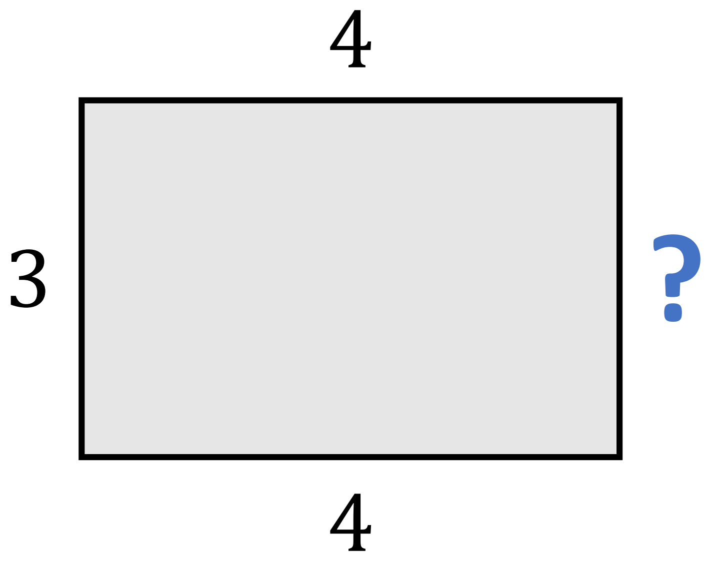
ABC #027 解説

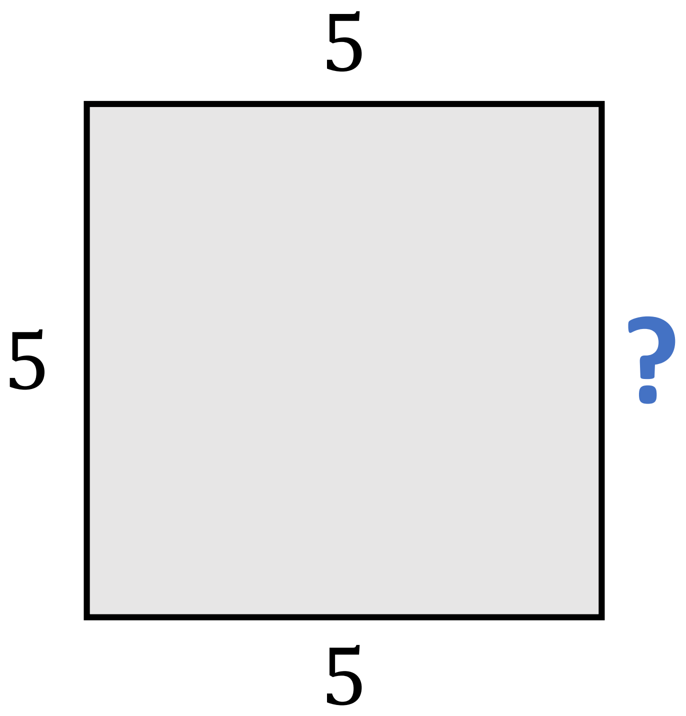
解説スライド担当 : @sugim48

問題 A – 長方形

問題概要

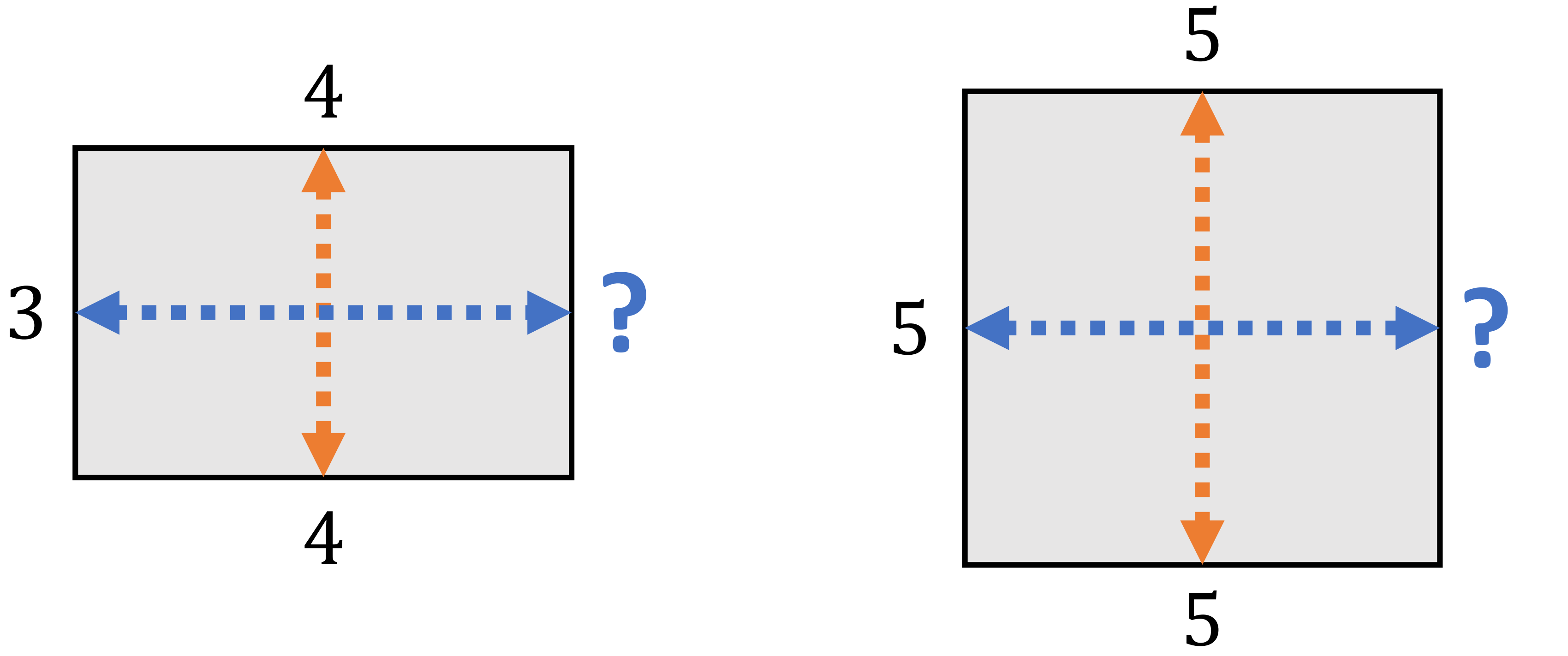
- ある長方形（正方形も含む）の 3 つの辺の長さが与えられる。残り 1 つの辺の長さを求めよ。





考察

- 向かい合う辺の長さは等しい。



解法

- 与えられた 3 つの数のうち、等しい組を見つけたら、余った数がそのまま答えになる。

解答例 (C++)

```
int x, y, z;
cin >> x >> y >> z;


int ans;
if (x == y) ans = z;
if (y == z) ans = x;
if (z == x) ans = y;


cout << ans << endl;
```

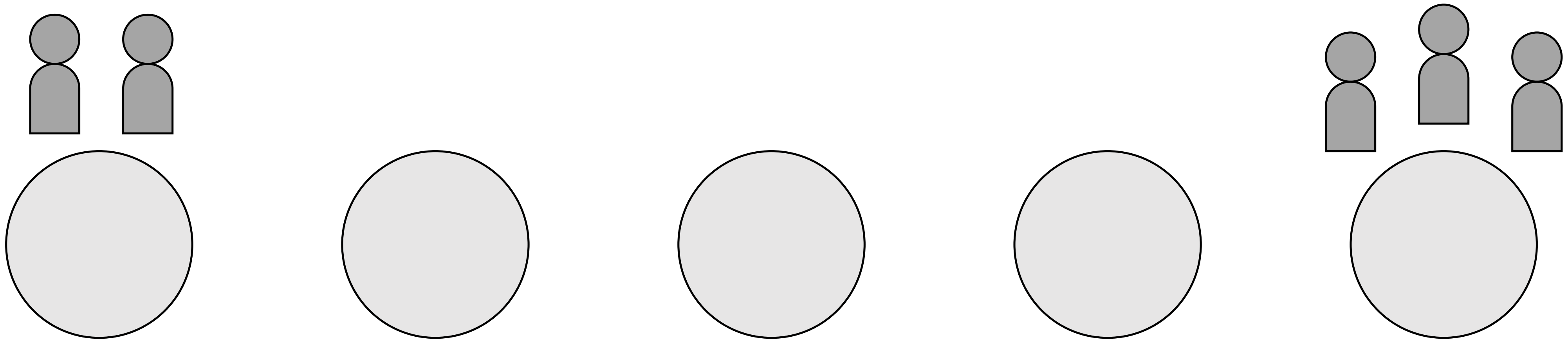
問題 B – 島と橋

問題概要

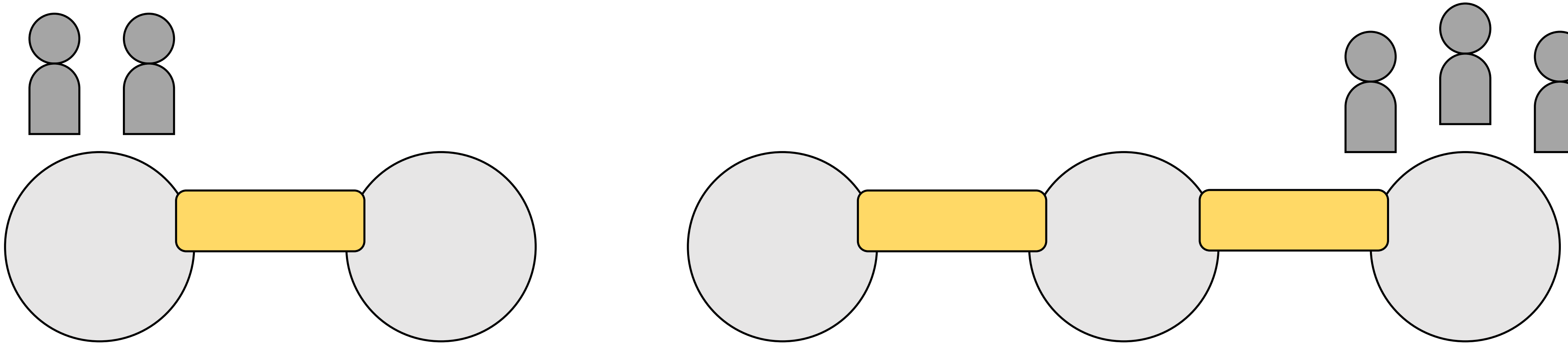
- N 個の島が横一列に並んでいる。左から i 番目の島には a_i 人の住人が住んでいる。
- 隣り合う島の間に橋を架け、住人を移動させることができる。
- すべての島に同じ人数の住人が住むようにできるか？ また、最小で何本の橋を架ければよいか？

- $2 \leq N \leq 100$
- $0 \leq a_i \leq 100$

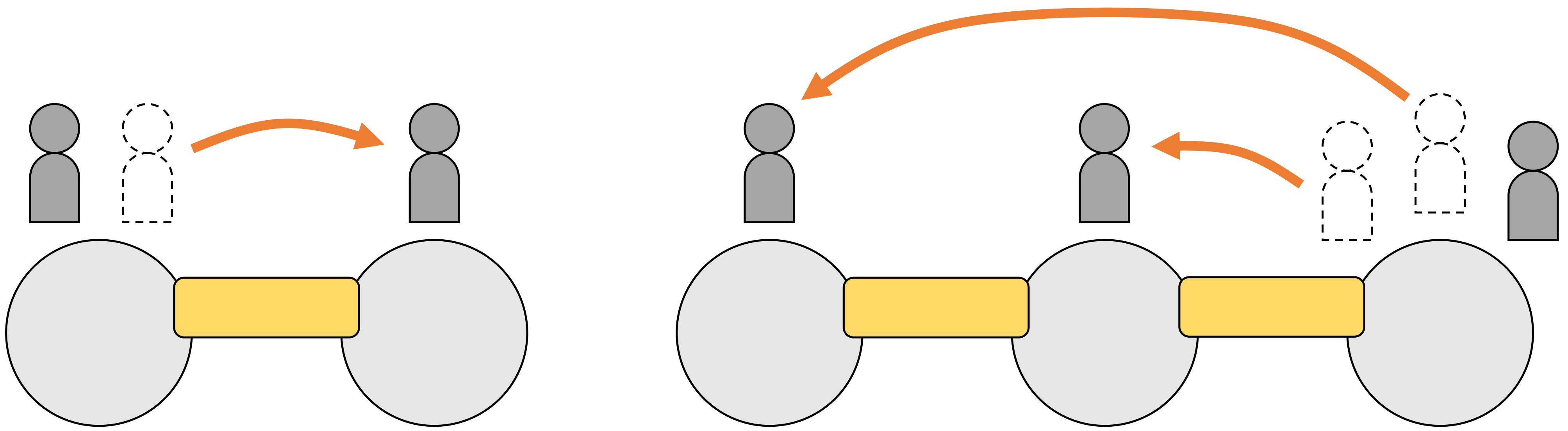
例



例



例



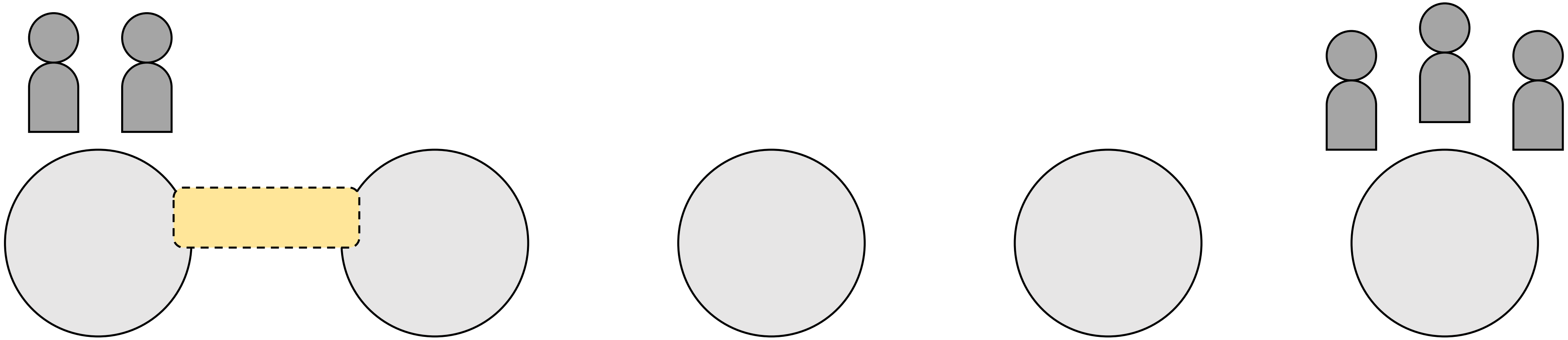
解法

- まず、$\sum_{i=1}^{N} a_i$ が N で割り切れなければ、不可能である。
- 割り切れるならば、それぞれの島に $\frac{1}{N}\sum_{i=1}^{N} a_i$ 人ずつ住むことになる。

解法

- 隣り合う島の間ごとに橋が必要か判定していく。
- 下図の橋は必要か？ 橋の左側が 1 人、橋の右側が 4 人になってほしいので、橋を左から右へ 1 人渡ることになる。

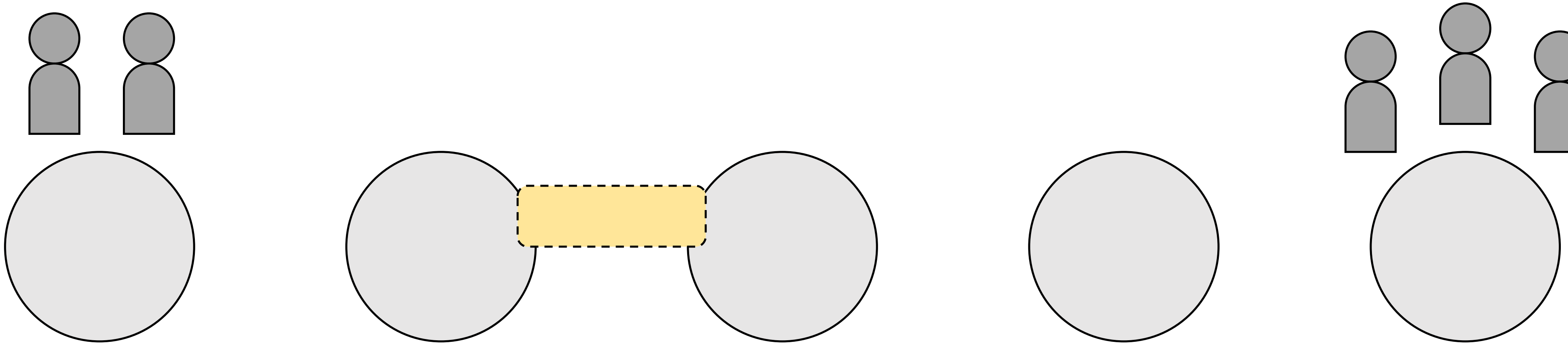
→ 必要



解法

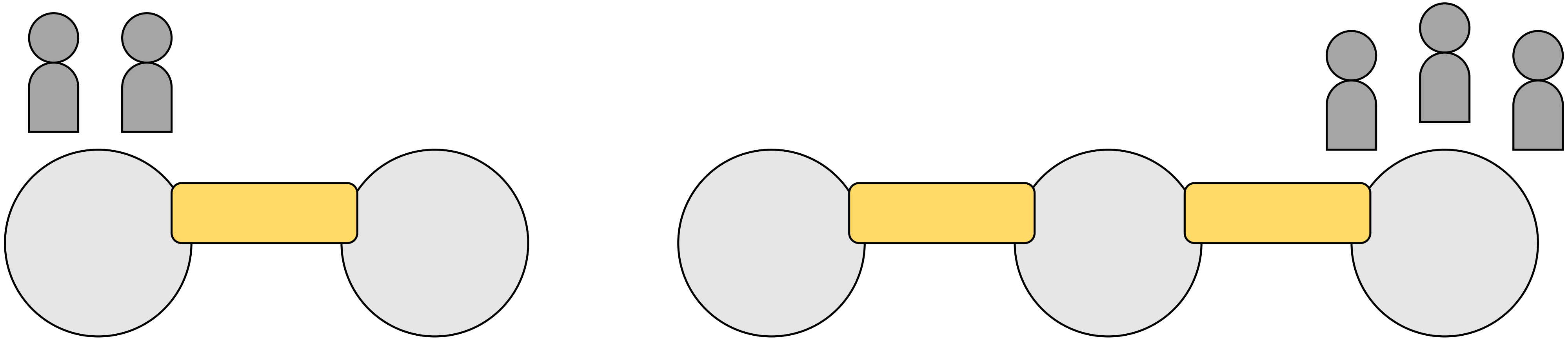
- 下図の橋は必要か？ 橋の左側が 2 人、橋の右側が 3 人になってほしいが、はじめからそうなっている。

→ 必要ではない



解法

- このようにして、隣り合う島の間ごとに橋が必要か判定し、必要と判定された橋の本数を答えればよい。



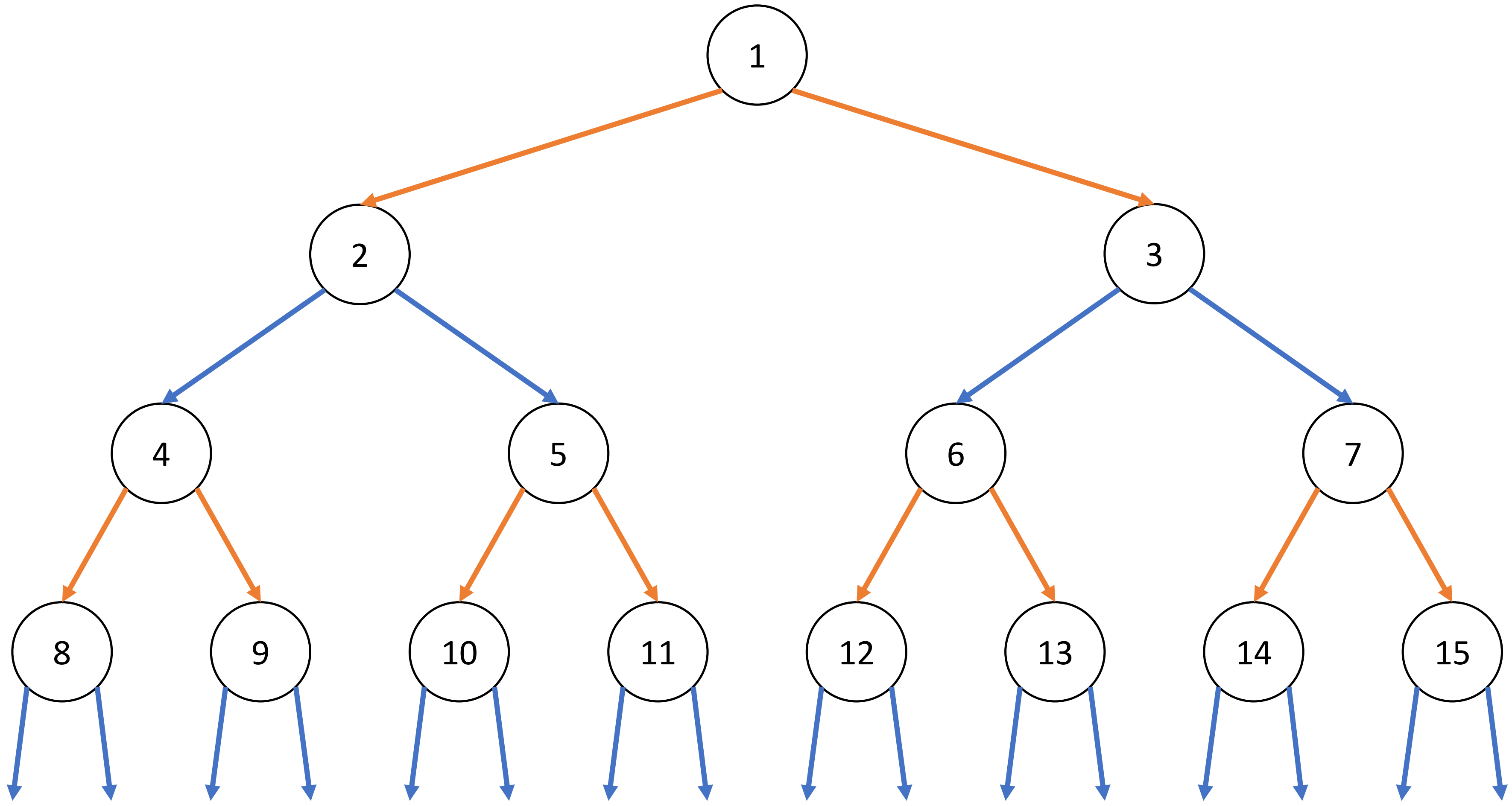
問題 C – 倍々ゲーム

問題概要

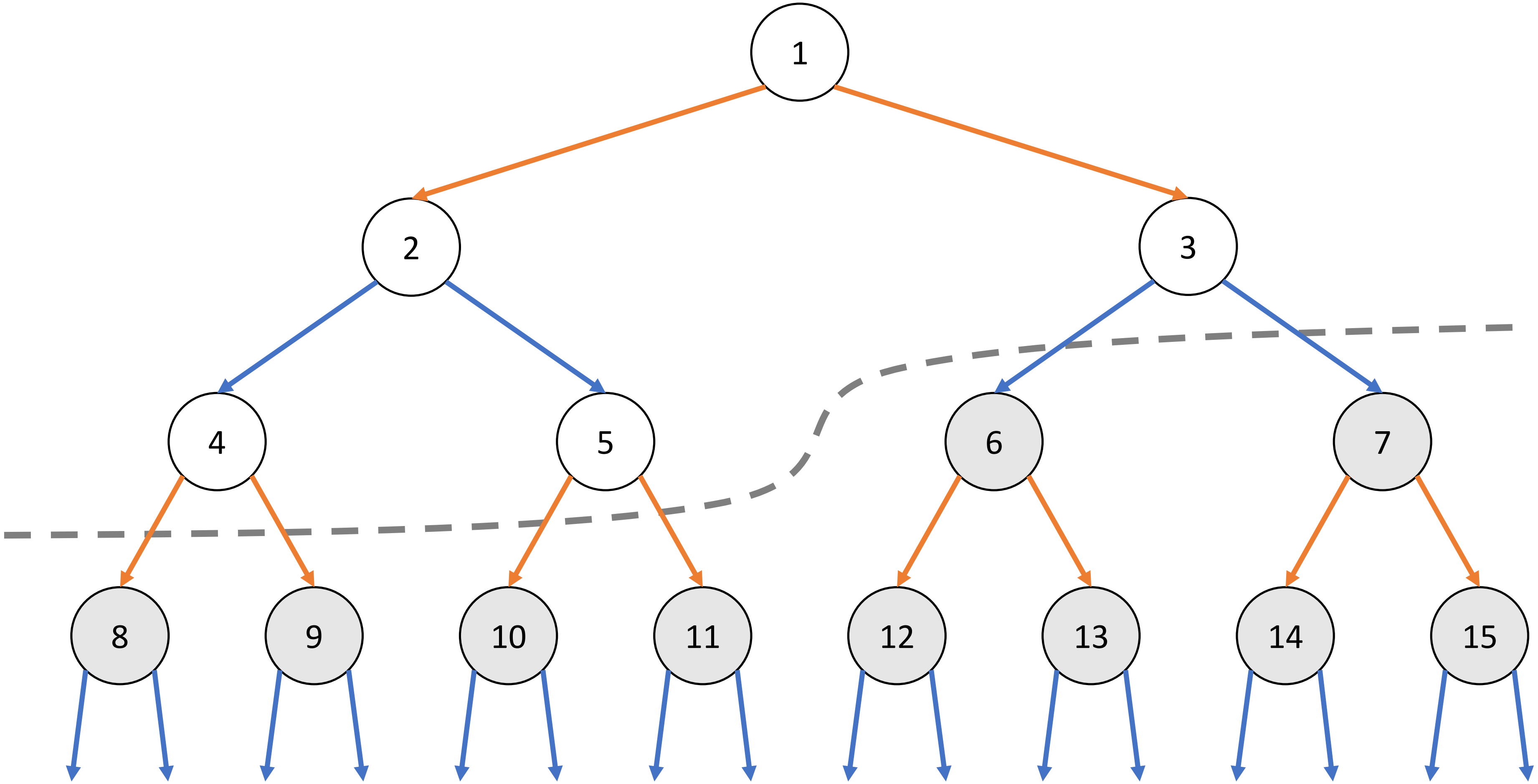
- A と B が二人ゲームで勝負する。
- 自然数 N が与えられる。$x = 1$ に初期化する。
- A → B → A → … の順に次の操作を行う。
  - ➢ x を $2x$ または $2x + 1$ に置き換える。
- $x > N$ にした人が負け。
- どちらかが勝つか求めよ。

- $1 \leq N \leq 10^{18}$

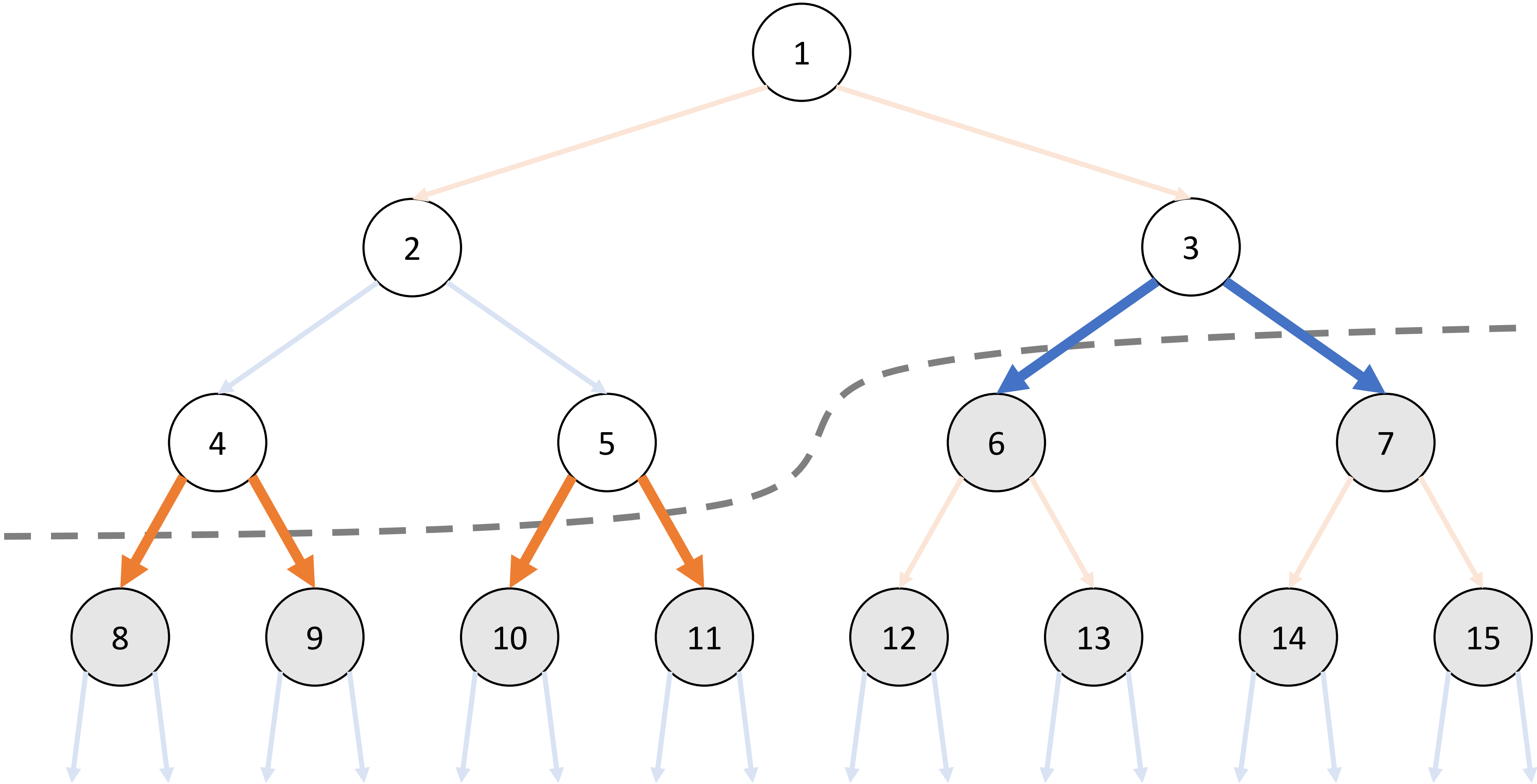
- A の操作を赤、B の操作を青で表すと、図のように x が変化する。



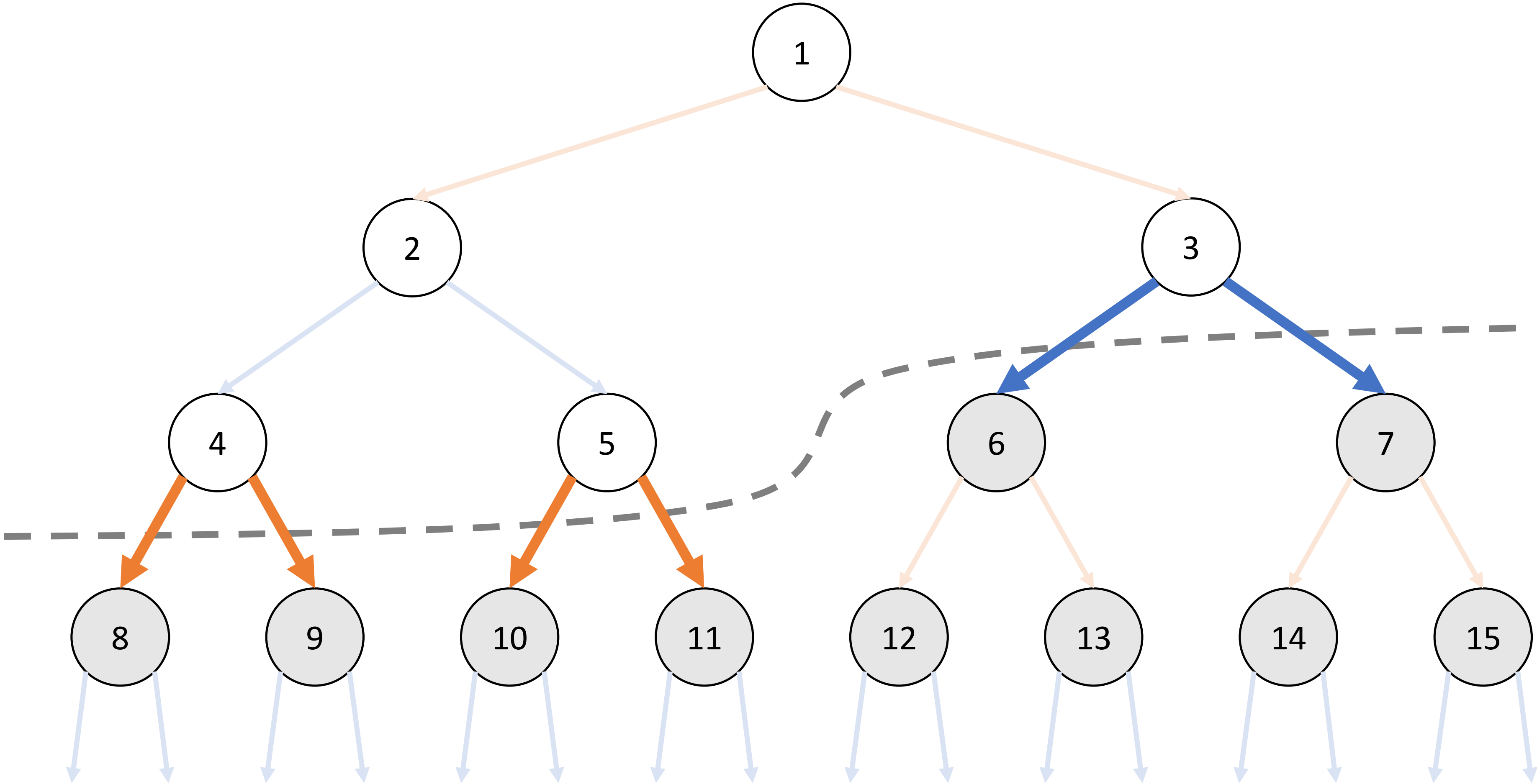
- 例えば $N=5$ のとき、OK の整数と NG の整数はこのように分離される。



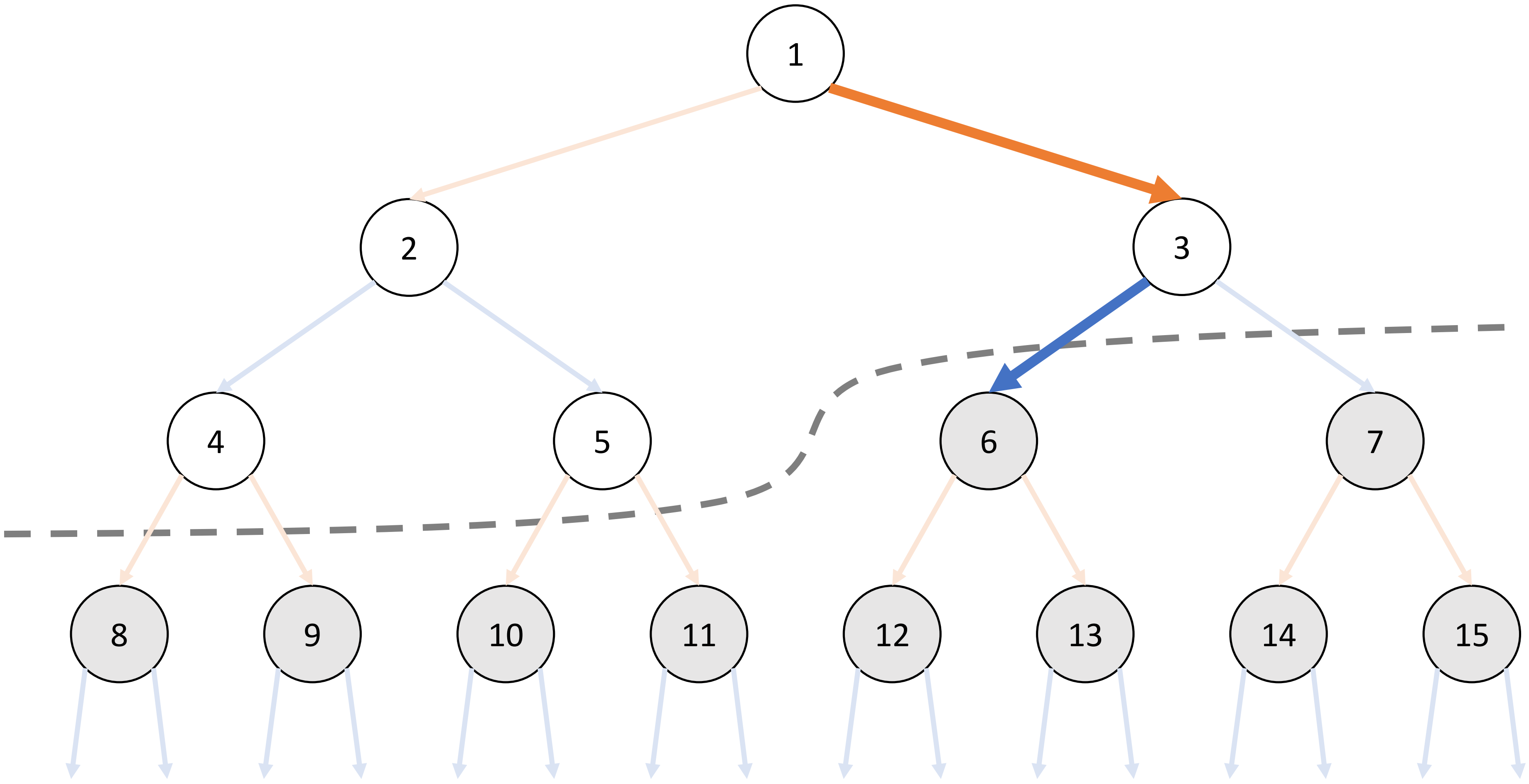
- 境界線をまたぐ操作だけに注目すると、A の操作は左に、B の操作は右に偏っている。



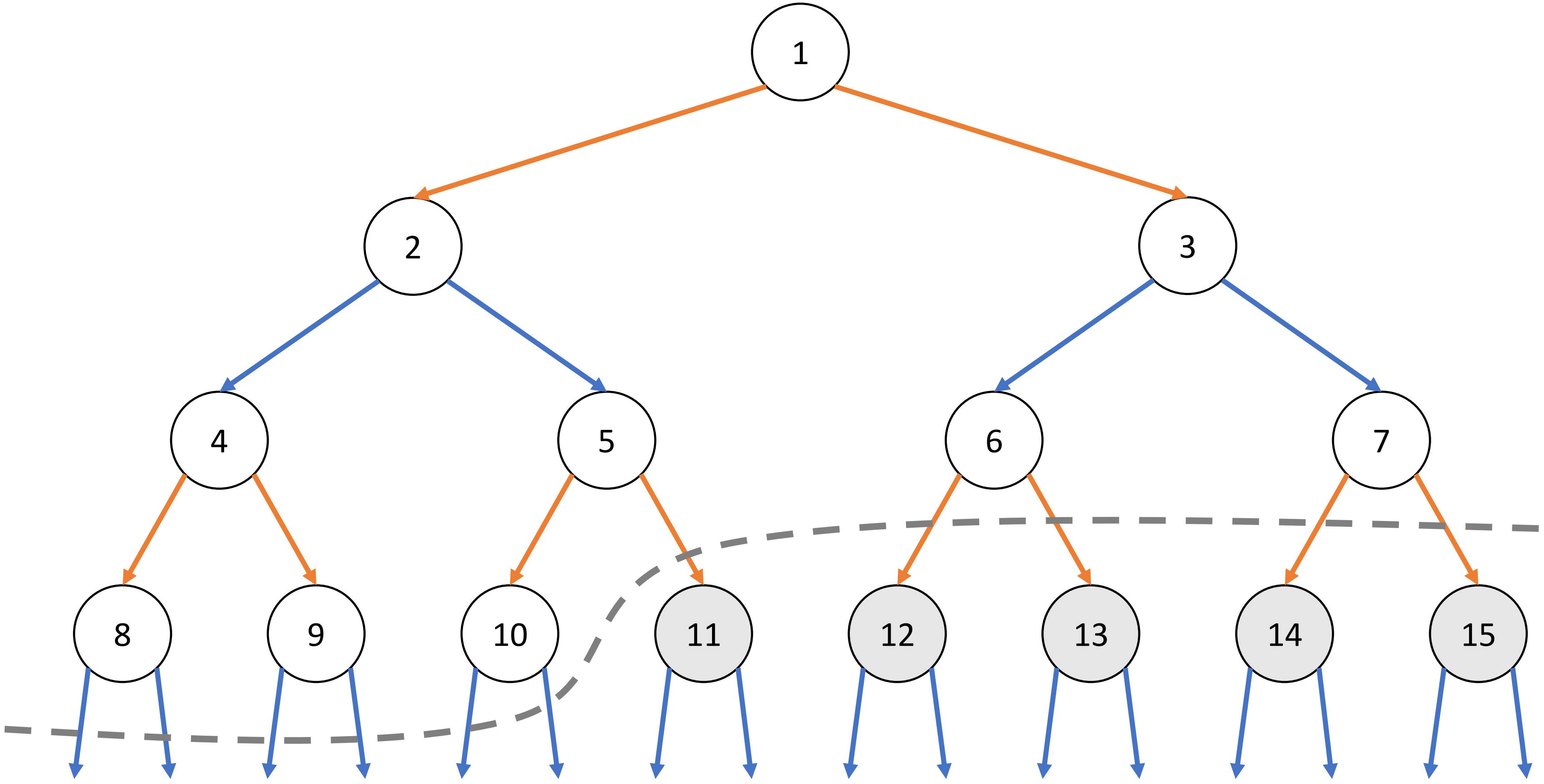
- 境界線をまたぐと負けてしまうので、A はできるだけ<u>右</u>に、B はできるだけ<u>左</u>に行きたがることが分かる。



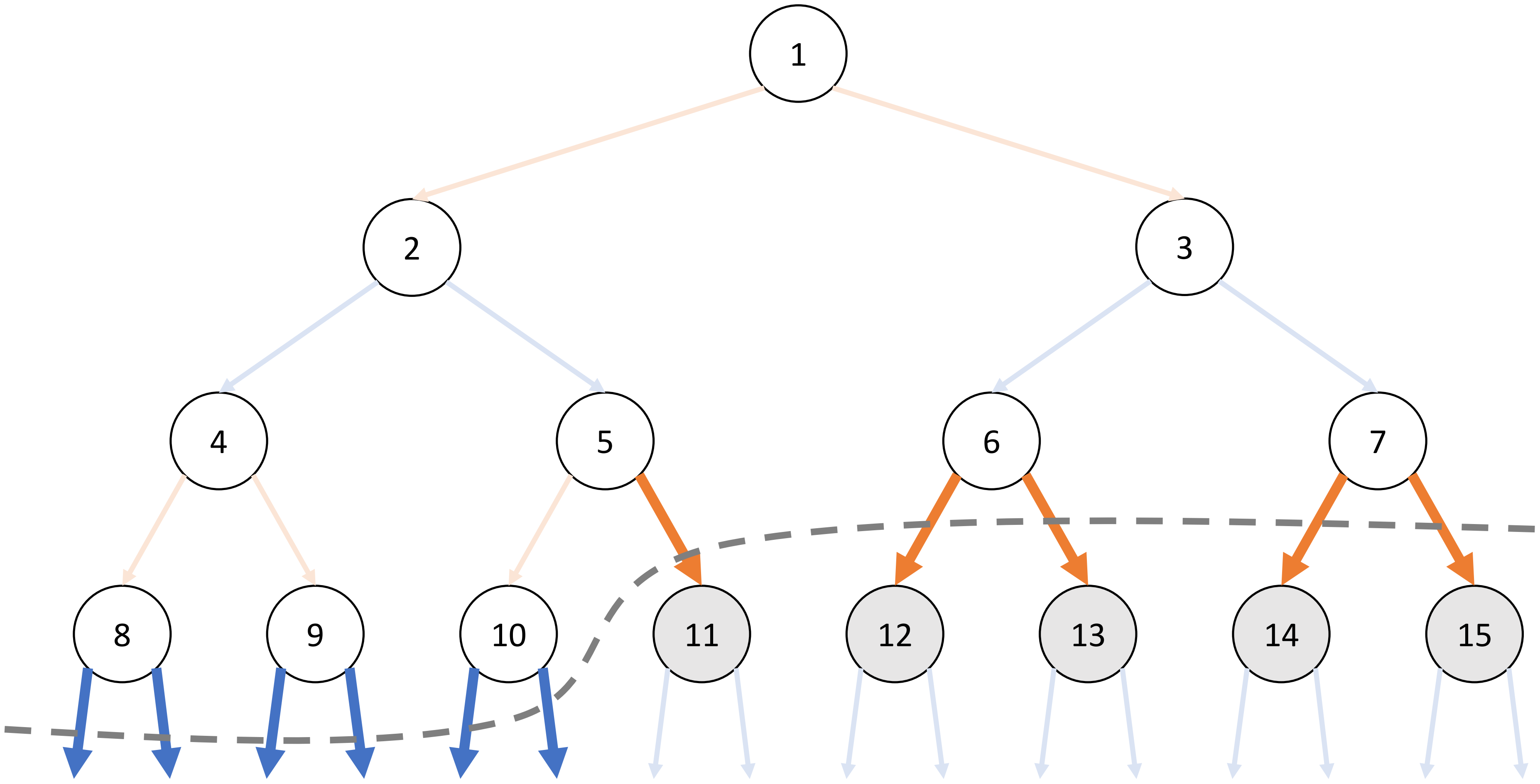
- これを実際にシミュレートすると B が負けると判定できる。



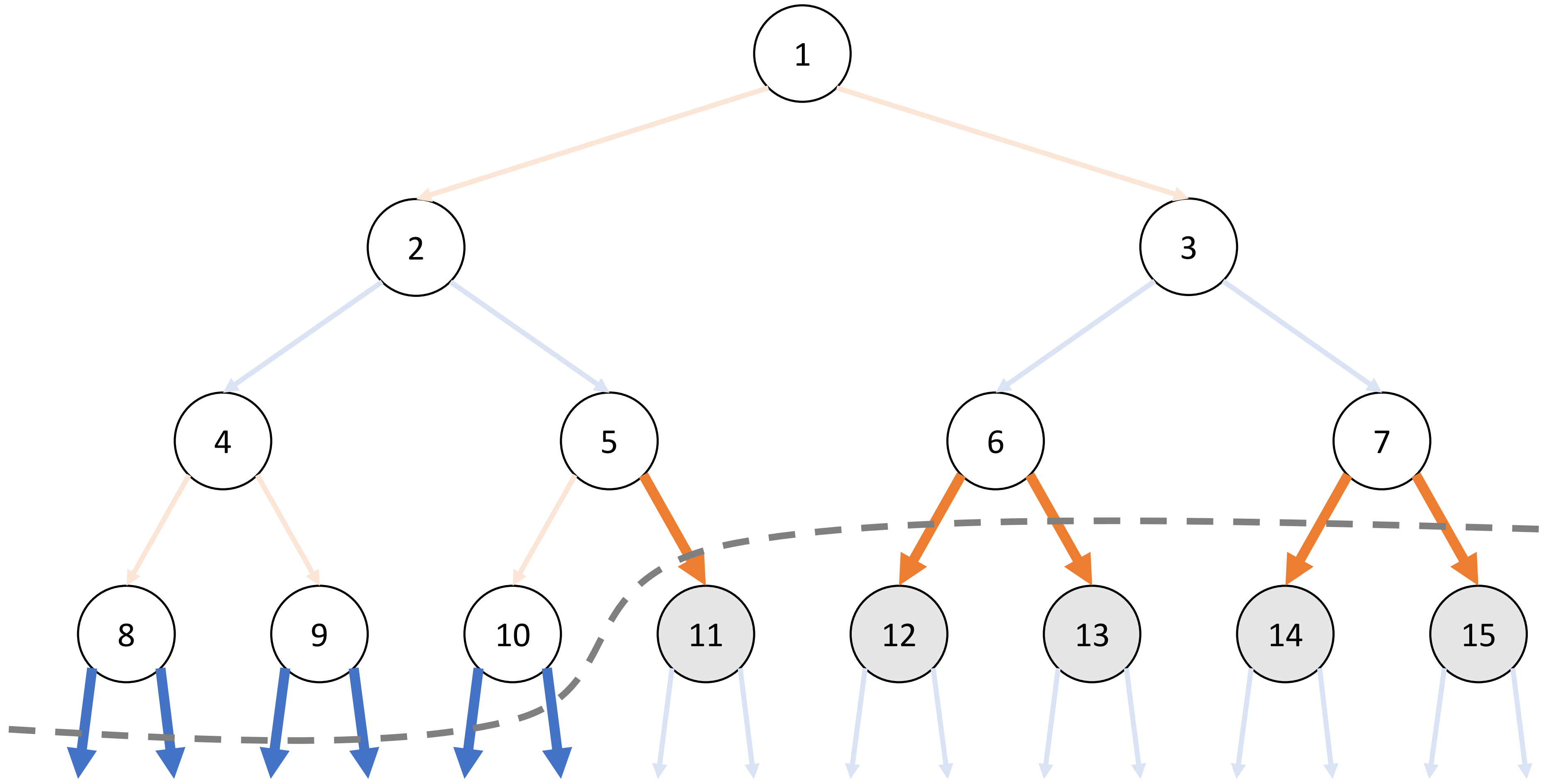
- 別の例として $N = 10$ のとき、OK の整数と NG の整数はこのように分離される。



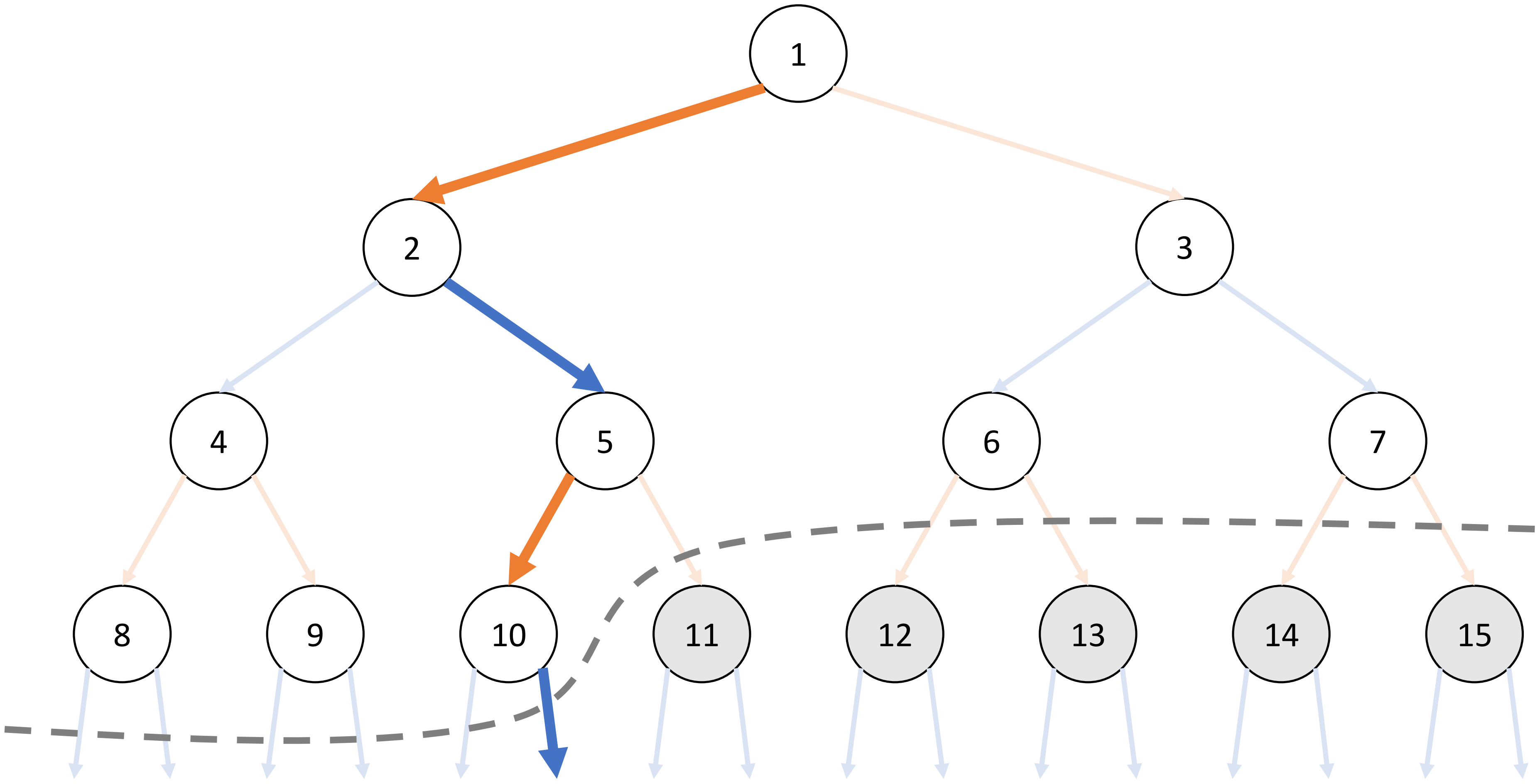
- 境界線をまたぐ操作だけに注目すると、A の操作は<u>右</u>に、B の操作は<u>左</u>に偏っている。



- 境界線をまたぐと負けてしまうので、A はできるだけ<u>左</u>に、B はできるだけ<u>右</u>に行きたがることが分かる。



- これを実際にシミュレートすると B が負けると判定できる。



解法

- N の深さの偶奇に応じて、A と B の戦略が決まる。
- A と B の戦略を実際にシミュレートして、どちらかが勝つか判定する。

- N の深さは次のようにして $O(logN)$ 時間で計算できる。

```
int depth = 0;
for (long long n = N; n > 0; n /= 2)
        depth++;
```

問題 D – ロボット

問題概要

- 数直線の原点にロボットが置かれている。はじめ、ロボットの幸福度は 0 である。
- このロボットが命令列 S を順に実行する。
  - **M** : 正か負の向きに距離 1 だけ移動する。
  - **+** : 今の座標を x とすると、幸福度が $+x$ だけ変化する。
  - **-** : 今の座標を x とすると、幸福度が -x だけ変化する。
- 最終的にロボットは原点に戻っていなければならない。
- 最終的な幸福度の最大値を求めよ。

- $1 \leq |S| \leq 10^5$

部分点解法

- $1 \leq |S| \leq 1{,}000$ と小さい。

→ 動的計画法

- dp[何文字目][座標] $\coloneqq$ (幸福度の最大値) を埋めていく。
- $dp[|S|][0]$ が答え。

- $O(|S|^2)$ で間に合う。

満点解法

- $1 \leq |S| \leq 10^5$ と大きいので、動的計画法では間に合わない。

→ もっと速い解法を考える。

- ロボットの正の向きへの移動を **>** 、負の向きへの移動を **<** と表すことにする。

考察

- >+<<-> という命令列を考える。

- 幸福度の変化量は
  - + ごとに (自分より左の > の個数) - (自分より左の < の個数)
  - - ごとに (自分より左の < の個数) - (自分より左の > の個数)

- 見方を変えると
  - > ごとに (自分より右の + の個数) - (自分より右の - の個数)
  - < ごとに (自分より右の - の個数) - (自分より右の + の個数)

考察

- (自分より右の + の個数) と (自分より右の - の個数) が分かれば、
  > または < を選んだときの幸福度の変化量を予言できる！

- 例） **M--M-M+M+**

	M	**-**	**-**	**M**	**-**	**M**	**+**	**M**	**+**
>	-1			$+1$		$+2$		$+1$	
<	$+1$			-1		-2		-1	

考察

- 最終的な幸福度を最大化したいので、幸福度が増える向きを貪欲に選んでいけばいいいか？

→「最終的にロボットは原点に戻っていなければならない」という条件を守れない。

	M	-	-	M	-	M	+	M	+
>	-1			$+1$		$+2$		$+1$	
<	$+1$			-1		-2		-1	

考察

- 最終的にロボットが原点に戻るためには、> と < を同じ回数だけ選ばなければならない。
- この制約下でできるだけ大きいものを選びたい。

	M	-	-	M	-	M	+	M	+
>	-1			$+1$		$+2$		$+1$	
<	$+1$			-1		-2		-1	

考察

- この行を昇順にソートすると → $[-1, +1, +1, +2]$
- 前半分を < に、後ろ半分を > に割り当てる → $[-1, +1, +1, +2]$
- $\{(+1) + (+2)\} - \{(-1) + (+1)\} = 3$ が答え！

	M	-	-	M	-	M	+	M	+
>	-1			$+1$		$+2$		$+1$	
<	$+1$			-1		-2		-1	

解法

- 命令列 S の各 **M** について、
  (自分より右の **+** の個数) - (自分より右の **-** の個数) を計算し、
  配列 A に格納する。
- A を昇順にソートする。
- (A の後ろ半分の総和) – (A の前半分の総和) が答え。

- $O(|S| \log|S|)$ で間に合う。
- なお、バケツソートを用いると $O(|S|)$