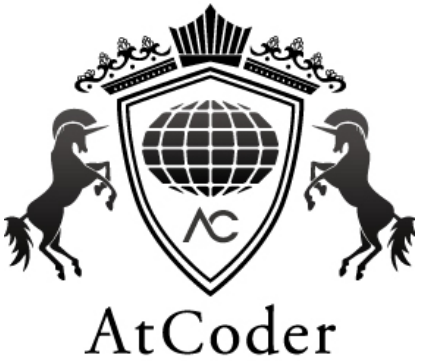
実践・最強最速のアルゴリズム勉強会
第二回　講義資料



AtCoder株式会社 代表取締役
高橋 直大

はじめに！

AtCoder

- 本講義では、ソースコードを扱います。
- 前面の資料だけでは見えづらいかもしれないので、手元で閲覧できるようにしましょう。
- URLはこちらから
  - http://www.slideshare.net/chokudai/WAP_AtCoder1
  - URLが打ちづらい場合は、Twitter: @chokudaiの最新発言から飛べるようにしておきます。
    - フォローもしてね！！！

目次

1. 勉強会の流れ
2. 競技プログラミングって？
3. シミュレーション問題
4. 全探索問題
5. 本日のまとめ

勉強会の流れ

1. 勉強会の日程
2. 1日の流れ

勉強会の日程

AtCoder

1日目　シミュレーションと全探索

2日目　色々な全探索

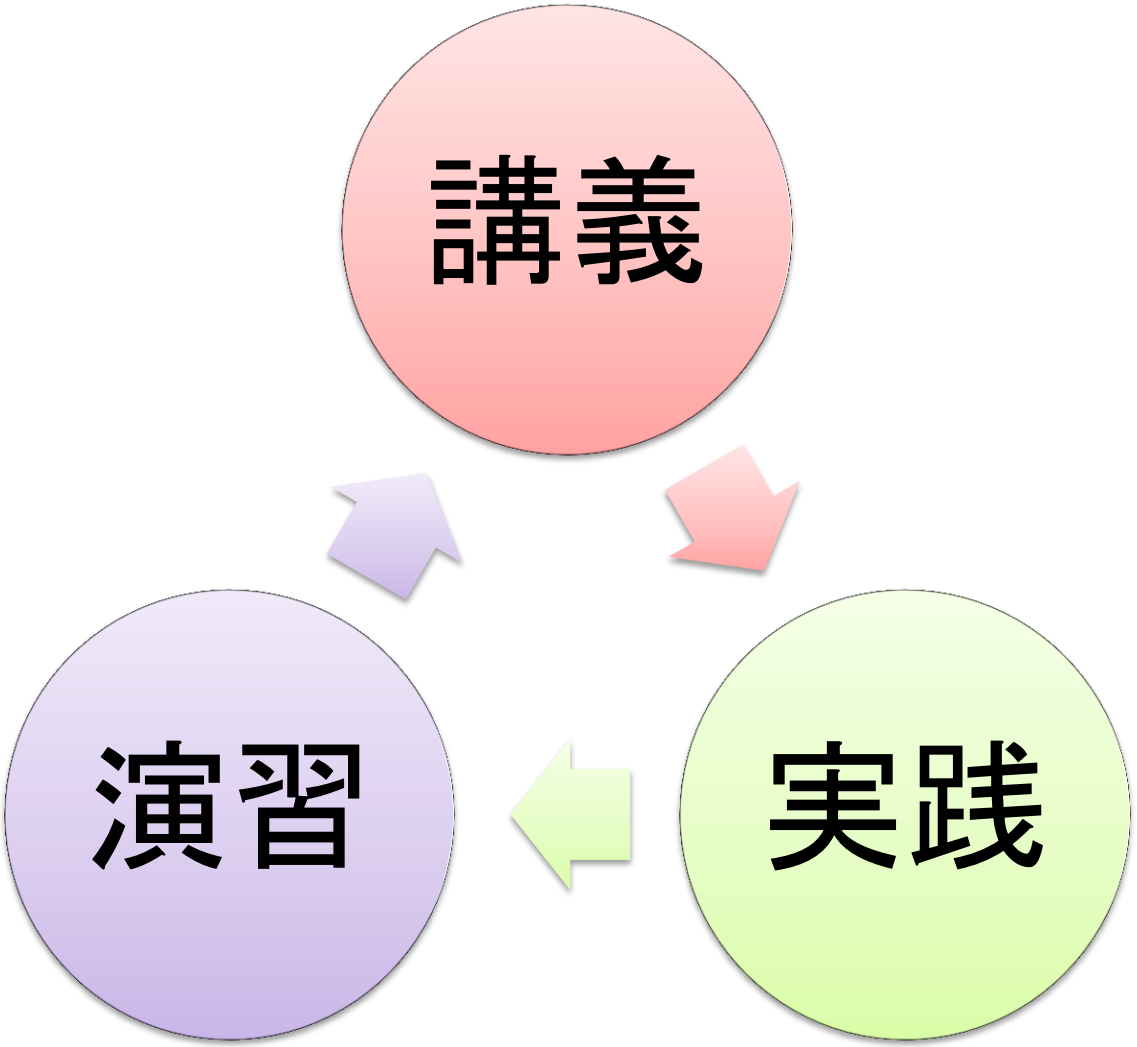
3日目　動的計画法とメモ化再帰

4日目　動的計画法と計算量を減らす工夫

5日目　難問に挑戦！

１日の流れ

AtCoder



１日の流れ

AtCoder

講義

- 基礎的なアルゴリズムを学ぶ
- 必要な知識を補う

実践

- 実際に問題例を見る
- コードでの表現方法を覚える

演習

- 自分で問題を解いてみる！
- コードを書いて理解を深める

１日の流れ

AtCoder

- 演習について
  - 実力差があると思うので、暇な時間が出来る人、ついていけない人、出ると思います。
    - どうしようもないので、早い人は支援に回ってもらえると嬉しいです！
  - 解らないことがあったら、#WAP_AtCoderでTwitterに投稿！
    - 多分早く終わった人が質問回答してくれます。
    - 具体例はこんな感じ
      - 「今やってる問題どれですか！」
      - 「この解答のWAが取れません！　http:// ~~」
      - 「コンパイルエラー出るよーなんでー＞＜　http:// ~~」
      - とりあえずコードが書けてたら、間違ってても提出してURLを貼りつけよう
  - もちろん、手を上げて質問してくれてもOK!
    - 回りきれる範囲では聞きに行きます。

今日の流れ

1. 前回までの復習
2. 今回やること

前回までの復習

AtCoder

- シミュレーション問題
  - 単純なシミュレーション問題
  - 繰り返しのあるシミュレーション問題
    - どちらも、言われたことを素直に実装！
- 全探索問題
  - 全探索問題のタイプ分け
    - Forループだけで書ける全探索・書けない全探索
    - Forループで書ける全探索をマスターする！

今回やること

AtCoder

- 全探索問題
  - 単純なforループで書けない全探索問題をイメージできるようにしよう！
    - どんな風に列挙すれば良いのか
    - 探索木とは何か
  - さらに、実際に実装できるようにしよう！
    - 幅優先探索
    - 深さ優先探索
  - その他便利な探索方法を覚えよう！
    - Bitを利用した二分木に対する全列挙
    - 順列に対する全列挙　←時間があれば。

今日の講義

1. どうやって探索するの？
2. 深さ優先探索
3. 幅優先探索
4. 深さ優先探索と幅優先探索の違い
5. 色々な探索

どうやって探索するの？

1. 探索木が必要になるケース
2. 探索木を探索する方法

探索木が必要になるケース

AtCoder

- 前回までの復習
  - 色々な全探索の種類
  - かんたん
    - 1から100000までの数字の中に、素数がいくつあるか答えなさい。
    - アルファベット3文字で構成された文字列のうち、画数が5以下で書けるものが何通りあるか求めなさい。
    - 10個のりんごを、3人で分けます。分け方が何通りあるかを答えなさい。
  - むずかしい
    - 将棋の駒の香車があります。1マス目から9マス目まで移動する時、2つ飛ばしで移動する動きがないパターンの数を答えなさい。
    - 3*3のパネルに、9種類の数字を1つずつ置くことが出来ます。魔法陣が作れるかどうかを判定しなさい。
    - 10問の〇×クイズを連続して解きます。6問以上正解して、なおかつ3問連続正解を含む、解答の仕方が何通りあるか答えなさい。

探索木が必要になるケース

AtCoder

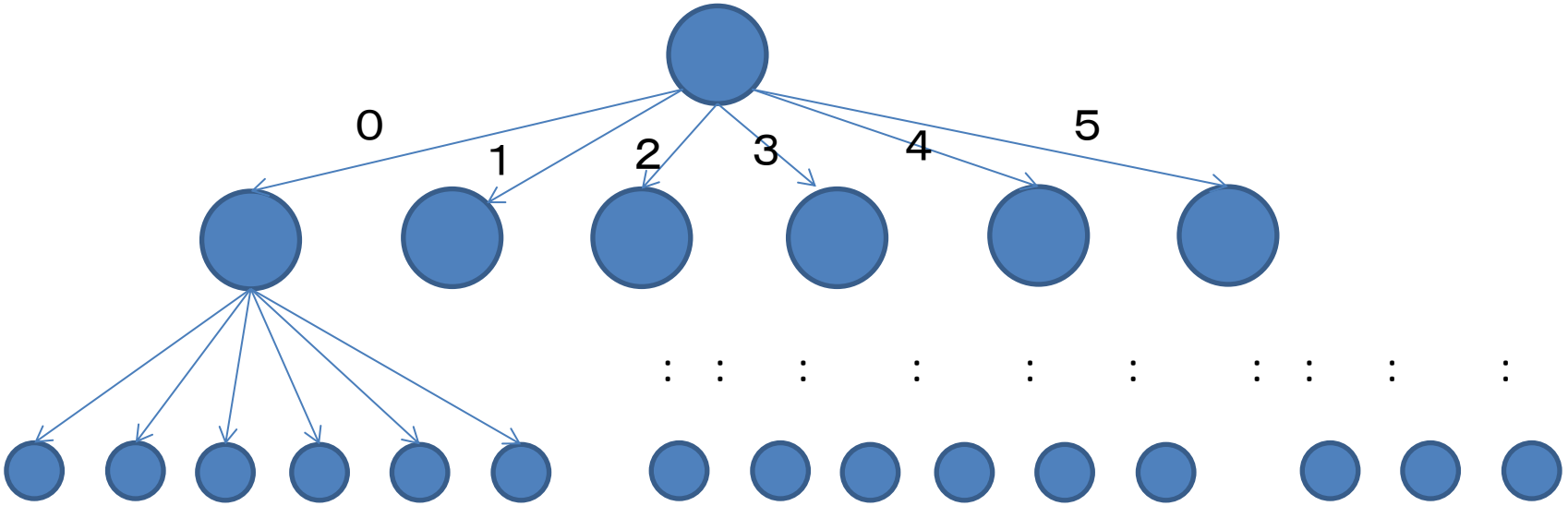
- 何が「かんたん」と「むずかしい」の差？
  - 少ないforループで書けるか否か！

- 例えば、この2つは大きく違う！
  - N個のりんごを、3人で分けます。
    - 3個のforループで表現できる
  - 10個のりんごを、N人で分けます。
    - N個のforループで表現できる？
      - Nが入力で与えられたらどうしようもない。

探索木が必要になるケース

AtCoder

- N個のりんごを、3人で分けます。
  - 仮にN=5とする

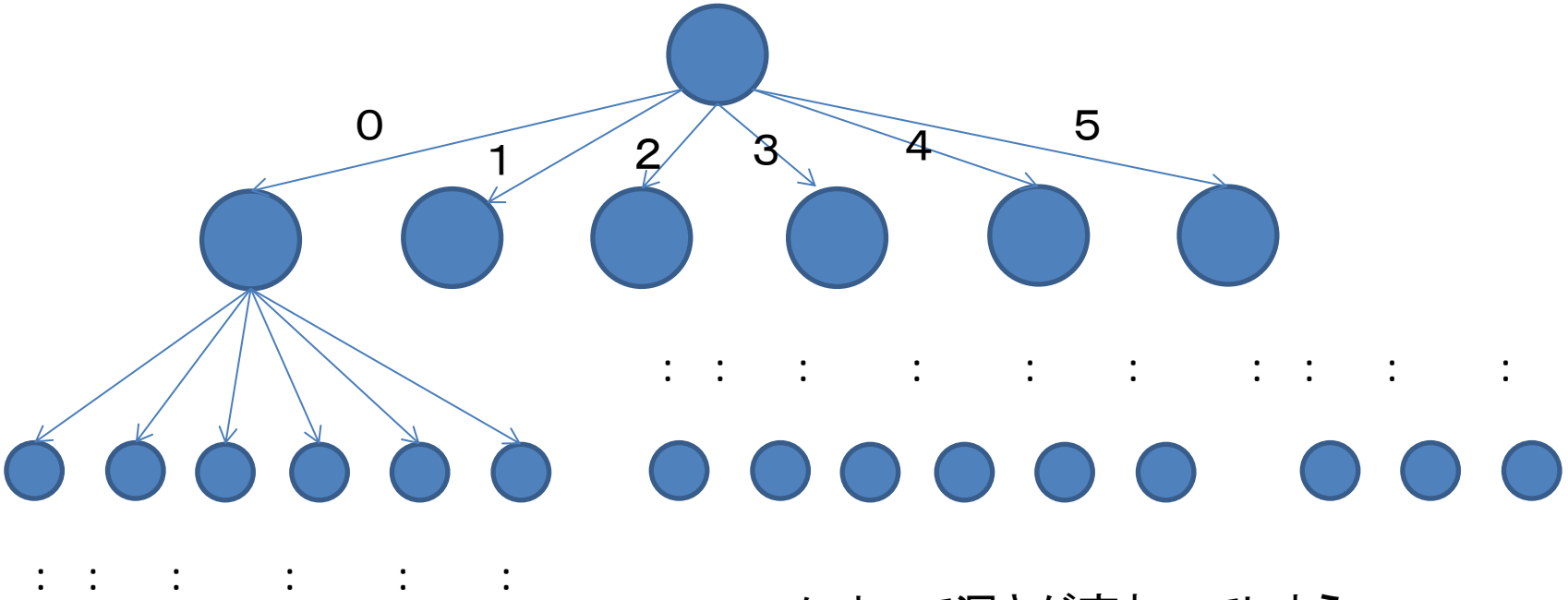


Nがいくつになろうと、深さは2まで。
だからforループ2回で探索できる！

探索木が必要になるケース

AtCoder

- 5個のりんごを、N人で分けます。
  - 仮にN=4とする



Nによって深さが変わってしまう。
Forループをたくさん書かないといけない。

探索木が必要になるケース

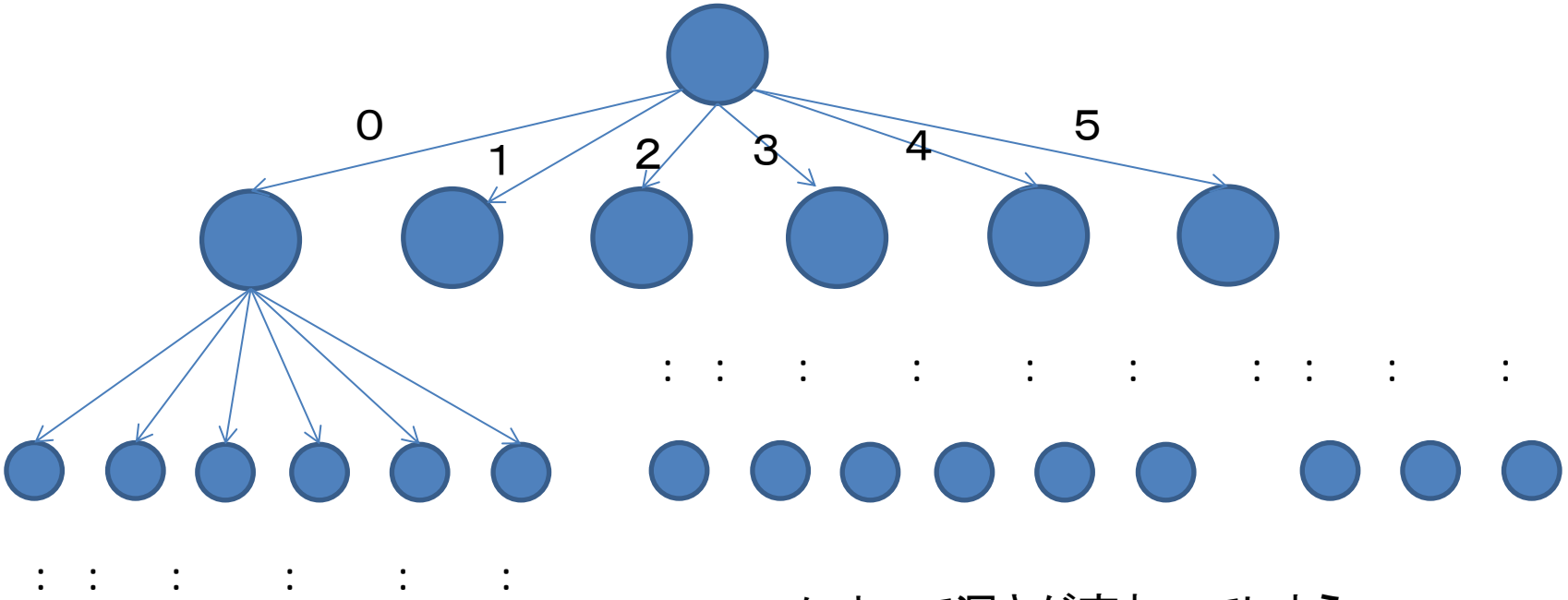
AtCoder

- 難しい全探索の解き方
  - 単純なforループでは、最早上手く解くことは出来ない。
  - forループよりももっと汎用性の高い、他の実装方法をしなくてはならない！

探索木が必要になるケース

AtCoder

- 要するに、こんな感じの探索木を書いた時に、この〇を全て辿れるようなアルゴリズムを考えれば良い。



Nによって深さが変わってしまう。
Forループをたくさん書かないといけない。

探索木が必要になるケース

AtCoder

- 辿る方法として有名なものは、この2つ！
  - 深さ優先探索
  - 幅優先探索

- 今回は、この2つを徹底的にマスターします！

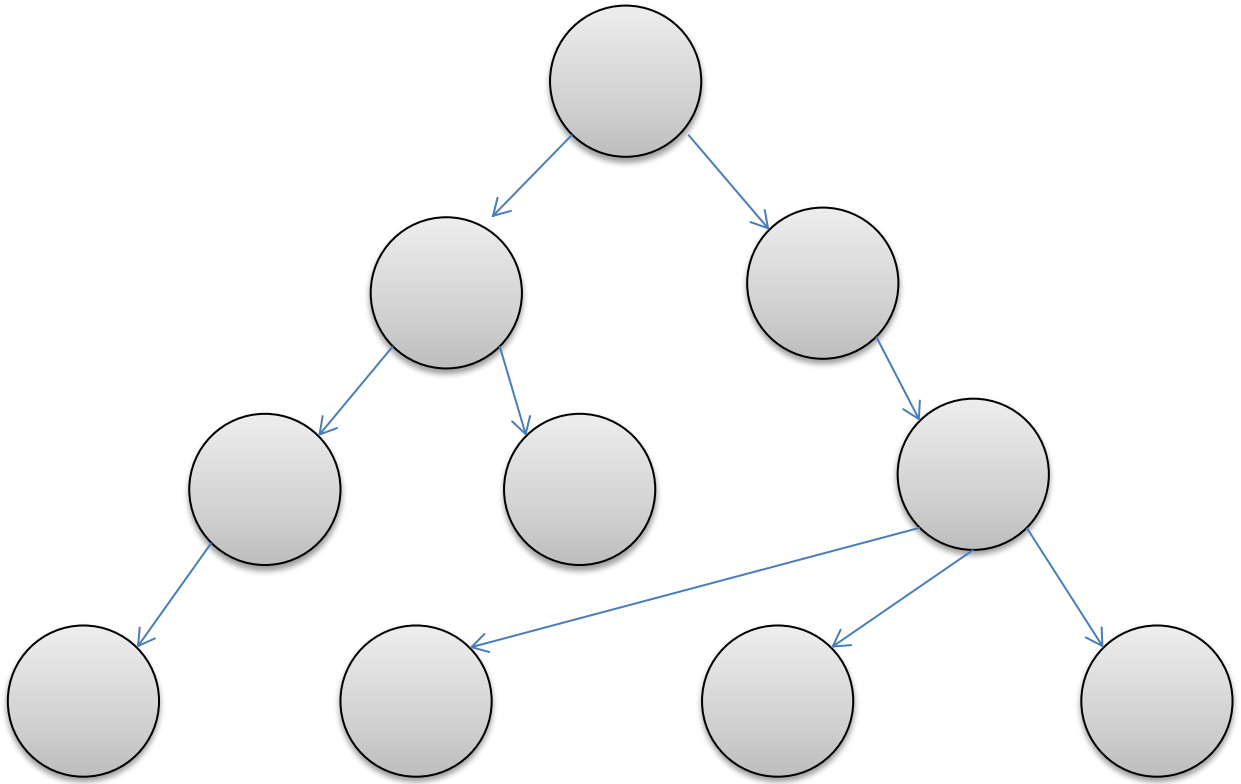
深さ優先探索

1. 深さ優先探索って？
2. 深さ優先探索の実装
3. 深さ優先探索が有効な場面

深さ優先探索って？

AtCoder

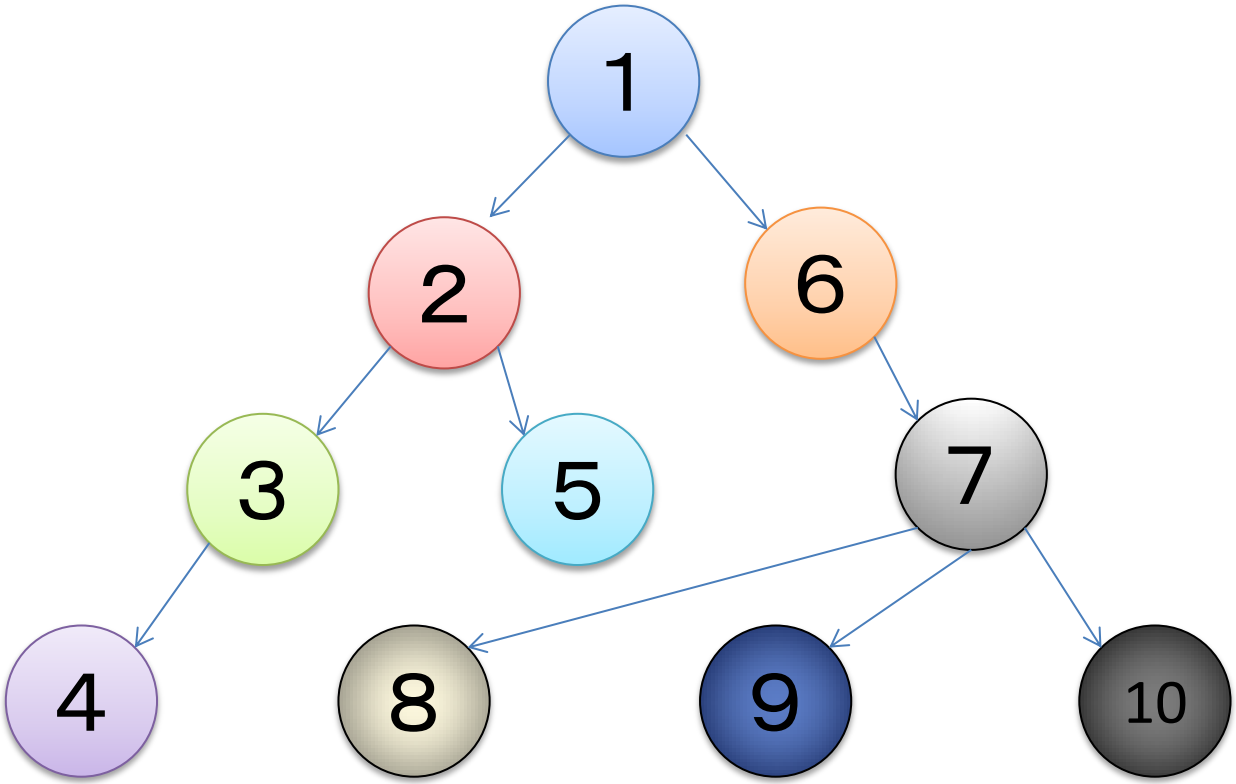
- 例えばこんな探索木があったとしましょう。



深さ優先探索って？

AtCoder

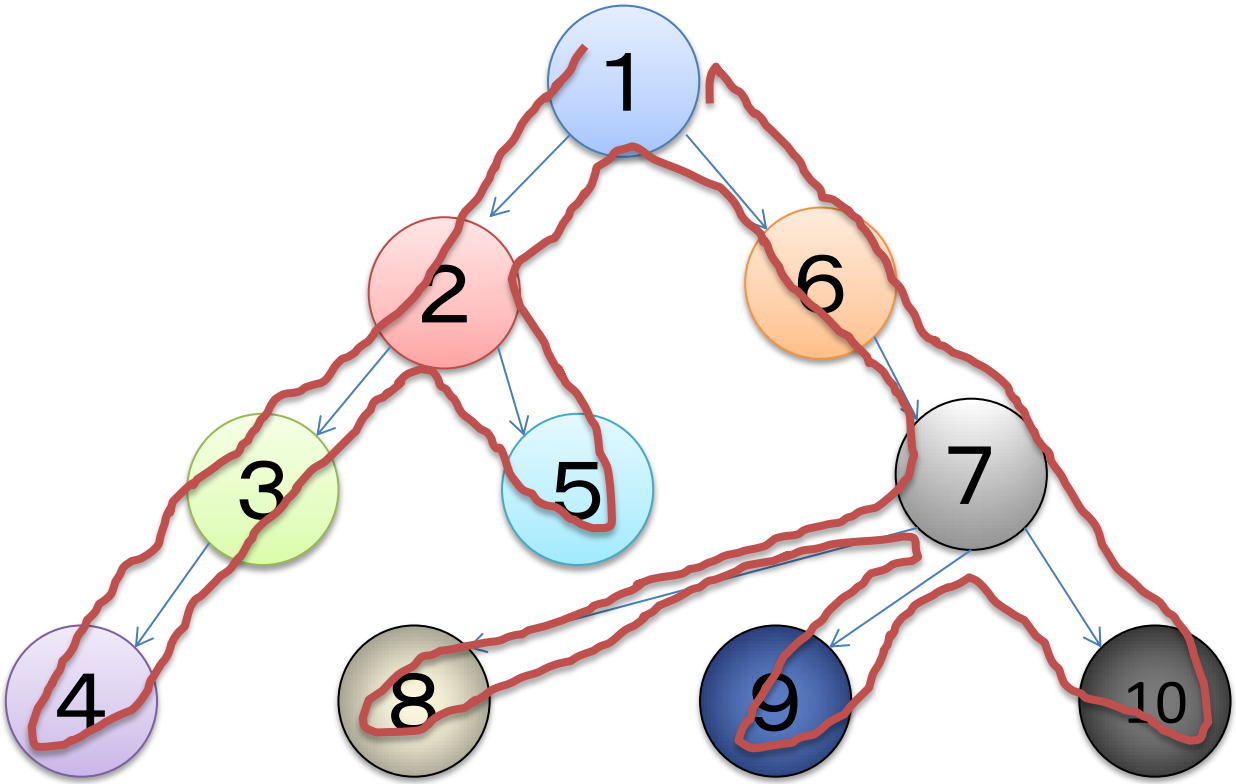
- 深さ優先探索は、こんな探索方法



深さ優先探索って？

AtCoder

- 深さ優先探索は、こんな探索方法
  - 一筆書きで探索します！



深さ優先探索って？

AtCoder

- 再帰関数を使った探索手法
  - 本当は使わないでも書けるけど、使うのが一般的
- 慣れるまでは大変だが、慣れれば極めて直感的に書ける。
- 実装イメージは以下のような感じ

```
int dfs(今の状態){
    int ret = 0;
    for( … ) ret += dfs(次の状態);
    return ret;
}
```

深さ優先探索の実装

AtCoder

- 問題
  - M個のリンゴを、N人で分ける時、分け方が何通りあるか？
- ここで作るべき関数を、こんな感じで書いてあげる。
  - Int dfs(int M, int N)
    - この関数を使えば、M個のりんごをN人で分けた時の、組み合わせの個数が解る、という前提を置く。

深さ優先探索の実装

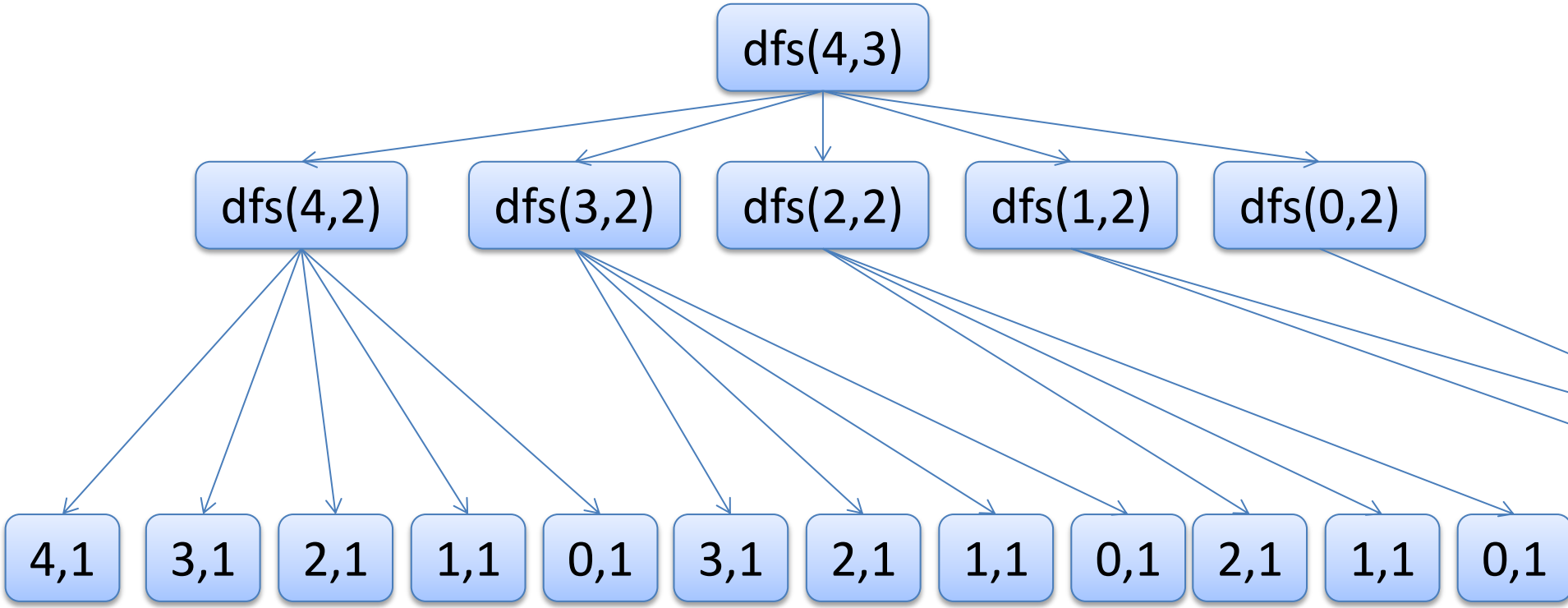
AtCoder

- Int dfs(int M, int N)の中身について
  - N > 1のとき
    - まだりんごを配るべき人が二人以上残っている。
    - 一人目の人に対して、「りんごを0個あげる」「りんごを1個あげる」「りんごを2個あげる」・・・「りんごをM個あげる」の全通りを試し、その結果の和を取りたい。
    - dfs(M, N) = dfs(0, N-1) + dfs(1, N-1) + … + dfs(M, N-1)と表せる
  - N = 1のとき
    - りんごを配るべき人が一人しか残っていない。
    - 残りのりんごを全てあげる以外の選択肢がない。
      - つまり、パターンは1通り

深さ優先探索の実装

AtCoder

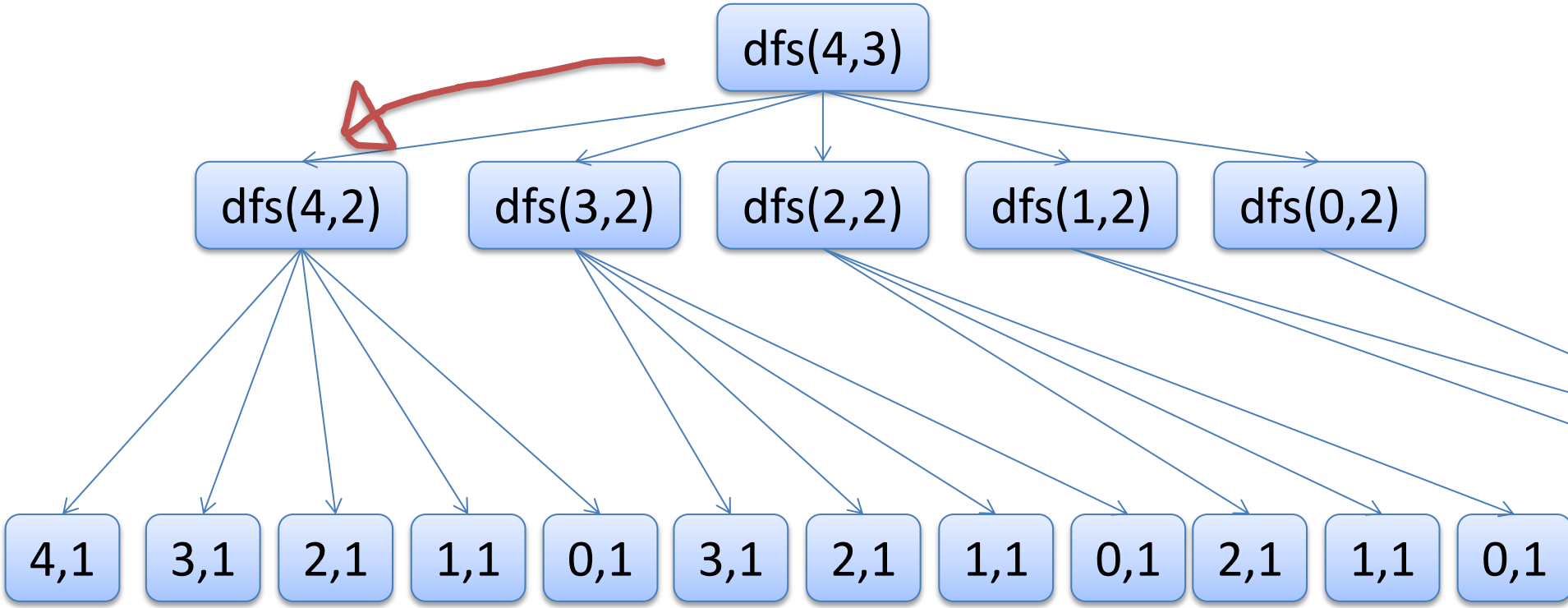
- 例：4つのりんごを3人で分ける。



深さ優先探索の実装

AtCoder

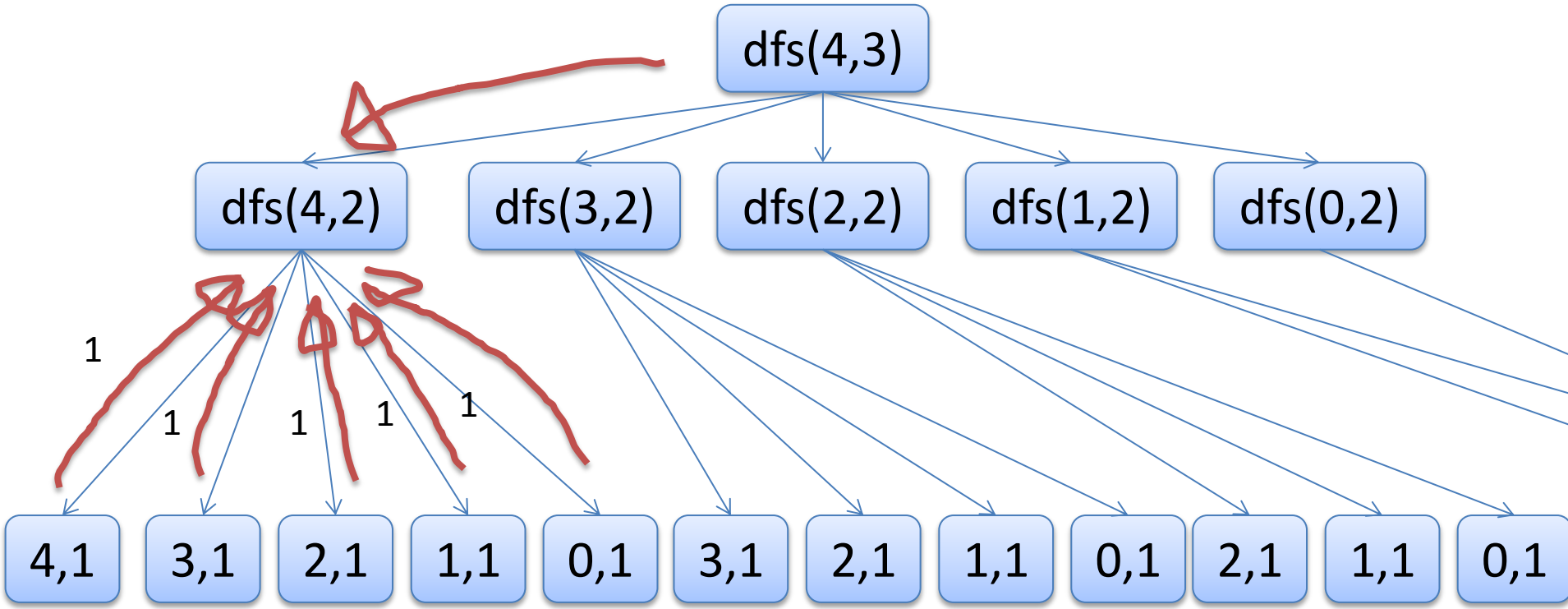
- 例：4つのりんごを3人で分ける。



深さ優先探索の実装

AtCoder

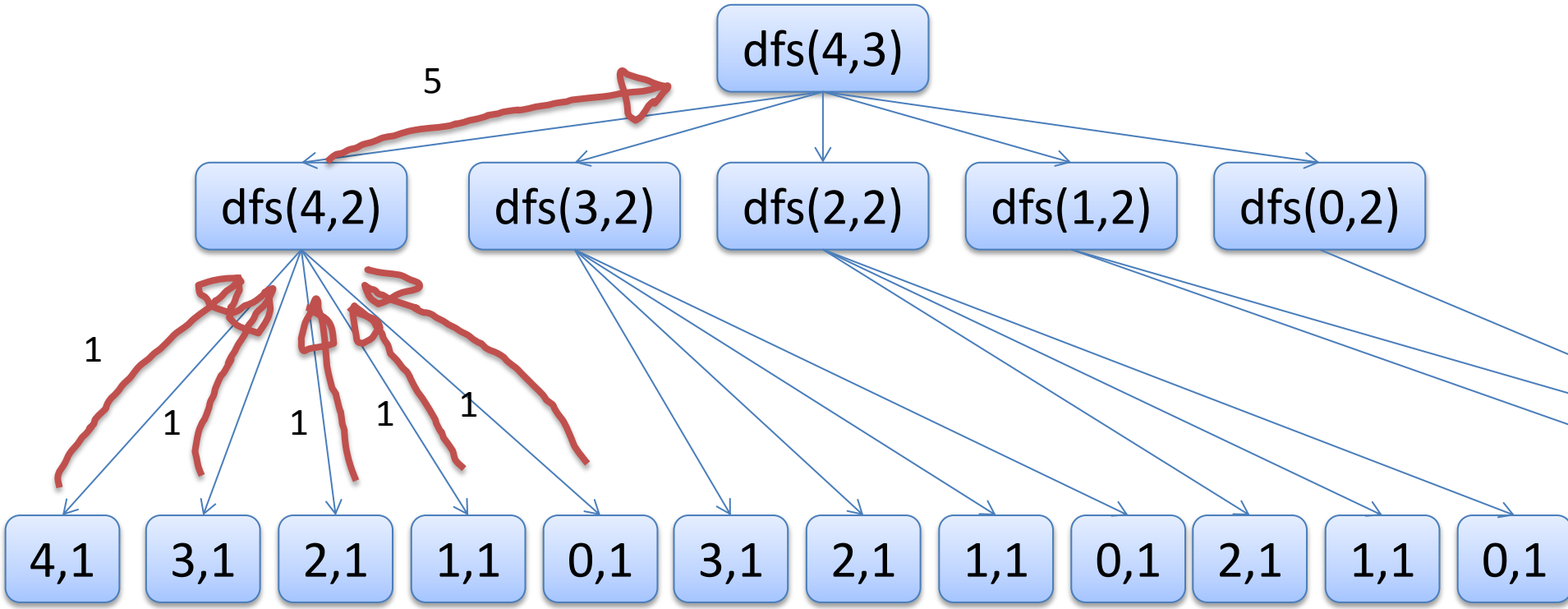
- 例：4つのりんごを3人で分ける。



深さ優先探索の実装

AtCoder

- 例：4つのりんごを3人で分ける。



深さ優先探索の実装

AtCoder

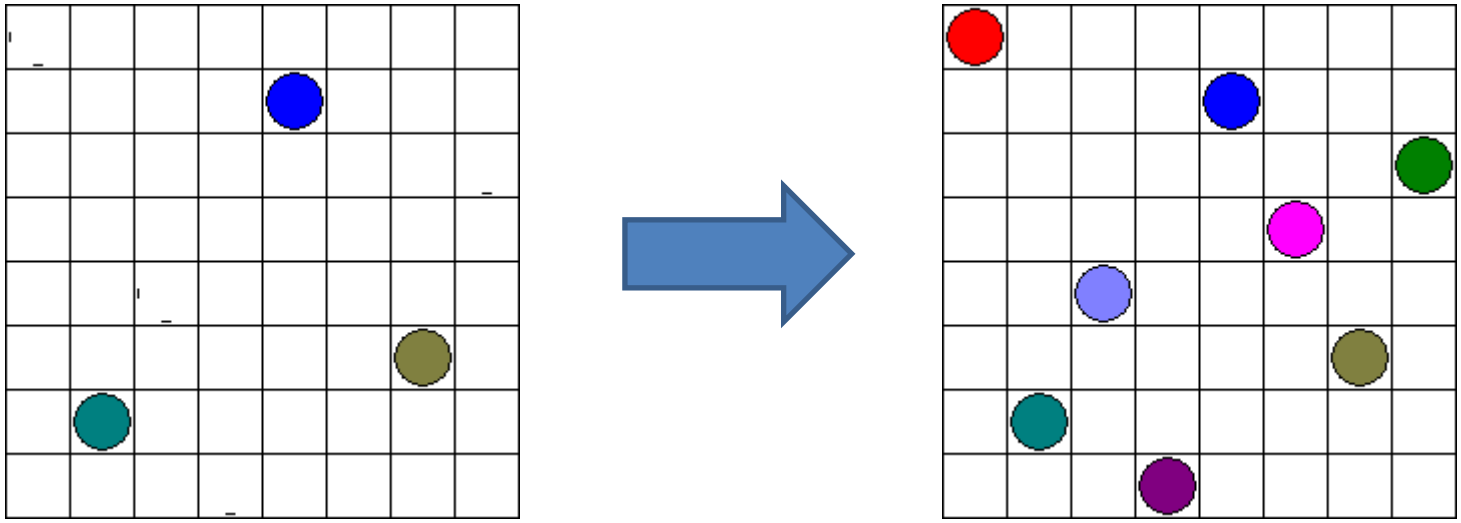
- ソースコード

```
int dfs(int m, int n){
    if(n==1) return 1;
    int ret = 0;
    for(int i=0;i<=m;i++){
        ret += dfs(m-i, n-1);
    }
    return ret;
}
```

深さ優先探索の実装　実践

AtCoder

- ARC001 C問題　パズルのお手伝い
  - http://arc001.contest.atcoder.jp/tasks/arc001_3
  - 問題概要
    - 8*8のマス目が与えられます。8個の駒を、「たて」「よこ」「ななめ」で同じ列に存在しないように配置したいです。最初に3個配置されているので、残りのマスを埋めなさい。無理ならNo Answerと出力



深さ優先探索の実装　実践

AtCoder

- 解き方
  - まず全てのマスに番号をつける

0	1	2	3	4	5	6	7
8							
					61	62	63

深さ優先探索の実装　実践

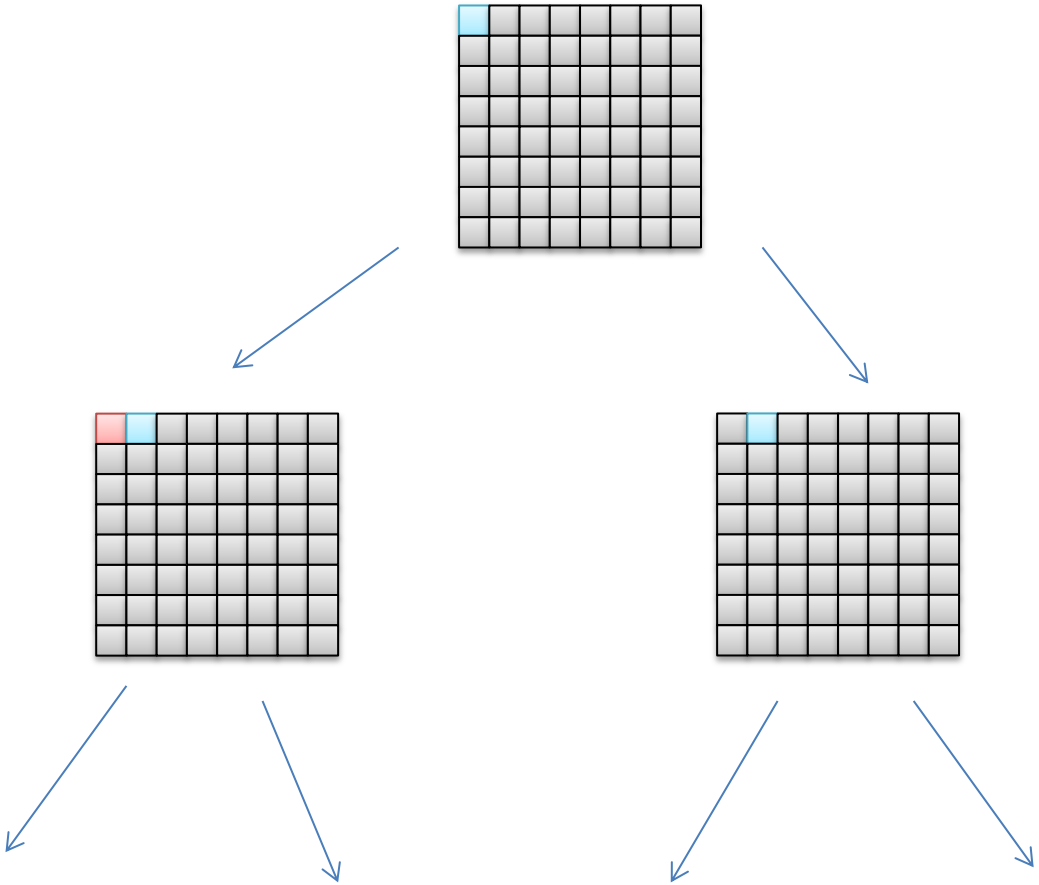
AtCoder

- 解き方
  - まず全てのマスに番号をつける
  - １番から順番に、「置く」「置かない」を全て試す。
    - 既に他の駒と干渉してしまって置けない場合は諦める。
  - 具体的には、以下のような関数を作る
    - bool dfs(今見ているマス, 残りマス数, 今の盤面)
      - 返り値は、「完成させられたか否か」
    - これを用いて、深さ優先探索を行う。
    - 「置く」「置かない」を試す。
      - すでに置けない場所なら「置く」は試さない。
      - 絶対に置かないといけないなら「置かない」は試さない。

深さ優先探索の実装　実践

AtCoder

- 探索のイメージ



深さ優先探索の実装　実践

AtCoder

- ソースコード
  - http://arc001.contest.atcoder.jp/submissions/145104

```
10    void run()
11    {
12        Scanner cin = new Scanner(System.in);
13
14        //入力を受け取り、char型の配列に写す
15        char[][] board = new char[8][8];
16        for(int i=0;i<8;i++){
17            String st = cin.next();
18            for(int j=0;j<8;j++){
19                board[i][j] = st.charAt(j);
20            }
21        }
22
```

深さ優先探索の実装　実践

AtCoder

- ソースコード

```
22
23          //もし条件を満たす解があれば出力
24          if(dfs(0, 8, board)){
25              for(int i=0;i<8;i++){
26                  for(int j=0;j<8;j++){
27                      System.out.print(board[i][j]);
28                  }
29                  System.out.println();
30              }
31          }
32          //なければNo Answerを出力
33          else System.out.println("No Answer");
34      }
```

深さ優先探索の実装　実践

AtCoder

- ソースコード

```
36
37    boolean dfs(int pos, int nokori, char[][] board)
38    {
39        //もし8個の駒が置けたなら、trueを返す。
40        if(nokori==0) return true;
41
42        //もしもう置く場所がないなら、falseを返す。
43        if(pos==64) return false;
44
45        //整数posを座標に変換
46        int y = pos / 8;
47        int x = pos % 8;
48
```

深さ優先探索の実装　実践

AtCoder

- ソースコード

```
49          //もし絶対に置かないといけない場所なら
50          if(board[y][x] == 'Q'){
51              //駒を置いても大丈夫なのであれば、置いて探索を続ける
52              if(isPuttable(y, x, board))
53                  if(dfs(pos+1, nokori-1, board)) return true;
54          }
55          //違えば、両方試す。
56          else{
57              //置ける場所なのであれば、置いてから探索を続ける。
58              if(isPuttable(y, x, board)){
59                  board[y][x] = 'Q';
60                  if(dfs(pos+1, nokori-1, board)) return true;
61                  board[y][x] = '.';
62              }
63
64              if(dfs(pos+1, nokori, board)) return true;
65          }
66          return false;
67      }
```

深さ優先探索の実装　実践

AtCoder

- ソースコード

```
69    boolean ok(int y, int x){
70        return y >= 0 && x >= 0 && y < 8 && x < 8;
71    }
72
73    boolean isPuttable(int y, int x, char[][] board){
74        //8方向全部調べる
75        for(int vy=-1; vy<=1; vy++){
76            for(int vx=-1; vx<=1; vx++){
77                if(vy==0 && vx==0) continue;
78                //はみ出るまで移動して、駒を見つけたらfalse
79                int ty=y, tx=x;
80                while(true){
81                    ty += vy; tx += vx;
82                    if(!ok(ty,tx)) break;
83                    if(board[ty][tx] == 'O') return false;
84                }
85            }
86        }
87        //何も重複がなければtrueを返す
88        return true;
89    }
90 }
```

深さ優先探索の実装　実践

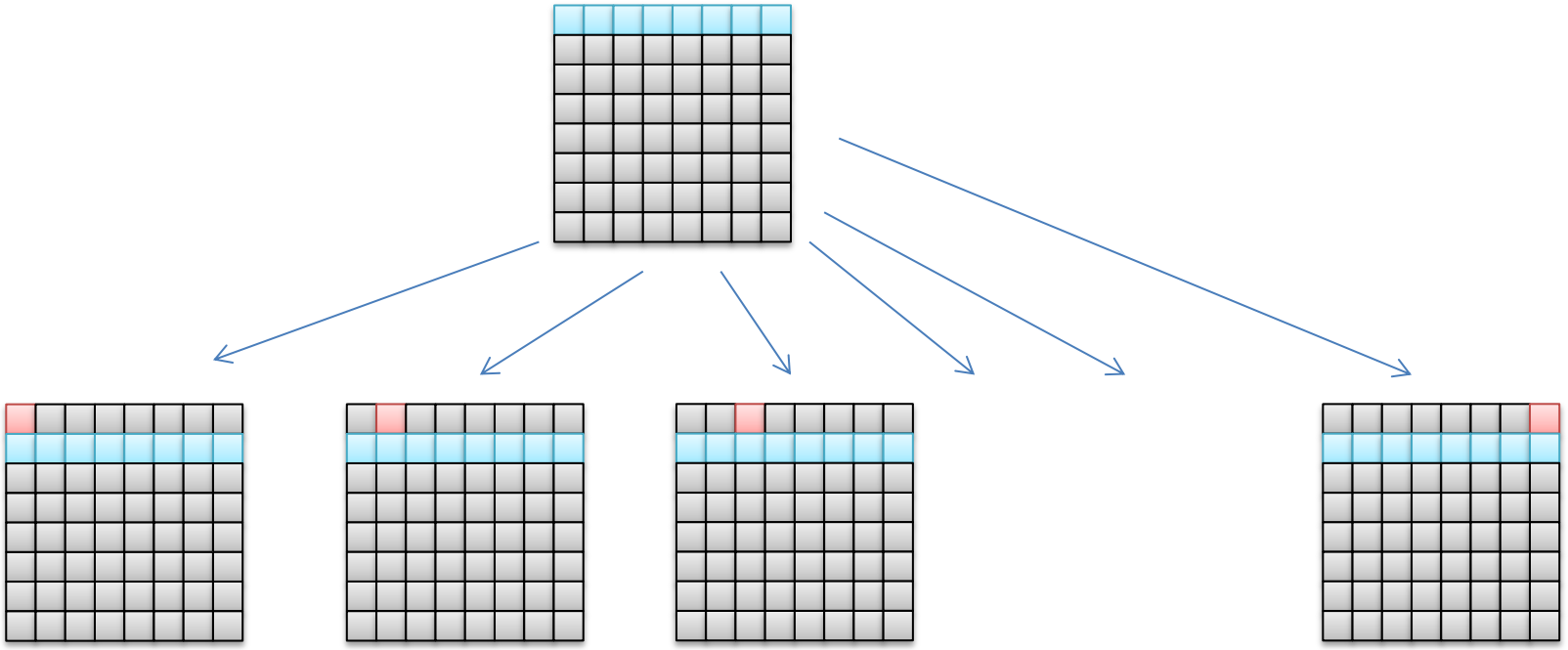
AtCoder

- 解き方2
  - 1行に1個しか置かないということは、各行に対して、どの列に置くかを割り当てれば良い。
    - 長さ8の順列を全列挙するのも良い
  - 割り当てた駒が、正しく置けているかを確かめていく。
  - 用意する再帰関数は、以下のような感じ。
    - dfs(今見ている場所, 盤面の状況)

深さ優先探索の実装　実践

AtCoder

- 探索のイメージ



深さ優先探索の実装　実践

AtCoder

- ソースコード
  - http://arc001.contest.atcoder.jp/submissions/145107

```
37    boolean dfs(int y, char[][] board)
38    {
39        //もし8個の駒が置けたなら、trueを返す。
40        if(y==8) return true;
41
42        //すでに置かれている場所がないか探す
43        int target = -1;
44        for(int x=0;x<8;x++){
45            if(board[y][x] == 'Q'){
46                //2か所以上に置かれているならfalseを返す。
47                if(target != -1) return false;
48                target = x;
49            }
50        }
```

深さ優先探索の実装　実践

AtCoder

- ソースコード

```
52          //もし絶対に置かないといけない場所があれば
53          if(target != -1){
54              //駒を置いても大丈夫なのであれば、置いて探索を続ける
55              if(isPuttable(y, target, board))
56                  if(dfs(y+1, board)) return true;
57          }
58          //なければ、全通り試す。
59          else{
60              for(int x=0;x<8;x++){
61              //置ける場所なのであれば、置いてから探索を続ける。
62                  if(isPuttable(y, x, board)){
63                      board[y][x] = 'Q';
64                      if(dfs(y+1, board)) return true;
65                      board[y][x] = '.';
66                  }
67              }
68          }
69          return false;
70      }
```

深さ優先探索の実装　実践

AtCoder

- おまけ
  - 盤面の状態を引数に入れたが、これは外に出しても良い
    - 覚えておくとたまに便利

```
boolean dfs(int y, char[][] board)
{
```

```
char[][] board;
boolean dfs(int y)
{
```

深さ優先探索の実装　実践

AtCoder

- 解き方3
  - ボードサイズが8パターンなんだから、8個forループを置けば良い！！！

    - やめましょう。
    - でも意外と実装短めだったりします。

深さ優先探索の実装　演習

AtCoder

- ARC009 C問題　高橋君、24歳
  - http://arc009.contest.atcoder.jp/tasks/arc009_3
  - 今回解くのは、これの部分点（small部分）
  - Largeが採点されないように、以下のような記述を
    - If(N>8) return;

- 問題概要
  - 高橋君はN枚の手紙を出した。
  - でもM枚は違う人に出してしまったらしい。
  - 出してしまった手紙の組み合わせとして、あり得るものの組み合わせの個数を答えなさい。

深さ優先探索の実装　演習

AtCoder

- 暇な人は以下の問題にチャレンジ！
- ARC014 C問題　魂の還る場所
  - http://arc014.contest.atcoder.jp/tasks/arc014_3
  - 部分点のみ
- オリジナル問題
  - 3*3の魔法陣がいくつ存在するか数え上げなさい。
  - ただし、反転・90度回転で同じとなる魔法陣は、同じものと見做します。
  - それが出来たら4*4を頑張ってみてください。

深さ優先探索の実装　演習

AtCoder

- 解き方
  - 誰宛ての手紙が誰に届いたかを全列挙する
  - それぞれに対し、いくつ間違っているかを列挙し、Kと等しくなった数を返す。
  - 作る関数はこんな感じ。
    - int dfs(int pos, boolean[] used, int nokori)
      - pos あと何人残っているか
      - used どの友人の手紙をすでに使ったか
      - nokori 間違っている人数は残り何人か。
        - » nokoriはなくても、具体的な順列を持って置くことで、最後に全て計算することで実装することは可能
        - » 途中で計算すると、具体的な順列を持っていなくても良くなる。

深さ優先探索の実装　演習

AtCoder

- ソースコード
  - http://arc009.contest.atcoder.jp/submissions/145109

```
    int N, K;
    void run()
    {
        Scanner cin = new Scanner(System.in);
        N = cin.nextInt();
        if(N>8) return;
        K = cin.nextInt();

        //どの人の手紙をすでに配ったかのフラグ
        boolean[] used = new boolean[N];
        //探索した結果を出力
        System.out.println(dfs(0, used, K));
    }
```

深さ優先探索の実装　演習

AtCoder

- ソースコード

```
23    int dfs(int pos, boolean[] used, int nokori){
24        //全員見終えたら、残った人数が一致してるかを調べる
25        if(pos==N){
26            if(nokori==0) return 1;
27            else return 0;
28        }
```

深さ優先探索の実装　演習

AtCoder

- ソースコード

```
30      int ret = 0;
31      //pos番目の人が、誰の手紙を貰ったか、全通り試す
32      for(int i=0;i<N;i++){
33          //既に配った手紙は使えない
34          if(used[i]) continue;
35          //先を調べる前に、フラグを立てておく
36          used[i] = true;
37          //残りの間違える個数を更新
38          int nextnokori = nokori;
39          if(i!=pos) nextnokori--;
40          //先を探索して答えを足す
41          ret += dfs(pos+1, used, nextnokori);
42          //調べ終わったらもとに戻す
43          used[i] = false;
44      }
45
46      return ret;
47  }
```

深さ優先探索が有効な場面

AtCoder

- とにかく全列挙がしたい時に使おう！
  - 「有限のもの」に対する全列挙ツールとしては最強です。
  - これだけ知っとけば問題ないです。

- 「無限のもの」に対してはちょっと弱い。
  - 無限にループしちゃいます。
  - そもそも全列挙が無理なので、余り気にしないで良い。
  - しかし、「全探索で解ける」と巷で言われている問題には、「全てを列挙出来るわけではない」問題も挙げられる。
    - この辺りは、幅優先探索での話で

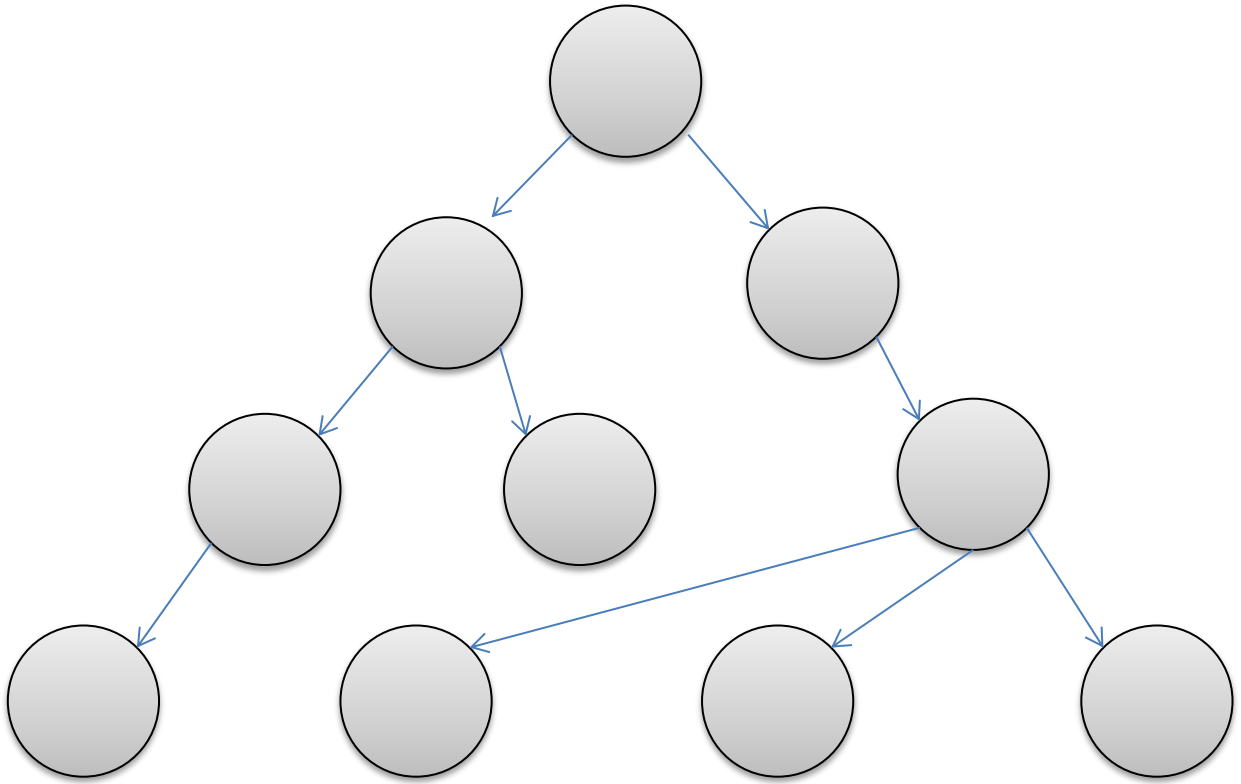
幅優先探索

1. 幅優先探索って？
2. 幅優先探索の実装
3. 幅優先探索が有効な場面

幅優先探索って？

AtCoder

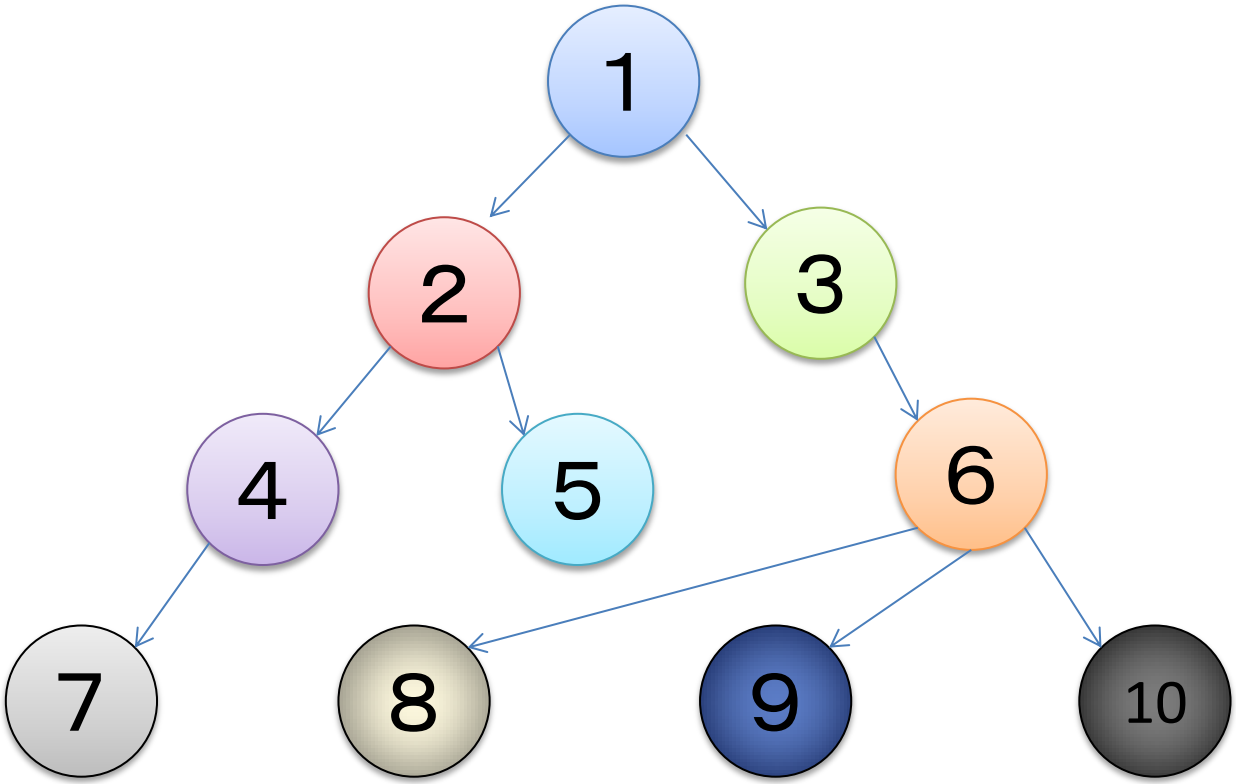
- 例えばこんな探索木があったとしましょう。



幅優先探索って？

AtCoder

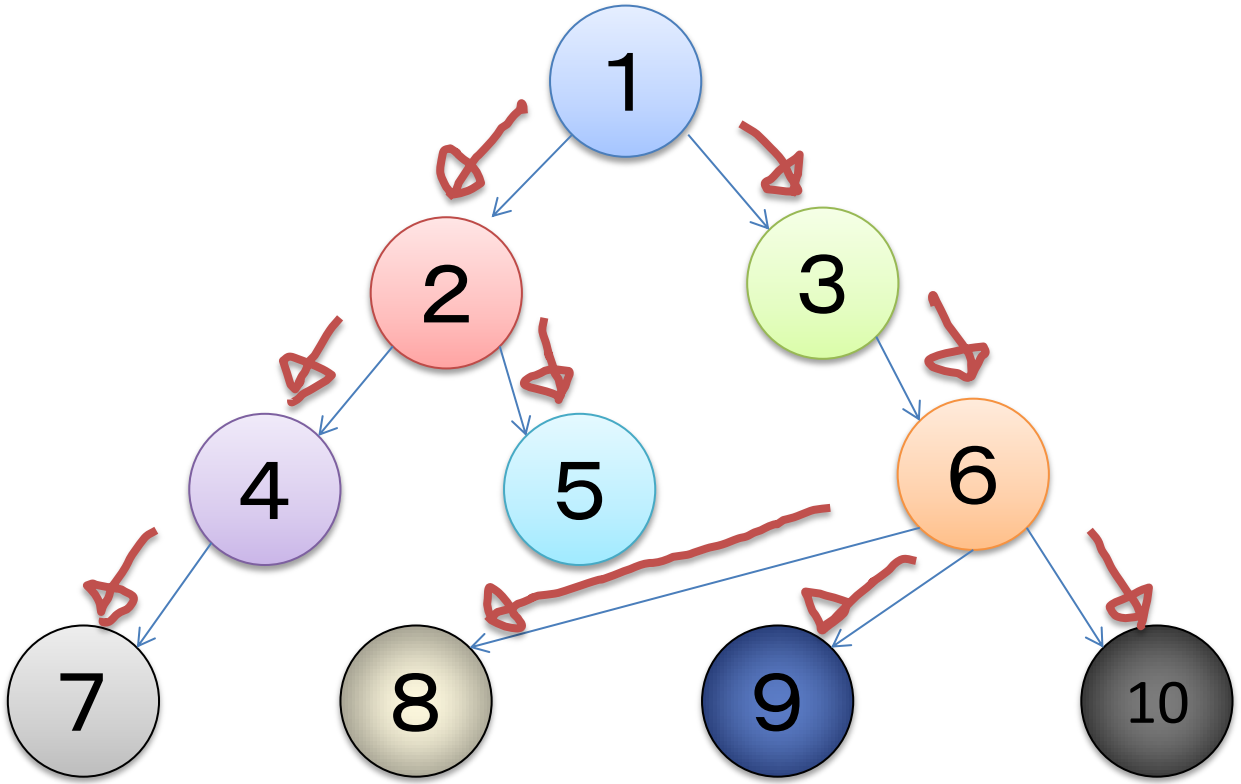
- 幅優先探索は、こんな探索方法



幅優先探索って？

AtCoder

- 幅優先探索は、こんな探索方法
  - 近い順に探索する！



幅優先探索って？

AtCoder

- キューとループを使って、近い順から
  - キューは、変数をたくさん放り込むと、放り込んだ順から取り出せるもの、と解釈してもらえばOKです。
  - 実装のイメージは下のような感じ

```
Queue<int> q = new Queue<int>();
q.push(初期状態);
while(!q.isEmpty()){
    int now = q.poll(); //今の状態を取り出す
    for( … ) q.add(次の状態);
}
```

幅優先探索の実装　実践

AtCoder

- ARC005 C問題　器物損壊！高橋君
  - http://arc005.contest.atcoder.jp/tasks/arc005_3

- 問題概要
  - 2次元空間の迷路のようなものが与えられる
  - 壁を2回まで壊すことが出来る
  - スタートからゴールに辿り着くことが出来るか出力せよ。

幅優先探索の実装　実践

AtCoder

- やり方
  - 幅優先探索で、0回の破壊で行ける範囲、及び1回目に破壊する壁を全て列挙する。
  - 幅優先探索で、1回目の破壊で行ける範囲、及び2回目に破壊する壁を全て列挙する。
  - 幅優先探索で、2回目の破壊で行ける範囲を列挙する。
  - 行ける範囲にGがあればYES

幅優先探索の実装　実践

AtCoder

- 幅優先部分のやり方
  - Queue<Integer> nowq; <- 今調べるもの
  - Queue<Integer> nextq; <- 次ターンに調べるもの

```
for(int i = 0; i < 3; i++){
    nextq = new Queue<Integer>();
    while(nowq.isEmpty()){
        int now = nowq.poll();
        for(…) ここで次の状態を列挙する
    }
}
```

幅優先探索の実装　実践

AtCoder

- ソースコード
  - http://arc005.contest.atcoder.jp/submissions/145134

```
10  int H, W;
11  void run()
12  {
13      Scanner cin = new Scanner(System.in);
14      H = cin.nextInt();
15      W = cin.nextInt();
16      String[] board = new String[H];
17      for(int i=0;i<H;i++){
18          board[i] = cin.next();
19      }
20
21      //sの場所を探す
22      int fx = -1;
23      int fy = -1;
24      for(int i=0;i<H;i++){
25          for(int j=0;j<W;j++){
26              if(board[i].charAt(j) == 's'){
27                  fy = i;
28                  fx = j;
29              }
30          }
31      }
```

幅優先探索の実装　実践

AtCoder

- ソースコード

```
33          //初期状態の挿入
34          Queue<Integer> nowq = new LinkedList<Integer>();
35          nowq.add(encode(fy, fx));
36
37          //移動先の列挙
38          int[] vy = new int[]{1,0,-1,0};
39          int[] vx = new int[]{0,1,0,-1};
40
41          //到達したかどうかのチェック用
42          boolean[][] check = new boolean[H][W];
43          check[fy][fx] = true;
44
```

幅優先探索の実装　実践

AtCoder

- ソースコード

```
//幅優先探索を三回繰り返す
for(int i = 0; i < 3; i++){

    //破壊した壁を入れるキューを用意する
    Queue<Integer> nextq = new LinkedList<Integer>();

    //幅優先探索
    while(!nowq.isEmpty()){
        //現在位置を取ってくる
        int now = nowq.poll();
        int y = now / 1000;
        int x = now % 1000;
```

幅優先探索の実装　実践

AtCoder

- ソースコード

```
58          //4方向に移動できるので、それぞれの移動先を調べる
59          for(int j=0;j<4;j++){
60              int ny = y + vy[j];
61              int nx = x + vx[j];
62
63              //範囲外だったらcontinue
64              if(!ok(ny,nx)) continue;
65              //既に調べてあるマスだったらcontinue
66              if(check[ny][nx]) continue;
67
68              //調査済みフラグを立てる
69              check[ny][nx] = true;
70
71              //ゴールなら終了
72              if(board[ny].charAt(nx) == 'g'){
73                  System.out.println("YES");
74                  return;
75              }
76              //壁なら壁用キューに入れる
77              else if(board[ny].charAt(nx) == '#'){
78                  nextq.add(encode(ny,nx));
79              }
80              //それ以外なら通常のキューに入れる
81              else{
82                  nowq.add(encode(ny,nx));
83              }
84          }
```

幅優先探索の実装　実践

AtCoder

- ソースコード

```
86              }
87              //次のキューに、壁を破壊したキューを移し替える
88              nowq = nextq;
89          }
90          //gに辿り着けなかったらNO
91          System.out.println("NO");
92          return;
93      }
94
95      //座標を1つの整数に変換する
96      int encode(int y, int x){
97          return y * 1000 + x;
98      }
99
00      //範囲内に収まっているかどうかのチェック
01      boolean ok(int y, int x){
02          return y >= 0 && x >= 0 && y < H && x < W;
03      }
```

幅優先探索の実装　演習

AtCoder

- ARC001 B問題 リモコン
  - http://arc001.contest.atcoder.jp/tasks/arc001_2

- 問題概要
  - エアコンの温度を、A度からB度に変更したい
  - エアコンのリモコンには、6つのボタンがついている。
    - 温度を1, 5, 10度上げる
    - 温度を1, 5, 10度下げる
  - 温度を

幅優先探索の実装　演習

AtCoder

- この問題の難しい点
  - 「全探索」で解ける問題だが、「パターンの全列挙」が出来るわけではない。
    - リモコンの押し方は無限通り存在する。
    - 深さ優先探索だと、ずっとループしてしまう！
    - 39 -> 40 -> 39 -> 40 -> 39 -> 40 -> 39 -> ……
    - ループしないように対策しても難しい。
  - だが、ボタンを押す回数を制限してしまえば、組み合わせは有限通りしか存在しない。
  - よって、押す回数が少ないパターンから順番に調べていく

幅優先探索の実装　演習

AtCoder

- 暇な人は次の問題に挑戦！
- ARC015 C問題　変わった単位
  - http://arc015.contest.atcoder.jp/tasks/arc015_3

幅優先探索の実装　演習

AtCoder

- やりかた
  - 最初に、キューに、初期状態Aを入れておく。
  - キューに入っているものを順番に取り出す。
    - そこから遷移出来る温度を全て取り出す。
    - もし初めて辿り着いた温度であれば、遷移回数をメモリに保存した上で、キューに入れる。
  - 目的の温度に辿り着いたら、遷移回数を出力する。

幅優先探索の実装　演習

AtCoder

- ソースコード
  - http://arc001.contest.atcoder.jp/submissions/145113

```
11    void run()
12    {
13        Scanner cin = new Scanner(System.in);
14        int A = cin.nextInt();
15        int B = cin.nextInt();
16
17        //幅優先探索の準備
18        Queue<Integer> q = new LinkedList<Integer>();
19        q.add(A);
20
21        //遷移回数を入れるMapを用意する
22        HashMap<Integer, Integer> hm = new HashMap<Integer, Integer>();
23        hm.put(A, 0);
24
25        //遷移先6箇所を配列にしておく
26        int[] v = new int[]{-10,-5,-1,1,5,10};
```

幅優先探索の実装　演習

AtCoder

- ソースコード

```
28          while(!q.isEmpty() && !hm.containsKey(B)){
29              int now = q.poll();
30              //６種類の遷移先を調べる
31              for(int i=0; i<v.length; i++){
32                  int next = now + v[i];
33
34                  //もし既にnextが発見されていたらcontinue
35                  if(hm.containsKey(next)) continue;
36
37                  //そうでなければ、nextをキューに追加
38                  q.add(next);
39                  //遷移回数も追加
40                  hm.put(next, hm.get(now) + 1);
41              }
42          }
43
44          System.out.println(hm.get(B));
45      }
46  }
```

深さ優先探索と幅優先探索の違い

深さ優先探索と幅優先探索の違い

AtCoder

- 深さ優先探索
  - 見つかった順番に調べていく
    - 辞書順最小を調べたり、とにかく全列挙したい時に適している！
  - 再帰関数を使って実装
    - 途中までの計算結果などを使いやすい
    - 解の復元が簡単！

- 幅優先探索
  - 近い順に調べていく
    - 最短距離や、最も短いものを探すのに適している！
  - whileでのループとキューを使って実装
    - 慣れれば実装は簡単？
    - 解の復元は大変

休憩！

再開はXX:XXから！

色々な探索

1. Bitを利用した二分木の全探索
2. 順列に対する全探索

BITを利用した二分木の全探索

1. bitをいかにして使うか？
2. Bitの実装

Bitを利用した二分木の全探索

AtCoder

- 問題
- OとXのみで構成されたN文字の文字列を列挙して、～～～～しなさい。
  - 深さ優先探索で行けるけど、再帰関数書くの大変・・・。

- 再帰関数を書かなくても、for文だけで書けてしまう！

Bitを利用した二分木の全探索

AtCoder

- 例えば、OとXのみで構成された5文字の文字列の全列挙をする時
  - for(int i=0; i<32;i++)
  - 実は、このfor文で全列挙出来てしまう

Bitを利用した二分木の全探索

AtCoder

- 整数を2進数で表す
  - 0 … 00000
  - 1 … 00001
  - 2 … 00010
  - 3 … 00011
  - 4 … 00100
  - …..
  - 31 … 11111
- 0から31までの数字はこんな感じ

Bitを利用した二分木の全探索

AtCoder

- 整数を2進数で表す
  - 0 … 00000 … OOOOO
  - 1 … 00001 … OOOOX
  - 2 … 00010 … OOOXO
  - 3 … 00011 … OOOXX
  - 4 … 00100 … OOXOO
  - …..
  - 31 … 11111 … XXXXX
- 0から31までの数字はこんな感じ
  - 0,1を、O,Xに対応させてしまえば良い！

Bitを利用した二分木の全探索

AtCoder

- 整数のk桁目のbitを、k番目の分岐に対する選択と解釈することにより、forループで処理可能になる！
  - ○×ゲームの解答を10回行った結果の全列挙
    - k番目のbitが0ならk問目は○、そうでなければ×
  - 香車の進み方の全列挙
    - k番目のbitが0ならkマス目には止まらない。そうでなければ止まる

- 他にも、整数のbitで表せるものはたくさんある！
  - 先ほどの手紙問題だと、「誰に手紙を渡したか」を整数1つで持つことも可能

Bitを利用した二分木の全探索

AtCoder

- 具体的な実装例

```
void run()
{
    Scanner cin = new Scanner(System.in);
    int N = cin.nextInt();

    for(int i=0;i<(1<<N);i++){
        String st = "";
        for(int j=0;j<N;j++){
            if((i>>j)%2==0) st += "o";
            else st+= "x";
        }
        System.out.println(st);
    }
}
```

Bitを利用した二分木の全探索

AtCoder

- 具体的な実装例
  - (1 << N)
    - 2のN乗。「2択がN回ある場合」の要素数。
  - ((i >> j) % 2)
    - iに対する、j番目のbit情報を取り出す。
      - » jは0から数えます。0-indexedです。
    - 例えば、i -> 12(= 1100)なら、
      - (12 >> 1) -> 6(110)
      - (12 >> 2) -> 3(11)
      - (12 >> 3) -> 1(1)
    - と、こんな感じで、右からj番目の要素を取り出せる

Bitを利用した二分木の全探索　実践

AtCoder

- ABC002 D問題 派閥
  - http://abc002.contest.atcoder.jp/tasks/abc002_4

- 問題概要
  - １２人の友人関係が与えられる
  - 全員が全員を友達だと思っているグループのうち、最大なものを作成したい
  - そのグループの人数を出力せよ

Bitを利用した二分木の全探索　実践

AtCoder

- やり方
  - グループの候補となる、人の集合を全て列挙する
    - それらのグループに対して、本当に人が友達同士になっているかどうか確認を行う。
    - 友達同士になっていれば、答えの最大値を更新する。
  - これは、深さ優先探索での列挙も出来るが、bitを使ったfor文での列挙の方がずっと楽！

Bitを利用した二分木の全探索　実践

AtCoder

- ソースコード
  - http://abc002.contest.atcoder.jp/submissions/145126

```
 8    void run()
 9    {
10        Scanner cin = new Scanner(System.in);
11        int N = cin.nextInt();
12        int M = cin.nextInt();
13
14        //友人情報を2次元配列に入れる
15        boolean[][] friend = new boolean[N][N];
16        for(int i=0;i<M;i++){
17            int a = cin.nextInt() - 1;
18            int b = cin.nextInt() - 1;
19            friend[a][b] = friend[b][a] = true;
20        }
21
22        int ret = 0;
```

Bitを利用した二分木の全探索　実践

AtCoder

- ソースコード

```
24      //forループとbitを用いた全列挙
25      for(int i=0;i<(1<<N);i++){
26          //この派閥の人数を数える
27          int count = 0;
28
29          boolean flag = true;
30          //任意のペアに対し、友達になっているかを確認する。
31          for(int a=0;a<N;a++){
32              if((i>>a)%2==0) continue;
33              count++;
34              for(int b=a+1;b<N;b++){
35                  if((i>>b)%2==0) continue;
36                  if(!friend[a][b]){
37                      flag = false;
38                  }
39              }
40
41          }
42          if(flag) ret = Math.max(ret, count);
43      }
44      System.out.println(ret);
45  }
```

Bitを利用した二分木の全探索　演習

AtCoder

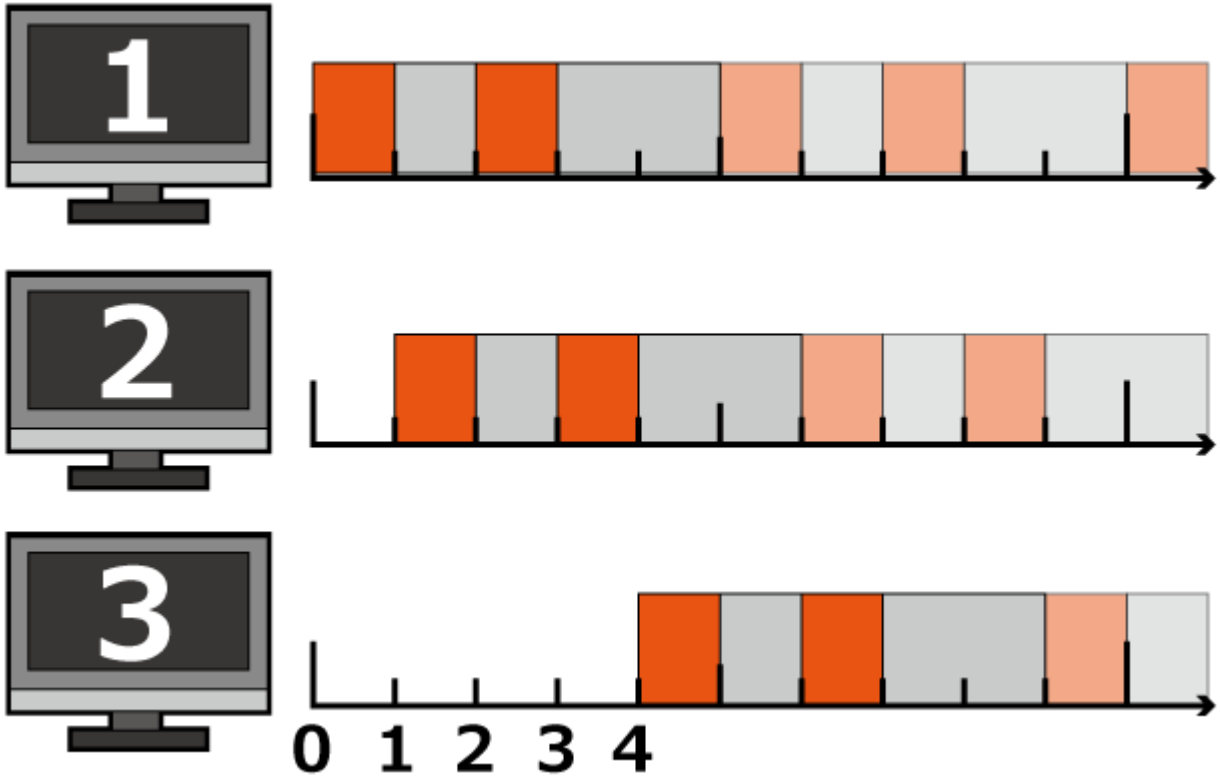
- ARC007 C問題 節約生活
  - http://arc007.contest.atcoder.jp/tasks/arc007_3

- 問題概要
  - 一定の周期で、映ったり映らなかったりを繰り返すテレビが複数台存在する
  - 全ての瞬間において、いずれかのテレビが映っているようにしたい
  - 必要なテレビの個数はいくつか？

Bitを利用した二分木の全探索　演習

AtCoder

- 問題概要
  - 例　oxoxx →　3つのテレビが必要



Bitを利用した二分木の全探索　演習

AtCoder

- 暇な人は以下の問題にチャレンジ！
- ARC014 C問題　魂の還る場所
  - http://arc014.contest.atcoder.jp/tasks/arc014_3
    - 部分点まで
- ARC017 C問題　無駄なものが嫌いな人
  - http://arc017.contest.atcoder.jp/tasks/arc017_3
    - そのままだと間に合わないので工夫が必要

Bitを利用した二分木の全探索　演習

AtCoder

- 解き方
  - テレビの再生時間をちょっとずつズラせば良い。
    - 0～9秒ずらす
      - 10種類しかない！
  - 10種類のテレビの使う・使わないの集合は、1024回のforループで書き表すことが出来る。
    - 各部分集合に対して、全てのタイミングでテレビが映っているかどうか調べれば良い。

Bitを利用した二分木の全探索　演習

AtCoder

- ソースコード
  - http://arc007.contest.atcoder.jp/submissions/145128

```
 8    void run()
 9    {
10        Scanner cin = new Scanner(System.in);
11        String st = cin.next();
12        int N = st.length();
13
14        //ox情報をboolean型の配列に写す
15        boolean[] b = new boolean[N];
16        for(int i=0;i<N;i++){
17            if(st.charAt(i) == 'o') b[i] = true;
18        }
```

Bitを利用した二分木の全探索　演習

AtCoder

- ソースコード

```
20      //forループを用いた全探索
21      int ret = 999;
22      for(int i=0;i<(1<<N);i++){
23          int count = 0;
24          //どのタイミングでテレビが付いているかをメモする
25          boolean[] now = new boolean[N];
26          
27          //各秒数に対して、それだけずらしたテレビが存在するか判定
28          for(int j=0;j<N;j++){
29              if((i>>j) % 2 == 0) continue;
30              count++;
31              
```

Bitを利用した二分木の全探索　演習

AtCoder

- ソースコード

```
20          //forループを用いた全探索
21          int ret = 999;
22          for(int i=0;i<(1<<N);i++){
23              int count = 0;
24              //どのタイミングでテレビが付いているかをメモする
25              boolean[] now = new boolean[N];
26              
27              //各秒数に対して、それだけずらしたテレビが存在するか判定
28              for(int j=0;j<N;j++){
29                  if((i>>j) % 2 == 0) continue;
30                  count++;
31                  
```

Bitを利用した二分木の全探索　演習

AtCoder

- おまけ
- 探索を整数のbit列で手抜きしたが、あるタイミングにテレビが映っていたかの判定も、bit列を使ってすることが出来る。

Bitを利用した二分木の全探索　演習

AtCoder

- ソースコード2
  - http://arc007.contest.atcoder.jp/submissions/145135

```
8    void run()
9    {
10       Scanner cin = new Scanner(System.in);
11       String st = cin.next();
12       int N = st.length();
13
14       //ox情報を整数に直す
15       int b = 0;
16       for(int i=0;i<N;i++){
17           if(st.charAt(i) == 'o') b |= (1<<i);
18       }
19
```

Bitを利用した二分木の全探索　演習

AtCoder

- ソースコード2

```
20          //forループを用いた全探索
21          int ret = 999;
22          for(int i=0;i<(1<<N);i++){
23              int count = 0;
24              int now = 0;
25              //各秒数に対して、それだけずらしたテレビが存在するか判定
26              for(int j=0;j<N;j++){
27                  if((i>>j) % 2 == 0) continue;
28                  count++;
29
30                  //各タイミングに対し、映像が映るかどうかの判定をする
31                  now |= b << j;
32                  now |= (b << j) >> N;
33
34              }
35              int target = (1<<N) - 1;
36              if((now & target) == target) ret = Math.min(ret, count);
37
38          }
39          System.out.println(ret);
40      }
41  }
```

順列に対する列挙

順列に対する列挙

AtCoder

- 順列を列挙したい時、結構あると思います。
  - {1,2,3}に対する列挙
    - {1,2,3}, {1,3,2}, {2,1,3}, {2,3,1}, {3,1,2}, {3,2,1}
  - {1,2,2}に対する列挙
    - {1,2,2}, {2,1,2}, {2,2,1}
- これらは、深さ優先探索を用いれば、問題なく列挙出来る。
- でも、深さ優先探索を組むのは結構面倒！

順列に対する列挙

AtCoder

- C++では順列の列挙は簡単？
  - next_permutationという、順列の列挙に役立つ便利なライブラリが入っている
- じゃあJavaは？
  - 便利なライブラリがないので、自分で書いておきましょう。
  - さっき手紙の問題で書いたね！やったね！

本日のまとめ

本日のまとめ

AtCoder

- あらゆるものを全列挙可能な、2つの探索方法
  - 深さ優先探索
  - 幅優先探索
  - この2つをマスターすれば、全探索問題は怖くない！
- 全列挙を楽にする、2つのツール
  - bitを使った、二分木の全探索
  - 順列に対する全探索

- これらが理解できていればOKです！