



アルゴリズムとデータ 構造

第6回探索のためのデータ構造(2) 木の巡回(なぞり)

第6回探索のためのデータ構造(2)

木の巡回(なぞり)

■ 今日の内容: 次を学ぶ

- 木の巡回(traversal)
- 再帰手続きを用いた木の巡回の実現方法
(簡単で美しいので有名)

■ ポイント

- 与えられた木に対して、その先行順(preorder)、中間順(inorder)、後行順(postorder)の番号列を求められる様になろう
- 再帰手続きを用いて、木の巡回のプログラムが書けるようになるろう

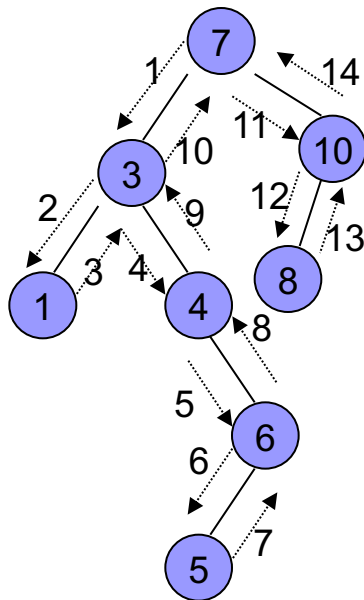
木の巡回(第5回探索(1)のスライド)

木の巡回* (traverse)とは、

木のすべての節点を組織だった方法で訪問すること

*)木の「なぞり」ともいう

深さ優先探索(depth-first search)による木の巡回



すべての節点の要素を次の3種類の順番で出力できる。

先行順(行きがけ順、前順、preorder)の巡回

最初の訪問時に出力

7, 3, 1, 4, 6, 5, 10, 8

中間順(通りがけ順、中順、inorder)の巡回

左の部分木のなぞりが終わった後に出力

1, 3, 4, 5, 6, 7, 8, 10

後行順(帰りがけ順、後順、postorder)の巡回

最後の訪問時に出力

1, 5, 6, 4, 3, 8, 10, 7

2分探索木を中間順出力すると
整列された要素のリストが
えられる！

木の巡回 (traversal)

- 与えられた根付き木Tの全頂点をちょうど一度ずつ訪問すること
 - 各頂点がちょうど一度ずつ現れるような、Tの頂点のリストのこと。(今後はこの意味)
- 次の用途に用いられる
 - 木構造として表現されたデータの処理
 - 人工知能分野における探索
 - 数式処理と言語処理
 - グラフアルゴリズムの解法

クイズ：迷路の通り方

- 問：今、あなたが巨大迷路の中（一番奥の宝の部屋）にいる。どうやったら、出口まで出られるか？

- こたえ？

- ☐ でたらめに歩き回る
- ☐ 分かれ道に来たら、かならず決めた方を選ぶ
- ☐ 。。。
- ☐ ？？？

クイズ：迷路の通り方

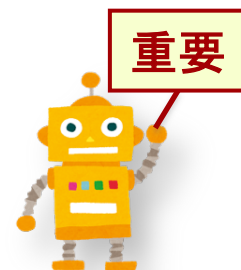
- 問：今、あなたが巨大迷路の中（一番奥の宝の部屋）にいる。どうやったら、出口まで出られるか？

- こたえ？

- 方法：右手を壁につけたまま、手を壁から離さないようにして歩く。
 - 結果：すると、必ず出られる*
なぜ？

*) 迷路の大きさに限
りがあるなら

木の巡回の種類



■ 巡回の方法:

- 基本は、根から出発して、全ての頂点を訪問する。

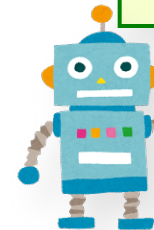
■ 木の探索には大きく分けて、2種類ある

□ 深さ優先探索 (**DFS**, Depth-First Search)

- 今いる場所からできるだけ深い方へいく
- DFSで得られる巡回は、先行順、中間順、後行順の3つ

□ 幅優先探索 (**BFS**, Breadth-First Search)

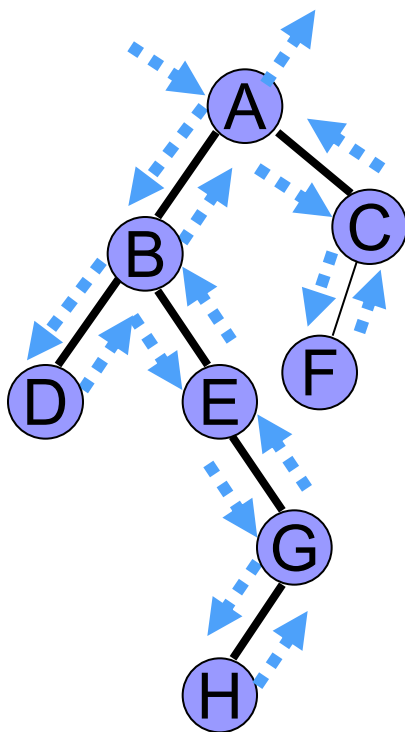
- 深さ(レベル) $d = 0, 1, 2, \dots$ を大きくしていく



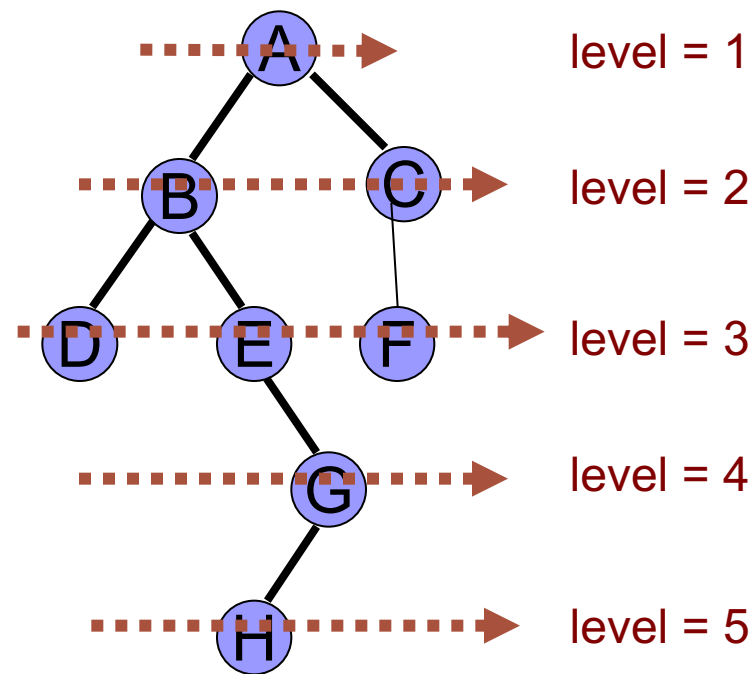
木の探索

質問:「二つの探索
の違いは何か？」

深さ優先探索 (DFS)



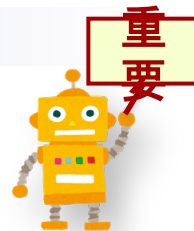
幅優先探索 (BFS)



注意: DFSだけでは、頂点の出力の順番は決まらない ⇒ 先行順、中間順、後行順

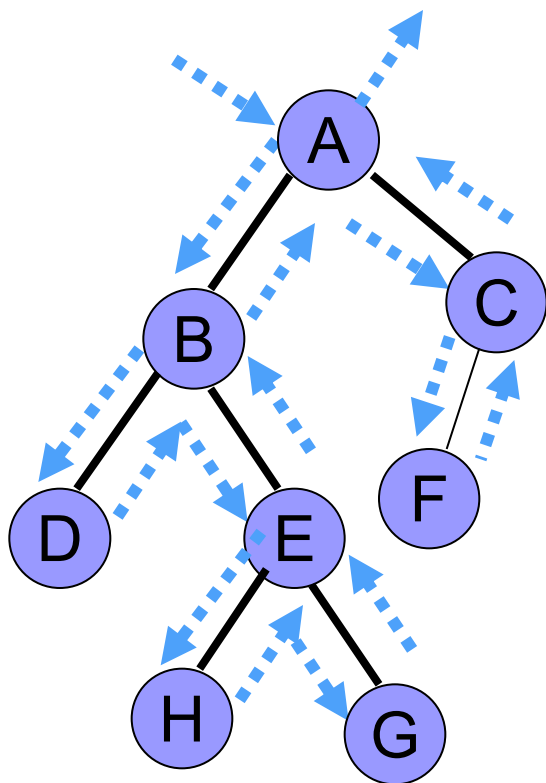
木の巡回の種類

- 深さ優先探索で、木の巡回 (traversal = 全頂点リスト) を出力するには、次の3つの方法がある
- 先行順 (行きがけ順、前置順、preorder)
- 中間順 (通りがけ順、中置順、inorder)
- 後行順 (帰りがけ順, postorder)



先行順 (preorder)

- 行きがけ順、前置順ともいう
- 各頂点vを最初に訪問する時に、頂点番号を出力する
 - 内部頂点は、上から来たときに出力される。葉は訪問したときに出力される。
- 結果として、木の頂点は上から下向きに出力される

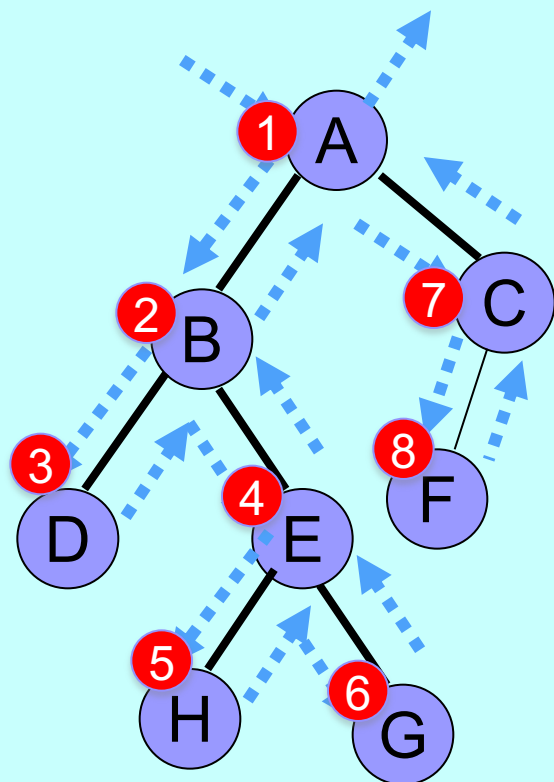


PREORDERの巡回

1 2 3 4 5 6 7 8

A, B, D, E, H, G, C, F

先行順：難しい人はこう考えよう！



PREORDERの巡回

A, B, D, E, H, G, C, F

■ ルール：各頂点 v を最初に訪問する時に、頂点番号を出力する

■ 考え方：

- 1) まず、根から全部の頂点を一筆書きで回る道を書く。
- 2) この道で**頂点の左側**（最初に頂点に来た時）に**赤丸**を書く。（中間順は下、後行順は右）
- 3) **赤丸**で頂点番号を出力する。

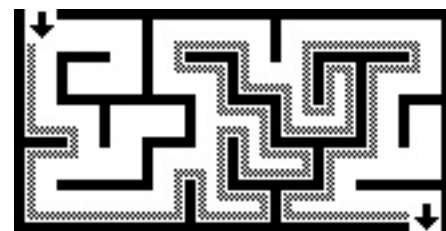
ティータイム:どんな複雑な迷路でも脱出できる方法 — 「右手法*」

- 前のページの方法は、子供向けのクイズの本にある「エジプトのピラミッド(ミノタ王の迷宮)の地下の迷路を脱出できる方法」と本質

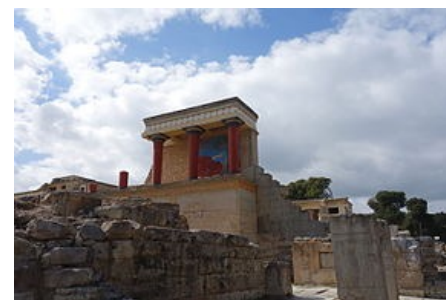
- 的の周法では、

- 迷路の壁に右手(同じ側の手)を当てながらずっと歩きます。
- 行き止まりに突き当たったら、手を壁から離さずにグルリと戻って歩き続けます。

- 迷路に出口があれば必ず出られます
(*)もちろんどちらの手でも構いません。ええ。途中で手を変えなければ。

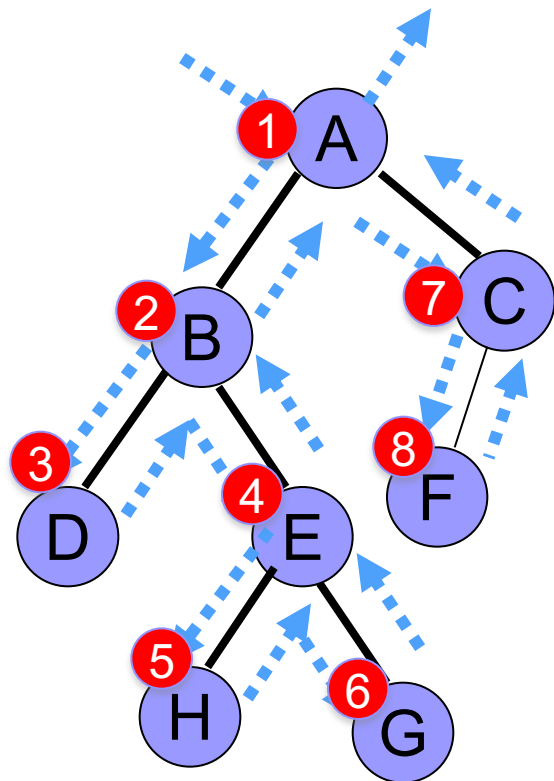


wikipedia: 迷路



wikipedia: クノッソスの宮殿(ミノタ王の宮殿)

再: 先行順 (preorder)



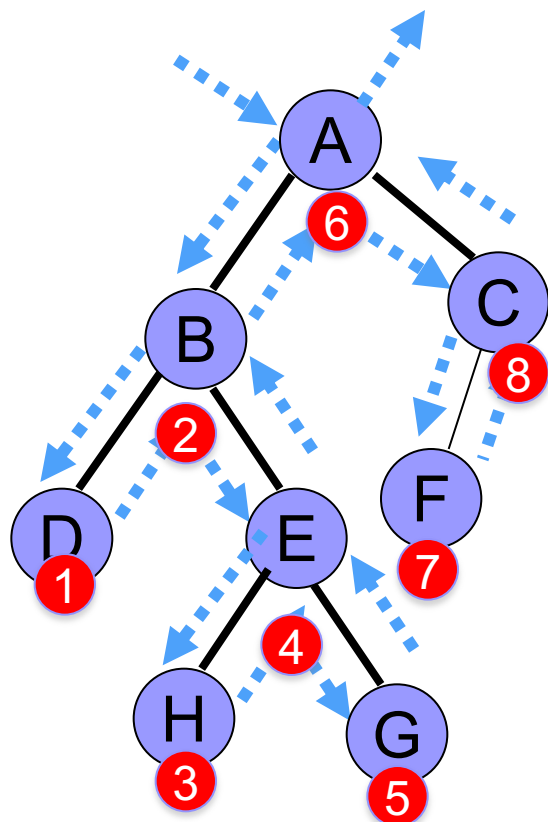
- 行きがけ順、前置順ともいう
- 各頂点vを最初に訪問する時に、頂点番号を出力する
 - 内部頂点は、上から来たときに出力される。葉は訪問したときに出力される。
- 結果として、木の頂点は上から下向きに出力される

PREORDERの巡回

1 2 3 4 5 6 7 8

A, B, D, E, H, G, C, F

中間順 (inorder)



- 通りがけ順、中置順ともいう
- 各頂点vを、(左から右へ)途中で訪問する時に、頂点番号を出力する
- 結果として、木の頂点は、左から右へ並んだ順で出力される

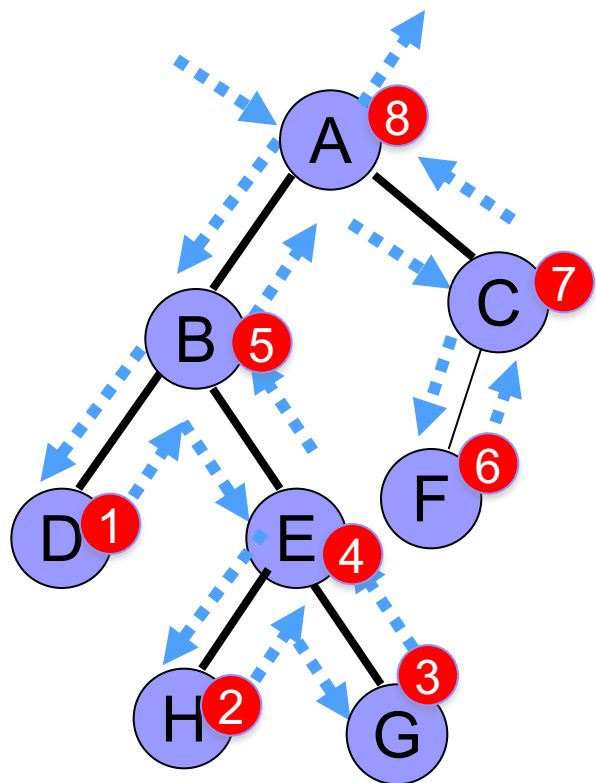
INORDERの巡回

1	2	3	4	5	6	7	8
<hr/>							
D	B	H	E	G	A	F	C



後行順 (postorder)

- 帰りがけ順、後置順ともいう
- 各頂点vを、(下から上に) **最後に訪問する時に**、頂点番号を出力する
- 結果として、木の頂点は、**下から上向き**に出力される



POSTORDERの巡回

1	2	3	4	5	6	7	8
<hr/>							
D	H	G	E	B	F	C	A

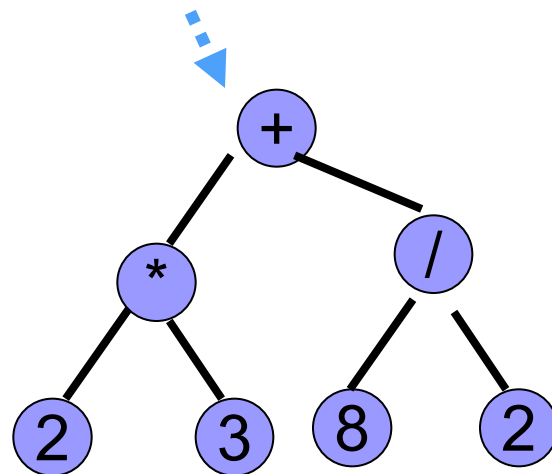
演習問題:

次の問に答えよ.

問1. 右の木Tの先行順と、中間順、後行順の巡回を書け.

問2. 先行順と、中間順、後行順の巡回を出力する再帰プログラムpreorderと、inorder、postorderを書け.

問3. 再帰を使わないでループでプログラムを書くにはどうするか?



```
//先行順巡回を出力するプログラム
void preorder(node *v) {
    vを出力する;
    preorder(v->left);
    preorder(v->right);
}

void main() {
    preorder(root) //木の根
}
```


先行順巡回のプログラム(再帰)

```
void preorder(node *v) {  
    if (v == null) return;  
    vを出力する;  
    preorder(v->left);  
    preorder(v->right);  
}  
  
void main() {  
    preorder(root)  
}
```

- 次の再帰手続きで実現できる
- 方法
 - 根からスタート
 - 頂点vに来たら、**まず出力して**、次に左と右の部分木を巡回する

中間順巡回のプログラム(再帰)

```
//中間順巡回の出力
void preorder(node *v) {
    if (v == null) return;
    preorder(v->left);
    vを出力する;
    preorder(v->right);
}

void main() {
    preorder(root)
}
```

- 次の再帰手続きで実現できる
- 先行順と似ているが、出力するタイミングが違う
 - 子供への再帰呼び出しと、再帰呼び出しの間で出力している

後行順巡回のプログラム(再帰)

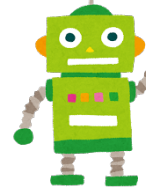
```
//後行順巡回の出力
void preorder(node *v) {
    if (v == null) return;
    preorder(v->left);
    preorder(v->right);
    vを出力する;
}

void main() {
    preorder(root)
}
```

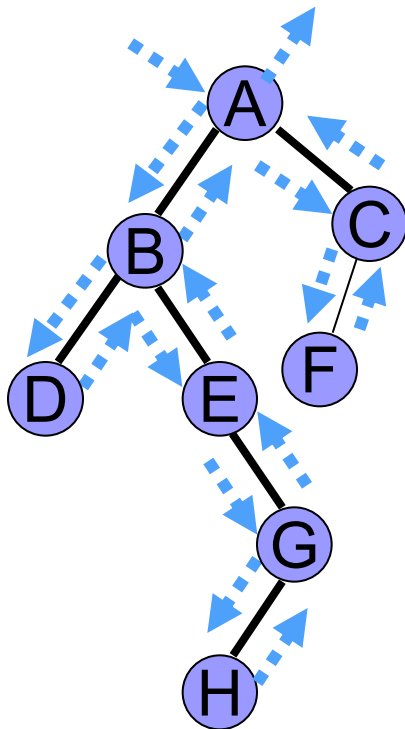
- 次の再帰手続きで実現できる
- 先行順と中間順と似ているが、出力するタイミングが違う
 - 子供への再帰呼び出しのあとで出力している

再帰を使わずにループで探索する

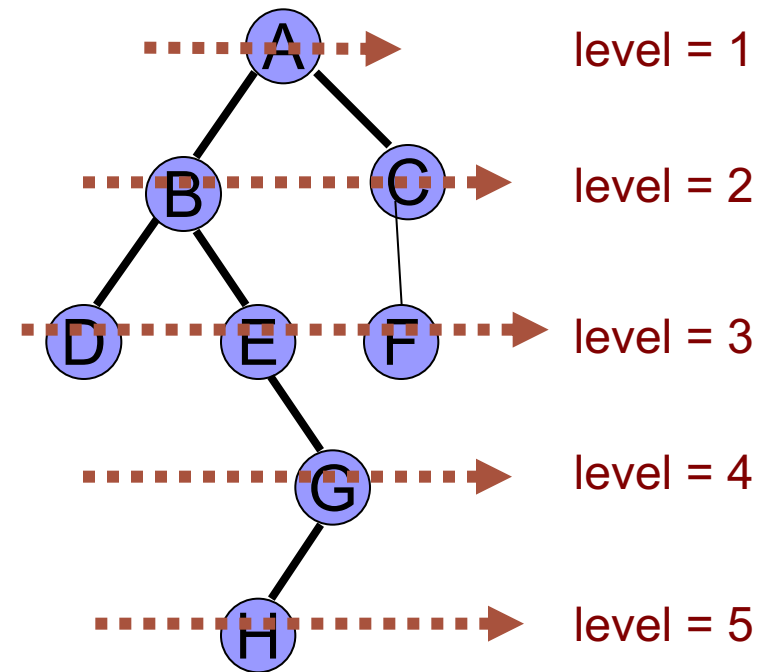
やや難しい



深さ優先探索 (DFS)



幅優先探索 (BFS)



注意: DFSだけでは、頂点の出力の順番は決まらない ⇒ 先行順、中間順、後行順

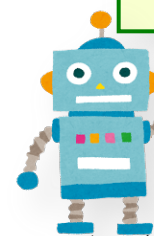
再帰を使わない巡回のプログラム

```
void main() { //preorder in DFS
    Stack S; //空のスタックS
    S.push(root);
    while (!S.isEmpty()) {
        node v = S.pop();
        if (v != null) {
            vを出力する;
            S.push(v.right);
            S.push(v.left);
        }
    }
}
```

深さ優先探索は、スタックとループを用いて実現できる

これは先行順、中間順、後行順のどれか、考えてみよう!

考えてみよう



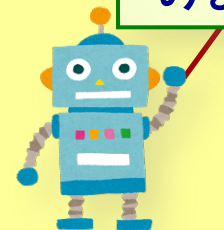
再帰を使わない巡回のプログラム

```
void main() { //BFS
    Queue S; //空のキューS
    S.enqueue(root);
    while (!S.isEmpty()) {
        node v = S.dequeue(); //要素vをとりだす
        if (v != null) {
            vを出力する;
            S.enqueue(v.left);
            S.enqueue(v.right);
        }
    }
}
```

幅優先探索による巡回は、スタックとループを用いて実現できる

なぜこれで正しいか考えてみよう!

考えてみよう



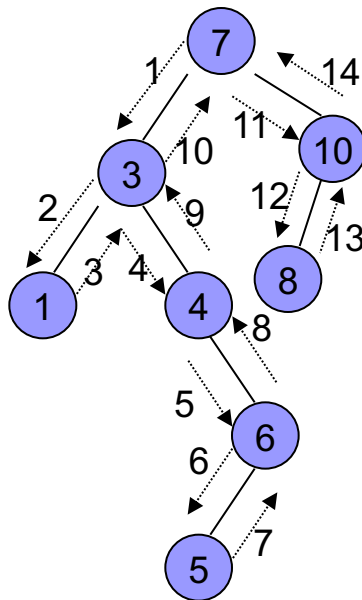
(前のページの問の答え: 先行順巡回)

まとめ: 木の巡回 (第5回探索(1)のスライド)

木の巡回(traverse)とは、

木のすべての節点を組織だった方法で訪問すること

深さ優先探索(depth-first search)による木の巡回



すべての節点の要素を次の3種類の順番で出力できる。

先行順(行きがけ順、preorder)

最初の訪問時に出力

7, 3, 1, 4, 6, 5, 10, 8

中間順(通りがけ順、inorder)

左の部分木のなぞりが終わった後に出力

1, 3, 4, 5, 6, 7, 8, 10

後行順(帰りがけ順、postorder)

最後の訪問時に出力

1, 5, 6, 4, 3, 8, 10, 7

2分探索木を中間順出力すると
整列された要素のリストが
えられる！

第6回探索のためのデータ構造(2)

木の巡回(なぞり)

今日は、二分探索木は一休みです。次回の(3)平衡二分探索木で再開します

■ 今日の内容：次を学ぶ

- 木の巡回(traversal)
- 再帰手続きを用いた木の巡回の実現方法
(簡単で美しいので有名)

■ ポイント

- 与えられた木に対して、その先行順(preorder)、中間順(inorder)、後行順(postorder)の番号列を求められる様になろう
- 再帰手続きを用いて、木の巡回のプログラムが書けるようになるろう