$
$$x = \mathrm{first\_child\_select}(i) = \mathrm{select}_0(L, i) + 1$$



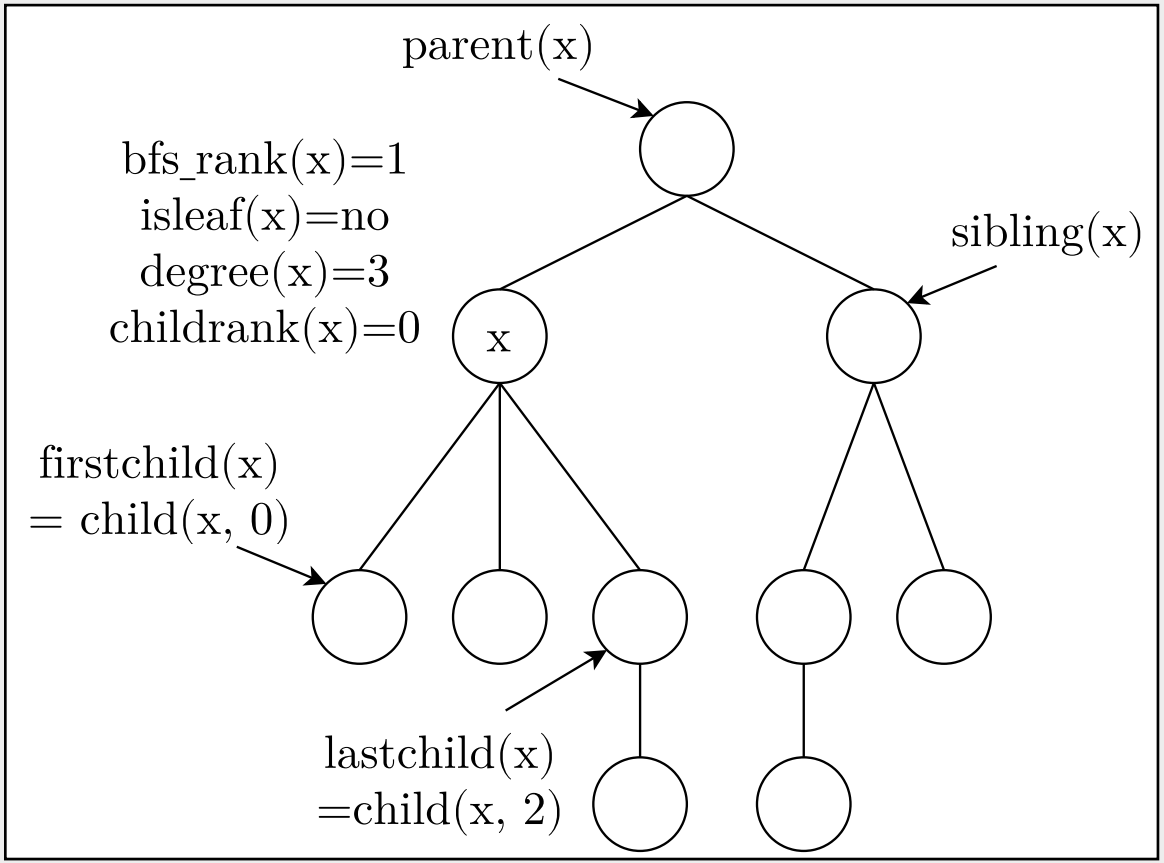
ノードの幅優先順 j についての帰納法によって示す。幅優先順が 0 の根については具体的に計算すると成立していることがわかる。
次に 0 から j まで成立するならば $j+1$ でも成立することを示すが，これは左の図から OK.

LOUDS による演算

これまでに説明した 4 つの演算を組み合わせることで次のような演算が実現できる。
x は位置。LOUDS をビットベクトルに載せれば全部定数時間で計算できる。

- $\mathrm{isleaf}(x)$:if $L\Big[\mathrm{first\_child\_select}\big(\mathrm{bfs\_rank}(x)\big)\Big] = 0$ then yes else no
- $\mathrm{parent}(x)$: $\mathrm{bfs\_select}\big(\mathrm{parent\_rank}(x)\big)$
- $\mathrm{firstchild}(x)$: $y = \mathrm{first\_child\_select}\big(\mathrm{bfs\_rank}(x)\big), \mathrm{if}\ L\big[y\big] = 0$ then -1 else y
- $\mathrm{lastchild}(x)$: $y = \mathrm{select}_0\big(\mathrm{bfs\_rank}(x) + 1\big) - 1, \mathrm{if}\ L\big[y\big] = 0$ then -1 else y
- $\mathrm{sibling}(x)$:if $L[x+1] = 0$ then -1 else $x+1$
- $\mathrm{degree}(x)$:if $\mathrm{isleaf}(x)$ then 0 else $\mathrm{lastchild}(x) - \mathrm{firstchild}(x) + 1$
- $\mathrm{child}\big(x, i\big)$:if $i \geq \mathrm{degree}(x)$ then -1 else $\mathrm{firstchild}(x) + i$
- $\mathrm{childrank}(x)$: $x - \mathrm{firstchild}\big(\mathrm{parent}(x)\big)$

LOUDS による演算



さっきの演算たち

ラベル付き木の LOUDS

Trie とかを作りたいならラベル付き木を扱わなければならない。
ラベル集合 A の配列 C を用意する。枝 (u, v) のラベル c を $C\big[\mathrm{bfs\_rank}(v)\big] = c$ として格納する。
x の子でラベル c を持つものを返す $\mathrm{child}(x, c)$ は，子を全部見る，もしくはソートしておいて 2 分探索するなどで計算できる。*3
情報を単に並べているだけなのでこれは簡潔な構造。

3 キャッシュに乗るのでそこそこ速い