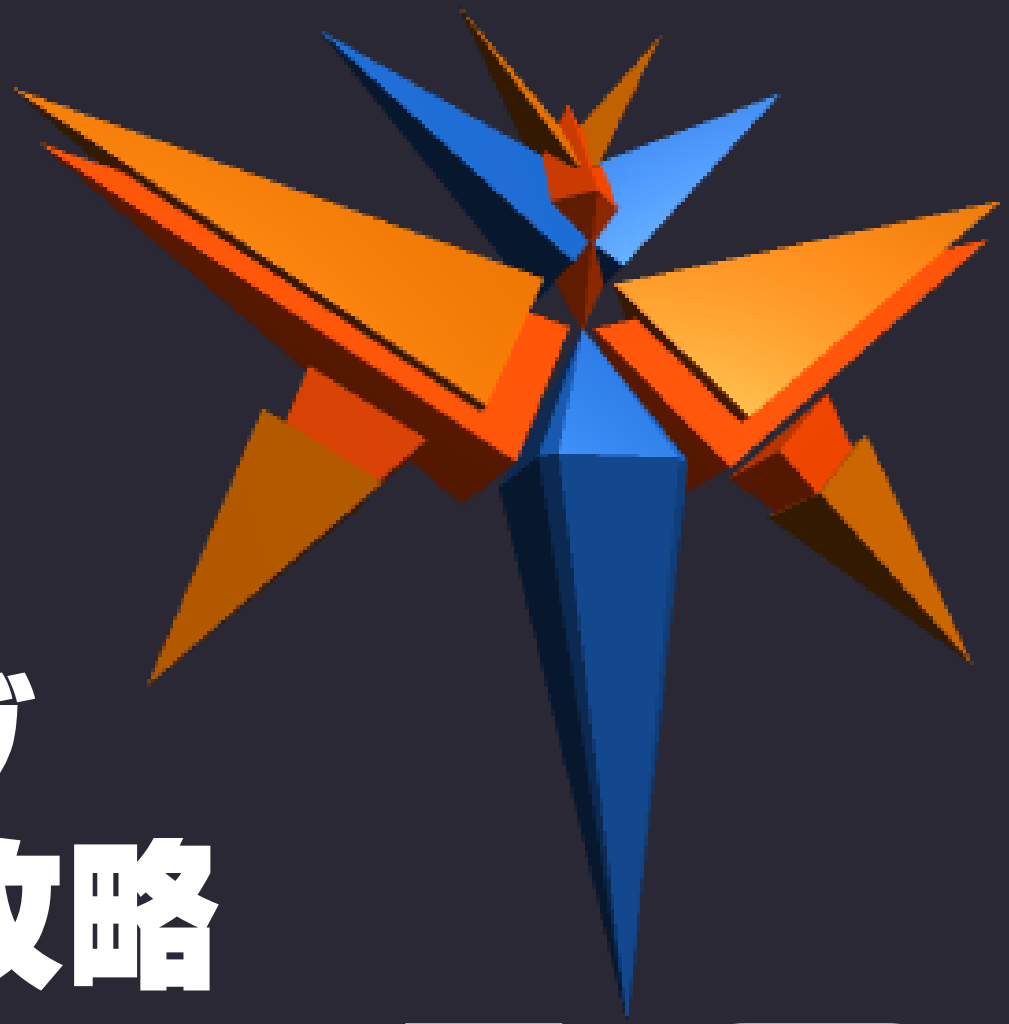


競技プログラミング
頻出アルゴリズム攻略

2016.01.13 Author: Moneto
(MATYLA-PG, @Moneto_Tk)

10

whoami

- @Moneto_Tk

- JOI出場経験あり
- 只の凡人老兵

whoami

- 同人ゲーム制作サークル MATYLA 代表

- 本業はゲームプログラム寄りの C# 書き
- 昔はC/C++/Python/Delphiとか使ってた
- 競技は 7 年前～ 4 年前くらいに人並み程度に
- JOI 本選などに出ていました

whoami

- 競技プログラミング的な話

- 高校の部活の同期に現 RedCoder が 3 人
- 故にレベル認識が時々おかしい
- 競技は 4 年くらい前に引退済み

whoami

- 昔からゲームプログラミング寄りの生活

- 最近は細々と
- コミケへのサークル参加（08年冬～）
- 各所での講演とかインストラクタとか（時々）
- 全日本学生ゲーム開発者連合での講演（時々）

競技プログラミングで必要な知識

アルゴリズムの知識

著名な問題やその定石

「変な」テクニック

「変な」テクニック

http://ichyo.jp/posts/2014-12-15-advent-calender/

- とにかく（可読性とかは置いておいて）高速に問題が解きたい人のための知識
- REPマクロとかの話はこちら

- 今日は敢えて触れません

アルゴリズムの知識

著名な問題やその定石

競技プログラミングで必要なアルゴリズムの知識

- 平易なものから難解なものまで
- 「知らなくても解ける」という例はあまりない

- 知識をつけて問題を解き、定石を学ぶ
- 大体これの繰り返し

競技プログラミングで必要なアルゴリズムの知識

- 競技にしか使えない、ということは少ない
- 実務開発でも活用できる場面が多い

- 競技プログラミングが専門ではない人でも、是非知っておいてほしい内容

今日の内容は
その中でも基本的なもの

DP
Dynamic Programming
動的計画法

BFS
Breadth First Search
幅優先探索

DFS
Depth First Search
深さ優先探索

取り扱う基準

- 基本的な内容
- 日本情報オリンピック本選出場者程度の知識
- 「これを学べば高校生に勝てるよ！」

DP
Dynamic Programming
動的計画法

DP

- 「動的計画法」
- これを使えるかどうかが日本情報オリンピック本選出場のボーダーになる程度には重要

- 「一度計算した値を覚えて再利用」
- 余計な計算をしないことで高速化・高効率化

DP

- 非常に簡単な例

- フィボナッチ数列の第n項を求める
- 書き方は最低３種類

- とりあえず何でもいいので書いてみましょう

DP

- 非常に簡単な例

- フィボナッチ数列の第n項を求める
- 書き方は最低３種類

- 1つだけ制約
- そのFib(n)、Fib(90)を計算できますか？

DP

- 非常に簡単な例

- フィボナッチ数列の第n項を求める
- 書き方は最低３種類

- 1つだけ制約
- そのFib(n)、Fib(90)を計算できますか？

DP

- 非常に簡単な例

- フィボナッチ数列の第n項を求める

- 2 つの落とし穴
- long long int でないと計算不可能
- 単純再帰では計算不可能

Tips

intの値を一瞬でも超えることが
分かった時点で
絶対にlong longを使おう

実力者でも気をつけるべきミス

DP

- 非常に簡単な例

- フィボナッチ数列の第n項を求める

- 単純な再帰：nが大きいと計算不能
- メモ化再帰：再帰の過程で値をメモしておく
- 動的計画法：計算の過程で値をメモしておく

Fib(n)

- 単純な再帰：nが大きいと計算不能

```
public static Int64 fib_rec(int n)
        {
            if (n == 0) return 0;
            else if (n == 1) return 1;
            return fib_rec(n - 1) + fib_rec(n - 2);
        }
```

Fib(n)

- メモ化再帰

```
public static Int64 fib_rec_m(int n)
        {
            if (n == 0) return 0;
            else if (n == 1) return 1;
            else if (table[n] != 0) return table[n];
            else return table[n]
             = fib_rec_m(n - 1) + fib_rec_m(n - 2);
        }
```

Fib(n)

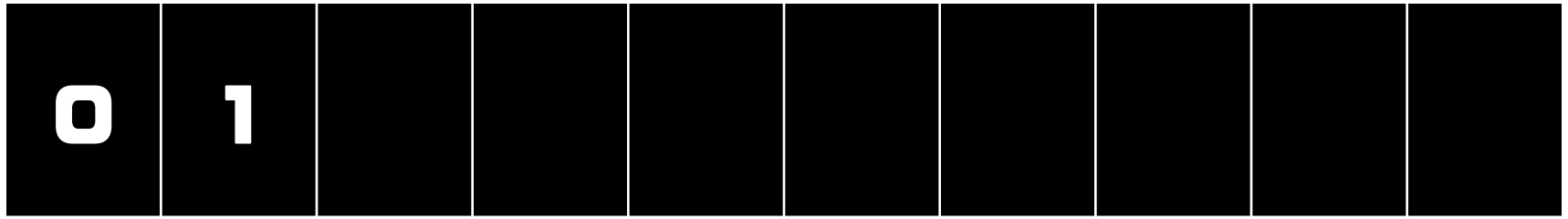
- 動的計画法

```
public static Int64 fib_dp(int n)
        {
            int[] dp = new int[n];
            dp[0] = 1;
            dp[1] = 1;
            for (int i = 0; i < n - 2; i++)
            {
                dp[i + 2] = dp[i] + dp[i + 1];
            }
            return dp[n - 1];
        }
```

- メモ用の配列を関数内で確保
- 再帰とは計算順序が異なることに注意
- これは初歩の初歩

Fib(n)

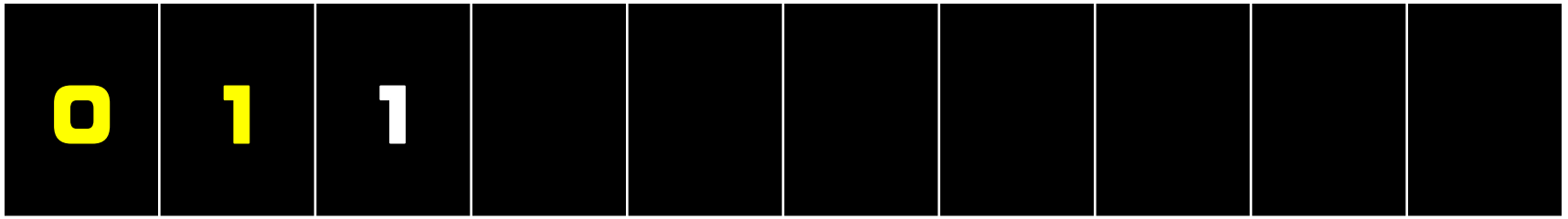
- Int64 dp[]



- 取り敢えず初期化

Fib(n)

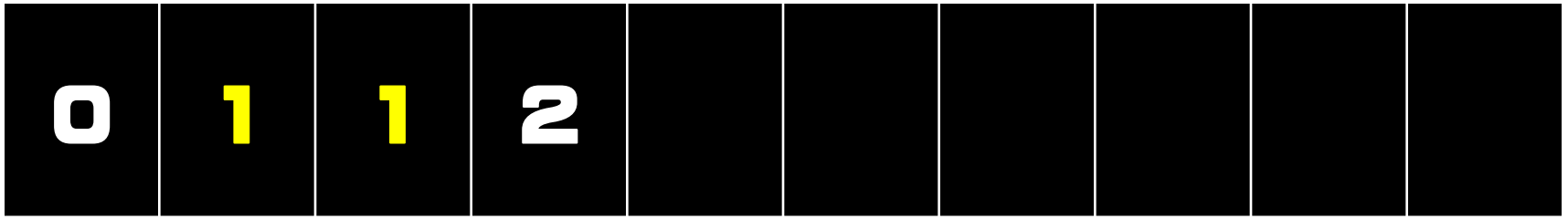
- Int64 dp[]



- dp[0]とdp[1]からdp[2]を計算

Fib(n)

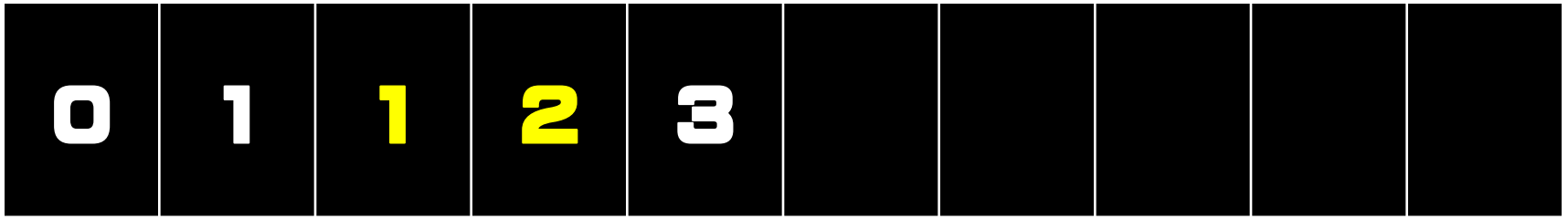
- Int64 dp[]



- dp[1]とdp[2]からdp[3]を計算
- それ以外の数値は参照しない

Fib(n)

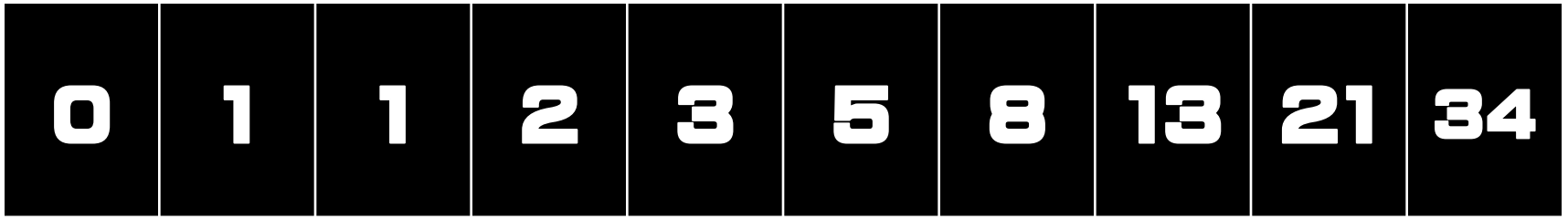
- Int64 dp[]



- それ以降も同様に進めていく
- 必要な数値は既に計算されている！

Fib(n)

- Int64 dp[]



- 線形時間で計算可能
- 同じ計算は2回以上行わない

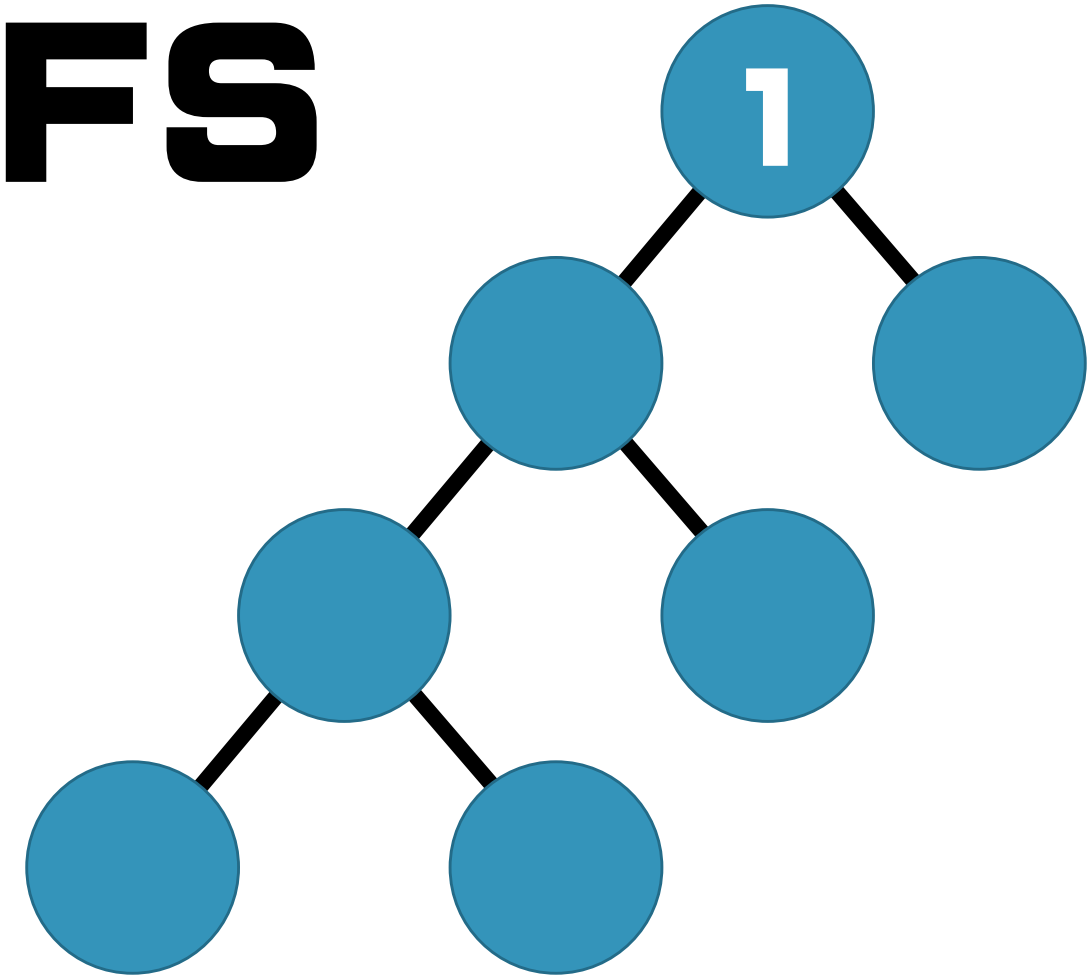
BFS
Breadth First Search
幅優先探索

BFS

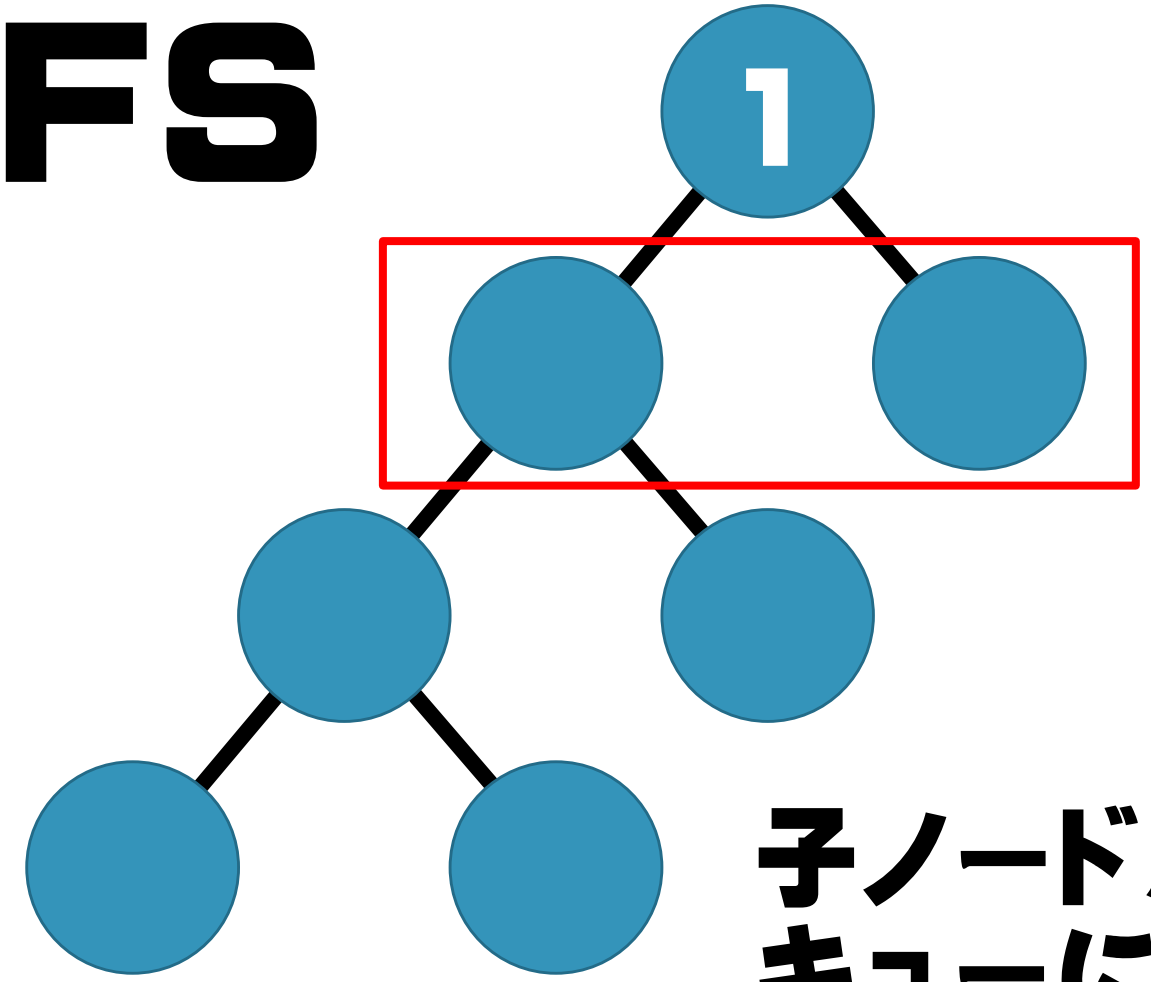
- グラフ構造の探索アルゴリズム
- 幅優先：とにかく探索範囲を広げていく

- 探索に当たり特殊な条件は考慮しない
- 最良優先探索（Best-First）とはこの点で異なる

BFS

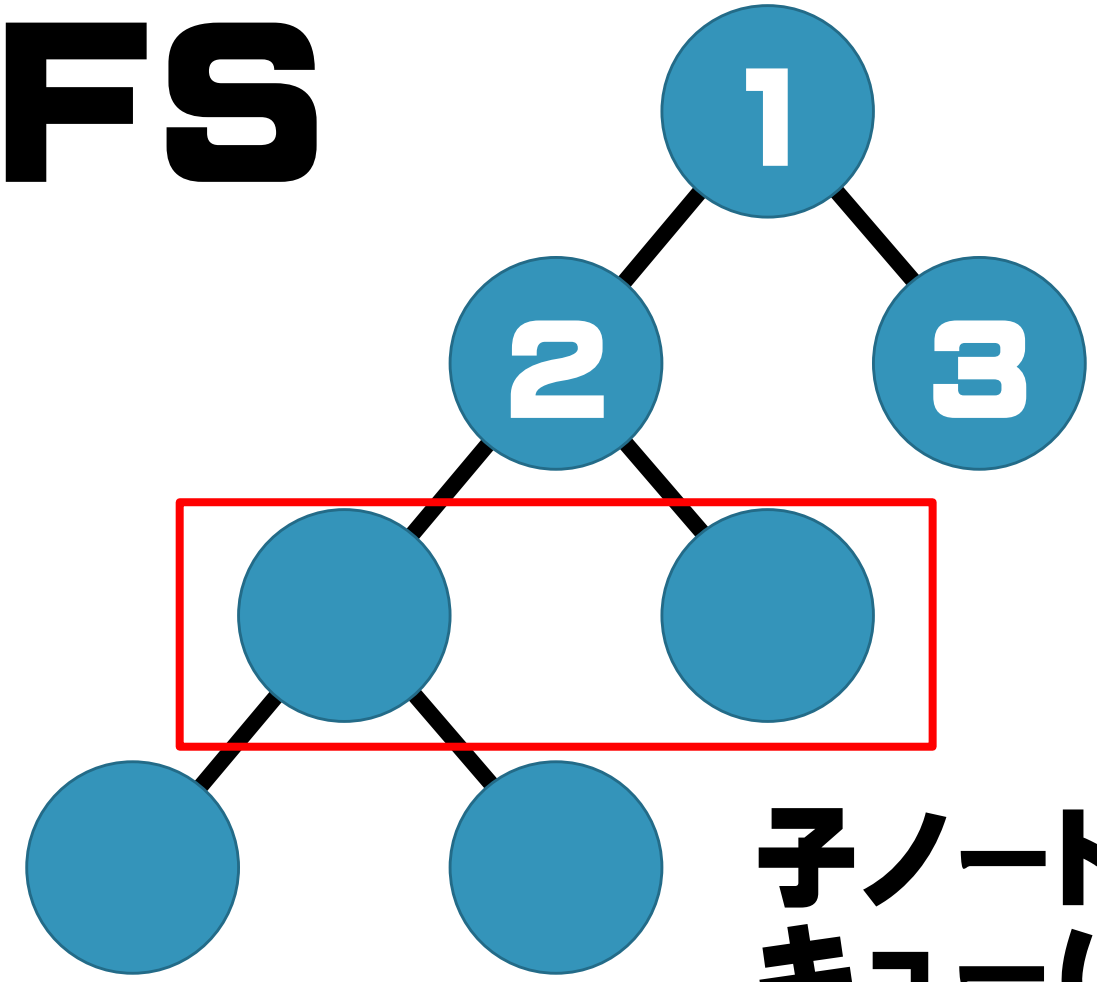


BFS



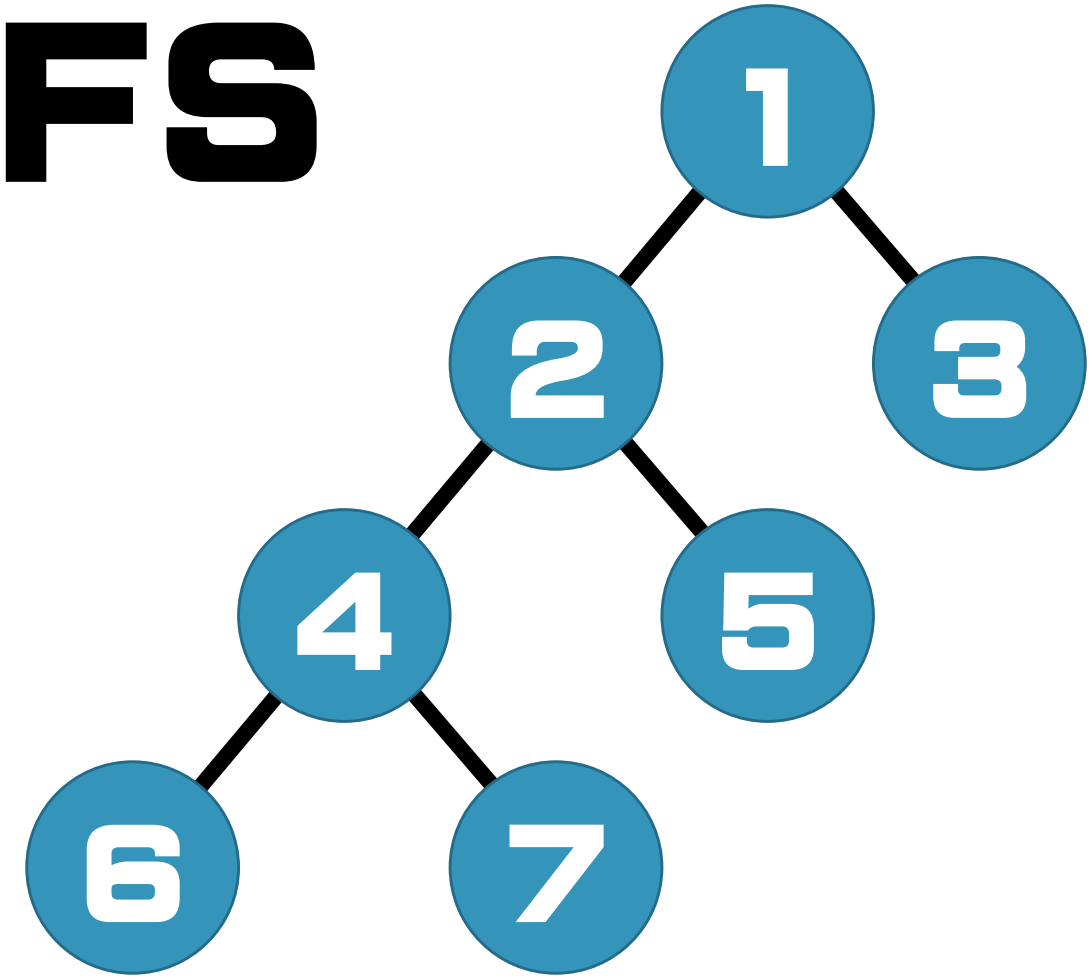
子ノードが見つかったら
キューに格納

BFS



子ノードが見つかったら
キューに格納

BFS



Best-First

- 名前が出てきたので一応説明

- 幅優先探索にヒューリスティクスを導入
- 特殊な条件（目的地との距離など）

- Dijkstra、A*法などはBest-Firstの有名な例

Best-First

- Dijkstra法、A*法
- 最短経路を求める有名なアルゴリズム

- 実務開発でも有用。何に使う？
- ネットワークのルーティング問題
- RTS等における移動経路の策定


RTS
Lunatic
Level4
COM1 (E.チーム1)
名無しの司令官 (L.チーム2)
Menu
58/110
20672
1
9248
0
14539
0
11983
http://neetpia.sakura.ne.jp/works/GensouSenryakutan/1.jpg
現在時間:13:44
FPS(描画):60.03
FPS(処理):59

RTS

行き先を指定すると
最適な移動経路を計算し
その通りに移動してくれる

考慮するべき条件
高低差（崖は迂回する）
建造物、森林、河川など

http://neetpia.sakura.ne.jp/works/GensouSenryakutan/1.jpg

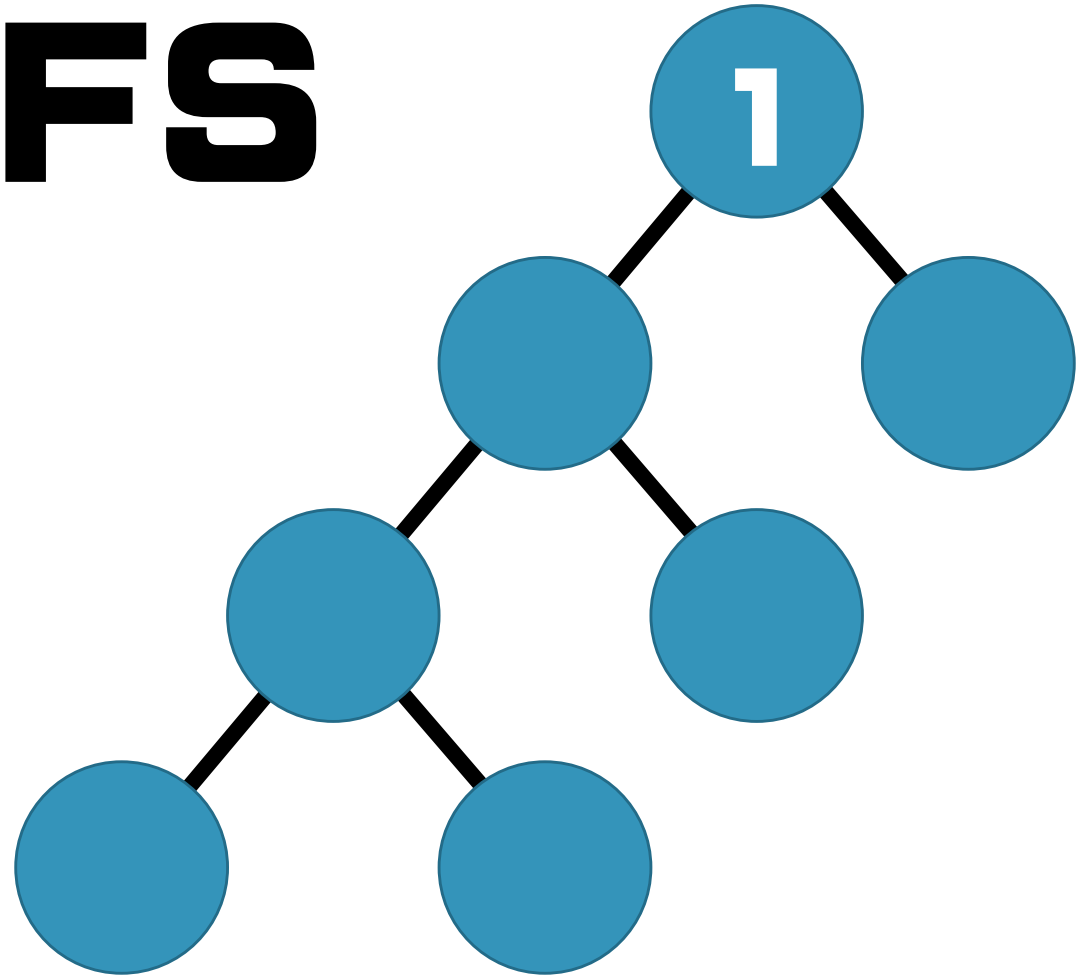
DFS
Depth First Search
深さ優先探索

DFS

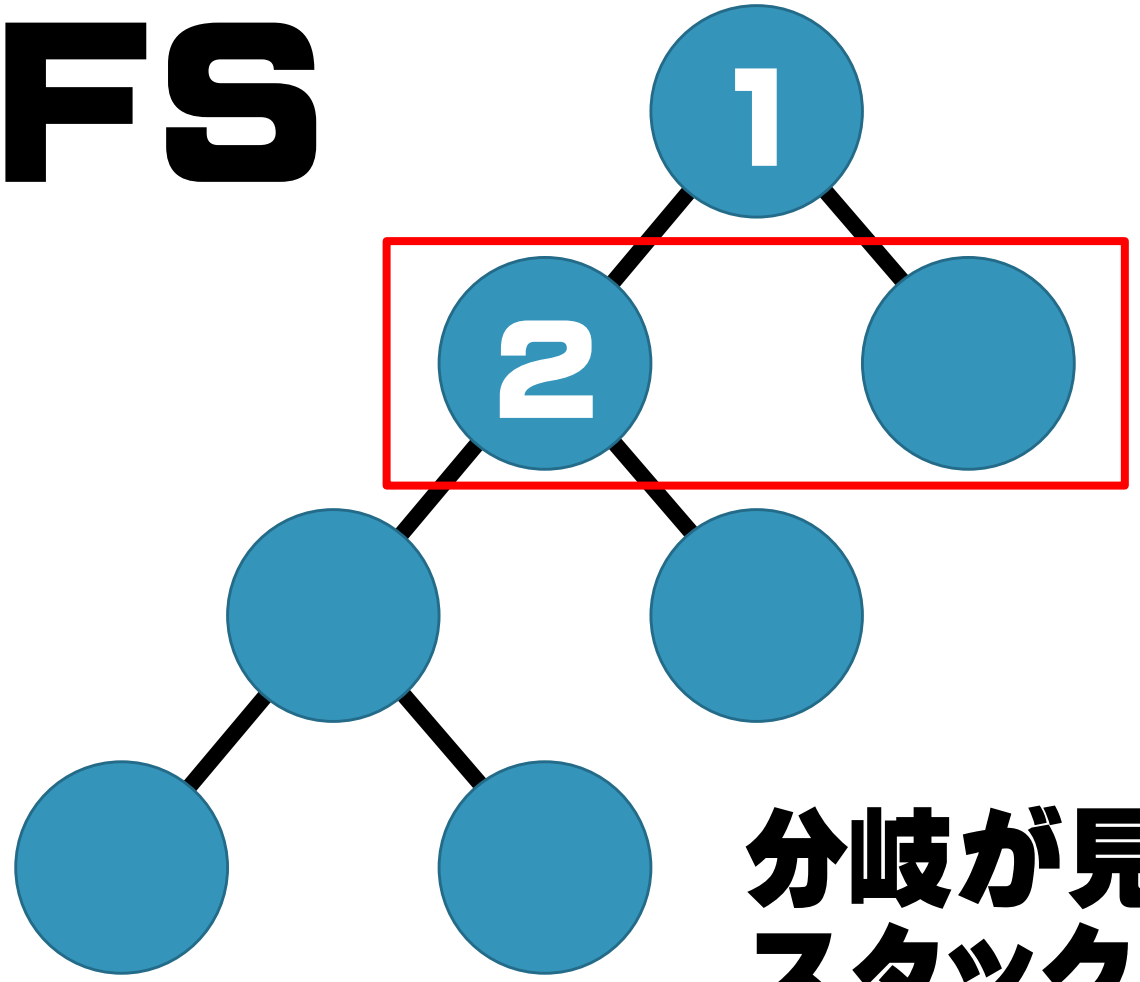
- グラフ構造の探索アルゴリズム
- 深さ優先：取り敢えず末端ノードまで探索

- JOIには出題されていない
- どちらかと云えばBFSの方が重要

DFS

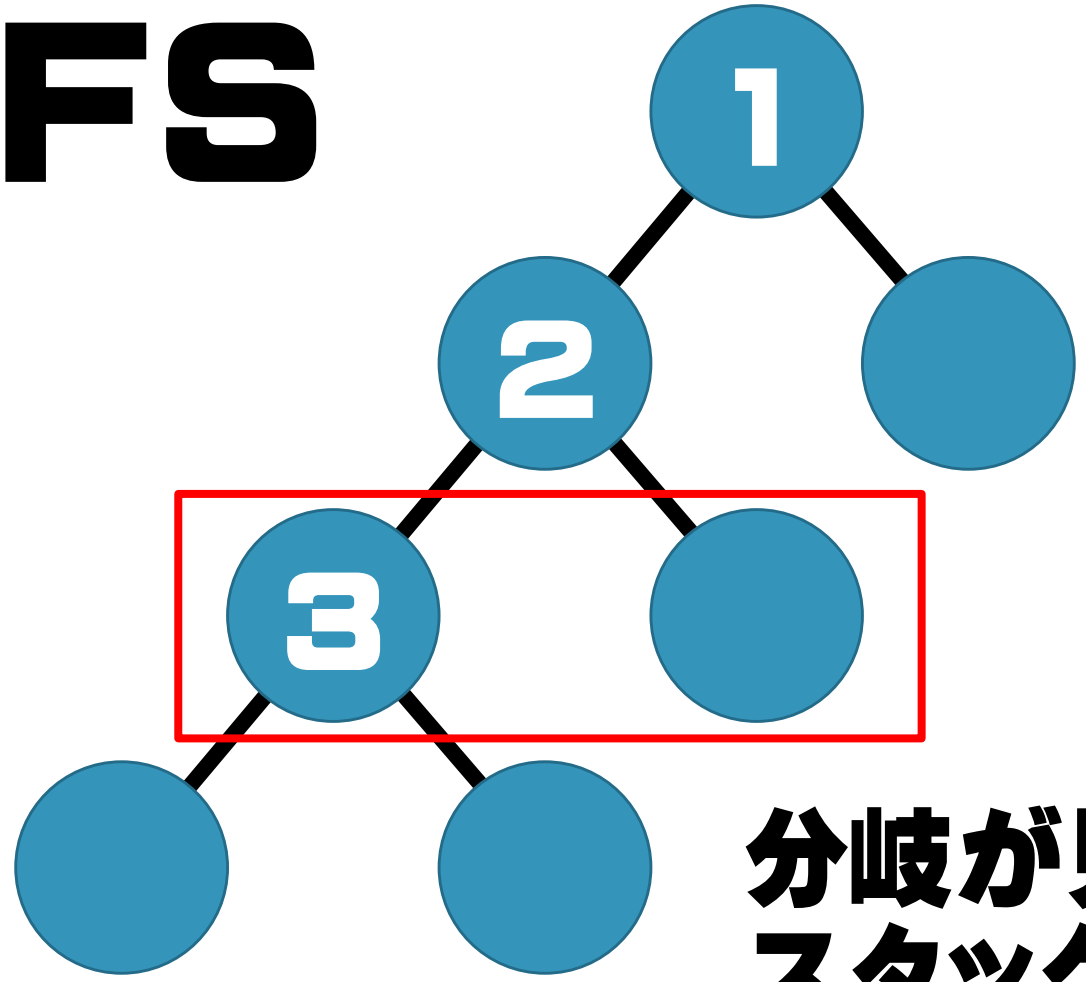


DFS



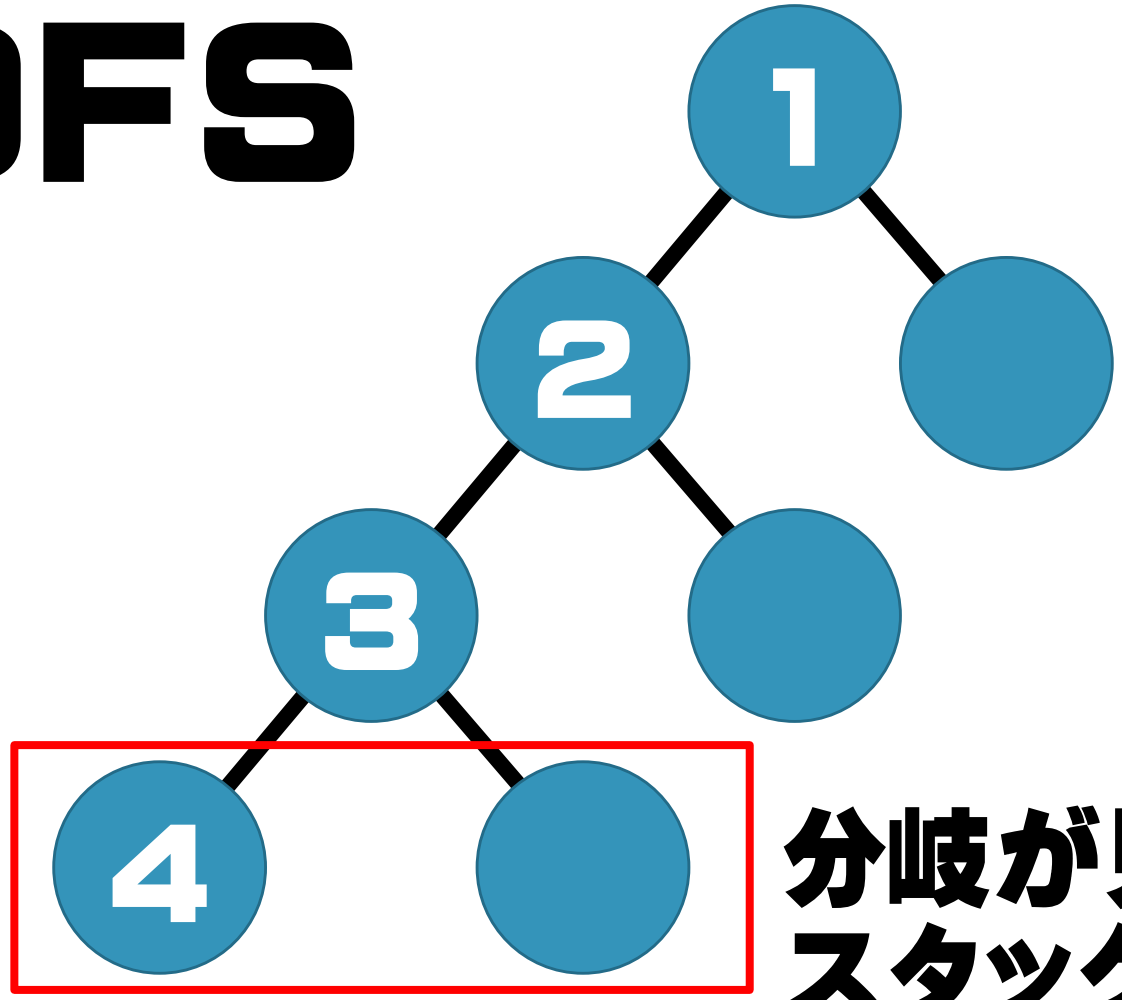
分岐が見つかったら
スタックに格納

DFS



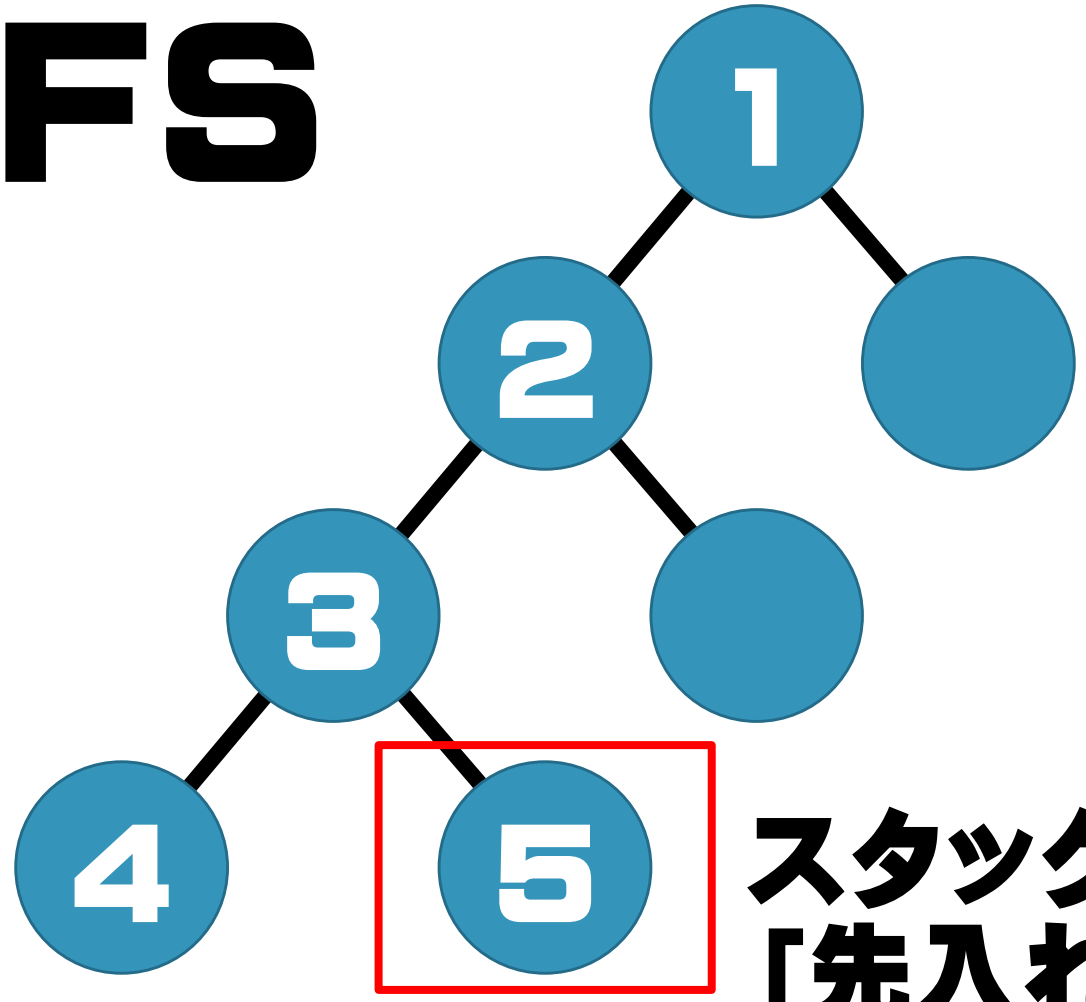
分岐が見つかったら
スタックに格納

DFS



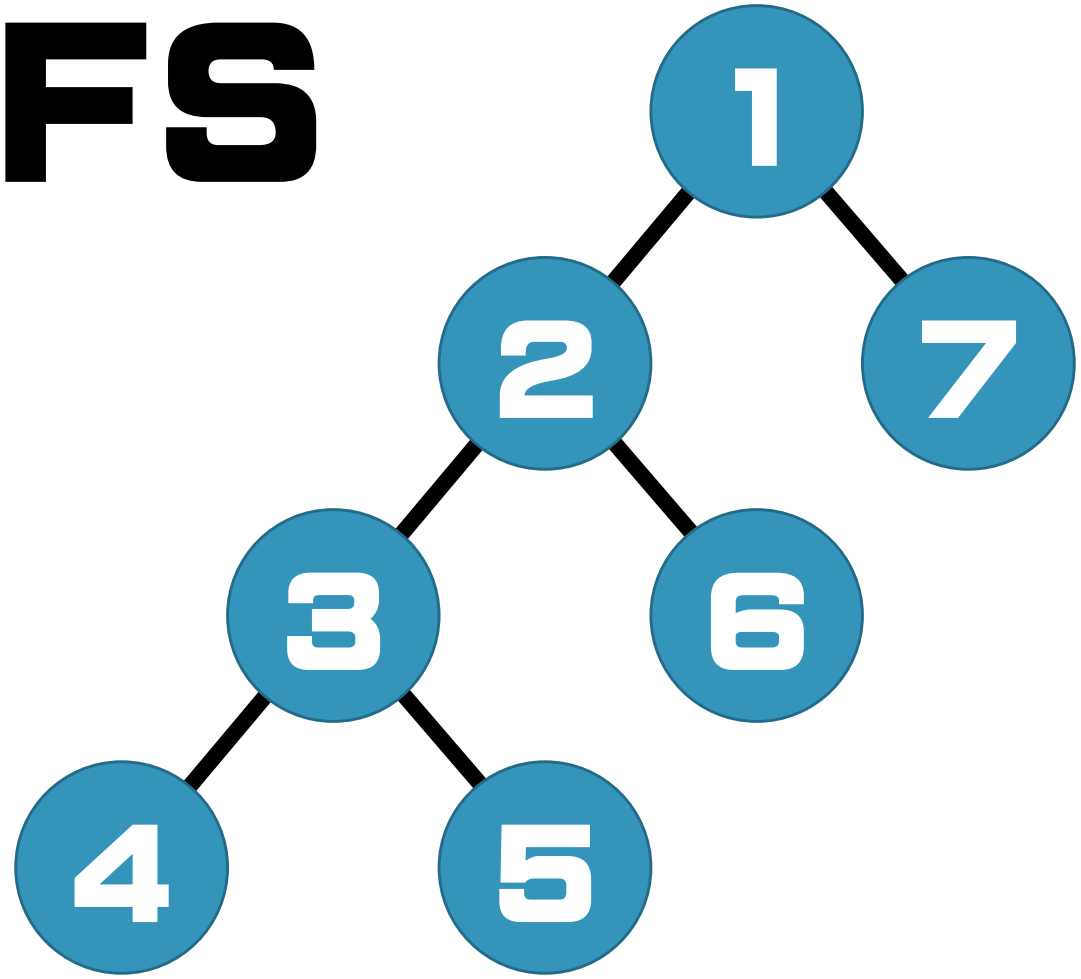
分岐が見つかったら
スタックに格納

DFS



スタックの処理順番
「先入れ後出し」

DFS



根底の考え方は
どれも簡単

手頃な練習

- AOJ #0568
- 出典：JOI2011/2012 予選 問題4 “Pasta”

- DPの簡単な練習問題
- 愚直な全探索では1セット/5セットしか解けない
- よくある落とし穴「〇〇で割った余り」

Tips

- 「〇〇で割った余り」
- 内部での累積値が〇〇を超えた時点で（超えていなくてもよいが）〇〇で割っておく
- 最後にまとめて割ろうとすると、結果が出る前に累積値がオーバーフローする場合がある

手頃な練習

- AOJ #0569
- 出典：JOI2011/2012 予選 問題5 “Illumination”

- BFSの簡単な練習問題
- 少しだけ頭を使う必要あり
- #0568と併せて完答すれば、JOI予選上位16名級

手頃な練習

- 実はAOJは結構おすすめ
- JOI/PCK等、国内の中高生向け大会の問題も収載
- これらは公式サイトに解答の方針が載っている

- 何も手がかりが無い状態より学びやすい
- 「高校生」のレベルを実感できる

Tips

- JOI出身者なら聞いたことのある話かも？

- C++を使う場合、標準入出力はstd::cin・coutよりもscanf・printfの方が高速

- printf・scanfは平易・高速、しかしフォーマットミスでバグを誘発することがあるので注意

ICPCの国内強豪は JOI出身者が多い

「JOI出身者」のレベルがいかに高いか

ちょっと

- もっと詳しく知りたい方は是非とも「蟻本」を買って色々学んでみましょう
- 2010年頃からの名著です

もうちょっと

- 日本情報オリンピック本選（2016.02.14）
- 問題は数日後にWeb上に公開されます

- 高校生の標準的なレベルと比較してみましょう

更にもうちょっと

- AOJ - http://judge.u-aizu.ac.jp/
- POJ - http://poj.org/
- TopCoder - https://www.topcoder.com/
- ICPCを始め国内外のコンテストが気になる方は是非これらのサイトで練習してみましょう

というわけで

- AOJ #0568, #0569
- 出典：JOI2011-2012 予選 問題4/5

- 1 問 45 分程度
- 上のレベルを目指すなら 1 問 30 分程度？

萌音さん@まったいら @Moneto_Tk · 23時間
「年明けたら暇」とか抜かしてた半月くらい前の自分をぶん殴りたい.jp

4:46 - 2016年1月12日 · 詳細

お疲れ様でした