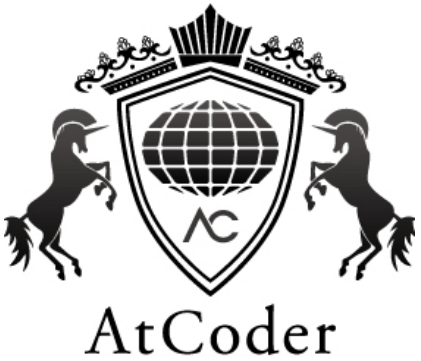
AtCoder Beginner Contest 010
解説



AtCoder株式会社 代表取締役
高橋 直大

競技プログラミングを始める前に

AtCoder

- 競技プログラミングをやったことがない人へ
  - まずはこっちのスライドを見よう！
  - http://www.slideshare.net/chokudai/abc004

A問題　ハンドルネーム

1. 問題概要
2. アルゴリズム

A問題　問題概要

AtCoder

- 文字列 S が与えられる
- S に “pp” を足して出力しなさい。

- 制約
- $1 \leqq |S| \leqq 10$

A問題　アルゴリズム

AtCoder

- 基本的なプログラムの流れ
  - 標準入力から、必要な入力を受け取る
    - 今回の場合は、S という1つの文字列
  - 問題で与えられた処理を行う
    - 今回は、S に “pp” を追加する
  - 標準出力へ、答えを出力する

A問題　アルゴリズム

AtCoder

- 入力
  - 1つの文字列を、標準入力から受け取る
    - Cであれば、scanf(“%s”, &s);　など
    - C++であれば、cin >> s;
    - 入力の受け取り方は、下記の練習問題に記載があります。
      - http://practice.contest.atcoder.jp/tasks/practice_1

A問題　アルゴリズム

AtCoder

- 今回の問題は、S に “pp” を足すだけ
- S += “pp”; などで、文字列の追加が可能
  - 言語によって文字列の弄り方は違うので、検索などで調べよう！
- ret = S + “pp”;のように、入力と別の文字列を作っても良い

- 文字列を足さなくても、順番に出力しても良い
  - Print(S);
  - Print(“pp¥n”); みたいな感じ。

A問題　アルゴリズム

AtCoder

- 出力
  - 求めた答えを、標準出力より出力する。
  - 言語によって違います。
    - printf(“%s¥n”, s); (C)
    - cout << S << endl; (C++)
    - System.out.println(S); (Java)
    - 各言語の標準出力は、下記の練習問題に記載があります。
      - http://practice.contest.atcoder.jp/tasks/practice_1

B問題　花占い

1. 問題概要
2. アルゴリズム

B問題　問題概要

AtCoder

- N個の整数の配列aが与えられる
  - これが、庭に存在する花の花びらを行う
- 2パターンの花占いを行う可能性がある
  - 「好き」「嫌い」のループ
  - 「好き」「嫌い」「大好き」のループ
- 「嫌い」にならないように、予め花びらを毟る
- 毟る必要のある花びらの枚数を出力しなさい。

B問題　アルゴリズム

AtCoder

- 入力
  - 整数nを受け取る
  - 数列aの数字をn個受け取る
    - 受け取り方は複数いくつかある
      - 1行纏めて受け取って、スペースでsplitする
      - 1つずつ受け取る
    - 言語によって受け取りやすい書き方が違う！
    - 詳しくはpracticeで確認しよう！
      - http://practice.contest.atcoder.jp/tasks/practice_1

B問題　アルゴリズム

AtCoder

- 処理
  - 各花びらについて、何枚花びらを毟る必要があるかを求める。
  - やり方は複数存在する。

B問題　アルゴリズム

AtCoder

- 解法1
  - 実際に試してみる
  - “suki”, “kirai”でループを回して、kiraiになったら失敗 など
  - もうちょっと簡単に、bool型の配列などに直す なども。
    - 「好き」「嫌い」 → true, false
    - 「好き」「嫌い」「大好き」 → true, false, true

- 解法2
  - あまりを利用する
    - 「好き」「嫌い」 → 2で割った時割り切れるとダメ
    - 「好き」「嫌い」「好き」 → 3で割った時、2余るとダメ

B問題　アルゴリズム

AtCoder

- 解法3
  - それぞれの枚数について、大丈夫かどうか予め書いておく
    - 1, 3, 7, 9の時大丈夫
    - {false, true, false, true, false, false, false, true, false, true}みたいな配列を予め作ってしまえば、判定の必要がない
- 解法4
  - それぞれの枚数について、峯る枚数を予め書いておく
    - {0, 0, 1, 0, 1, 2, 3, 0, 1, 0}
    - これなら足し算するだけ
- どちらも入力ミスに注意！

B問題　アルゴリズム

AtCoder

- 出力
  - A問題と同じく、答えを出力するだけ
  - Print(ret)みたいな感じ

C問題　浮気調査

1. 問題概要
2. アルゴリズム

C問題　問題概要

AtCoder

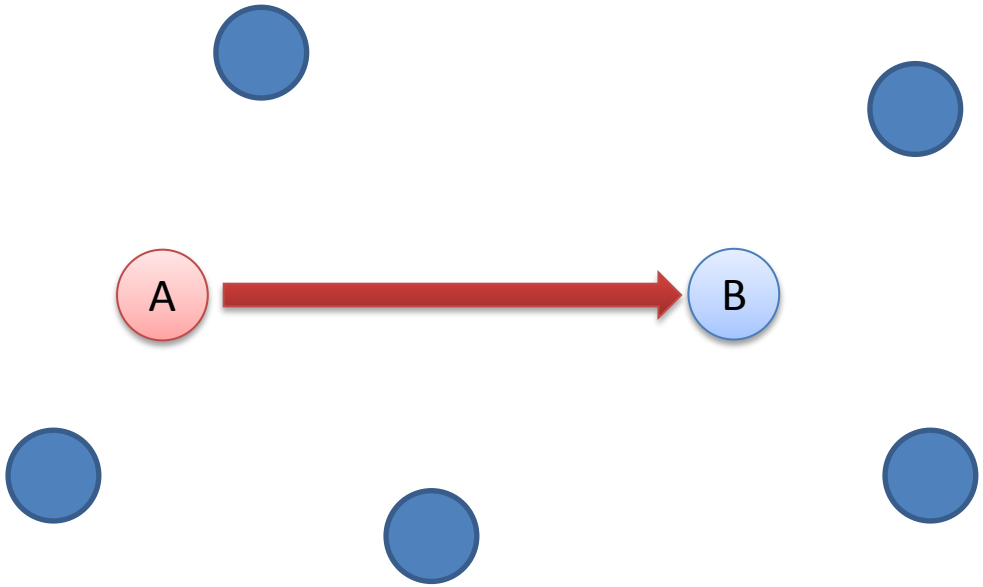
- 地点Aから地点Bに移動します。
- 最高速度と、かかった時間が与えられます。
- 寄り道出来る場所の候補が与えられます。
- どこかに寄り道することが可能か出力しなさい。

- 制約
  - 0 ≦ 全ての座標 ≦ 1000
  - 1 ≦ 分速V ≦ 100
  - 1 ≦ 時間T ≦ 50
  - 1 ≦ 寄り道候補n ≦ 1000

C問題　アルゴリズム

AtCoder

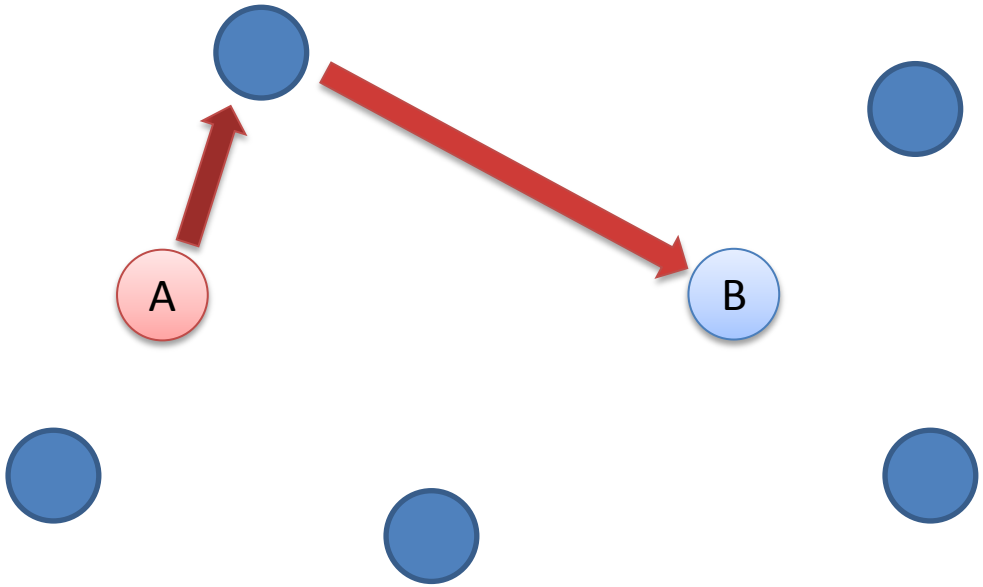
- 図のような移動をしたい



C問題　アルゴリズム

AtCoder

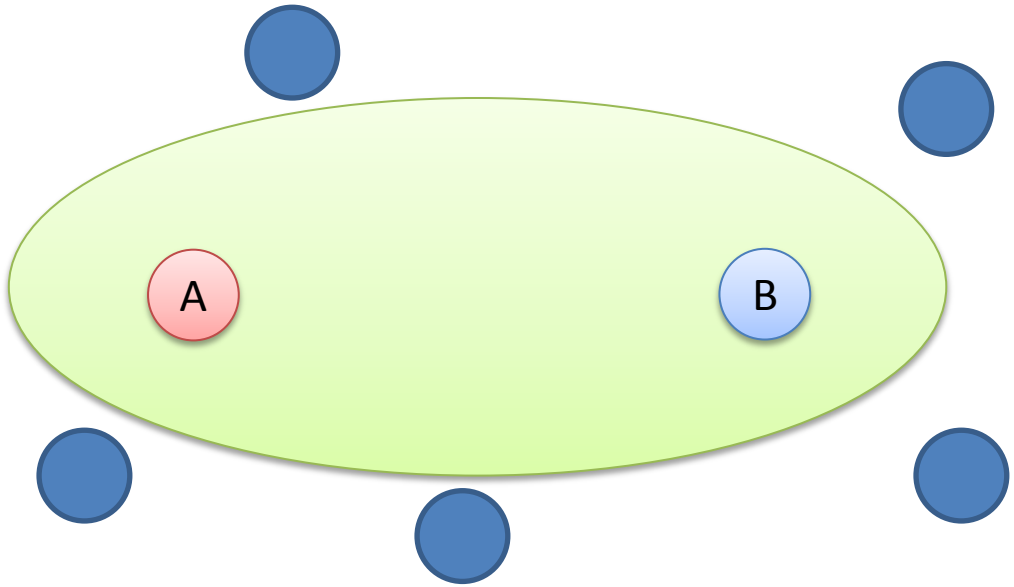
- 図のような移動をしたい
- こうした寄り道が可能か考える



C問題　アルゴリズム

AtCoder

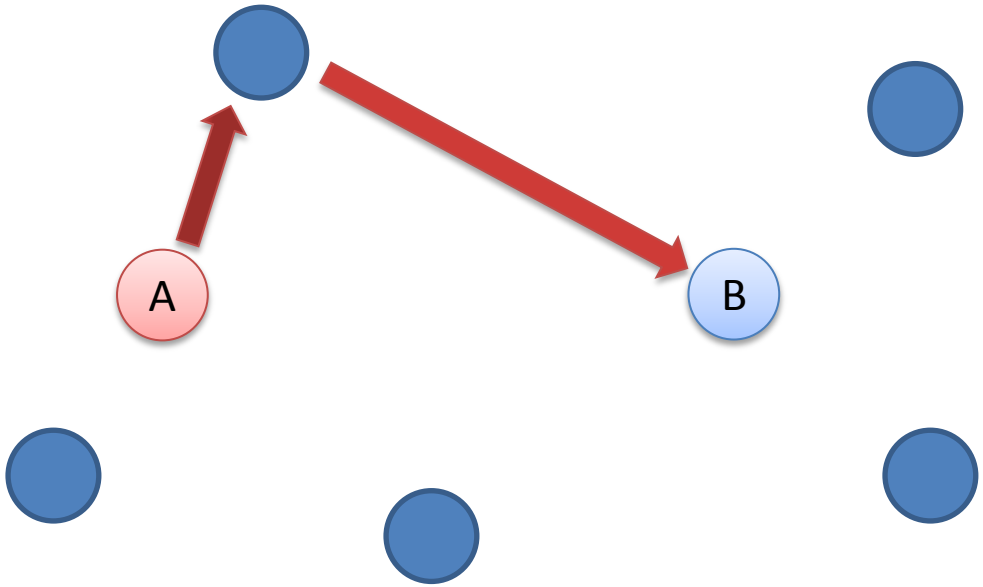
- 寄り道可能な範囲は、こんな感じの楕円になる！
  - AとBの間に、紐を結んで、紐をぴんと張った時に描ける図形は楕円であることは、そこそこ有名
  - ・・・だが、今回は、こんな知識を使う必要は全くない。



C問題　アルゴリズム

AtCoder

- こうした寄り道が可能か考える
  - 赤の矢印の長さが、T×V以下になっているかどうかを考えれば良い！
  - であれば、矢印の長さを計算して、足せば良い



C問題　アルゴリズム

AtCoder

- 距離の計算方法
  - X座標がx、Y座標がy離れている場合
  - Sqrt(x * x + y * y)で計算出来る！
    - 三平方の定理

C問題　アルゴリズム

AtCoder

- 注意点
  - 小数での演算になるので、誤差に注意しよう！
  - eps(非常に小さい値)を使うと良いことが多い
    - If(T * V >= dist1 + dist2) …
      - これだと、誤差がちょっと出ると死ぬ
    - If(T * V + eps >= dist1 + dist2) …
      - これだと誤差に強くなる
        - » なぜなら、T*Vぴったりの距離を作ることは可能だが、T*Vよりほんのちょっとだけ少ない値、というのは非常に作るのが難しい
  - 楕円の方程式からきっちり解けば、誤差なしで計算も出来るかも。

D問題　浮気防止

1. 問題概要
2. アルゴリズム

D問題　問題概要

AtCoder

- 高橋君の作ったSNSの、友人関係が与えられる。
- 特定の人たちに対して、メッセージを届かないようにしたい。
- このため、以下のような工作を行う。
  - 友人関係を1つ解消する。
  - 一人のパスワードを変更し、メッセージを閲覧不可能にする
- 工作を行う回数の必要数を出力しなさい

D問題　問題概要

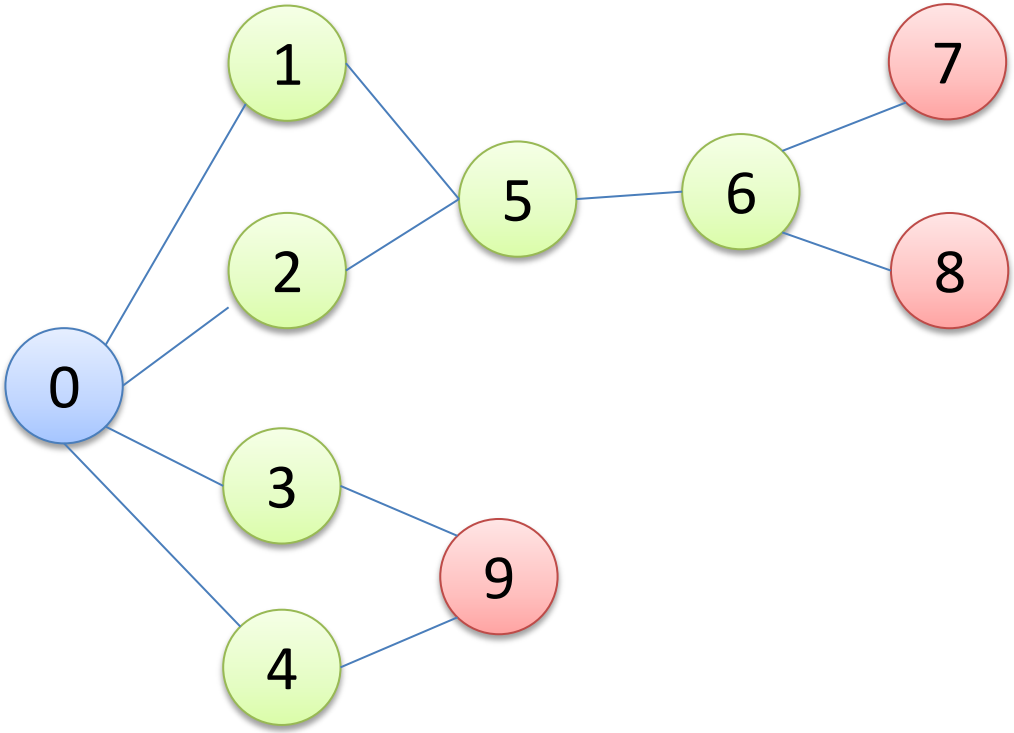
AtCoder

- 制約
  - $1 \leqq V \leqq 100$
  - $0 \leqq G \leqq V-1$
  - $0 \leqq E \leqq V * (V-1) / 2$
- 部分点の制約
  - $0 \leqq E \leqq V * (V-1) / 2$

D問題　部分点1　アルゴリズム

AtCoder

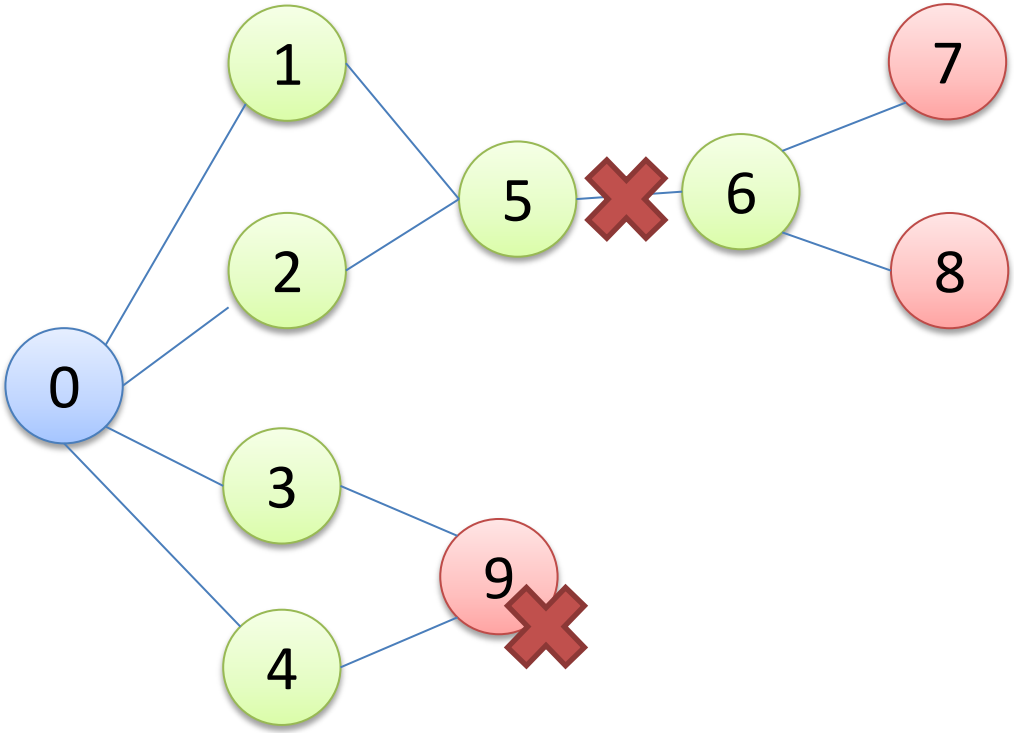
- 以下のような図を考える



D問題　部分点1　アルゴリズム

AtCoder

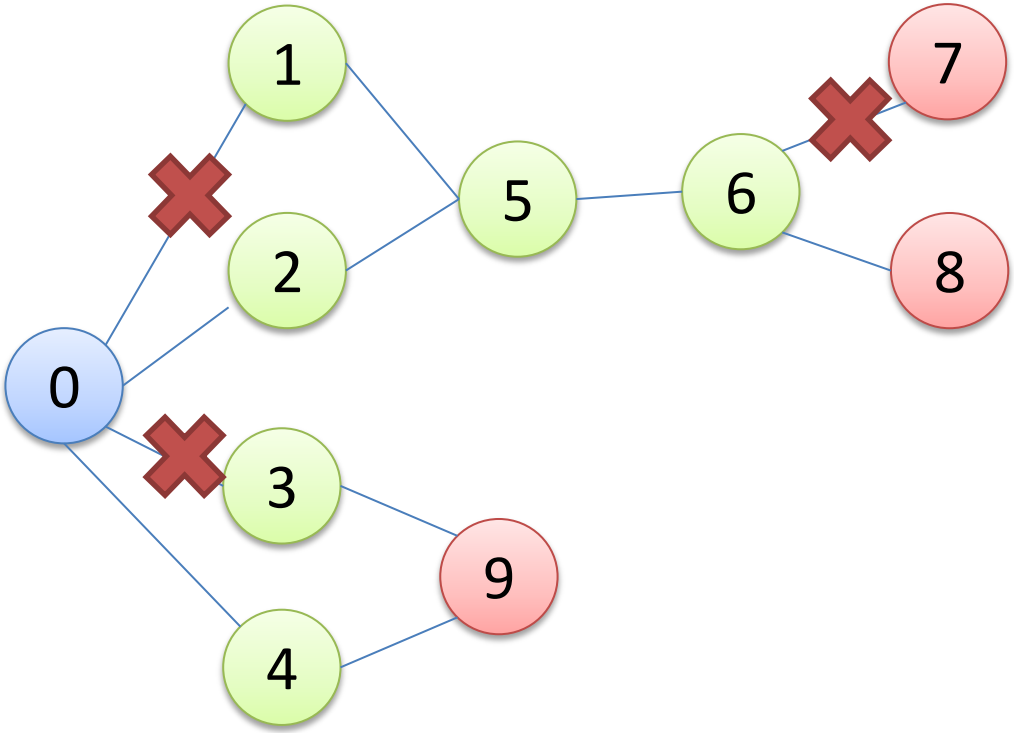
- 以下のような図を考える
- 最適解はこんな感じ



D問題　部分点1　アルゴリズム

AtCoder

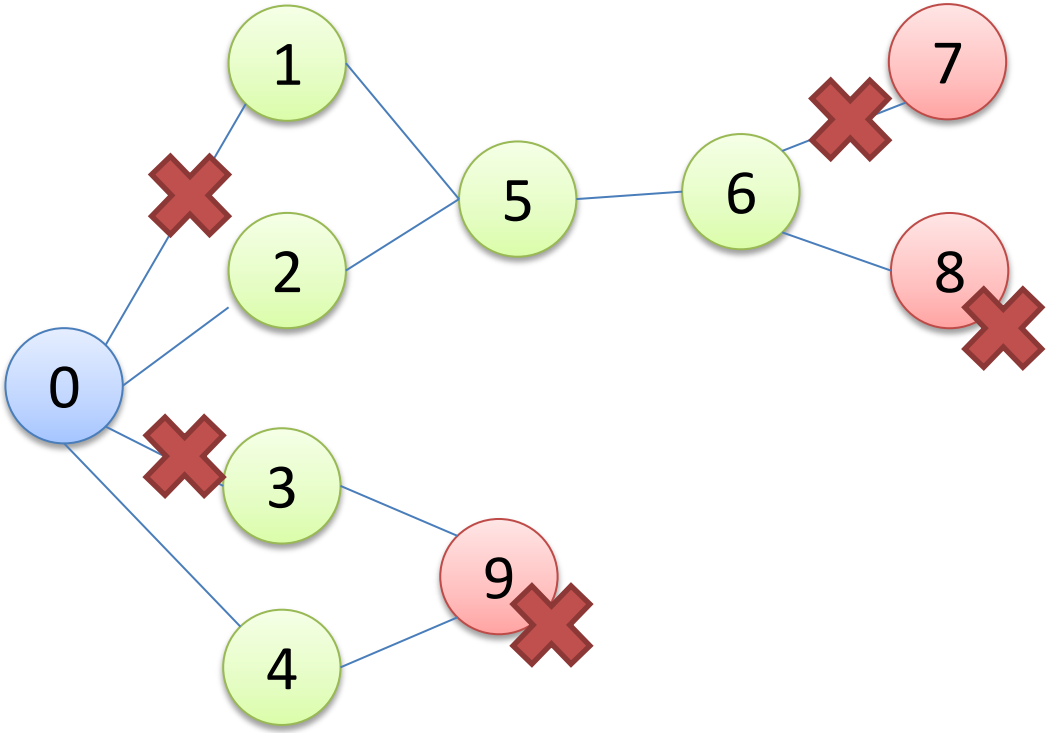
- 以下のような図を考える
- 適当に線に×をつける



D問題　部分点1　アルゴリズム

AtCoder

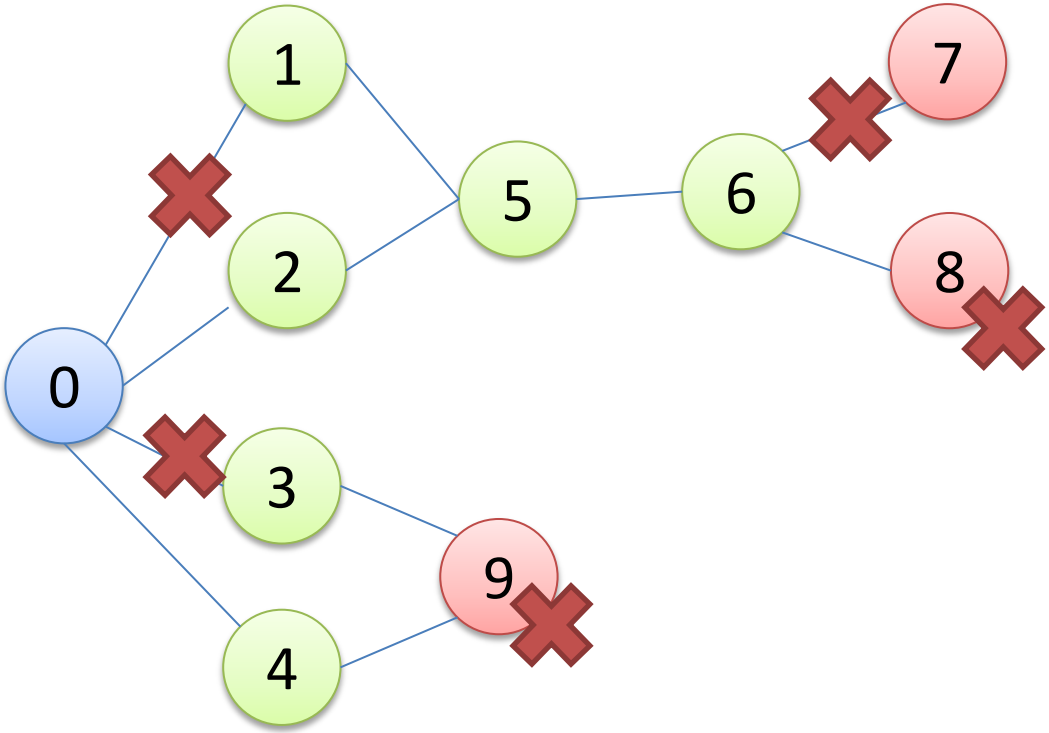
- 以下のような図を考える
- 適当に線に×をつける
- 高橋君から辿りつけてしまう女の子のパスを変える



D問題　部分点1　アルゴリズム

AtCoder

- つまり、「工作を行う友人関係」の数さえ分かれば、高橋君から辿りつける友人を探索で求め、その数を求めれば良い！



D問題　部分点１　アルゴリズム

AtCoder

- 部分点１の時は、友人関係の時は12以下
  - つまり、工作を行うかどうかは2^12通り！
  - 2^12通り全て試してしまえば良い

- 全探索のやり方は、幾つかある
  - 深さ12の深さ優先探索を行う
  - 整数のbitを利用して、0から(1<<12)-1までのループを回す

D問題　部分点１　アルゴリズム

AtCoder

- 整数のbitを利用する方法
  - 例えば、E=3の時、0から7までのループを回す
    - 0 → 2進数だと000
    - 1 → 2進数だと001
    - 2 → 2進数だと010
    - 3 → 2進数だと011
    - 4 → 2進数だと100
    - 5 → 2進数だと101
    - 6 → 2進数だと110
    - 7 → 2進数だと111
  - これを利用する。

D問題　部分点1　アルゴリズム

AtCoder

- 整数のbitを利用する方法
  - 例えば、E=3の時、0から7までのループを回す
    - 0 → 2進数だと000
    - 1 → 2進数だと001
    - 2 → 2進数だと010
    - 3 → 2進数だと011
    - 4 → 2進数だと100
    - 5 → 2進数だと101
    - 6 → 2進数だと110
    - 7 → 2進数だと111
  - 一つ目の人間関係は、1ケタ目を見る

D問題　部分点1　アルゴリズム

AtCoder

- 整数のbitを利用する方法
  - 例えば、E=3の時、0から7までのループを回す
    - 0 → 2進数だと000
    - 1 → 2進数だと001
    - 2 → 2進数だと010
    - 3 → 2進数だと011
    - 4 → 2進数だと100
    - 5 → 2進数だと101
    - 6 → 2進数だと110
    - 7 → 2進数だと111
  - 2つ目の人間関係は、2ケタ目を見る

D問題　部分点1　アルゴリズム

AtCoder

- 整数のbitを利用する方法
  - K桁目のbitを取得するには？　（0ケタ目から数えて）
    - 求めたい整数がiとして
    - (i >> k) % 2　を計算すれば良い！
  - K個bitを右にずらした後、2で割った余りを求めれば良い！

D問題　部分点1　アルゴリズム

AtCoder

- 計算量
  - 全探索の回数 O(2^E)
  - それぞれの幅優先探索の処理数 O(V)
  - 併せて、計算量はO(V 2^E)
  - 10万程度なので余裕で間に合う！
    - 豆知識：C++なら1億回の計算で1秒くらい。

D問題　完答　アルゴリズム

AtCoder

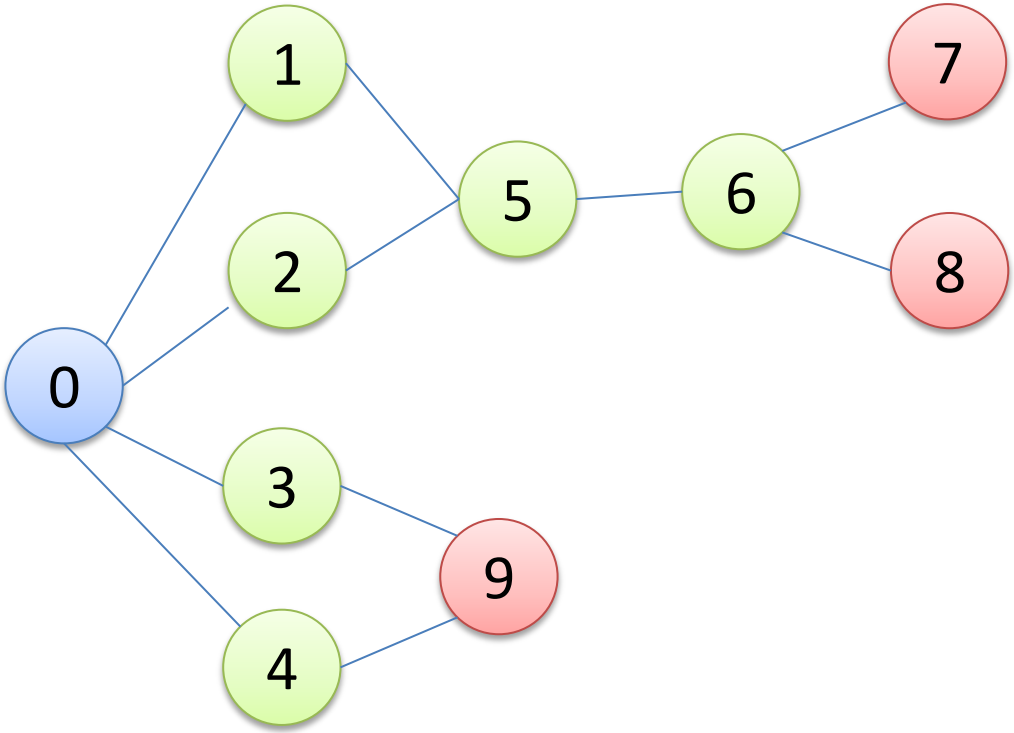
- 部分点解法のままだと・・・？
  - Eは最大4500まで
  - 2^4500は地球が爆発しても列挙できない。
  - 絶対に間に合わない！

- 何かもっと早いアルゴリズムを考えなくてはならない

D問題　完答　アルゴリズム

AtCoder

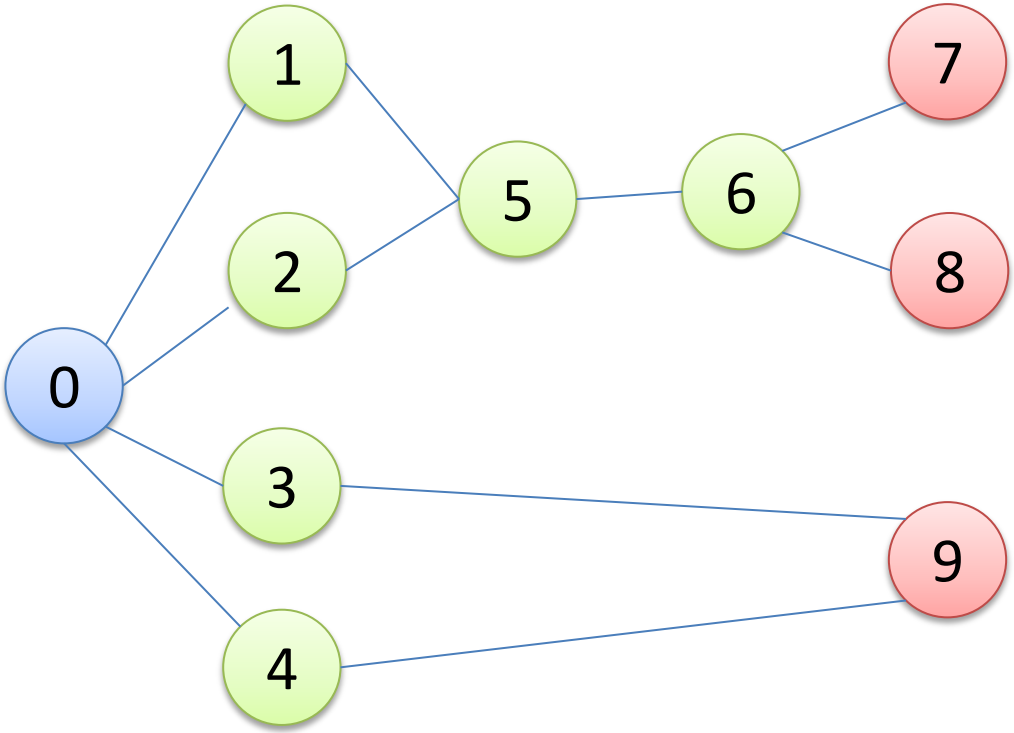
- 以下のような図を考える



D問題　完答　アルゴリズム

AtCoder

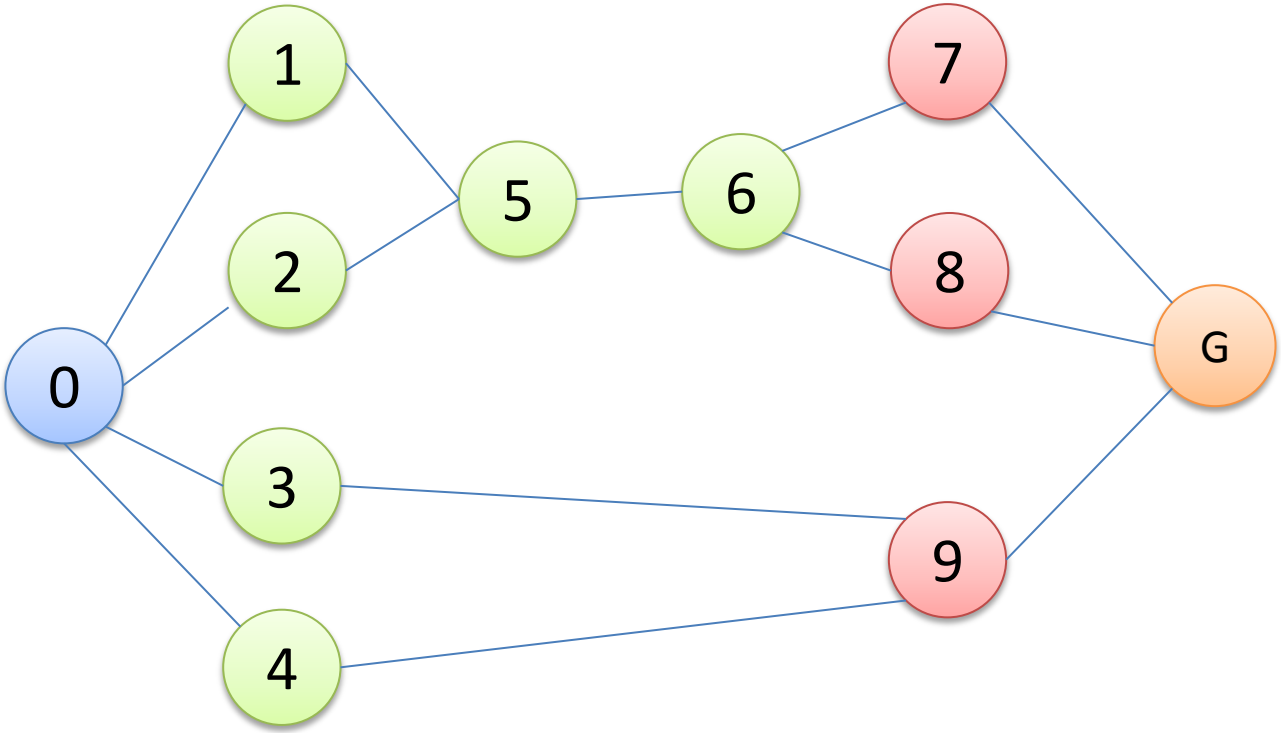
- 以下のような図を考える　ちょっとずらす
  - 工作の種類が2パターンあるのが面倒なので、これを纏めたい



D問題　完答　アルゴリズム

AtCoder

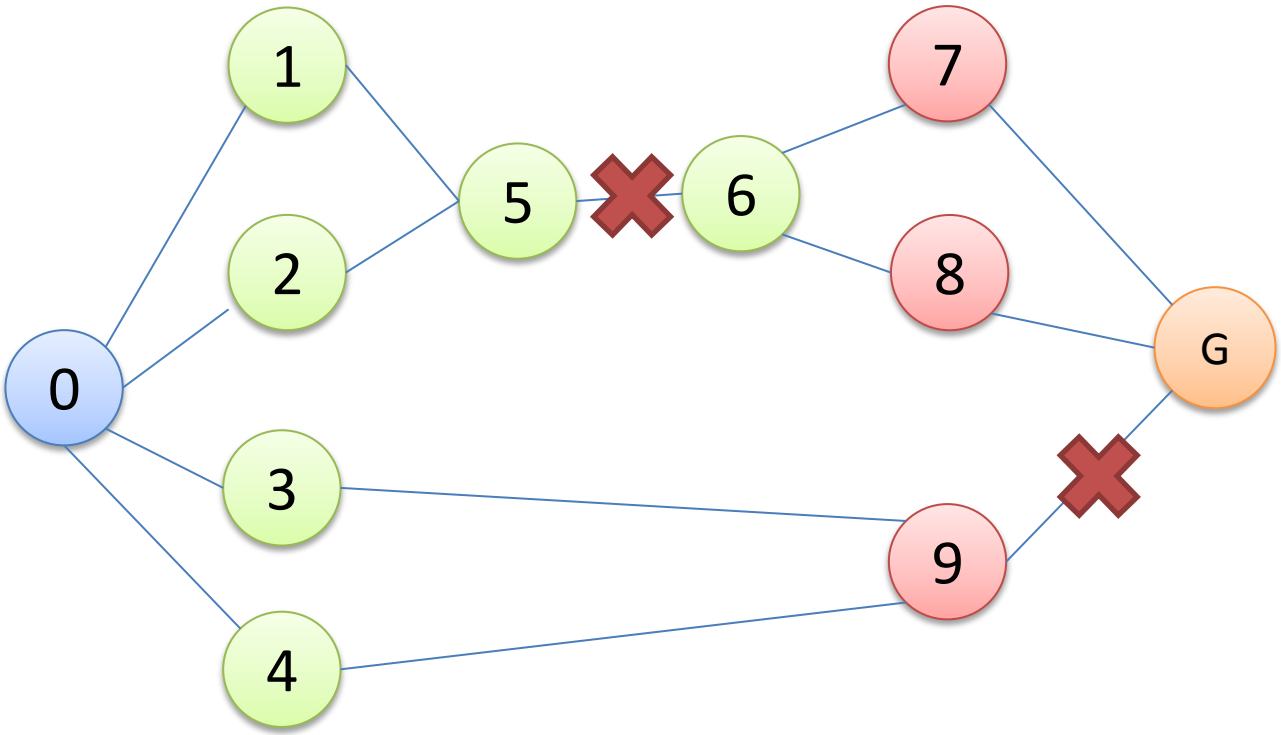
- 以下のような図を考える　ちょっとずらす
- 最後に、「メッセージをログインして閲覧する」という処理を追加する



D問題　完答　アルゴリズム

AtCoder

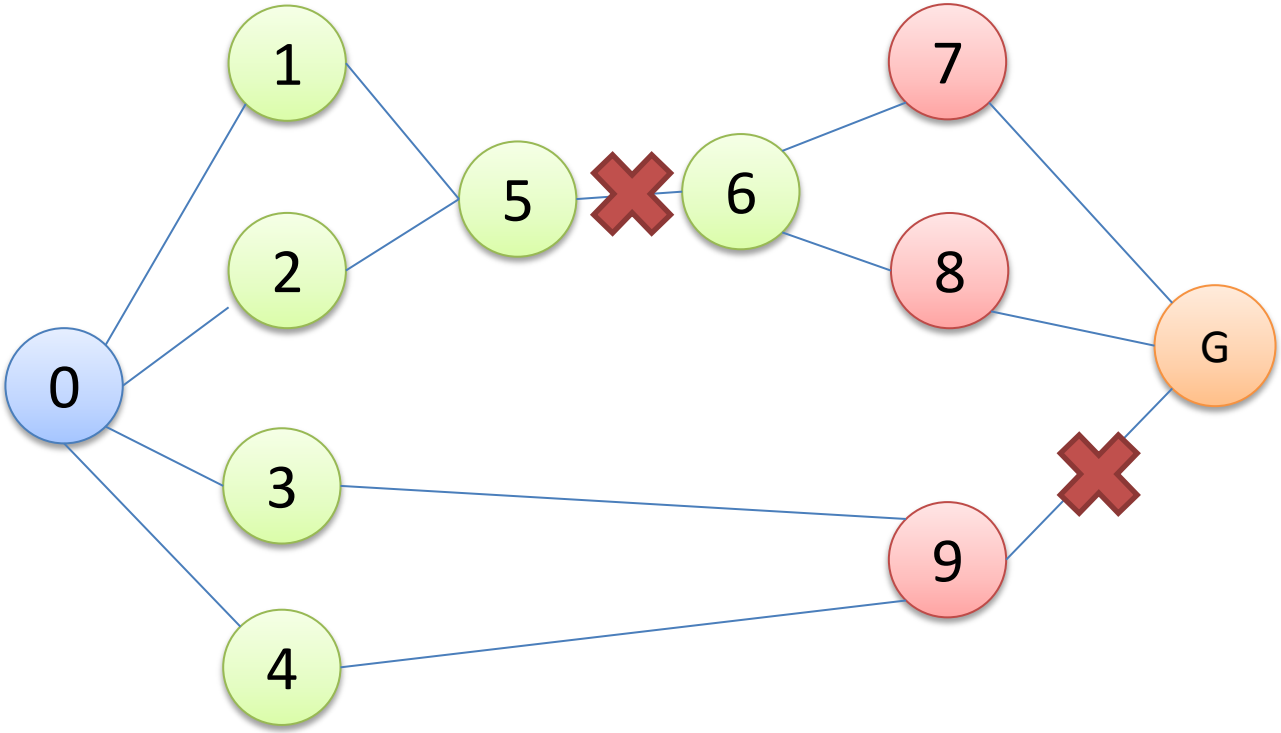
- この図に対し、0からGに行けなくなるように、線に×をつければ良い。



D問題　完答　アルゴリズム

AtCoder

- この図に対し、0からGに行けなくなるように、線に×をつければ良い。
  - このように、切断するための最小数を「最小カット」と言う



D問題　完答　アルゴリズム

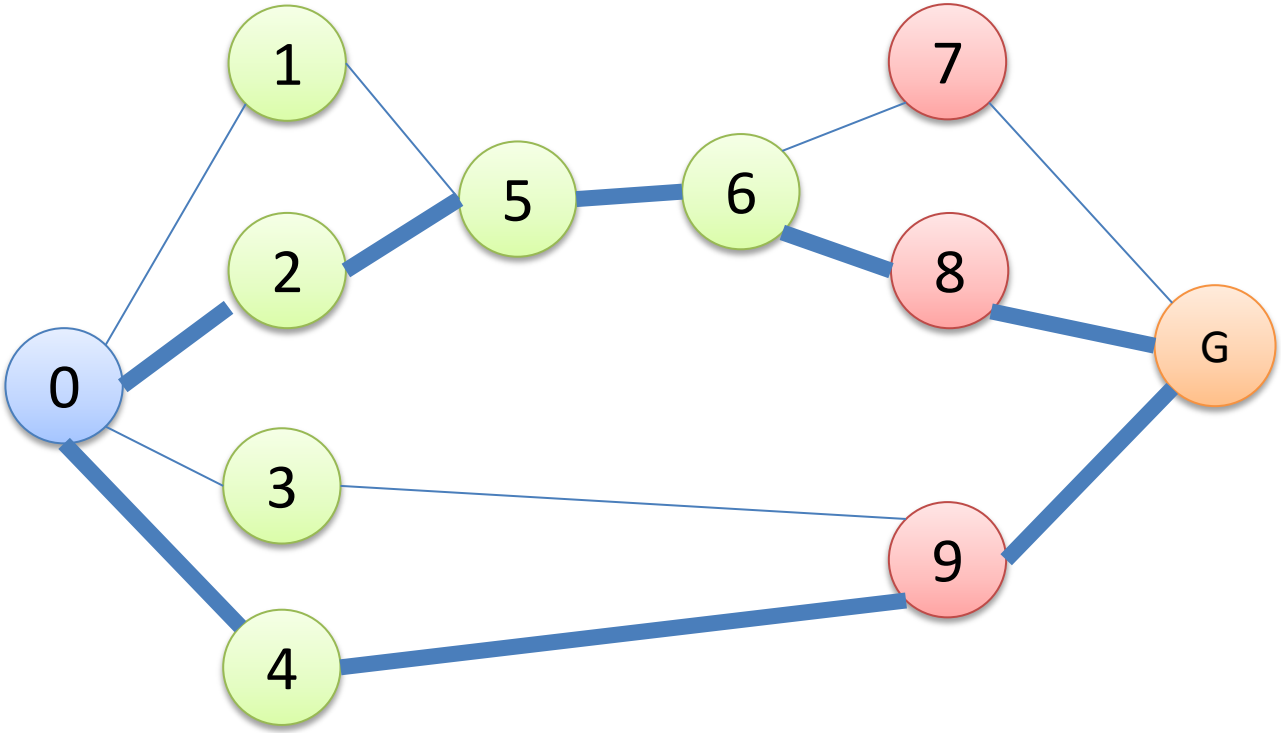
AtCoder

- 最小カットの求め方
  - 最小カット最大フロー定理を利用しよう！
    - グラフの最小カットは、最大フローと一致するよ！という定理です
  - じゃあ最大フローってなあに？

D問題　完答　アルゴリズム

AtCoder

- 最大フローって？
  - 0からGに辿り着くための、線が何本引けるか、という問題
    - 今回の場合は2本　これ以上引くことは出来ない。



D問題　完答　アルゴリズム

AtCoder

- 最大フローって？
  - 0からGに辿り着くための、線が何本引けるか、という問題
    - 今回の場合は2本　これ以上引くことは出来ない。
  - 今回の問題の場合は、全ての辺の容量が1だが、容量が1でない場合でも良い
    - つまり、同じ線に2本も3本も線を通しても良い、という制約でも良い
    - 良くわからなかったら気にしないでOKです。

D問題　完答　アルゴリズム

AtCoder

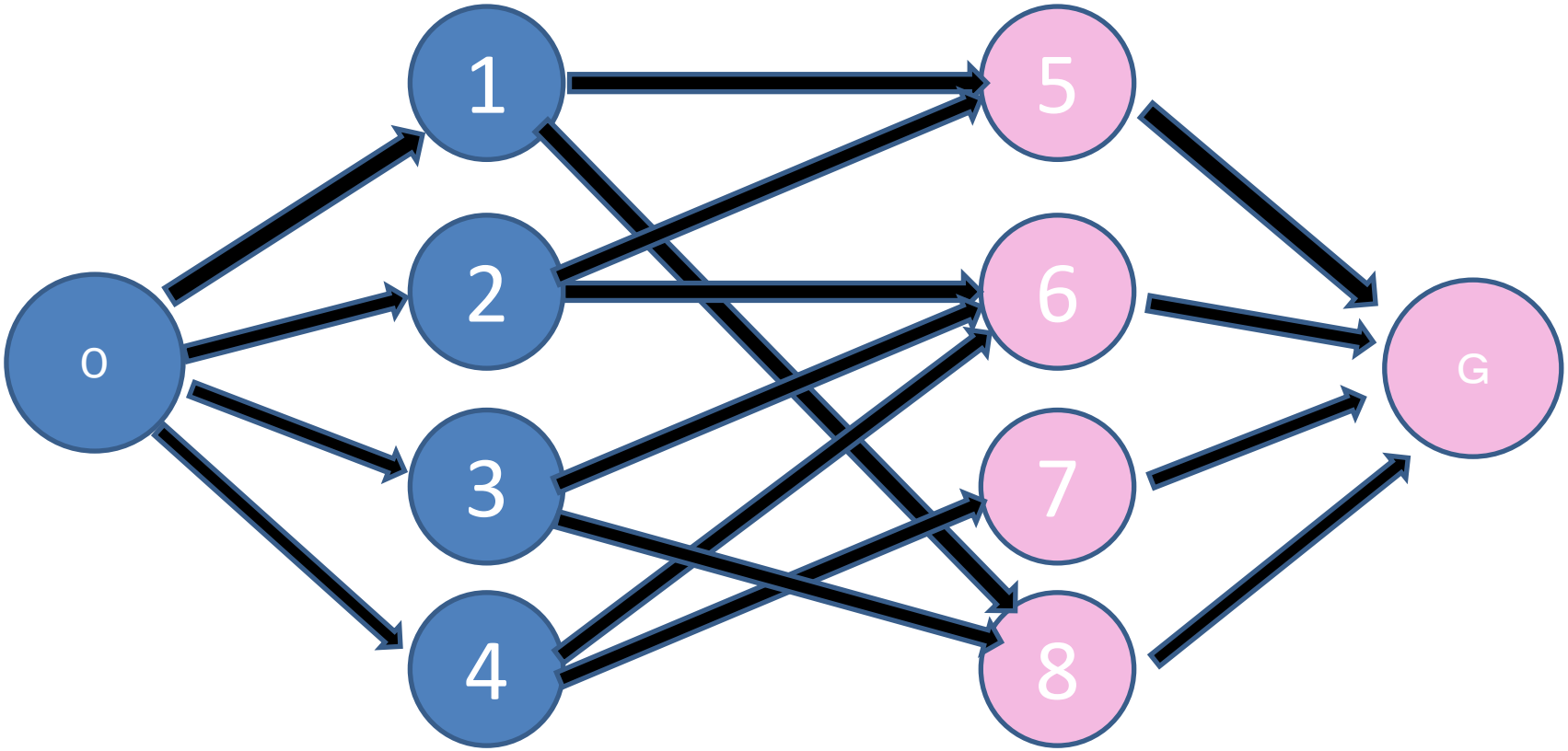
- 最大フローの求め方
  - 幅優先探索で、何回Gまでたどり着けるか計算しよう！！

    - ・・・・本当にそれでいいの？

D問題　完答　アルゴリズム

AtCoder

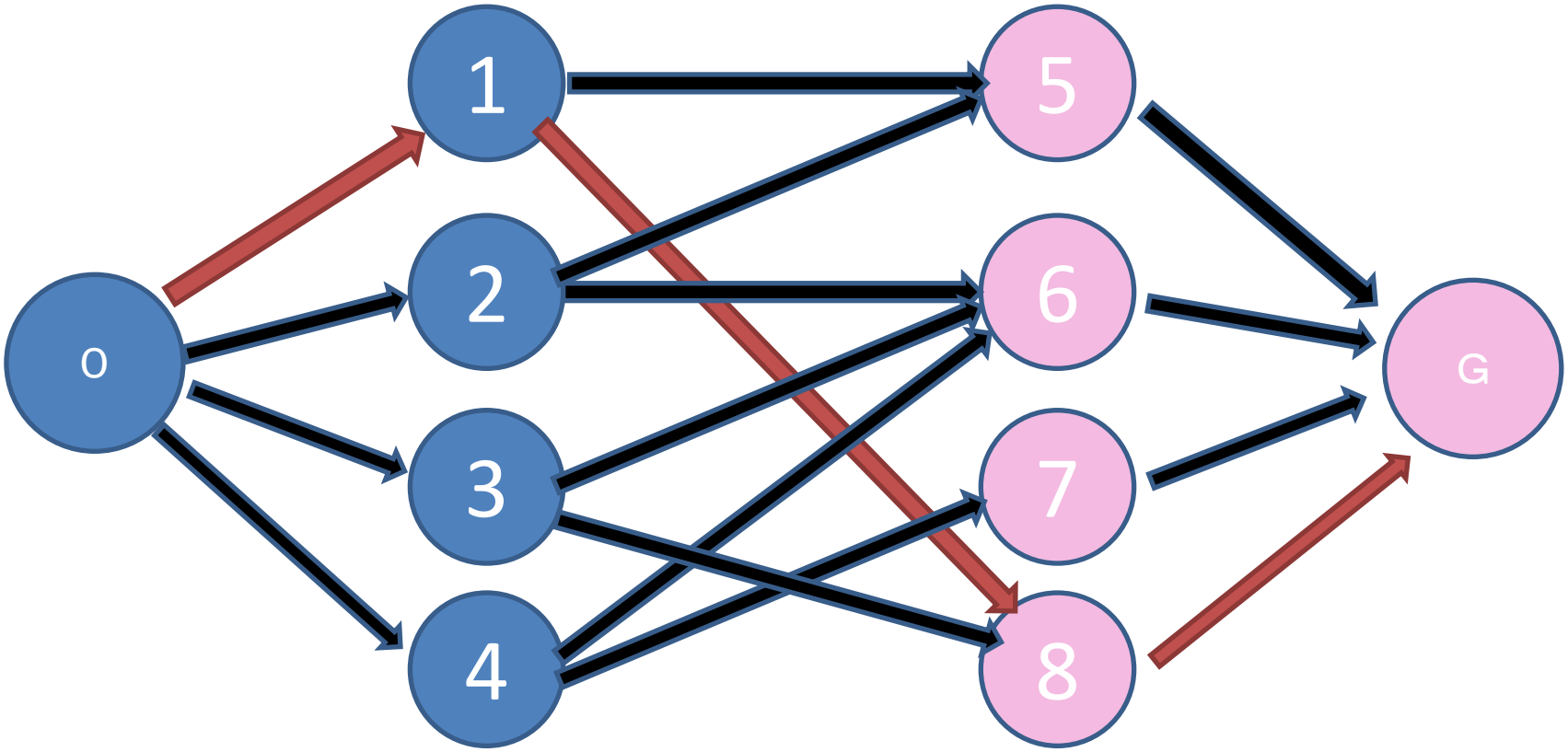
- 実際にやってみよう！
  - 矢印になってますが、今は気にしないでください。



D問題　完答　アルゴリズム

AtCoder

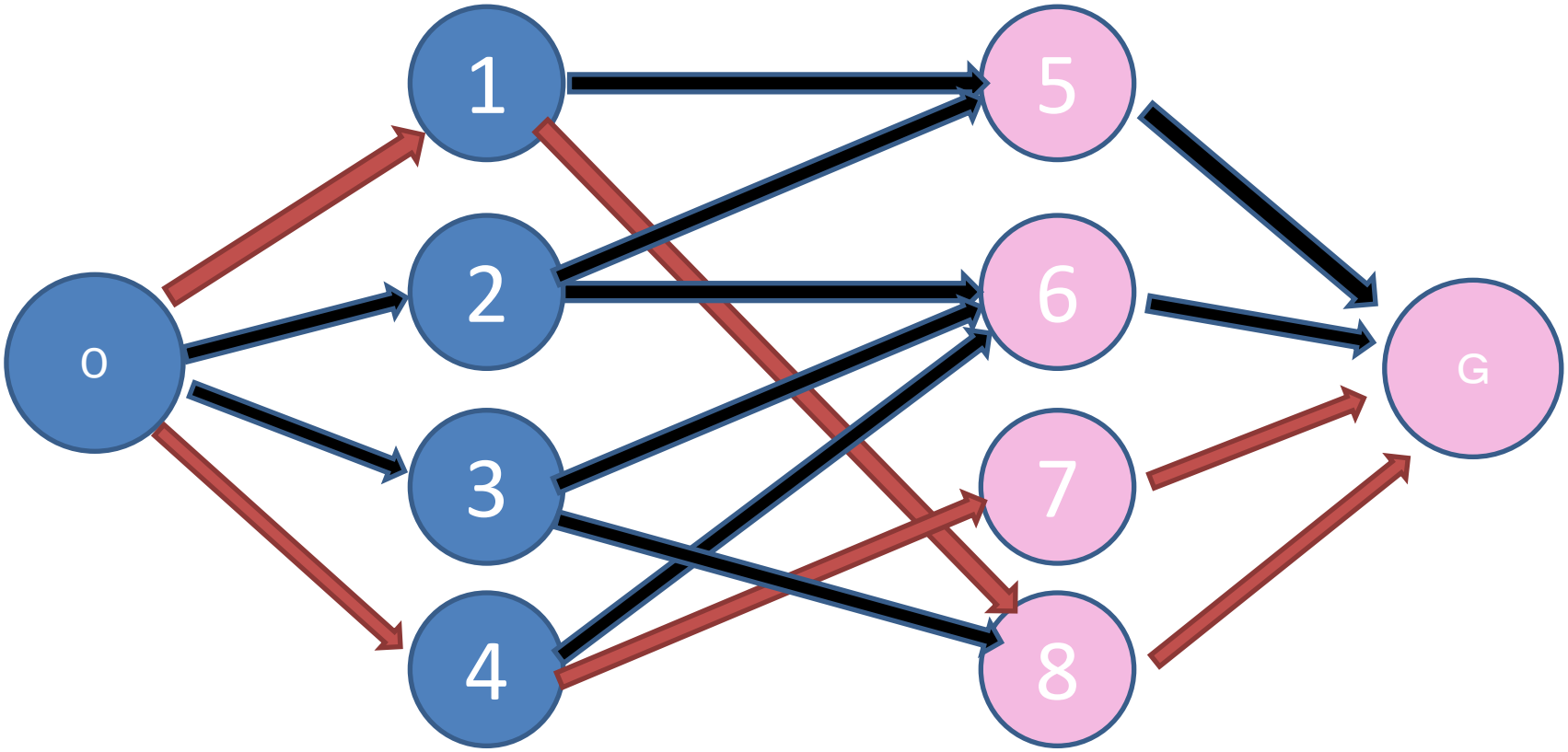
- 適当に1本ずつ引く



D問題　完答　アルゴリズム

AtCoder

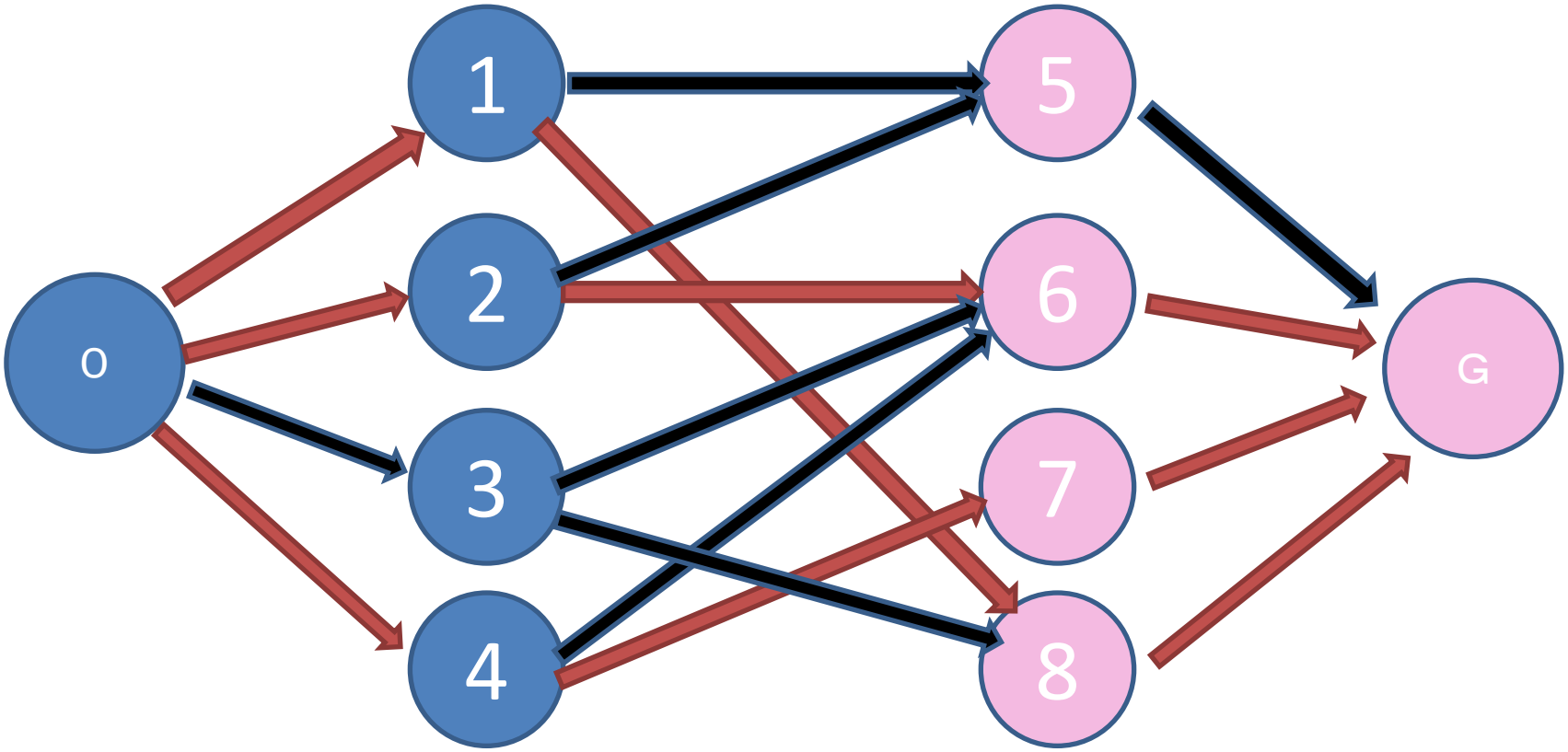
- 適当に1本ずつ引く



D問題　完答　アルゴリズム

AtCoder

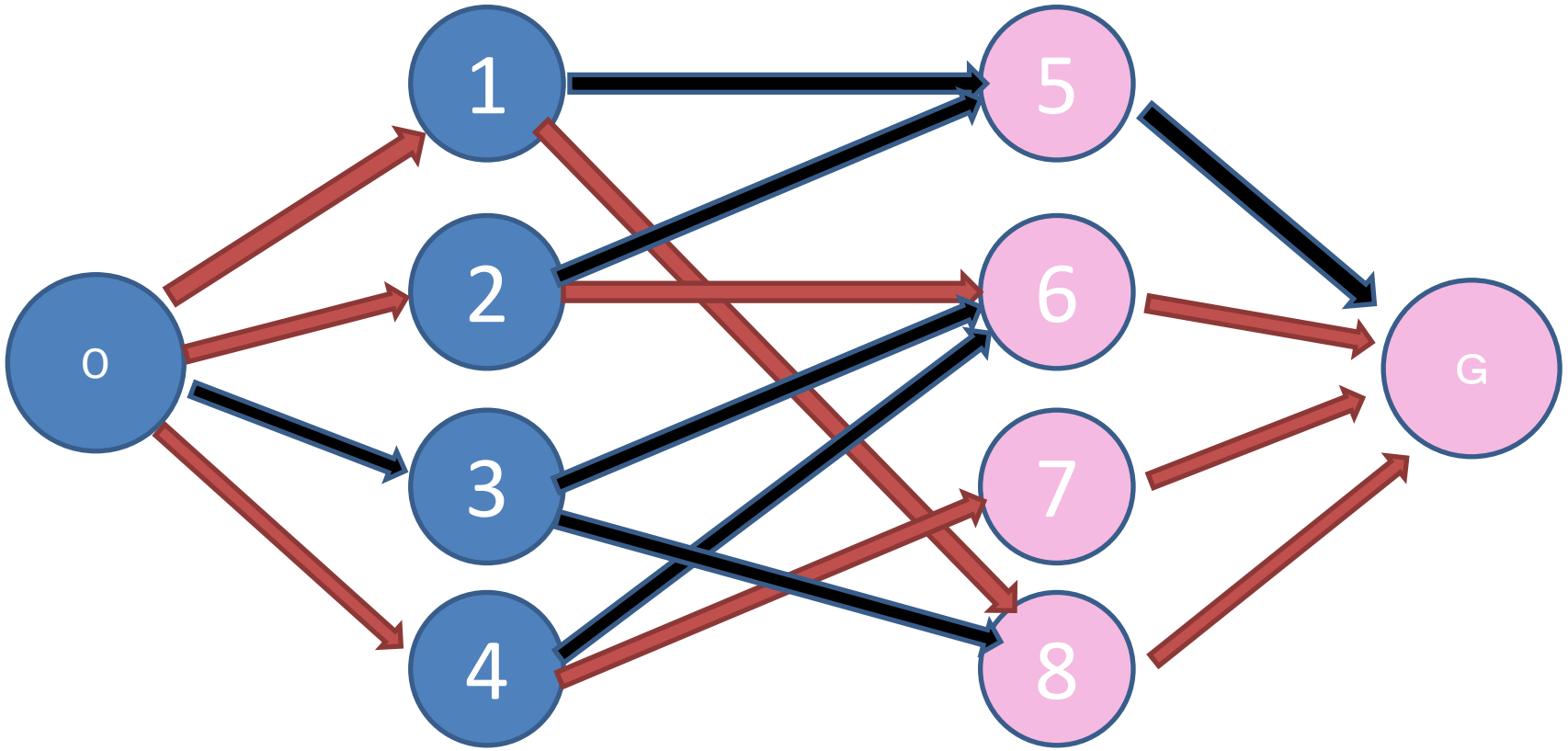
- 適当に1本ずつ引く



D問題　完答　アルゴリズム

AtCoder

- 適当に1本ずつ引く
  - もう引けないので答えは3?　　本当は4



D問題　完答　アルゴリズム

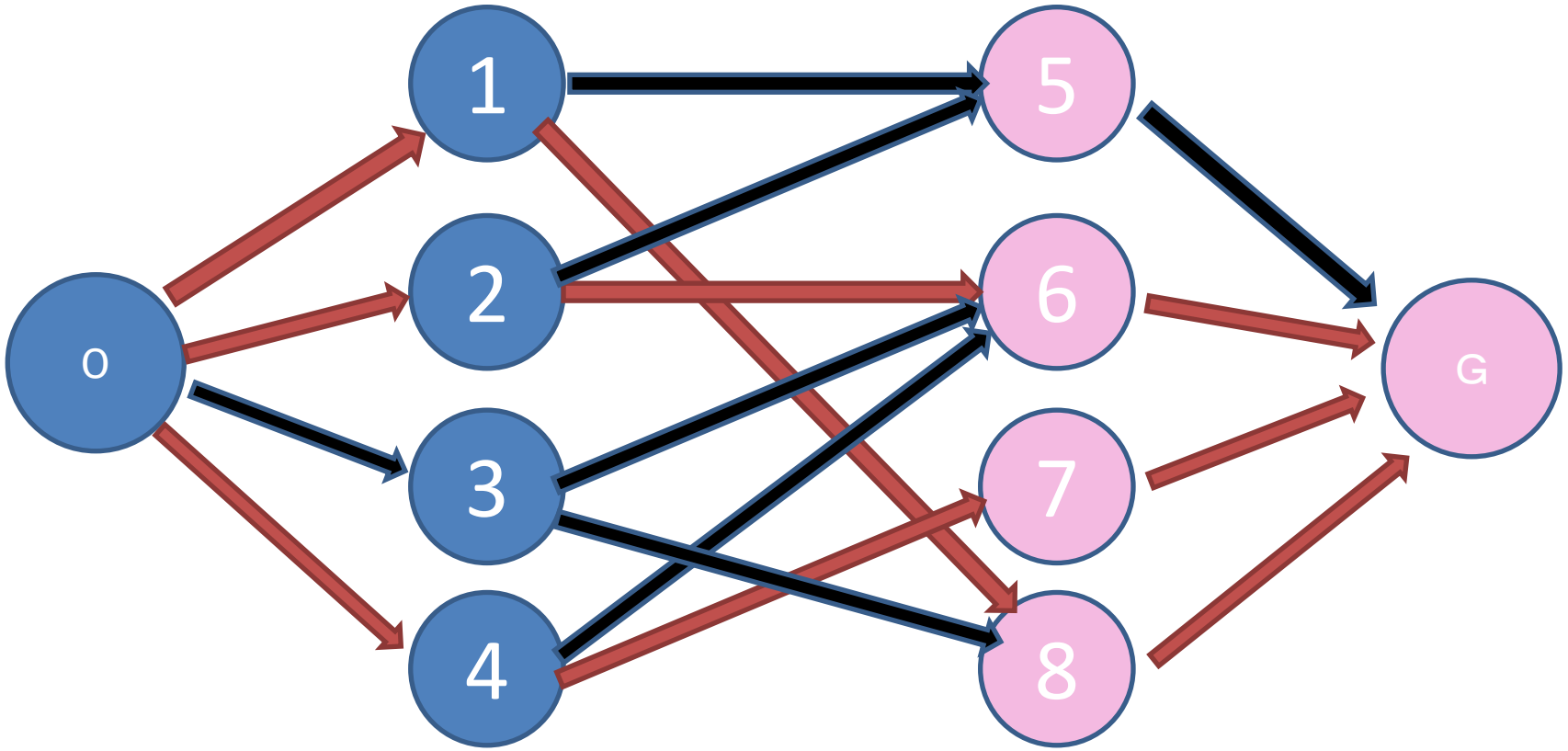
AtCoder

- 普通に幅優先探索してもダメ
  - 使う辺の順番によって、正しくない解になってしまう。
  - ここで、特別な処理をしてあげることによって、正しい解を出すことが出来る！

D問題　完答　アルゴリズム

AtCoder

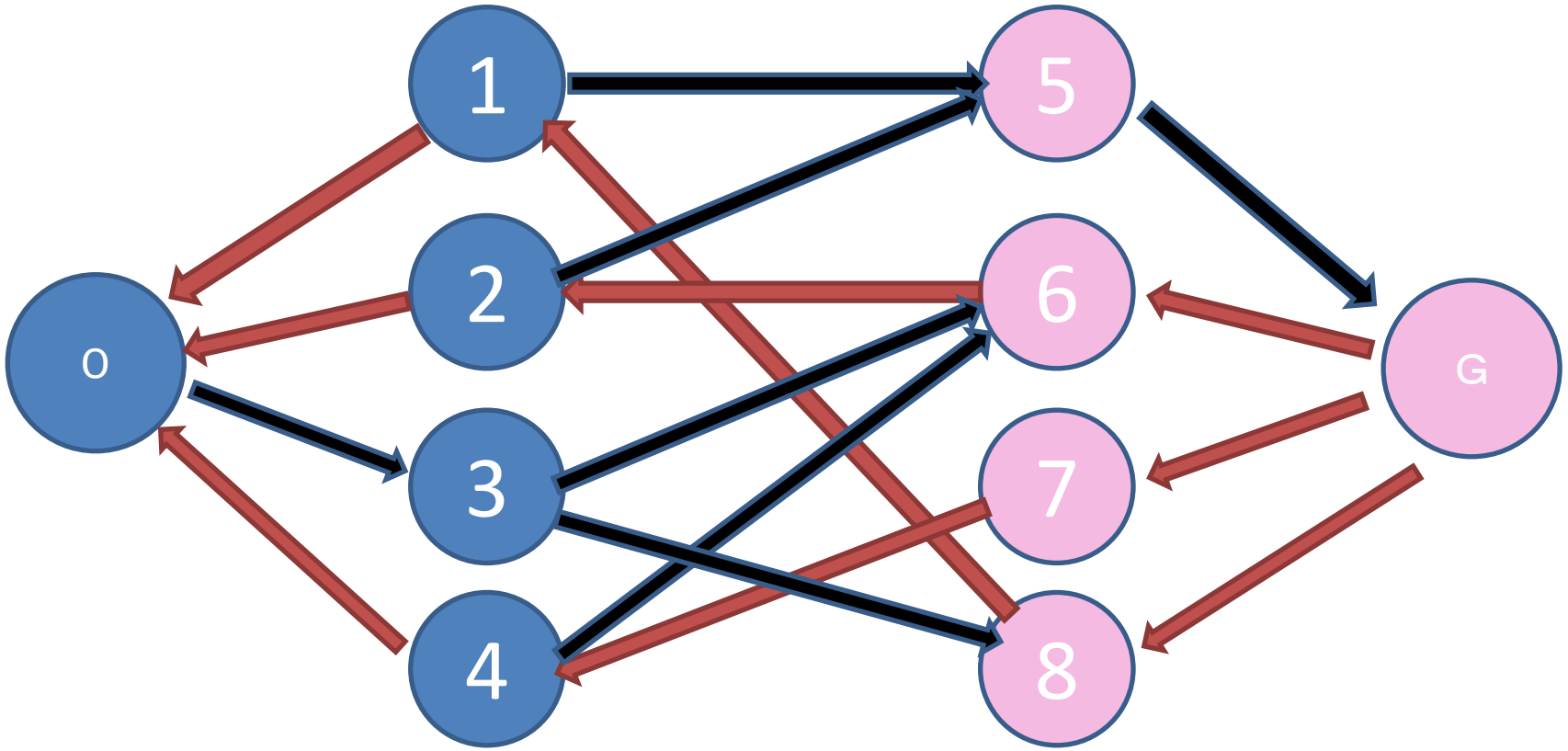
- 今までは、普通に矢印にフラグを付けるだけだった。



D問題　完答　アルゴリズム

AtCoder

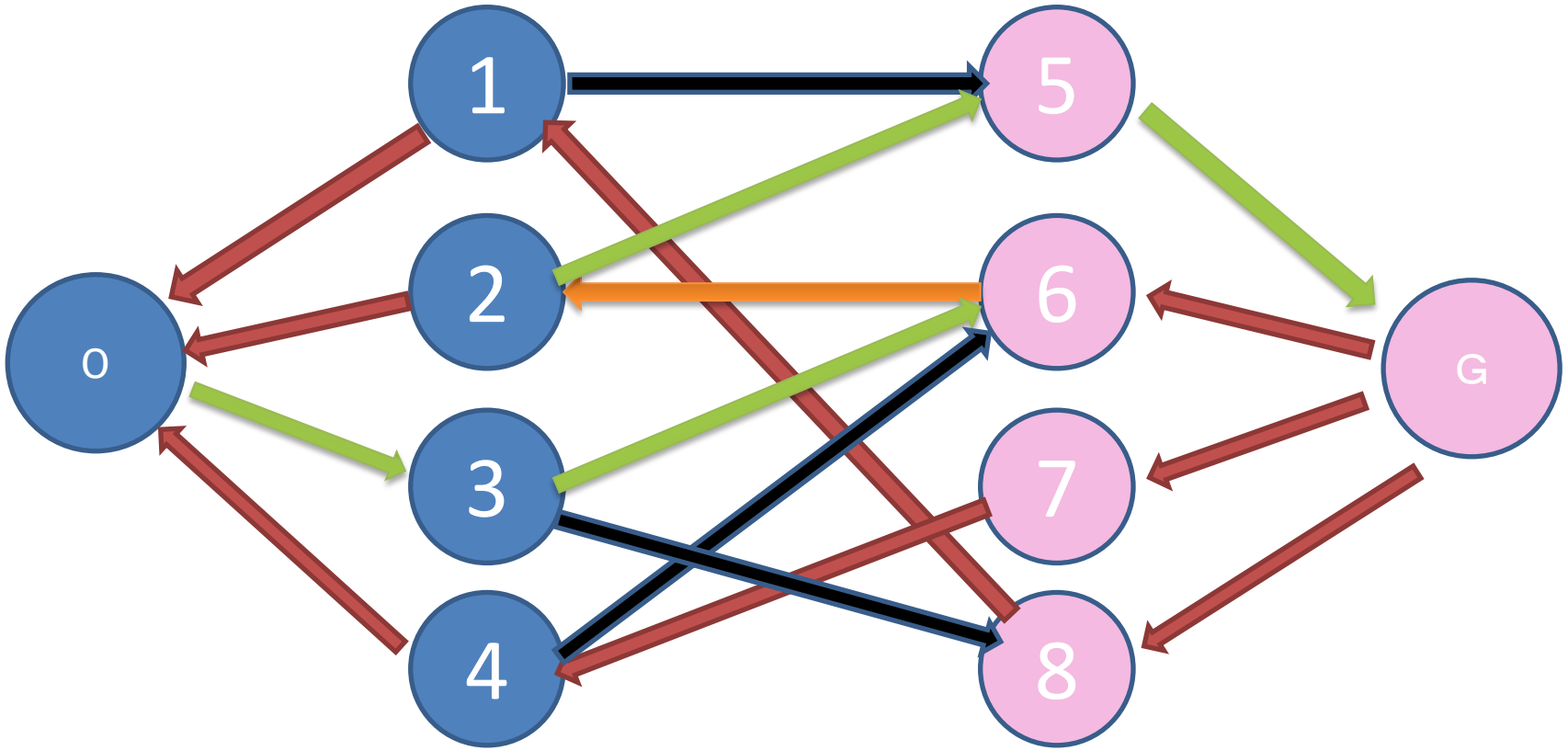
- 今までは、普通に矢印にフラグを付けるだけだった。
  - 今まで通ったところの矢印の向きを変えてみよう！！！



D問題　完答　アルゴリズム

AtCoder

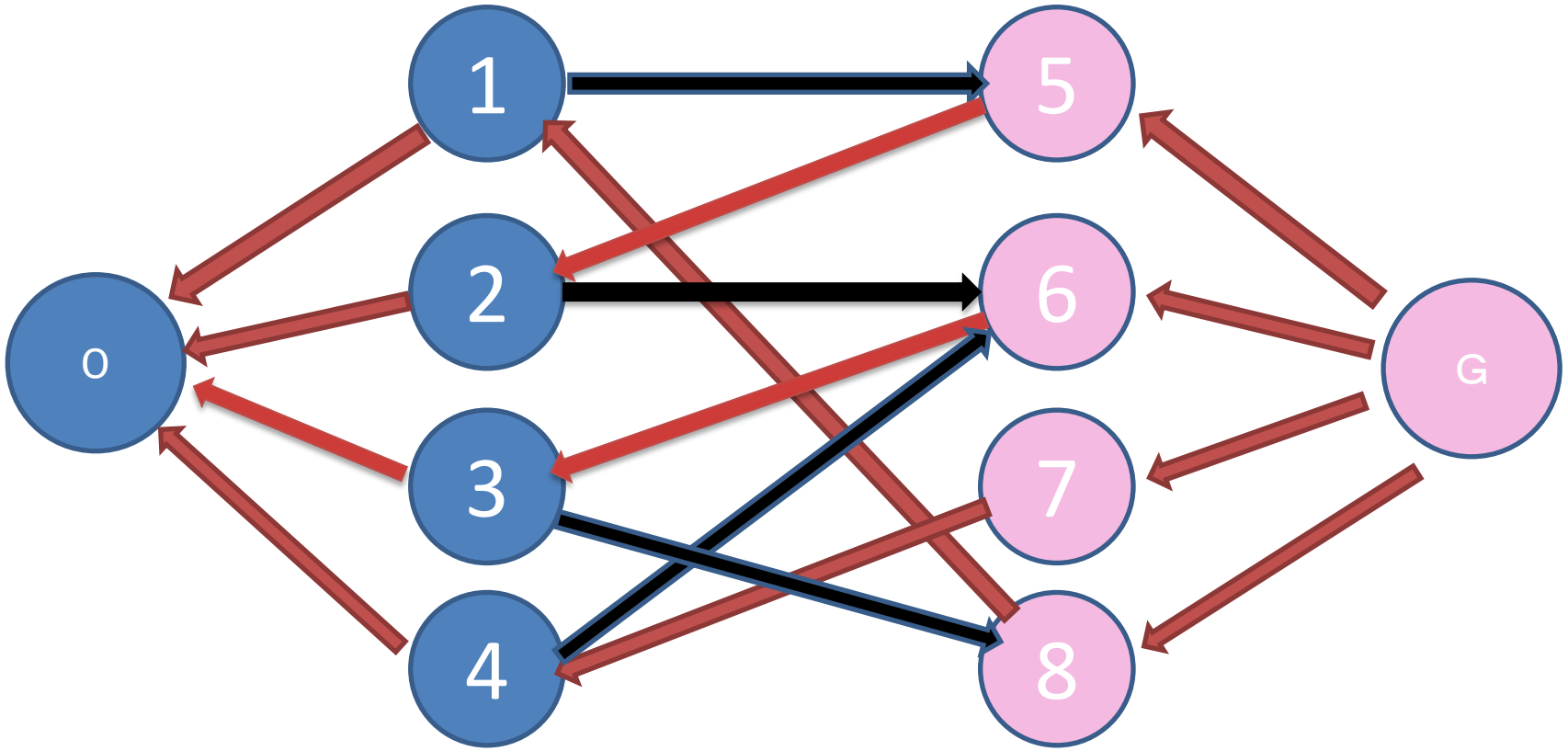
- 今まで通ったところの矢印の向きを変えてみる
  - 新しいルートが出来た！



D問題　完答　アルゴリズム

AtCoder

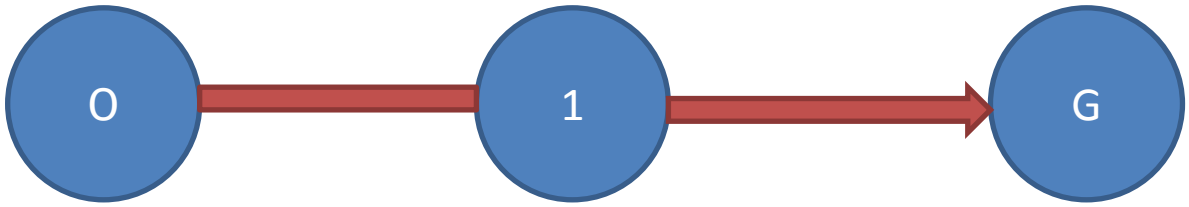
- これで、正しい答えを求めることが出来る！



D問題　完答　アルゴリズム

AtCoder

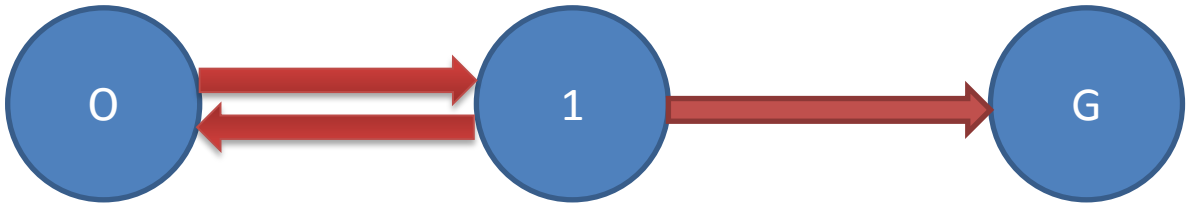
- 注意点
  - 今回の問題は、矢印じゃなくて、両方向に繋がっている



D問題　完答　アルゴリズム

AtCoder

- 注意点
  - 今回の問題は、矢印じゃなくて、両方向に繋がっている
  - であれば、矢印2つに変換しちゃおう！
    - これで先ほどのアルゴリズムが問題なく使えます。



D問題　完答　アルゴリズム

AtCoder

- 注意点
  - 実装が出来ない！という方へ
    - 今回のアルゴリズムは、かなり実装が難しいです。
    - 他の人のソースコードは、最大フローを求める色々なアルゴリズムを使っている場合があります。
      - Edmonds-Karp
        - » 先ほど説明した幅優先探索で求めるアルゴリズム
      - Dinic
        - » 幅優先探索と深さ優先探索を組み合わせるアルゴリズム
      - Goldberg-Tarjan
        - » ヒューリスティックでなんか早くなるアルゴリズム
    - 個別のアルゴリズムに興味があれば、本や参考サイトで調べることをお勧めします。