プロコン塾(15/11/4)
深さ優先探索

制御情報3年　宮川大樹

深さ優先探索

- 全探索の手法の一つ
- 幅優先探索と同じく、問題をグラフ（木構造）で表す
- 幅優先探索とはグラフの参照の仕方が違う

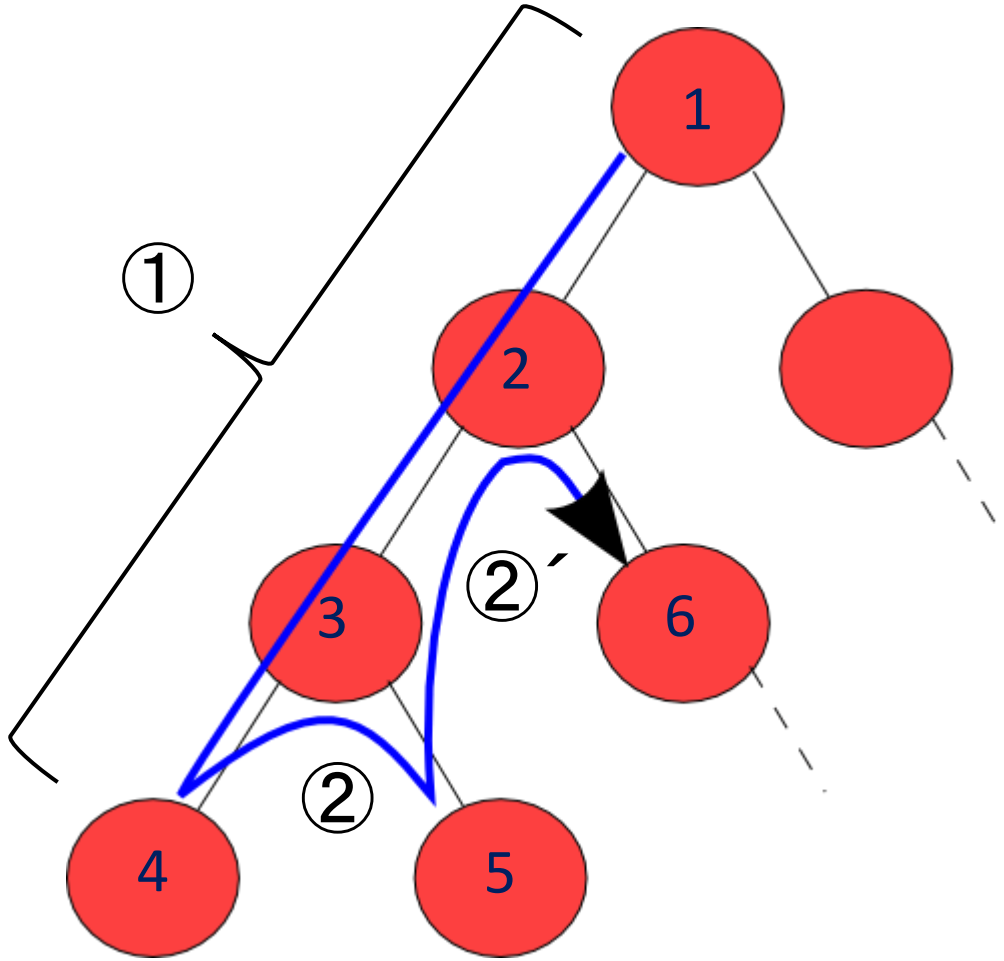
参照の方法

① 最初のノードから、目的のノードか子を持たないノードに行き着くまで、深く伸びていく。

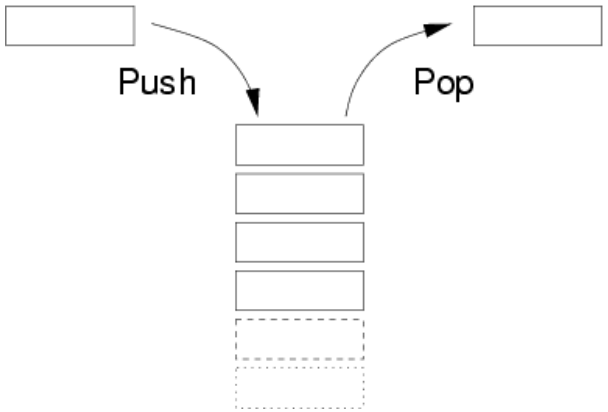
② その後は、深さをひとつ戻して未探索のノードを参照していくことを繰り返す。

木構造の参照順が一筆書きでかけることも特徴のひとつ。

※ボードゲームの例で考えると良いかもしれない



実装方法



- スタック
- 再帰関数
- for文

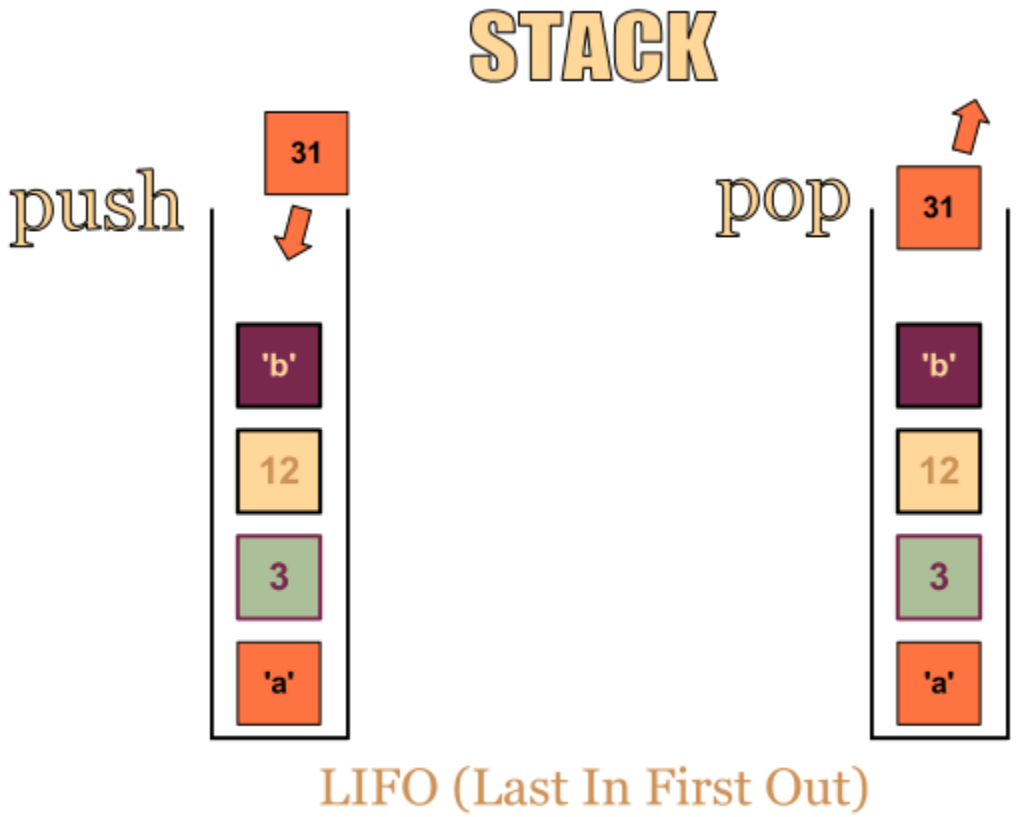
http://miffysora.wikidot.com/knapsack

```
for(int i=0;i<…) {
    for(int j=0;j<…) {
        for(int k=0;k<…) {
            ……
        }
    }
}
```

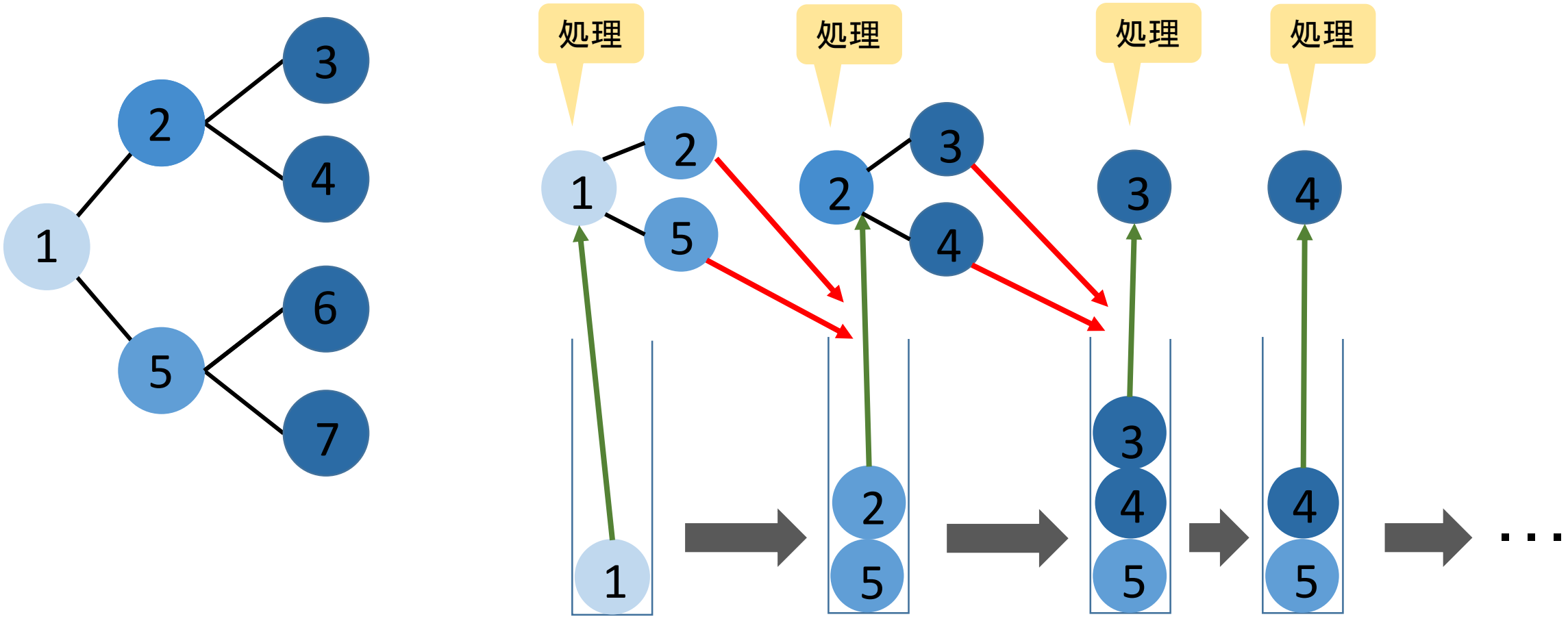
先週の内容に合わせて、今回はスタックによる実装の仕方を紹介する。

スタックとは

- データを後入れ先出しする構造(LIFO : Last In First Out)



スタックによるDFSの実装



```
Stack<E> st = Stack<E>();        //スタックを用意

E first = new E();               //最初のノードを生成
st.push(first);                  //スタックに最初のノードを入れる

while(st.size > 0) {             //スタックの中身がなくなるまで繰り返す
        E e = st.pop();          //スタックからノードを取り出す
        //取り出したノードに対して何らかの操作

        if(…) {                  //ノードが末尾でなければ子ノードをスタックに追加
                st.push(new …);
        }
}
```

- 例題

あなたは数列 A に数字を N 個持っています（ex. 1, 4, 7, …)。
ある数字 K があるとき、自分の持っている数字をいくつか足し合わせて、kを作ることが可能か答えなさい。

例. N = 4

A : 1, 2, 4, 7

K = 13

答え : Yes

- 考え方

  前回の買い物の問題とほぼ同じ。

  - 数列の数字のそれぞれについて足し合わせるか、そうしないかを考える。
  - よってひとつのノードは数列のどのインデックスを参照しているか（深さ）と、数値を加えるか否か、そしてそれまでの合計値をもつことになるだろう。

DFSの特徴

- 長所
  実装が簡単
  うまく実装すれば、メモリ効率がよい
  グラフが膨大でも解にたどり着けることがある

- 短所
  深いパスがあるとそこに時間をとられる
  解が最短とは限らない

- その他の問題もあります

http://www27.atpages.jp/tasuku/pro/2_algo.pdf

- オンラインジャッジを使用していくのもいいかも

Aizu Online Judge

http://judge.u-aizu.ac.jp/onlinejudge/index.jsp

Paiza

http://paiza.jp/