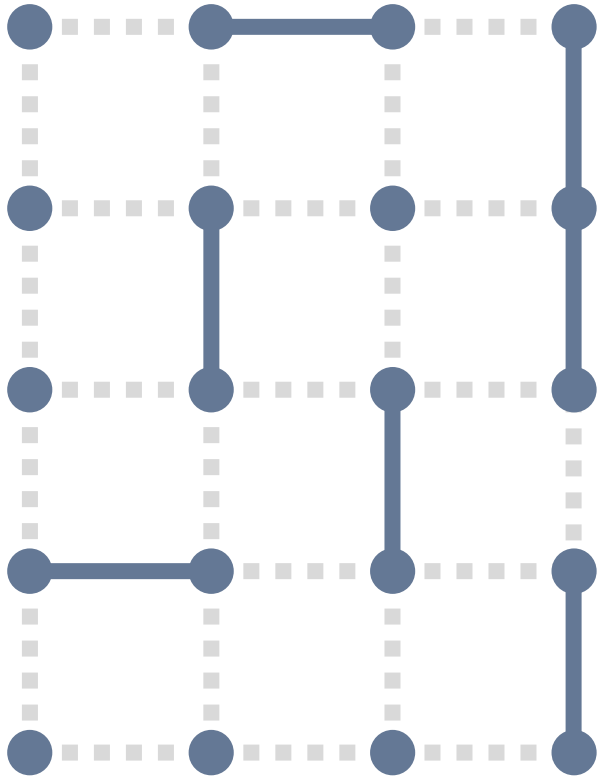
道路網の整備2 解説

米田優峻 (E869120)

2024 年 2 月 4 日

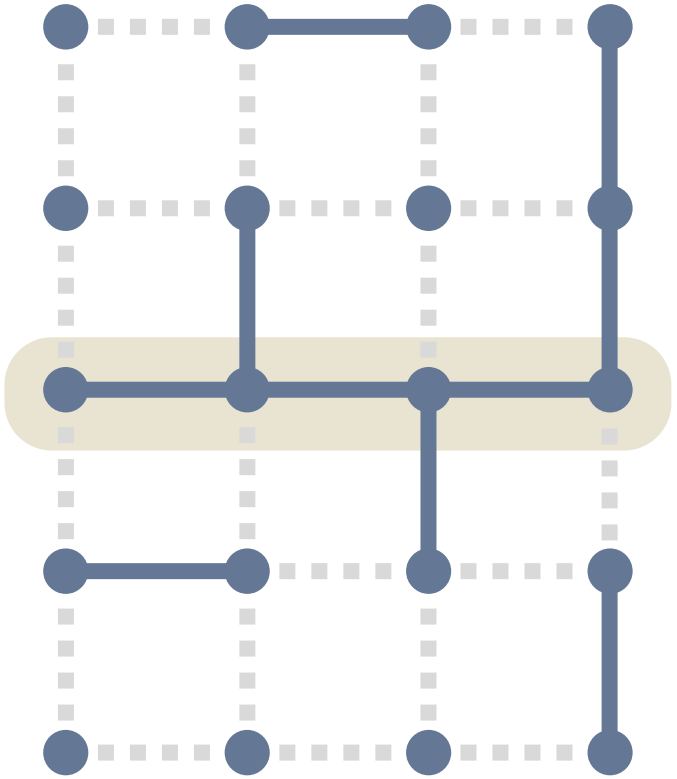
問題概要

- H 行 W 列の碁盤目状道路がある
- 右図のように、道路には通れる部分と通れない部分がある



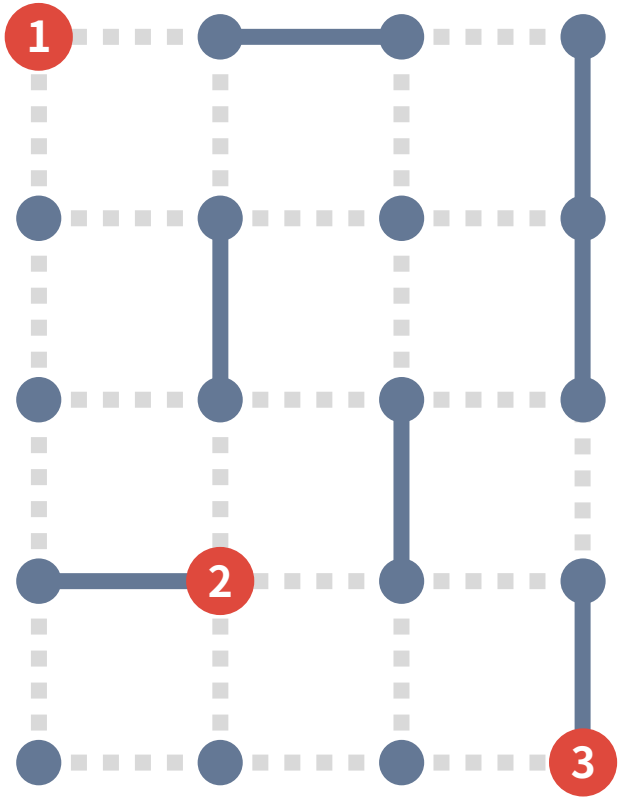
問題概要

- H 行 W 列の碁盤目状道路がある
- 右図のように、道路には通れる部分と通れない部分がある
- あなたは C_i 日間の整備により、i 行目の道路を全部「通れる状態」にすることができる

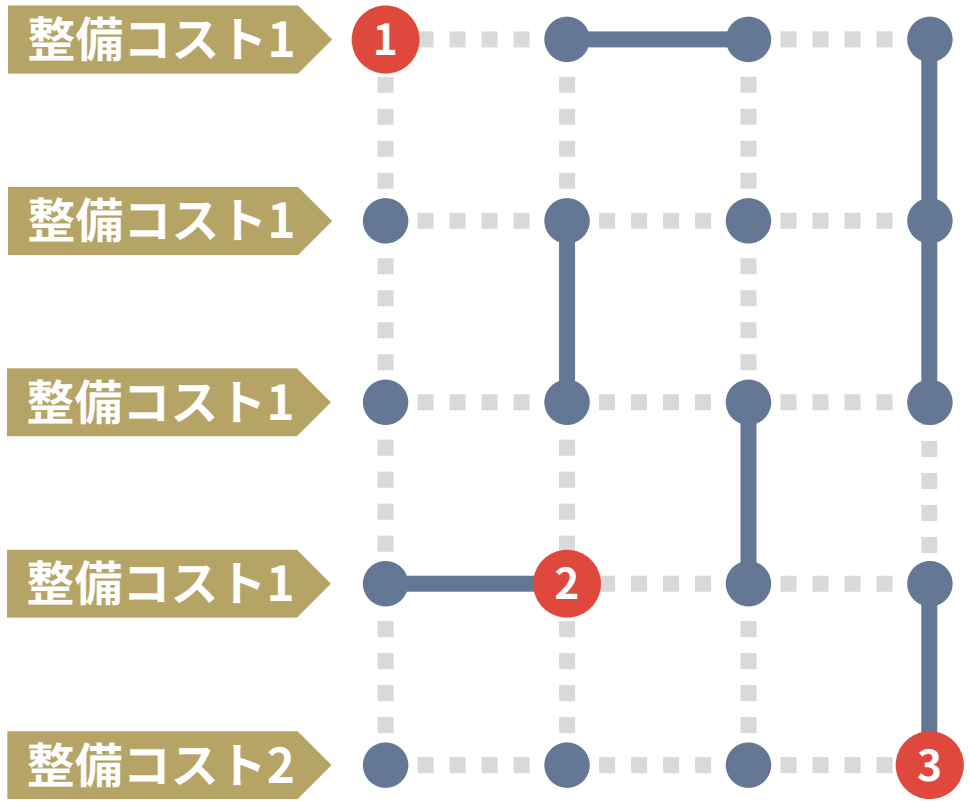


問題概要

- H 行 W 列の碁盤目状道路がある
- 右図のように、道路には通れる部分と通れない部分がある
- あなたは C_i 日間の整備により、i 行目の道路を全部「通れる状態」にすることができる
- 交差点 $(X_1, Y_1), (X_2, Y_2), \ldots, (X_T, Y_T)$ を連結にするには何日かかるか？

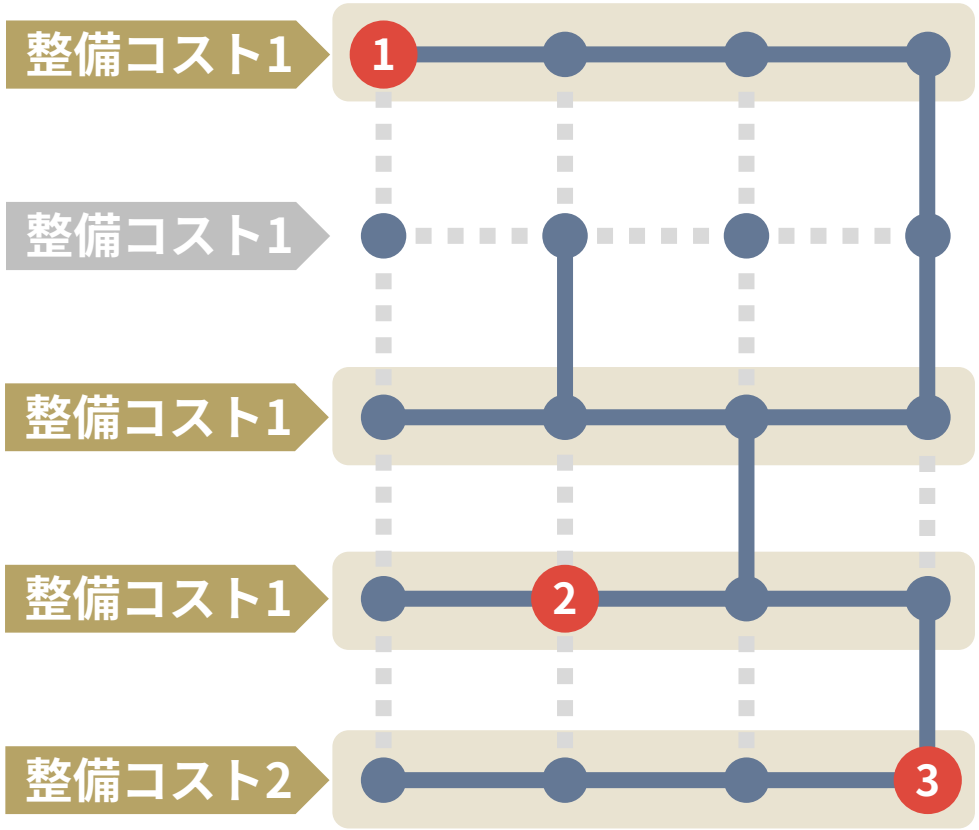


問題概要: 具体例



- たとえば、左図の道路の地点 1, 2, 3 を連結にするにはコストいくつ必要か？

問題概要: 具体例



- たとえば、左図の道路の地点 1, 2, 3 を連結にするにはコストいくつ必要か？
- 1, 3, 4, 5 行目を整備するのが最適で、コスト 5 が必要

問題概要

問題概要はわかりましたか？

問題概要 (制約)

- 碁盤目状道路の大きさ: $H \times W \leq 1\,000\,000$
- 整備にかかるコスト: $C_i = 1$ または 2
- 質問回数: $Q \leq 100\,000$ (T の値の合計は $200\,000$ 以下)

注意

- 小課題 5 までは $C_i = 1$ が続きます
- しばらくの間は、どの行の整備も 1 日で出来ると思って解説を読んでください

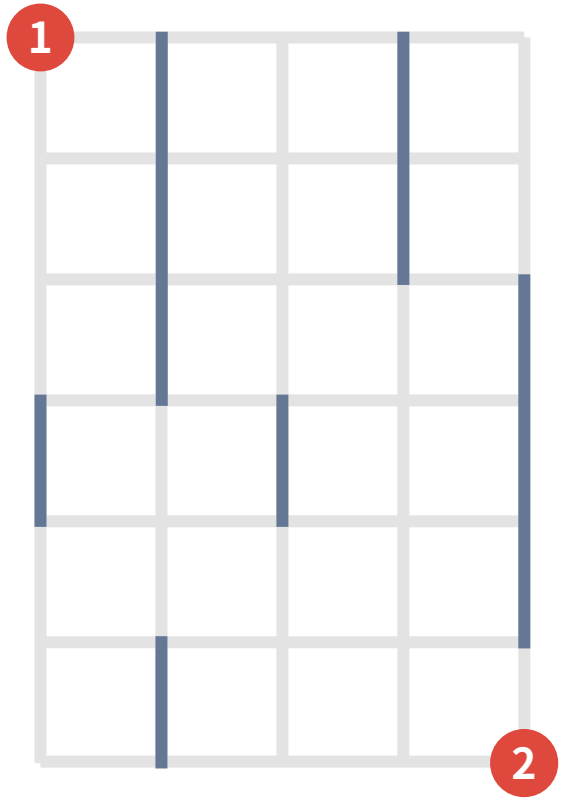
小課題 1, 2

$C_i = 1, Q \leq 5, T = 2$

小課題 1, 2

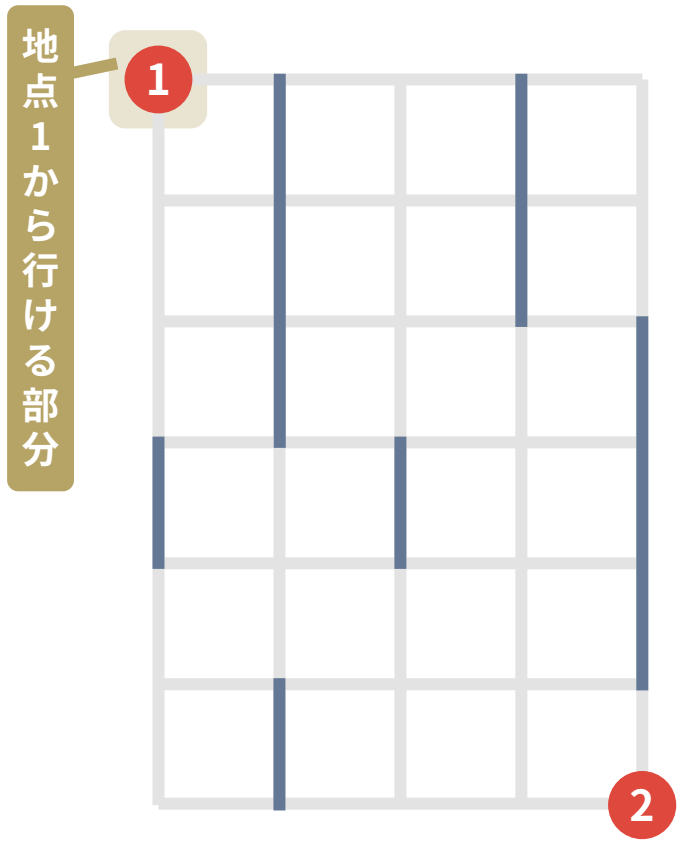
基本方針：できるだけ下で整備する

小課題 1, 2: 具体例 A



例: 下図の 1 と 2 を連結にすることを考える

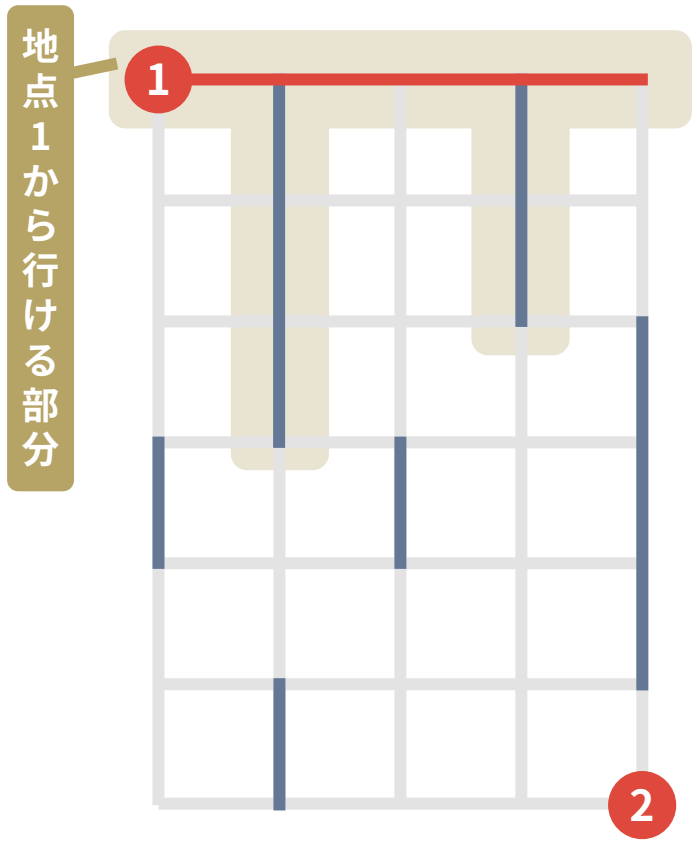
小課題 1, 2: 具体例 A



例: 下図の 1 と 2 を連結にすることを考える

- 現時点で地点 1 からは 1 行目まで到達可能 → 1 行目を整備

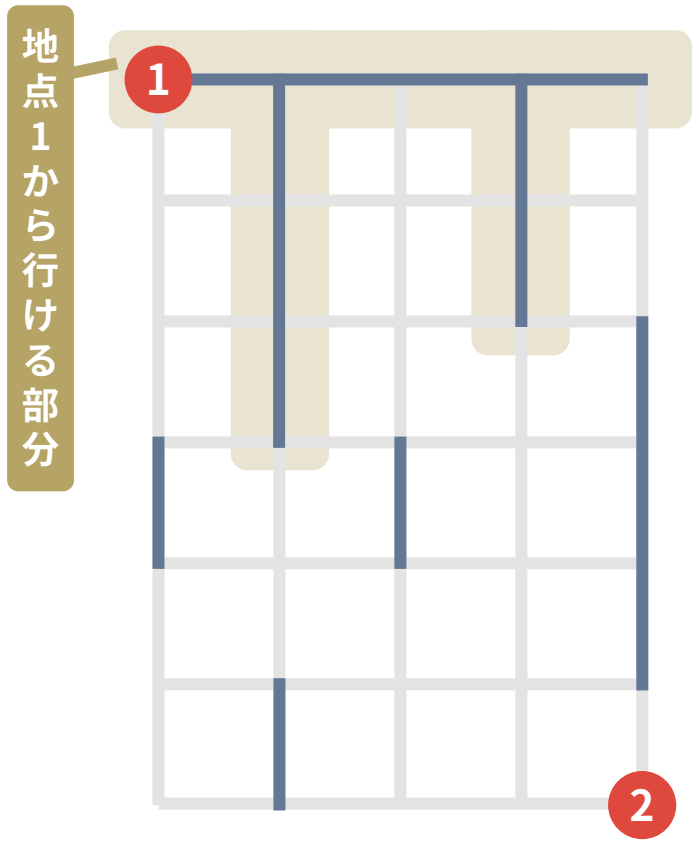
小課題 1, 2: 具体例 A



例: 下図の 1 と 2 を連結にすることを考える

- 現時点で地点 1 からは 1 行目まで到達可能 → 1 行目を整備

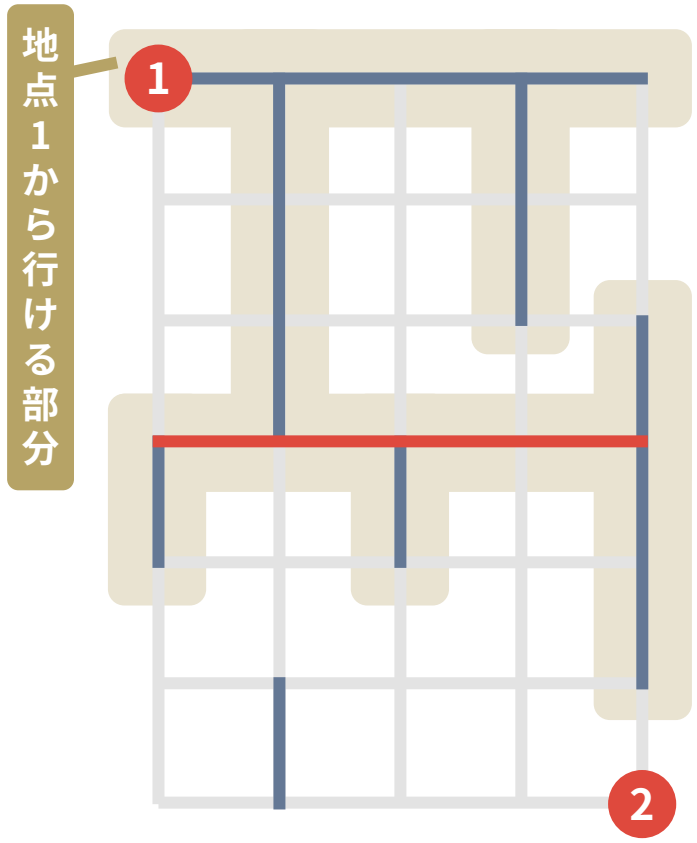
小課題 1, 2: 具体例 A



例: 下図の 1 と 2 を連結にすることを考える

- 現時点で地点 1 からは 1 行目まで到達可能 → 1 行目を整備
- 現時点で地点 1 からは 4 行目まで到達可能 → 4 行目を整備

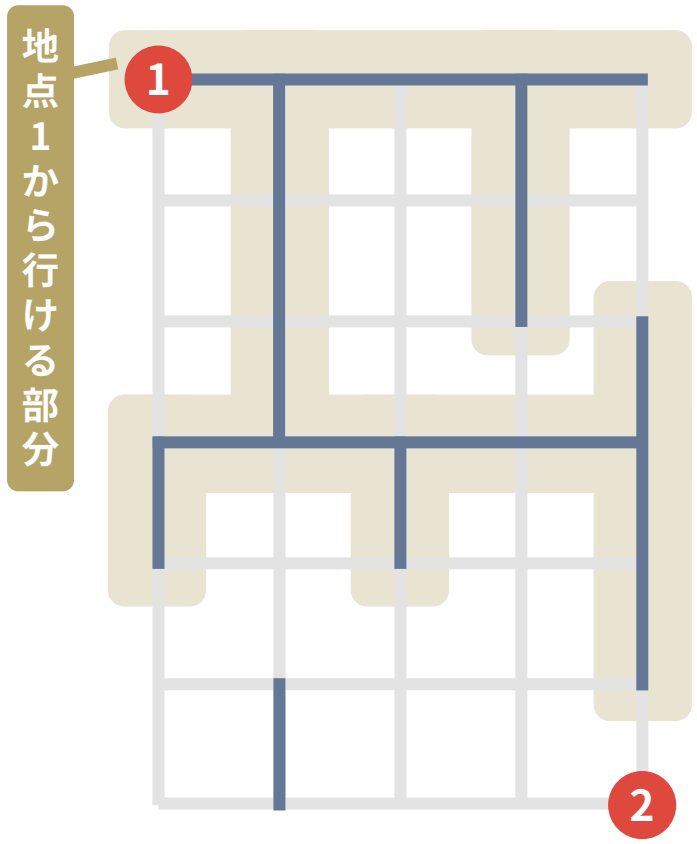
小課題 1, 2: 具体例 A



例: 下図の 1 と 2 を連結にすることを考える

- 現時点で地点 1 からは 1 行目まで到達可能 → 1 行目を整備
- 現時点で地点 1 からは 4 行目まで到達可能 → 4 行目を整備

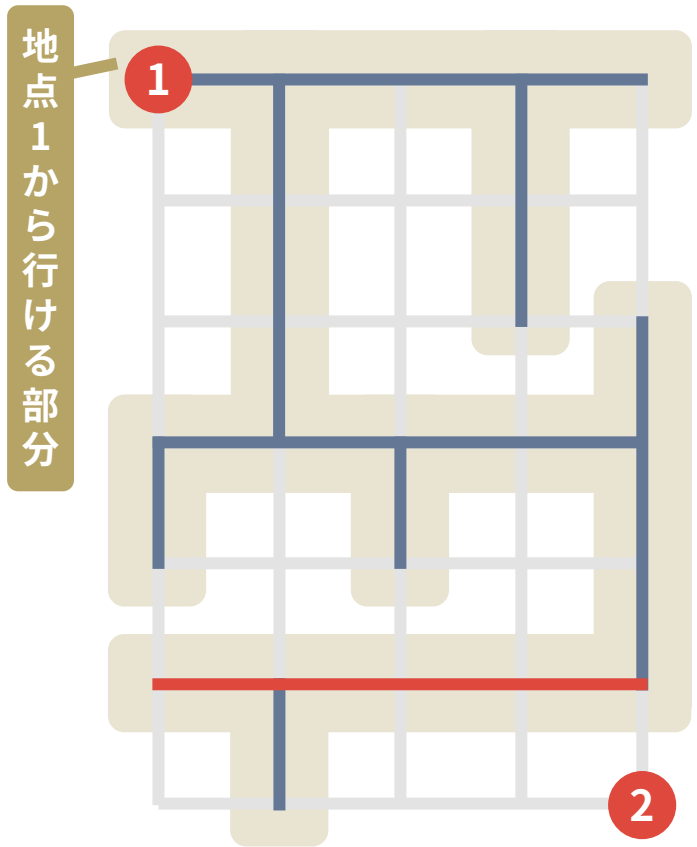
小課題 1, 2: 具体例 A



例: 下図の 1 と 2 を連結にすることを考える

- 現時点で地点 1 からは 1 行目まで到達可能 → 1 行目を整備
- 現時点で地点 1 からは 4 行目まで到達可能 → 4 行目を整備
- 現時点で地点 1 からは 6 行目まで到達可能 → 6 行目を整備

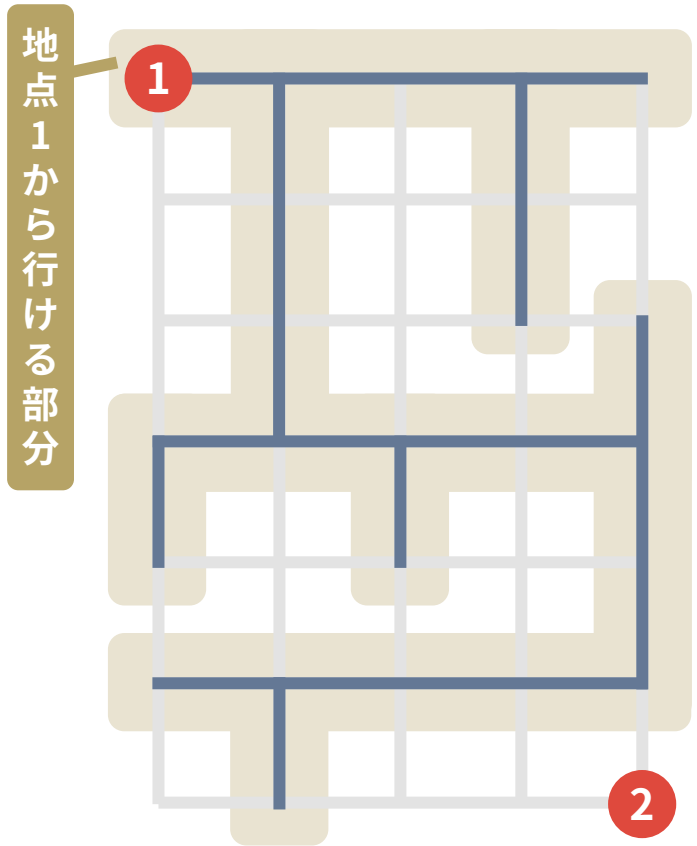
小課題 1, 2: 具体例 A



例: 下図の 1 と 2 を連結にすることを考える

- 現時点で地点 1 からは 1 行目まで到達可能 → 1 行目を整備
- 現時点で地点 1 からは 4 行目まで到達可能 → 4 行目を整備
- 現時点で地点 1 からは 6 行目まで到達可能 → 6 行目を整備

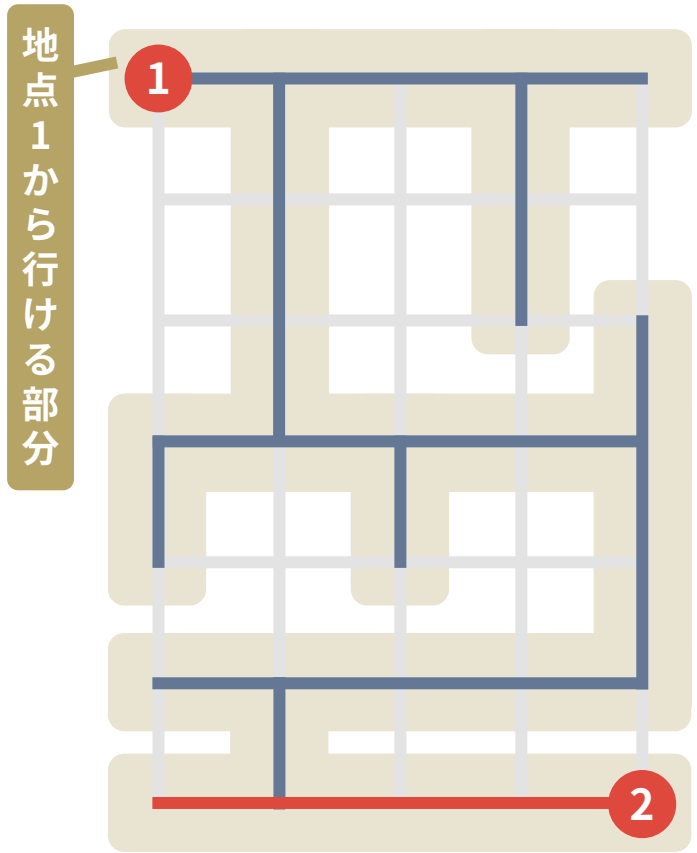
小課題 1, 2: 具体例 A



例: 下図の 1 と 2 を連結にすることを考える

- 現時点で地点 1 からは 1 行目まで到達可能 → 1 行目を整備
- 現時点で地点 1 からは 4 行目まで到達可能 → 4 行目を整備
- 現時点で地点 1 からは 6 行目まで到達可能 → 6 行目を整備
- 現時点で地点 1 からは 7 行目まで到達可能 → 7 行目を整備

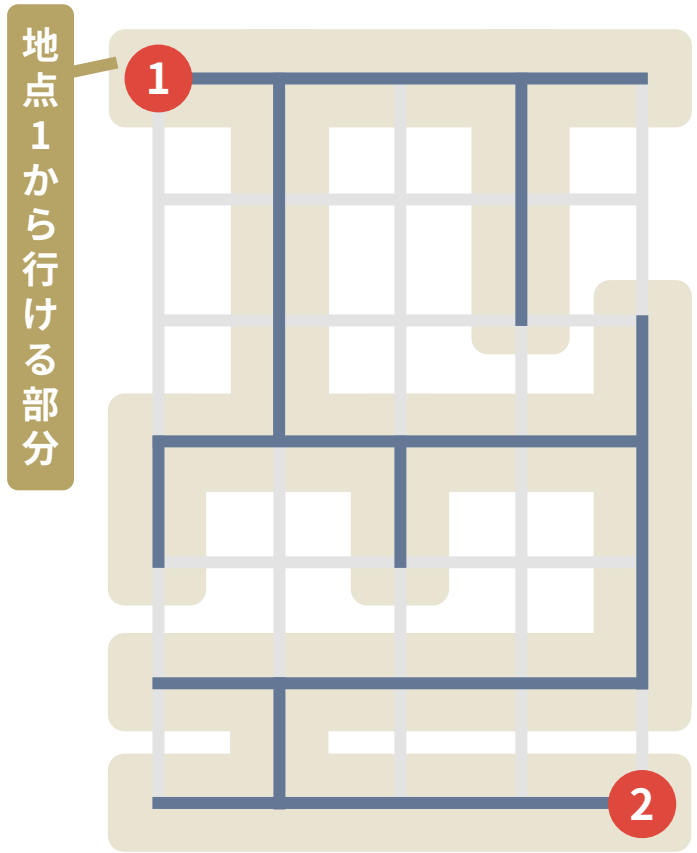
小課題 1, 2: 具体例 A



例: 下図の 1 と 2 を連結にすることを考える

- 現時点で地点 1 からは 1 行目まで到達可能 → 1 行目を整備
- 現時点で地点 1 からは 4 行目まで到達可能 → 4 行目を整備
- 現時点で地点 1 からは 6 行目まで到達可能 → 6 行目を整備
- 現時点で地点 1 からは 7 行目まで到達可能 → 7 行目を整備

小課題 1, 2: 具体例 A

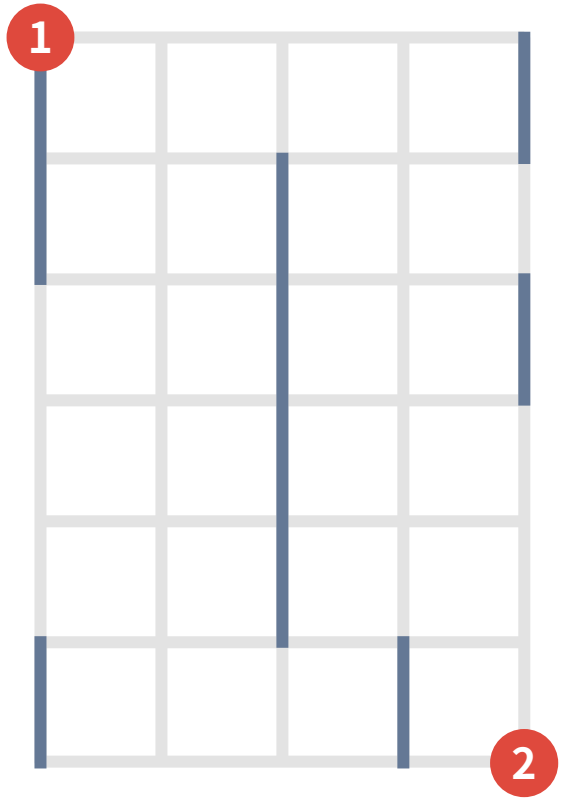


例: 下図の 1 と 2 を連結にすることを考える

- 現時点で地点 1 からは 1 行目まで到達可能 → 1 行目を整備
- 現時点で地点 1 からは 4 行目まで到達可能 → 4 行目を整備
- 現時点で地点 1 からは 6 行目まで到達可能 → 6 行目を整備
- 現時点で地点 1 からは 7 行目まで到達可能 → 7 行目を整備

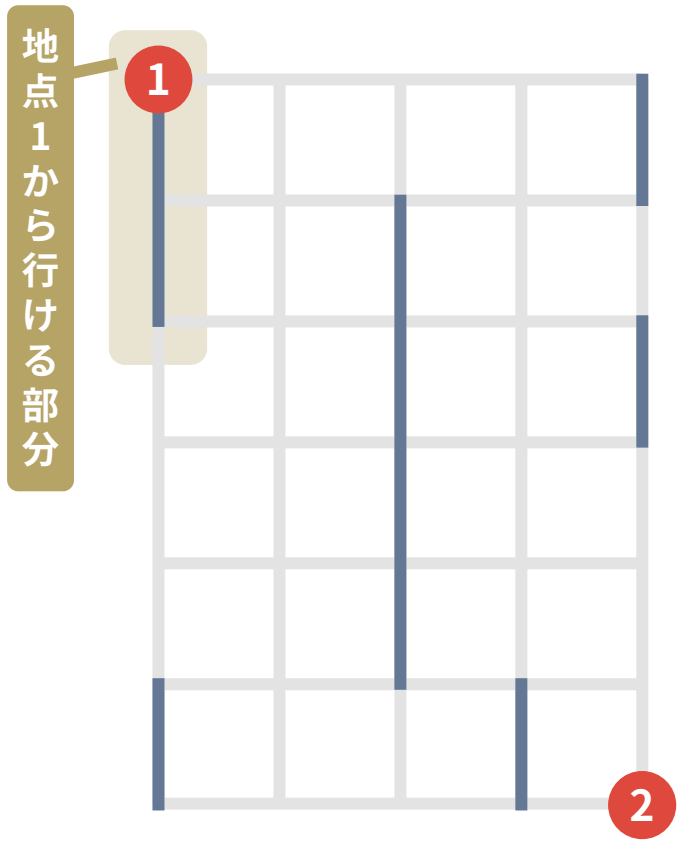
4 回の整備で連結にできた！
(これが最適解)

小課題 1, 2: 具体例 B



例: 下図の 1 と 2 を連結にすることを考える

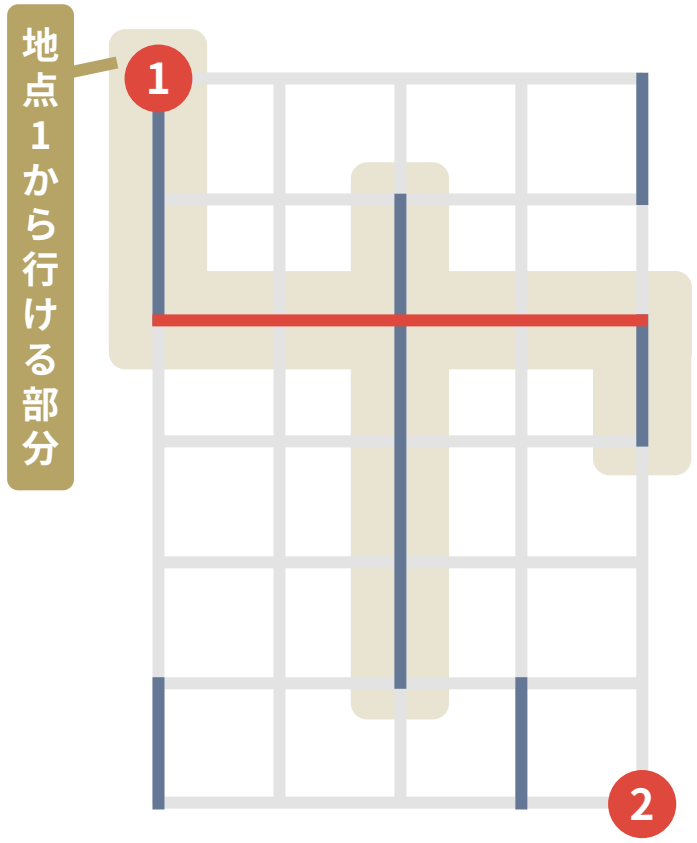
小課題 1, 2: 具体例 B



例: 下図の 1 と 2 を連結にすることを考える

- 現時点で地点 1 からは 3 行目まで到達可能 → 3 行目を整備

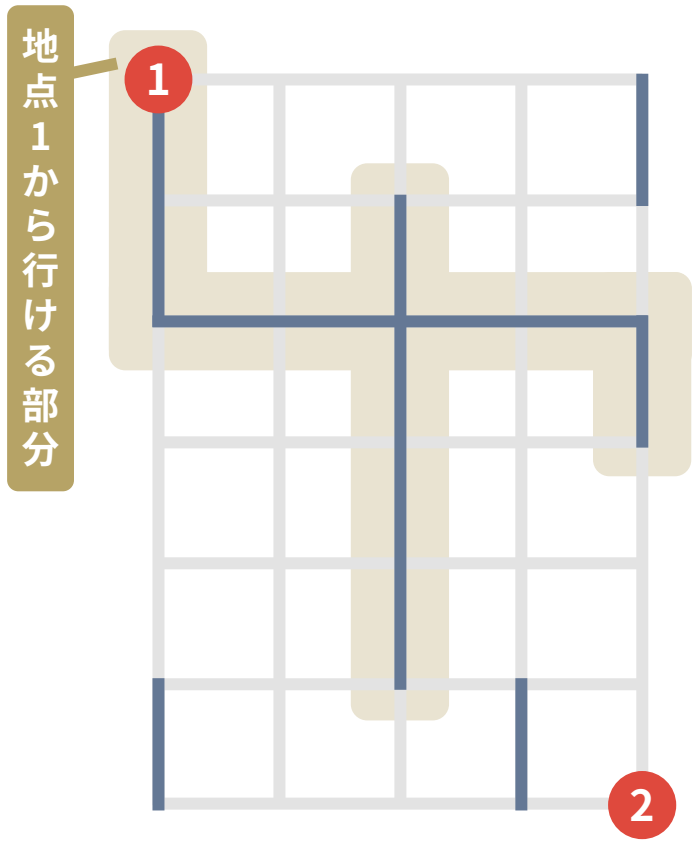
小課題 1, 2: 具体例 B



例: 下図の 1 と 2 を連結にすることを考える

- 現時点で地点 1 からは 3 行目まで到達可能 → 3 行目を整備

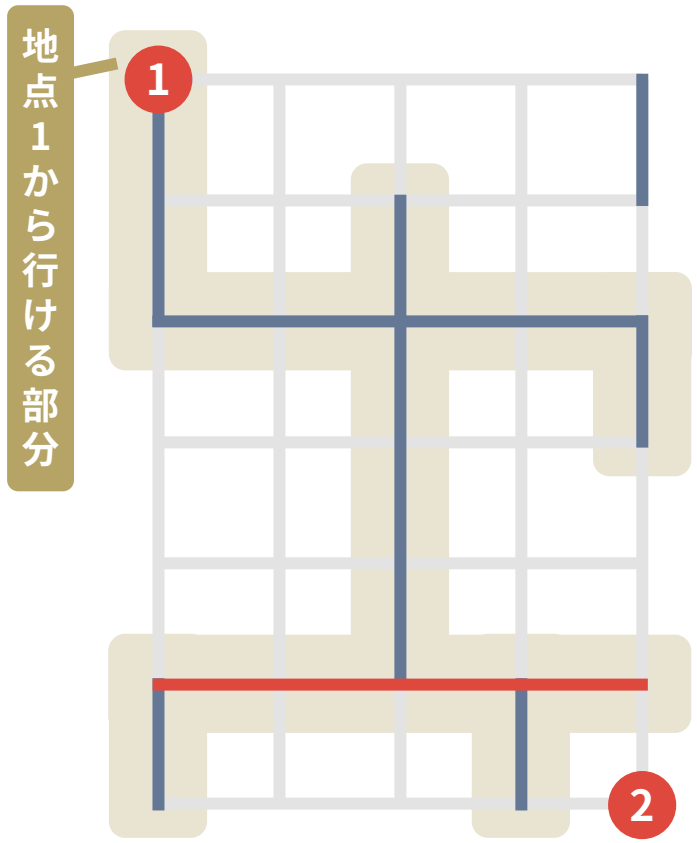
小課題 1, 2: 具体例 B



例: 下図の 1 と 2 を連結にすることを考える

- 現時点で地点 1 からは 3 行目まで到達可能 → 3 行目を整備
- 現時点で地点 1 からは 6 行目まで到達可能 → 6 行目を整備

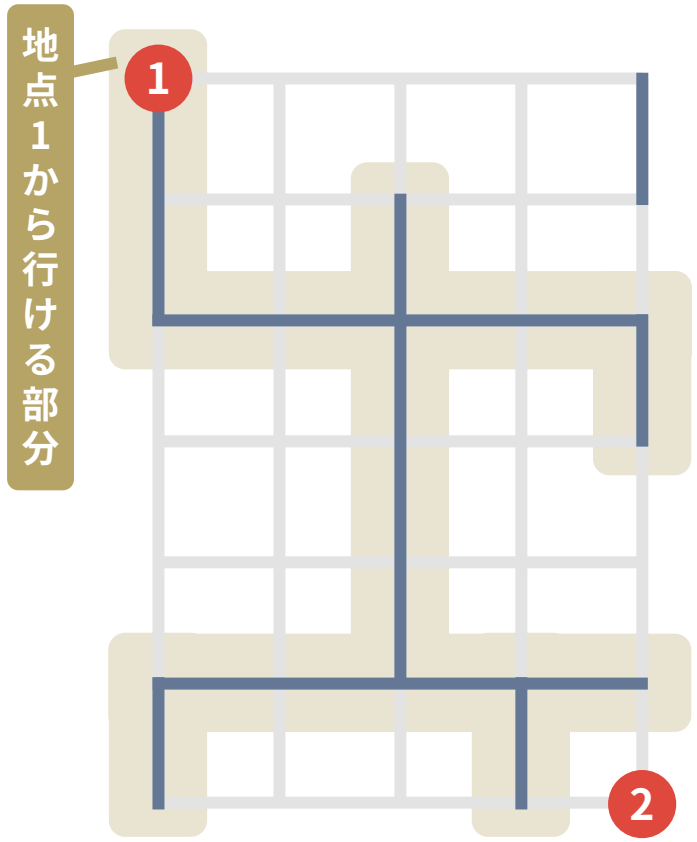
小課題 1, 2: 具体例 B



例: 下図の 1 と 2 を連結にすることを考える

- 現時点で地点 1 からは 3 行目まで到達可能 → 3 行目を整備
- 現時点で地点 1 からは 6 行目まで到達可能 → 6 行目を整備

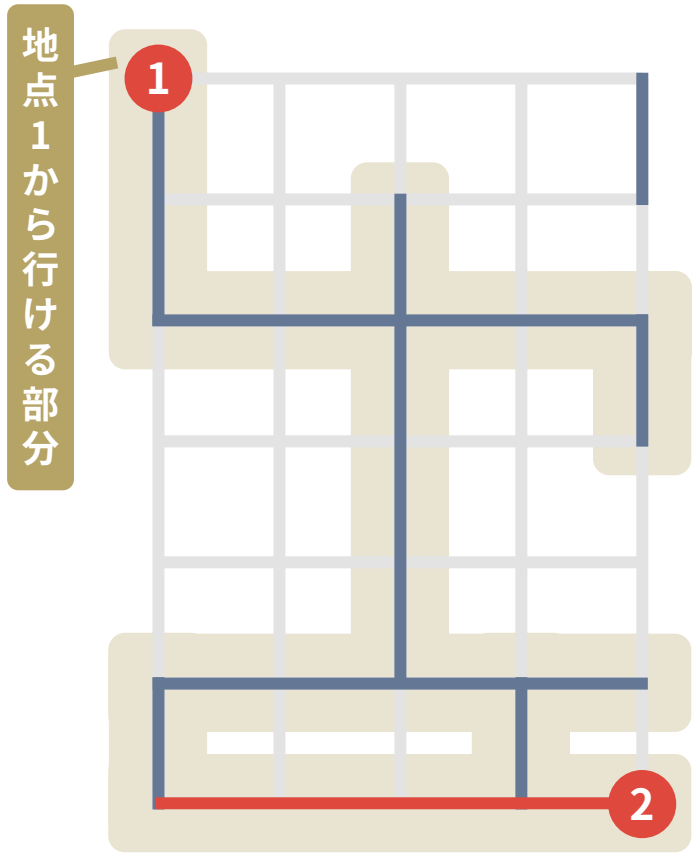
小課題 1, 2: 具体例 B



例: 下図の 1 と 2 を連結にすることを考える

- 現時点で地点 1 からは 3 行目まで到達可能 → 3 行目を整備
- 現時点で地点 1 からは 6 行目まで到達可能 → 6 行目を整備
- 現時点で地点 1 からは 7 行目まで到達可能 → 7 行目を整備

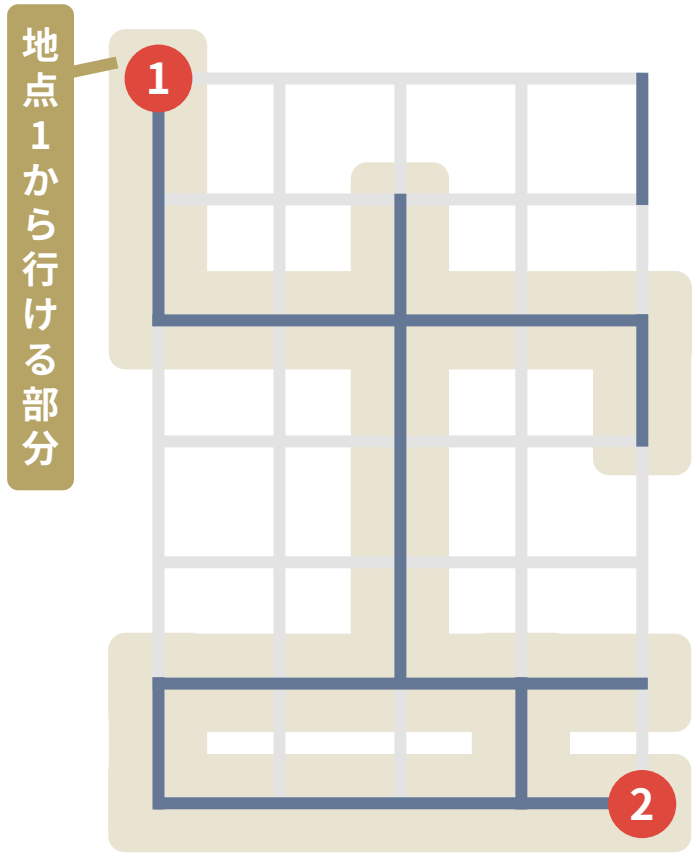
小課題 1, 2: 具体例 B



例: 下図の 1 と 2 を連結にすることを考える

- 現時点で地点 1 からは 3 行目まで到達可能 → 3 行目を整備
- 現時点で地点 1 からは 6 行目まで到達可能 → 6 行目を整備
- 現時点で地点 1 からは 7 行目まで到達可能 → 7 行目を整備

小課題 1, 2: 具体例 B



例: 下図の 1 と 2 を連結にすることを考える

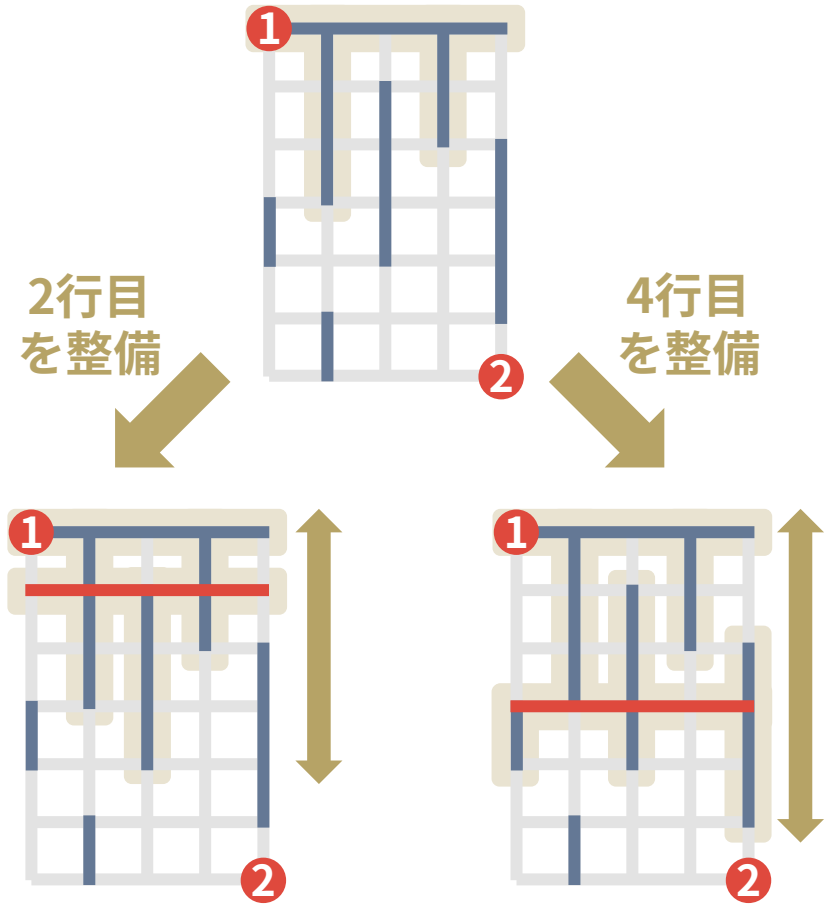
- 現時点で地点 1 からは 3 行目まで到達可能 → 3 行目を整備
- 現時点で地点 1 からは 6 行目まで到達可能 → 6 行目を整備
- 現時点で地点 1 からは 7 行目まで到達可能 → 7 行目を整備

3 回の整備で連結にできた！
(これが最適解)

最適性の証明

一番下を整備し続けるのが最適な理由

- 直感的には、下の行を整備したほうが、到達可能な行が広がるため
- たとえば右図で 2 行目を整備した場合と 4 行目を整備した場合では、後者の方が到達可能な行が広い

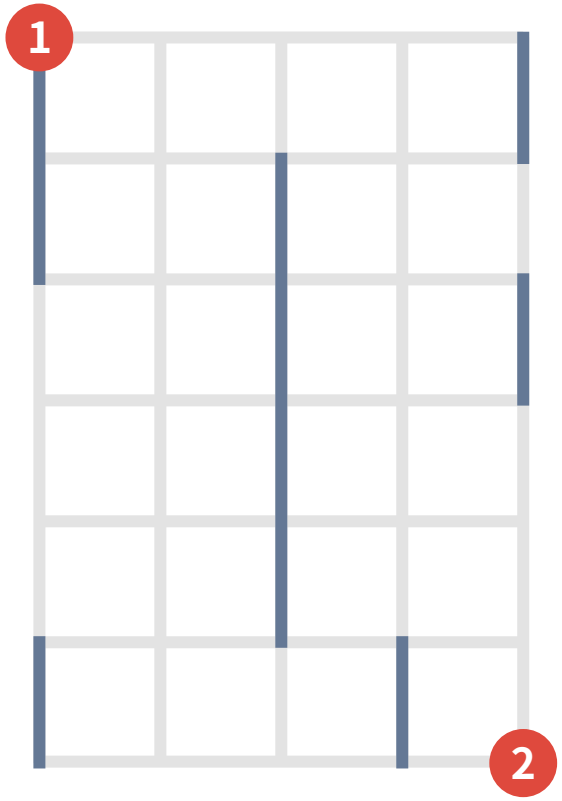


小課題 1, 2

具体的にどう実装するか？

小課題 1, 2

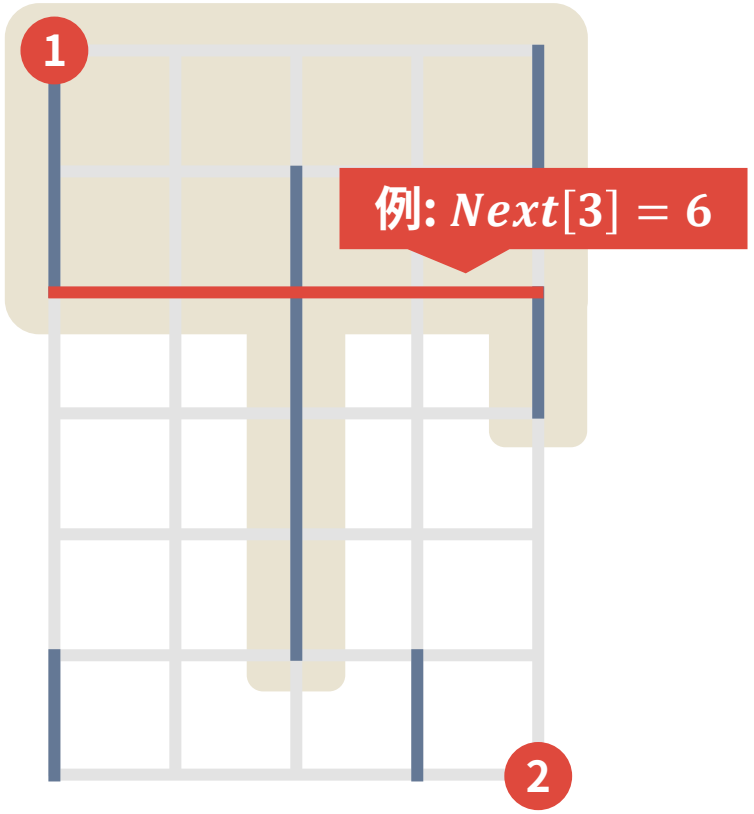
- i 行目まで到達可能なとき、i 行目を整備した場合に到達できる一番下の行 $Next[i]$ が求まれば実装は簡単※



※具体的には、i 行目に整備した後は $Next[i]$ 行目を整備すれば良い

小課題 1, 2

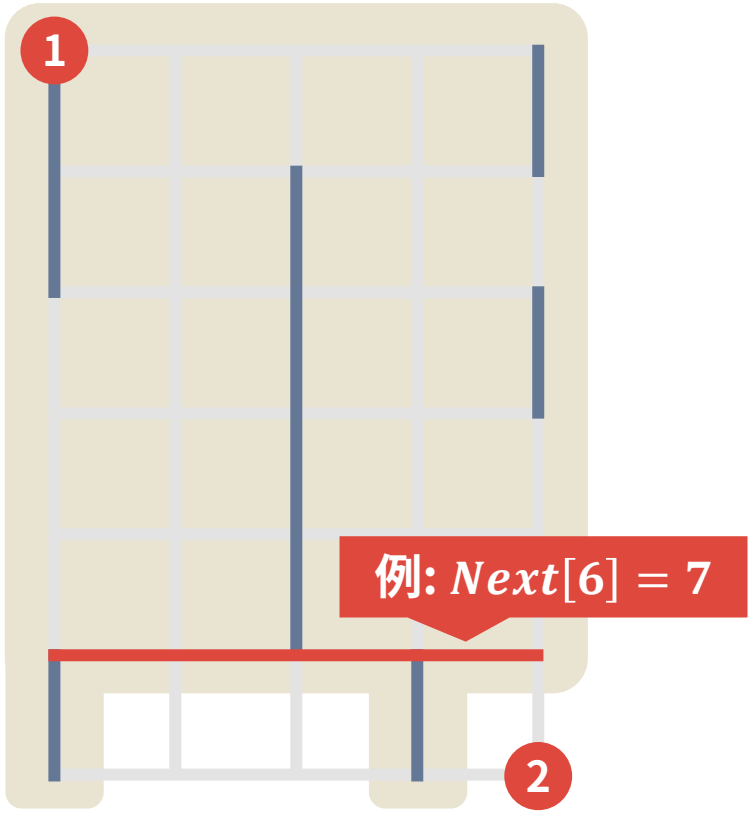
- i 行目まで到達可能なとき、i 行目を整備した場合に到達できる一番下の行 $Next[i]$ が求まれば実装は簡単※



※具体的には、i 行目に整備した後は $Next[i]$ 行目を整備すれば良い

小課題 1, 2

- i 行目まで到達可能なとき、i 行目を整備した場合に到達できる一番下の行 $Next[i]$ が求まれば実装は簡単※



※具体的には、i 行目に整備した後は $Next[i]$ 行目を整備すれば良い

小課題 1, 2

そこで $Next[i]$ は次のように計算できる

小課題 1, 2

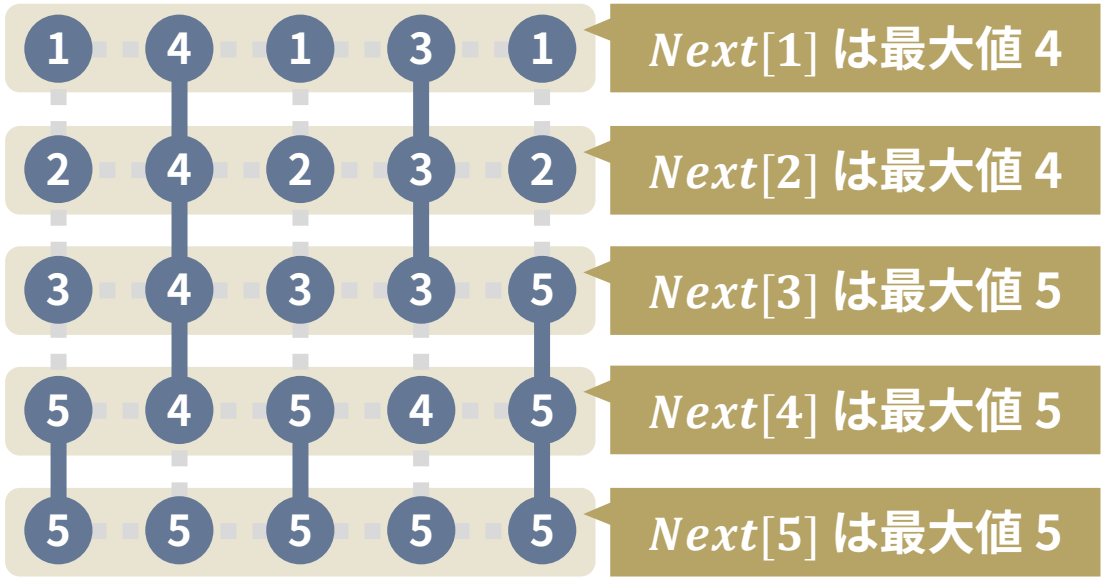
ステップ 1

各交差点について、この場所から到達できる一番下の行を計算
幅優先探索などで $O(HW)$ で計算できる

小課題 1, 2

ステップ 2

そこで、$Next[i]$ は i 行目の数字の最大値である
→ $O(HW)$ で全部計算できる



小課題 1, 2 まとめ

以下のような方針で小課題 1, 2 を解くことができた

- i 行目を整備した場合に到達できる最大の行 $Next[i]$ を計算
- $Next[i]$ をもとに「一番下を整備する貪欲法」を適用

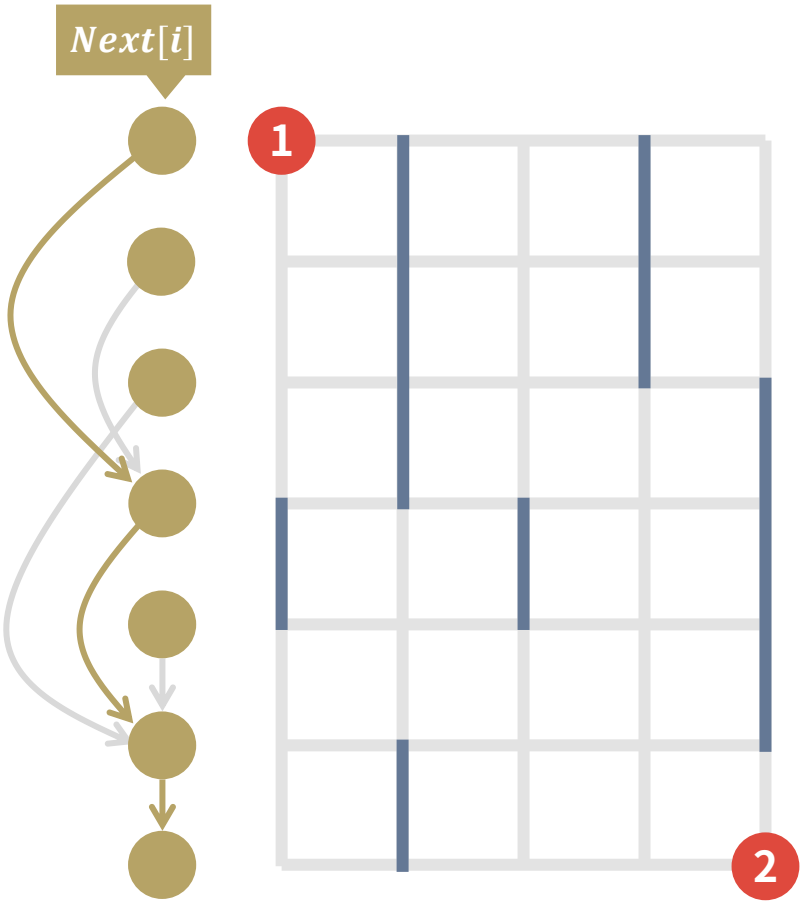
小課題 4

$C_i = 1, T = 2$

クエリはたくさん

小課題 4

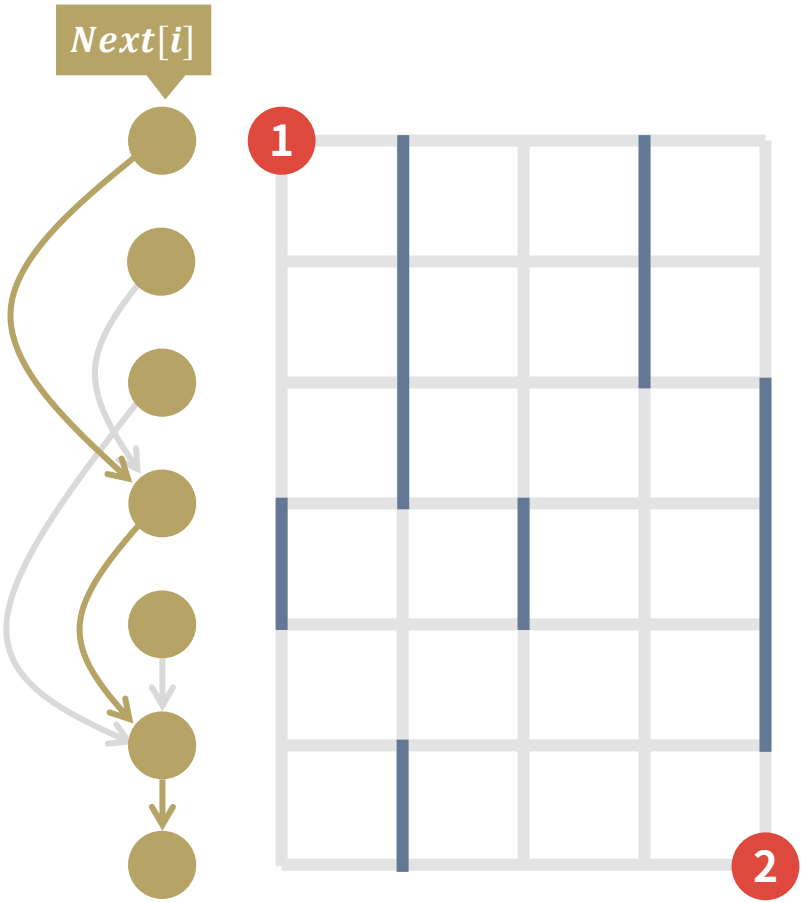
- 本問題の答えは、i 行目から $Next[i]$ 行目まで 1 手で行けるとき、a 行目から b 行目まで何手で行けるかとなる※
- たとえば右図の場合、1 行目から 7 行目まで何手で行けるかが答えとなる



※厳密には、この値に 1 を足した値 (最初と最後の整備を考える必要があるため)

小課題 4

- 本問題の答えは、i 行目から $Next[i]$ 行目まで 1 手で行けるとき、a 行目から b 行目まで何手で行けるかとなる※
- たとえば右図の場合、1 行目から 7 行目まで何手で行けるかが答えとなる



➡ **しかし、$Q \leq 100000$ なので**
自然に実装すると
$(H, W) = (500000, 2)$ などのケースで TLE

※厳密には、この値に 1 を足した値 (最初と最後の整備を考える必要があるため)

小課題 4

- 本問題の答えは、i 行目から $Next[i]$ 行目まで 1 手で行けるとき、a 行目から b 行目まで何手で行けるかとなる※
- たとえば右図の場合、1 行目から 7 行目まで何手で行けるかが答えとなる

➡ しかし、自然に実装すると $(H, W) = (500000, 2)$ などのケースで TLE

ダブリングを使おう！

※厳密には、この値に 1 を足した値（最初と最後の頂点を含める必要があるため）

ダブリングとは

- ダブリングとは、1, 2, 4, 8, 16, … 手先を前計算することで、n 手先を高速に計算するテクニック

ダブリングとは

- ダブリングとは、1, 2, 4, 8, 16, … 手先を前計算することで、n 手先を高速に計算するテクニック
- 例として、以下のような場合について、地点 1 から地点 8 まで何手で行けるかを計算することを考える

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10

ダブリングとは: 具体例

ステップ 1　2 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	??	??	??	??	??	??	??	??	??	??
4 手先	??	??	??	??	??	??	??	??	??	??

ダブリングとは: 具体例

ステップ 1　2 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	??	??	??	??	??	??	??	??	??
4 手先	??	??	??	??	??	??	??	??	??	??

1 の 1 手先は 2
2 の 1 手先は 4

ダブリングとは: 具体例

ステップ 1　2 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	??	??	??	??	??	??	??	??
4 手先	??	??	??	??	??	??	??	??	??	??

2 の 1 手先は 4
4 の 1 手先は 6

ダブリングとは: 具体例

ステップ 1　2 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	??	??	??	??	??	??	??
4 手先	??	??	??	??	??	??	??	??	??	??

3 の 1 手先は 4
4 の 1 手先は 6

ダブリングとは: 具体例

ステップ 1　2 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	??	??	??	??	??	??
4 手先	??	??	??			??	??	??	??	??

4 の 1 手先は 6
6 の 1 手先は 7

ダブリングとは: 具体例

ステップ 1　2 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	??	??	??	??	??
4 手先	??	??	??	??	??	??	??	??	??	??

5 の 1 手先は 7
7 の 1 手先は 8

ダブリングとは: 具体例

ステップ 1　2 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	??	??	??	??
4 手先	??	??	??	??	??	??	??	??	??	??

6 の 1 手先は 7
7 の 1 手先は 8

ダブリングとは: 具体例

ステップ 1　2 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	9	??	??	??
4 手先	??	??	??	??	??	??	??	??	??	??

7 の 1 手先は 8
8 の 1 手先は 9

ダブリングとは: 具体例

ステップ 1　2 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	9	10	??	??
4 手先	??	??	??	??	??	??	??	??	??	??

8 の 1 手先は 9
9 の 1 手先は 10

ダブリングとは: 具体例

ステップ 1　2 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	9	10	10	??
4 手先	??	??	??	??	??	??	??	??	??	??

9 の 1 手先は 10
10 の 1 手先は 10

ダブリングとは: 具体例

ステップ 1　2 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	9	10	10	10
4 手先	??	??	??	??	??	??	??	??	??	??

10 の 1 手先は 10
10 の 1 手先は 10

ダブリングとは: 具体例

ステップ 2　4 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	9	10	10	10
4 手先	??	??	??	??	??	??	??	??	??	??

ダブリングとは: 具体例

ステップ 2　4 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	9	10	10	10
4 手先	7	??	??	??	??	??	??	??	??	??

1 の 2 手先は 4
4 の 2 手先は 7

ダブリングとは: 具体例

ステップ 2　4 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	9	10	10	10
4 手先	7	8	??	??	??	??	??	??	??	??

2 の 2 手先は 6
6 の 2 手先は 8

ダブリングとは: 具体例

ステップ 2　4 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	9	10	10	10
4 手先	7	8	8	??	??	??	??	??	??	??

3 の 2 手先は 6
6 の 2 手先は 8

ダブリングとは: 具体例

ステップ 2　4 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	9	10	10	10
4 手先	7	8	8	9	??	??	??	??	??	??

4 の 2 手先は 7
7 の 2 手先は 9

ダブリングとは: 具体例

ステップ 2　4 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	9	10	10	10
4 手先	7	8	8	9	10	??	??	??	??	??

5 の 2 手先は 8
8 の 2 手先は 10

ダブリングとは: 具体例

ステップ 2　4 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	9	10	10	10
4 手先	7	8	8	9	10	10	??	??	??	??

6 の 2 手先は 8
8 の 2 手先は 10

ダブリングとは: 具体例

ステップ 2　4 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	9	10	10	10
4 手先	7	8	8	9	10	10	10	??	??	??

7 の 2 手先は 9
9 の 2 手先は 10

ダブリングとは: 具体例

ステップ 2　4 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	9	10	10	10
4 手先	7	8	8	9	10	10	10	10	??	??

8 の 2 手先は 10
10 の 2 手先は 10

ダブリングとは: 具体例

ステップ 2　4 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	9	10	10	10
4 手先	7	8	8	9	10	10	10	10	10	??

9 の 2 手先は 10
10 の 2 手先は 10

ダブリングとは: 具体例

ステップ 2　4 手先を計算する

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	9	10	10	10
4 手先	7	8	8	9	10	10	10	10	10	10

9 の 2 手先は 10
10 の 2 手先は 10

ダブリングとは: 具体例

ステップ 3　地点 1 から地点 8 に行くのに何手かかるかを二分探索で計算

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	9	10	10	10
4 手先	7	8	8	9	10	10	10	10	10	10

ダブリングとは: 具体例

ステップ 3　地点 1 から地点 8 に行くのに何手かかるかを二分探索で計算

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	9	10	10	10
4 手先	7	8	8	9	10	10	10	10	10	10

地点 1 から 4 手で地点 7
→答えは 4 以上

ダブリングとは: 具体例

ステップ 3　地点 1 から地点 8 に行くのに何手かかるかを二分探索で計算

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	9	10	10	10
4 手先	7	8	8	9	10	10	10	10	10	10

地点 7 に進む

地点 1 から 4 手で地点 7
→答えは 4 以上

ダブリングとは: 具体例

ステップ 3　地点 1 から地点 8 に行くのに何手かかるかを二分探索で計算

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	9	10	10	10
4 手先	7	8	8	9	10	10	10	10	10	10

地点 7 に進む

地点 1 から 4 手で地点 7
→答えは 4 以上

地点 7 から 2 手で地点 9
→答えは 4+2 未満

ダブリングとは: 具体例

ステップ 3　地点 1 から地点 8 に行くのに何手かかるかを二分探索で計算

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8	9	10	10
2 手先	4	6	6	7	8	8	9	10	10	10
4 手先	7	8	8	9	10	10	10	10	10	10

地点 7 に進む

地点 1 から 4 手で地点 7
→答えは 4 以上

地点 7 から 2 手で地点 9
→答えは 4+2 未満

ダブリングとは: 具体例

ステップ 3　地点 1 から地点 8 に行くのに何手かかるかを二分探索で計算

i	1	2	3	4	5	6	7	8	9	10
1 手先 ($Next[i]$)	2	4	4	6	7	7	8			
2 手先	4	6	6	7	8		9	10	10	10
4 手先	7		8	9	10	10				10

地点 7 から 1 手で地点 8
→答えは 4+1 以上

地点 7 に進む

地点 7 から 2 手で地点 9
→答えは 4+2 未満

地点 1 から 4 手で地点 7
→答えは 4 以上

ダブリングとは: 具体例

答えは 5 だと分かった！

小課題 4

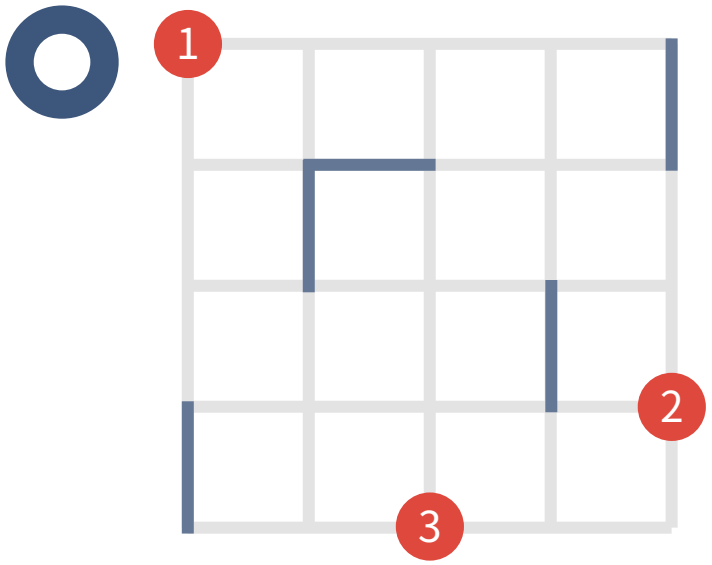
- このように、ダブリングの技法を使うと各クエリにつき $O(\log H)$ で解くことができる
- 全体の計算量は $O((H+Q)\log H)$ となり、小課題 3 に通る

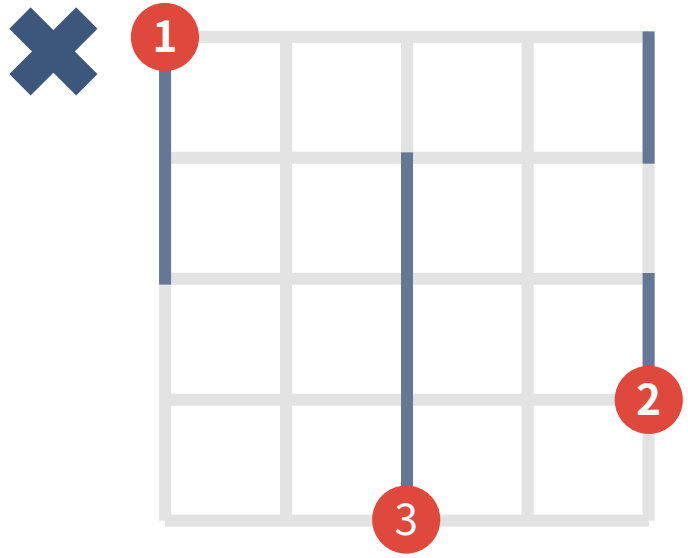
小課題 3, 5

$C_i = 1$

T が 3 以上になることもある

小課題 3, 5





いきなり一般の場合を考えるのは難しいので

連結にすべき地点が道と繋がっていない場合を考えよう

小課題 3, 5

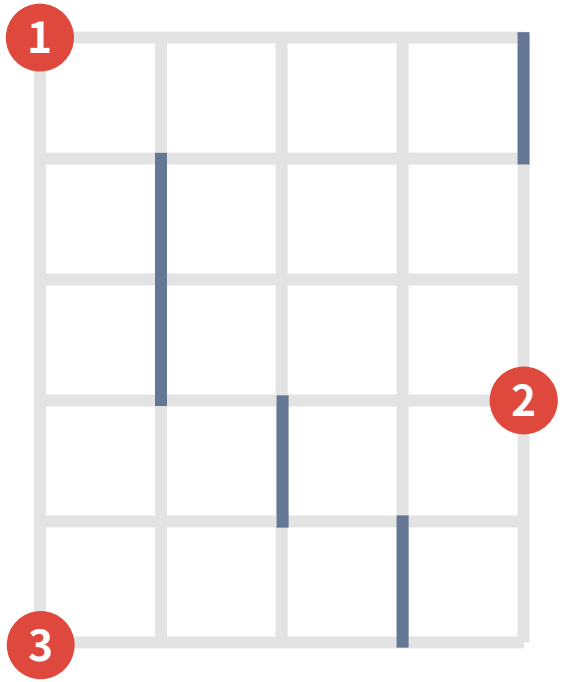
- 座標 $(X_1, Y_1), \ldots, (X_T, Y_T)$ (上から順) を連結にしたいが、これらの座標は道と繋がっていないとする
- このとき、$X_1, X_2, \ldots, X_T$ 行目は明らかに整備しなければならない

小課題 3, 5

- 座標 $(X_1, Y_1), \ldots, (X_T, Y_T)$ (上から順) を連結にしたいが、これらの座標は道と繋がっていないとする
- このとき、$X_1, X_2, \ldots, X_T$ 行目は明らかに整備しなければならない
- また、X_i 行目と X_{i+1} 行目の間の部分は独立に考えられるので、単純な足し算で答えが求められる

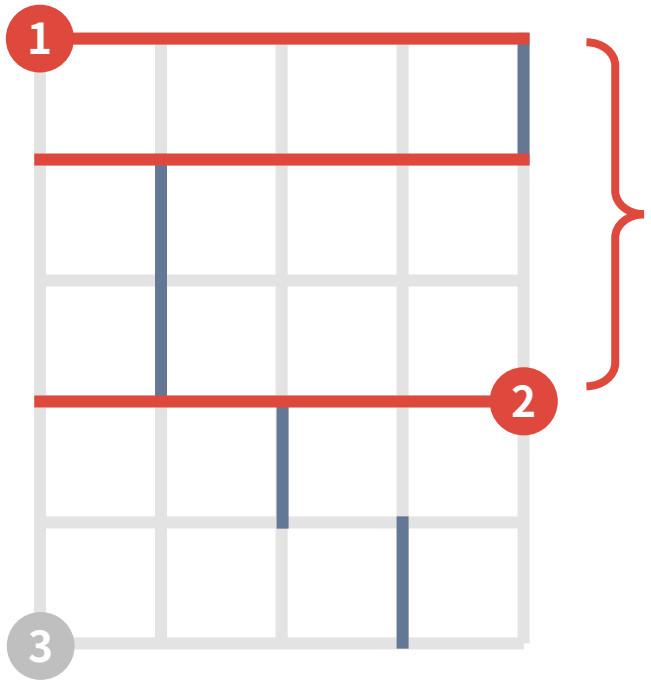
小課題 3, 5

具体例　下図の 1〜3 を連結にすることを考える



小課題 3, 5

具体例　下図の 1〜3 を連結にすることを考える

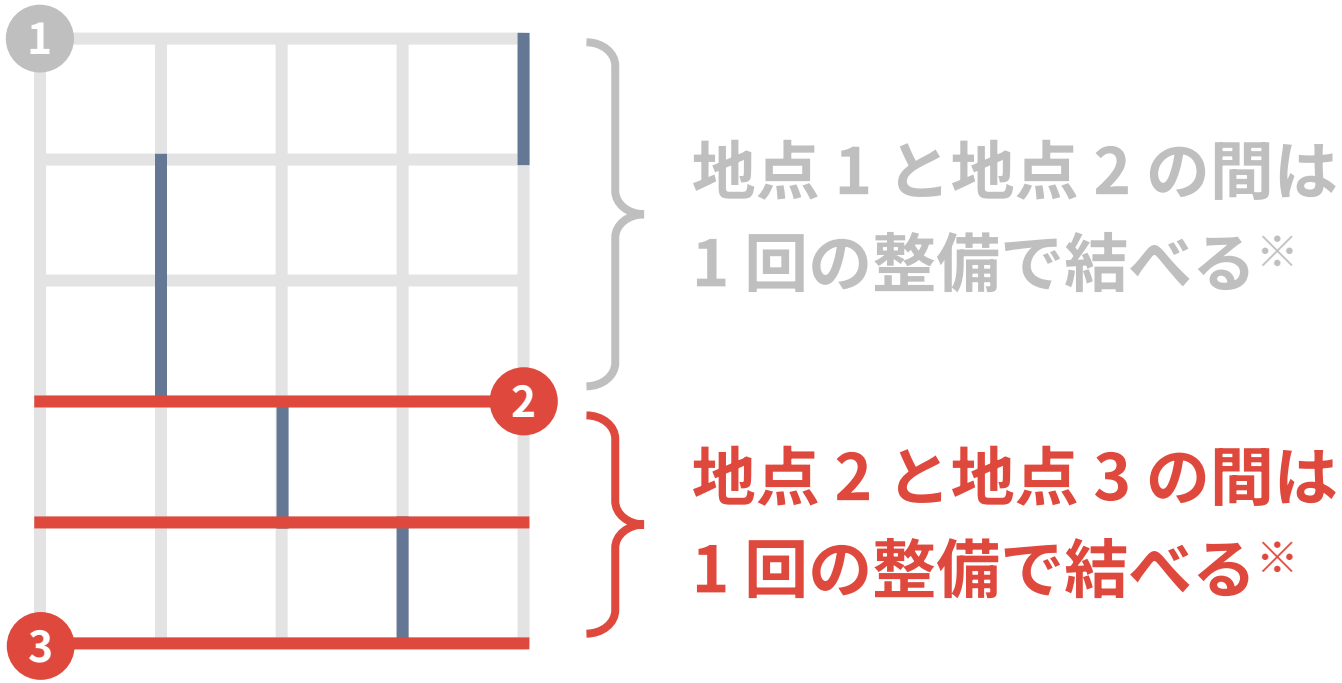


地点 1 と地点 2 の間は
1 回の整備で結べる※

※上端と下端を含まない

小課題 3, 5

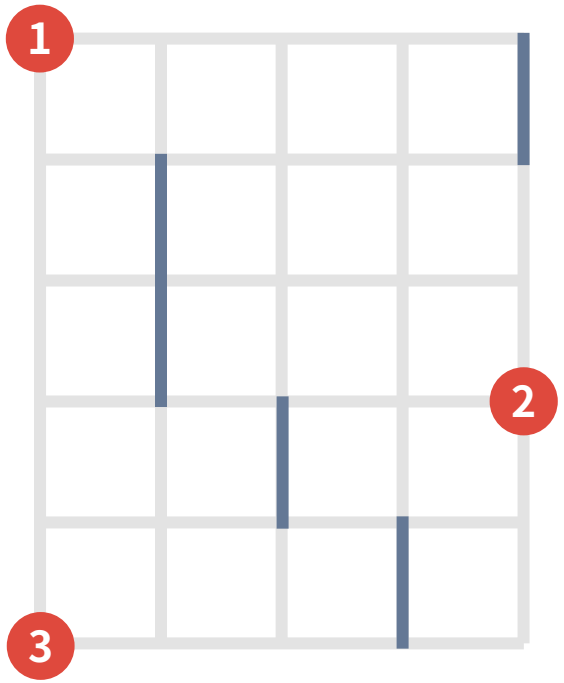
具体例　下図の 1〜3 を連結にすることを考える



※上端と下端を含まない

小課題 3, 5

具体例　下図の 1〜3 を連結にすることを考える



地点 1 と地点 2 の間は
1 回の整備で結べる※

地点 2 と地点 3 の間は
1 回の整備で結べる※

→ 全体では
3+1+1=5 回の整備

※上端と下端を含まない

小課題 3, 5

このように、道が繋がっていない場合は

単純な足し算で解くことができる

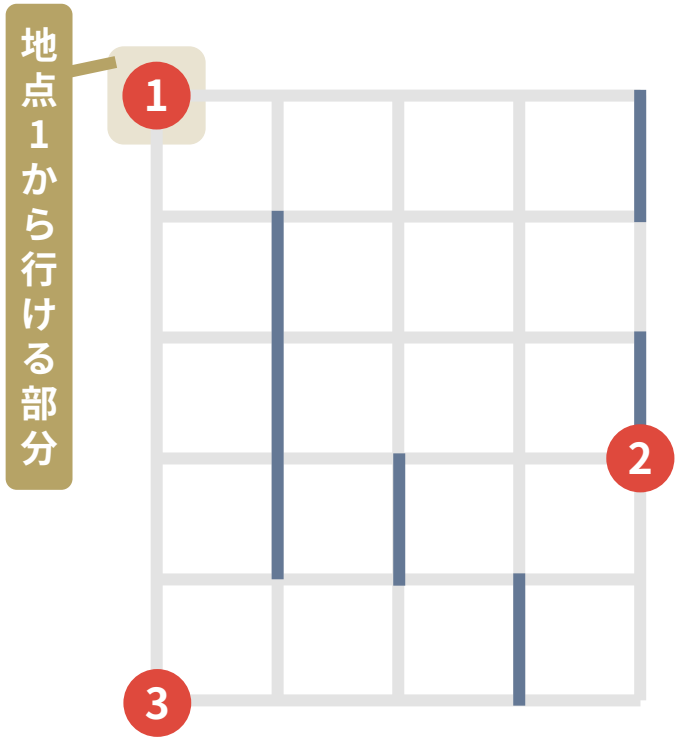
しかし、現実はそうではない

小課題 3, 5

- そこでクエリの連結成分が $[l_1, r_1], \ldots, [l_T, r_T]$ 行目を占めているとする
- このとき、次のような貪欲法で最適解を出すことができる
  - $\min(r_1, r_2, \ldots, r_T)$ 行目と $\max(l_1, l_2, \ldots, l_T)$ 行目を連結にすることを考える
  - 連結にするにあたっては小課題 3 と同様、できるだけ下を整備する貪欲法を使う。ただし $[l_1, r_1], [l_2, r_2], \ldots, [l_T, r_T]$ を一気に飛び越えないようにする

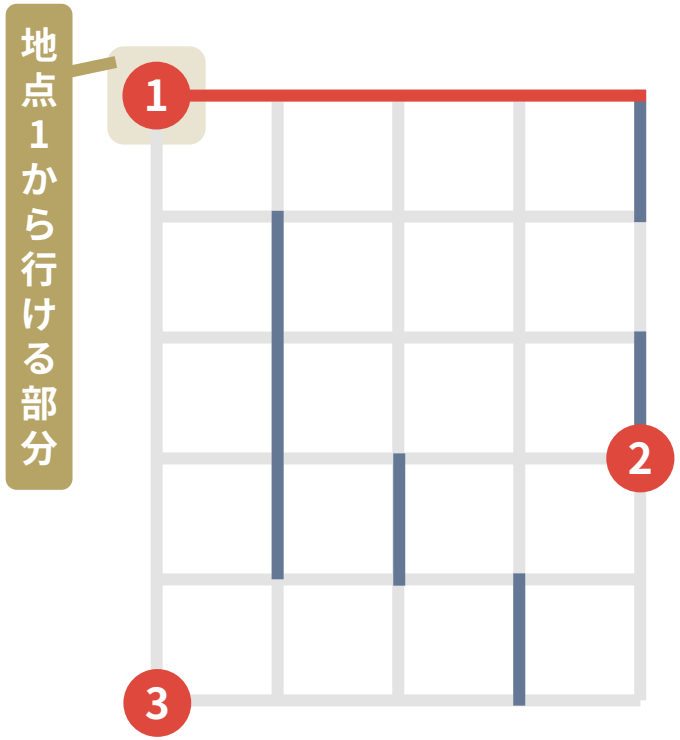
小課題 3, 5: 具体例

具体例　下図の 1〜3 を連結にする (つまり 1 行目から 6 行目に行きたい)



小課題 3, 5: 具体例

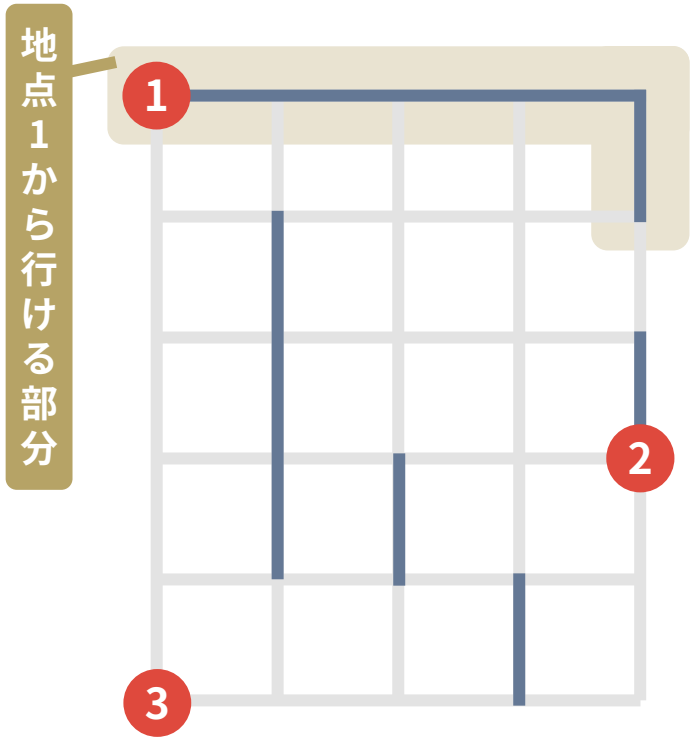
具体例　下図の 1〜3 を連結にする (つまり 1 行目から 6 行目に行きたい)



- 最初はスタート地点の 1 行目を整備する

小課題 3, 5: 具体例

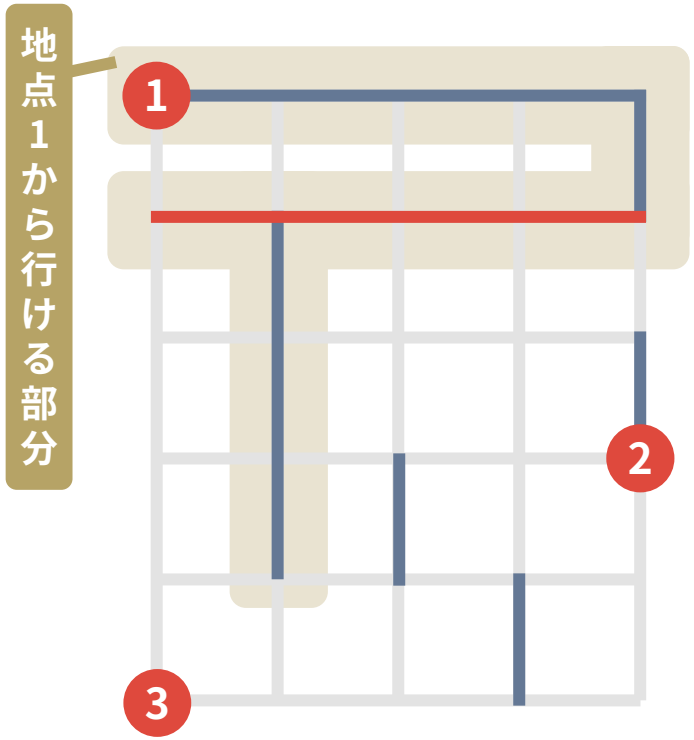
具体例　下図の 1〜3 を連結にする (つまり 1 行目から 6 行目に行きたい)



- 最初はスタート地点の 1 行目を整備する
- 現時点では 2 行目まで移動可能 → 2 行目を整備

小課題 3, 5: 具体例

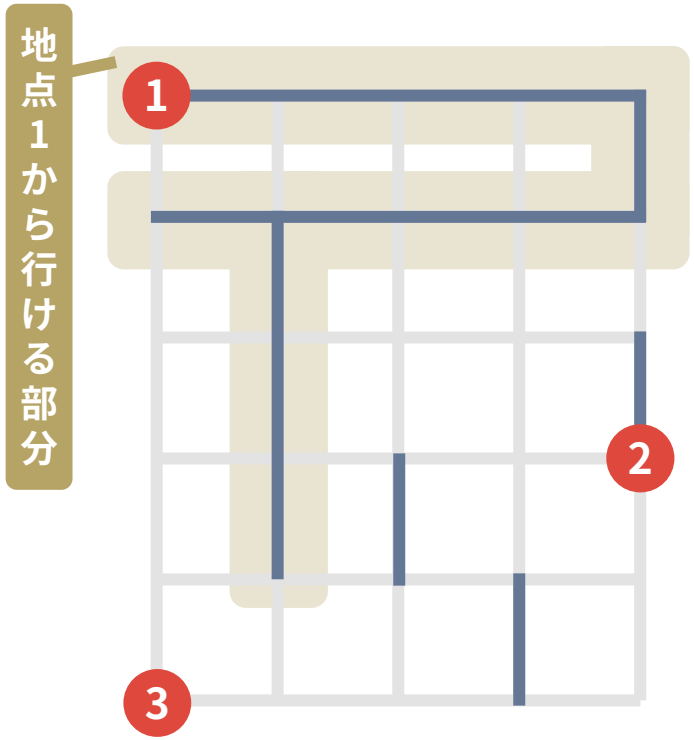
具体例　下図の 1〜3 を連結にする (つまり 1 行目から 6 行目に行きたい)



- 最初はスタート地点の 1 行目を整備する
- 現時点では 2 行目まで移動可能 → 2 行目を整備
- ここまでは貪欲法と同じ

小課題 3, 5: 具体例

具体例　下図の 1〜3 を連結にする (つまり 1 行目から 6 行目に行きたい)

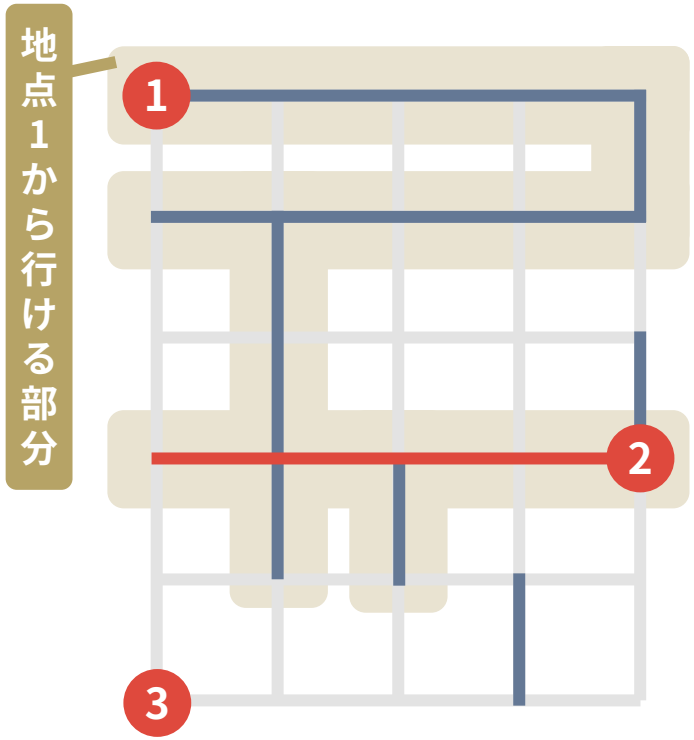


- 最初はスタート地点の 1 行目を整備する
- 現時点では 2 行目まで移動可能 → 2 行目を整備
- ここまでは貪欲法と同じ
- 現時点で 5 行目まで行けるが、5 行目を整備すると地点 2 の連結成分 (3〜4 行目) を飛び越えてしまう → 4 行目を整備

小課題 3, 5: 具体例

具体例

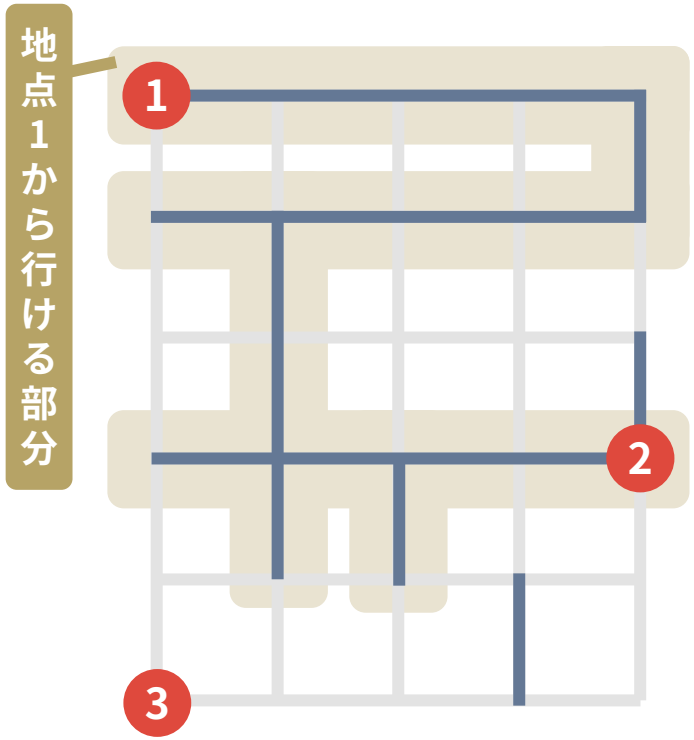
下図の 1〜3 を連結にする (つまり 1 行目から 6 行目に行きたい)



- 最初はスタート地点の 1 行目を整備する
- 現時点では 2 行目まで移動可能 → 2 行目を整備
- ここまでは貪欲法と同じ
- 現時点で 5 行目まで行けるが、5 行目を整備すると地点 2 の連結成分 (3〜4 行目) を飛び越えてしまう → 4 行目を整備

小課題 3, 5: 具体例

具体例　下図の 1〜3 を連結にする (つまり 1 行目から 6 行目に行きたい)

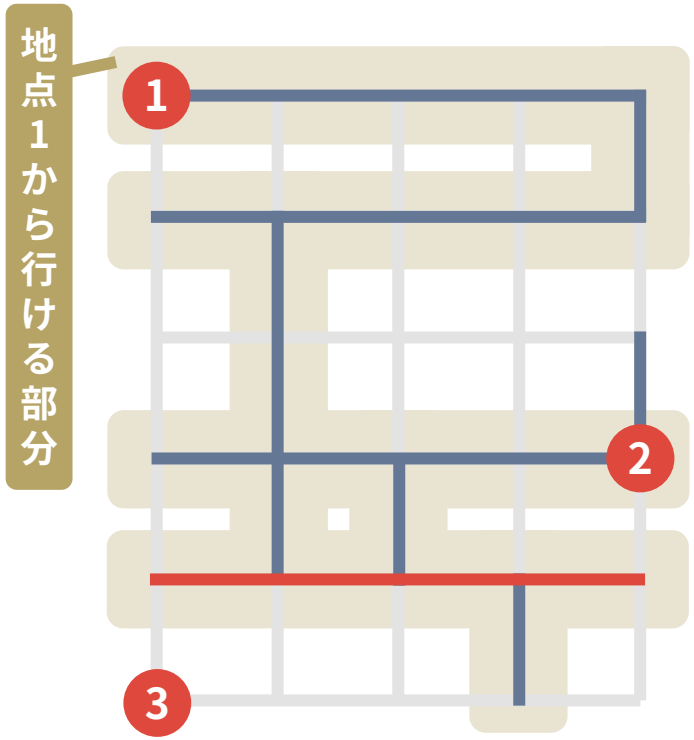


- 最初はスタート地点の 1 行目を整備する
- 現時点では 2 行目まで移動可能 → 2 行目を整備
- ここまでは貪欲法と同じ
- 現時点で 5 行目まで行けるが、5 行目を整備すると地点 2 の連結成分 (3〜4 行目) を飛び越えてしまう → 4 行目を整備
- 現時点で 5 行目まで行け、地点 2 の連結成分も一気には飛び越えない → 5 行目を整備

小課題 3, 5: 具体例

具体例

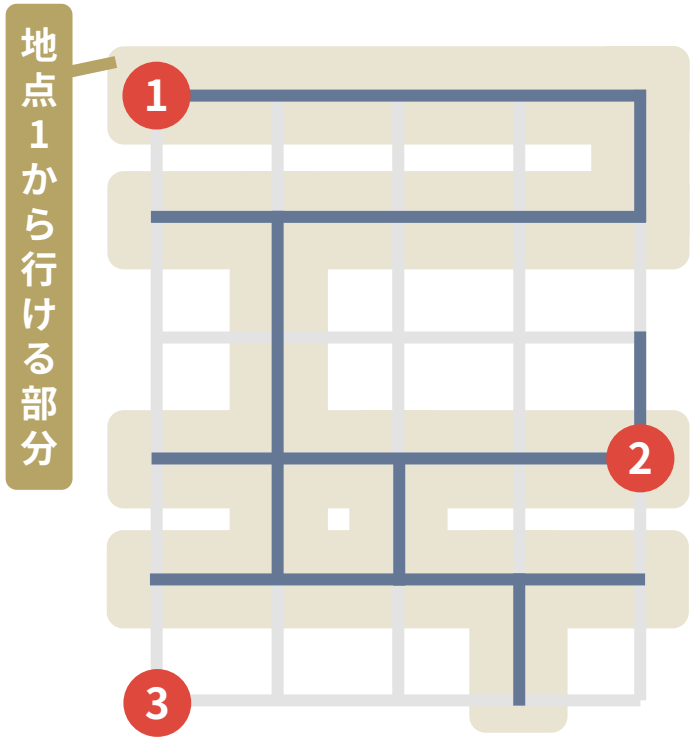
下図の 1〜3 を連結にする (つまり 1 行目から 6 行目に行きたい)



- 最初はスタート地点の 1 行目を整備する
- 現時点では 2 行目まで移動可能 → 2 行目を整備
- ここまでは貪欲法と同じ
- 現時点で 5 行目まで行けるが、5 行目を整備すると地点 2 の連結成分 (3〜4 行目) を飛び越えてしまう → 4 行目を整備
- 現時点で 5 行目まで行け、地点 2 の連結成分も一気には飛び越えない → 5 行目を整備

小課題 3, 5: 具体例

具体例　下図の 1〜3 を連結にする (つまり 1 行目から 6 行目に行きたい)

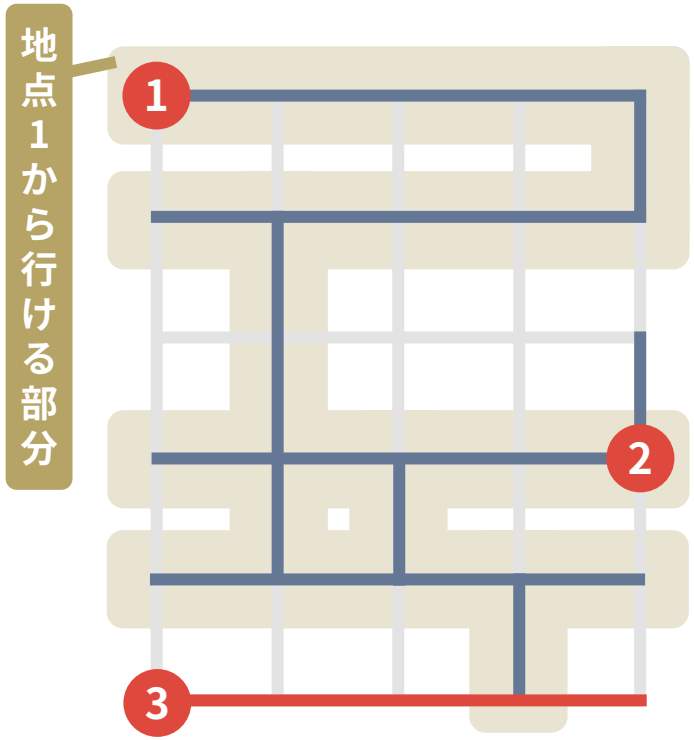


- 最初はスタート地点の 1 行目を整備する
- 現時点では 2 行目まで移動可能 → 2 行目を整備
- ここまでは貪欲法と同じ
- 現時点で 5 行目まで行けるが、5 行目を整備すると地点 2 の連結成分 (3〜4 行目) を飛び越えてしまう → 4 行目を整備
- 現時点で 5 行目まで行け、地点 2 の連結成分も一気には飛び越えない → 5 行目を整備
- 現時点で 6 行目まで行ける → 6 行目を整備

小課題 3, 5: 具体例

具体例

下図の 1〜3 を連結にする (つまり 1 行目から 6 行目に行きたい)



- 最初はスタート地点の 1 行目を整備する
- 現時点では 2 行目まで移動可能 → 2 行目を整備
- ここまでは貪欲法と同じ
- 現時点で 5 行目まで行けるが、5 行目を整備すると地点 2 の連結成分 (3〜4 行目) を飛び越えてしまう → 4 行目を整備
- 現時点で 5 行目まで行け、地点 2 の連結成分も一気には飛び越えない → 5 行目を整備
- 現時点で 6 行目まで行ける → 6 行目を整備

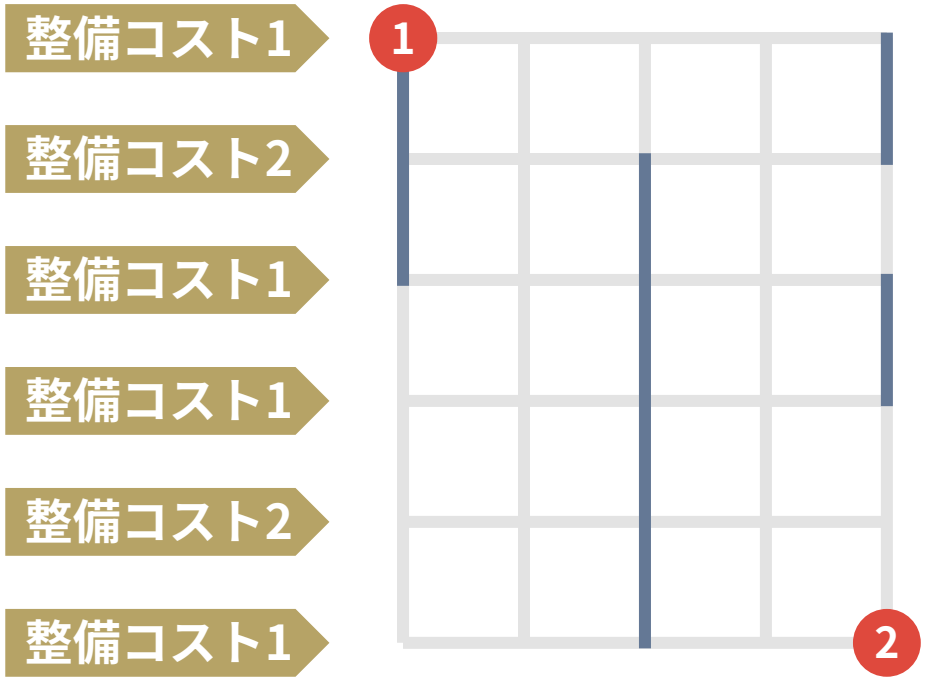
5 回で連結にできた！

小課題 6

$Q \leq 5$

$C_i = 1$ または $C_i = 2$

小課題 6 の注意



以降の小課題は、C_i の値が 2 になることもある

小課題 6

- $T = 2$ の場合を考える
- $PreCost1[i]$ を i 行目の直前の「コスト 1 の行」とする
- $PreCost2[i]$ を i 行目の直前の「コスト 2 の行」とする
- このとき、i 行目を整備したとき、次に整備すべき場所は以下の二択※
  - $PreCost1\left[Next[i]\right]$
  - $PreCost2\left[Next[i]\right]$

※ $Next[i]$ は i 行目を整備した時に到達できる最下行。小課題 1, 2 の解説を参照

小課題 6

したがって、以下の形の動的計画法をすると解ける

- $dp[i]$: 一番上の地点と i 行目が連結であり、i 行目を整備したときのコストの最小値

状態遷移

- $dp\left[PreCost1\left[Next[i]\right]\right] = \min\left(dp\left[PreCost1\left[Next[i]\right]\right], dp[i]+1\right)$
- $dp\left[PreCost2\left[Next[i]\right]\right] = \min\left(dp\left[PreCost2\left[Next[i]\right]\right], dp[i]+2\right)$

小課題 6

したがって、以下の形の動的計画法をすると解ける

- $dp[i]$: 一番上の地点と i 行目が連結であり、i 行目を整備したときのコストの最小値

状態遷移

- $dp\left[PreCost1\left[Next[i]\right]\right] = \min\left(dp\left[PreCost1\left[Next[i]\right]\right], dp[i]+1\right)$
- $dp\left[PreCost2\left[Next[i]\right]\right] = \min\left(dp\left[PreCost2\left[Next[i]\right]\right], dp[i]+2\right)$

$T \geq 3$ の場合も、クエリの区間 $[l_i, r_i]$ を一気に飛び越えないように注意して DP をすると解ける

60点

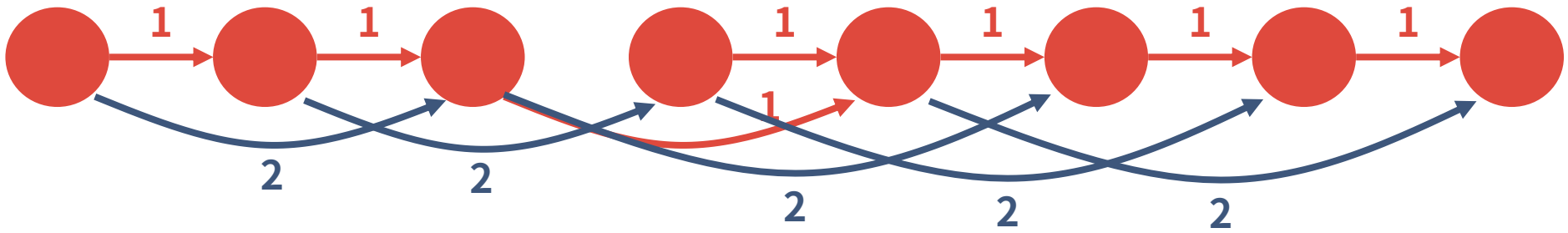
小課題 7

$T = 2$

クエリの制限は無し

小課題 7

- この小課題を解くには、以下のような問題を解く必要がある
  - 地点 i から $PreCost1[Next[i]]$ に行くのにコスト 1 かかる
  - 地点 i から $PreCost2[Next[i]]$ に行くのにコスト 2 かかる
  - 地点 a から地点 b 以上に行くのに最小でコストいくつ必要か？
  - $O(\log N)$ のオーダーで求めよ

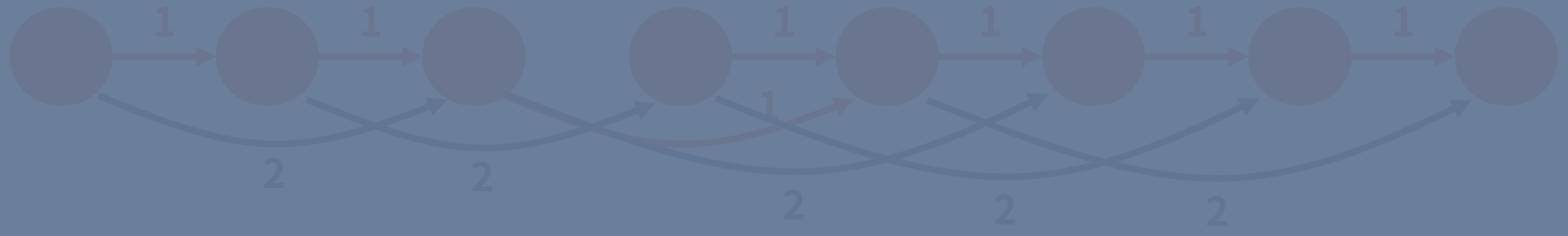


小課題 7

- この小課題を解くには、以下のような問題を解く必要がある
  - 地点 i から $PreCost1[Next[i]]$ に行くのにコスト 1 かかる
  - 地点 i から $PreCost2[Next[i]]$ に行くのにコスト 2 かかる
  - 地点 A から地点 B までの最小コストを求める必要がある
  - $O(\log N)$ のオーダーで求めよ

コスト 1 だとダブリングで解けるが…

コスト 2 ではどうするのか？



小課題 7: 考察

- 以下のようなダブリングを考える
  - 各地点 i から、コスト $2^k-1, 2^k+0, 2^k+1$ で行ける最大の場所を計算

小課題 7: 考察

- 以下のようなダブリングを考える
  - 各地点 i から、コスト $2^k-1, 2^k+0, 2^k+1$ で行ける最大の場所を計算

- これらの場所は、次のようにして計算できる
  - 2^k-1: (コスト $2^{k-1}-1$ + コスト $2^{k-1}+0$) または (コスト $2^{k-1}+0$ + コスト $2^{k-1}-1$)
  - 2^k+0: (コスト $2^{k-1}-1$ + コスト $2^{k-1}+1$) または (コスト $2^{k-1}+0$ + コスト $2^{k-1}+0$)
  - 2^k+1: (コスト $2^{k-1}+0$ + コスト $2^{k-1}+1$) または (コスト $2^{k-1}+1$ + コスト $2^{k-1}+0$)

小課題 7: 考察

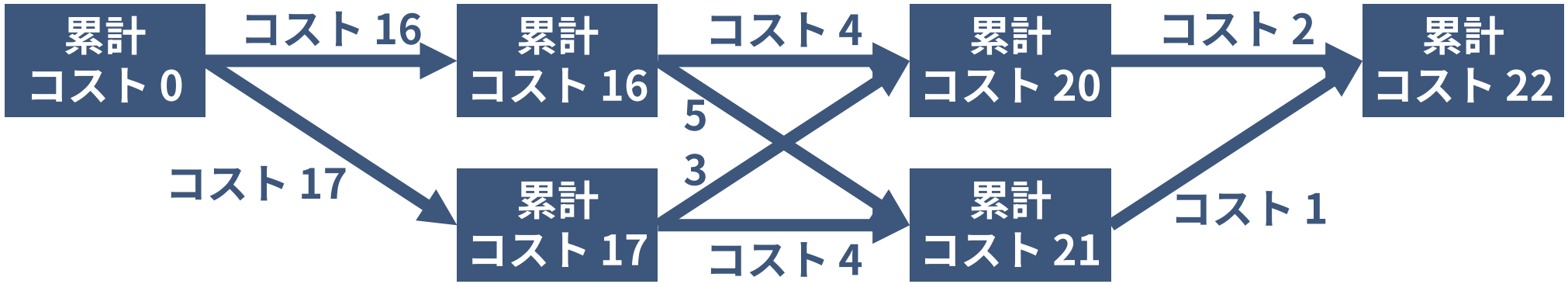
一般の x についてコスト x で行ける場所は
どうやって計算すれば良いか？

小課題 7: 具体例

- まずは例として、コスト 22 を計算したいとする (22=16+4+2)

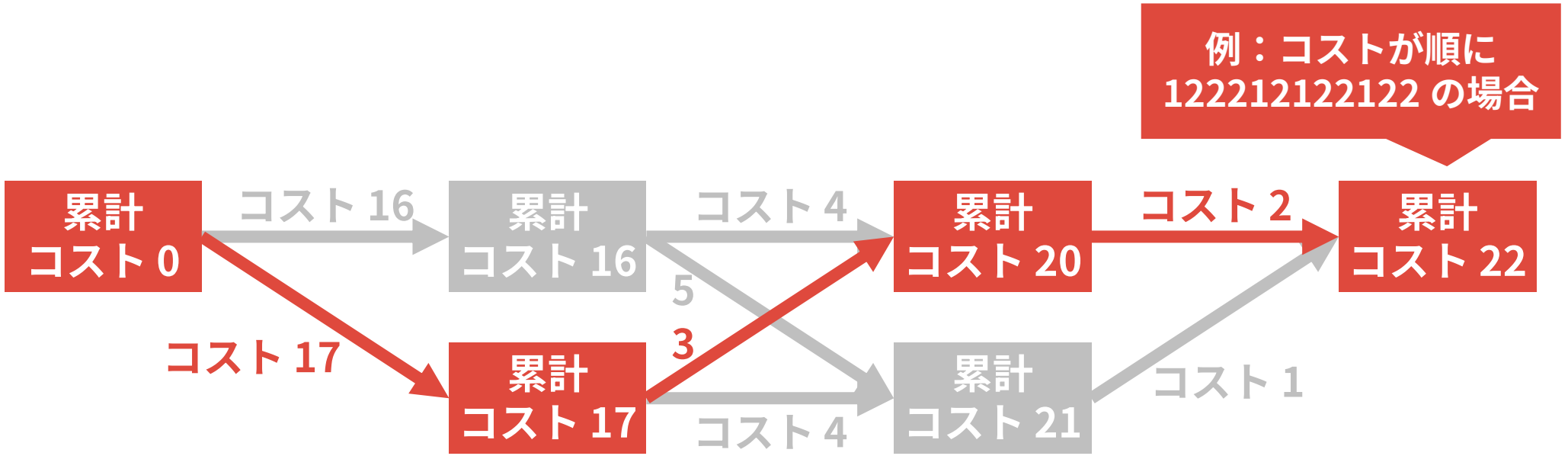
小課題 7: 具体例

- まずは例として、コスト 22 を計算したいとする (22=16+4+2)
- このとき、どのようなコスト 22 の経路も以下の形で表せる



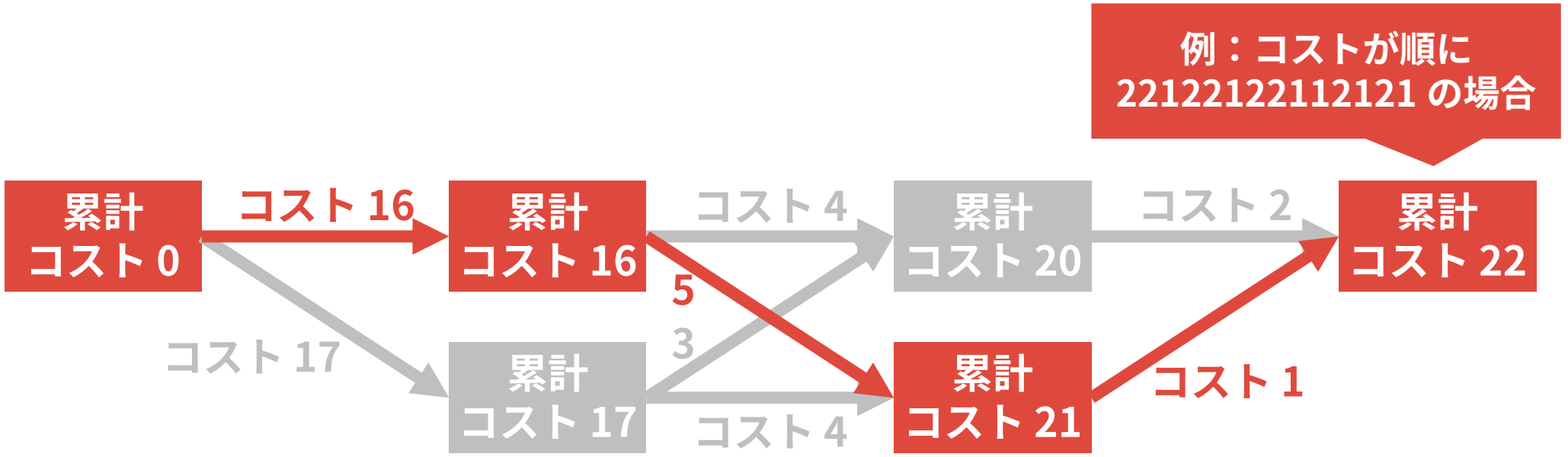
小課題 7: 具体例

- まずは例として、コスト 22 を計算したいとする (22=16+4+2)
- このとき、どのようなコスト 22 の経路も以下の形で表せる



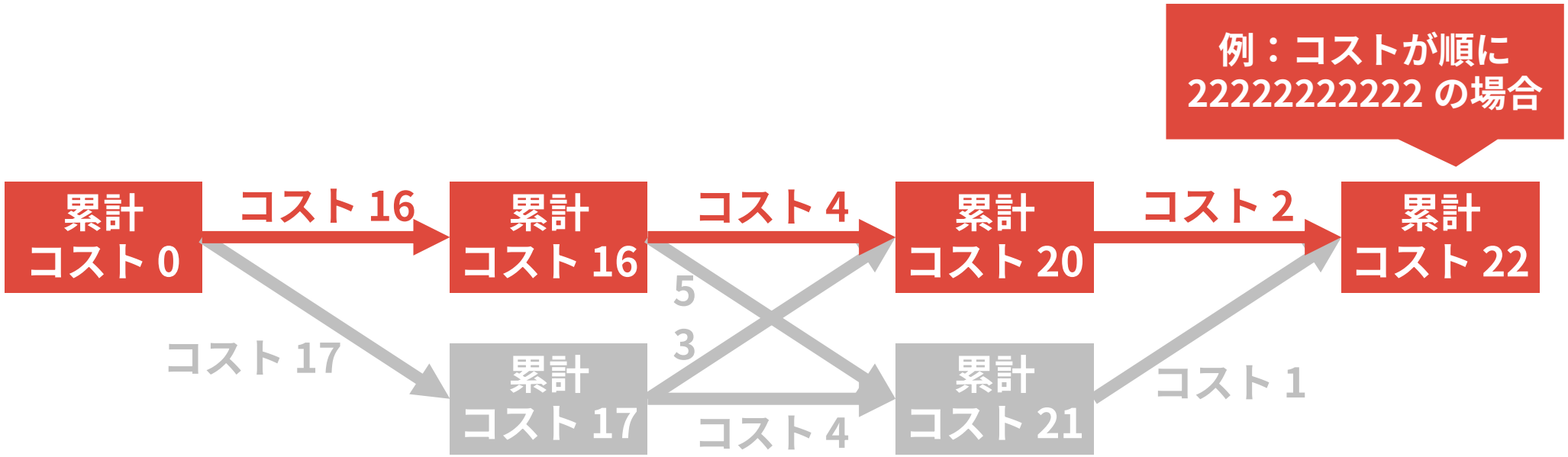
小課題 7: 具体例

- まずは例として、コスト 22 を計算したいとする (22=16+4+2)
- このとき、どのようなコスト 22 の経路も以下の形で表せる



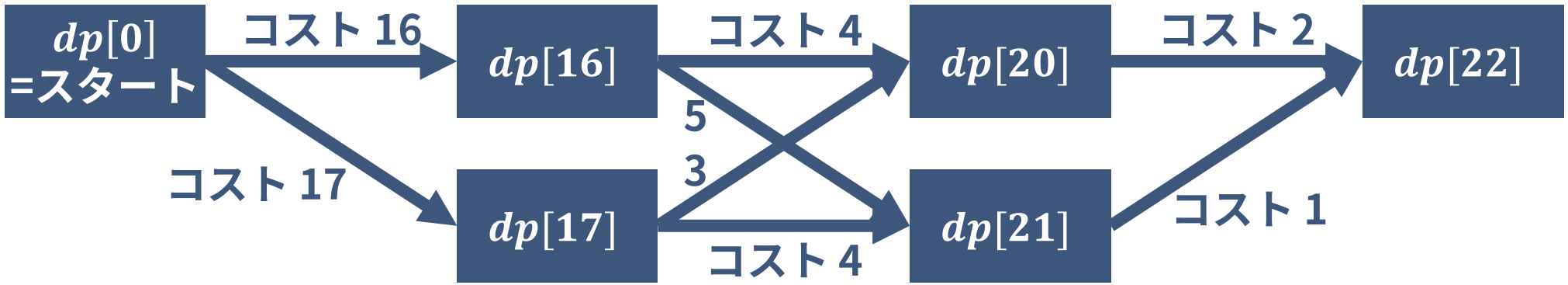
小課題 7: 具体例

- まずは例として、コスト 22 を計算したいとする (22=16+4+2)
- このとき、どのようなコスト 22 の経路も以下の形で表せる



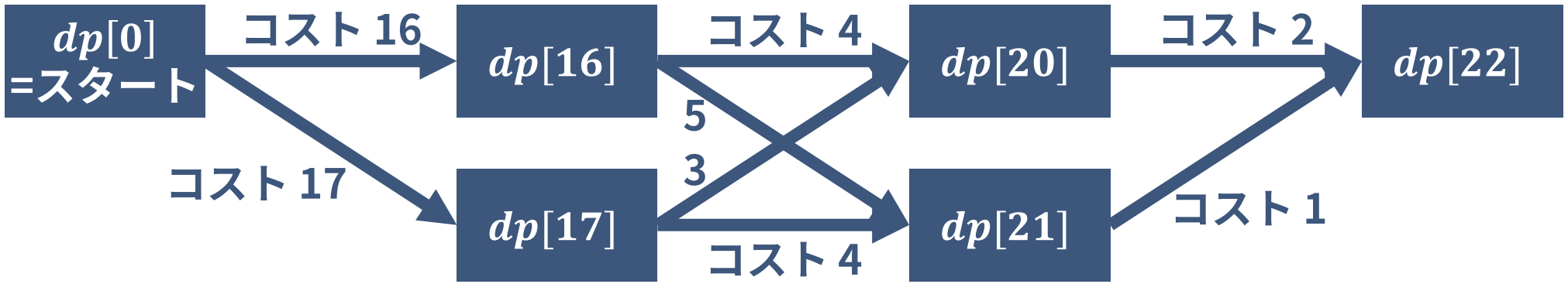
小課題 7: 具体例

- ここで、次のような DP を考える
  - $dp[i]$: コスト i で行ける位置の最大値 (最終的な答えは $dp[22]$)



小課題 7: 具体例

- ここで、次のような DP を考える
  - $dp[i]$: コスト i で行ける位置の最大値 (最終的な答えは $dp[22]$)
- コスト $2^k - 1, 2^k, 2^k + 1$ が前計算されているので、各 i について $dp[i]$ を $O(1)$ で計算できる！



小課題 7: 具体例

- ここで、次のような DP を考える
  - $dp[i]$: コスト i で行ける位置の最大値 (最終的な答えは $dp[22]$)
- コスト $2^k - 1, 2^k, 2^k + 1$ が前計算されているので、各 i について $dp[i]$ を $O(1)$ で計算できる！

これでコスト 22 の場合が
計算量 $O(\log 22)$ で解けた

小課題 7

一般の場合でも、以下のようにしてコスト x で行ける最大の位置がわかる

- コスト x で行ける最大の位置を計算したいとする
- $x = 2^{n_1} + 2^{n_2} + 2^{n_3} + \cdots + 2^{n_t}$ とする
- このとき、$i = 1, 2, \ldots, t$ の順に以下を計算する ($dp0[t]$ が答え)
  - $dp0[i]$: コスト $2^{n_1} + \cdots + 2^{n_i} + 0$ で行ける最大の位置
  - $dp1[i]$: コスト $2^{n_1} + \cdots + 2^{n_i} + 1$ で行ける最大の位置

小課題 7

- さて、小課題 7 では以下のような問題を解く必要があった
  - 地点 a から地点 b まで行くのに最小でコストいくつ必要か？

小課題 7

- さて、小課題 7 では以下のような問題を解く必要があった
  - 地点 a から地点 b まで行くのに最小でコストいくつ必要か？

- 前述の問題 (地点 a からコスト x で行ける最大の位置) とは少し違う
- しかし x を二分探索すると、$O(\log H)$ で解ける※

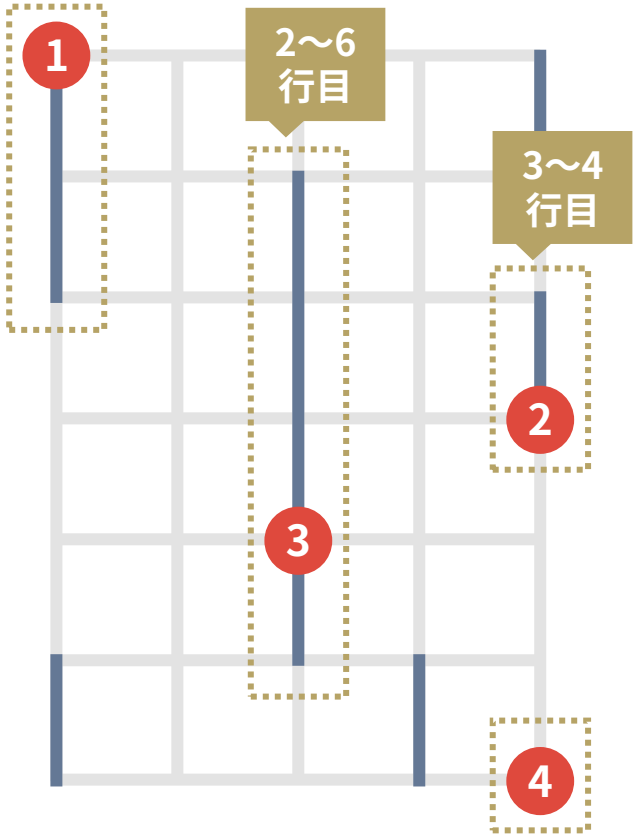
※単純に実装すると log が 2 つになるが、工夫すると log が 1 つになる

小課題 8

追加の制約はない

小課題 8

- クエリの連結成分の区間を $[l_1, r_1], \ldots, [l_T, r_T]$ 行目とする
- このとき、$[l_i, r_i]$ が $[l_j, r_j]$ に完全に含まれている場合、区間 $[l_j, r_j]$ は無視してよい
- たとえば下図の場合、地点 3 は無視して、地点 1, 2, 4 のみを連結にする問題だと考えて良い



小課題 8

したがって、クエリの区間を $[l_i, r_i]$ とするとき

$l_1 < \cdots < l_T, r_1 < \cdots < r_T$ を仮定できる

小課題 8

120

- そこで次のような動的計画法を考える※
  - $dp1[i]$: 地点 1 から地点 i までを連結にするのにかかる最小コスト
  - $dp2[i]$: 地点 1 から地点 i までを連結にする場合、コスト $dp1[i]+1$ をかければ何行目まで行けるか

※ $l_1 < l_2 < \cdots < l_T$ および $r_1 < r_2 < \cdots < r_T$ が成り立つとする。つまり地点 1 は一番上

小課題 8

- そこで次のような動的計画法を考える※
  - $dp1[i]$: 地点 1 から地点 i までを連結にするのにかかる最小コスト
  - $dp2[i]$: 地点 1 から地点 i までを連結にする場合、コスト $dp1[i]+1$ をかければ何行目まで行けるか

小課題 7 で解いた問題を使えば、$dp1[i-1], dp2[i-1]$ から $dp1[i], dp2[i]$ を計算することができる

小課題7の問題

- 地点 a からコスト x でどこまで行けるか？
- 地点 a から地点 b まで最小コストいくつで行けるか？

※ $l_1 < l_2 < \cdots < l_T$ および $r_1 < r_2 < \cdots < r_T$ が成り立つとする。つまり地点 1 は一番上

小課題 8

これで計算量 $O(T \log H)$ で
各クエリを解くことができた！

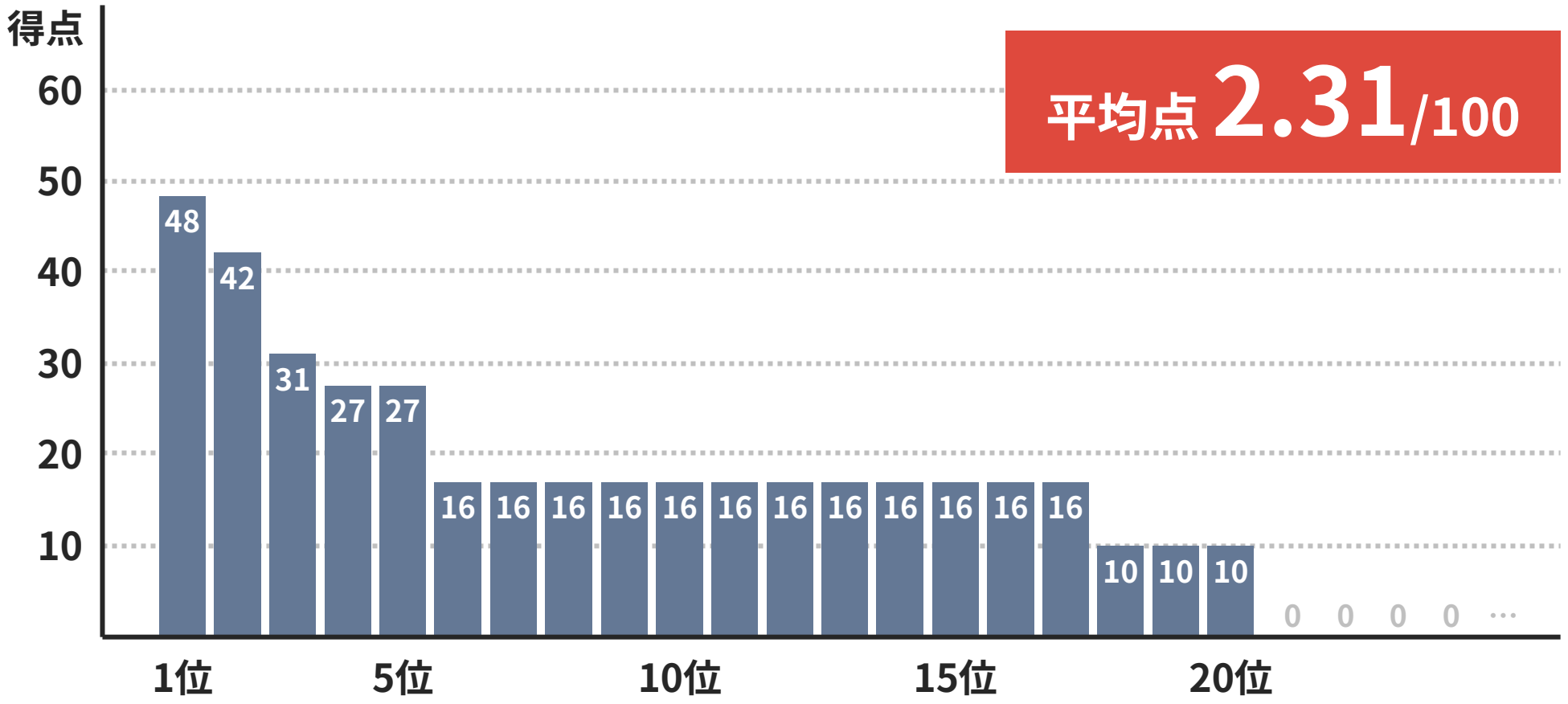
小課題 8 の実装

- この問題は実装と場合分けが非常に大変
- しかし、以下のようなテクニックを使うとバグりにくくなったり、バグ取りが速くなったりする
  - コードに適切なコメントを書く
  - 実装する前に紙に実装方針を書く
  - 自分なりのコーディングスタイルを身につける　etc.

100点

得点分布

得点分布


得点
60
50
40
30
20
10
平均点 2.31/100
48
42
31
27
27
16
16
16
16
16
16
16
16
16
16
16
16
10
10
10
0
0
0
0
…
1位
5位
10位
15位
20位