春合宿 Day 3

自然公園(Natural Park)　解説

チューター : 城下慎也 – IOI 2011 タイ大会 日本代表

2017/3/22

問題概要

- N 頂点 M 辺のグラフが与えられる。
- 2 つの頂点 A, B と、他に使用可能な頂点の集合を指定した際に、A, B 間を移動可能かを質問することができる。
- M 本の辺すべてを特定せよ。

- $N \leqq 1{,}400, M \leqq 1{,}500$, 質問回数 $\leqq 45{,}000$
- ***どの頂点も次数が 7 以下。***

問題概要

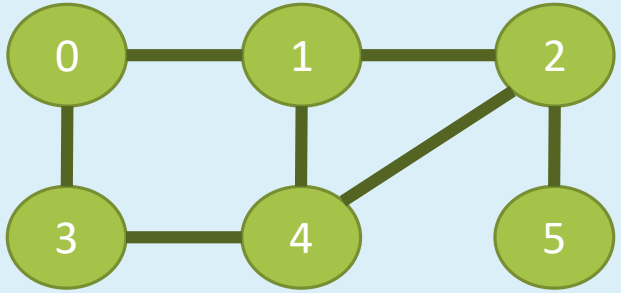
- N 頂点 M 辺のグラフが与えられる。
- 2 つの頂点 A, B と、他に使用可能な頂点の集合を指定した際に、A, B 間を移動可能かを質問することができる。
- M 本の辺すべてを特定せよ。

- $N \leqq 1{,}400, M \leqq 1{,}500$, 質問回数 $\leqq 45{,}000$
- ***どの頂点も次数が 7 以下。***

→一見重要そうではあるが実は罠で、**小課題 5 まで使用しない**。

例

- サンプルの図で考える。

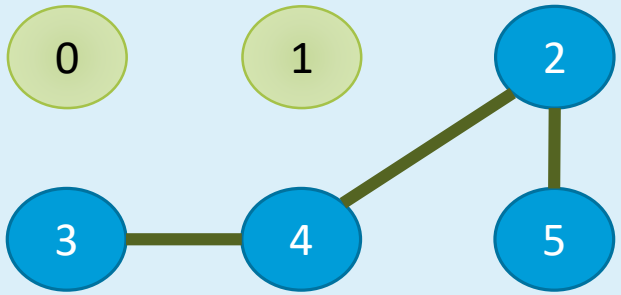


質問[1]
頂点 2,3,4,5 のみ使って
頂点 3 から 5 に行ける？

IOIちゃん（想像図）

例

- サンプルの図で考える。

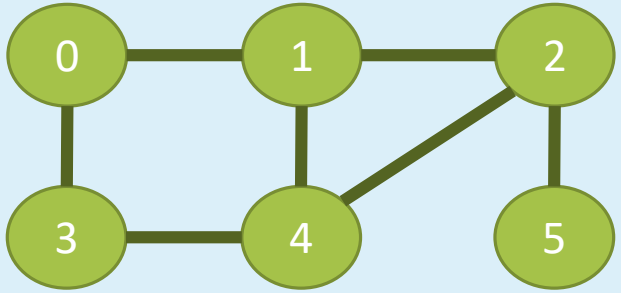


質問[1]
頂点 2,3,4,5 のみ使って
頂点 3 から 5 に行ける？

IOIちゃん（想像図）

例

- サンプルの図で考える。

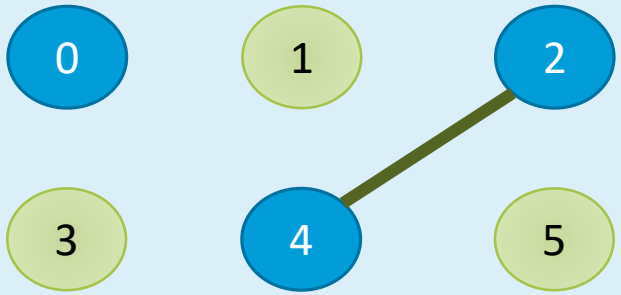


質問[2]
頂点 0,2,4 のみ使って
頂点 0 から 4 に行ける？

IOIちゃん（想像図）

例

- サンプルの図で考える。



質問[2]
頂点 0,2,4 のみ使って
頂点 0 から 4 に行ける？

IOIちゃん（想像図）

小課題 1 (10 点) 解法

- N が小さい。
- N が小さいので、すべての A, B の組について質問する余裕がある。

→ A, B 以外の頂点を使用不可能にすれば、結果は A, B 間の道の有無にしか依存しない！

- よって、すべての A, B の組について、その組となる辺があるかを判定すればよい。

小課題 2 (10 点)

- 島の構造が直線である (頂点 0 および $N-1$ が両端)。
- 頂点 0 から隣を求めていこうかな…

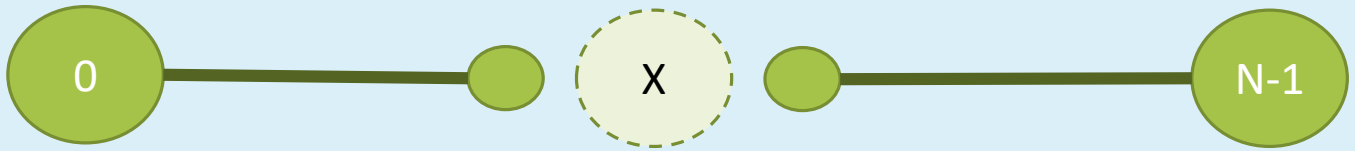
小課題 2 (10 点)

- 島の構造が直線である (頂点 0 および $N-1$ が両端)。
- 頂点 0 から隣を求めていこうかな… → 残念ながらうまくいかない。

- いろいろ試してみて手がかりを探っていこう！

小課題 2 (10 点)

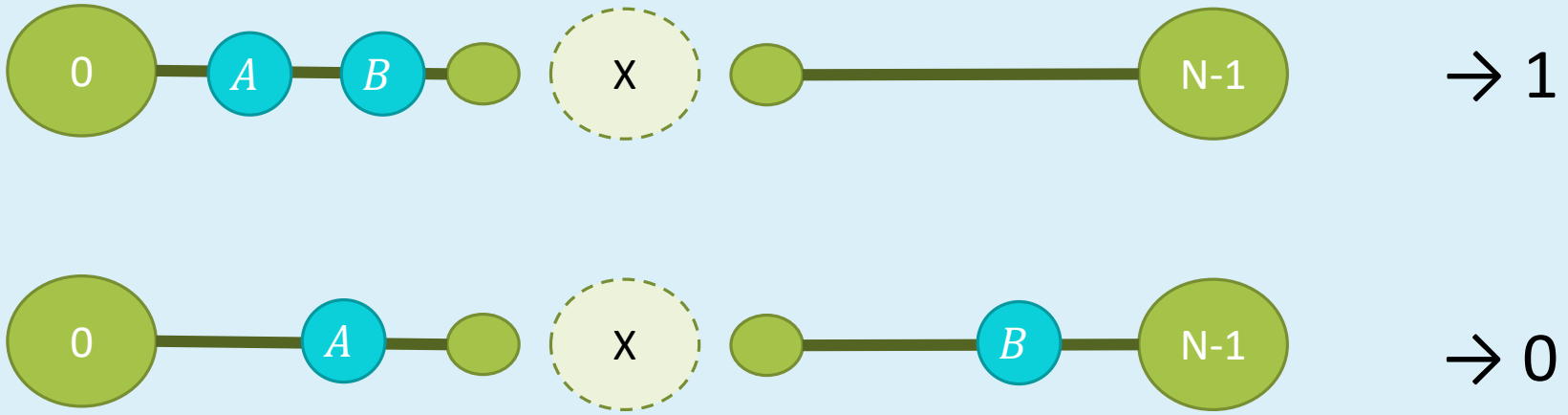
- そもそも小課題 1 では質問の性質を（ほぼ）使っていない。
- ある頂点 X のみを使用しないことにした場合にどうなるか考えてみよう。



小課題 2 (10 点)

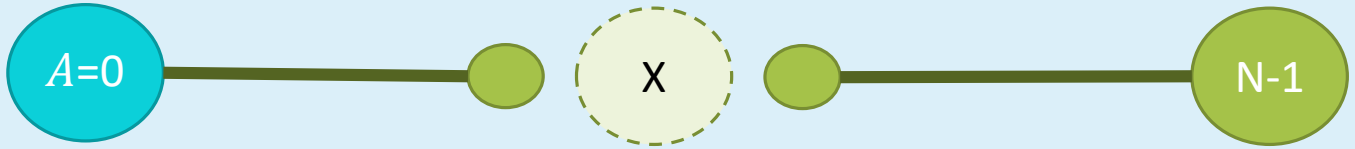
- そもそも小課題 1 では質問の性質を（ほぼ）使っていない。
- ある頂点 X のみを使用しないことにした場合にどうなるか考えてみよう。

→ A, B が同じ側なら 1、そうでないなら 0 が返ってくる。



小課題 2 (10 点)

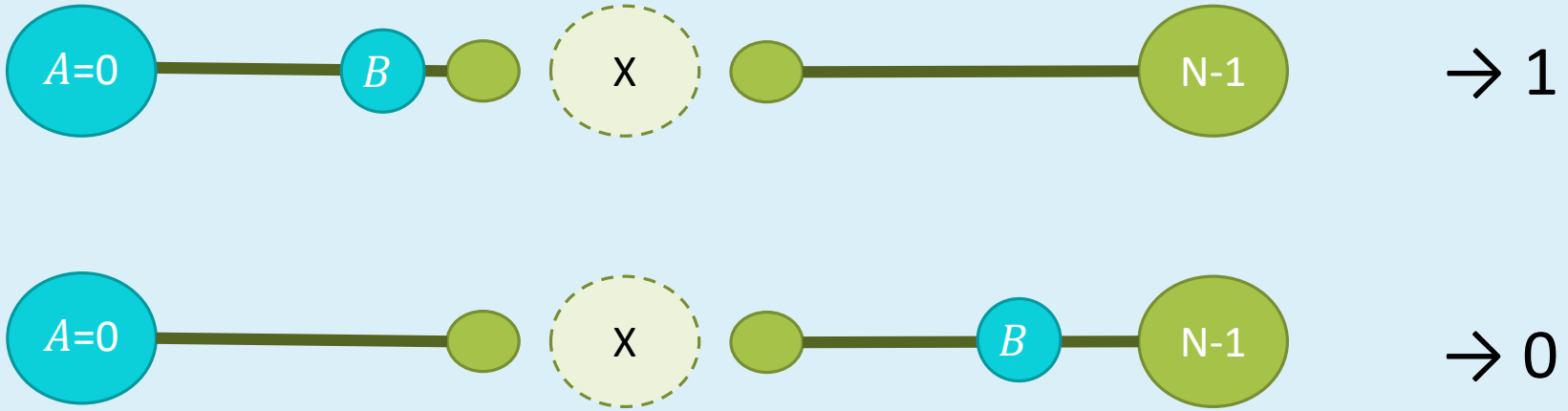
- じゃあ、さらに $A = 0$ と固定した場合は？



小課題 2 (10 点)

- じゃあ、さらに $A = 0$ と固定した場合は？

→ B が X より A に**より近い**なら 1、そうでないなら 0 が返ってくる。

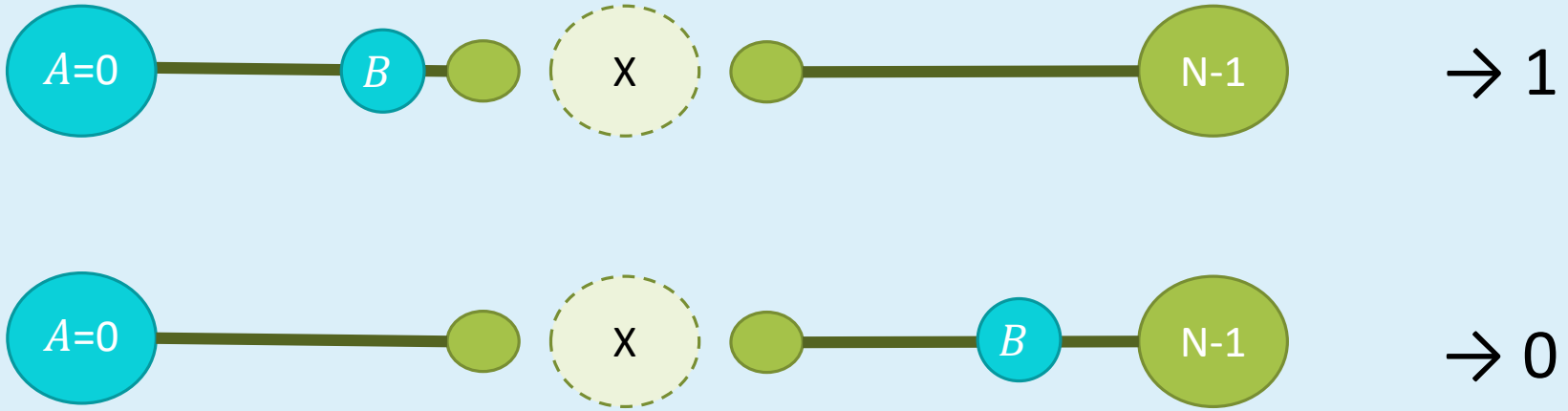


小課題 2 (10 点)

- じゃあ、さらに $A = 0$ と固定した場合は？

→ B が X より A に**より近い**なら 1、そうでないなら 0 が返ってくる。

比較関数では！？



小課題 2 (10 点) 解法

- $1,2,\ldots,N-2$ をソートします。
- 頂点 A,B を比べたいとき、比較関数は次のようにすればよい:
  1. 頂点 A のみ使用不可、それ以外使用可能にして頂点 0 から頂点 B に行けるか尋ねる。
  2. もし行けるなら、B の方が小さい(0 に近い)
  2. 行けないなら、A の方が小さい(0 に近い)

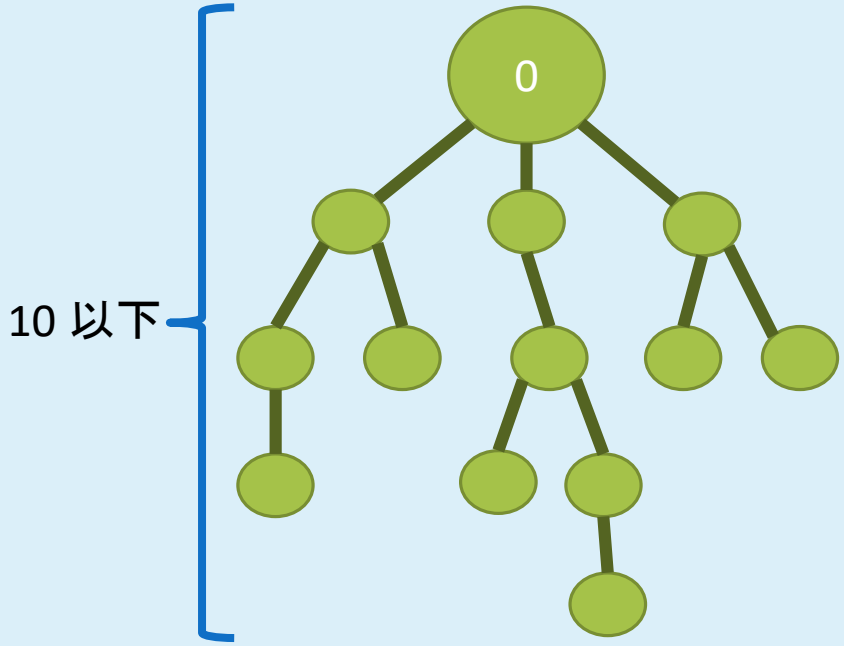
- ソート後、0, [ソート列], $N-1$ の順に繋いで出力します。
- $O(N\log N)$ のソートを用いれば良い。

小課題 3 (27 点)

- 小課題 2 とは全く異なる形状となっている。
- 頂点 0 から 8 個以下の中継地しか必要とせずにすべての頂点に移動可能な木構造である。→どういうこと？

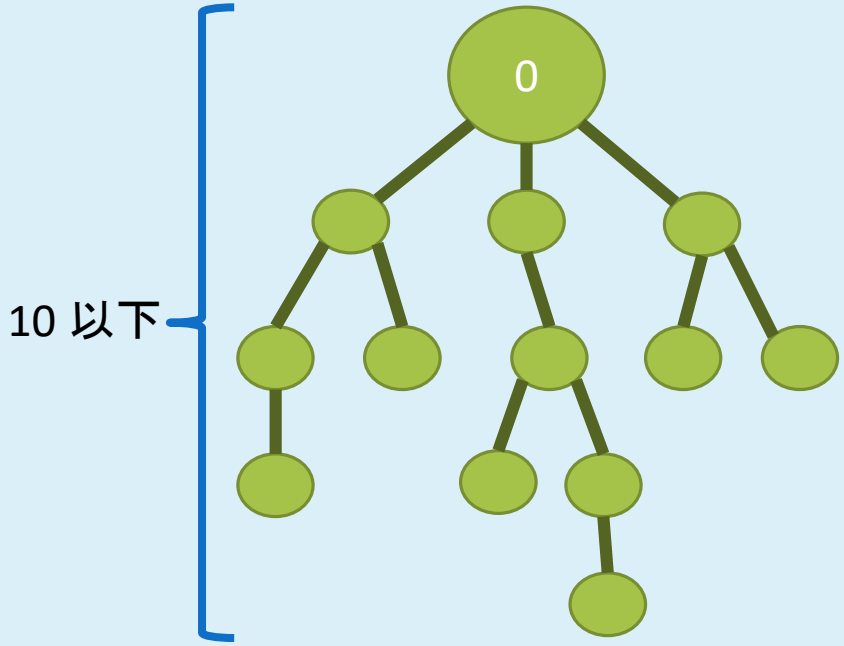
小課題 3 (27 点)

- 小課題 2 とは全く異なる形状となっている。
- 頂点 0 から 8 個以下の中継地しか必要とせずにすべての頂点に移動可能な木構造である。→どういうこと？
- 右図のように、浅い深さの木しか登場しない。



小課題 3 (27 点)

- 小課題 2 とは全く異なる形状となっている。
- 頂点 0 から 8 個以下の中継地しか必要とせずにすべての頂点に移動可能な木構造である。→どういうこと？
- 右図のように、浅い深さの木しか登場しない。
- 浅い木って何か嬉しいの？



小課題 3 (27 点)

- 結論から言うと、**それぞれのノードの深さを決定**することができる。

小課題 3 (27 点)

- 結論から言うと、**それぞれのノードの深さを決定**することができる。
  - やり方：深さ 0 の頂点は頂点 0 のみ。

    深さ $k(k \geqq 0)$ 以下のノード全体からなる集合に対し、それ以外の頂点それぞれについて**直接繋がっているか判定**する。直接繋がっているなら深さ $k+1$ である。

小課題 3 (27 点)

- 結論から言うと、**それぞれのノードの深さを決定**することができる。
  - やり方：深さ 0 の頂点は頂点 0 のみ。
    深さ $k(k \geqq 0)$ 以下のノード全体からなる集合に対し、それ以外の頂点それぞれについて**直接繋がっているか判定**する。直接繋がっているなら深さ $k+1$ である。

これは各頂点に対し、深さ k 以下全体のみを使用可能とした際に 0 とつながっているかで判定

小課題 3 (27 点)

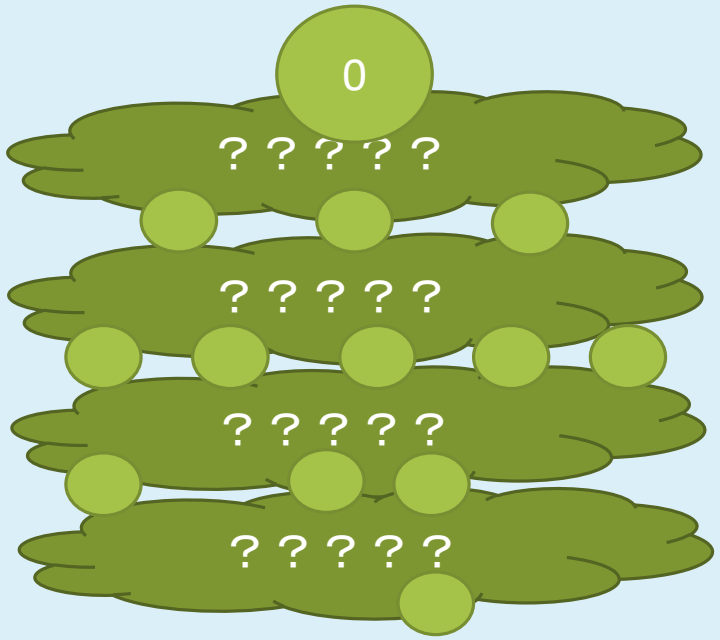
- 結論から言うと、**それぞれのノードの深さを決定**することができる。
  - やり方：深さ 0 の頂点は頂点 0 のみ。
    深さ $k(k \geqq 0)$ 以下のノード全体からなる集合に対し、それ以外の頂点それぞれについて**直接繋がっているか判定**する。直接繋がっているなら深さ $k+1$ である。

これは各頂点に対し、深さ k 以下全体のみを使用可能とした際に 0 とつながっているかで判定

- 各深さごとに最悪 $O(N)$ 回かかってしまうが、小課題 3 の性質より $9N$ 回以内にすべて判定できる。

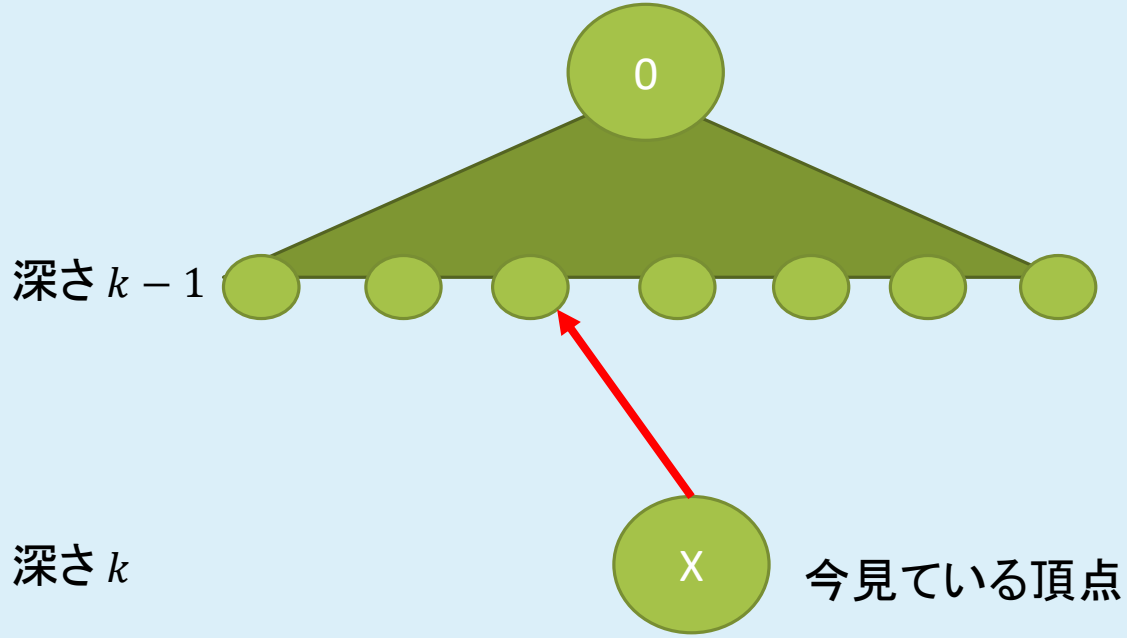
小課題 3 (27 点)

- しかしながら深さが分かっただけでは正解は得られない。辺が分かっていないためである。
- 辺を特定するためにどうすればいいのか考えてみる。



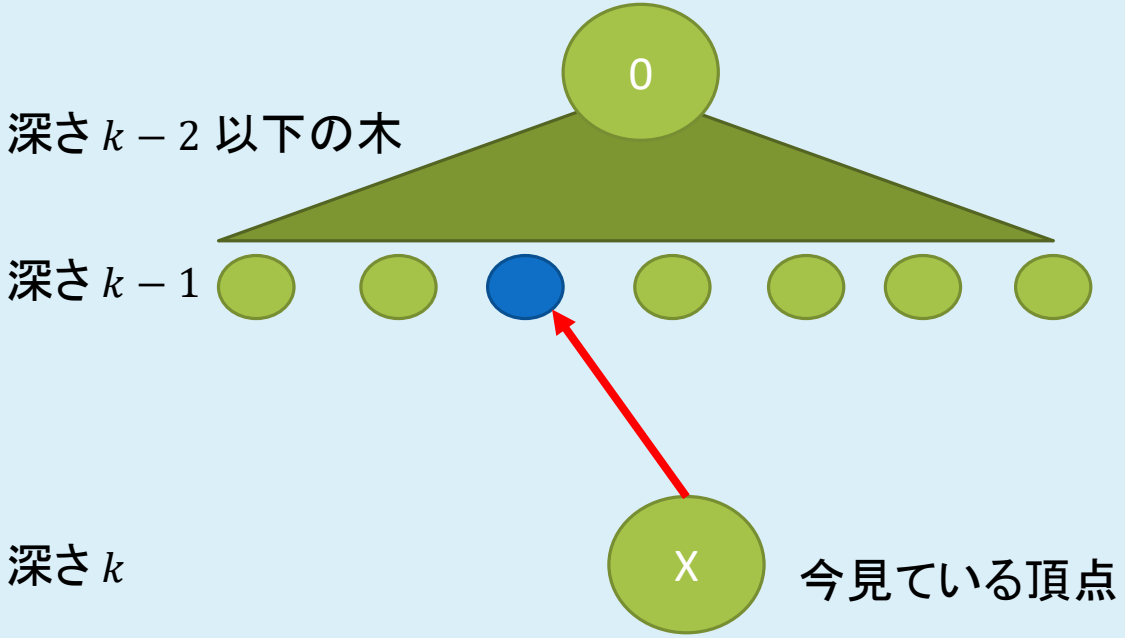
小課題 3 (27 点)

- 分かっていること:
  1. 頂点 0 以外のどの頂点からも、それより深さが 1 低い頂点に辺が出ている。
  2. そのような辺は各頂点ごとに 1 本だけである（そうでないと木にならない）。



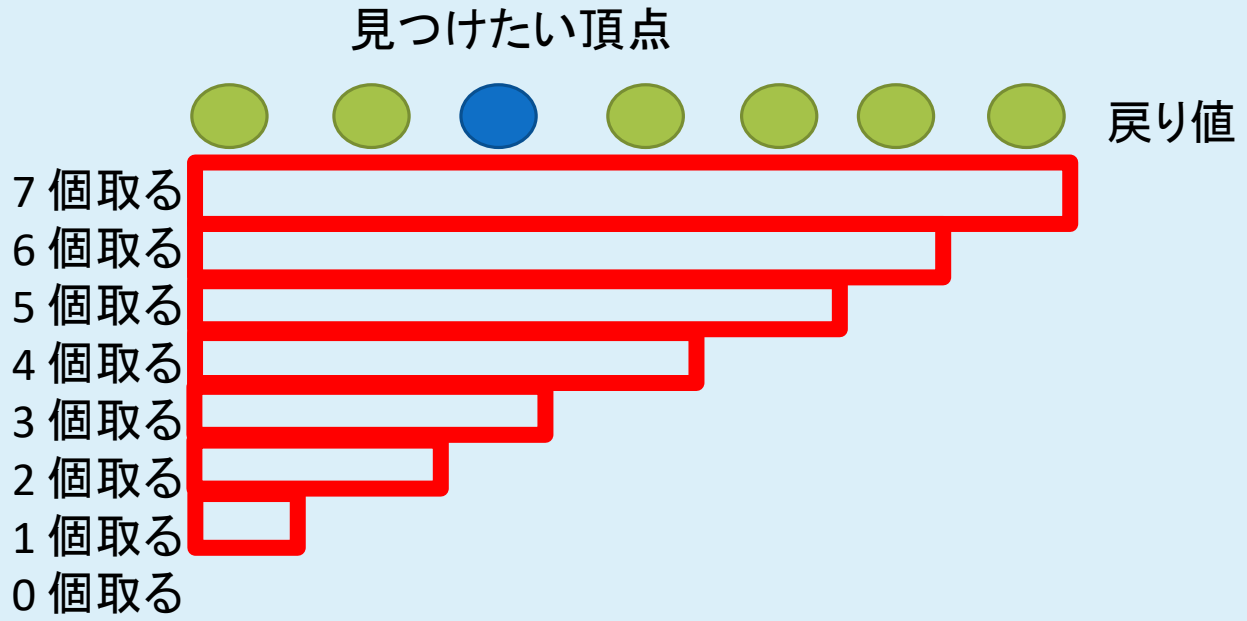
小課題 3 (27 点)

- 分かっていること:
  1. 頂点 0 以外のどの頂点からも、それより深さが 1 低い頂点に辺が出ている。
  2. そのようは辺は各頂点ごとに 1 本だけである（そうでないと木にならない）。
- 深さ $k-2$ 以下の頂点からなる木は必ず採用することにして、追加で深さ $k-1$ の頂点集合をどのように選択すると頂点 0 と X が連結になる/ならないか考えてみる。
- 他の深さ k-1 の頂点集合の選び方によらず、X の親となる頂点を頂点を選べば 1、そうでなければ 0 が返ってくる。



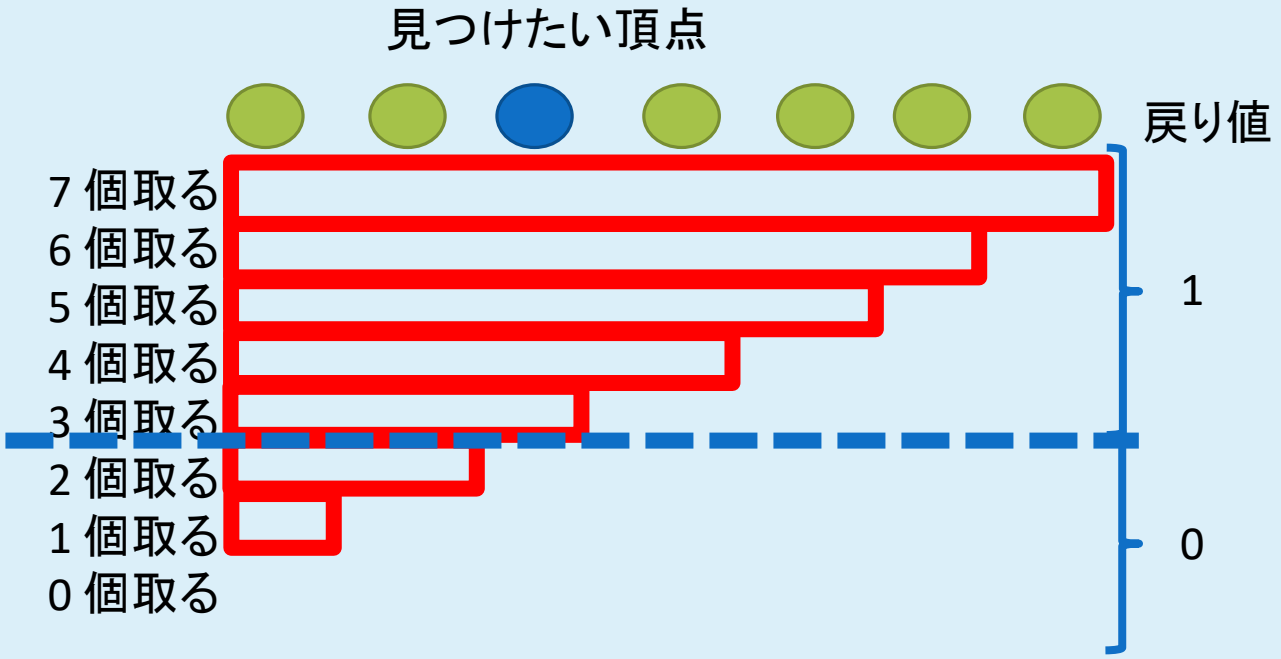
小課題 3 (27 点)

- 適当に左から右に並べて、左から連続していくつかとる方針で考えてみる。



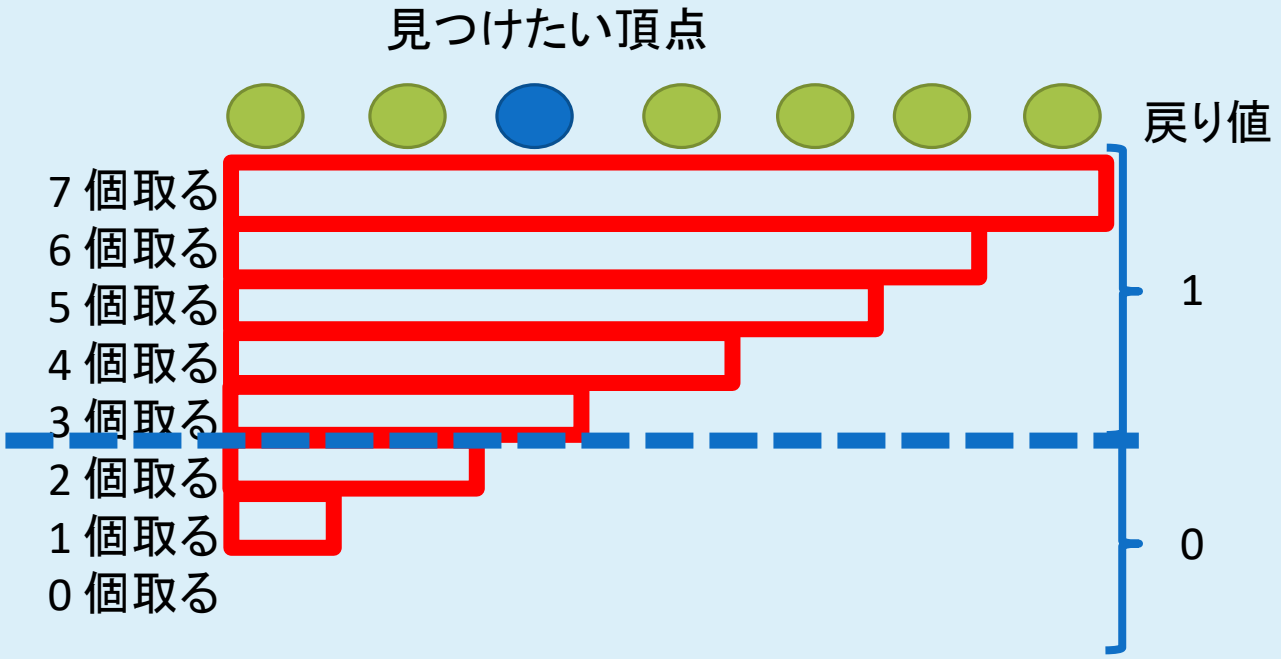
小課題 3 (27 点)

- 適当に左から右に並べて、左から連続していくつかとる方針で考えてみる。
- 選んだ頂点を**境目にして 0/1 が二分**される。



小課題 3 (27 点)

- 適当に左から右に並べて、左から連続していくつかとる方針で考えてみる。
- 選んだ頂点を**境目にして 0/1 が二分**される。
- **二分探索**を用いればどの辺も $\log N$ ステップで発見することができる。



小課題 3 (27 点) 解法

- $9N$ 回調べて各頂点の深さを調べる。
- 各頂点について、**二分探索**を用いれば親ノードを特定でき、結果すべての辺を発見できる。

- 全体で $N(\log N + 9)$ 回の質問回数、$N = 1{,}400$ とすれば $28{,}000$ 回くらいの質問で解くことができる。

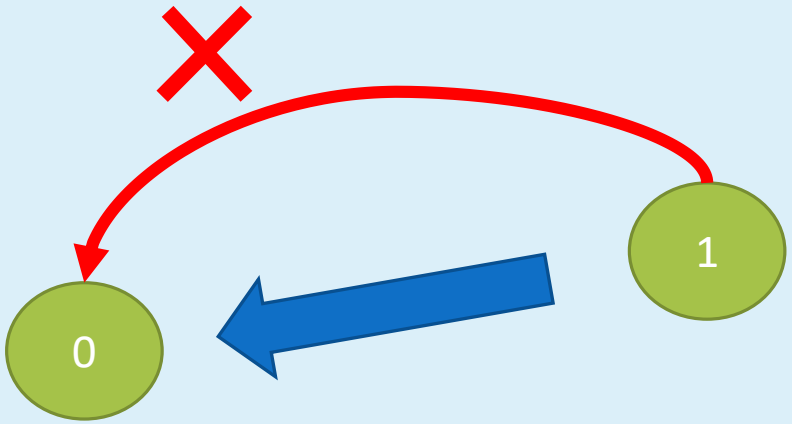
小課題 4 (30 点)

- ここから難易度が跳ね上がる。
- 木構造のすべてが登場する。
  - 当然、小課題 2 や 3 の方法では対処しきれないケースも出てくる。

- しかしながら、うまく閃けば小課題 2,3 をすっとばして一気に 3 種類解くことができる！

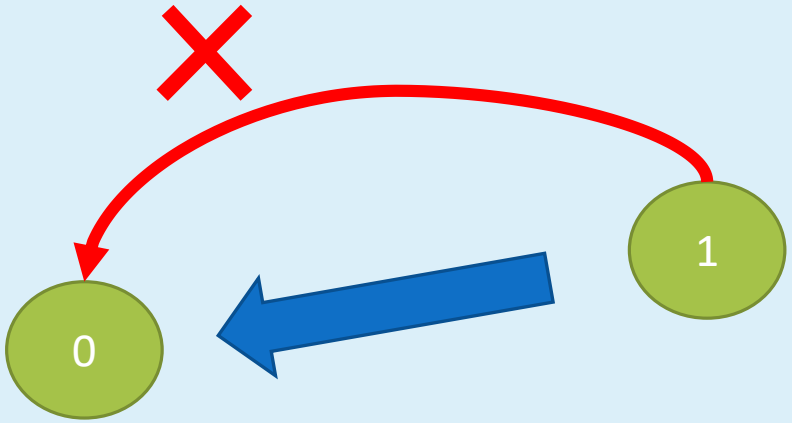
小課題 4 (30 点)

- 頂点 0 から拡張していく方針を考える。
- 例えば頂点 1 を含めよう！
  - うまく直接つながっていれば最高だが、現実はそう甘くはない。



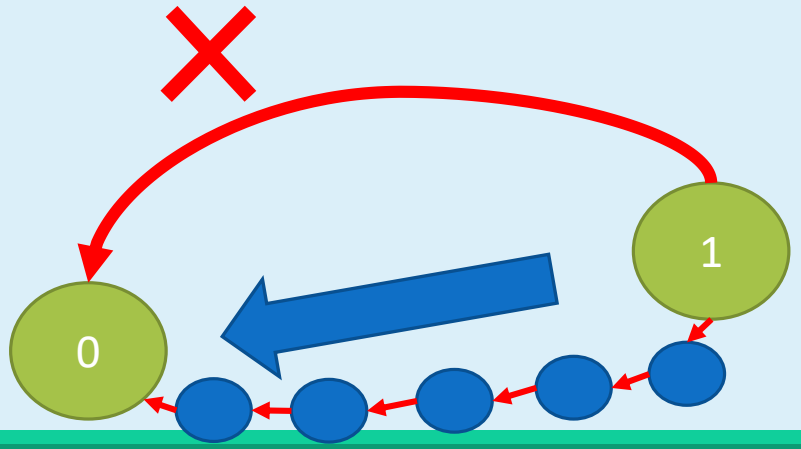
小課題 4 (30 点)

- 頂点 0 から拡張していく方針を考える。
- 例えば頂点 1 を含めよう！
  - うまく直接つながっていれば最高だが、現実はそう甘くはない。
  - こういった場合、残念ながら頂点 1 を**直ちに繋ぐことは厳しい**。



小課題 4 (30 点)

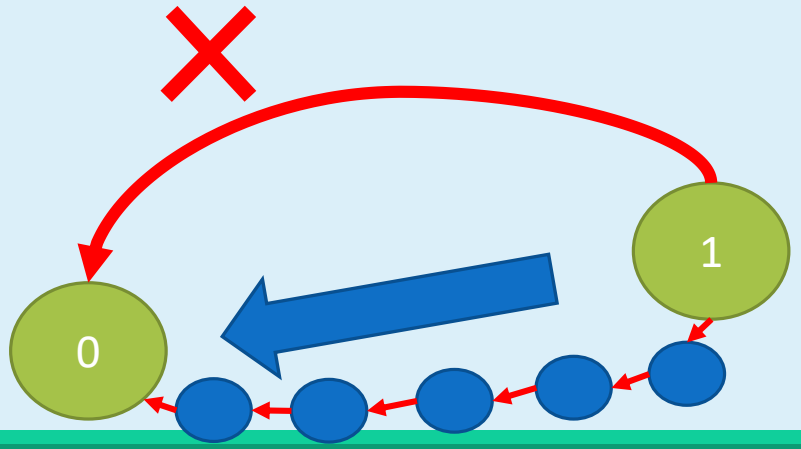
- 頂点 0 から拡張していく方針を考える。
- 例えば頂点 1 を含めよう！
  - うまく直接つながっていれば最高だが、現実はそう甘くはない。
  - こういった場合、残念ながら頂点 1 を**直ちに繋ぐことは厳しい**。
  - 頂点 1 がよくなかった理由は、頂点 1 より近い別の頂点が間に挟まっているからである。



小課題 4 (30 点)

- 頂点 0 から拡張していく方針を考える。
- 例えば頂点 1 を含めよう！
  - うまく直接つながっていれば最高だが、現実はそう甘くはない。
  - こういった場合、残念ながら頂点 1 を**直ちに繋ぐことは厳しい**。
  - 頂点 1 がよくなかった理由は、頂点 1 より近い別の頂点が間に挟まっているからである。

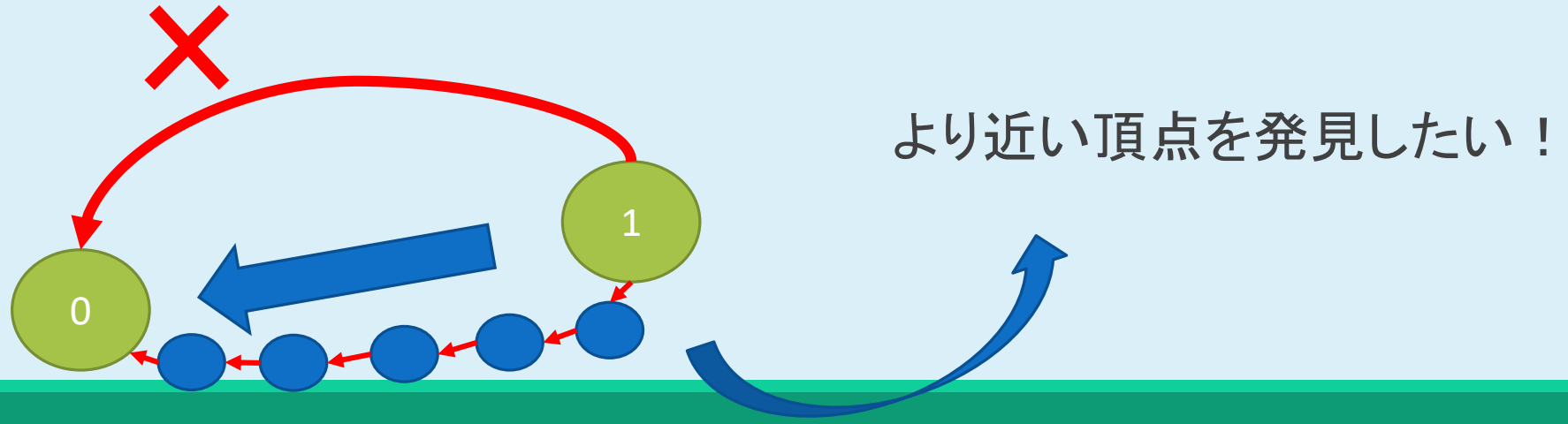
→ その頂点は直接頂点 0 とは繋がっているかもしれない！



小課題 4 (30 点)

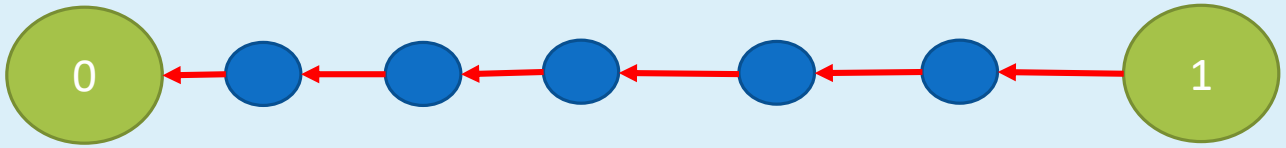
- 頂点 0 から拡張していく方針を考える。
- 例えば頂点 1 を含めよう！
  - うまく直接つながっていれば最高だが、現実はそう甘くはない。
  - こういった場合、残念ながら頂点 1 を**直ちに繋ぐことは厳しい**。
  - 頂点 1 がよくなかった理由は、頂点 1 より近い別の頂点が間に挟まっているからである。

→ その頂点は直接頂点 0 とは繋がっているかもしれない！



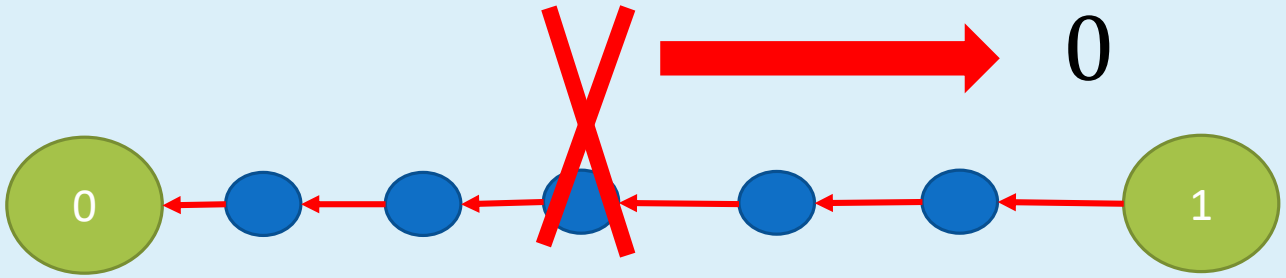
小課題 4 (30 点)

- 頂点 0 と 1 の移動可能性は何に影響されるか考えてみる。



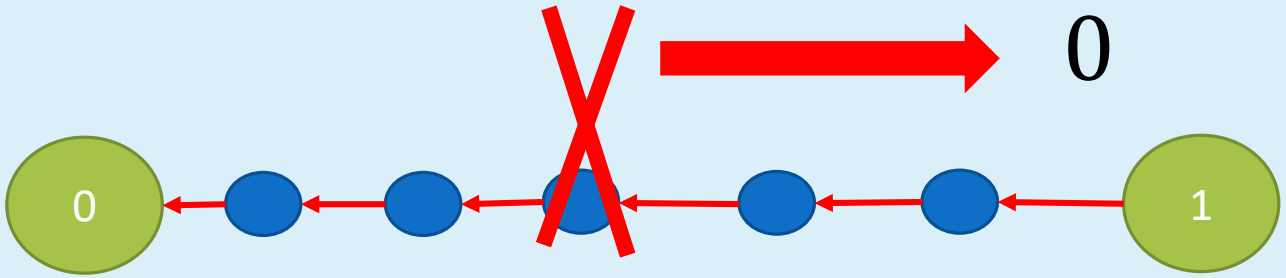
小課題 4 (30 点)

- 頂点 0 と 1 の移動可能性は何に影響されるか考えてみる。
- 中間にあるノードが消滅した瞬間、移動不可能になる（木なので（点素）パスは 1 通りしかないため）



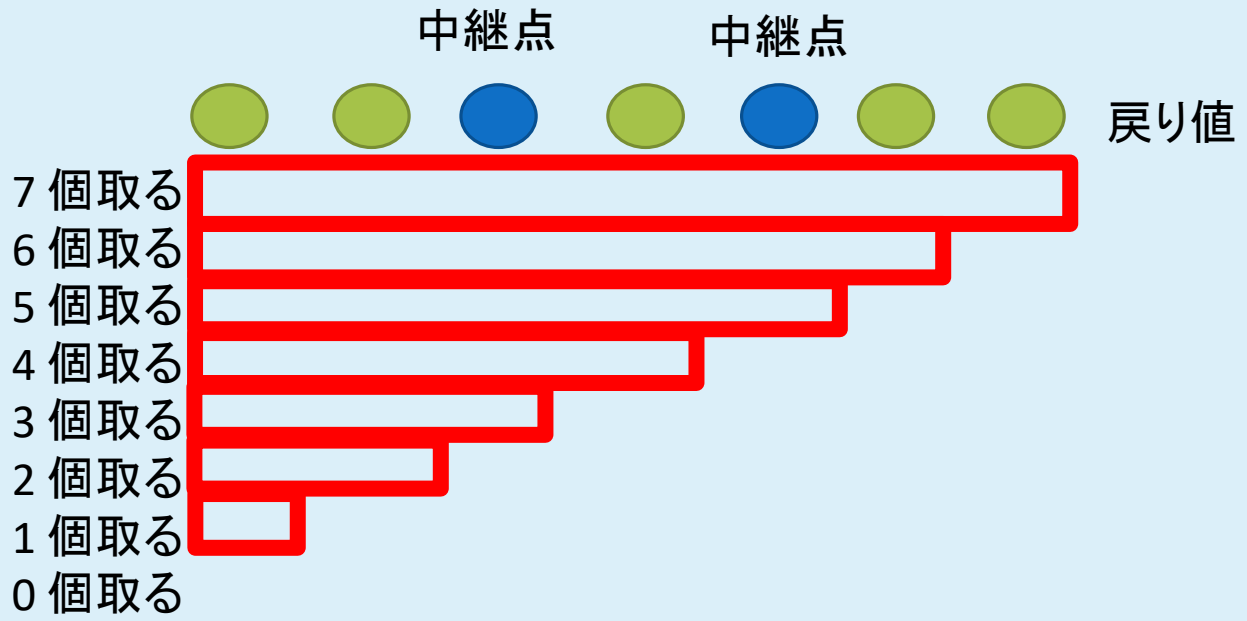
小課題 4 (30 点)

- 頂点 0 と 1 の移動可能性は何に影響されるか考えてみる。
- 中間にあるノードが消滅した瞬間、移動不可能になる（木なので（点素）パスは 1 通りしかないため）
- 実は、小課題 3 のような**二分探索**が役に立ちます。



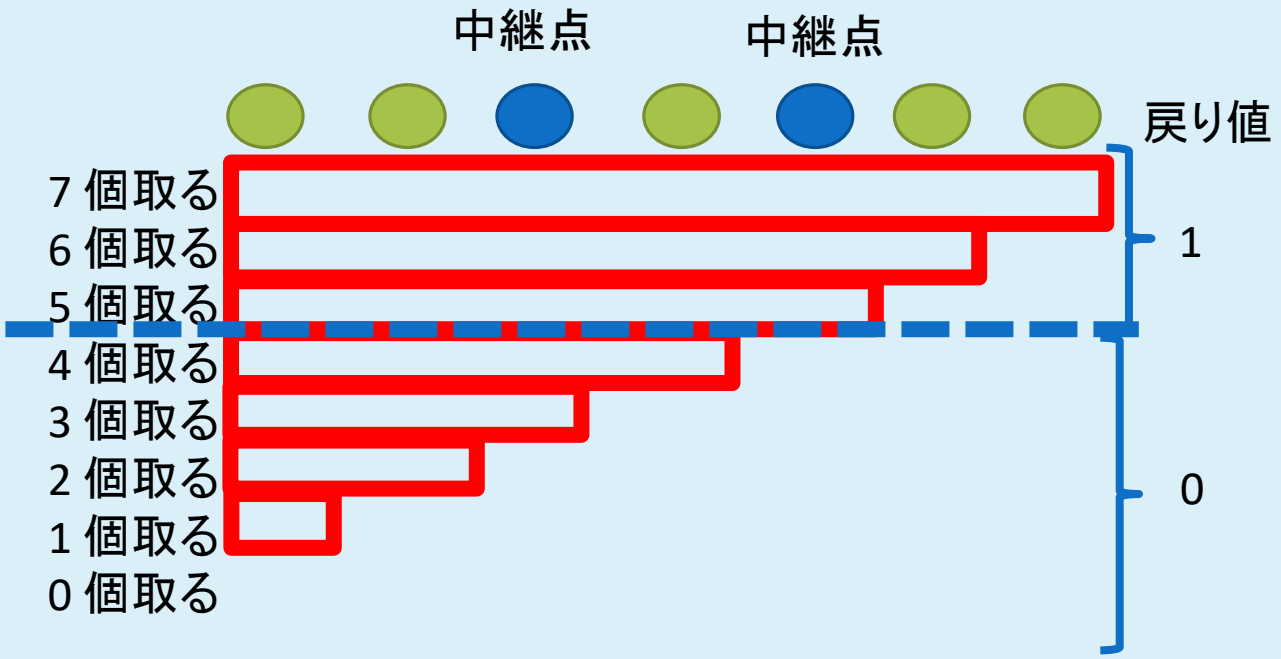
小課題 4 (30 点)

- 適当に左から右に並べて、左から連続していくつかとる方針で考えてみる。



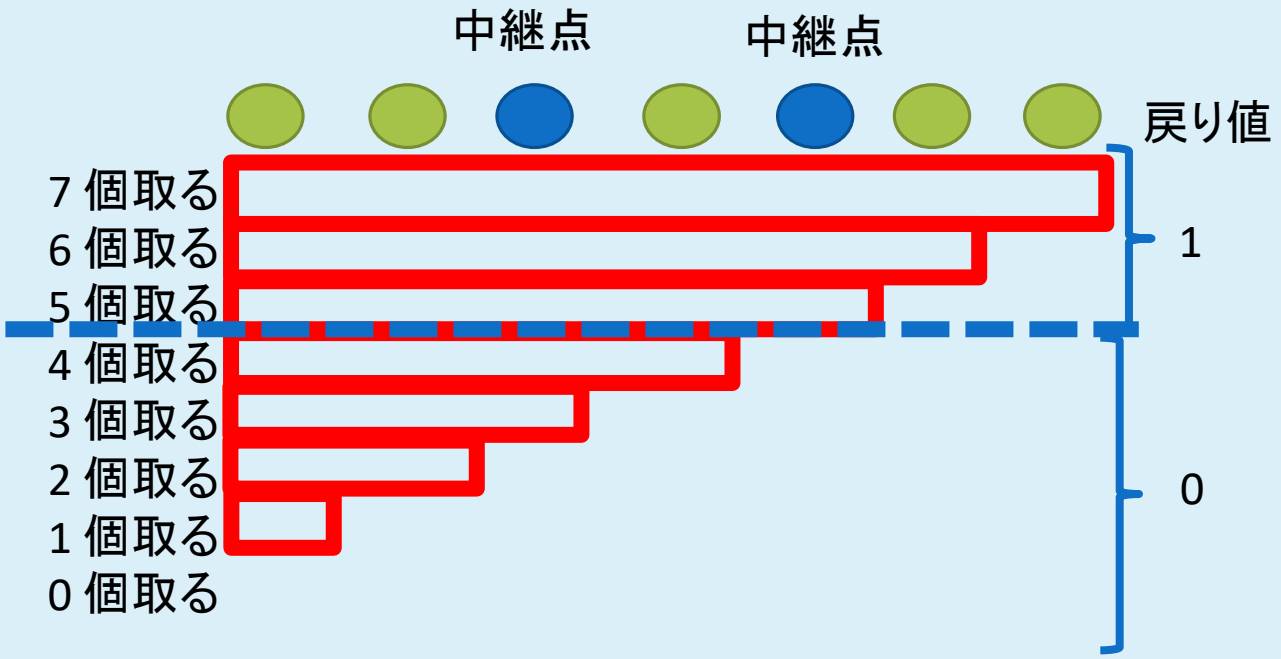
小課題 4 (30 点)

- 適当に左から右に並べて、左から連続していくつかとる方針で考えてみる。
- 中継点のうち**最も右にあるものを境目にして 0/1 が二分**される。



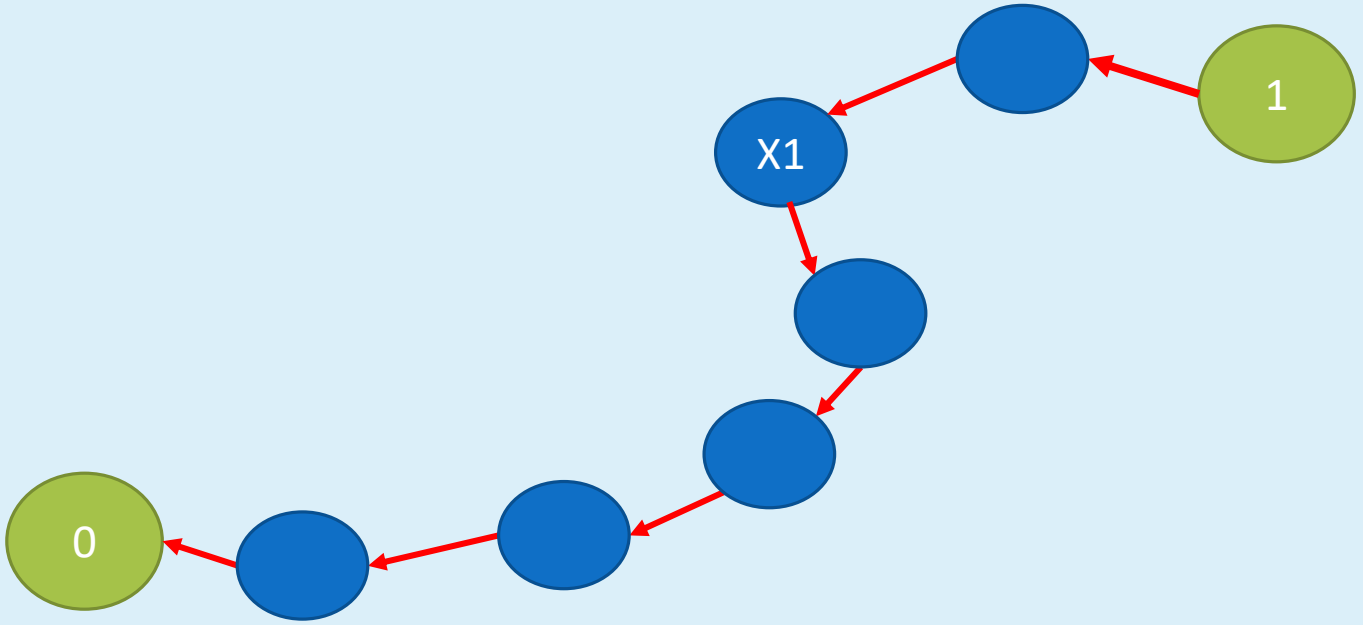
小課題 4 (30 点)

- 適当に左から右に並べて、左から連続していくつかとる方針で考えてみる。
- 中継点のうち**最も右にあるものを境目にして 0/1 が二分**される。
- **二分探索**を用いればいずれかの中継点を $\log N$ ステップで発見することができる！



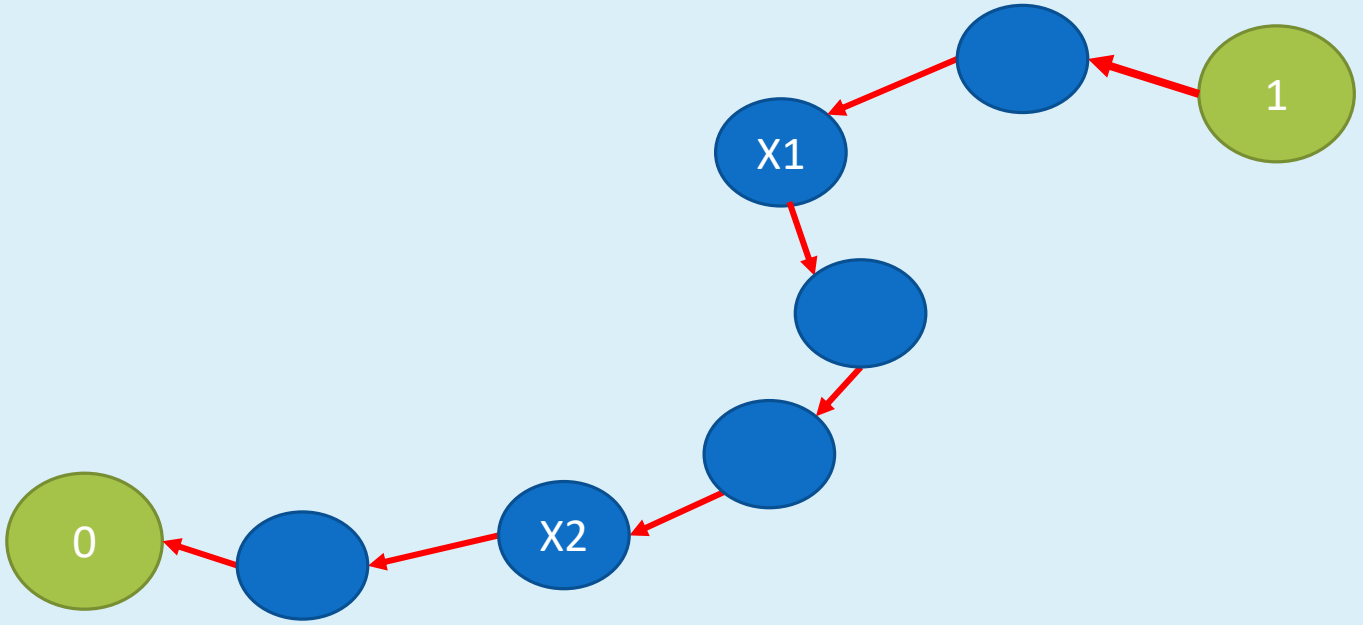
小課題 4 (30 点)

- とりあえず中継点（X1 とおく）が決まった。この後どうしよう？
- 残念ながら直接つながるのはやはり稀。



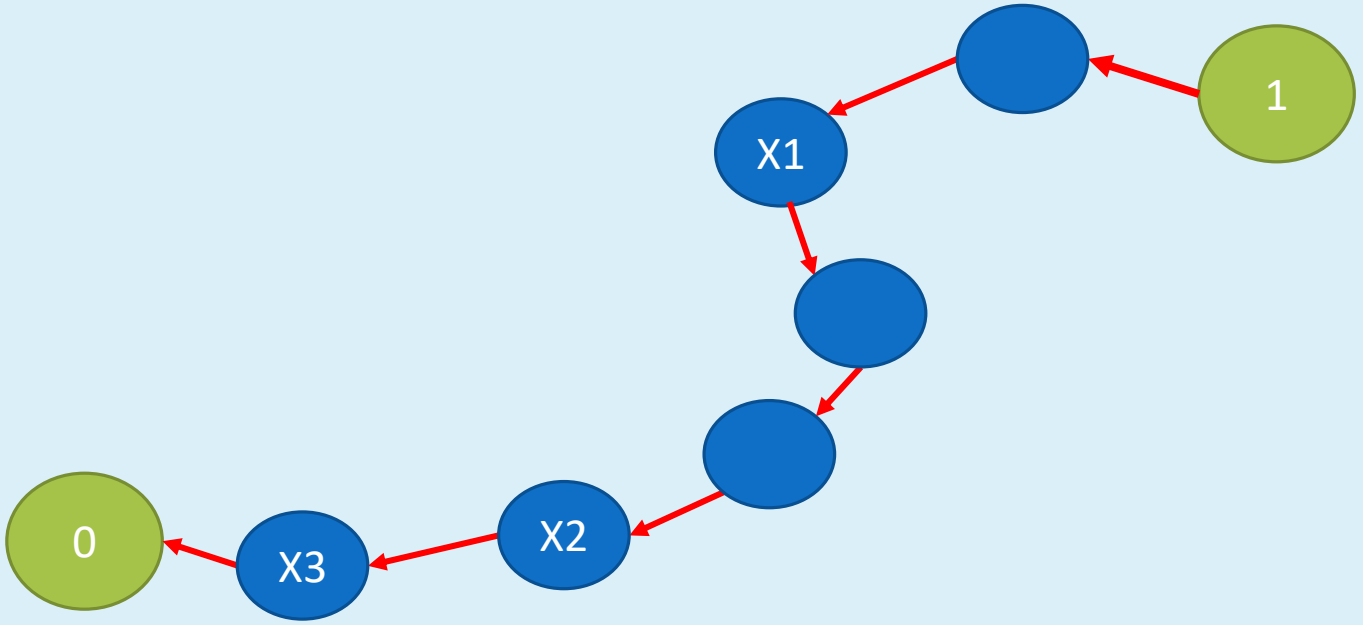
小課題 4 (30 点)

- とりあえず中継点（X1 とおく）が決まった。この後どうしよう？
- 残念ながら直接つながるのはやはり稀。
- じゃあ X1 を頂点 1 と同様に扱うと？→より近い頂点 X2 が出る。



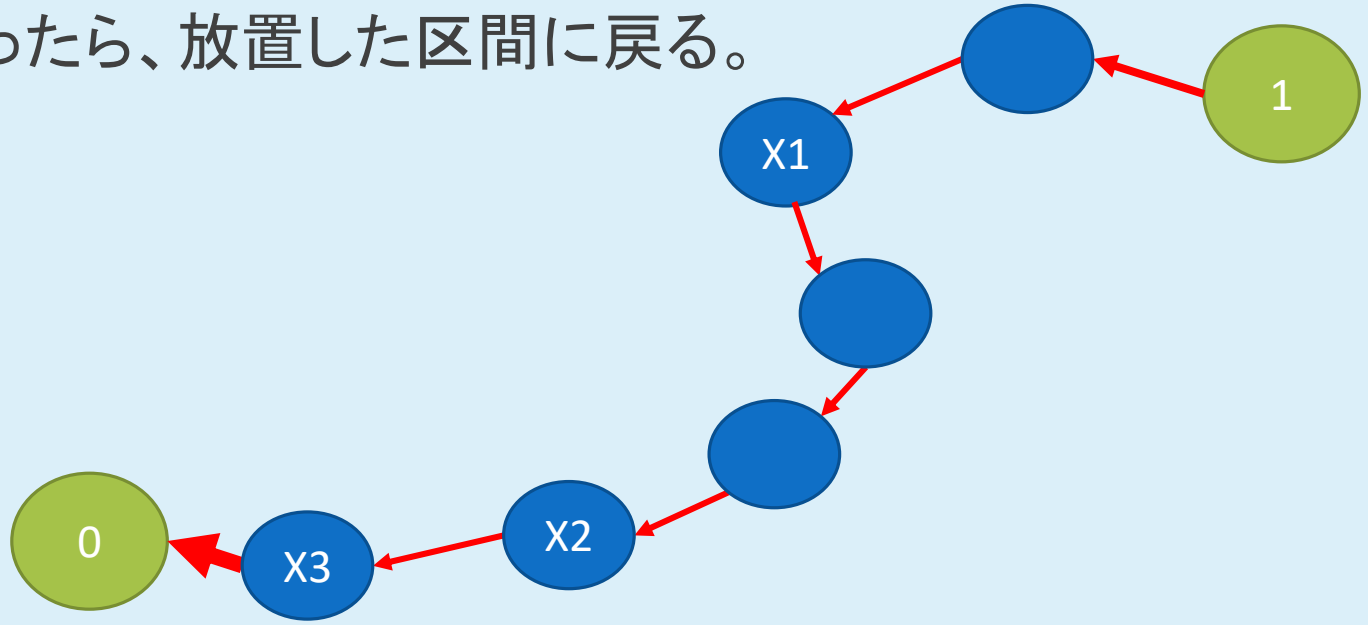
小課題 4 (30 点)

- とりあえず中継点（X1 とおく）が決まった。この後どうしよう？
- 残念ながら直接つながるのはやはり稀。
- じゃあ X1 を頂点 1 と同様に扱うと？→より近い頂点 X2 が出る。
- それでもだめなら繰り返す（X3→…）



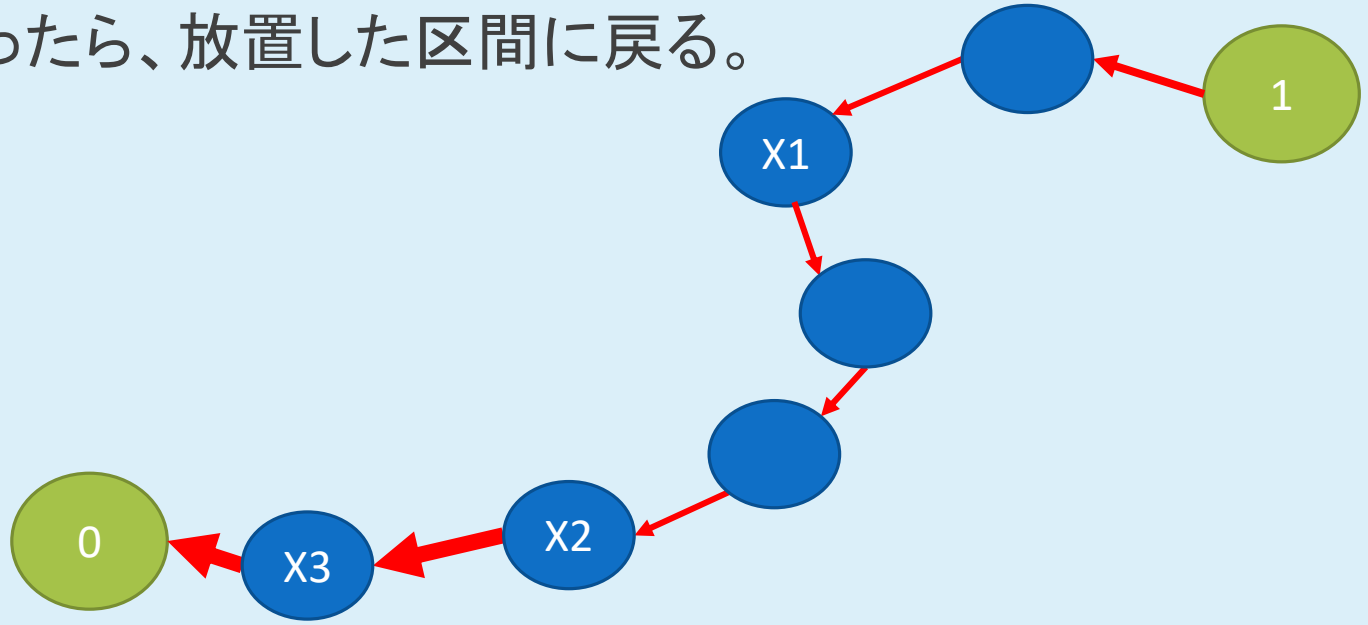
小課題 4 (30 点)

- とりあえず中継点（X1 とおく）が決まった。この後どうしよう？
- 残念ながら直接つながるのはやはり稀。
- じゃあ X1 を頂点 1 と同様に扱うと？→より近い頂点 X2 が出る。
- それでもだめなら繰り返す（X3→…）
- 繋がったら、放置した区間に戻る。



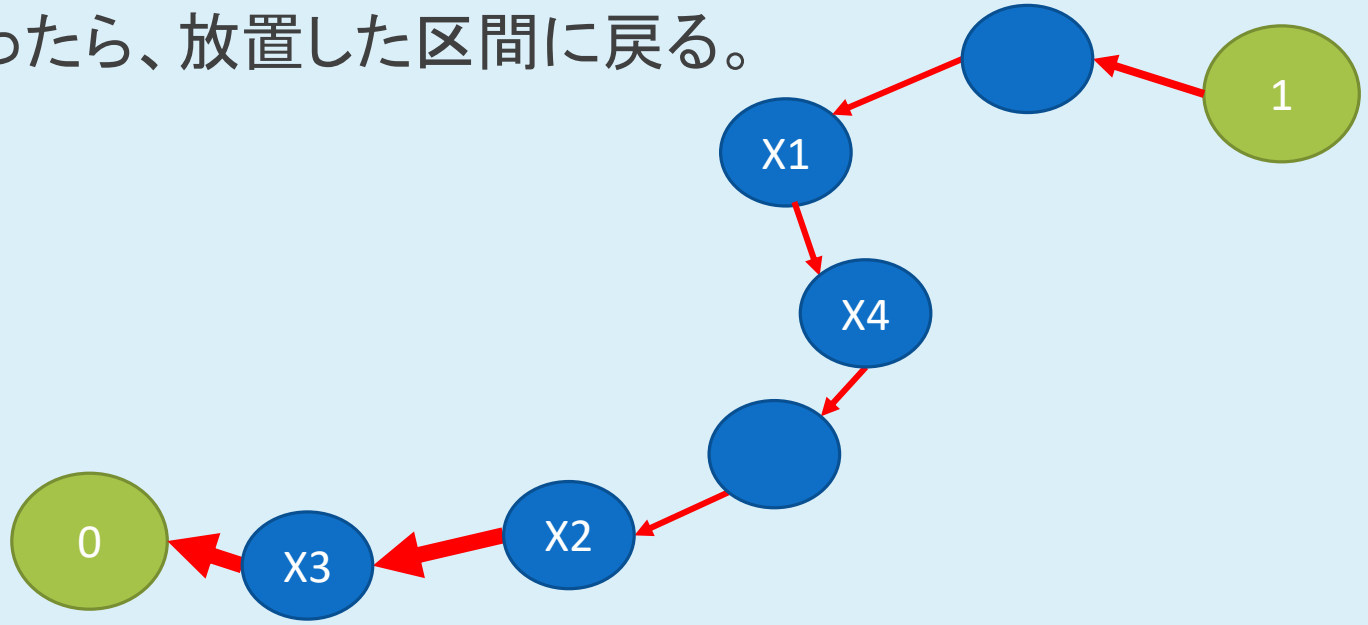
小課題 4 (30 点)

- とりあえず中継点（X1 とおく）が決まった。この後どうしよう？
- 残念ながら直接つながるのはやはり稀。
- じゃあ X1 を頂点 1 と同様に扱うと？→より近い頂点 X2 が出る。
- それでもだめなら繰り返す（X3→…）
- 繋がったら、放置した区間に戻る。



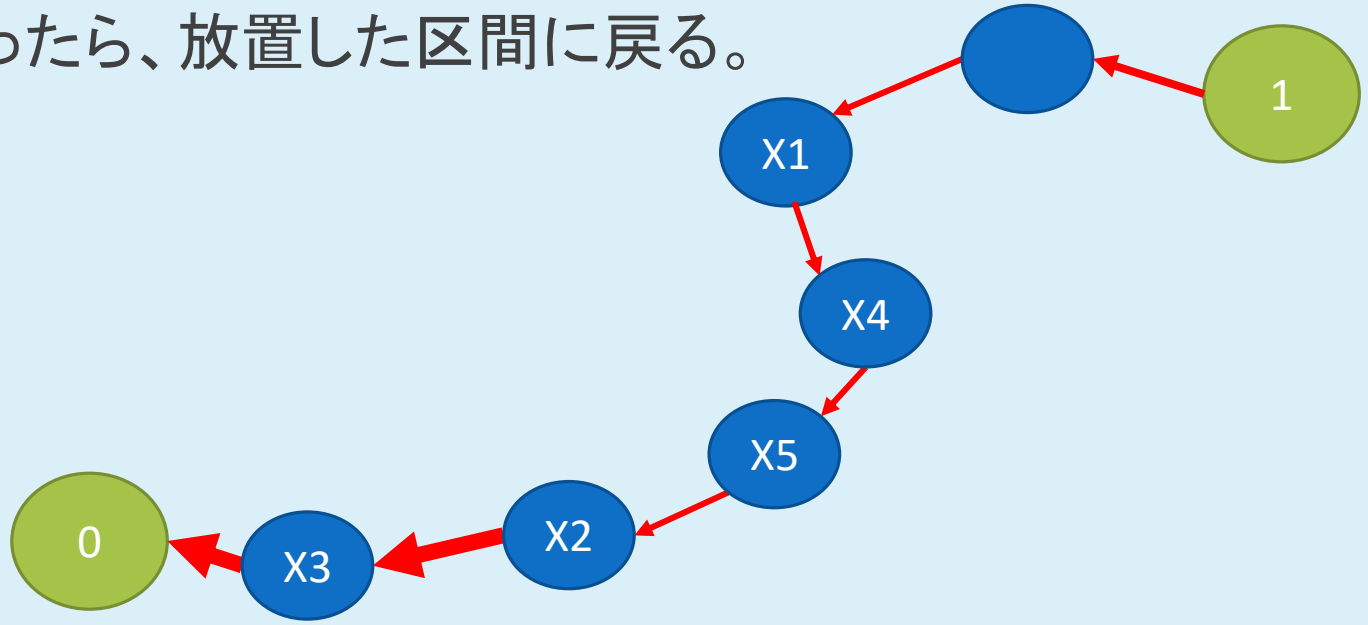
小課題 4 (30 点)

- とりあえず中継点（X1 とおく）が決まった。この後どうしよう？
- 残念ながら直接つながるのはやはり稀。
- じゃあ X1 を頂点 1 と同様に扱うと？→より近い頂点 X2 が出る。
- それでもだめなら繰り返す（X3→…）
- 繋がったら、放置した区間に戻る。



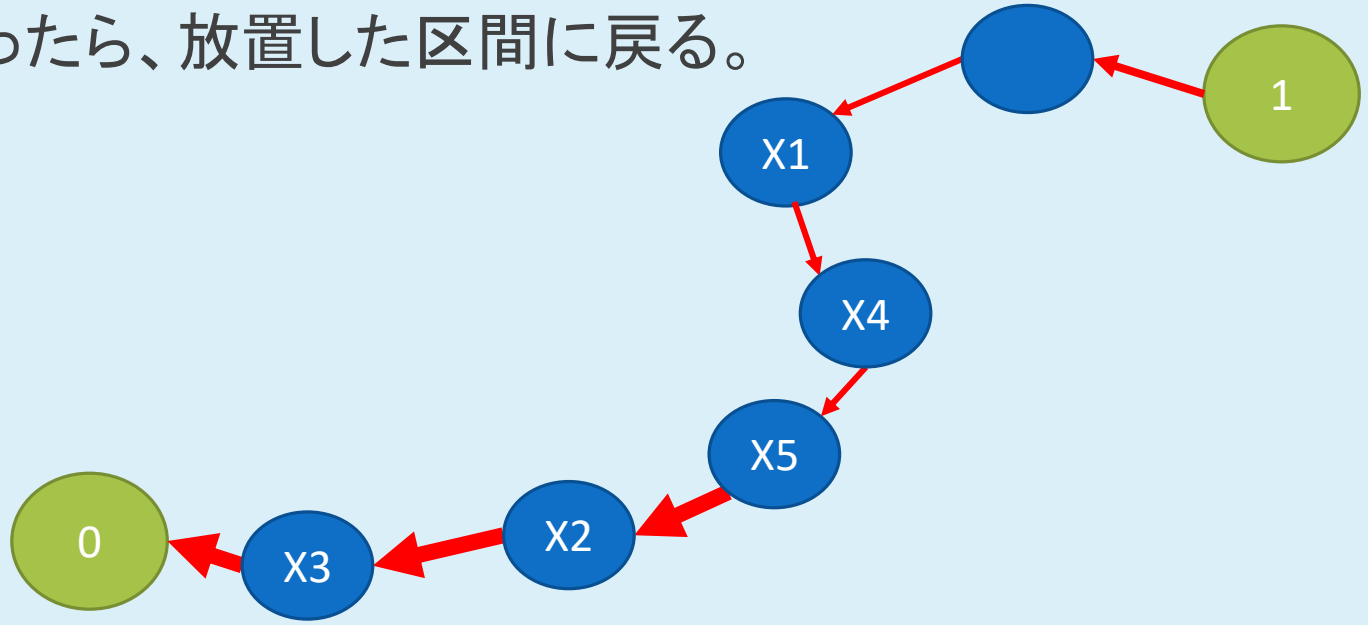
小課題 4 (30 点)

- とりあえず中継点（X1 とおく）が決まった。この後どうしよう？
- 残念ながら直接つながるのははやはり稀。
- じゃあ X1 を頂点 1 と同様に扱うと？→より近い頂点 X2 が出る。
- それでもだめなら繰り返す（X3→…）
- 繋がったら、放置した区間に戻る。



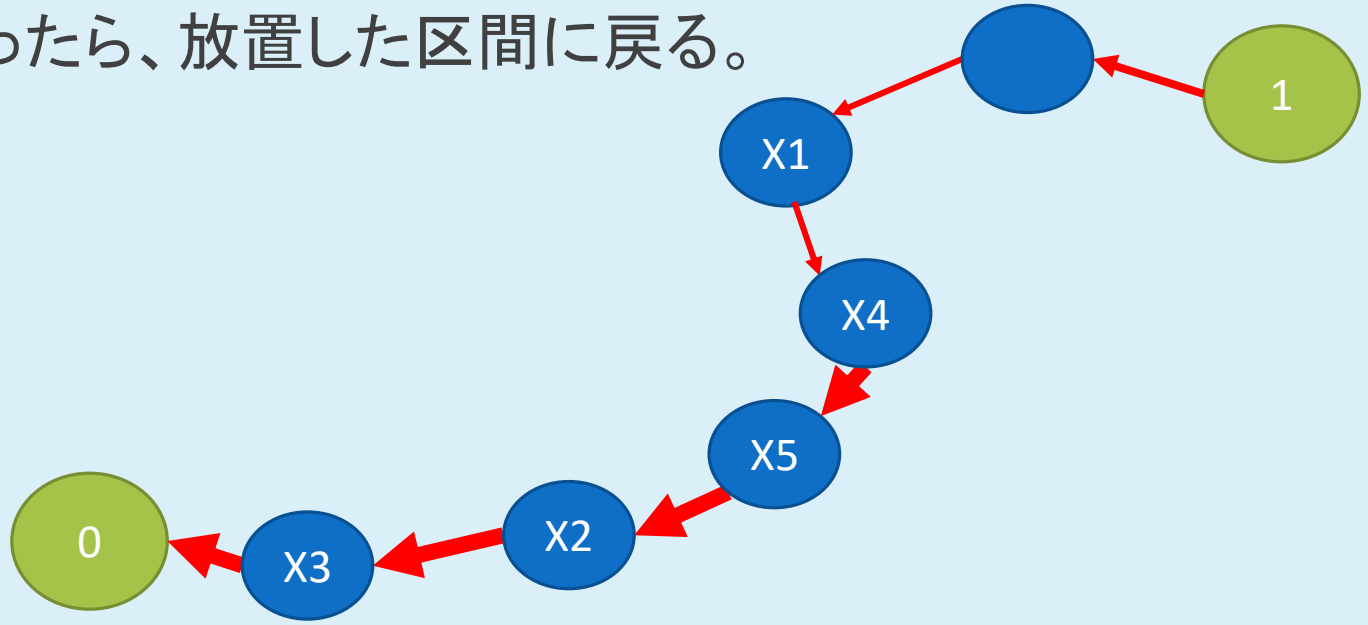
小課題 4 (30 点)

- とりあえず中継点（X1 とおく）が決まった。この後どうしよう？
- 残念ながら直接つながるのははやはり稀。
- じゃあ X1 を頂点 1 と同様に扱うと？→より近い頂点 X2 が出る。
- それでもだめなら繰り返す（X3→…）
- 繋がったら、放置した区間に戻る。



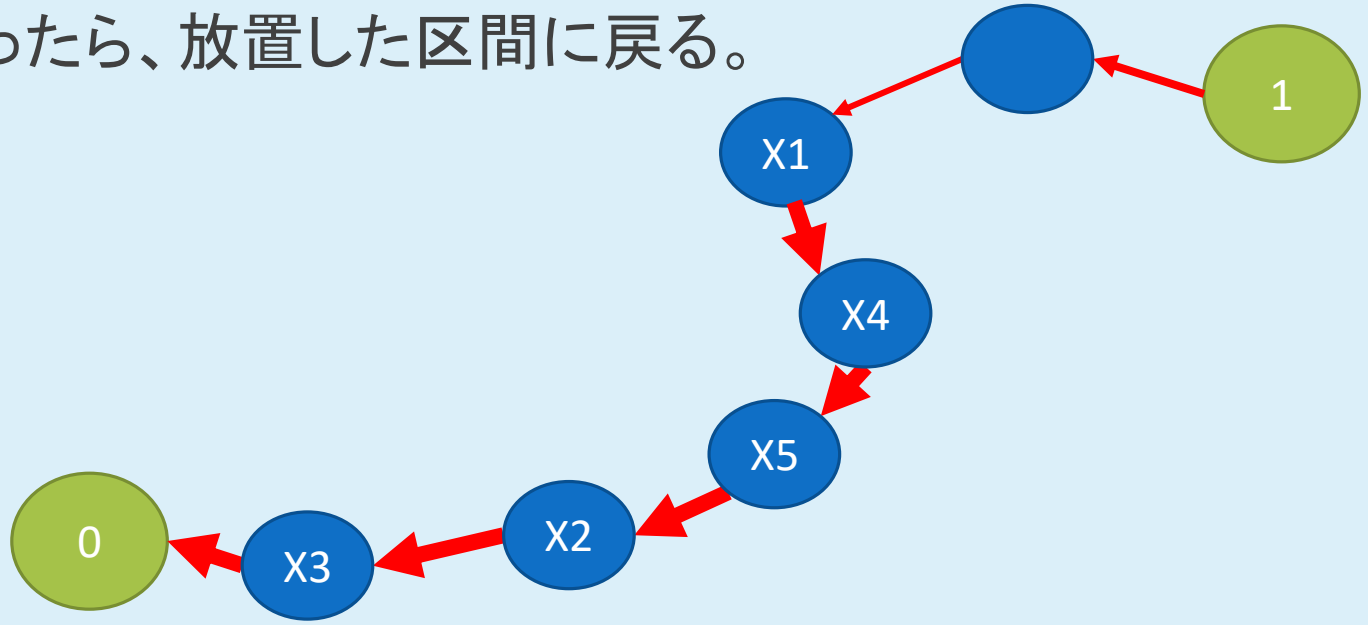
小課題 4 (30 点)

- とりあえず中継点（X1 とおく）が決まった。この後どうしよう？
- 残念ながら直接つながるのはやはり稀。
- じゃあ X1 を頂点 1 と同様に扱うと？→より近い頂点 X2 が出る。
- それでもだめなら繰り返す（X3→…）
- 繋がったら、放置した区間に戻る。



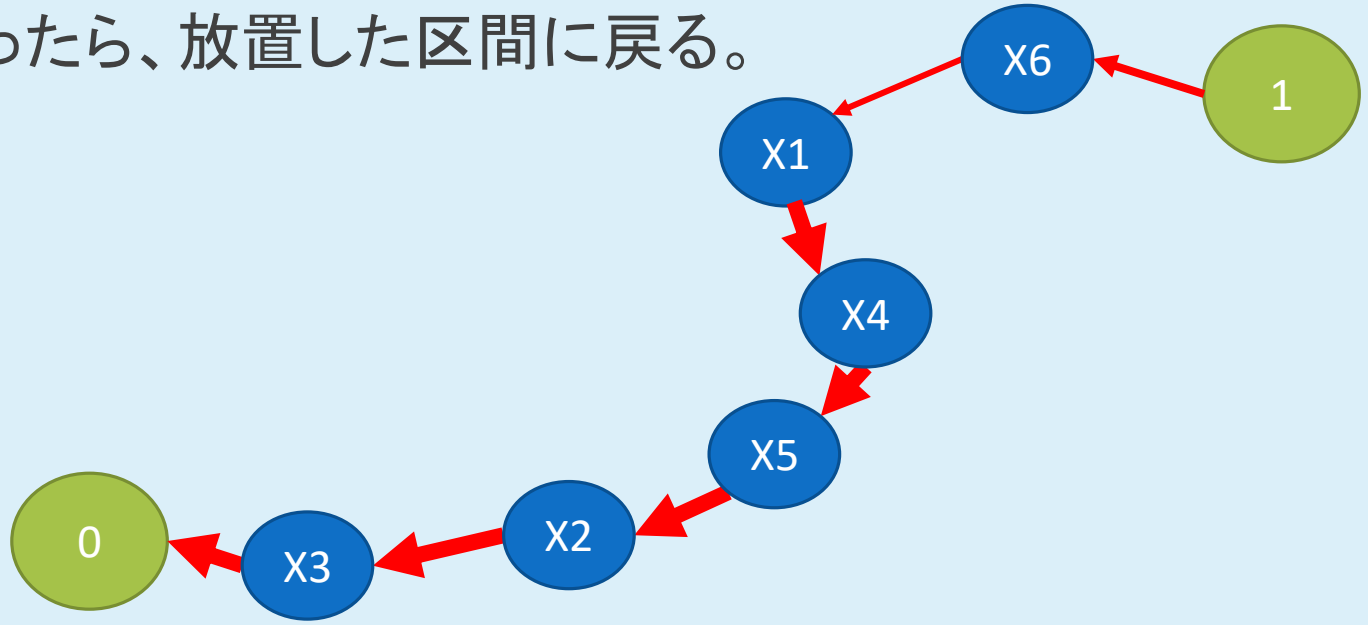
小課題 4 (30 点)

- とりあえず中継点（X1 とおく）が決まった。この後どうしよう？
- 残念ながら直接つながるのはやはり稀。
- じゃあ X1 を頂点 1 と同様に扱うと？→より近い頂点 X2 が出る。
- それでもだめなら繰り返す（X3→…）
- 繋がったら、放置した区間に戻る。



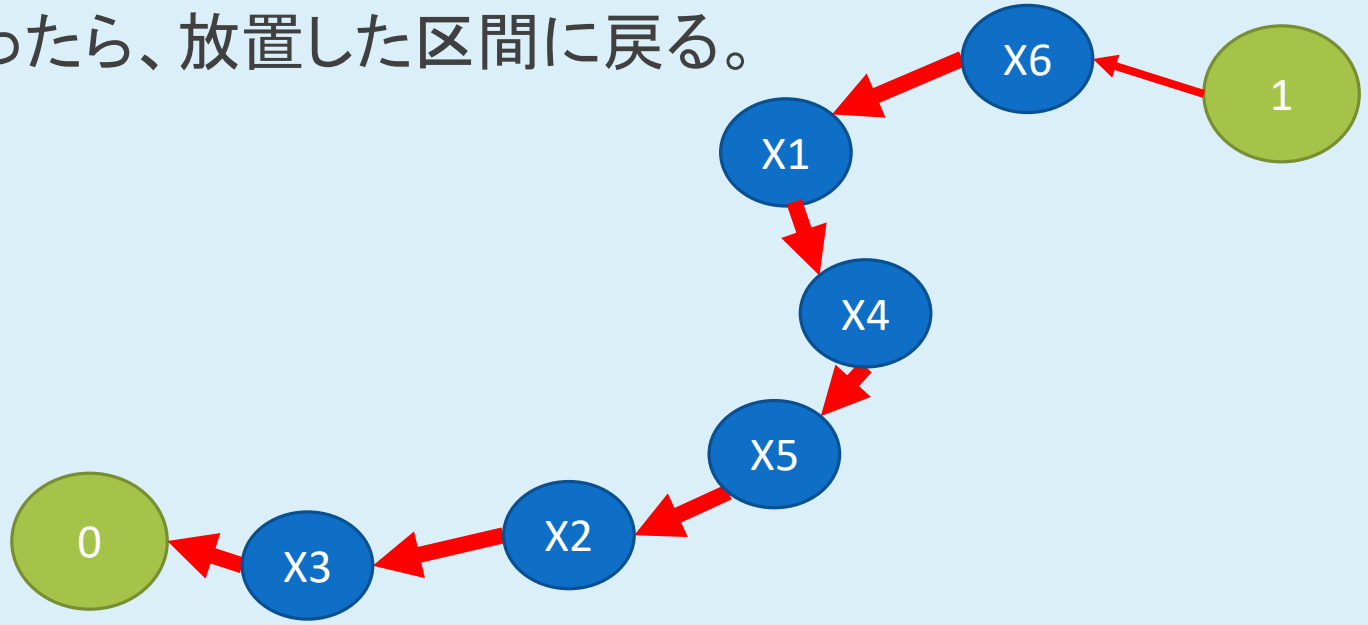
小課題 4 (30 点)

- とりあえず中継点（X1 とおく）が決まった。この後どうしよう？
- 残念ながら直接つながるのはやはり稀。
- じゃあ X1 を頂点 1 と同様に扱うと？→より近い頂点 X2 が出る。
- それでもだめなら繰り返す（X3→…）
- 繋がったら、放置した区間に戻る。



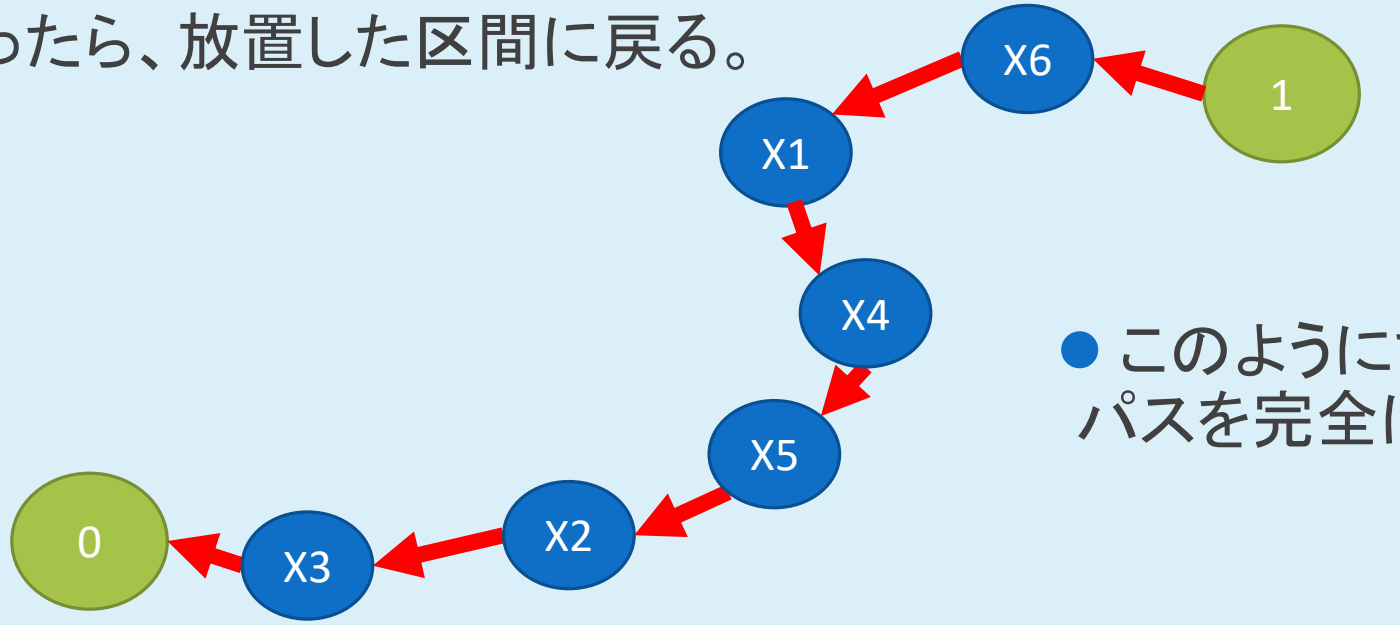
小課題 4 (30 点)

- とりあえず中継点（X1 とおく）が決まった。この後どうしよう？
- 残念ながら直接つながるのはやはり稀。
- じゃあ X1 を頂点 1 と同様に扱うと？→より近い頂点 X2 が出る。
- それでもだめなら繰り返す（X3→…）
- 繋がったら、放置した区間に戻る。



小課題 4 (30 点)

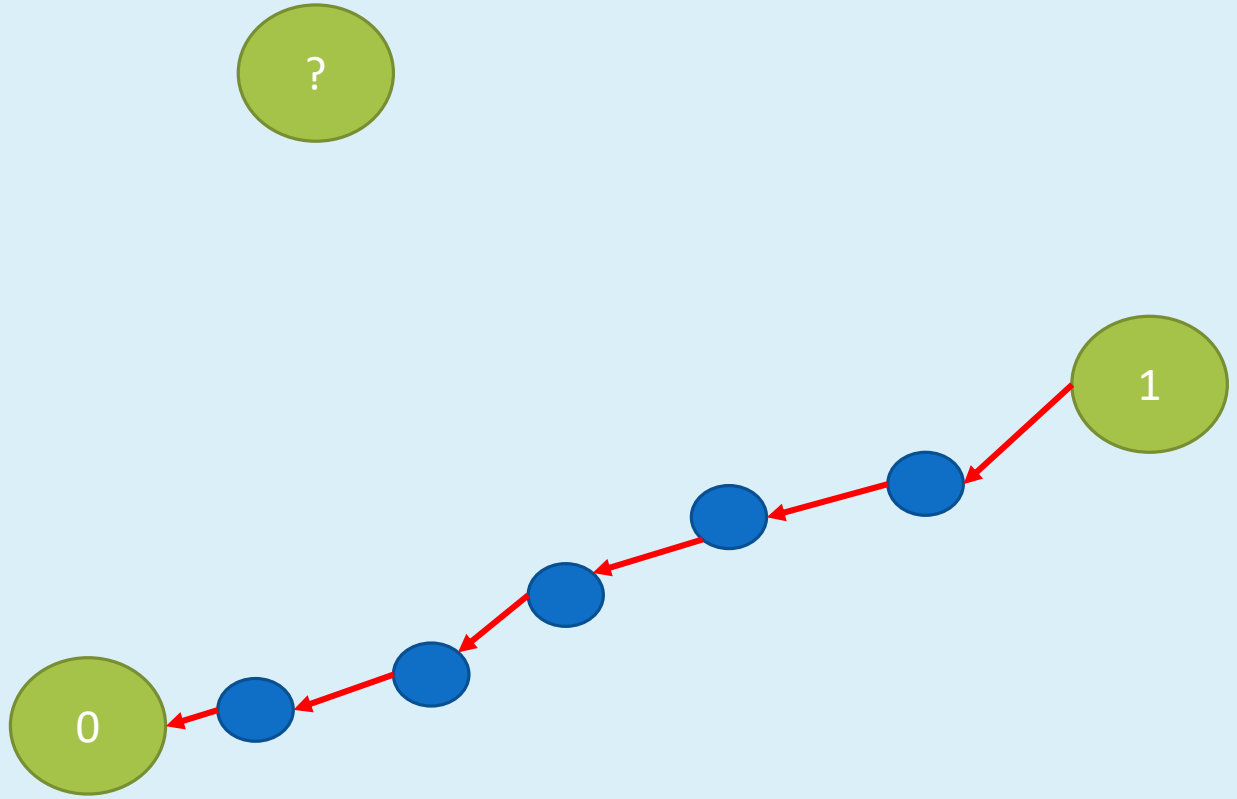
- とりあえず中継点（X1 とおく）が決まった。この後どうしよう？
- 残念ながら直接つながるのはやはり稀。
- じゃあ X1 を頂点 1 と同様に扱うと？→より近い頂点 X2 が出る。
- それでもだめなら繰り返す（X3→…）
- 繋がったら、放置した区間に戻る。



- このようにすれば、頂点 0 と 1 をつなぐパスを完全に特定できる！

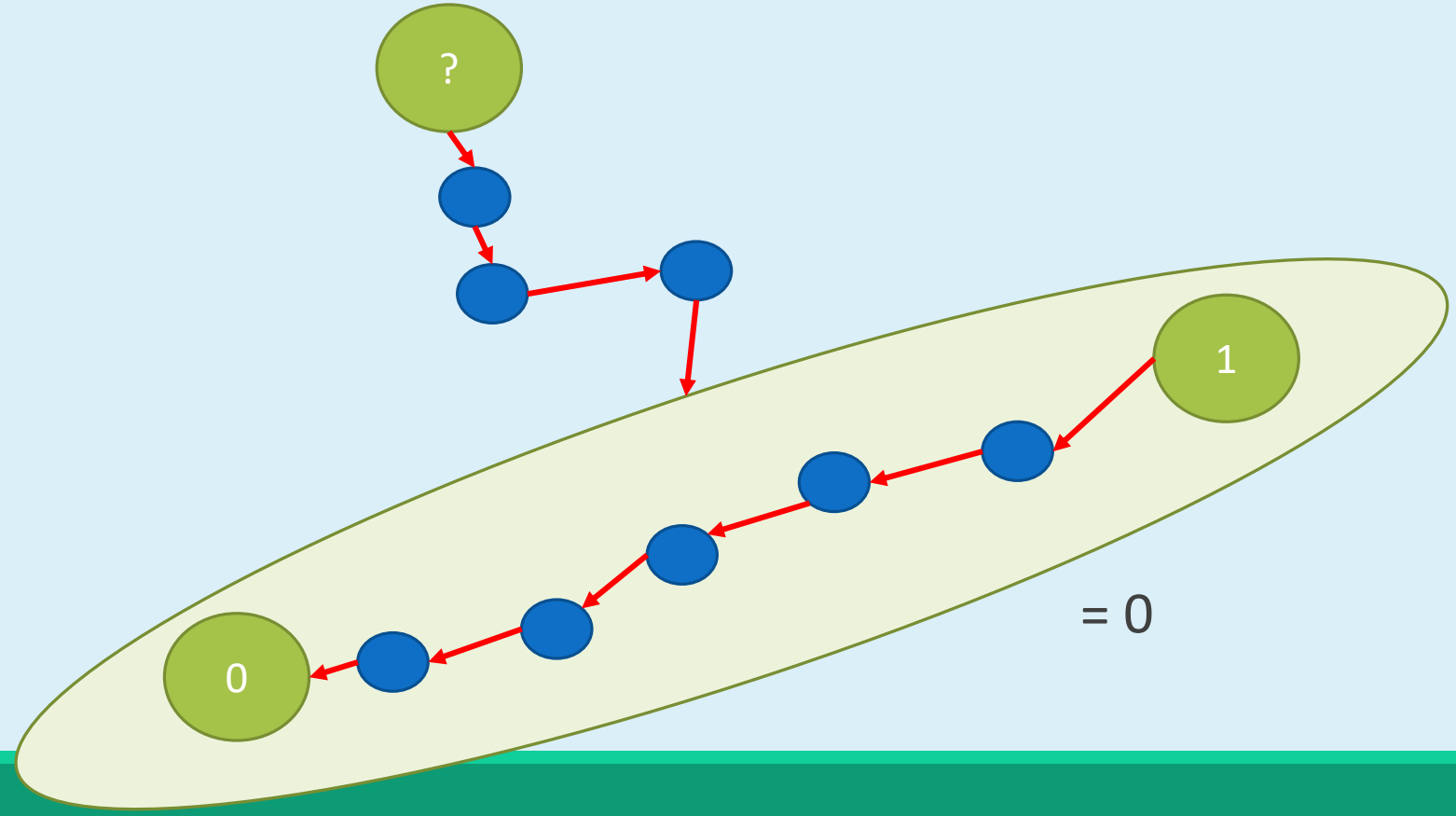
小課題 4 (30 点)

- それが終わった後は？



小課題 4 (30 点)

- それが終わった後は？
- わかっている部分全体を頂点 0 と同じに扱えば、同様にできる。

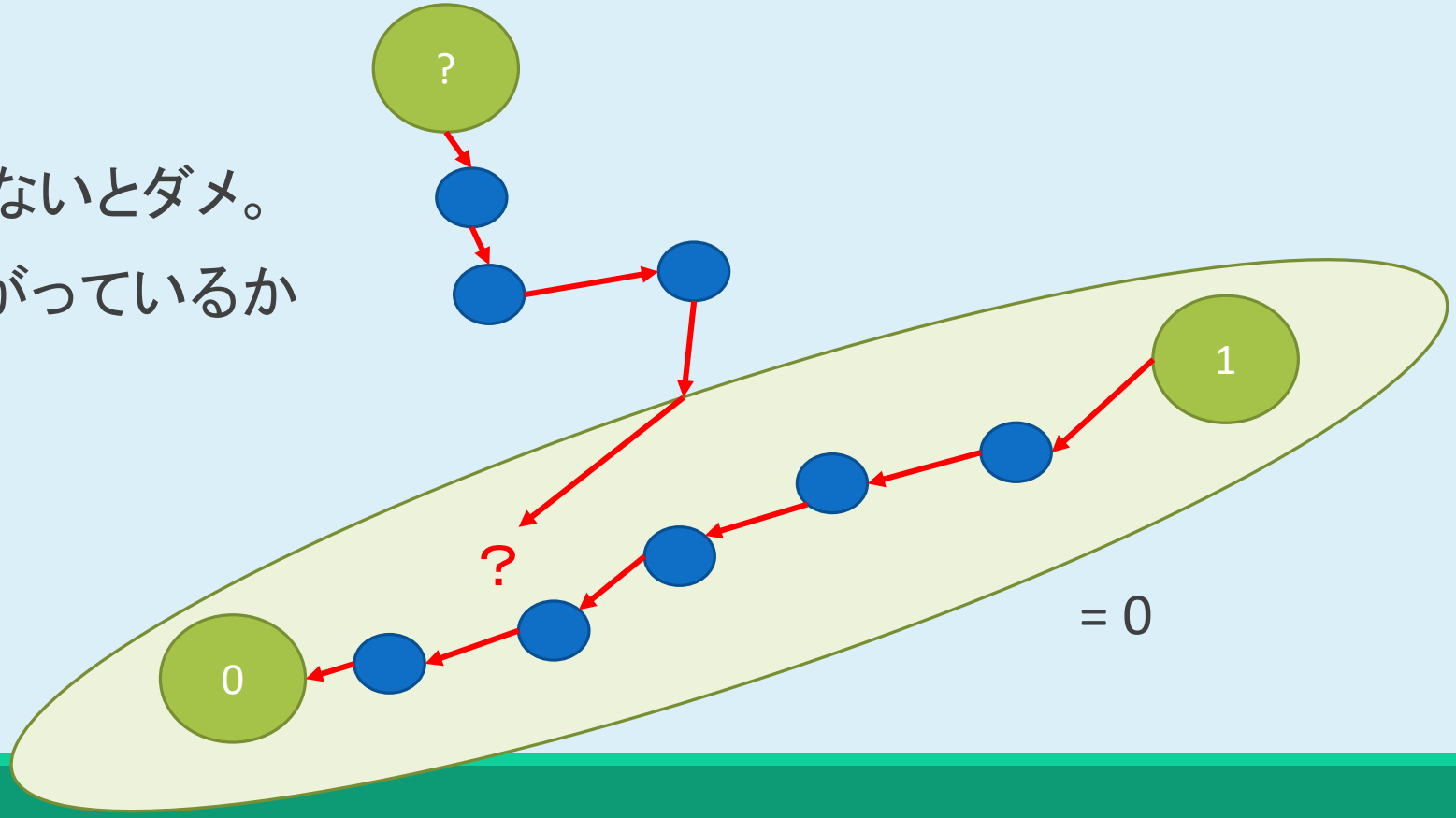


小課題 4 (30 点)

- それが終わった後は？
- わかっている部分全体を頂点 0 と同じに扱えば、同様にできる。

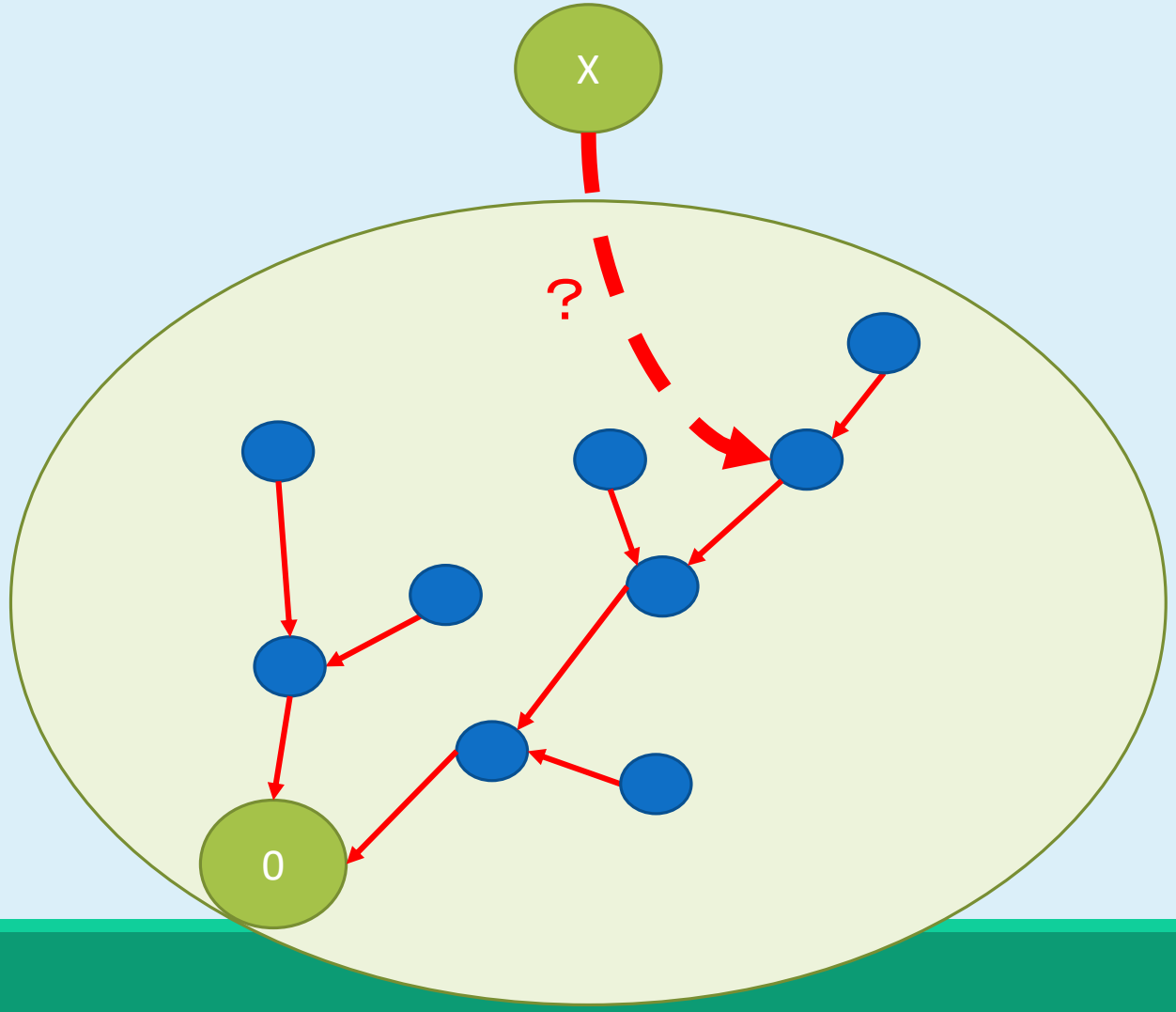
- ただし最後の辺の扱いは変えないとダメ。

（既知の頂点集合のどれにつながっているか分からないため。）



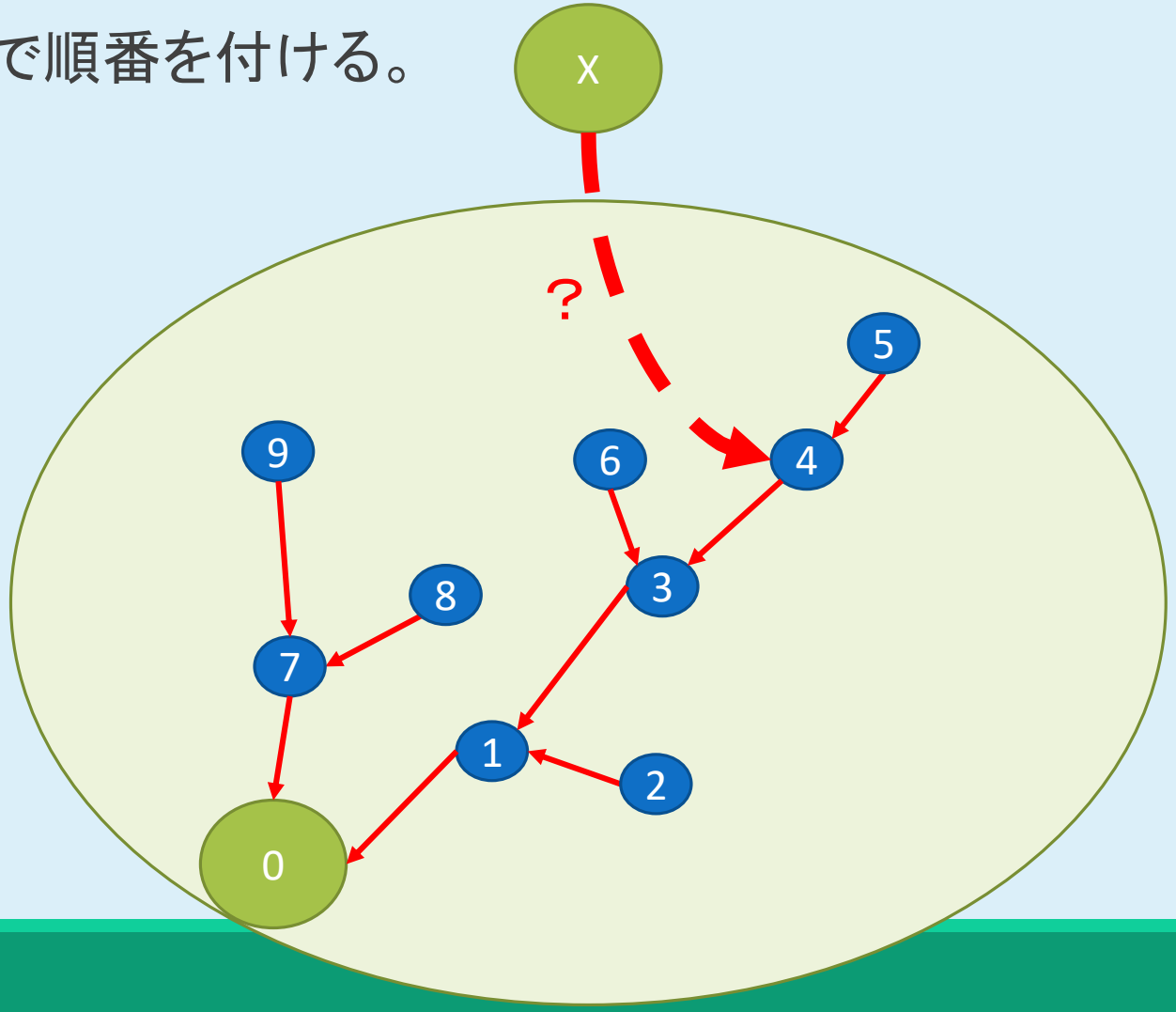
小課題 4 (30 点)

- これも**二分探索**で特定できる。



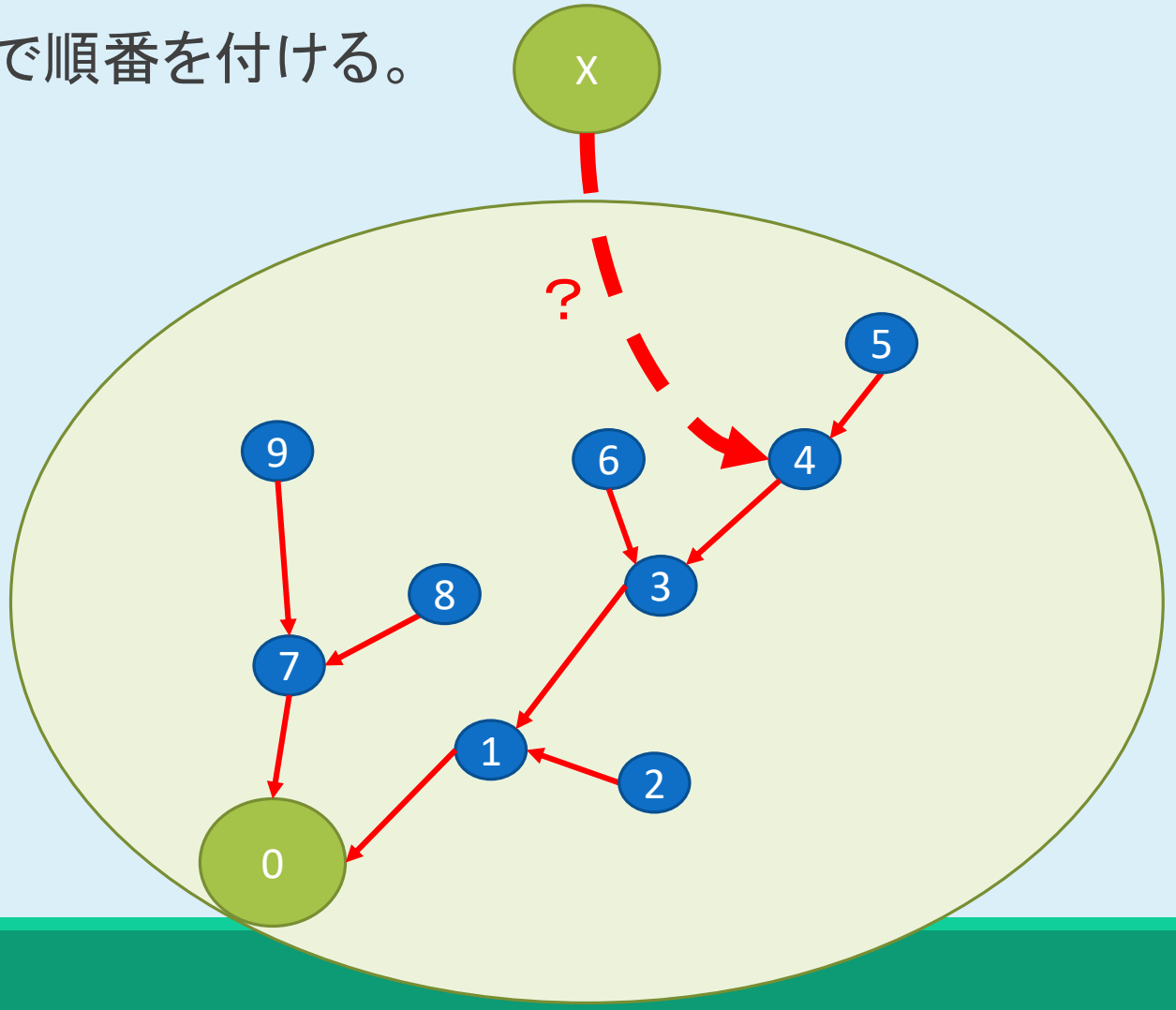
小課題 4 (30 点)

- これも**二分探索**で特定できる。
- 木に DFS 順（BFS 順でも良い）で順番を付ける。



小課題 4 (30 点)

- これも**二分探索**で特定できる。
- 木に DFS 順（BFS 順でも良い）で順番を付ける。
- 先頭から何番目まで含めると頂点 0 に到達できるかで**二分探索**する。



小課題 4 (30 点) 解法

- 以降の説明のため、スタックを導入します（既知ノード集合：0 との接続関係が確定した木構造とする。初期は頂点 0 のみ）。

1. スタックが空なら未探索ノードを入れる。
2. 今のスタックトップが既知ノードとつながっているか見る。

→つながっていないなら二分探索して間の頂点を

見つけてスタックに追加する

→　つながっているなら 3. をしてスタックから既知ノード集合に。

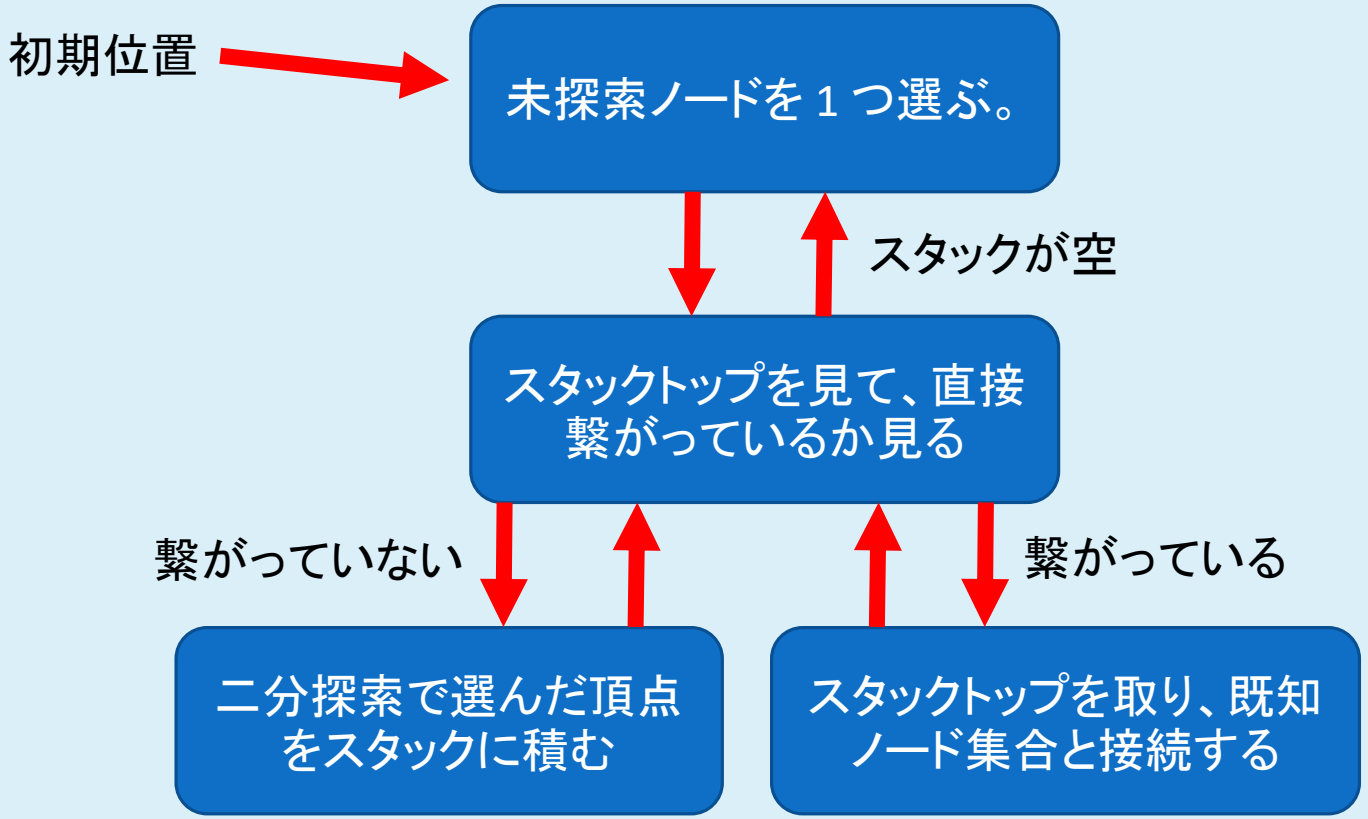
3. 既知ノードとの辺の接続関係を二分探索で決定する。

- $2N$ 回ほど二分探索をし、N 回ほど 2 の最初の判定を行うので、

  全体で $(2logN+1)N \leqq 32{,}200$ 回ほどで良く、OK。

小課題 4 (30 点) 解法

- 図にするとこのような形。



小課題 5 (23 点)

- 一般グラフ。

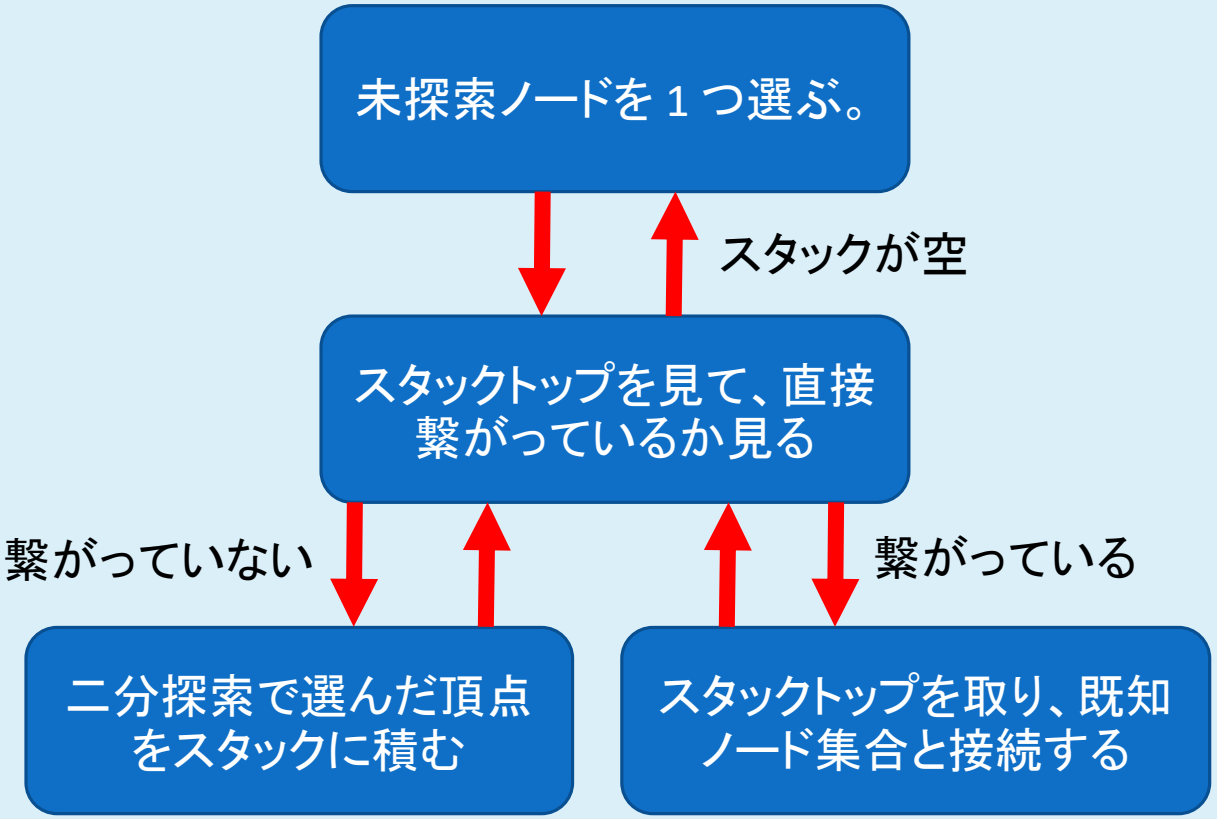
小課題 5 (23 点)

- 一般グラフ。

- 一見どうしようもなさそうですが、実は**小課題 4 の解法を少しいじるだけ**で満点をとれます。

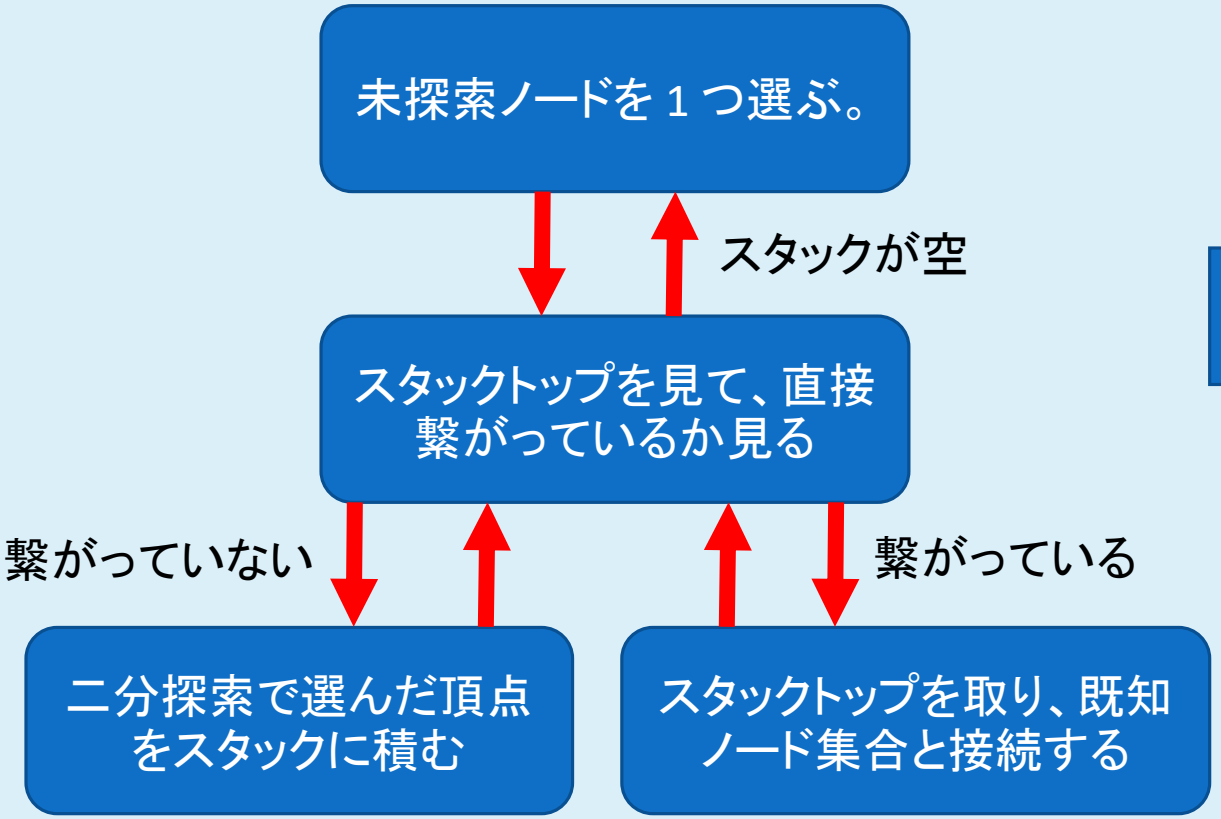
小課題 5 (23 点)

- 小課題 4 の解法を確認してみる。

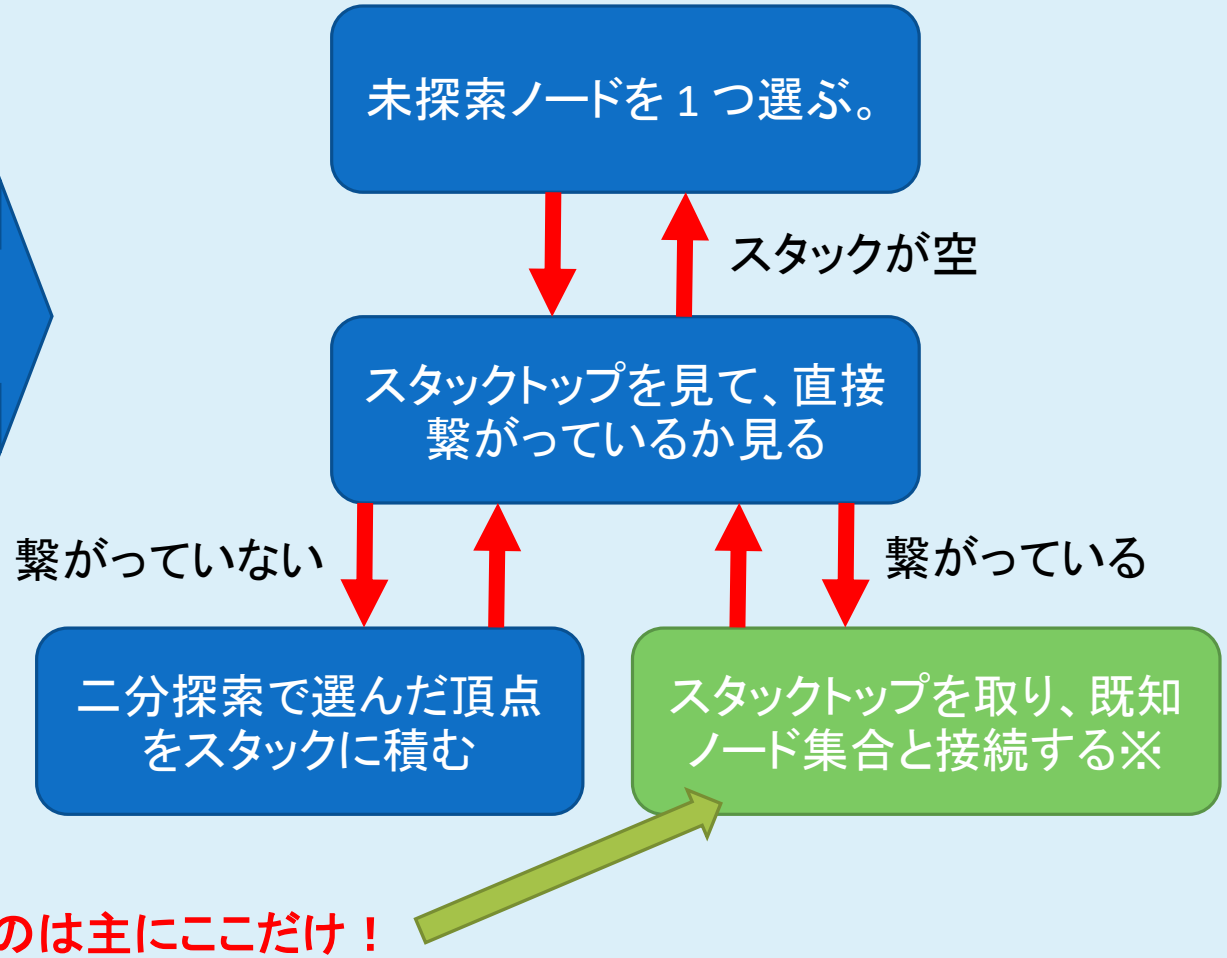


小課題 5 (23 点)

- 小課題 4 の解法を確認してみる。



- 小課題 5 の解法はほとんど同じ。



小課題 5 (23 点)

- 確かに一般グラフでもスタック周りは同じように動きそうだけれど、スタックの中身はどうなるの？
- 木ならばパス上という性質が成り立っていたが、一般グラフだとどんな性質が成り立つの？

小課題 5 (23 点)

- 確かに一般グラフでもスタック周りは同じように動きそうだけれど、スタックの中身はどうなるの？
- 木ならばパス上という性質が成り立っていたが、一般グラフだとどんな性質が成り立つの？

- 実は下記のことが言えます：

命題：
スタックのどの要素についても、どのタイミングにおいても、それよりスタックの奥に置かれているいずれの頂点も使用せずに頂点 0 に行くことができる。

どういうことかというと、スタックの中身を [X1,X2,…,Xk] としたとき、Xi (1 ≦i≦k) が、頂点X1,…,X(i-1) のいずれをも使用せずに頂点 0 に到達可能であるという性質が常に成立するという主張。

小課題 5 (23 点)

命題:
スタックのどの要素についても、どのタイミングにおいても、それよりスタックの奥に置かれているいずれの頂点も使用せずに頂点 0 に行くことができる。

- 証明(概略):

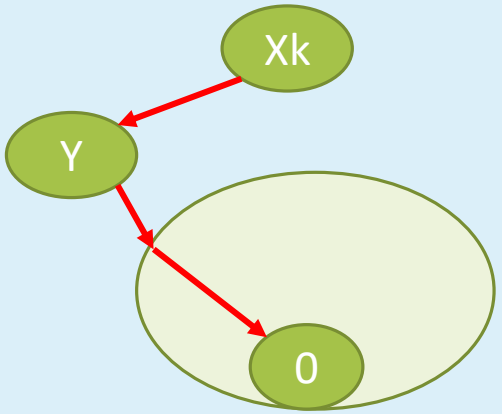
スタック内の要素は上から順に取られるので、一度スタックに入ったらそれより下の頂点集合は変化しない。→スタックに追加する際に性質が満たされていればOK。

今、スタックに [X1,X2,…,Xk] が積まれているとして、新たに Y を積むことを考える。

Y が選ばれたということは、「X から Y を経由して既知ノード集合に入り頂点 0 へ至る点素パスが存在する」ということである。(∵ Y を選べなくなった瞬間に X から 0 へのパスがすべて消滅する)

このうち Y 以降を考えると、仮定より Xk より前の要素存在しないし、点素なので Xk も登場しない。

よって Y をスタックに入れても性質が壊れない！□



小課題 5 (23 点)

- 以上のスタックを使うと何が嬉しいのか？

→**スタックトップからは常に既知ノード集合に到達可能**である。

- スタックに追加する回数は高々 $N-1$ 回なので、効率的にすべてのノードを既知ノード集合に加えることができるようになる！

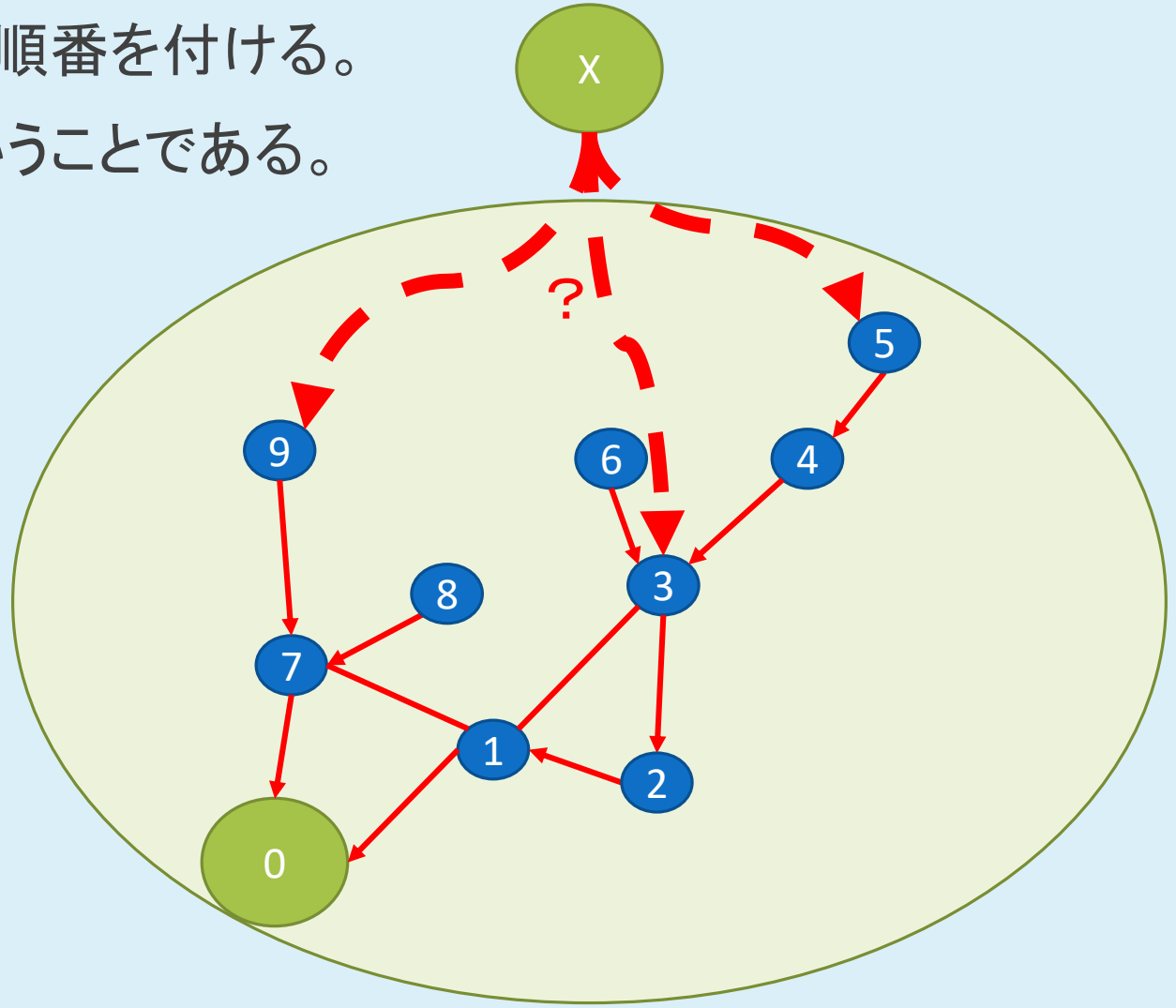
小課題 5 (23 点)

- さて、ここまでは既知ノード集合に接続する頂点の選び方である。
- では、実際に繋ぐときはどうすればいいのか？

→ 小課題 4 のやり方とほとんど同じである。

小課題 5 (23 点)

- 一般グラフに DFS 順（BFS 順でも良い）で順番を付ける。
- 厄介なのが、辺が**複数本**伸びているということである。

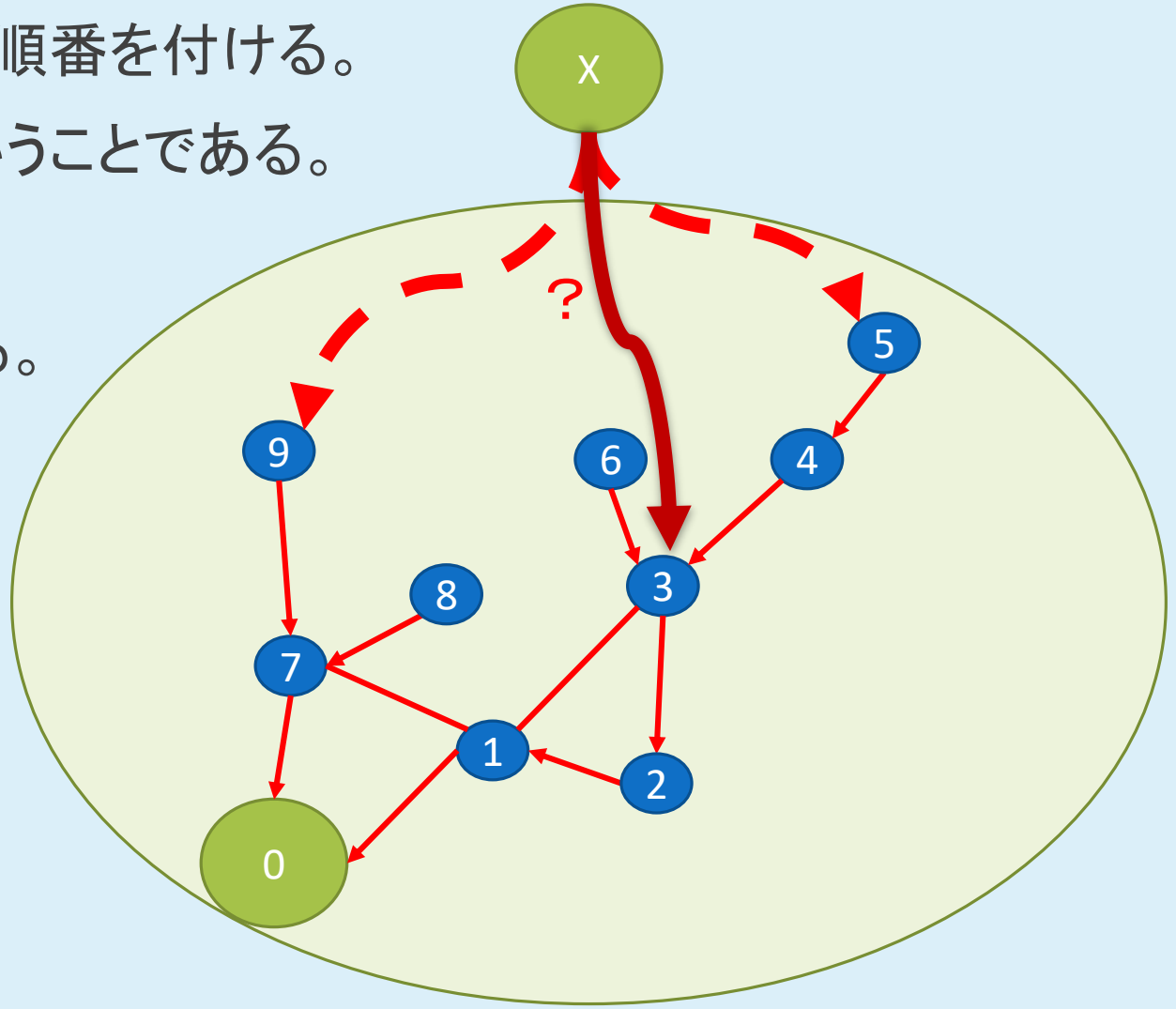


小課題 5 (23 点)

- 一般グラフに DFS 順（BFS 順でも良い）で順番を付ける。
- 厄介なのが、辺が**複数本**伸びているということである。

- **二分探索**すれば、辺を 1 本は特定できる。

（右の例だと 3 への辺が発見できる）



小課題 5 (23 点)

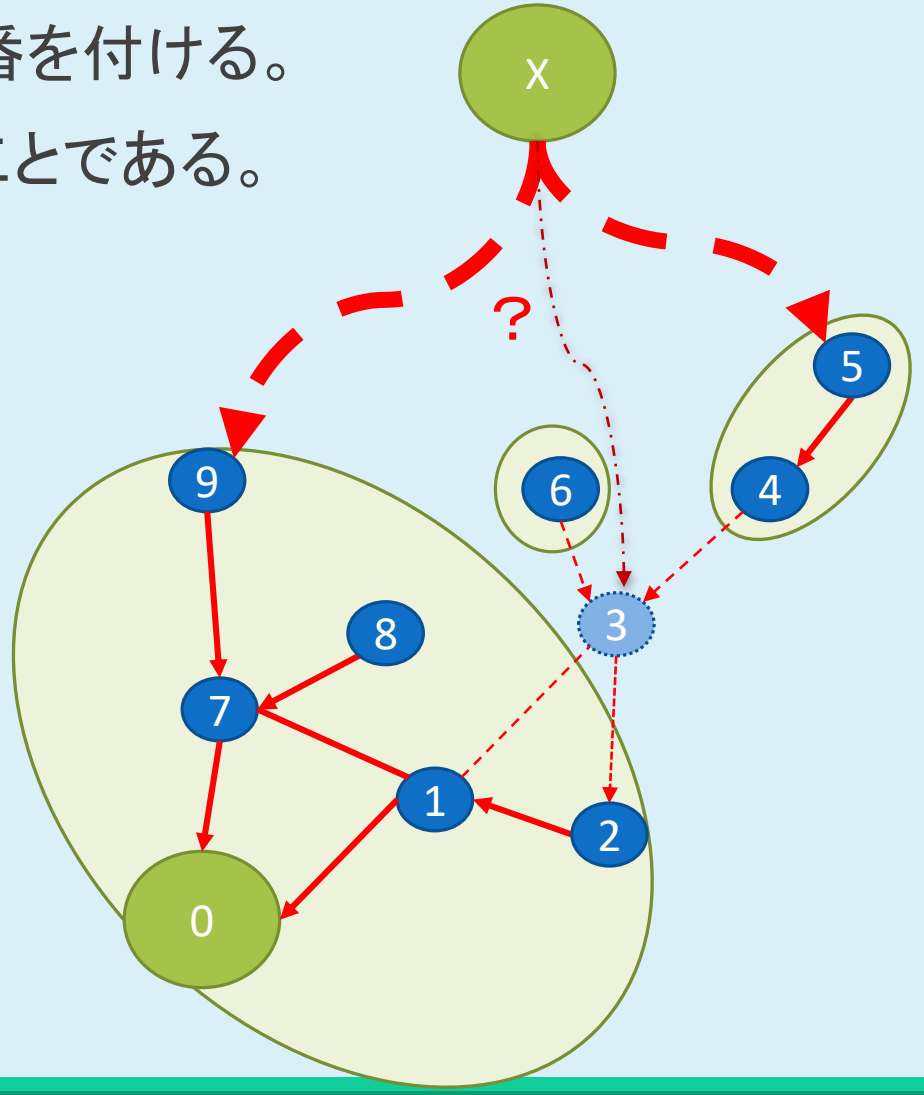
- 一般グラフに DFS 順（BFS 順でも良い）で順番を付ける。
- 厄介なのが、辺が**複数本**伸びているということである。

- **二分探索**すれば、辺を 1 本は特定できる。

（右の例だと 3 への辺が発見できる）

- 残りの頂点は、発見した辺に関する頂点を除いた各連結成分について見ることで発見できる！

（構造は知っているので他の連結成分についても頂点 0 の役割を担う頂点を決定できる。）

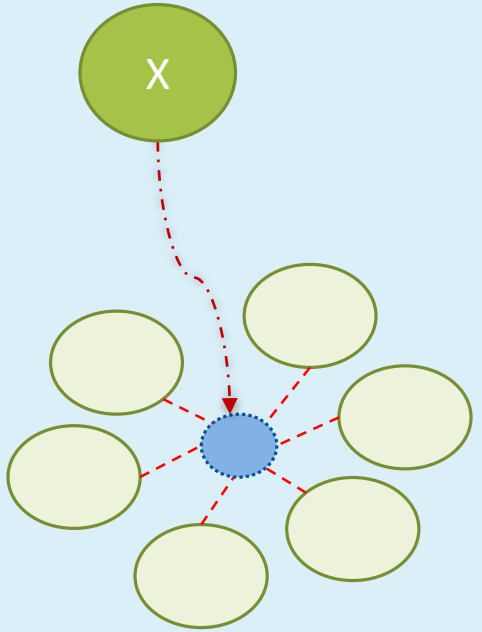


小課題 5 (23 点)

- しかし、増えた連結成分に等しい個数だけ追加でコストがかかってしまう。(大量に連結成分が増えると最悪なことになる)

小課題 5 (23 点)

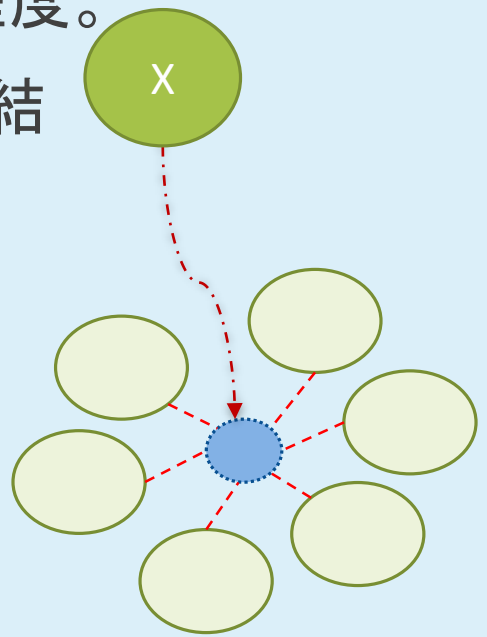
- しかし、増えた連結成分に等しい個数だけ追加でコストがかかってしまう。(大量に連結成分が増えると最悪なことになる)
- ところでどの頂点も**次数が 7 以下**だったので、追加で見なければならなくなる連結成分は**高々 6 個**である。



小課題 5 (23 点)

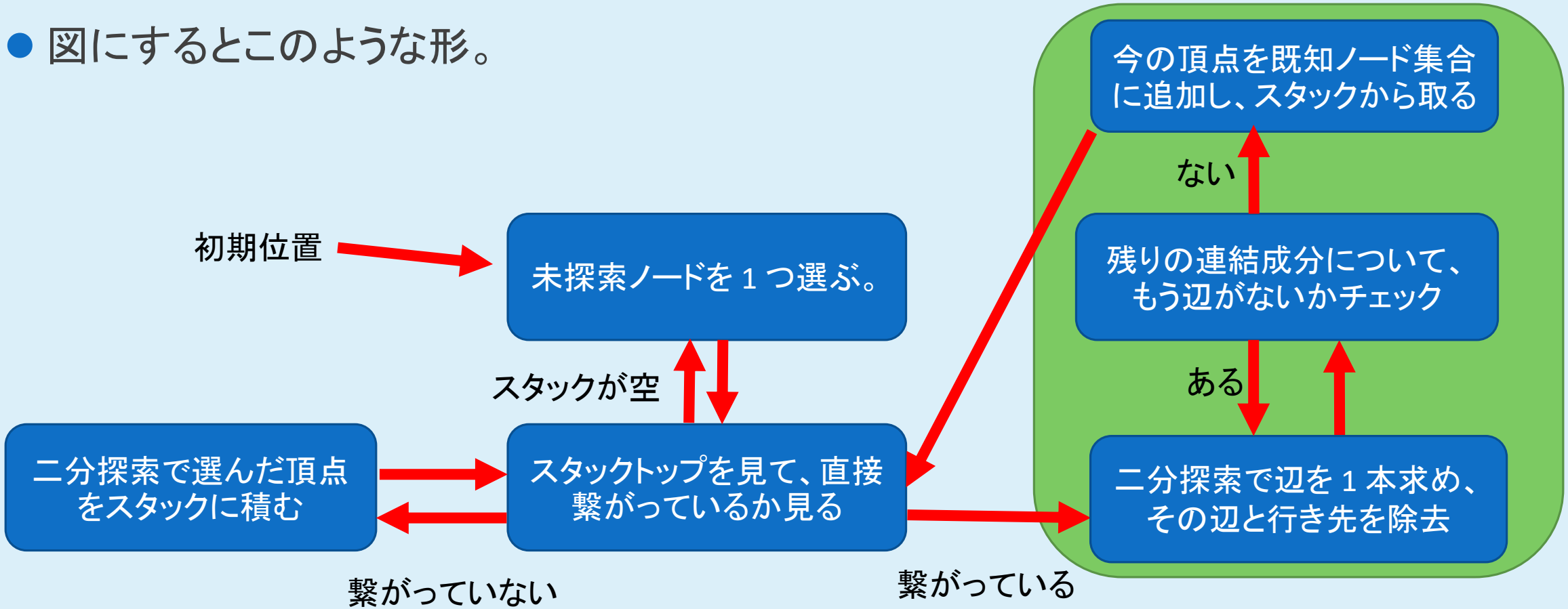
- しかし、増えた連結成分に等しい個数だけ追加でコストがかかってしまう。(大量に連結成分が増えると最悪なことになる)
- ところでどの頂点も**次数が 7 以下**だったので、追加で見なければならなくなる連結成分は**高々 6 個**である。
- ということは、辺の発見以外で要求される追加コストは $\boldsymbol{6M}$ 回程度。

(∵辺があると言われるのが M 回で、辺がないと言われるのが連結成分の個数回。解析すると $\boldsymbol{6M}$ 回くらいになる。)



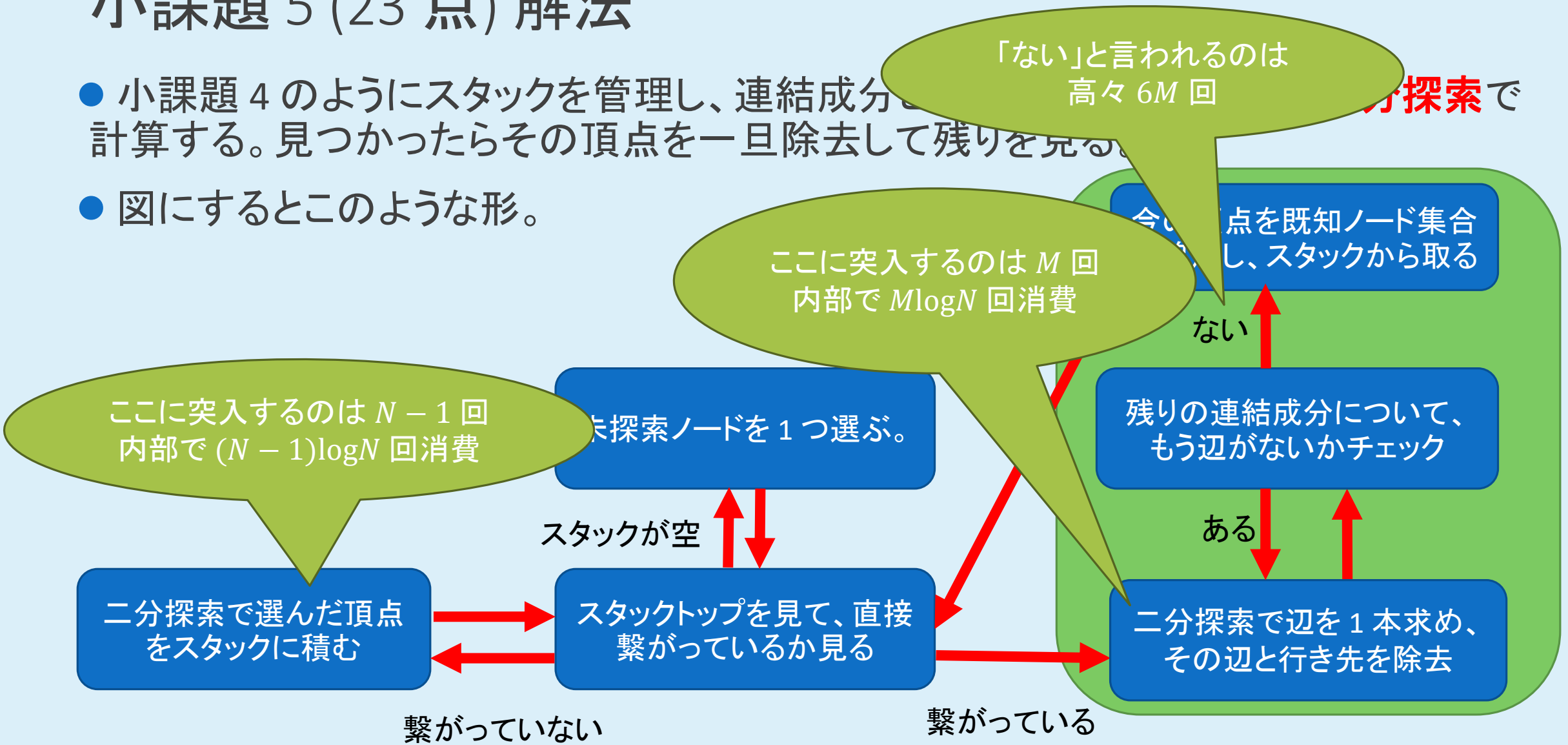
小課題 5 (23 点) 解法

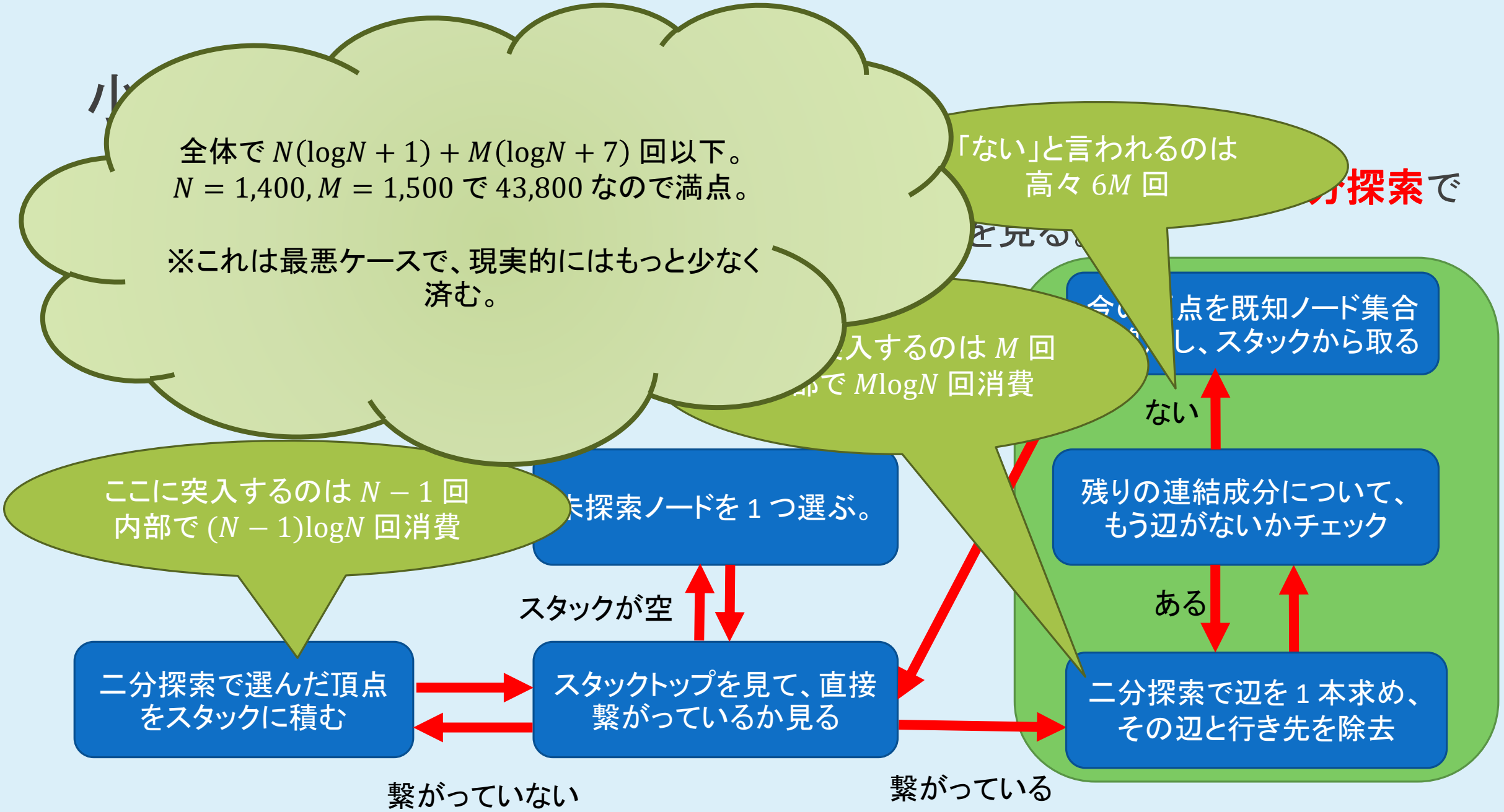
- 小課題 4 のようにスタックを管理し、連結成分ごとに辺があるか判定し**二分探索**で計算する。見つかったらその頂点を一旦除去して残りを見る。
- 図にするとこのような形。

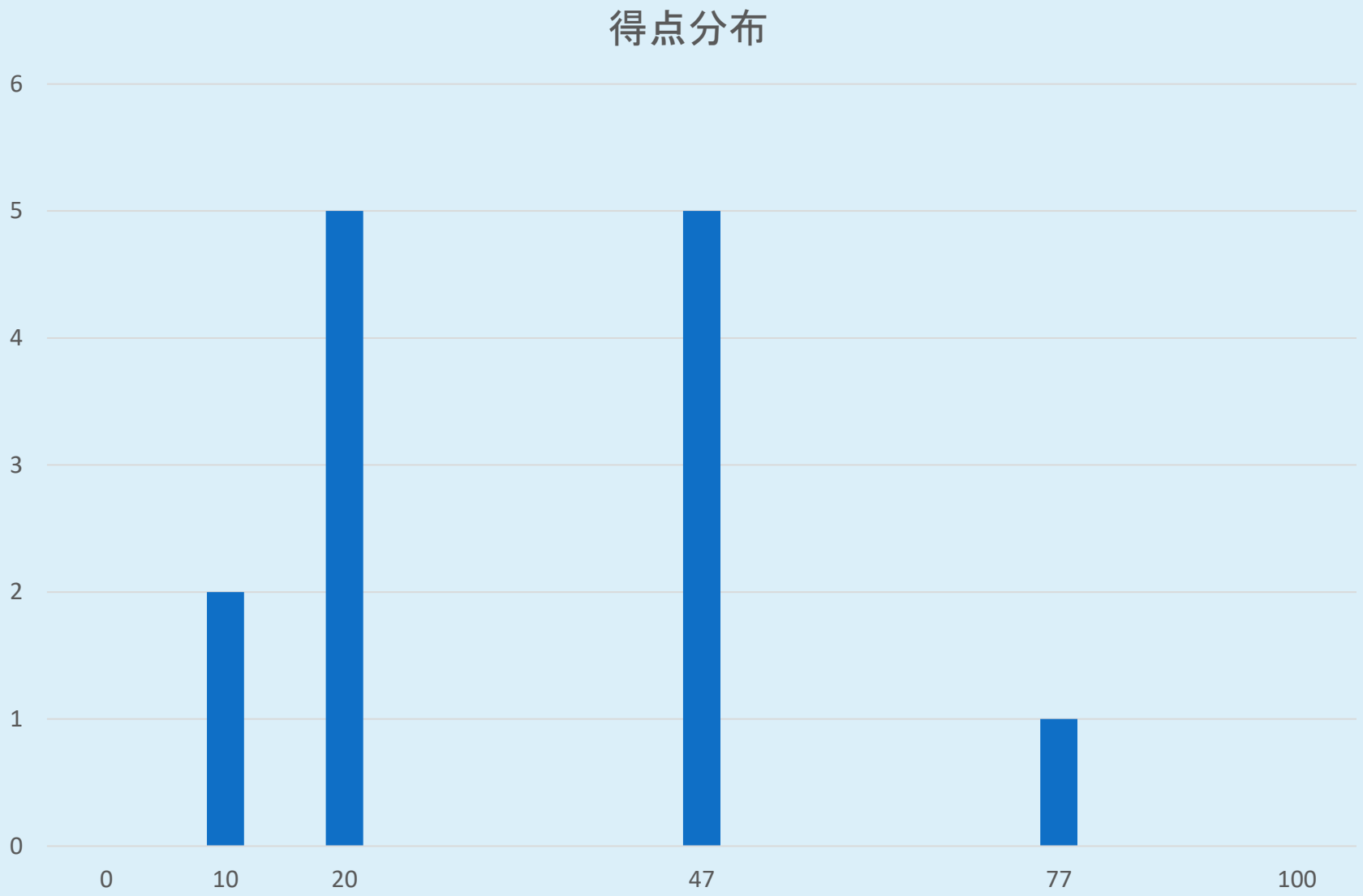


小課題 5 (23 点) 解法

- 小課題 4 のようにスタックを管理し、連結成分と[illegible]**二分探索**で計算する。見つかったらその頂点を一旦除去して残りを見る。
- 図にするとこのような形。




全体で $N(\log N+1)+M(\log N+7)$ 回以下。
$N=1{,}400, M=1{,}500$ で 43,800 なので満点。
※これは最悪ケースで、現実的にはもっと少なく済む。
「ない」と言われるのは高々 $6M$ 回
ここに突入するのは $N-1$ 回
内部で $(N-1)\log N$ 回消費
ここに突入するのは M 回
内部で $M\log N$ 回消費
未探索ノードを 1 つ選ぶ。
スタックが空
二分探索で選んだ頂点をスタックに積む
スタックトップを見て、直接繋がっているか見る
繋がっていない
繋がっている
ない
ある
残りの連結成分について、もう辺がないかチェック
二分探索で辺を 1 本求め、その辺と行き先を除去



※欠席1名を除く