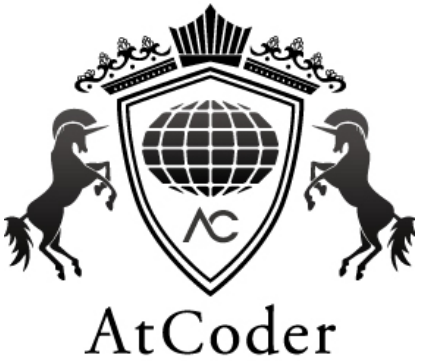
AtCoder Beginner Contest 012
解説



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　スワップ

1. 問題概要
2. アルゴリズム

A問題　問題概要

AtCoder

- 整数A, Bが与えられる
- AとBの中身を入れ替える
- A, Bを出力しなさい。

- 制約
- $1 \leqq A, B \leqq 100$

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、A, B という2つの整数
  - 問題で与えられた処理を行う
    - 今回は、AとBの中身を入れ替える
  - 標準出力へ、答えを出力する

A問題　アルゴリズム

AtCoder

- 入力
  - 2つの整数を、標準入力から受け取る
    - Cであれば、scanf(“%d %d”, &A, &B);　など
    - C++であれば、cin >> A >> B;
    - 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

- 処理部分
  - 今回は、AとBを入れ替える！
  - 入れ替え方はいくつか存在する
    - 標準ライブラリとかのswapを使う
      - Swap(A, B)みたいな感じ
      - 言語によってあったりなかったり
    - 普通に入れ替える
      - Int C = A; A = B; B = C;
      - 1回別の変数に数字を避難させておくことで、簡単にできる！
    - かっこよく入れ替える
      - A ^= B; B ^= A; A ^= B;
      - XOR交換アルゴリズムと呼ばれるテクニック
        - » 無駄にかっこいいけど競技プログラミングで役に立つことは少ない

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(“%d %d¥n”, A, B); (C)
    - cout << A << “ “ << B << endl; (C++)
    - System.out.println(A + “ “ + B); (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

A問題　アルゴリズム

AtCoder

- おまけ
  - AとBを入れ替える、と問題文に書かれている
  - しかし、AとBを入れ替える必要はない！
  - B, Aの順番で出力すればいいだけ
    - 問題文で「〇〇しろ」と書かれているからといって、必ずしもそれに従う必要はない
    - 「こうしたら同じことがもっと簡単に出来るのではないか」というのを常に考えるようにしよう！

B問題　入浴時間

1. 問題概要
2. アルゴリズム

B問題　問題概要

AtCoder

- 秒数Nが与えられる。
- HH:MM:SSのフォーマットに変換しなさい。

- 制約
- $0 \leqq N \leqq 86399$

B問題　アルゴリズム

AtCoder

- 入力
  - 整数Nを受け取る
    - 解らない場合は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
  - 秒数から、時間、分、秒のフォーマットに変換する
  - やり方は複数存在する
    - 各言語の標準ライブラリについている、時刻への変換機能を使う
      - 高級な言語であれば大体ついてる
        - » C# だとDateTimeとか
    - 時間、分、秒をそれぞれ計算して求める。
      - 時間 →　N / 3600
      - 分 →　(N / 60) % 60
      - 秒 → N % 60
      - フォーマットを合わせるのも自分でやる必要あり

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(S)みたいな感じ
  - H, M, Sを自力求めた場合はちょっと大変？
    - Print(H + “:” + M + “:” + S) みたいな感じ？
      - これだと、たとえば0:13:8のように、二ケタ表示にならない。
    - これの解決法も二つに分かれる
      - 標準ライブラリを使う
        - » 言語ごとに違うので調べてみよう！
      - 自力でフォーマットをそろえる
        - » H, M, Sをそれぞれ文字列にする
        - » 各文字列が2文字以下だったら、2文字になるまで先頭に0をつける
        - » あとは:でつなぐだけ。

C問題　九九足し算

1. 問題概要
2. アルゴリズム

C問題　問題概要

AtCoder

- 九九に出てくる数字を全部足したら、答えがNに
- 一つ足し忘れたことが解っている
- 足し忘れたのはどの演算かを答えなさい

- 制約
  - 1944 ≦ N≦ 2024
    - この制約が書いてあることで、実は実装量を大幅に減らせる

C問題　アルゴリズム

AtCoder

- まずは、「足し忘れて足りない数字がいくつか」を考える必要がある
  - 正しい答えをMとして、M-Nが足りない数字である。
    - これをPとする
- Mの求め方はいくつか存在する
  - 直接計算して求める
    - 二重ループを回して足し算する
  - 制約条件から取ってくる！
    - Nの最大値に1を足した2025が正しい答えであることが解る
      - ずるい
      - 制約を広めに書いている可能性があるので、最小値と最大値が80であり、これが9*9 – 1*1であることに気付いた時のみ使って良い
  - 45×45を計算する！

C問題　アルゴリズム

AtCoder

- 足りない数字Pが求まったら、P = A*Bとなるような A, Bの組み合わせを探す
- 探し方はいくつかある
  - Aを1～9まで試してみる
    - B = P / Aで求めることが出来る
  - Pの約数を全列挙する
    - 結局ループを回すので、個のアルゴリズムより明らかに劣る
  - A,Bを2重ループで全て試す
  - 素直に全探索を行うのが良い！

D問題　バスと避けられない運命

1. 問題概要
2. アルゴリズム

D問題　問題概要

AtCoder

- バスの路線図が与えられる
- 一番移動に時間がかかってしまうバス停のペアの、移動にかかる時間を出力しなさい

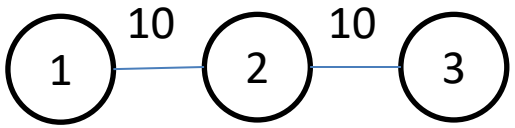
- 制約
  - 1≦N≦ 300

D問題　アルゴリズム

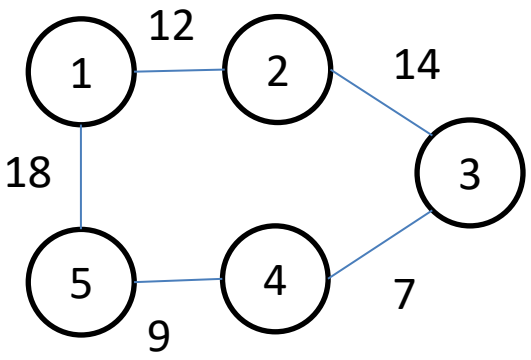
AtCoder

- 今回の問題は、グラフにして考えると良い！
  - グラフって？
    - 丸（頂点）と、線（辺）で、様々な状態を表したもの！
  - 例えばどう表すの？
    - 下のような感じ

入力例1



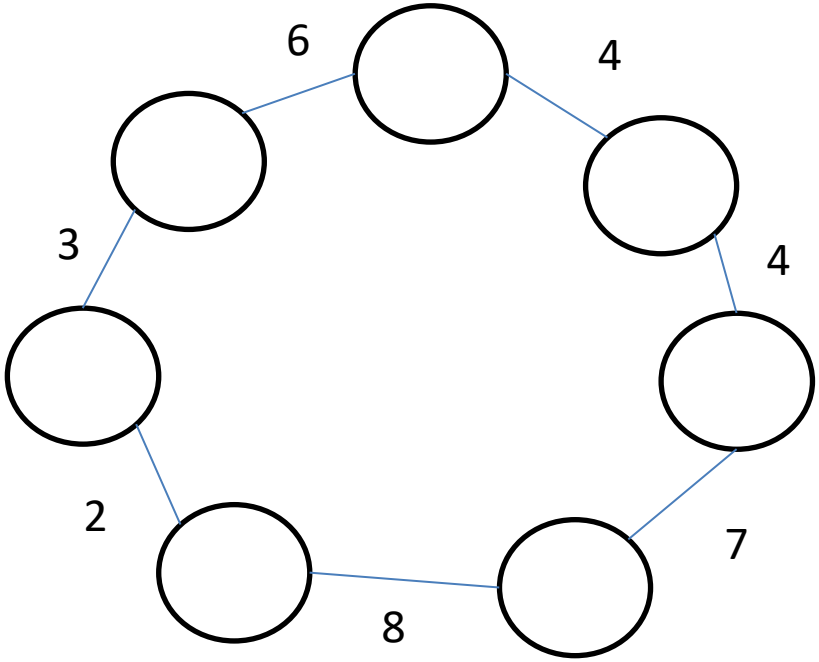
入力例2



D問題　アルゴリズム

AtCoder

- 最も良いバス停を探すには？
  - 上手い方法を探すのは難しい！
    - 以下のような例でも既にパッと見ではわからない

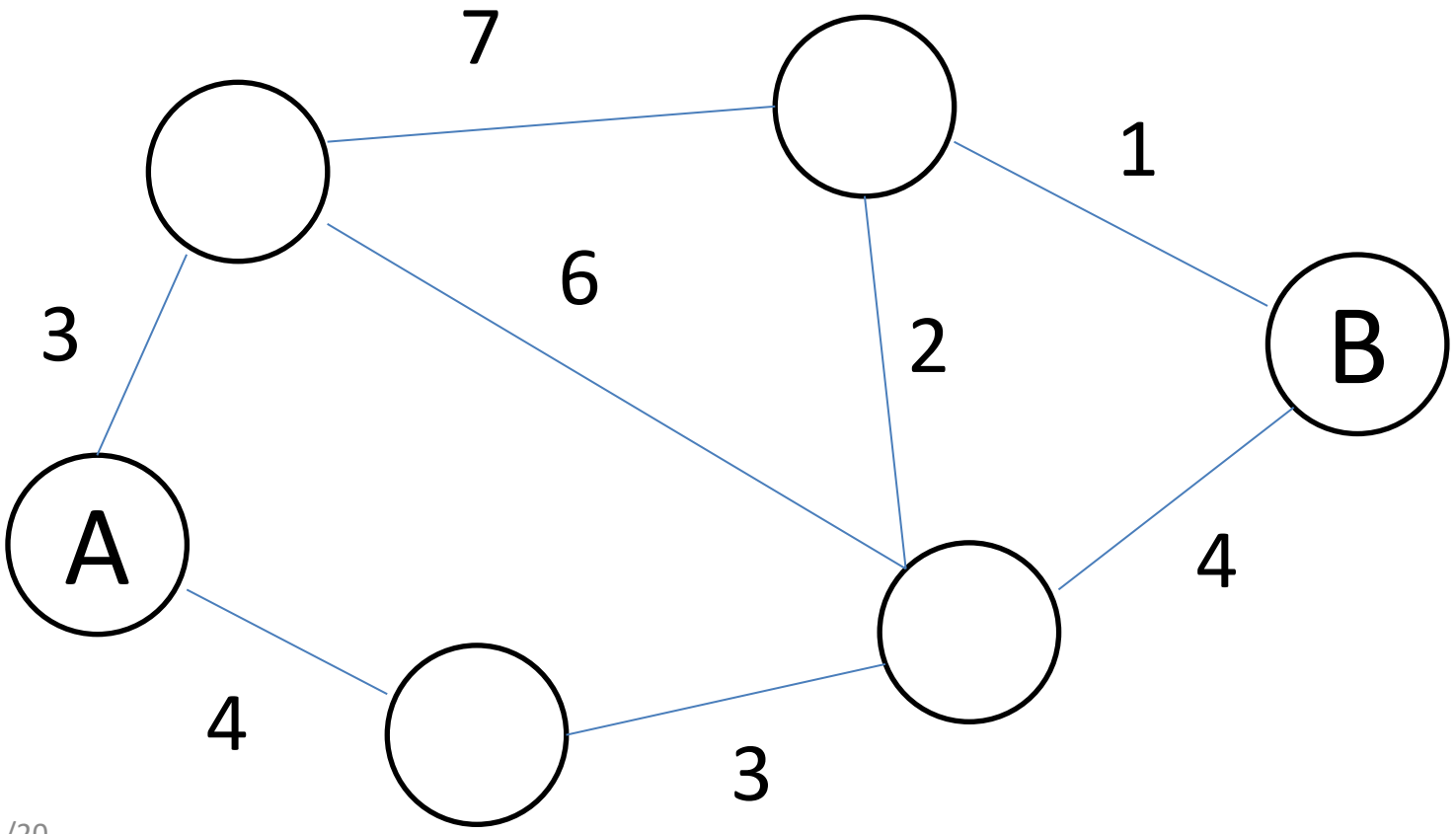


D問題　アルゴリズム

AtCoder

- 最も良いバス停を探すには？
  - 上手い方法を探すのは難しい！
  - ではどうするか？
  - 全てのバス停とバス停の間の距離を求めてしまえば良い！
    - でも、それって計算時間的に出来るの？

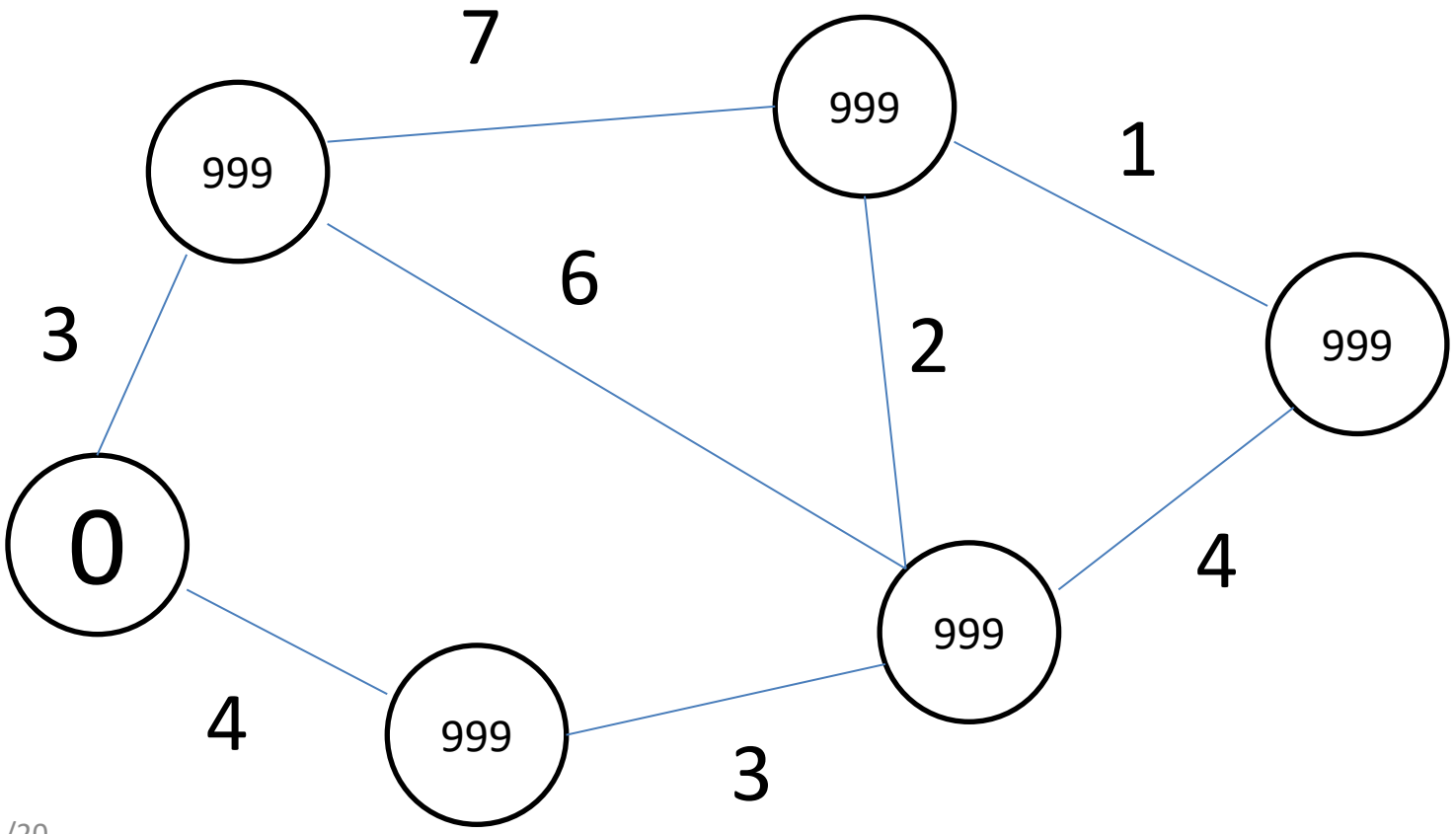
- 最短経路の求め方（ダイクストラ法）
  - AからBへの最短経路を求める方法



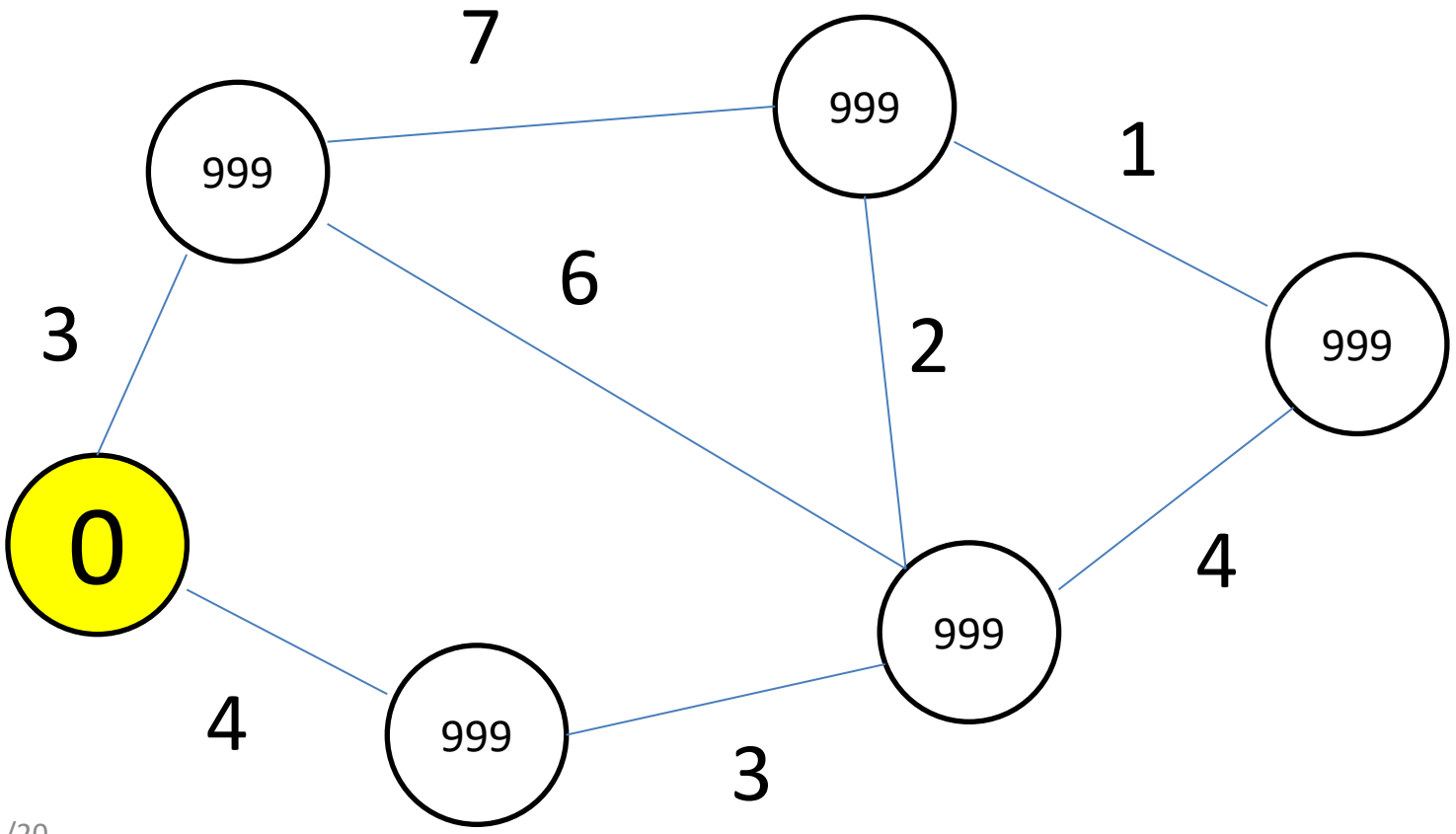
D問題　アルゴリズム

AtCoder

- 最短経路の求め方（ダイクストラ法）
  - AからBへの最短経路を求める方法
    - まず始点に0を入れ、他の場所には大きな数を入れる



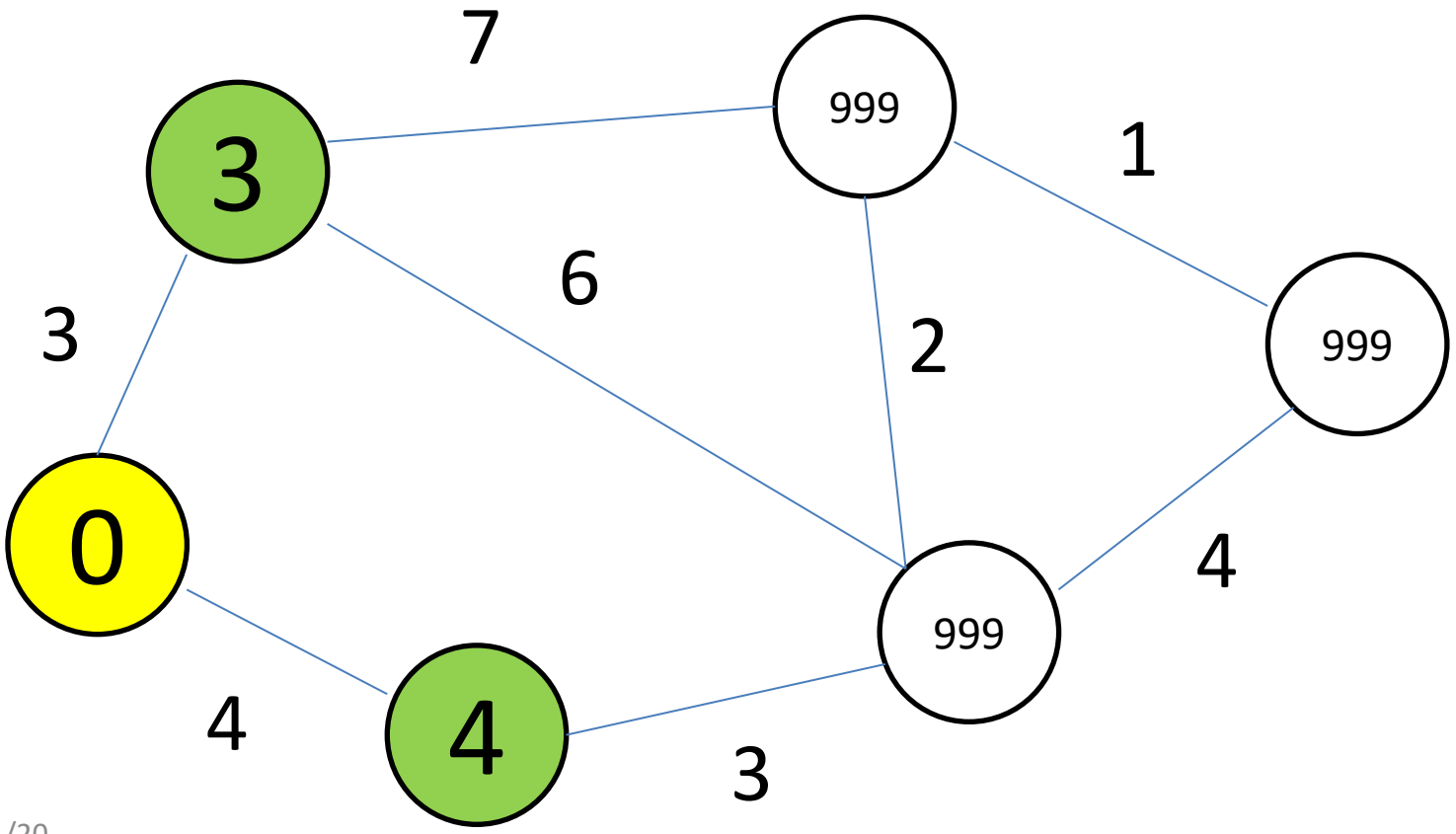
- 最短経路の求め方（ダイクストラ法）
  - AからBへの最短経路を求める方法
    - 各頂点のうち、最も数字の小さい頂点を選ぶ



D問題　アルゴリズム

AtCoder

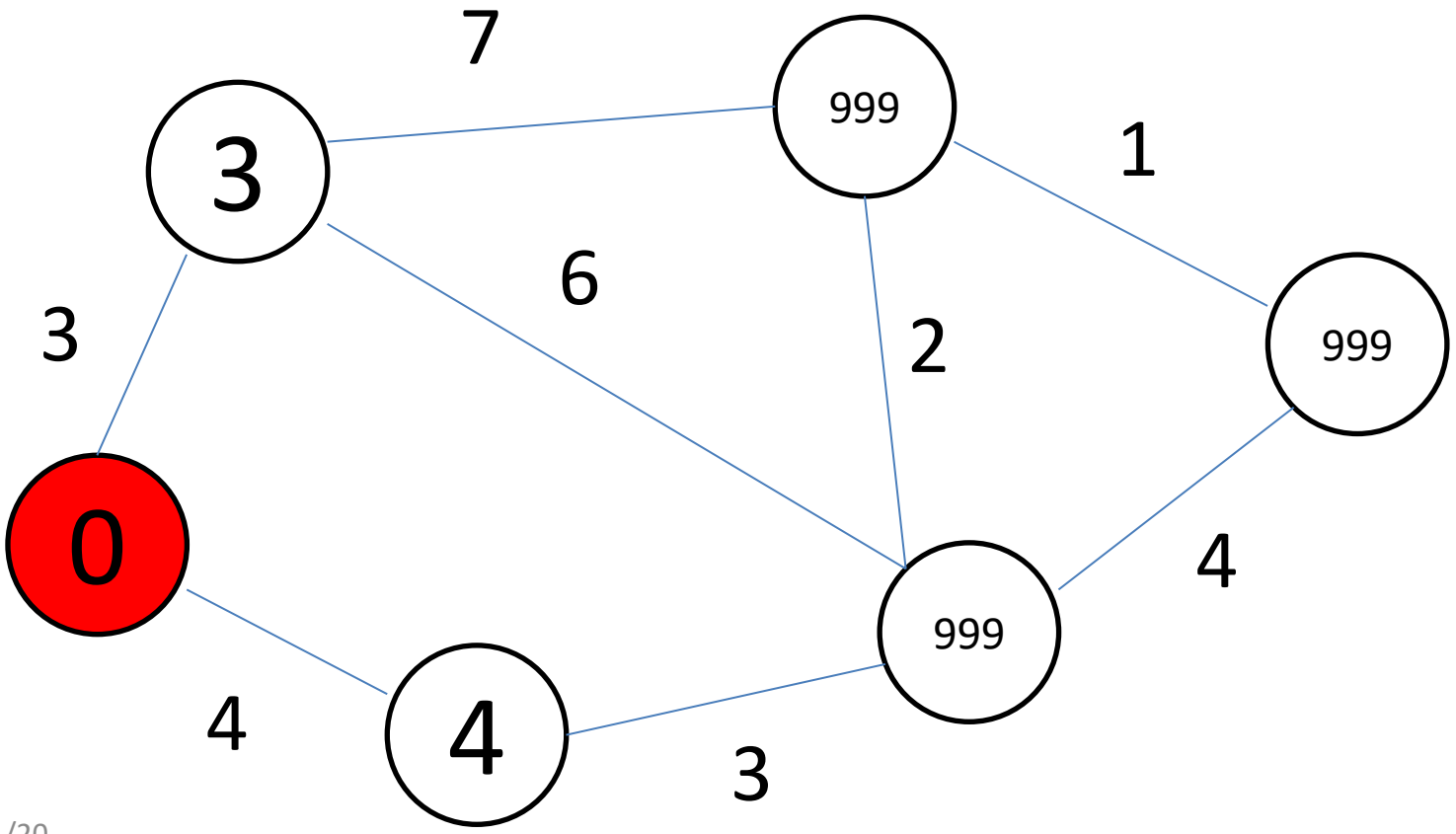
- 最短経路の求め方（ダイクストラ法）
  - AからBへの最短経路を求める方法
    - そこから、周りの値を更新していく



D問題　アルゴリズム

AtCoder

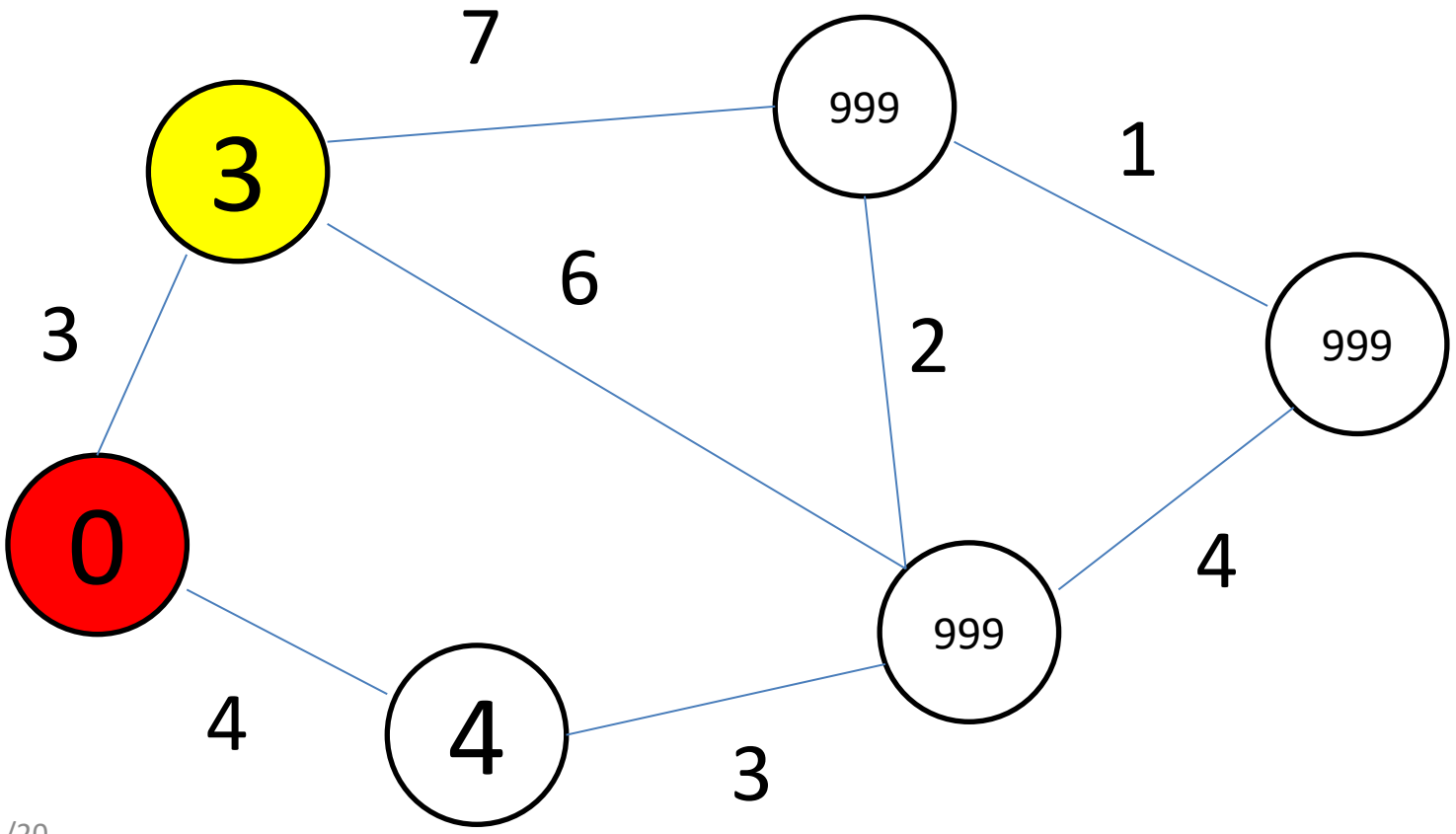
- 最短経路の求め方（ダイクストラ法）
  - AからBへの最短経路を求める方法
    - 今調べた頂点を使用済みとする



D問題　アルゴリズム

AtCoder

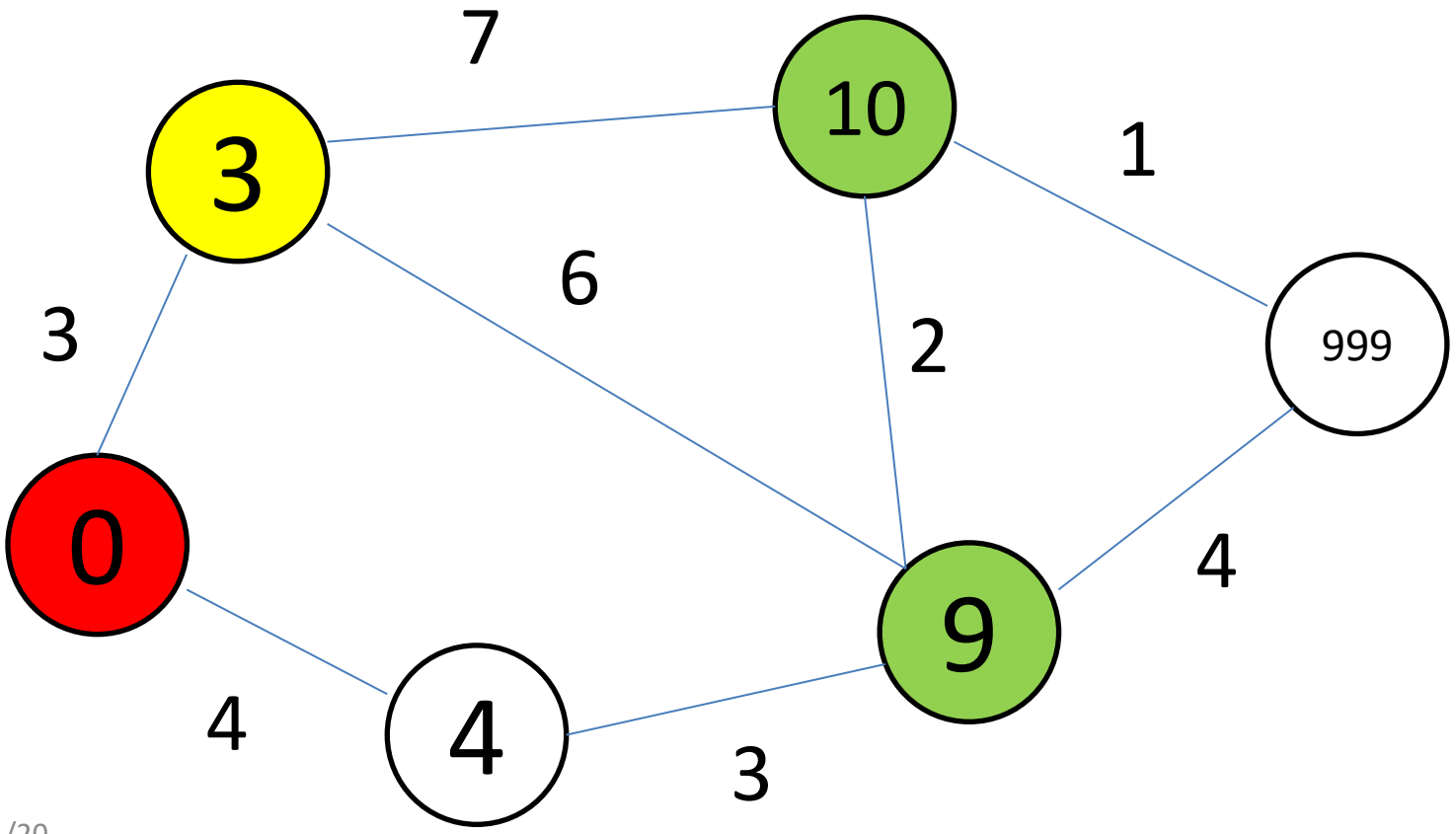
- 最短経路の求め方（ダイクストラ法）
  - AからBへの最短経路を求める方法
    - 再度、残った頂点から、最も数字の小さい頂点を選ぶ



D問題　アルゴリズム

AtCoder

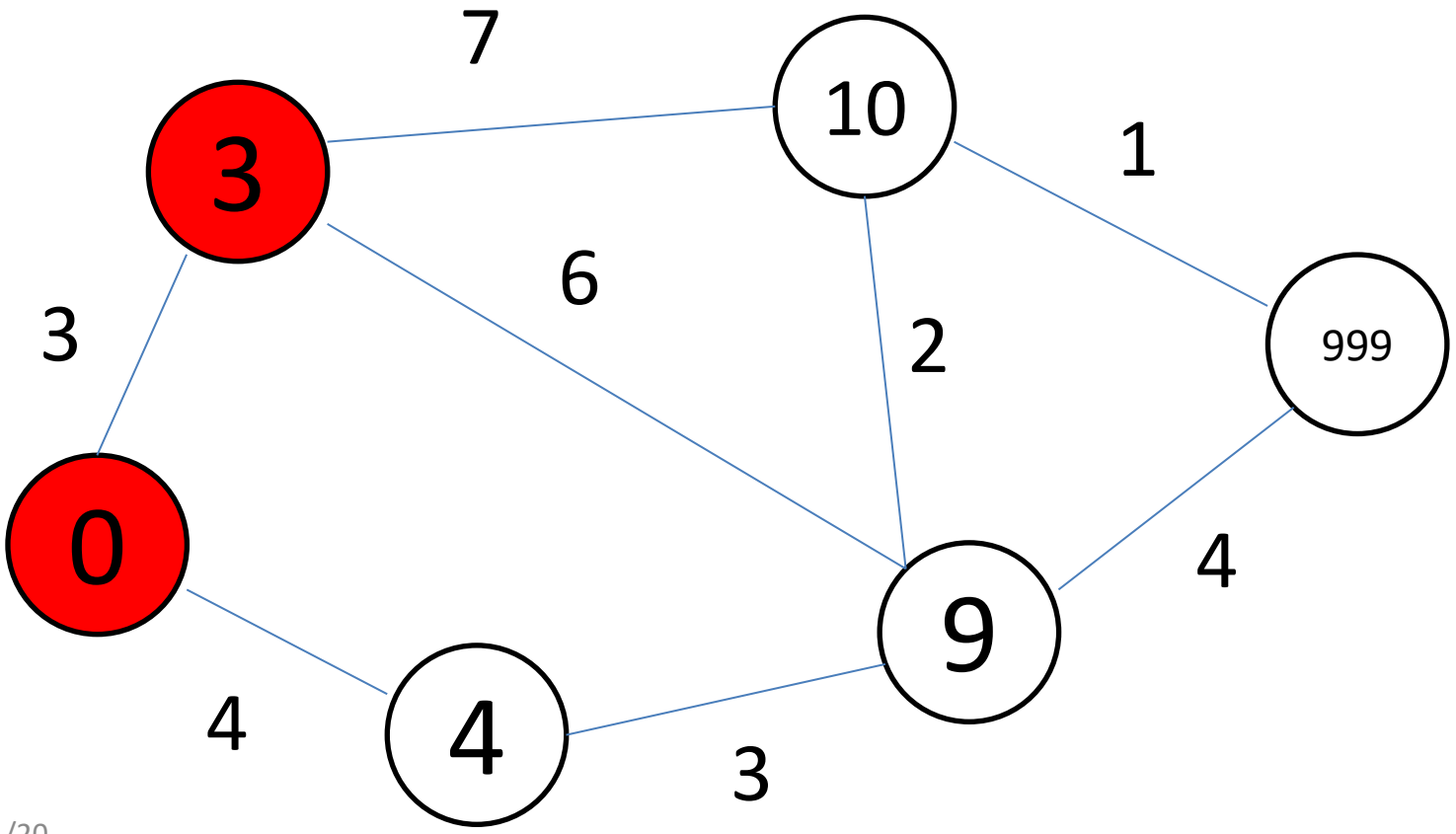
- 最短経路の求め方（ダイクストラ法）
  - AからBへの最短経路を求める方法
    - 周りを更新する



D問題　アルゴリズム

AtCoder

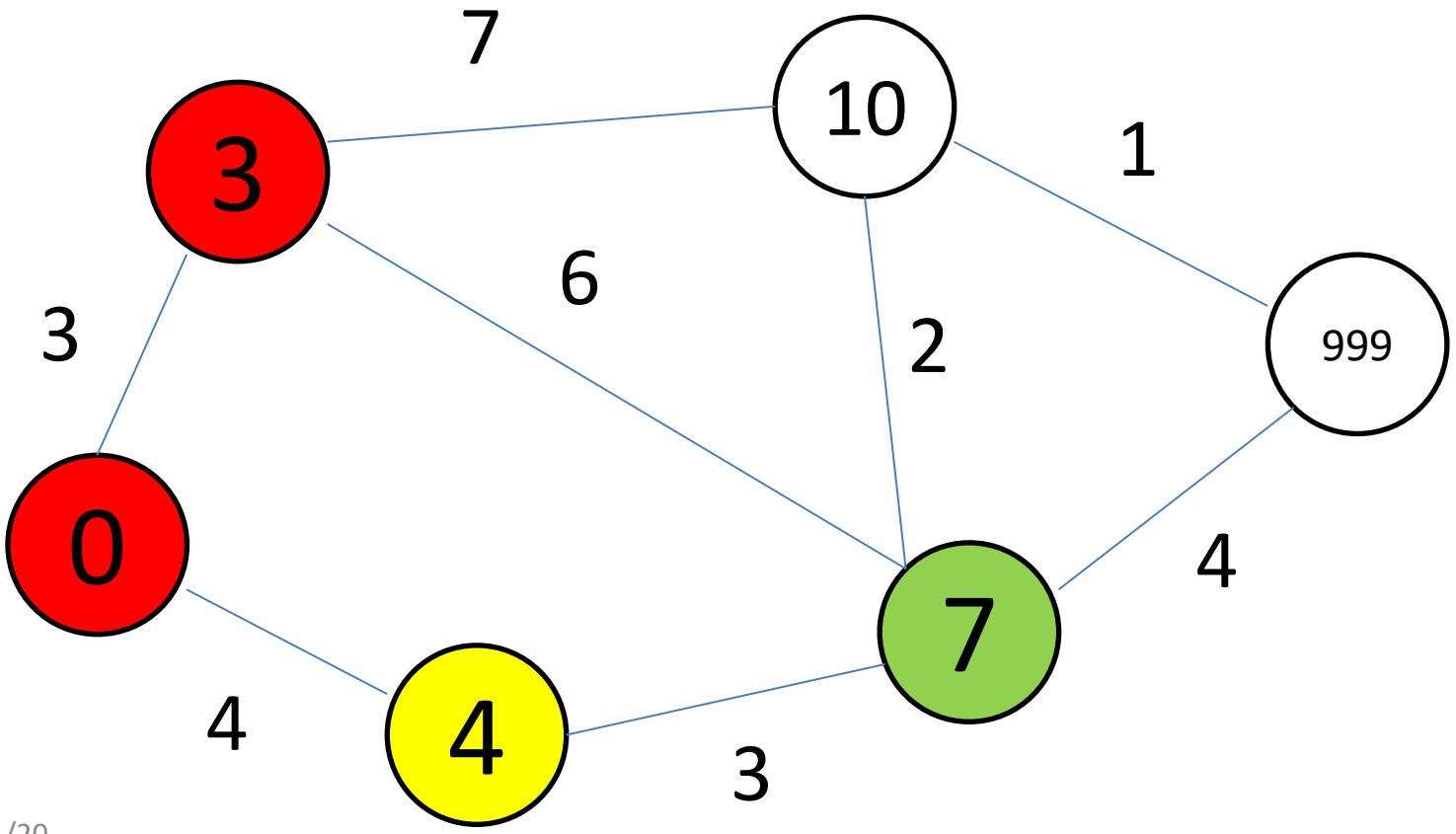
- 最短経路の求め方（ダイクストラ法）
  - AからBへの最短経路を求める方法
    - 使用済みにする



D問題　アルゴリズム

AtCoder

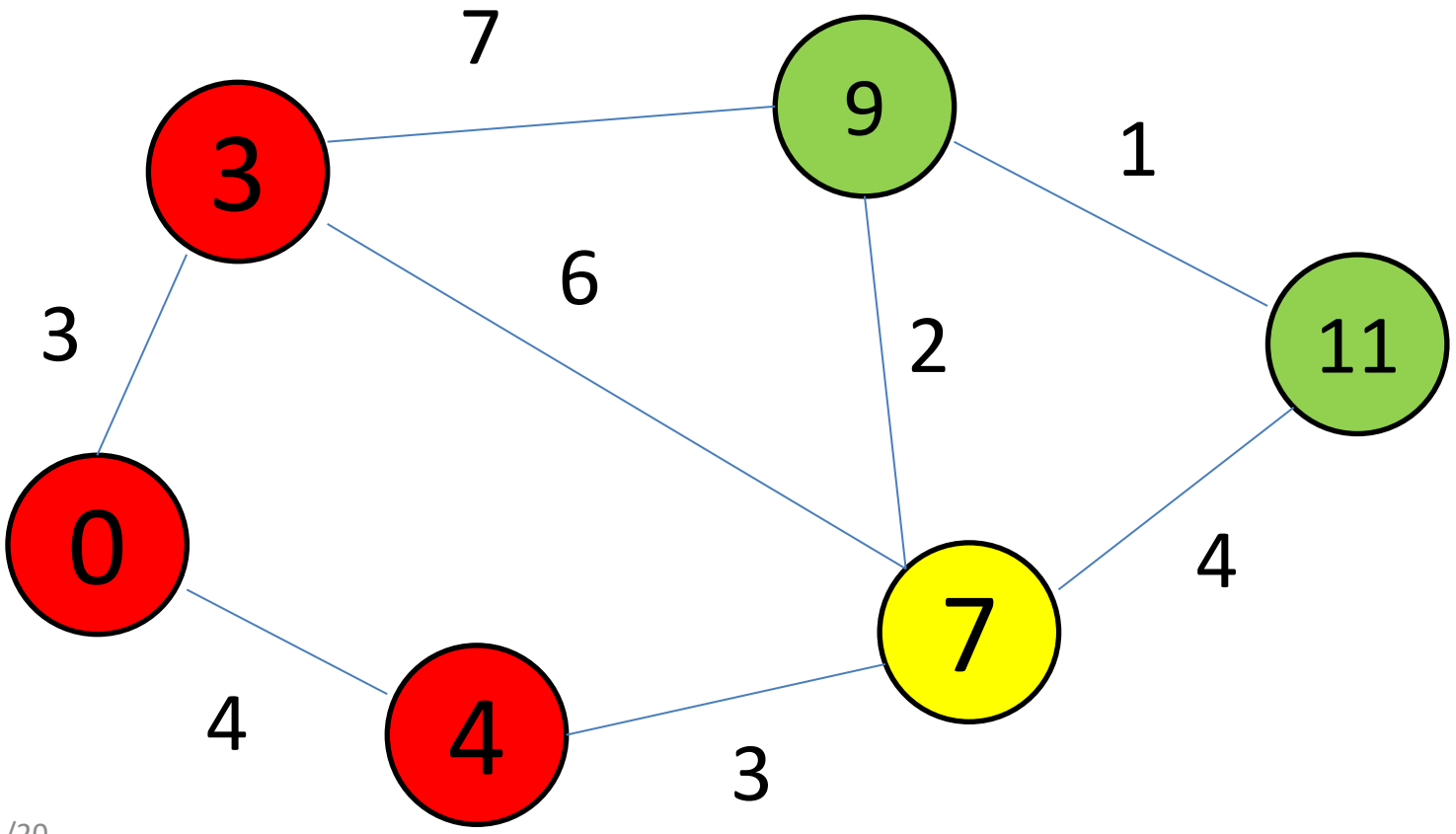
- 最短経路の求め方（ダイクストラ法）
  - AからBへの最短経路を求める方法
    - 同様に繰り返す



D問題　アルゴリズム

AtCoder

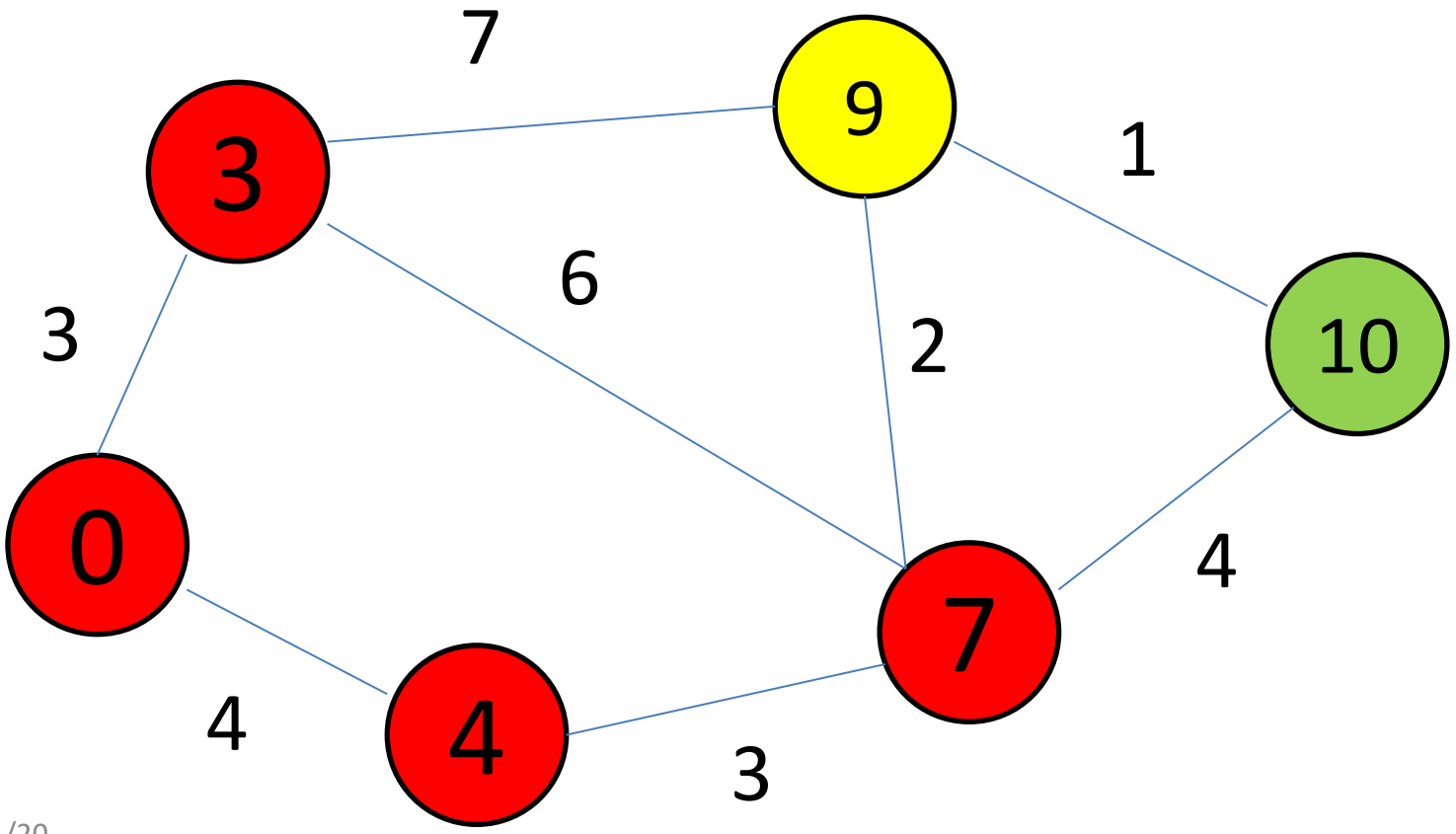
- 最短経路の求め方（ダイクストラ法）
  - AからBへの最短経路を求める方法
    - 同様に繰り返す



D問題　アルゴリズム

AtCoder

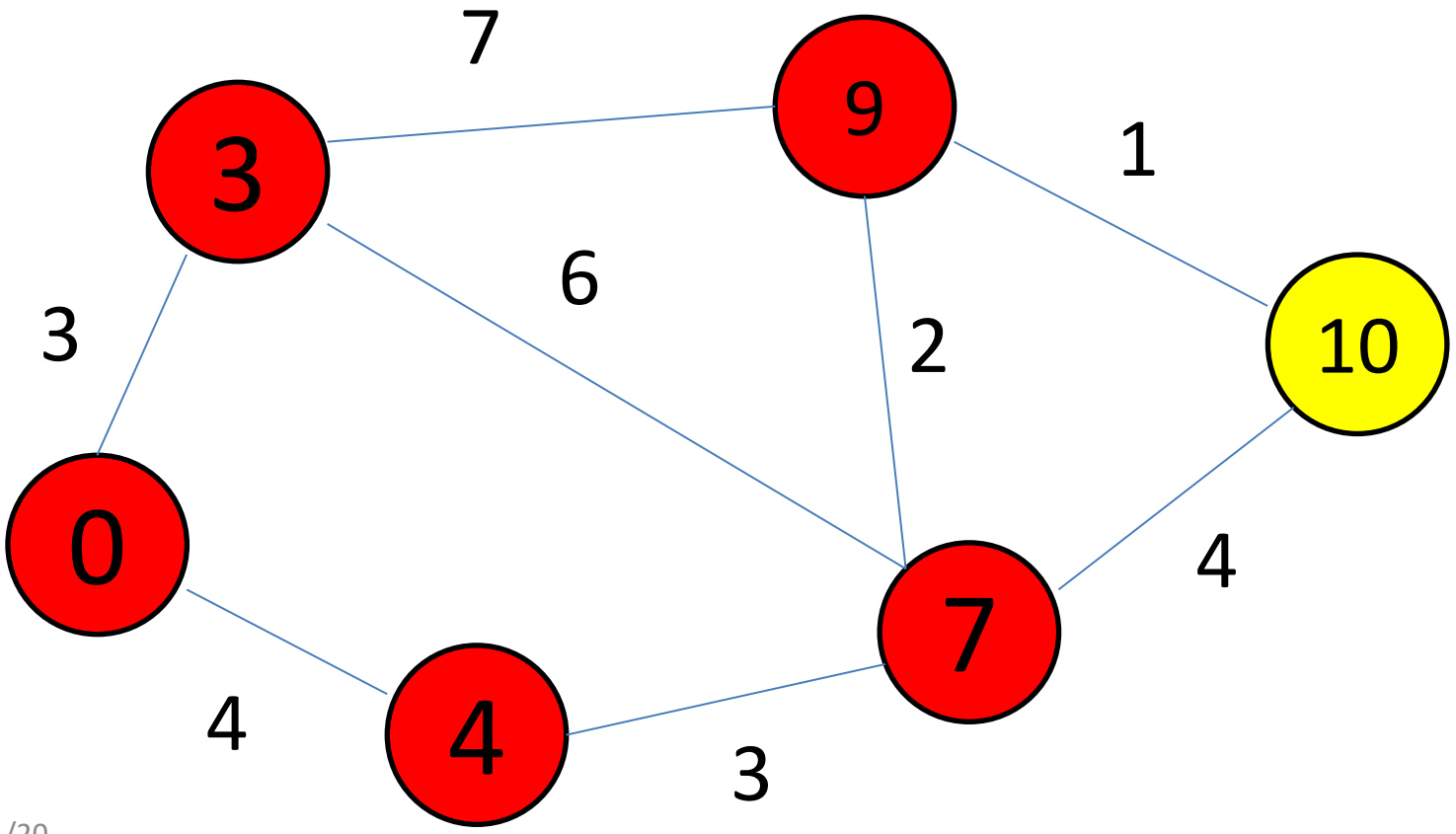
- 最短経路の求め方（ダイクストラ法）
  - AからBへの最短経路を求める方法
    - 同様に繰り返す



D問題　アルゴリズム

AtCoder

- 最短経路の求め方（ダイクストラ法）
  - AからBへの最短経路を求める方法
    - 同様に繰り返す



D問題　アルゴリズム

AtCoder

- 最短経路の求め方（ダイクストラ法）
  - 計算量はどれくらい？
  - 頂点数をV、辺の数をEとする。
  - アルゴリズムは以下のようになる
    - V回ループを回す
      - 最も数字（コスト）が少ない頂点を探す
      - そこから、各辺について調べ、その先の頂点の数字を更新する
  - 計算量は？

D問題　アルゴリズム

AtCoder

- 最短経路の求め方（ダイクストラ法）
  - 計算量はどれくらい？
  - 頂点数をVEとする。
  - アルゴリズムは以下のようになる
    - V回ループを回す（最後の頂点まで調べる必要があるため）
      - 最も数字（コスト）が少ない頂点を探す
      - そこから、各辺について調べ、その先の頂点の数字を更新する
  - 計算量は？
    - 頂点を探す処理は、最大V個
    - 各頂点から出ている辺も、重複さえなければ最大V個
  - よって、O(V^2)となる。

D問題　アルゴリズム

AtCoder

- AからBへの最短経路を求める計算がO(N^2)
  - つまり、場所のペアの組み合わせはO(N^2)パターンあるから、合わせてO(N^4)？間に合わない？

D問題　アルゴリズム

AtCoder

- AからBへの最短経路を求める計算がO(N^2)
  - つまり、場所のペアの組み合わせはO(N^2)パターンあるから、合わせてO(N^4)？間に合わない？
- AからBへの最短経路を求めるだけでなく、Aから他の全ての頂点までの最短経路をO(N^2)で求められるのがダイクストラ法
  - よって、ダイクストラ法を使うのはN回で良く、O(N^3)
  - N=300の時、N^3は2700万なので、高速な言語なら簡単に通せる

D問題　アルゴリズム

AtCoder

- その他の方法
  - ワーシャルフロイドと呼ばれるアルゴリズムを使うと、凄く簡単に求めることが出来る！
  - ダイクストラ法との違い
    - ダイクストラは、単一始点最短路を求めるアルゴリズム
      - ある頂点を始点とした時、それ以外の頂点への最短路を求める
    - ワーシャルフロイドは、全点対間最短路を求めるアルゴリズム
      - 任意の頂点から任意の頂点までの最短路を求めるアルゴリズム
        - » 今回の目的にぴったり！

D問題　アルゴリズム

AtCoder

- ワーシャルフロイド法の実装
  - 非常に簡単、以下のようなアルゴリズムで実装可能！
  - 各点同士の距離を表す配列distには、あらかじめ、以下のように数字設定をする
    - dist[i][i] = 0;
    - dist[a_i][b_i] = dist[b_i][a_i] = t_i;
      - 今回の問題では、両側に行けるので、両方に入れないとダメ！
    - その他は、非常に大きい値を入れておく。
      - 二つ足してもオーバーフローしない程度にしましょう！

```
For(int K = 0; K < N; K++)
    For(int I = 0; I < N; I++)
        For(int J = 0; j < N; j++)
            dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j]);
```

D問題　アルゴリズム

AtCoder

- ワーシャルフロイドの仕組み

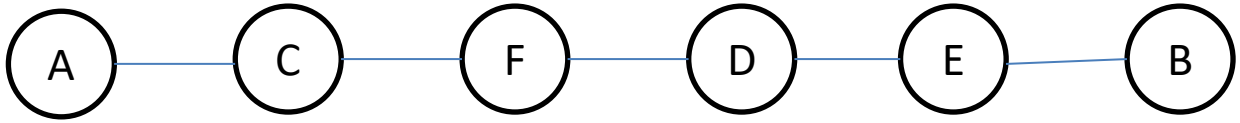
```
For(int K = 0; K < N; K++)
    For(int I = 0; I < N; I++)
        For(int J = 0; j < N; j++)
            dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j]);
```

- Kが中継点であり、中継点を全通り試している。
  - 中継点を全部試せば、全ての点同士の最短経路を求められそう

D問題　アルゴリズム

AtCoder

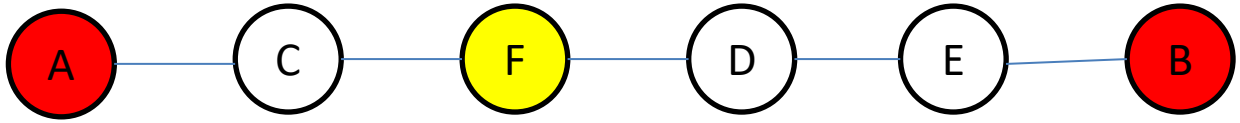
- ワーシャルフロイドのイメージ
  - A,Bの最短経路が、A,C,F,D,E,Bだったとする
    - 中継点はアルファベット順に調べるものとする



D問題　アルゴリズム

AtCoder

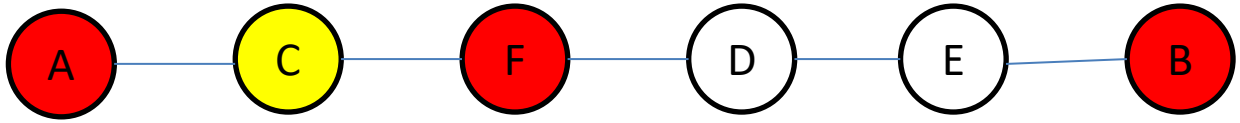
- ワーシャルフロイドのイメージ
  - A,Bの最短経路が、A,C,F,D,E,Bだったとする
    - 中継点はアルファベット順に調べるものとする
  - A,Bの最短路が正しく求められるには、最後の中継点Fを調べる時に、A-Fの最短路と、F-Bの最短路が求まっていれば良い



D問題　アルゴリズム

AtCoder

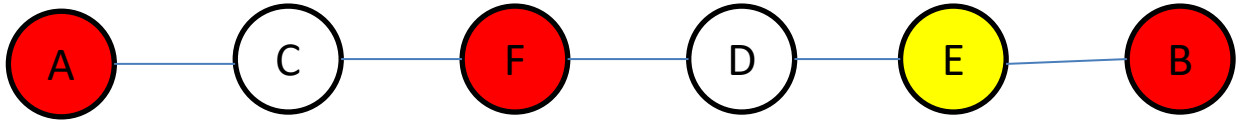
- ワーシャルフロイドのイメージ
  - A,Bの最短経路が、A,C,F,D,E,Bだったとする
    - 中継点はアルファベット順に調べるものとする
  - A,Bの最短路が正しく求められるには、最後の中継点Fを調べる時に、A-Fの最短路と、F-Bの最短路が求まっていれば良い
  - A-Fの最短路は、A-Fにある最後の中継点Cを見た時に、A-C、C-Fは１辺しかなく、A-Fの最短路がA,C,F,D,E,Bということは、A-C,C-Fはこれ以外のルートが最短であることはないので、最短路は正しく求まっている。



D問題　アルゴリズム

AtCoder

- ワーシャルフロイドのイメージ
  - A,Bの最短経路が、A,C,F,D,E,Bだったとする
    - 中継点はアルファベット順に調べるものとする
  - A,Bの最短路が正しく求められるには、最後の中継点Fを調べる時に、A-Fの最短路と、F-Bの最短路が求まっていれば良い
  - F-B間の最短路も、同様にEから調べていけば、最短路が順番に求まっていることが解る
    - このように、最短路になる部分を再帰的に見ていくと、最短路が順番に求まっていることが確認できる。



D問題　アルゴリズム

AtCoder

- まとめ
  - 全点対間最短経路を求める時には、ワーシャルフロイドを使おう！
    - 計算量はO(V^3)
      - 疎なグラフ（辺が少ないグラフ）の時は、全点対間最短経路を求める時でも、優先度付きキューを利用したダイクストラを繰り返した方が早いこともあるので一応注意
    - 実装が凄く簡単！
      - 4行でかける！

D問題　アルゴリズム

AtCoder

- 注意点
  - 遅い言語では通りません！
    - Perl, Ruby, Pythonなど
    - 計算量が一定以上重い問題だと、LL系の言語では、通すことが出来ません。
      - 言語の選択もコンテストのうちです！
      - テストの作り方が甘いので、上手く誤魔化せば通るかも
        - » 各ノードのコストが少ない辺ベスト20だけでダイクストラとか。
        - » 普通に書いて通らないことは検証していますが、誤魔化して通るかは未検証です。

D問題　アルゴリズム

AtCoder

- おまけ
  - ダイクストラ法は、O((E+V)logV)な書き方もあります。
    - 優先度付きキューをうまいこと使えば良い