木構造

図１において，○で示しているものをノード，線で示しているものを枝と呼びます．一番上の○は大元のノードであり，根と呼ばれます．ノードには親子関係があり，図の X に着目すると，その下流には二つのノードがあり，これはXに対しての子です．また，Xに対しての上流にもノード（この例では根になっている）がありますが，これは X に対しての親と呼ばれるノードになります．最下流のノードは葉と呼びます．根からどれくらい離れているかはレベルで表され，根のレベルを 0 とし，枝を一つ下流にたどるとレベルは一つ増加します．木の中にあるノードに着目すると，そこから下流の部分も木構造となります．このような木の一部分である木を部分木と呼び，この図では四角で囲んだ部分は X を根とする部分木です．

２分探索木

子の数が 2 以下であり，左の子，右の子と呼ばれる子の区別がある場合を２分木といい，全てのノードに対して「その左部分木のノードのキー値は，そのキー値より小さく，その右部分木のノードのキー値は，そのキー値より大きい.」という条件を満たす２分木を２分探索木と呼びます．２分探索木はバランス良く構築されている場合，探索を高速に行うことができます．以降では２分探索木の操作について述べていきます．

※２分探索木は同一キー値を持つノードが複数存在することはありません．

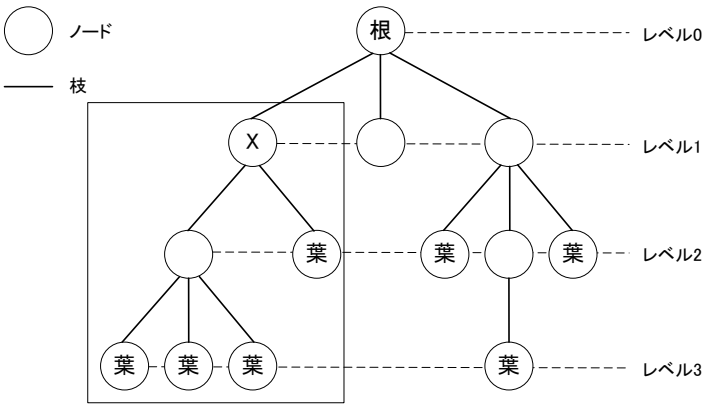


図１　木

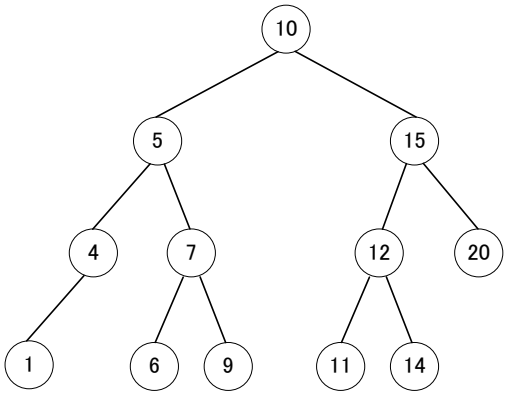


図２　２分探索木

ノードの探索

２分探索木の探索例を図３，４に示します．根に着目し，目的とする値の方が小さければ左の子ノード，大きければ右の子ノードとたぐっていくことによって実現します．

(a)3を探索

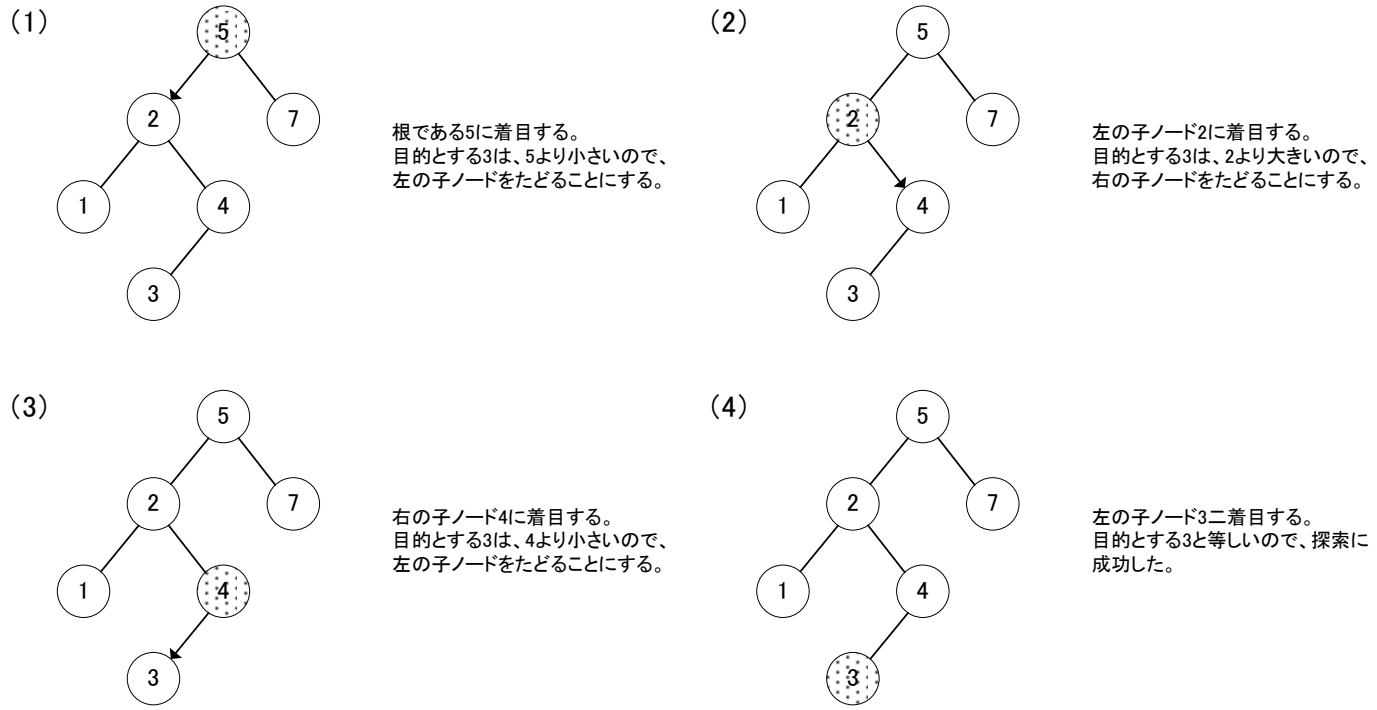


図３　２分探索木からのノードの探索（探索成功）

(b)8を探索

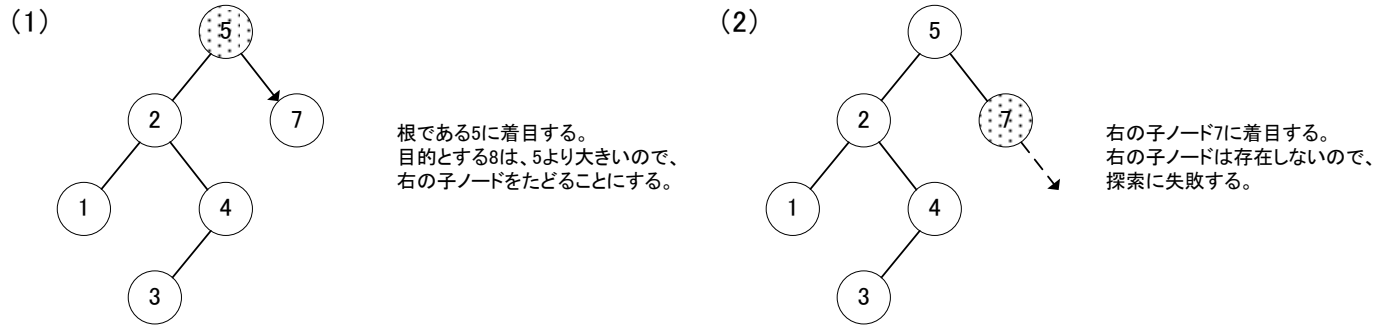
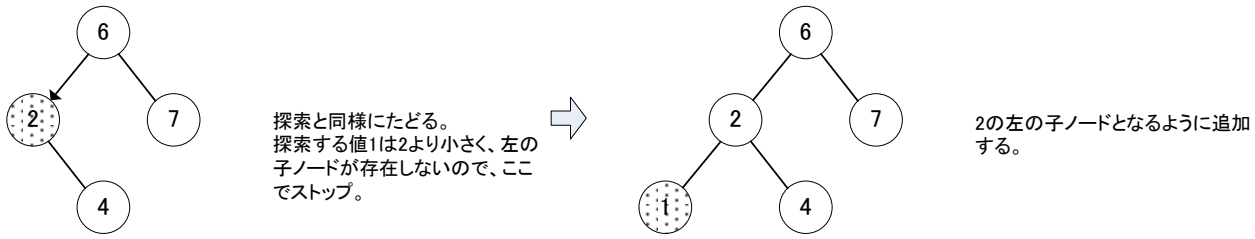


図４　２分探索木からのノードの探索（探索失敗）

ノードの挿入

ノードを挿入するためには，そのキー値をもっているノードが存在するかどうかを探索し，存在しないときのみ挿入を行います．図５（a）は２分探索木に１を挿入する手順です．最初に探索と同じ方法でノードをたどると 2 のノードでストップし，その後，このノードの左の子ノードとして挿入することになります．1 を挿入した状態でさらに 5 を挿入する手続きを図５（b）に示します．4 のノードの右の子として挿入しています．

(a)1の挿入



(b)5の挿入

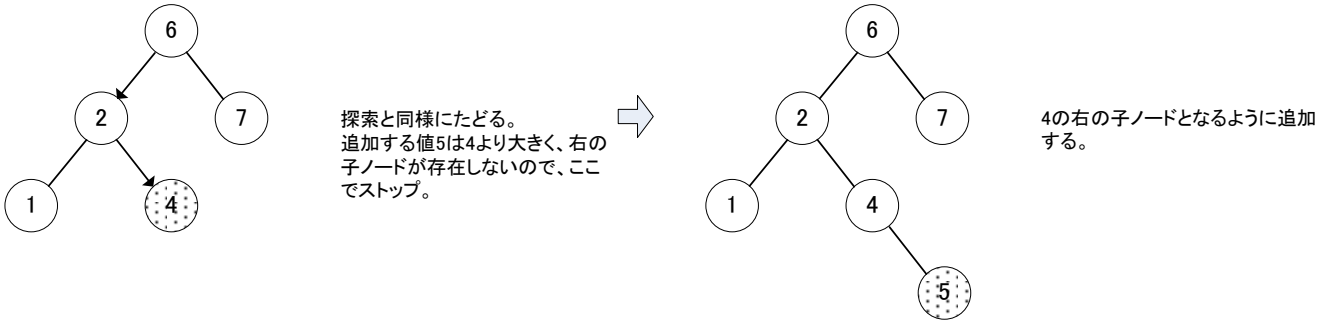


図５　２分探索木への挿入

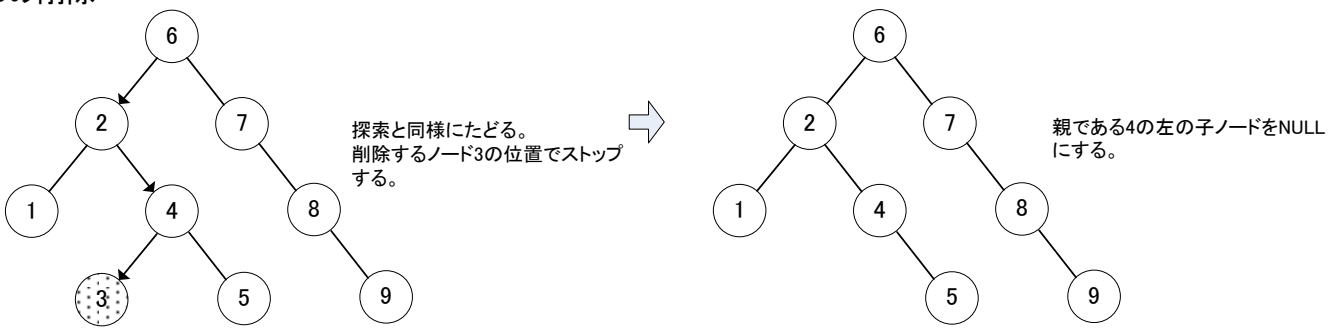
ノードの削除

ノードの削除は以下の３つに分けて考えます.

（1）子ノードをもたないノードの削除
（2）１つだけ子ノードをもつノードの削除
（3）２つの子ノードをもつノードの削除

基本的には探索を用いて削除したいキーを捜し，（1）から（3）の場合に分けて，２分探索木が維持できるようにノードの連結を変更します．そのため，（3）の処理についてはやや複雑です.

(a)3の削除



(b)9の削除

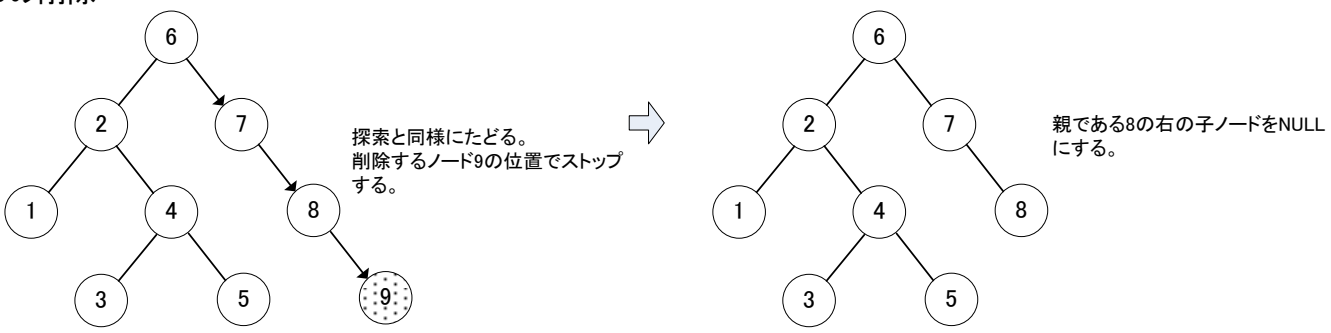
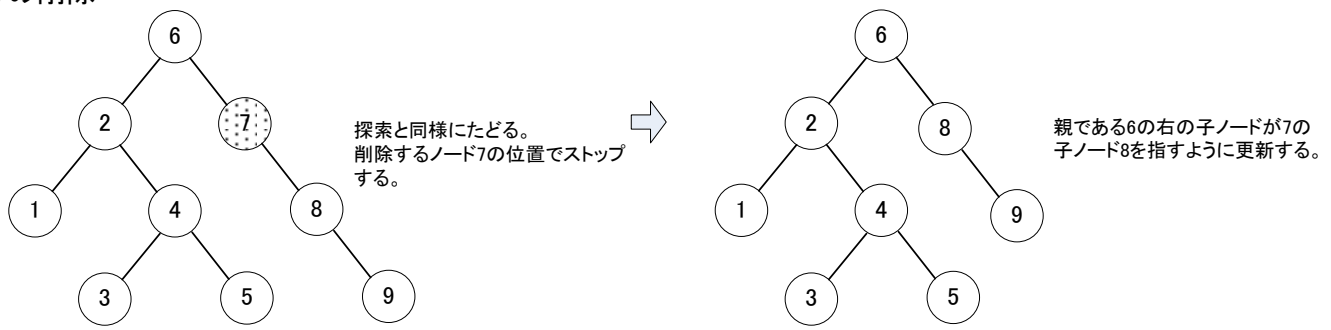


図６　子ノードをもたないノードの削除

(a)7の削除



(b)1の削除

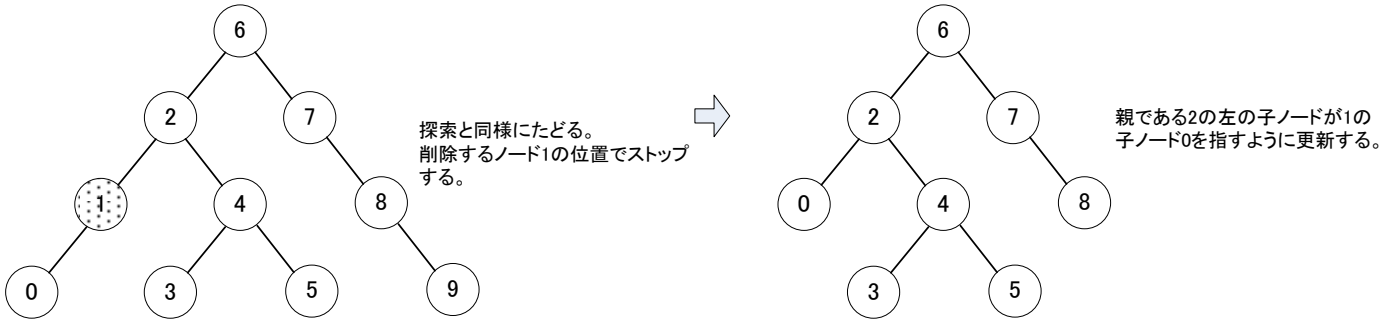


図７　１つだけ子ノードをもつノードの削除

(b)5の削除

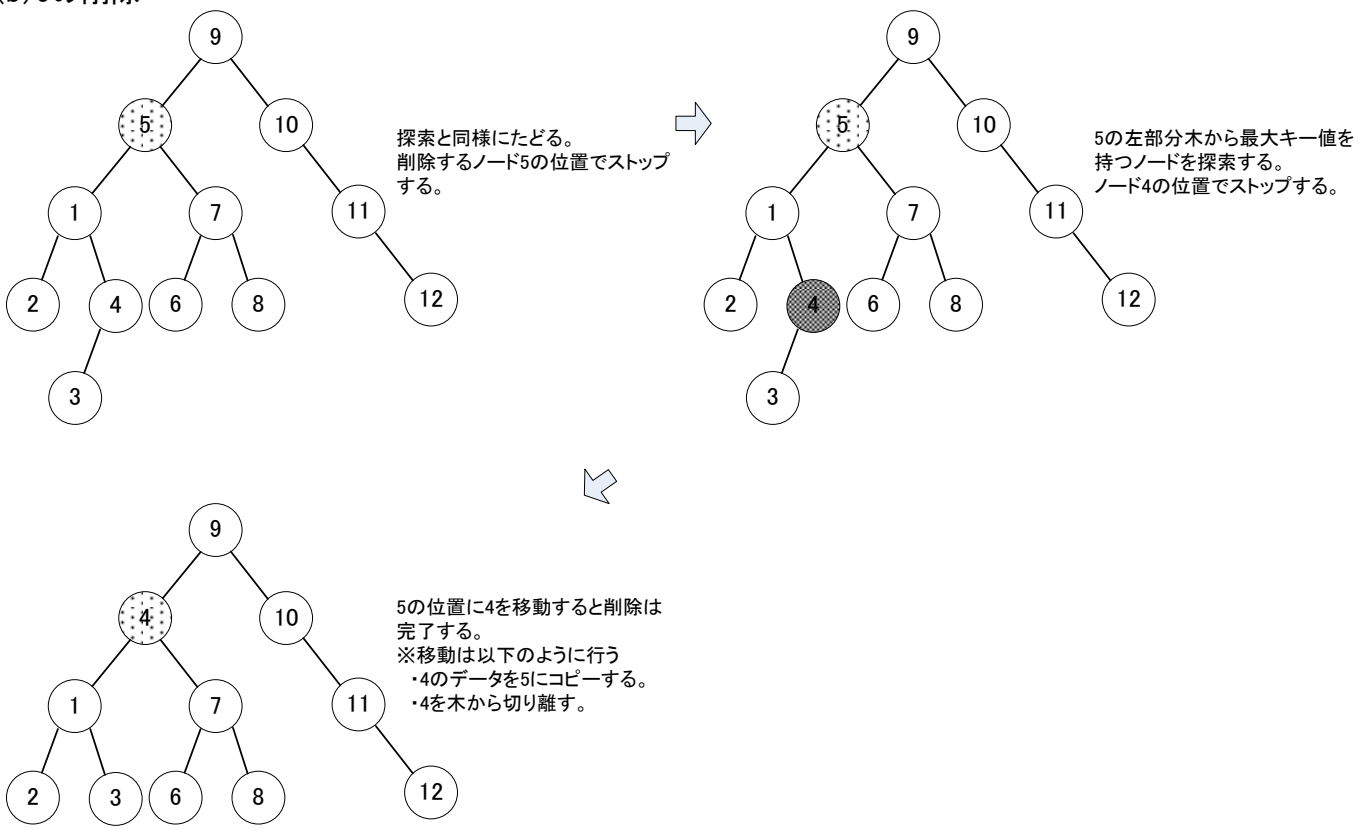


図８　２つの子ノードをもつノードの削除

課題６

木構造を用いたオリジナルのプログラムを作成しなさい.

プログラム例：

２分探索木の構成は与えられたデータ順によって，図９のようになってしまいます．そこで，図１０のようにバランスの良い２分探索木に再構成するプログラムを作成してください．上手く再構成できたかも確認して下さい.

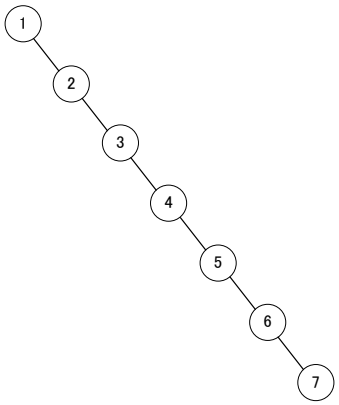


図９　バランスの悪い２分探索木

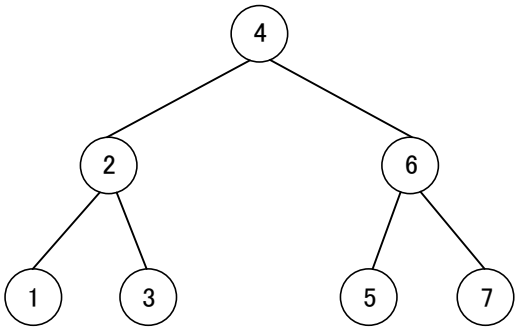


図１０　バランスの良い２分探索木