# スタンプラリー解説

一階堂

# 問題概要

問題文を読みましょう。

# 問題概要

上り線と下り線のある路線があり、各駅の

上り線のホーム → スタンプ台
下り線のホーム → スタンプ台
スタンプ台 → 上り線のホーム
スタンプ台 → 下り線のホーム

の移動時間と駅間の移動時間が定まっている。
各駅にひとつずつあるスタンプを集める最短時間を求めよ。

# Subtask 1 (10点)

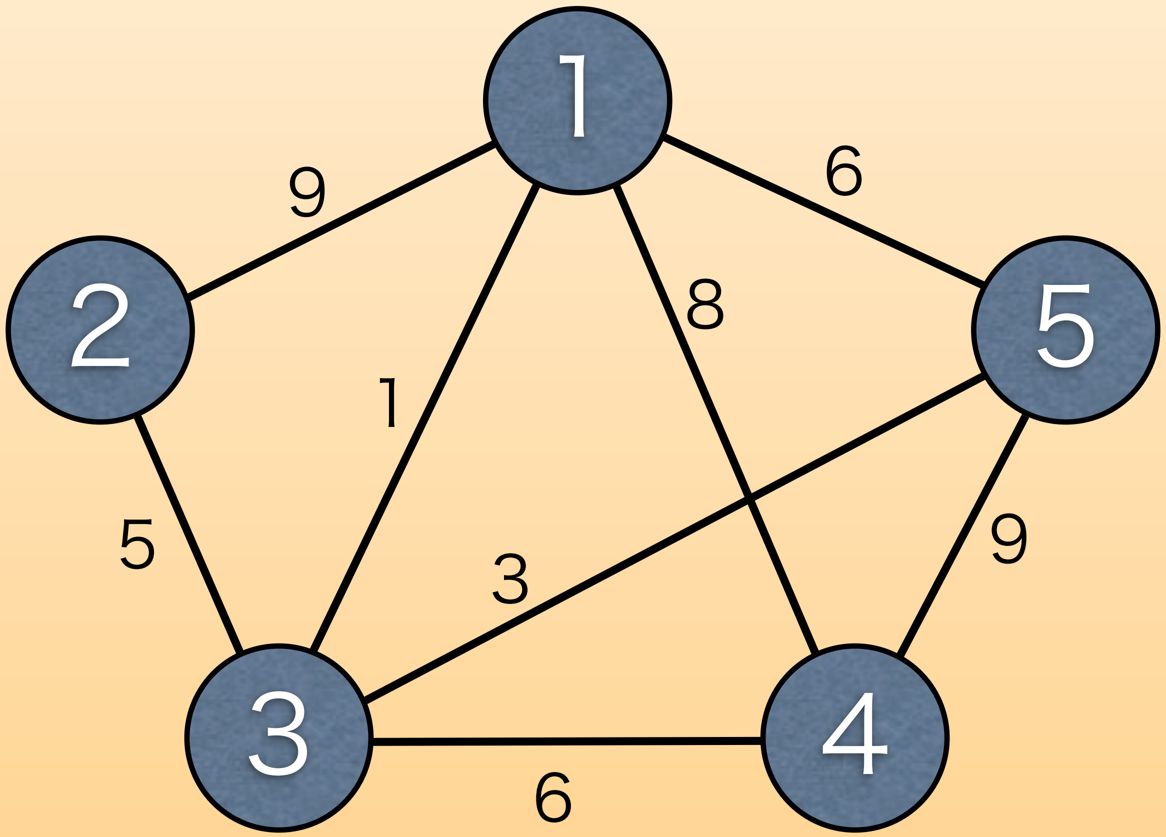
・N ≦ 16

この問題は、**巡回セールスマン問題(TSP)**

巡回セールスマン問題を O(2^N * N^2) で解けば、subtask1を正解し、10点を得られる。

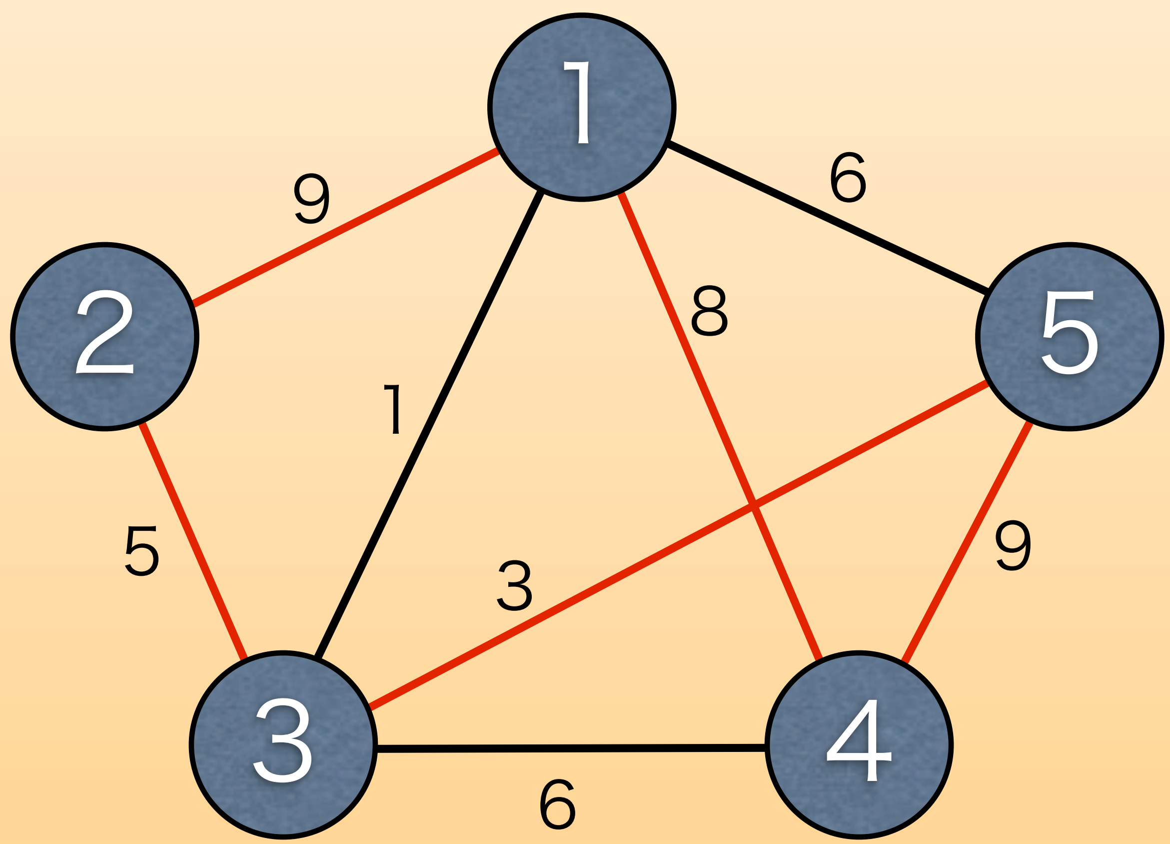
# 巡回セールスマン問題とは

重み付きグラフ（辺にコストがついたグラフ）が与えられた時に、最小コストで全ての頂点を１回ずつ巡る経路を求める問題。



# 巡回セールスマン問題とは

重み付きグラフ（辺にコストがついたグラフ）が与えられた時に、最小コストで全ての頂点を１回ずつ巡る経路を求める問題。



# 巡回セールスマン問題の解き方

DP[既に訪れた頂点の集合][今いる頂点]

状態数：2^N * N
遷移数：１つの状態あたりO(N)
計算量：O(2^N * N^2)

集合は２進数で表現すると良い。

# グラフの作り方

巡回セールスマン問題として解くためには、

「駅iのスタンプ台から駅jのスタンプまで
　行くための最短時間」

を適切に求めて完全グラフを作っておけば良い。

# Subtask 2 (75点)

・いい感じのDPがしたい。
とりあえず駅ごとに独立に考えることが出来ないかを考えてみる。

# Subtask 2 (75点)

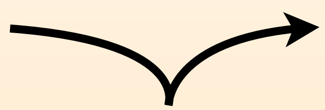
各駅のスタンプ台を通るための方法は、

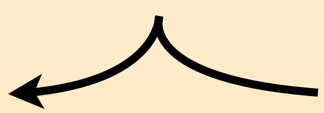
(A) 上り線から来て上り線に戻る： Vi + Ui
(B) 下り線から来て下り線に戻る： Di + Ei
(C) 上り線から来て下り線に戻る： Vi + Ei
(D) 下り線から来て上り線に戻る： Di + Ui

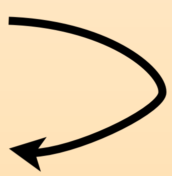
の４通り。

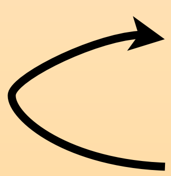
# Subtask 2 (75点)

図にしてみると、

(A) 

(B) 

(C) 

(D) 

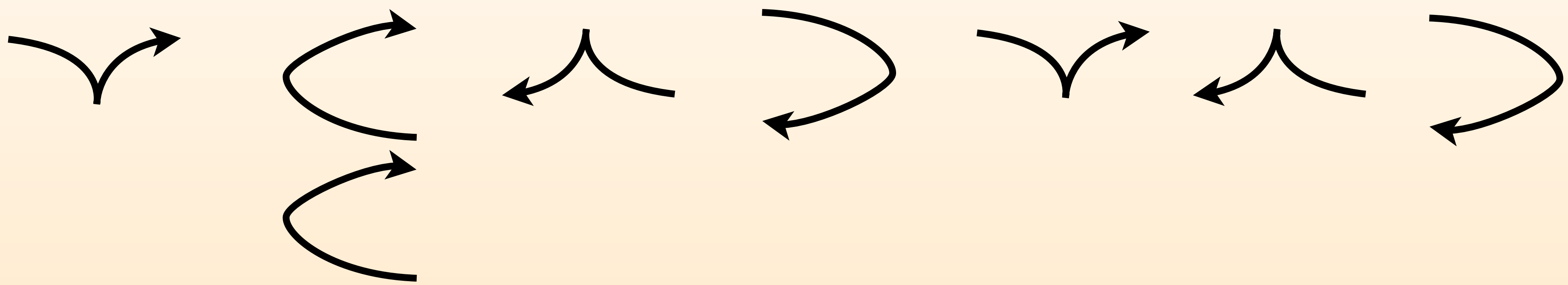
という感じ。

# Subtask 2 (75点)

要するに、各場所に

を１個以上ずつ並べる。
この並べ方が満たすべき性質を考察する。

# 列が満たすべき性質

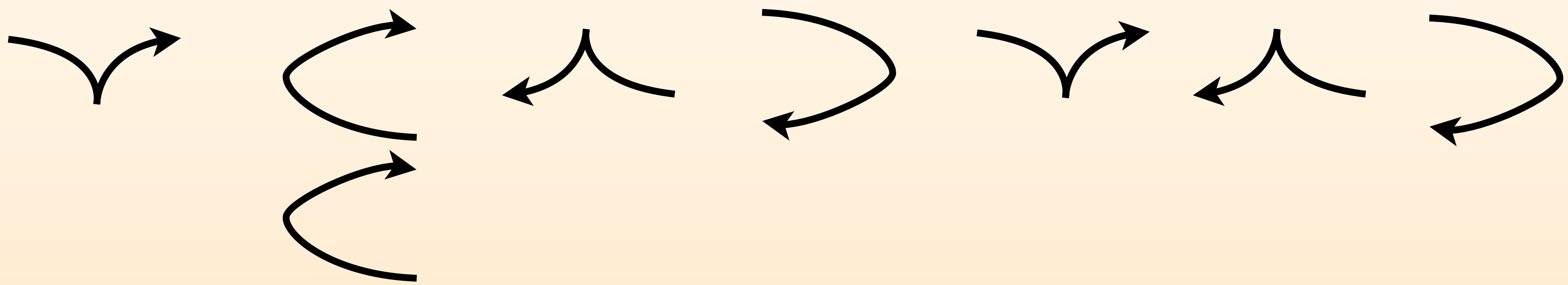


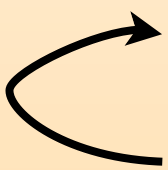
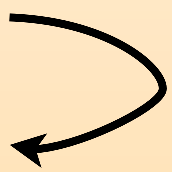
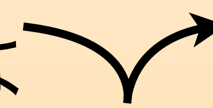
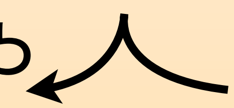
・各場所において の個数が常に より多い

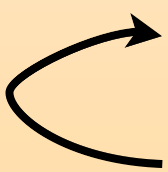
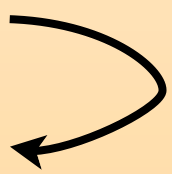
・ の合計が の合計と等しい

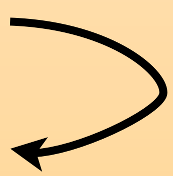
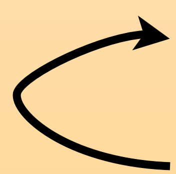
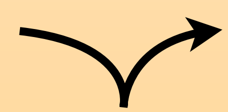
・ は が より多いところだけに置ける

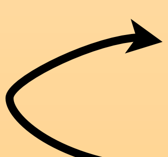
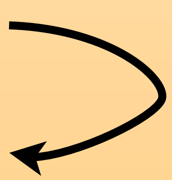
# 意味のある列の性質



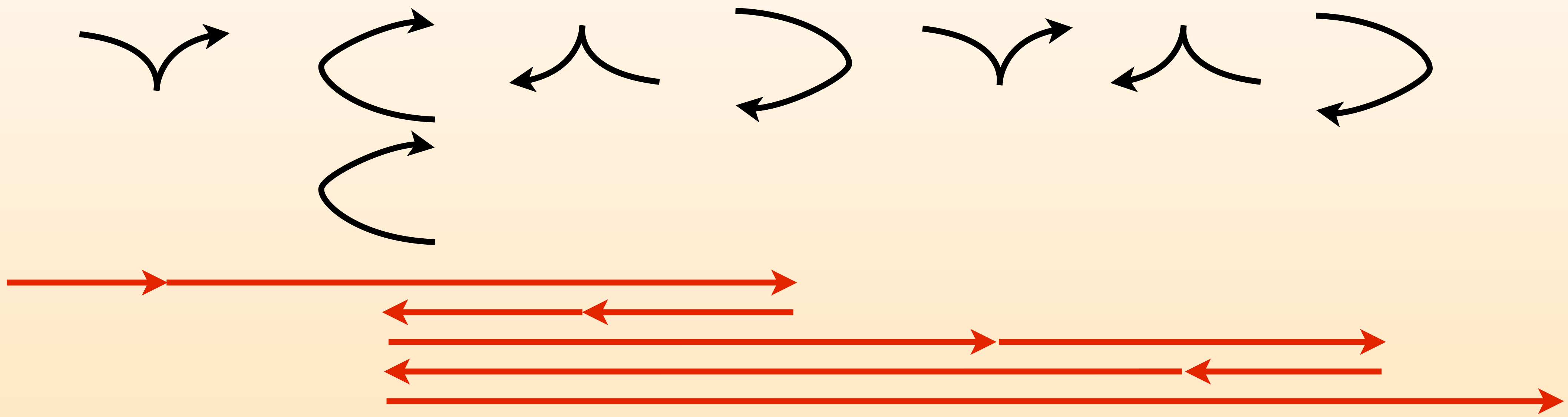
・  や  があるところには  や  は不要

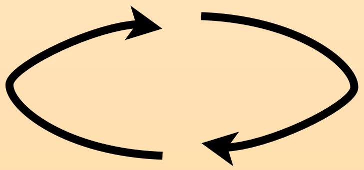
・  と  を同じところに置くのは無駄

理由：  ＋  よりも  の方が真に小さいから

・  や  の個数は高々N個くらいで十分

# 電車での移動は？



・T × (各駅間を囲んでる  の数 × 2 + 1)

# DPの状態と遷移

DP[今注目してる駅][ の個数ー の個数]

DP[i][j]からの遷移：

・だけ使う
・だけ使う（j ≧ 1のときのみ）

→ O(1)

・だけをいくつか使う
・だけをいくつか使う

→ O(N)

# DPの計算量

DP[今注目してる駅][ の個数ー の個数]

状態数：N^2
遷移数：O(N)
計算量：O(N^3)

ここまででSubtask1とSubtask2に正解し、合計85点を得ることが出来る。

# Subtask 3 (15点)

DP[今注目してる駅][ の個数ー の個数]

DP[i][j]からの遷移：

・だけ使う

・だけ使う（ j >= 1のときのみ）

O(1)

・だけをいくつか使う

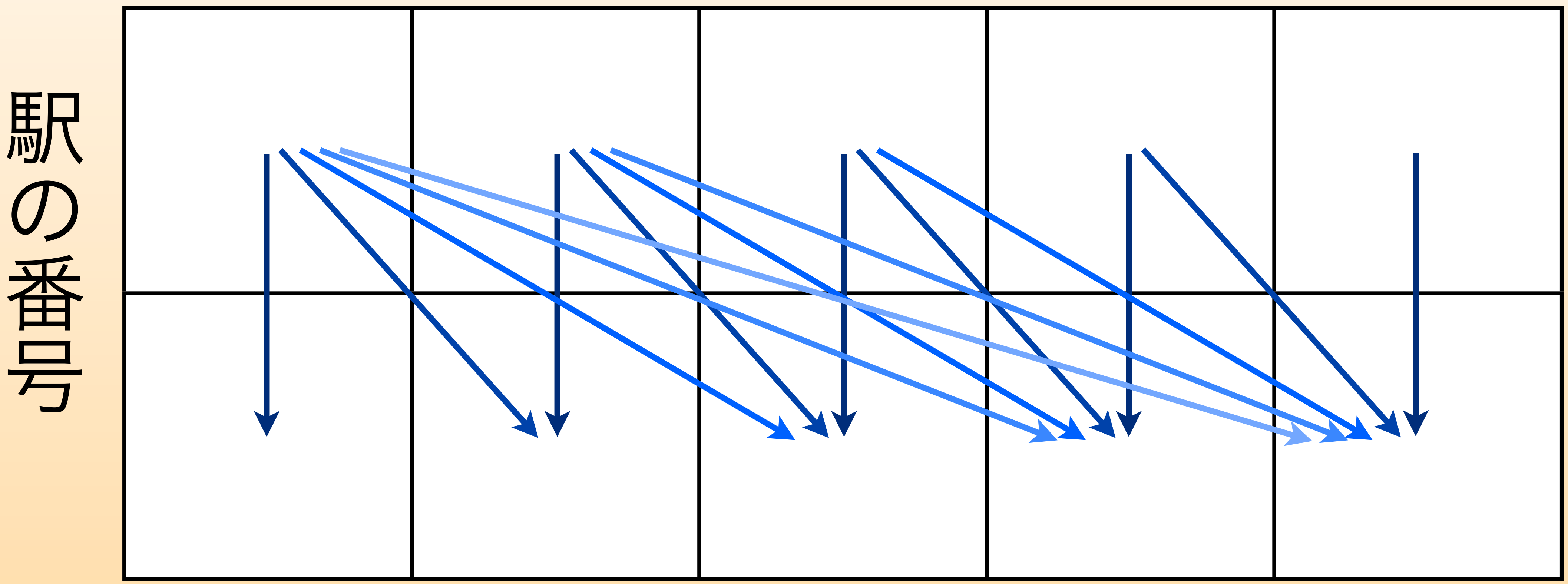
・だけをいくつか使う

O(N)

ここを改善する

# だけをいくつか使う

の個数ー　の個数

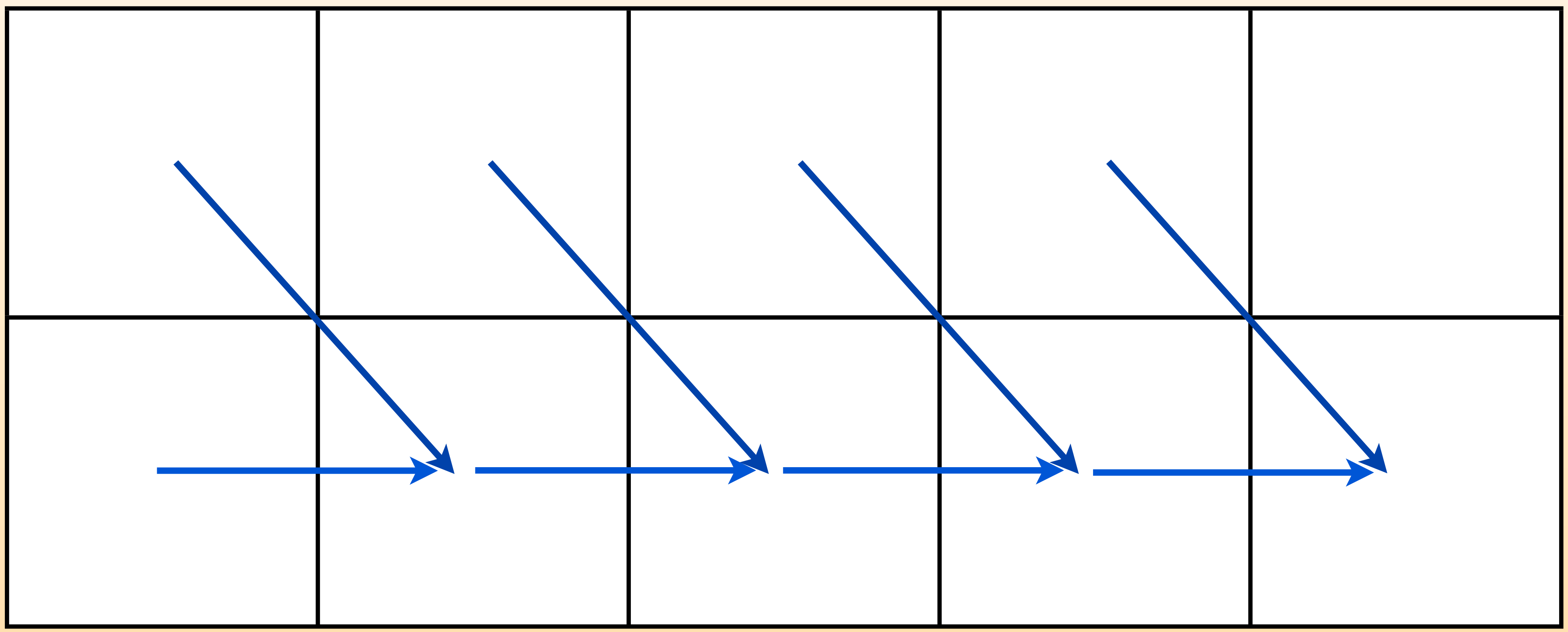



**これを**

# だけをいくつか使う

の個数ー　の個数

駅の番号



**こうじゃ**

（よくある改善）

# だけをいくつか使う

式で書くと、

DP[i+1][j] = min(DP[i][j-k]+hoge | k = 0 ~ j-1)

↓

DP[i+1][j] = min(DP[i][j-1]+hoge, DP[i+1][j-1]+piyo)

みたいな雰囲気

も逆方向に同様なことをすれば良い。

# 改善したDPの計算量

状態数：N^2

遷移数：O(1)

計算量：O(N^2)

これで満点が取れる。

# 統計