Gensparkで作成した大学教員が実際に作る資料の雰囲気やスタイルを再現した見本

2026年度前期 データ構造とアルゴリズム 第5回

二分探索木とバランス木（Binary Search Trees & Balanced Trees）

○○大学 情報科学研究科
教授 山本 健司
配布資料 No.05

1. 本日の到達目標とキーワード

【到達目標】
(1) 二分探索木(BST)の探索・挿入・削除アルゴリズムの挙動と計算量を説明できる。
(2) 木のバランスが崩れる問題点と、回転操作によるバランス維持(AVL/赤黒木)の概念を理解する。

【キーワード】 BST条件, 中順走査(Inorder), 計算量O(log n), 回転(Rotation), AVL木

2. 二分探索木（BST）の定義と特性

定義（BST Property）:
任意のノード x について、
- xの左部分木内の全てのキー ≤ x.key
- xの右部分木内の全てのキー ≥ x.key

特性: 中順走査(Inorder Tree Walk)を行うと、キーが昇順に出力される。

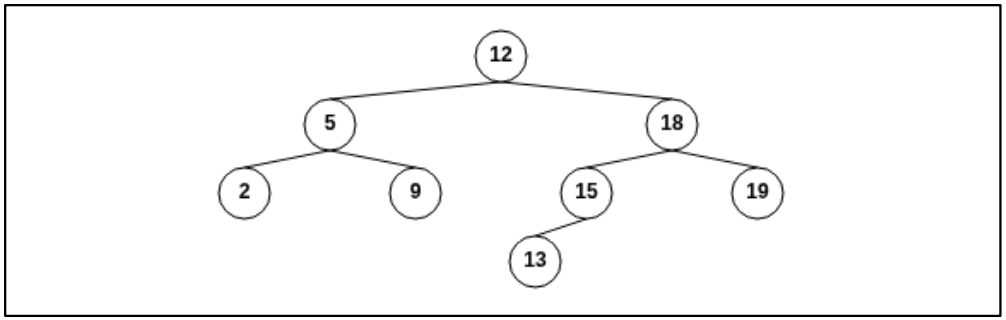
擬似コード: 挿入操作（Iterative）

```
BST-Insert(T, z) {
 y = NIL;
 x = T.root;
 while (x != NIL) {
 y = x;
 if (z.key < x.key) x = x.left;
 else x = x.right;
 }
 z.p = y;
 if (y == NIL) T.root = z; // Tree was empty
 else if (z.key < y.key) y.left = z;
 else y.right = z;
 }
```

計算量:
探索・挿入・削除の計算量は、木の高さ *h* に比例する。
- 平均: *O(log n)*（ランダムにデータが来る場合）
- 最悪: *O(n)*（すでに整列されたデータが来ると線形リストになる → バランス木の必要性）

3. 演習問題：BSTの構築

問題: 以下の数列を順に挿入して空の二分探索木を構築せよ。
入力: `{ 12, 5, 18, 2, 9, 15, 19, 13 }`



4. バランスの維持：回転（Rotation）

木の高さが *O(n)* になるのを防ぐため、挿入・削除後に局所的な操作「回転」を行って高さを *O(log n)* に保つ。
右回転 (Right Rotate): 左の子を持ち上げ、自分を右の子にする。
左回転 (Left Rotate): 右の子を持ち上げ、自分を左の子にする。

```
          [ y ] Right Rotate [ x ]
            / ¥ ----------> / ¥
          [ x ] c <---------- a [ y ]
            / ¥ Left Rotate / ¥
              a b b c
```

まとめ & 次回予告

・BSTは平均的には高速だが、偏ると遅くなる。
・AVL木や赤黒木は、回転操作により常に高さを対数オーダーに保つ。
次回： ヒープ構造と優先度付きキュー（Priority Queue）**※課題#3 提出締切厳守**

※ 実在の研究者や研究内容とは一切関係ありません。