JOI 2024/25 本選 4

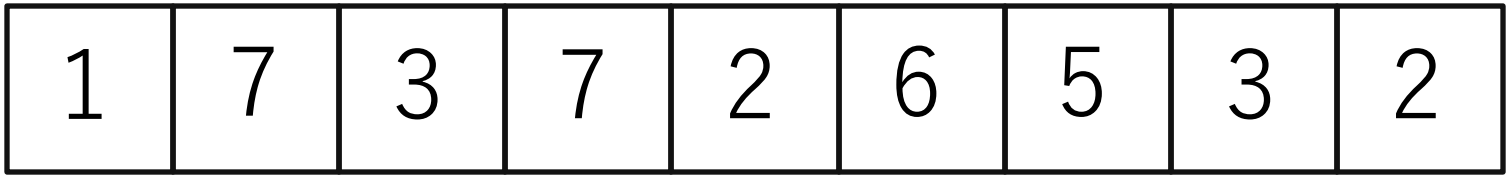
長いだけのネクタイ 2
(Just Long Neckties 2)

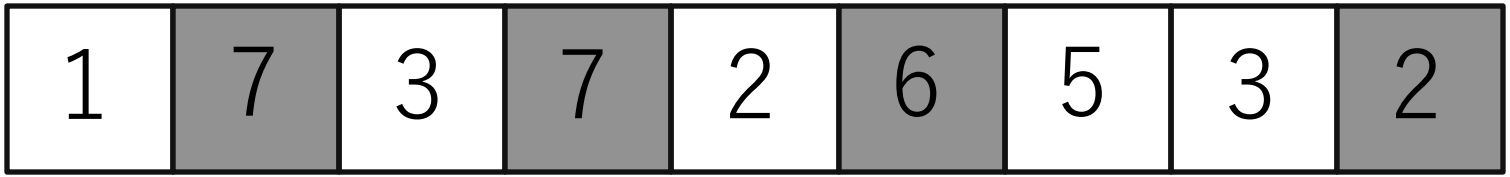
解説：渡邉雄斗 (yuto1115)

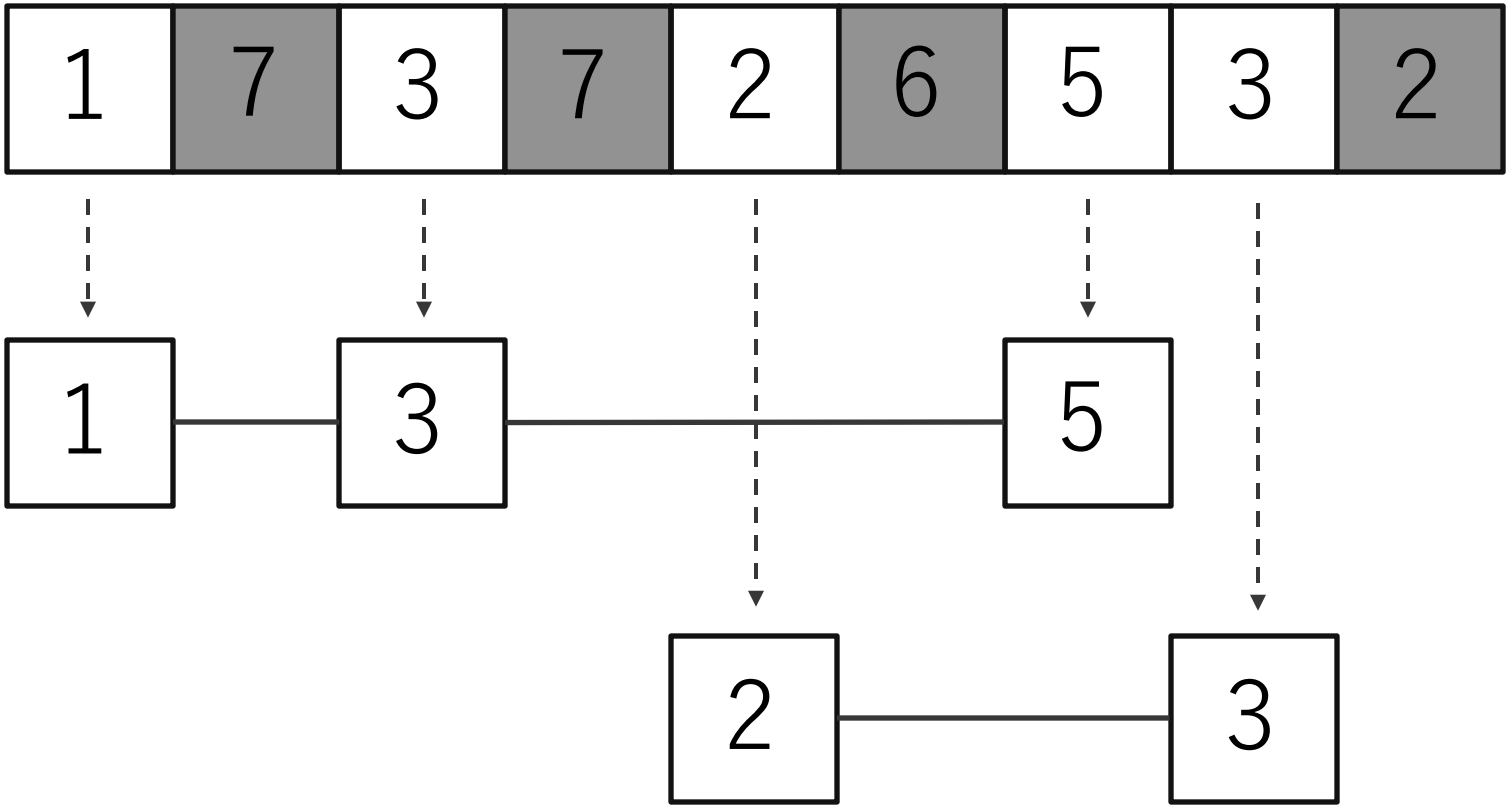
問題概要

問題概要

- 長さ N の数列 $A = (A_1, A_2, \ldots, A_N)$ が与えられる
- まず、数列のいくつかの要素を選んで消す
    - ただし、連続する要素をどちらも消してはいけない
- 残る要素をいくつかの広義単調増加部分列に分割する
- 最小でいくつの部分列に分割できるか？


1
7
3
7
2
6
5
3
2


1
7
3
7
2
6
5
3
2


1
7
3
7
2
6
5
3
2
1
3
5
2
3

考察 1: 初歩的な DP

小課題 2 $N \leq 500, A_i \leq 2$ (6 pts.)

小課題 3 $N \leq 500, A_i \leq 5$ (12 pts.)

考察 1: 初歩的な DP

- まずは答えの範囲を考えたい
  - 最小でも 1 は必要 $(N \geq 2)$
  - $L = max\ A_i$ として、L あれば必ず可能

    （値が i である要素をまとめて i 番目の列とする）

考察 1: 初歩的な DP

小課題 2 $N \leq 500, A_i \leq 2$

- 答えは 1 or 2 なので、1 が可能かどうか調べたい
  - 残る要素が広義単調増加となるように消す要素を選べるか？
- $dp_i =$「i 番目まで決めた (i 番目は残した) とき、

  残した要素が単調増加」が可能かどうか (true / false)

考察 1: 初歩的な DP

小課題 2 $N \leq 500, A_i \leq 2$

- $dp_i =$「i 番目まで決めた (i 番目は残した) とき、

  残した要素が単調増加」が可能かどうか (true / false)
- 初期条件：$dp_0 =$ true
- 遷移：$dp_i =$ true かつ $A_i \leq A_{i+1}$ なら $dp_{i+1} =$ true

  $dp_i =$ true かつ $A_i \leq A_{i+2}$ なら $dp_{i+2} =$ true

考察 1: 初歩的な DP

小課題 2 $N \leq 500, A_i \leq 2$

- $O(N)$ の DP で解くことができる
- (別解)

  答えが 2

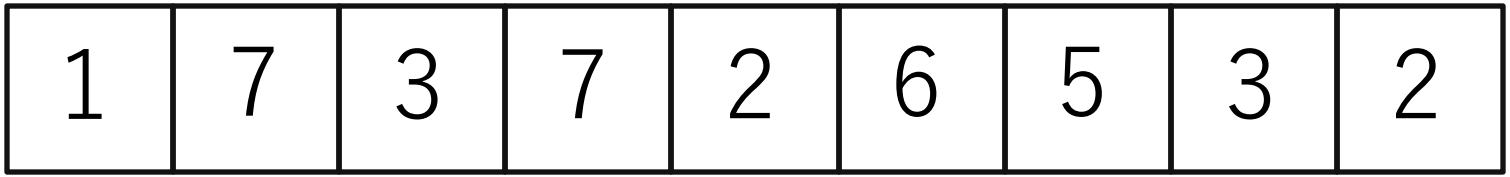
  $\Leftrightarrow A_i = A_{i+1} = 2, A_j = A_{j+1} = 1$ なる $i < j$ が存在

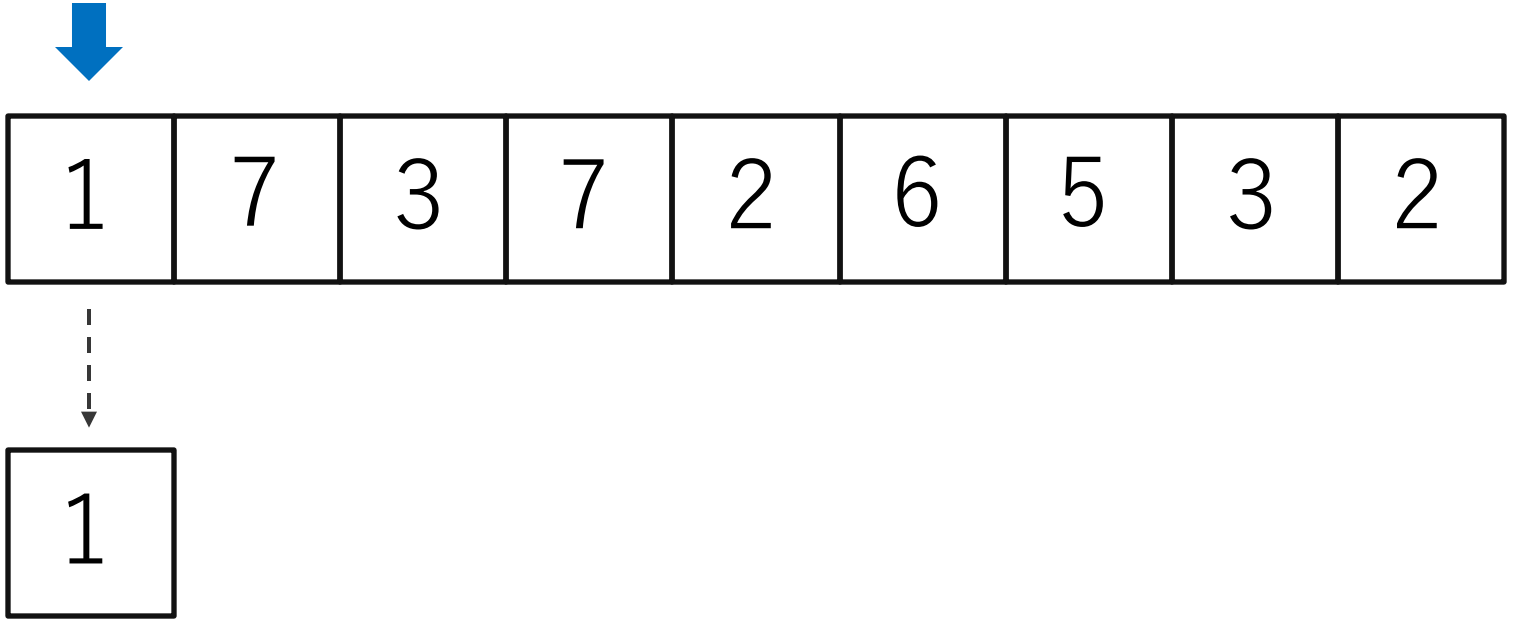
考察 1: 初歩的な DP

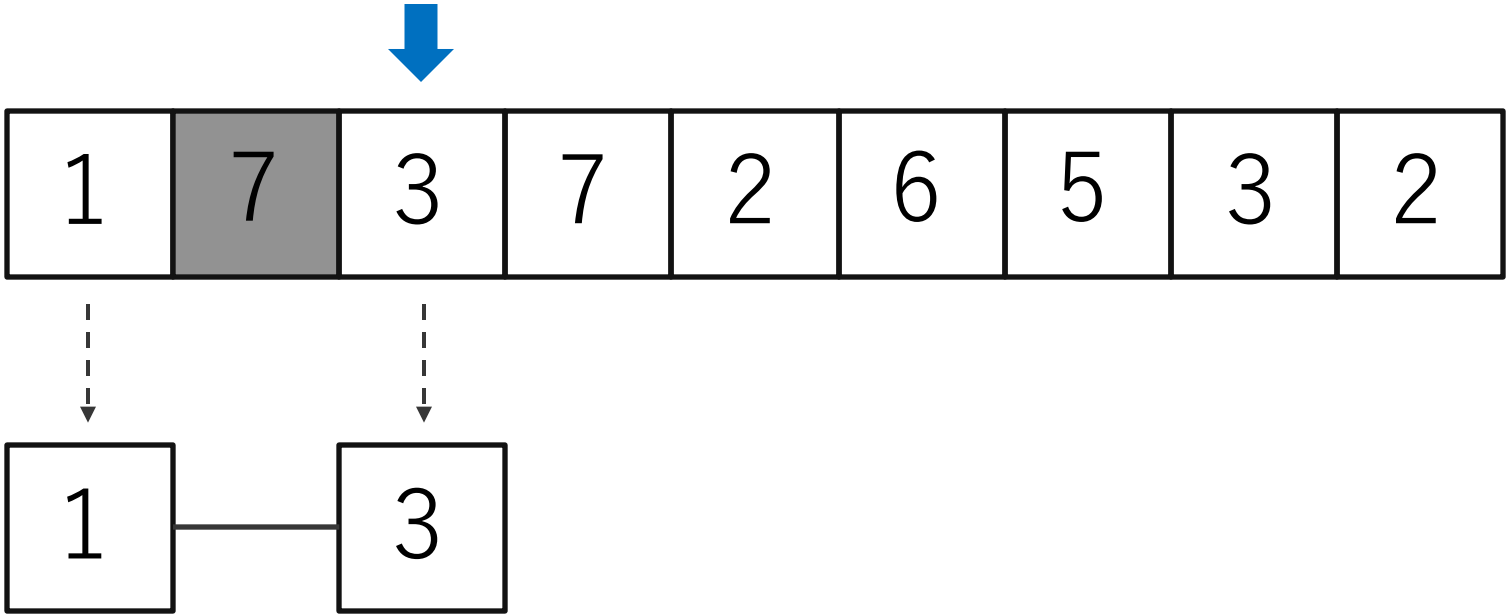
小課題 3 $N \leq 500, A_i \leq 5$

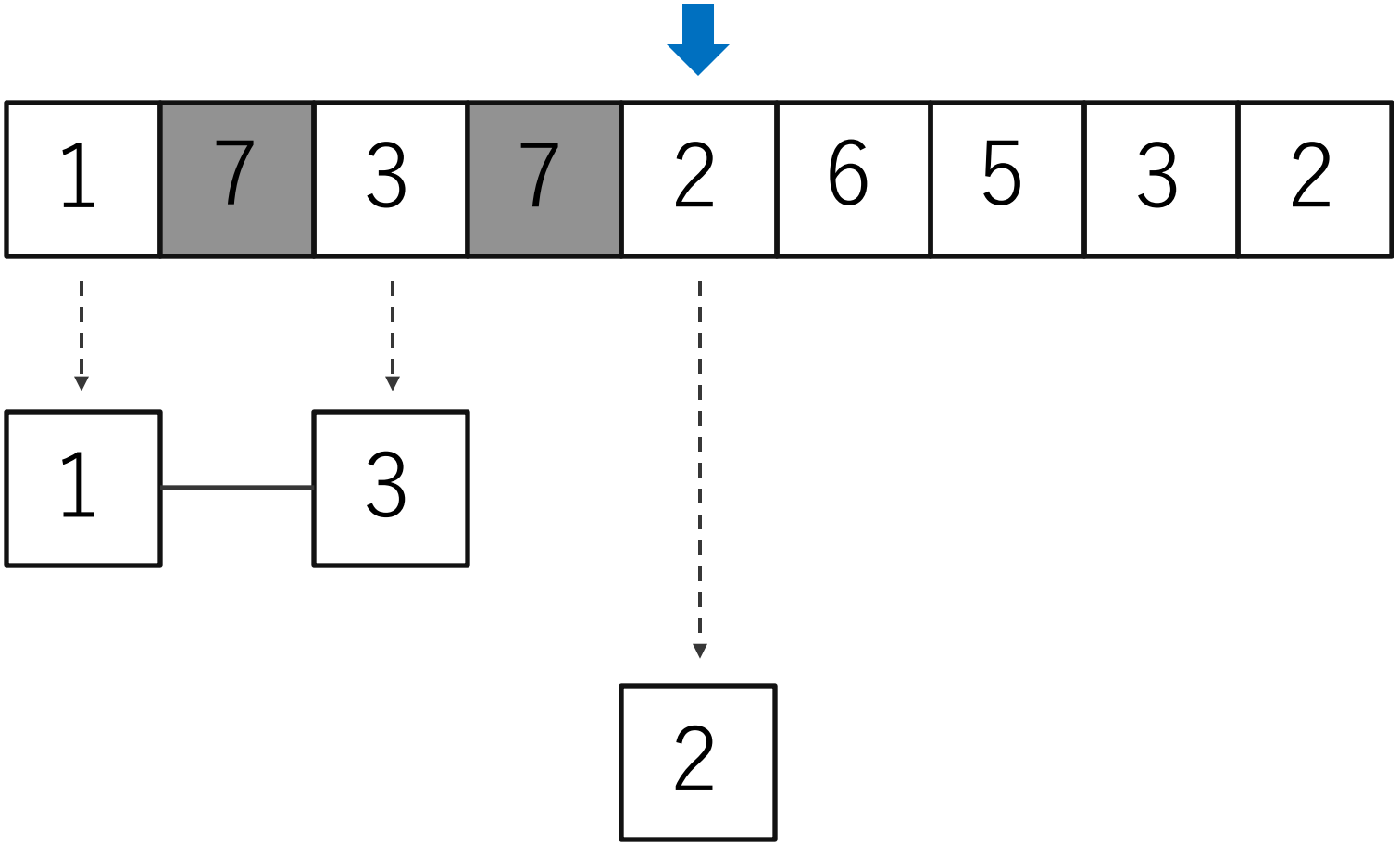
- 答えは 1 以上 5 以下
- 先頭要素から見ていって、残すかどうかを決めつつ、

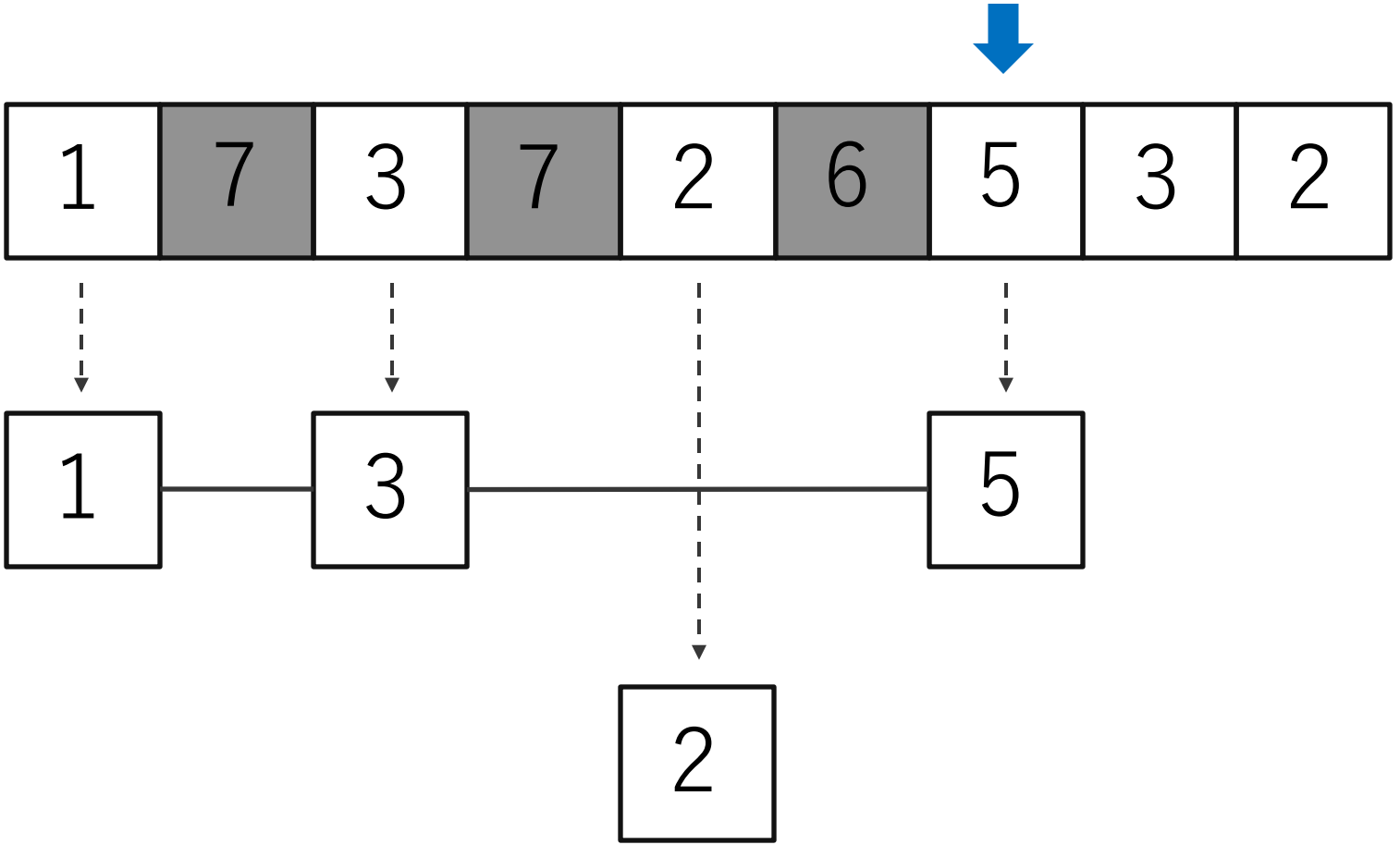
  同時に部分列を構築していく

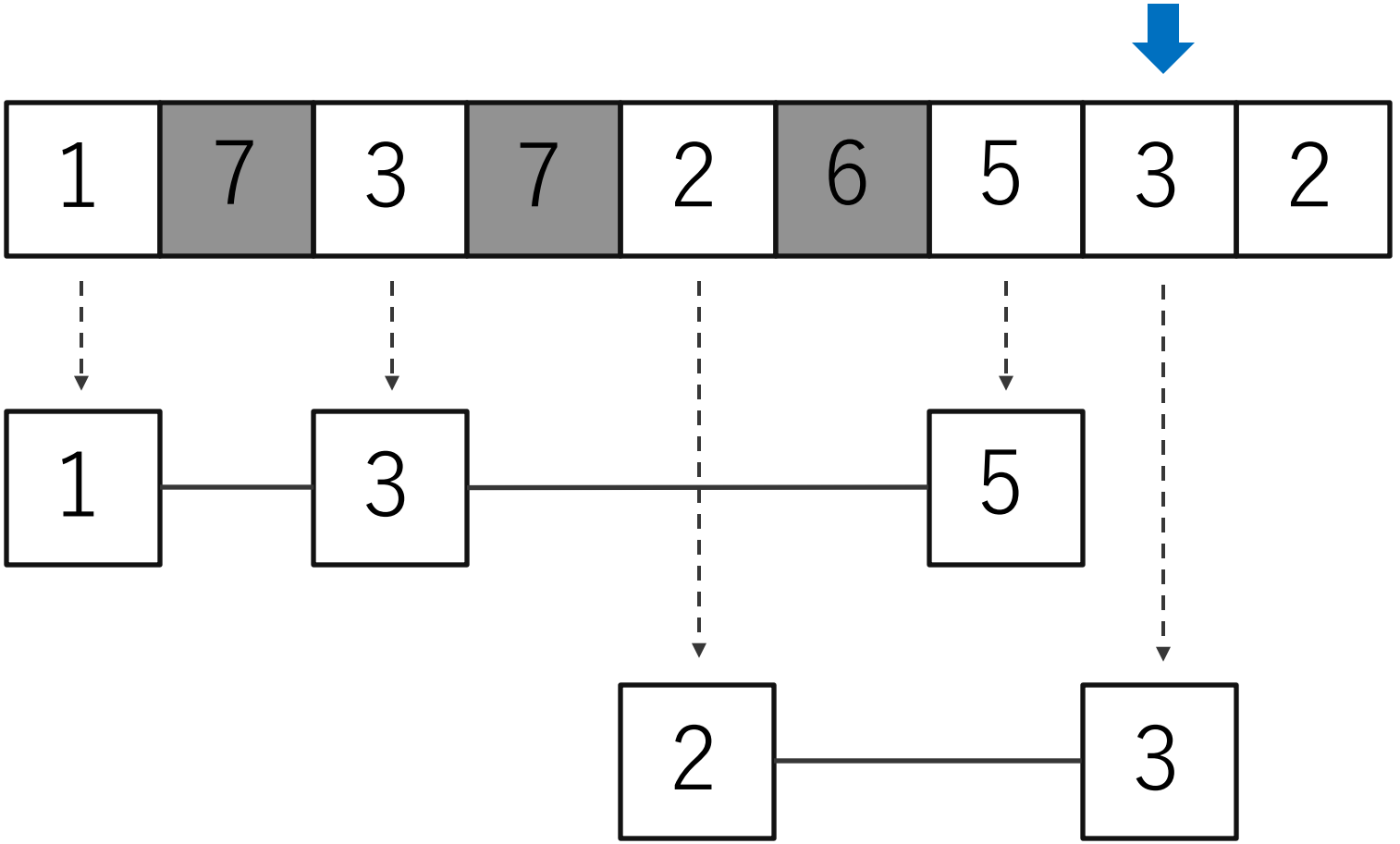

1
7
3
7
2
6
5
3
2

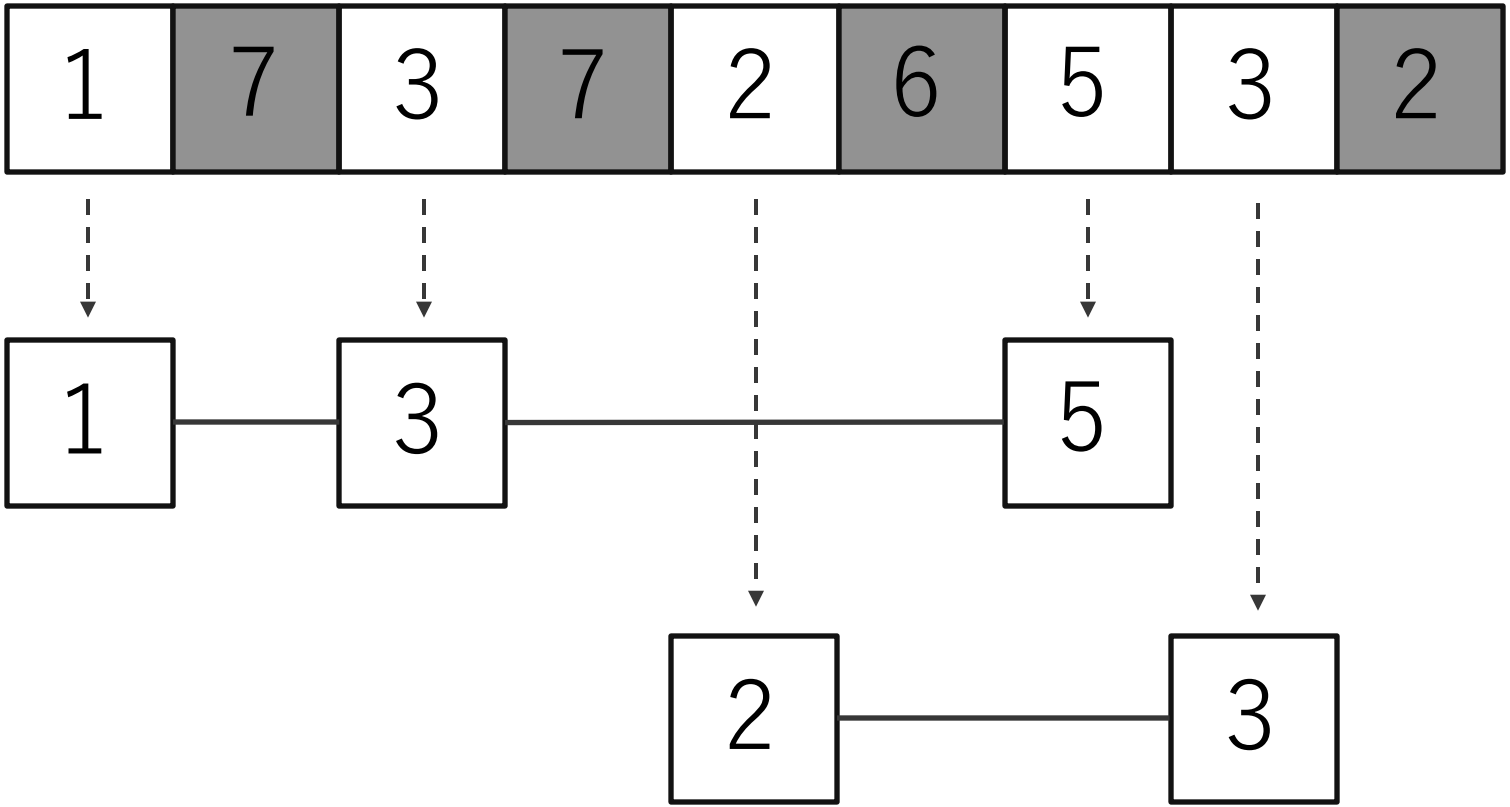

1
7
3
7
2
6
5
3
2
1

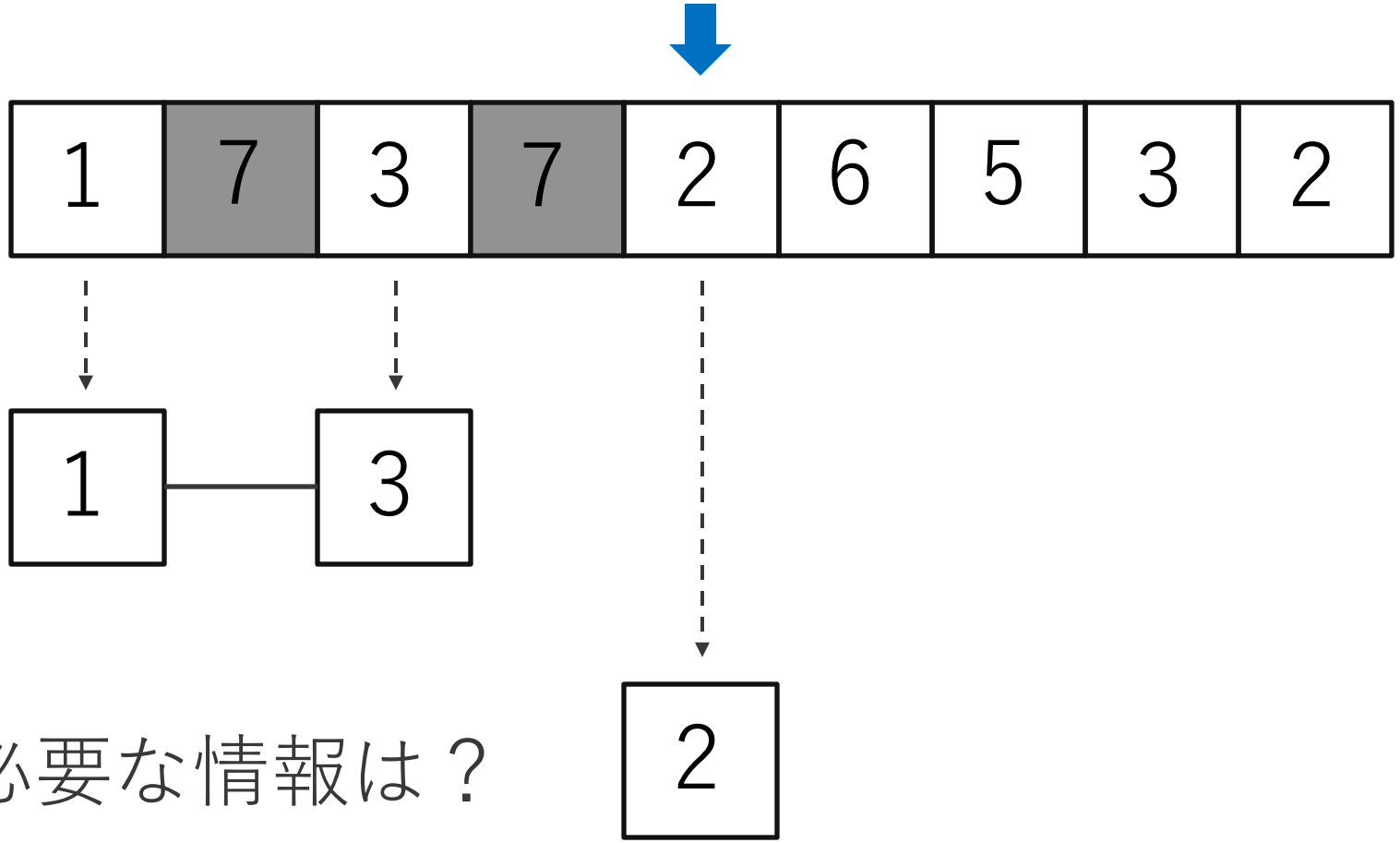

1
7
3
7
2
6
5
3
2
1
3


1
7
3
7
2
6
5
3
2
1
3
2

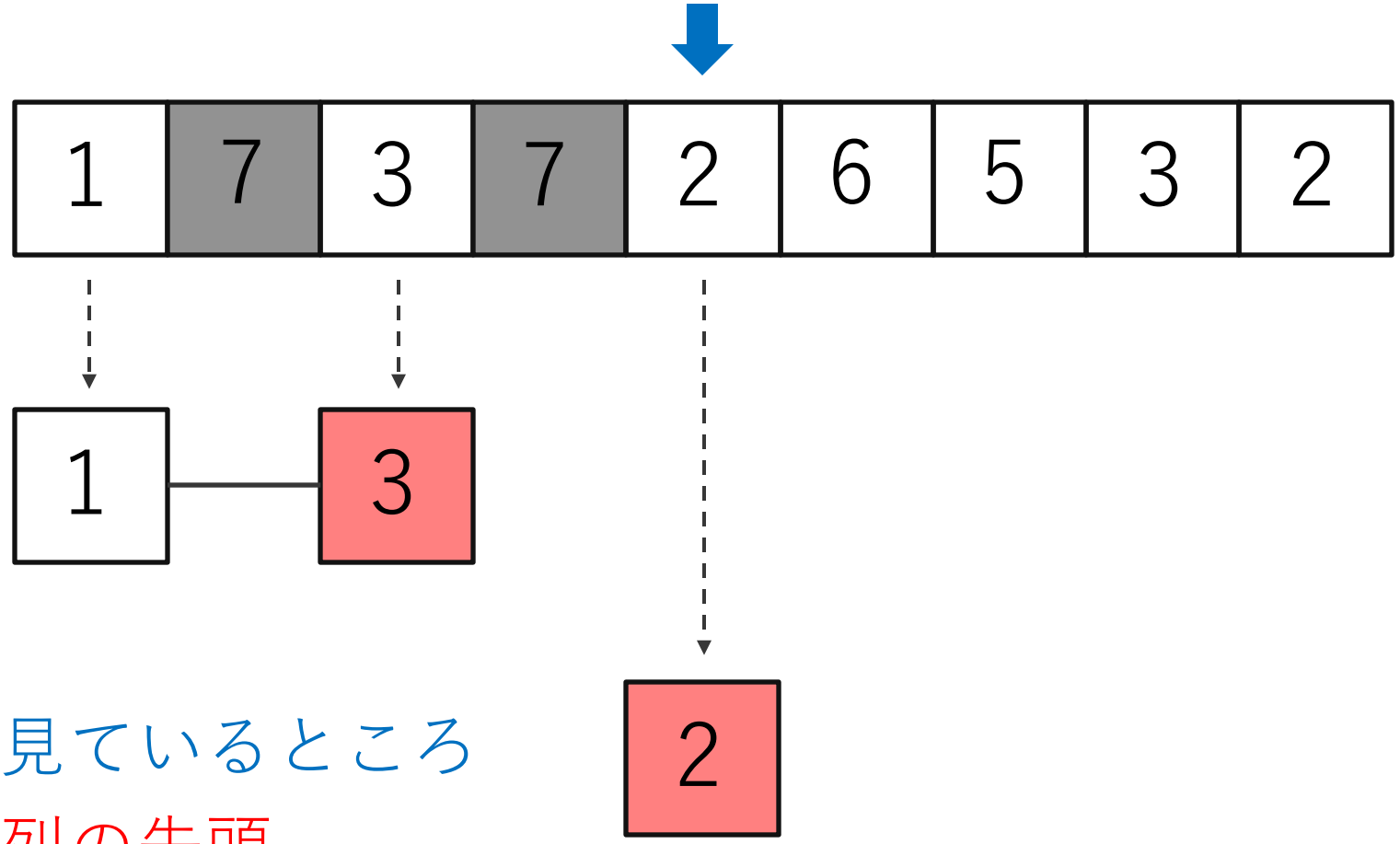

1
7
3
7
2
6
5
3
2
1
3
5
2


1
7
3
7
2
6
5
3
2
1
3
5
2
3


1
7
3
7
2
6
5
3
2
1
3
5
2
3



必要な情報は？


1
7
3
7
2
6
5
3
2
1
3
2
今見ているところ
各列の先頭

考察 1: 初歩的な DP

小課題 3 $N \leq 500, A_i \leq 5$

- 列は最大 5 個あれば十分
- $dp_{i,p,q,r,s,t}$ =「i 番目まで決めた (i 番目は残した) とき、列 1,2,3,4,5 の先頭がそれぞれ p, q, r, s, t」は可能かどうか (true / false)
- 空の列は先頭が 0 ということにしておいて、

  最終的に先頭が 0 の列の個数が最大のものが最適

考察 1: 初歩的な DP

小課題 3 $N \leq 500, A_i \leq 5$

- $dp_{i,p,q,r,s,t}$ =「i 番目まで決めた (i 番目は残した) とき、列 1,2,3,4,5 の先頭がそれぞれ p, q, r, s, t」は可能かどうか (true / false)
- 遷移：次に残す要素 ($i+1$ or $i+2$) とその要素を足す列を全探索
- $L = max\ A_i$ として、 状態 $O(NL^L)$、遷移 $O(L)$

考察 2: 貪欲法

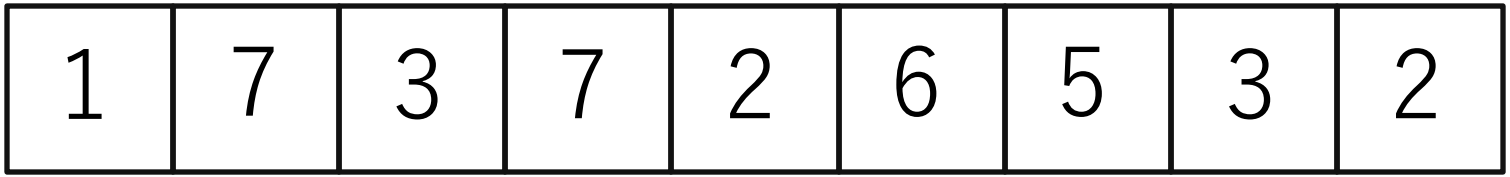
小課題 1 $N \leq 15, A_i \leq 21$ (10 pts.)

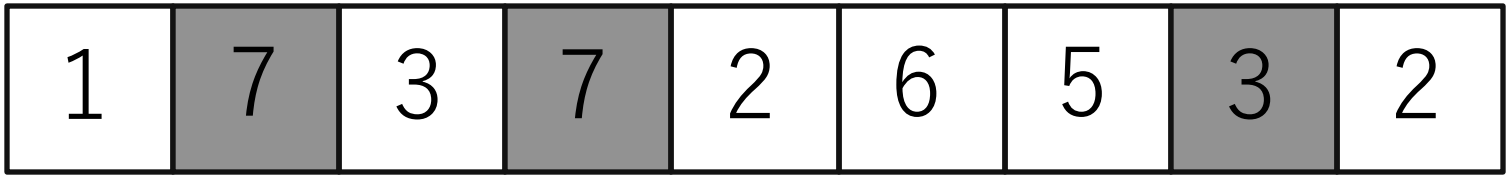
考察 2: 貪欲法

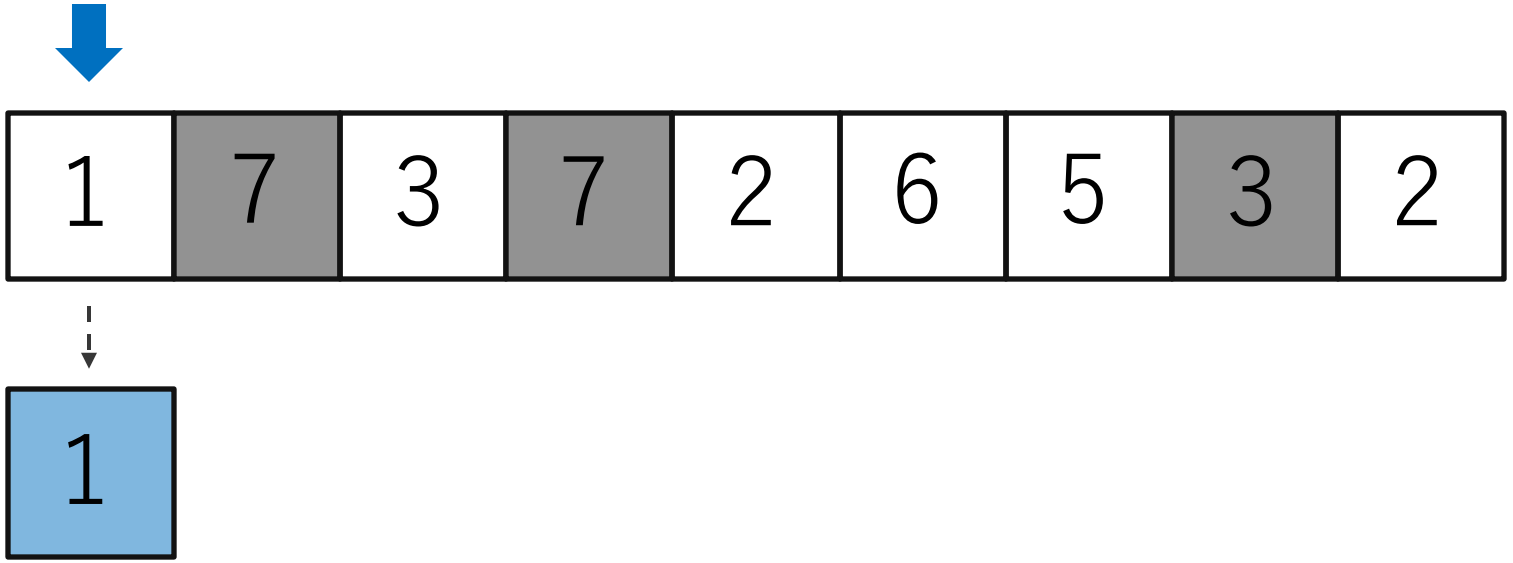
小課題 1 $N \leq 15, A_i \leq 21$

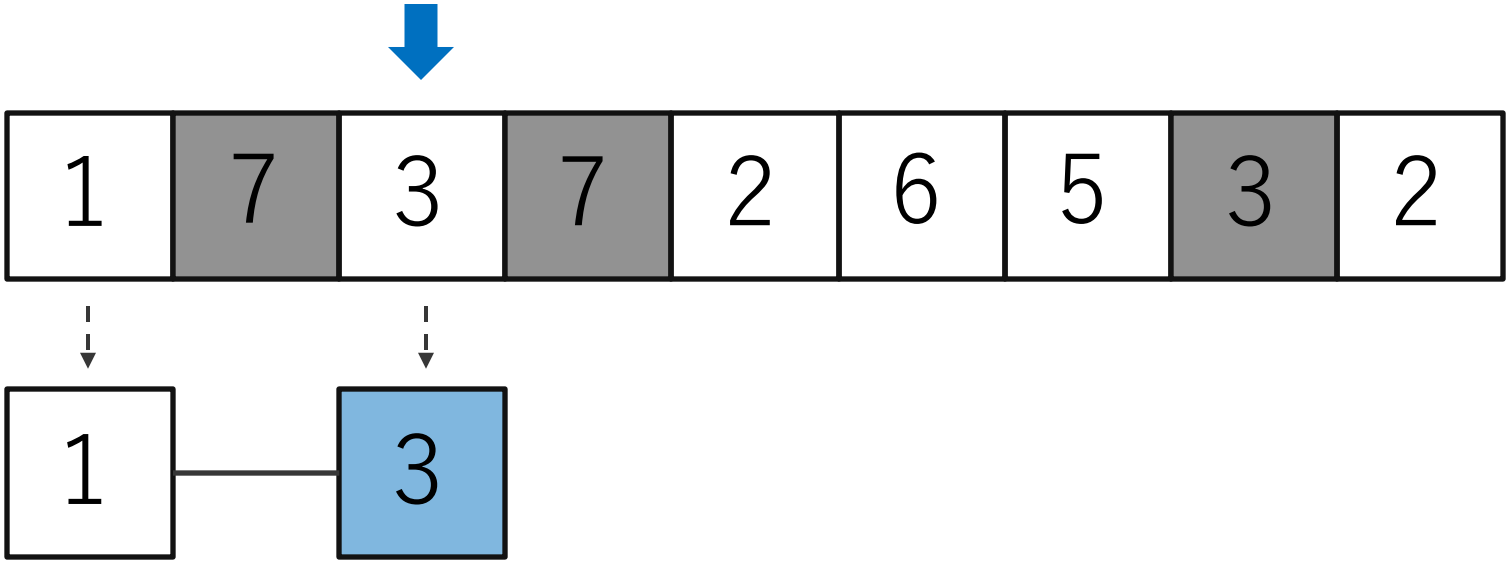
- N が小さいので、どの要素を残すかを $O(2^N)$ 通り全探索できる
- 残す要素が決まったとき、最小で何個の部分列に分割可能か求めたい
  - $L = max\ A_i$ の値が大きいので DP はできない → 貪欲法
  - 追加できる列があるなら、そのうち先頭が最大のものに追加

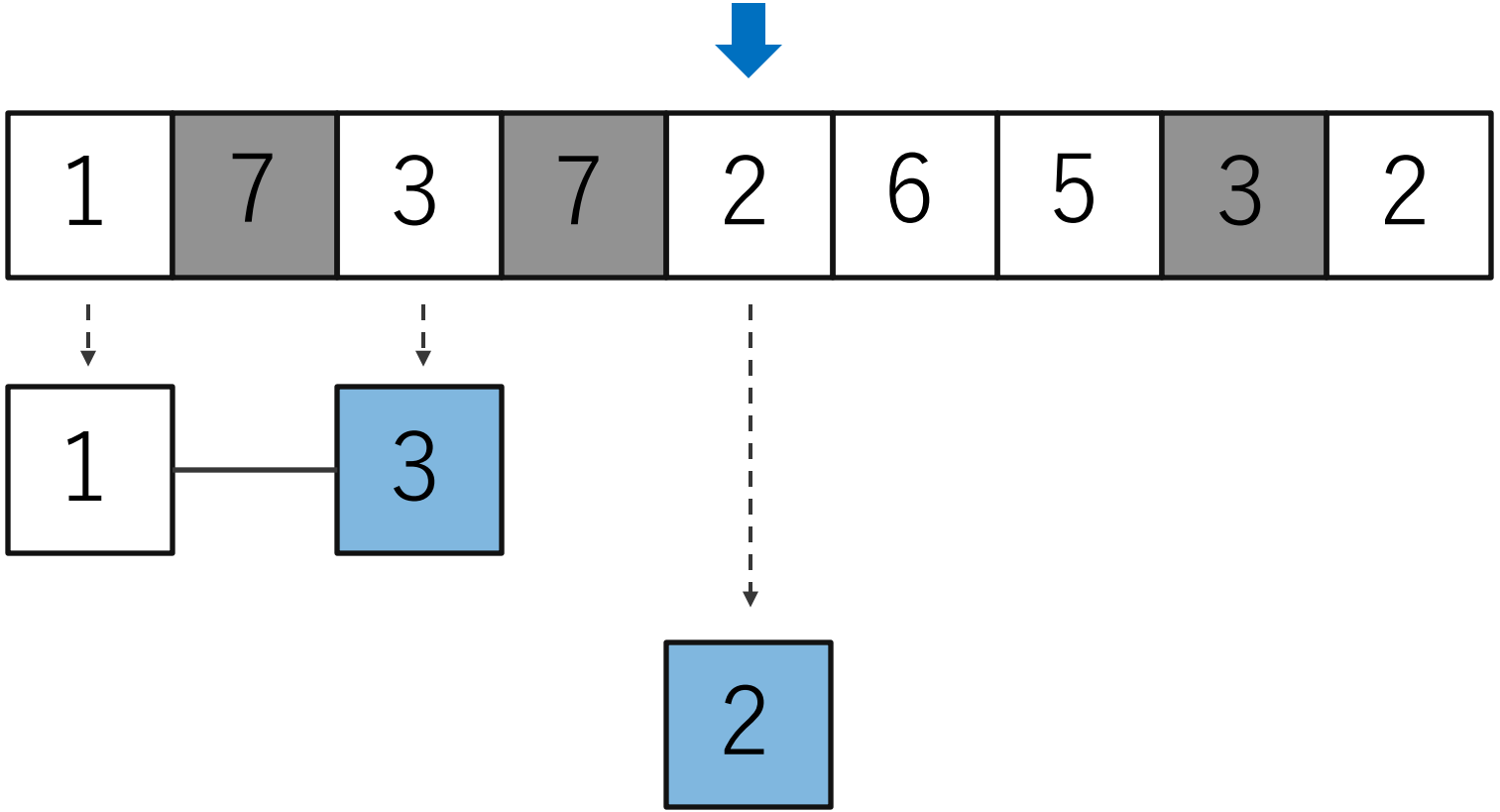
    追加できる列がないなら、新しい列を作る

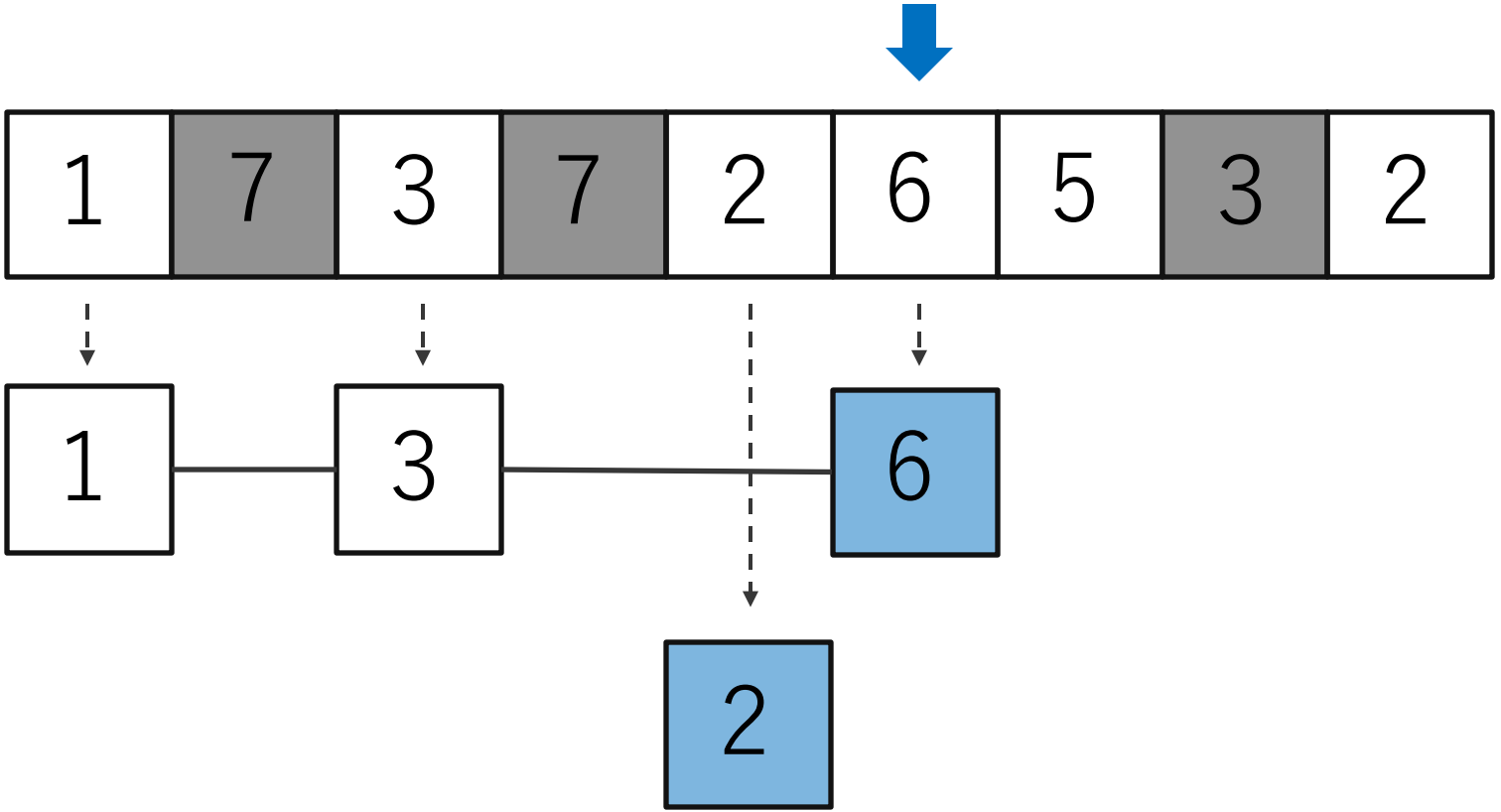

1
7
3
7
2
6
5
3
2

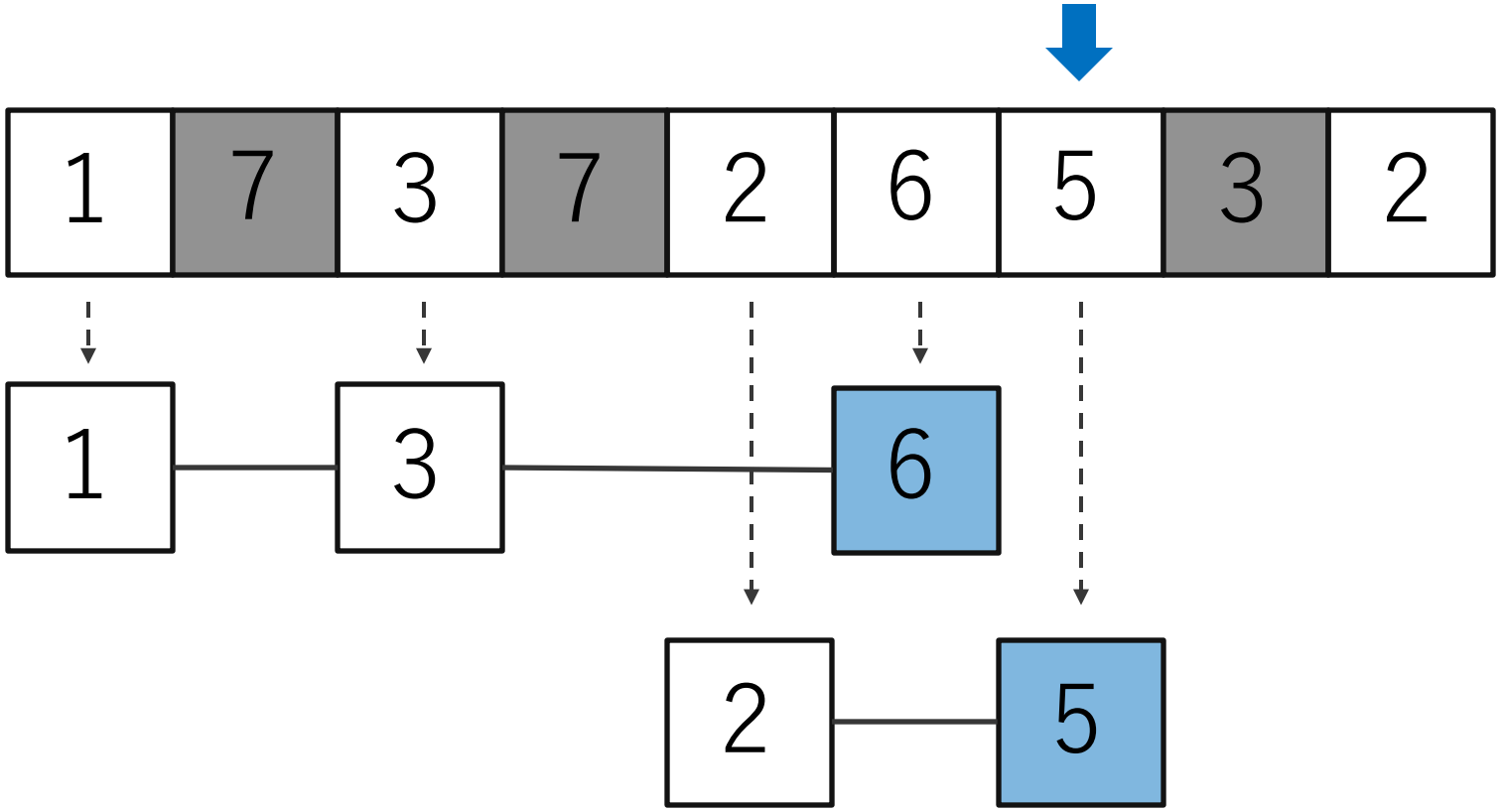

1
7
3
7
2
6
5
3
2

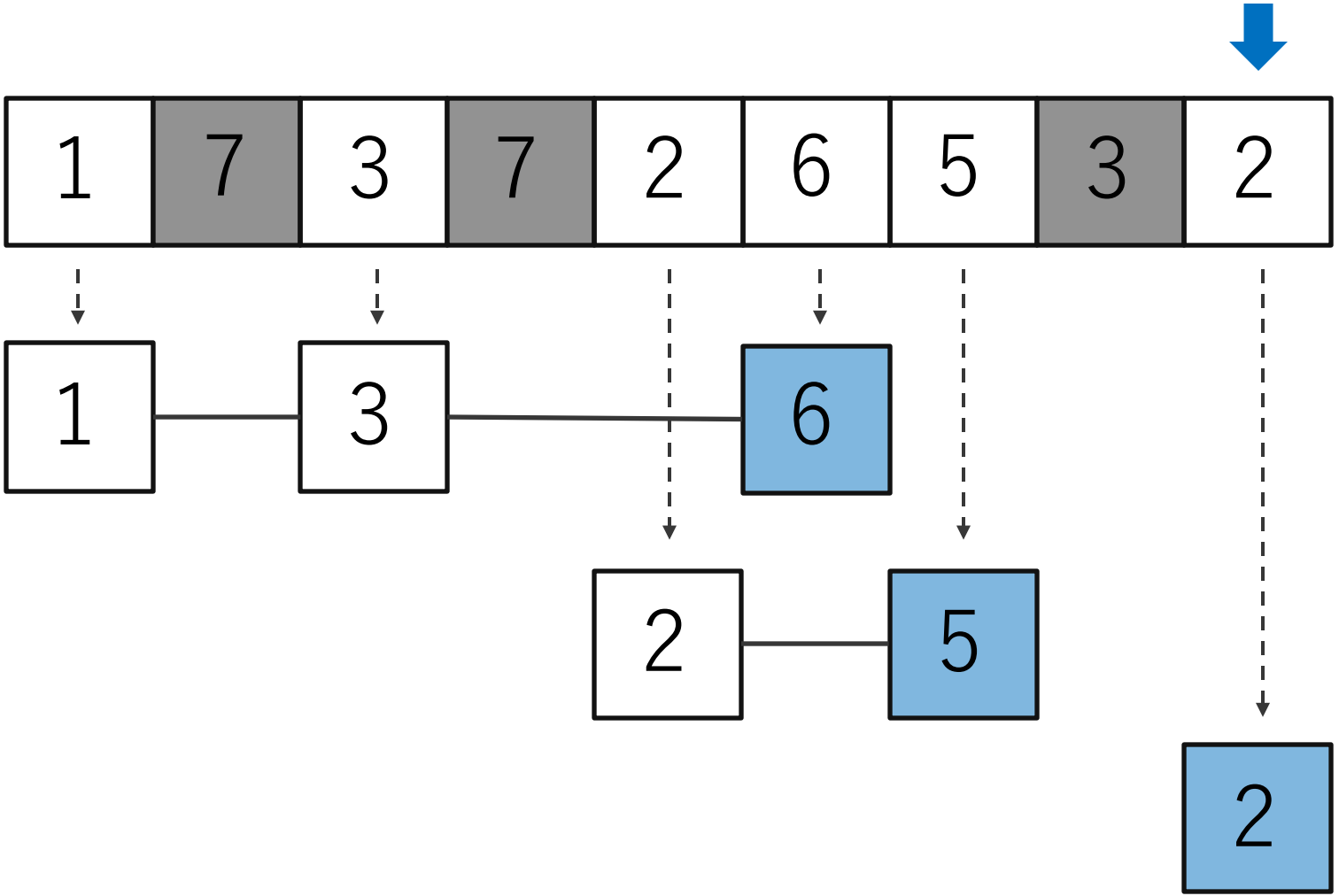

1
7
3
7
2
6
5
3
2
1


1
7
3
7
2
6
5
3
2
1
3


1
7
3
7
2
6
5
3
2
1
3
2


1
7
3
7
2
6
5
3
2
1
3
6
2


1
7
3
7
2
6
5
3
2
1
3
6
2
5


1
7
3
7
2
6
5
3
2
1
3
6
2
5
2

考察 2: 貪欲法

小課題 1 $N \leq 15, A_i \leq 21$

- $O(2^N)$ 通り全探索 → 貪欲法で各 $O(N^2)$ など

考察 3: bit DP

小課題 4 $N \leq 500, A_i \leq 15$ (18 pts.)

考察 3: bit DP

小課題 4 $N \leq 500, A_i \leq 15$

- $O(NL\text{^}L)$ 状態の DP に、先ほどの貪欲法を統合する
    - 元々の遷移： 次に残す要素を決めた上で、追加する列を全探索

      改善後：次に残す要素を決めたとき、追加する列を**貪欲法で決定**
- 状態数は結局変わらない…?

考察 3: bit DP

小課題 4 $N \leq 500, A_i \leq 15$

- $O(NL\text{^}L)$ 状態の DP に、先ほどの貪欲法を統合する
- 観察：貪欲法を用いた場合、どの途中状態においても、

  各列の先頭要素はすべて相異なる
- 列の先頭要素は $\{1,2,\ldots,L\}$ の部分集合とみなせ、2^L 通り

考察 3: bit DP

小課題 4 $N \leq 500, A_i \leq 15$

- $dp_{i,S}$ =「i 番目まで決めた (i 番目は残した) とき、各列の先頭要素として現れる値の集合が S」は可能かどうか (true / false)
- 状態 $O(N2\text{^}L)$、遷移 $O(L)$ の bit DP
- 自然な実装をすれば全体でも計算量 $O(N2\text{^}L)$

考察 4: DP 高速化

小課題 5 $N \leq 5 \times 10^5, A_i \leq 15$ (26 pts.)

小課題 6 $N \leq 5 \times 10^5, A_i \leq 21$ (10 pts.)

小課題 7 $N \leq 5 \times 10^6, A_i \leq 21$ (18 pts.)

(満点)

考察 4: DP 高速化

- 小課題 4 で用いた bit DP の復習

$A =$

2	1	1	2	2	1	2

S \ i	0	1	2	3	4	5	6	7
{}								
{1}								
{2}								
{1, 2}								

$A =$

2	1	1	2	2	1	2

S \ i	0	1	2	3	4	5	6	7
{}	○							
{1}								
{2}								
{1, 2}								

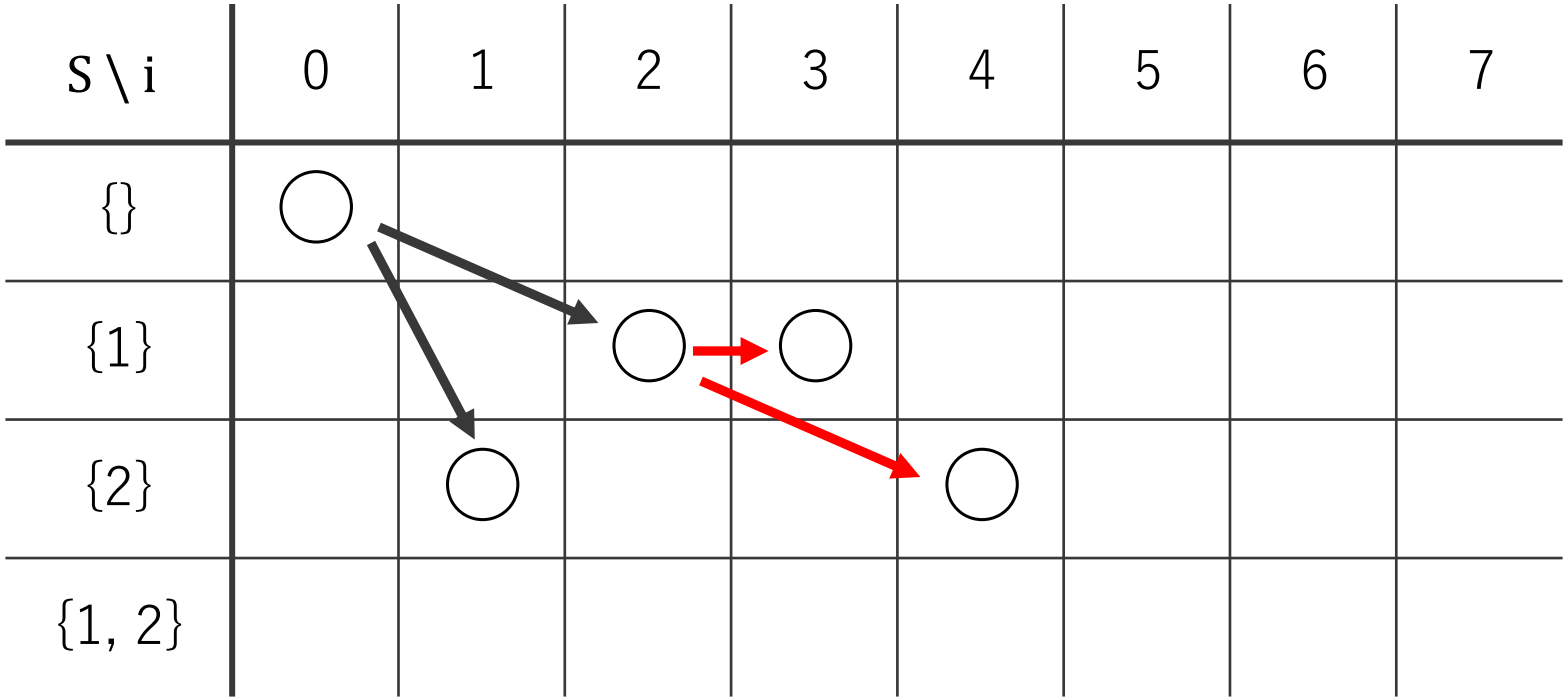
$A =$

2	1	1	2	2	1	2

S \ i	0	1	2	3	4	5	6	7
{}	○							
{1}			○					
{2}		○						
{1, 2}								

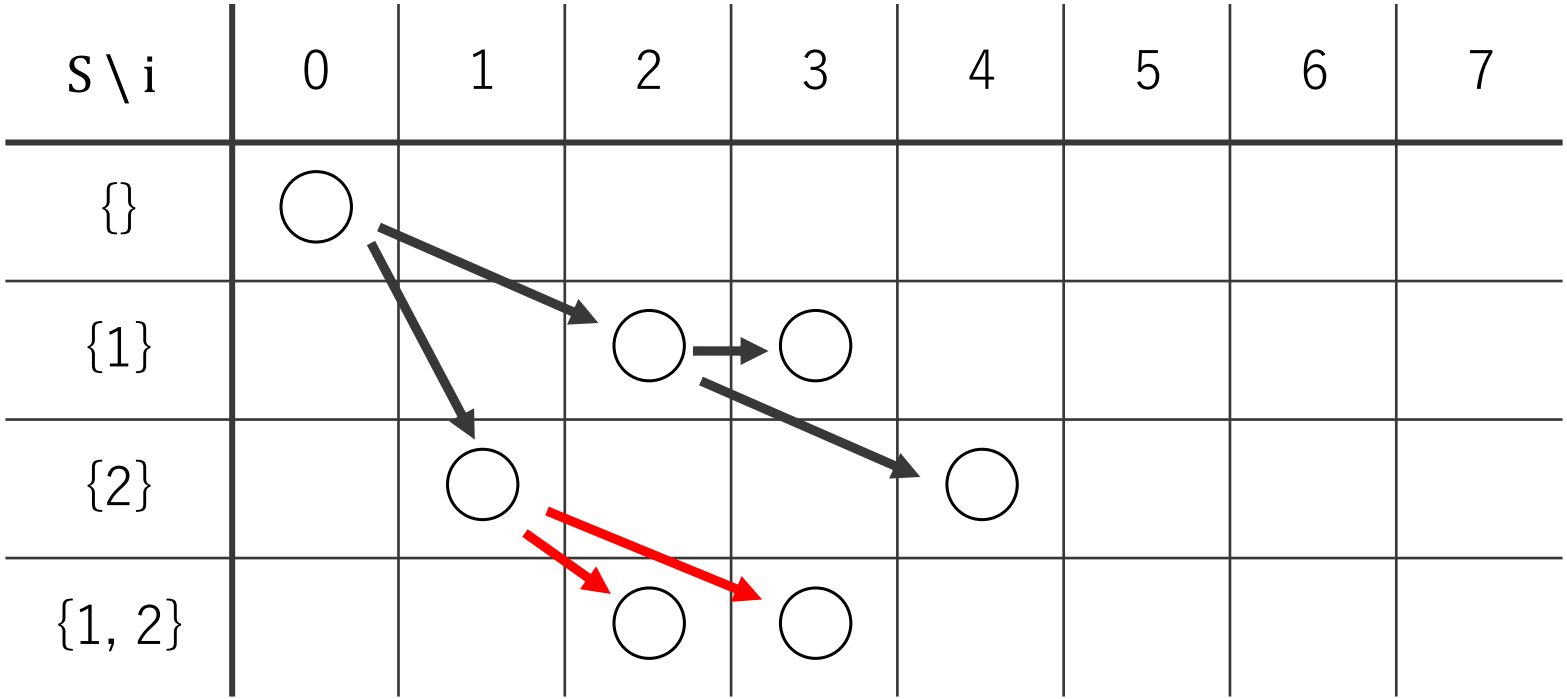
$A =$

2	1	1	2	2	1	2



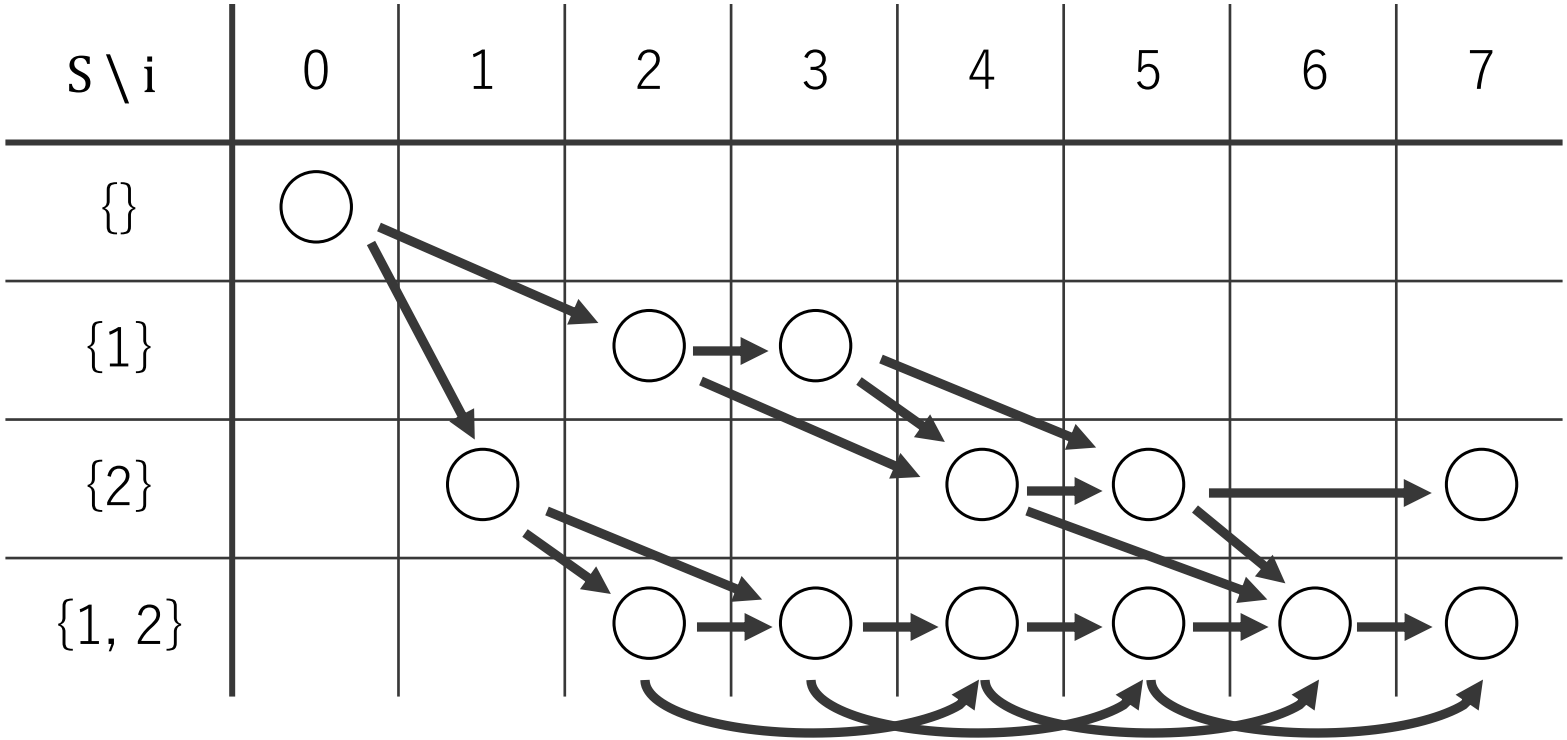
$A =$

2	1	1	2	2	1	2



$A =$

2	1	1	2	2	1	2

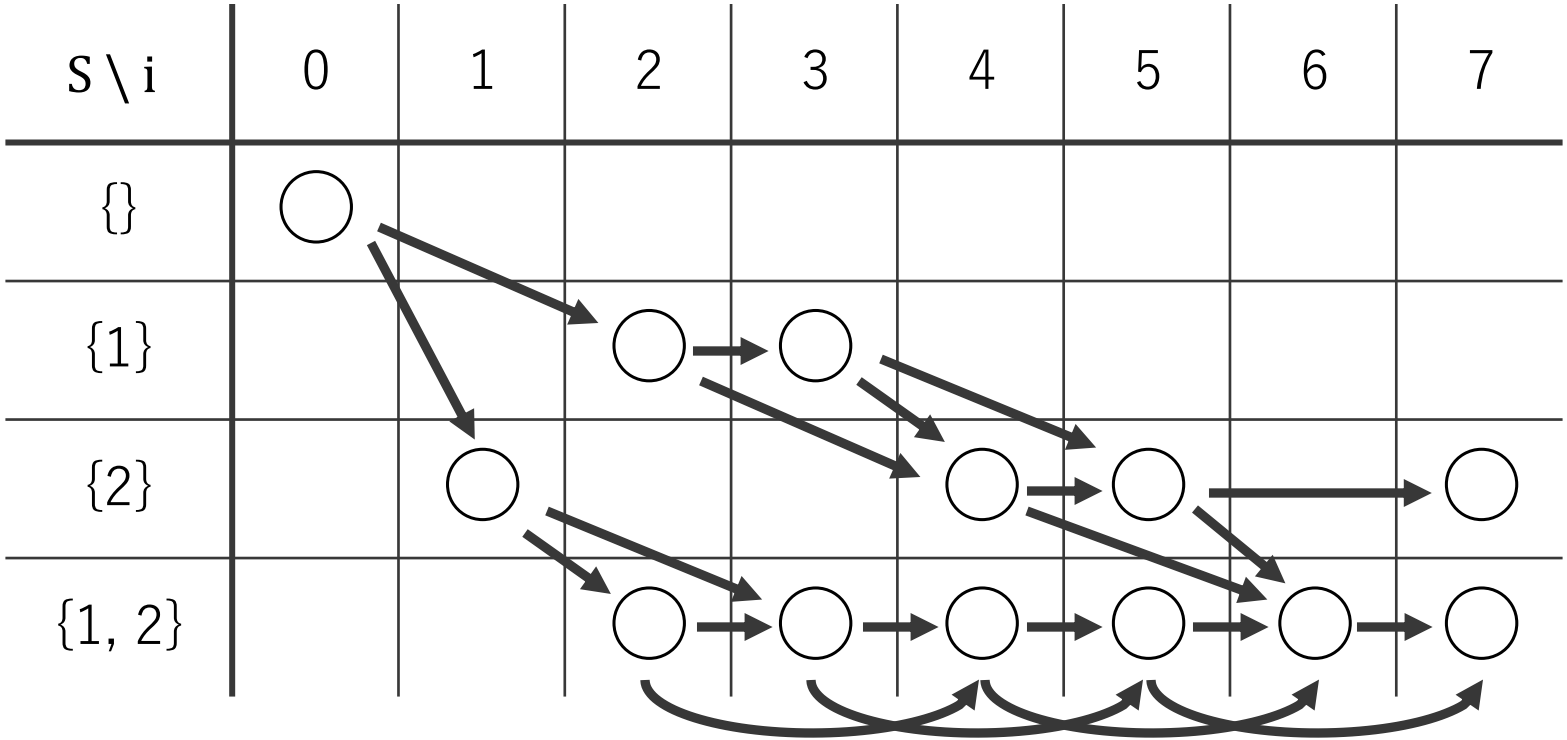


考察 4: DP 高速化

- 小課題 4 で用いた bit DP の復習
  - $N2^L$ 個の bool 値を保持している
  - 保持する情報の量を減らしたい
  - 不要な情報 はないだろうか？

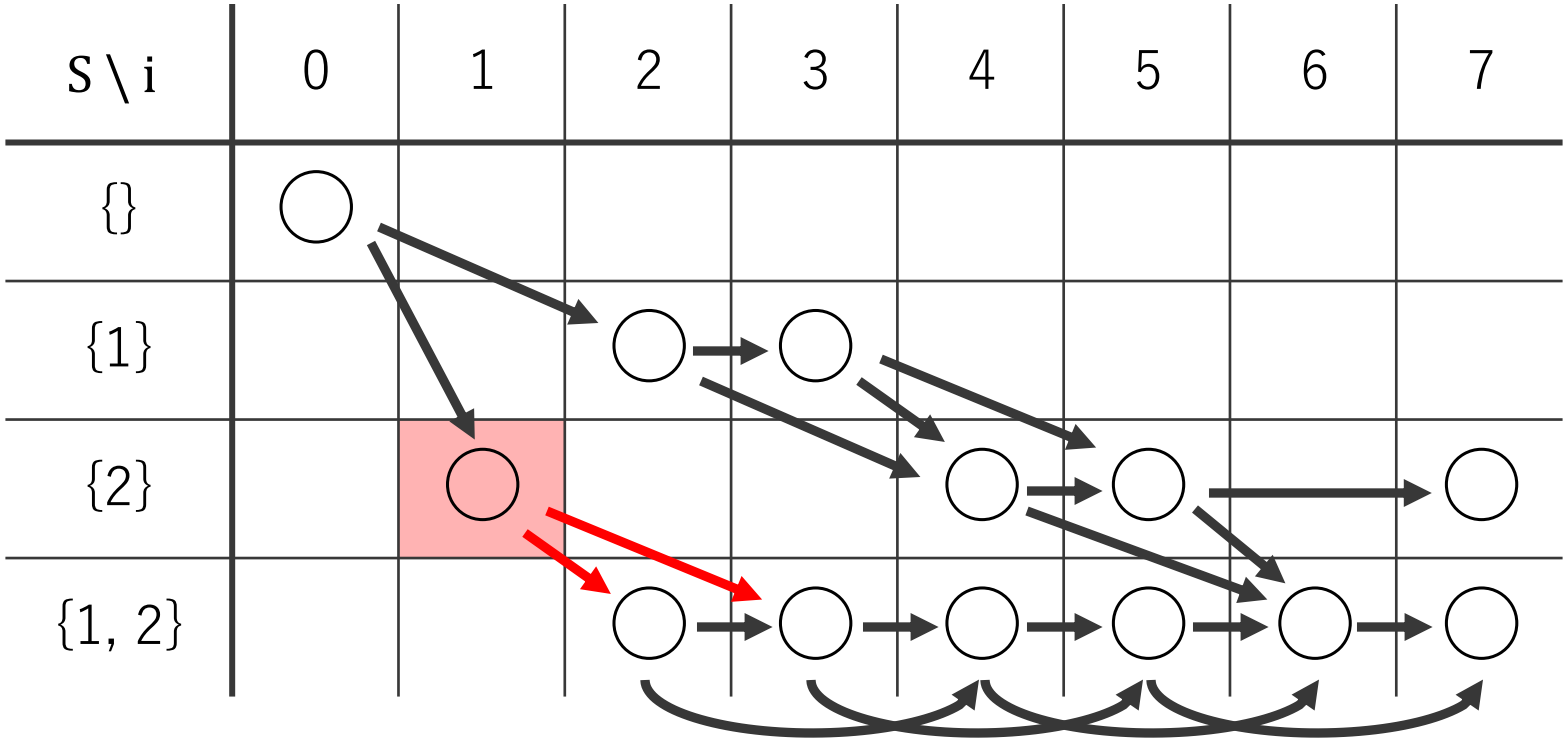
$A =$

2	1	1	2	2	1	2



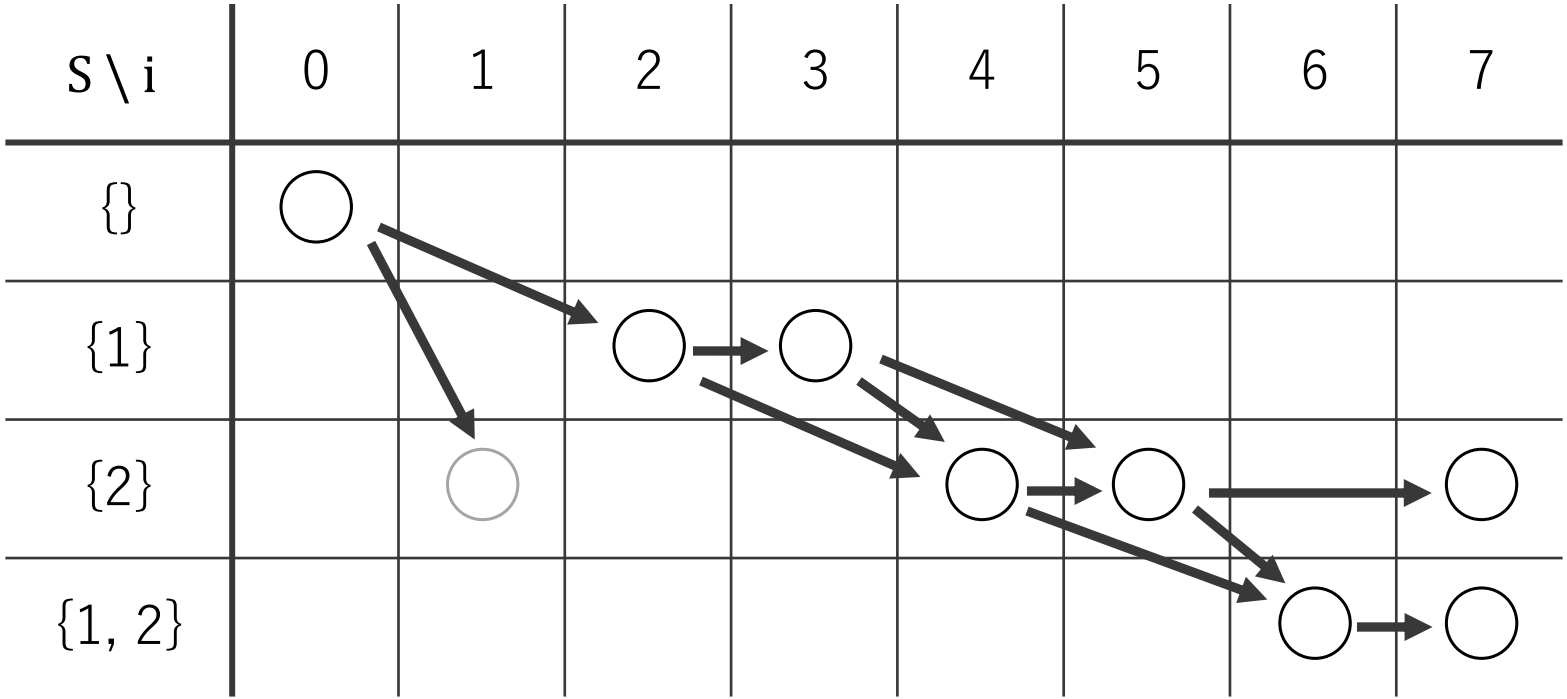
$A =$

2	1	1	2	2	1	2



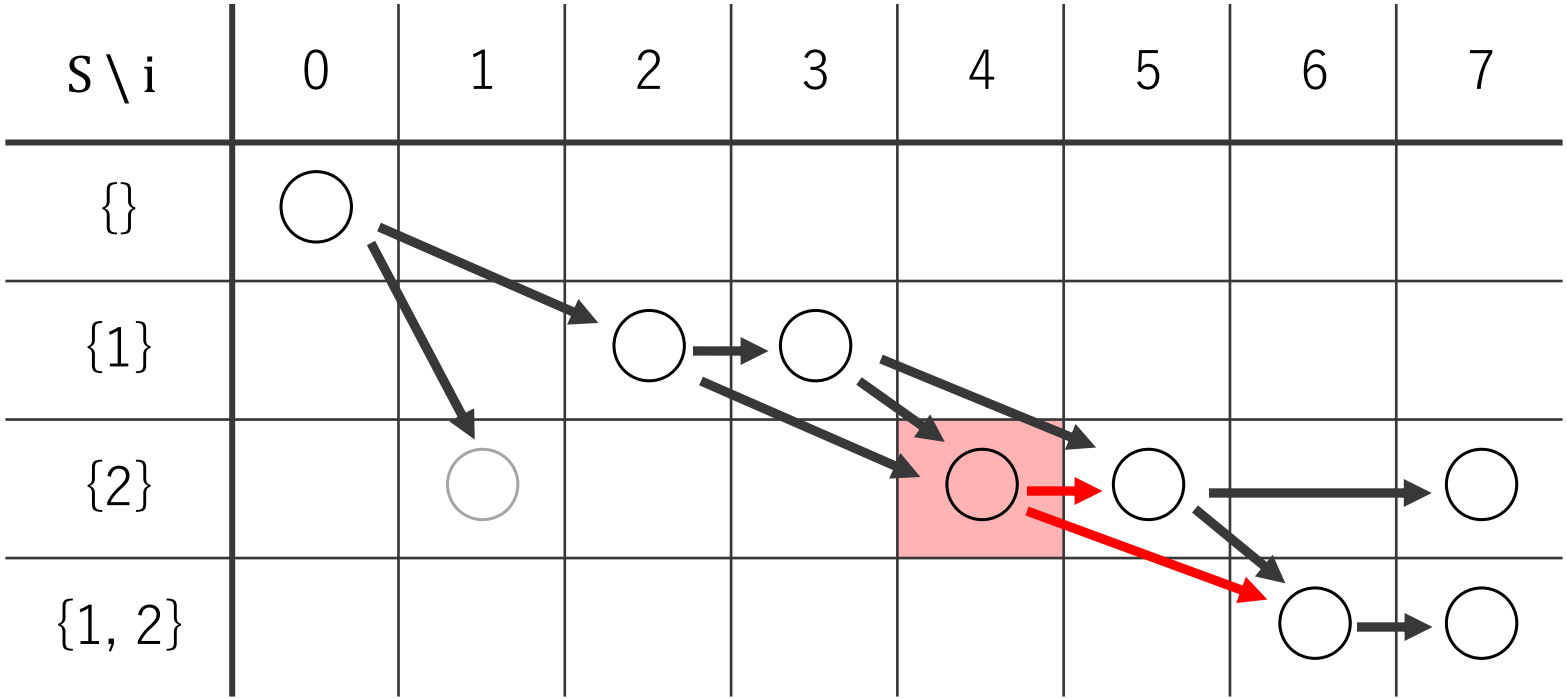
$A =$

2	1	1	2	2	1	2



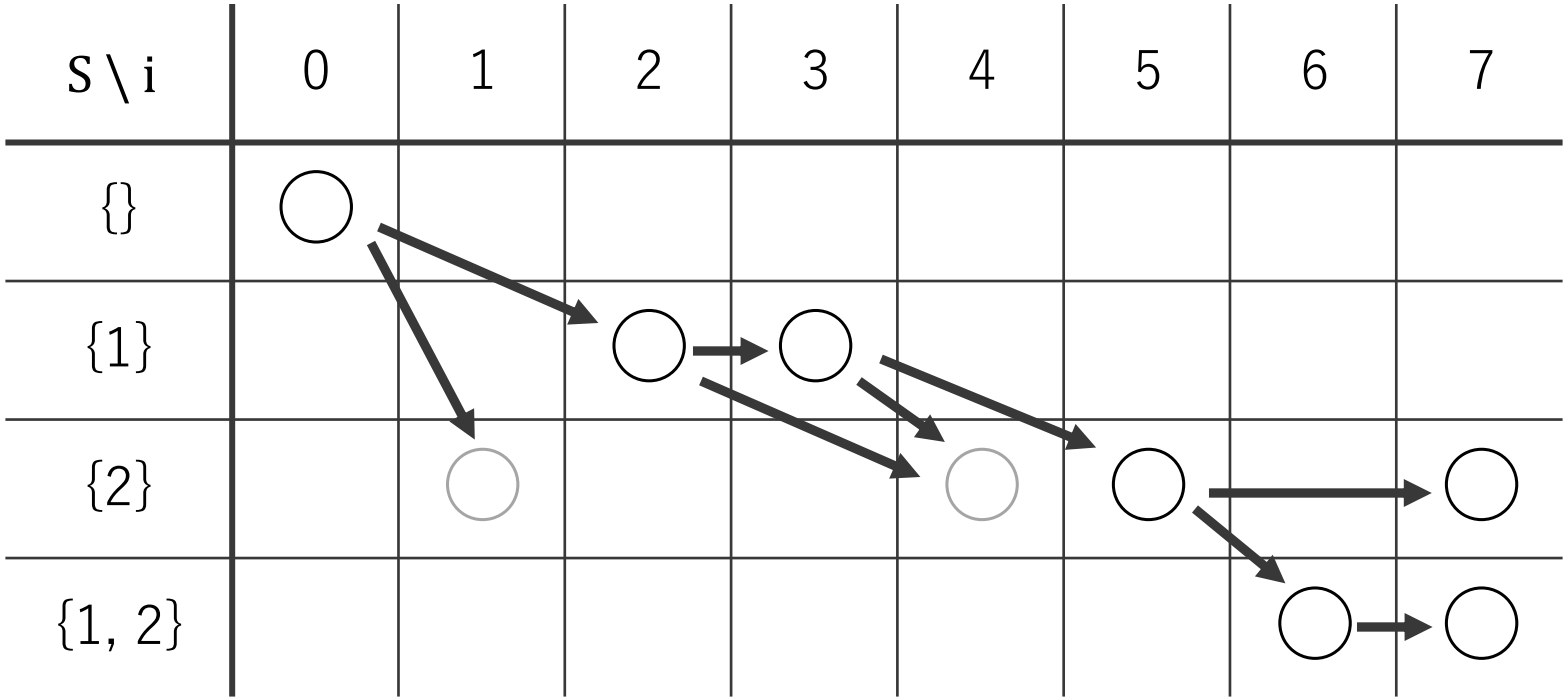
$A =$

2	1	1	2	2	1	2



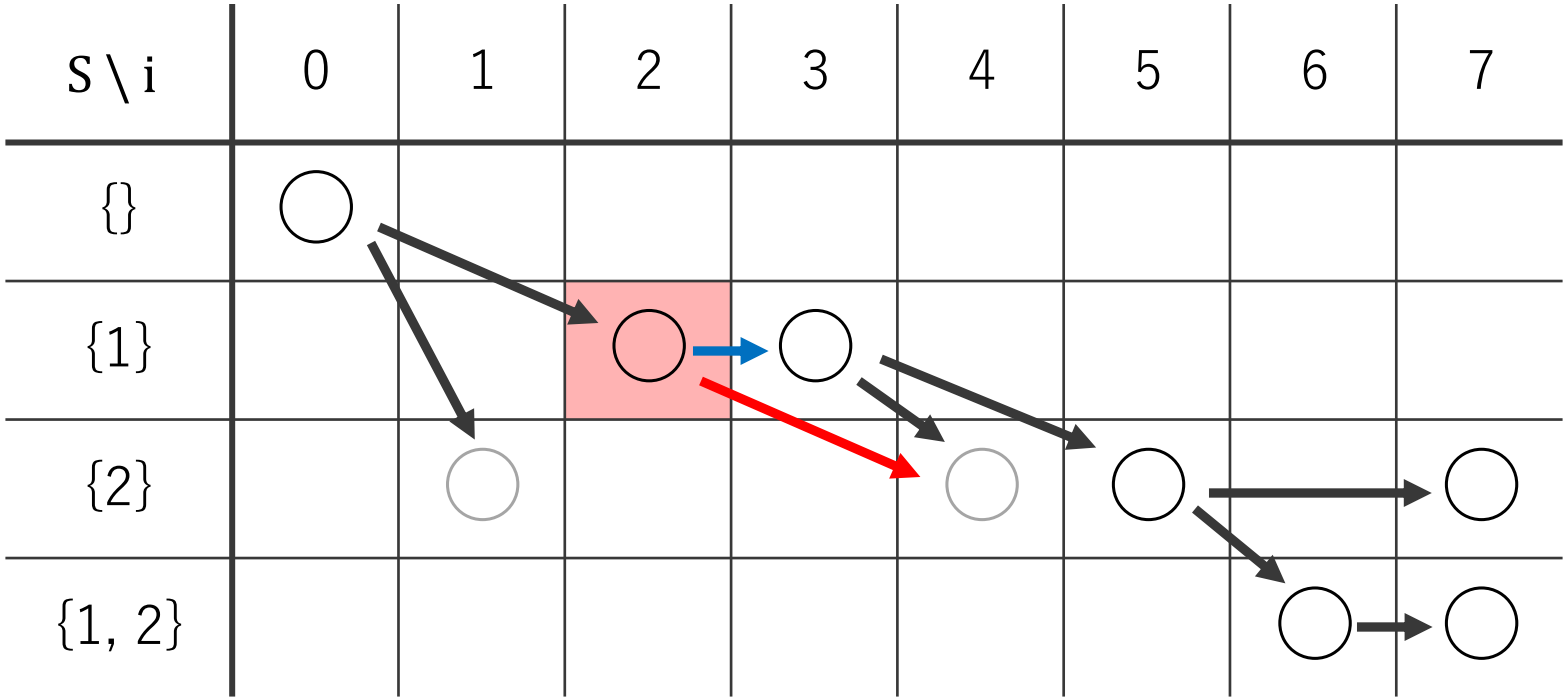
$A =$

2	1	1	2	2	1	2



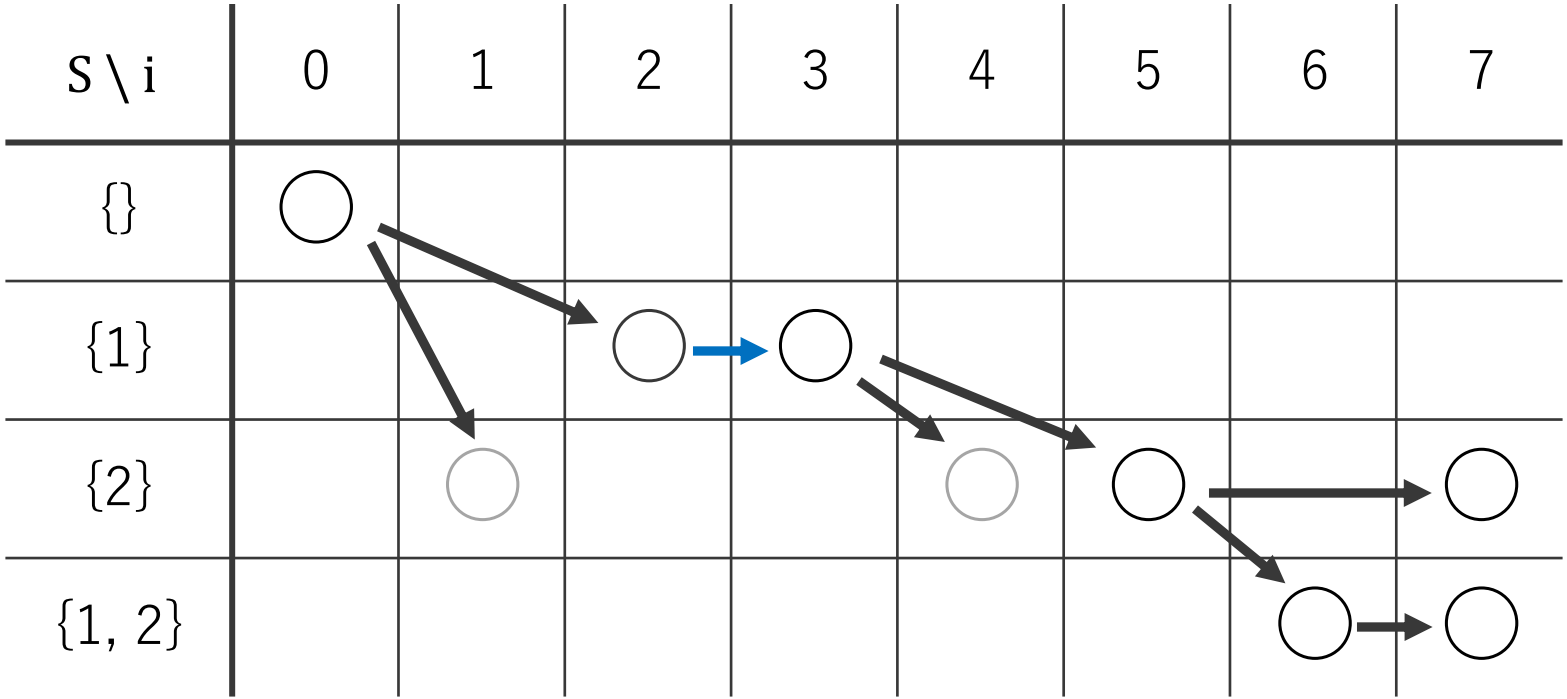
$A =$

2	1	1	2	2	1	2



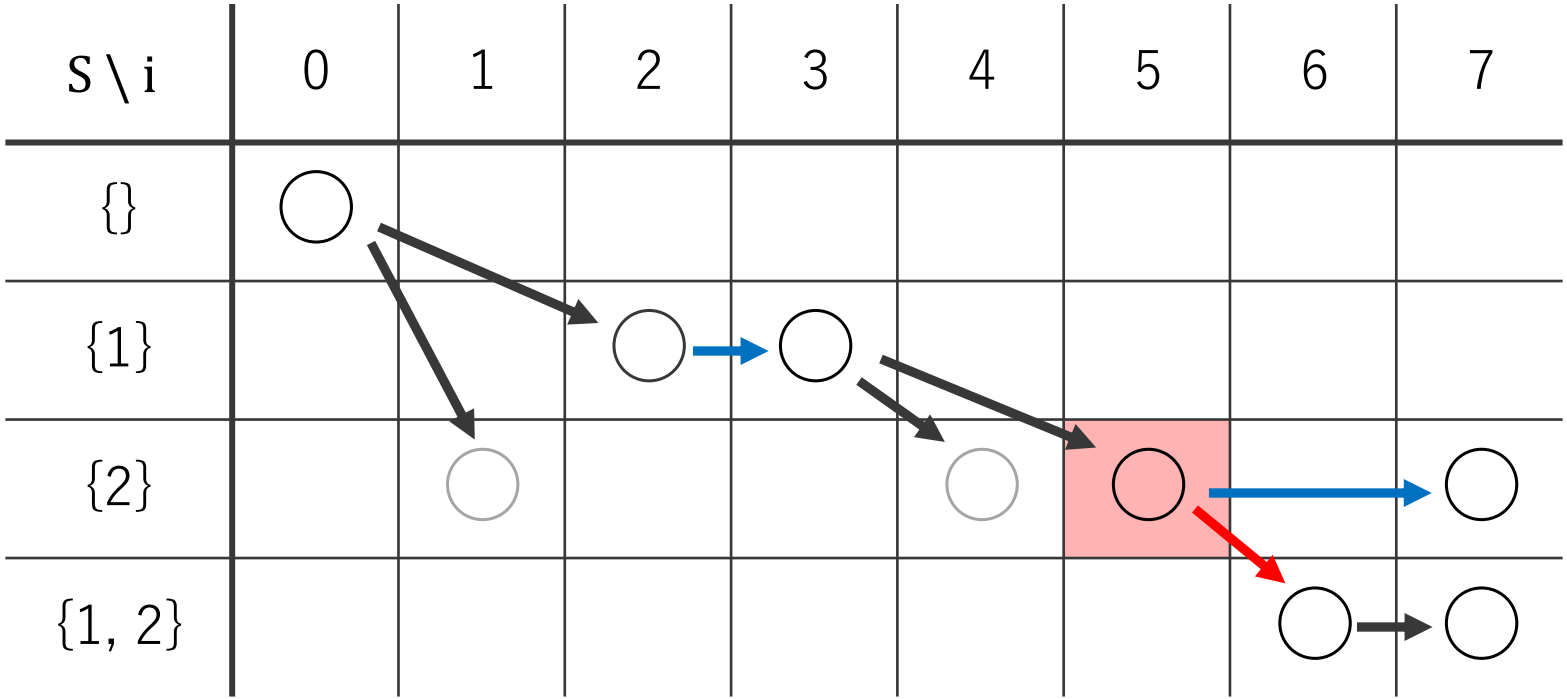
$A =$

2	1	1	2	2	1	2



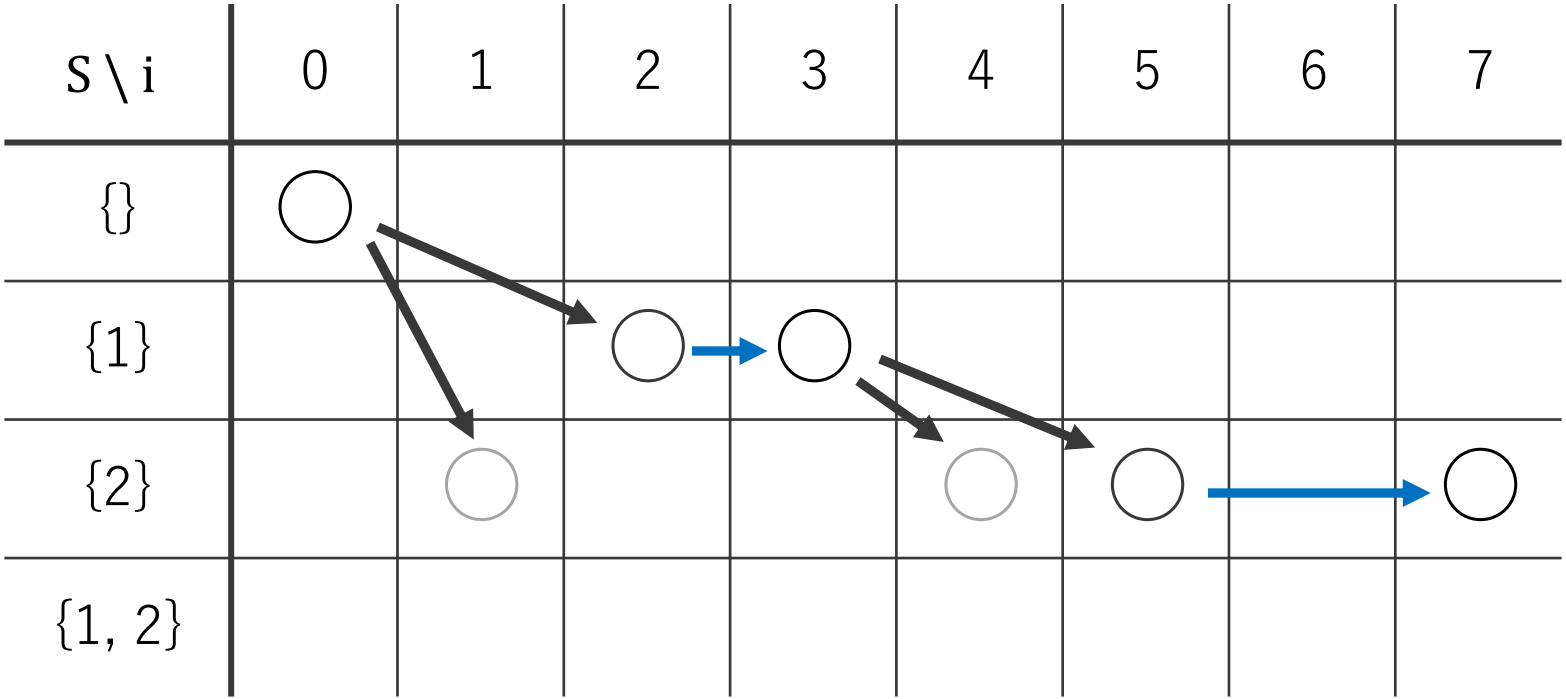
$A =$

2	1	1	2	2	1	2



$A =$

2	1	1	2	2	1	2



考察 4: DP 高速化

- 小課題 4 で用いた bit DP の復習
  - $N2^L$ 個の bool 値を保持している
  - 保持する情報の量を減らしたい
  - 不要な情報 はないだろうか？
- **各 S について、$dp_{i,S}$ = true となる最大の i さえ分かればよい**

  (各行について、最も右にある ○ の位置さえ分かればよい)

考察 4: DP 高速化

DP テーブルの各行 S について、

- $R(S)$ ：○ が書き込まれる最も右の位置

これを求めるためには、以下の値が必要：

- $R_{in}(S)$：**他の行** からの矢印が伸びてきている最も右の位置

考察 4: DP 高速化

$\mathrm{dp}_{\mathrm{i,S}}$ から $\mathrm{dp}_{\mathrm{j,T}}$ への遷移があるとき、
常に $\mathrm{S} \leq \mathrm{T}$ が成立するため

S (の要素を昇順に並べた列) の辞書順に $R_{in}(S), R(S)$ を求められる

- $R_{in}(S)$ ：これまでに処理した行から行 S への遷移 (矢印) の中で最も右にある遷移先の位置
- $R(S)$：$R_{in}(S)$ からスタートして、行 S 内での遷移を繰り返して到達できる最も右の位置
- 行 S から他の行への遷移は、$R(S)$ からのものだけでよい

$A =$

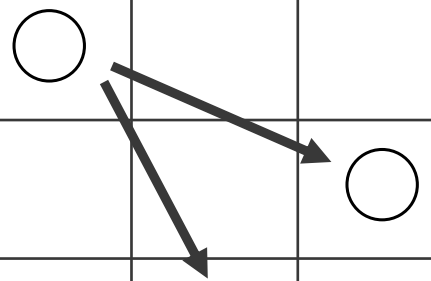
2	1	1	2	2	1	2

S \ i	0	1	2	3	4	5	6	7
{}	○							
{1}								
{2}								
{1, 2}								

$A =$

2	1	1	2	2	1	2

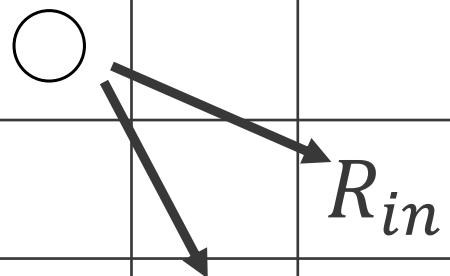
S \ i	0	1	2	3	4	5	6	7
{}	○							
{1}			○					
{2}		○						
{1, 2}								



$A =$

2	1	1	2	2	1	2

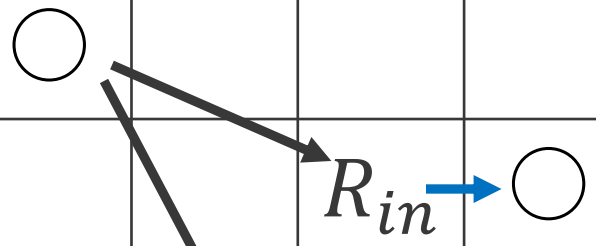
S \ i	0	1	2	3	4	5	6	7
{}	○							
{1}								
{2}		○						
{1, 2}								



$A =$

2	1	1	2	2	1	2

S \ i	0	1	2	3	4	5	6	7
{}	○							
{1}				○				
{2}		○						
{1, 2}								



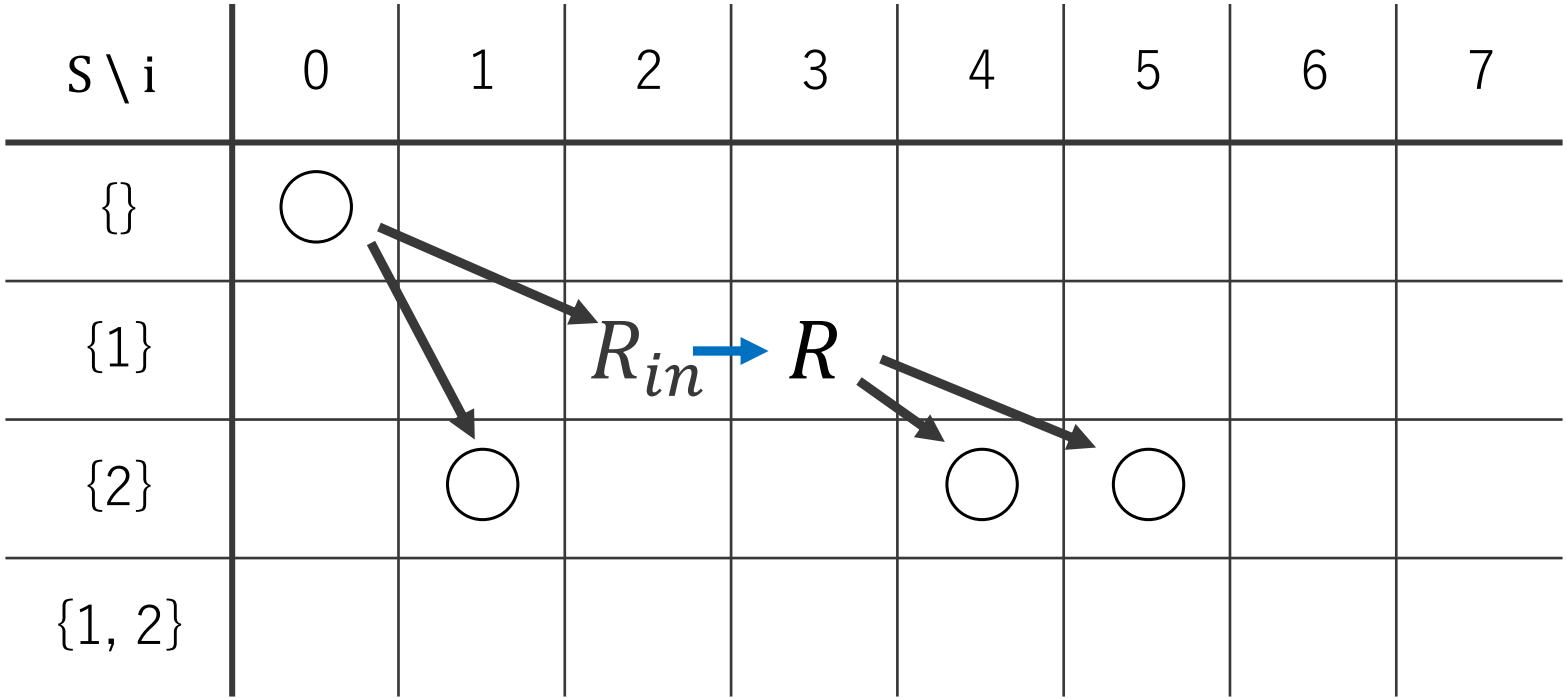
$A =$

2	1	1	2	2	1	2

S \ i	0	1	2	3	4	5	6	7
{}	○							
{1}			R_{in} → R					
{2}		○						
{1, 2}								

$A =$

2	1	1	2	2	1	2



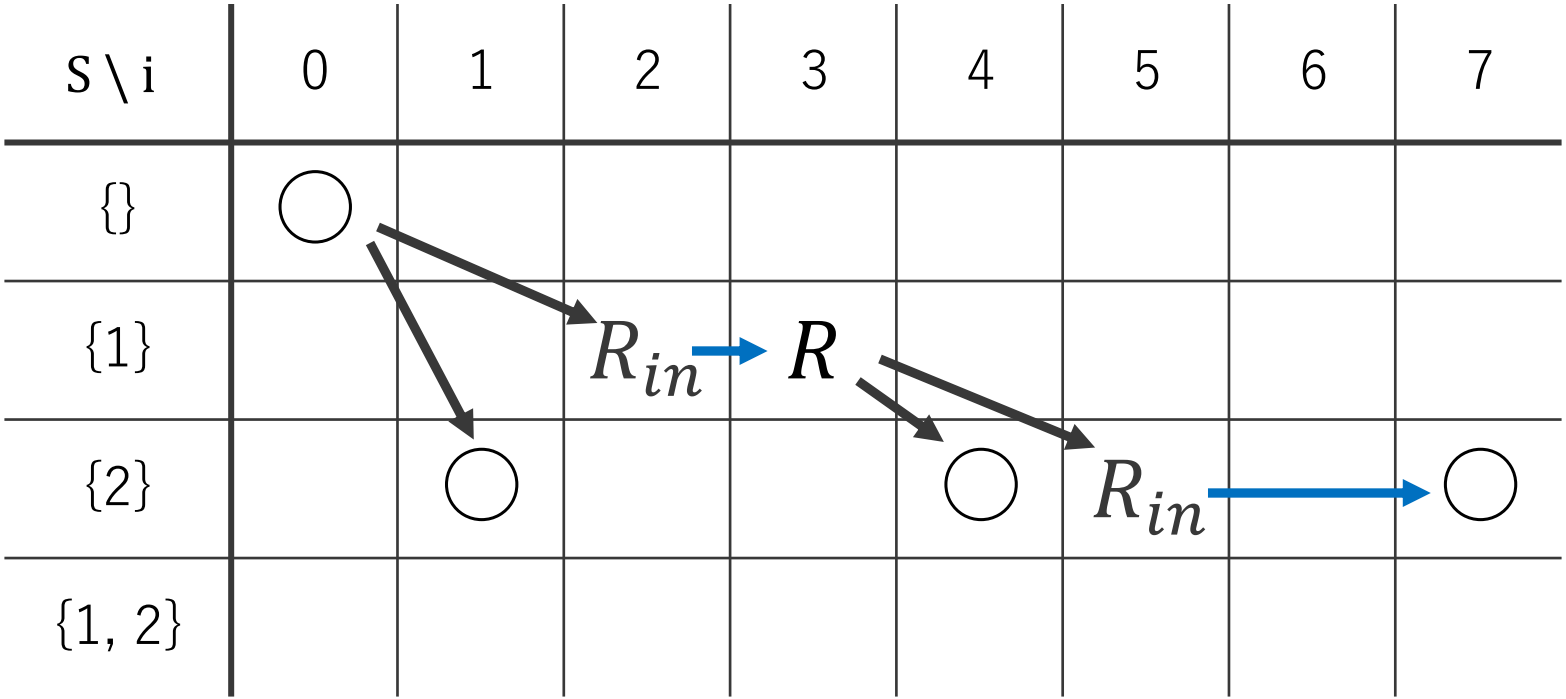
$A =$

2	1	1	2	2	1	2

S \ i	0	1	2	3	4	5	6	7
{}	○							
{1}			R_{in} →	R				
{2}		○			○	R_{in}		
{1, 2}								

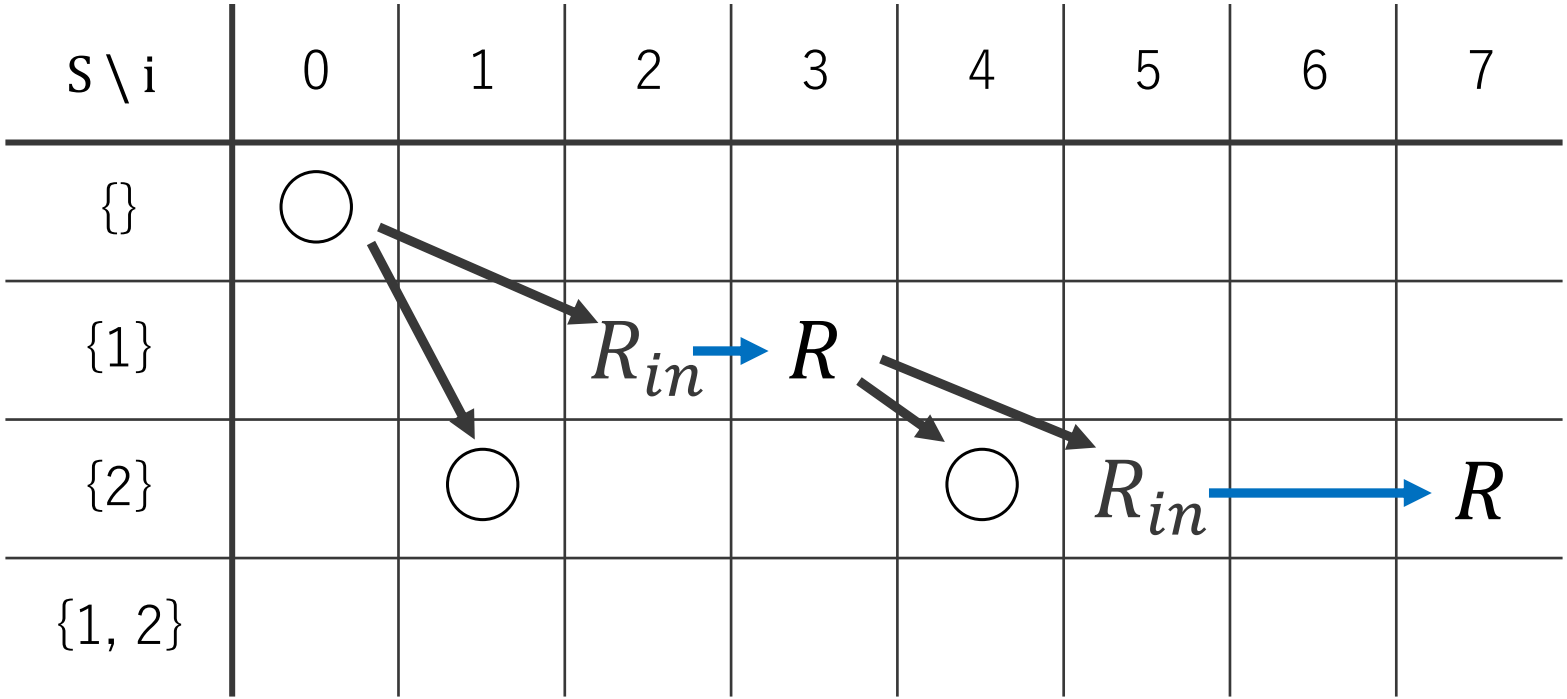
$A =$

2	1	1	2	2	1	2



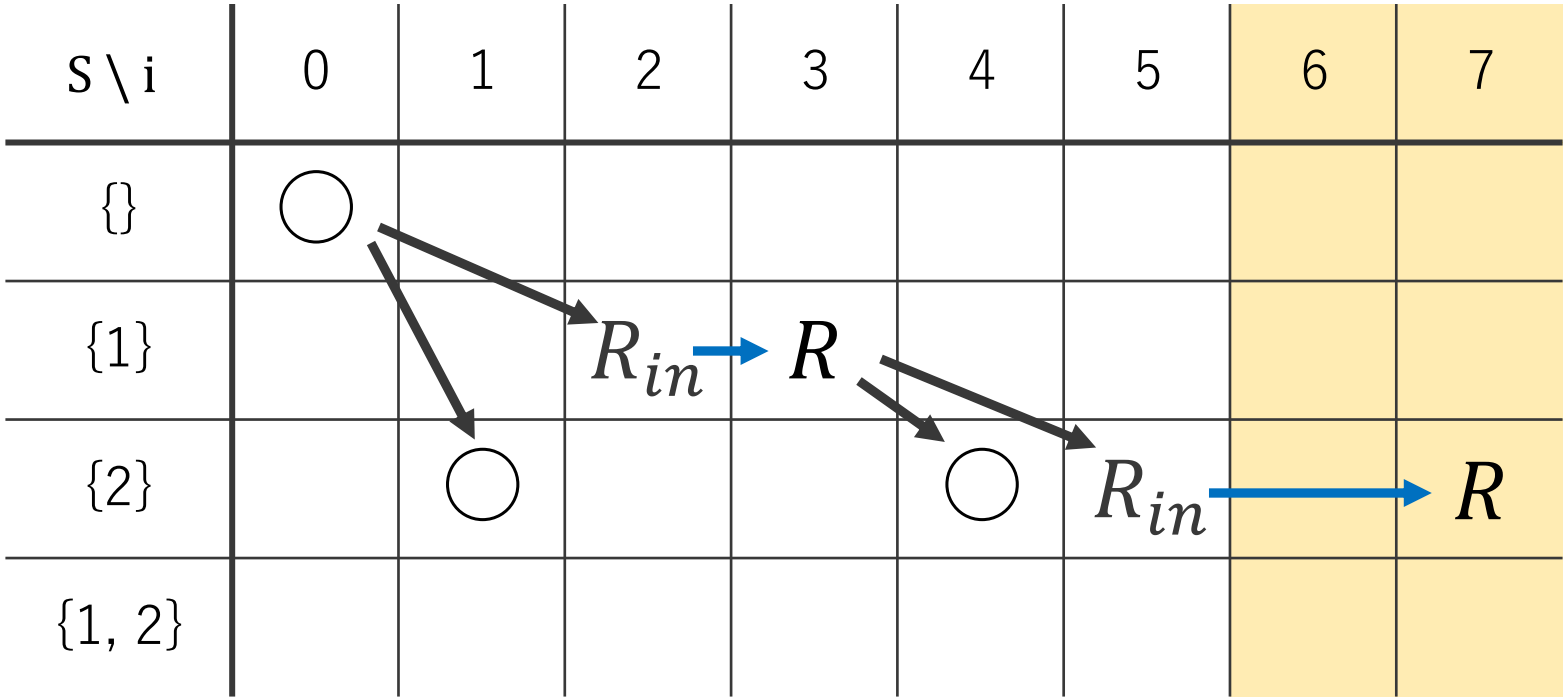
$A =$

2	1	1	2	2	1	2



$A =$

2	1	1	2	2	1	2



考察 4: DP 高速化

計算量を考える

- 保持する情報は $R_{in}(S), R(S)$ のみなので $O(2\text{^}L)$ 個
- 行う計算は主に以下の二つ
  - 各 S について、$\mathrm{R}_{in}(S)$ から $R(S)$ を求める
  - 各 S について、$R(S)$ からの遷移を処理する

考察 4: DP 高速化

計算量を考える

- 保持する情報は $R_{in}(S), R(S)$ のみなので $O(2\text{^}L)$ 個
- 行う計算は主に以下の二つ
  - 各 S について、$\mathrm{R}_{in}(S)$ から $R(S)$ を求める
  - 各 S について、$R(S)$ からの遷移を処理する　← $O(2^L \cdot L)$

考察 4: DP 高速化

計算量を考える

- 保持する情報は $R_{in}(S), R(S)$ のみなので $O(2\text{^}L)$ 個
- 行う計算は主に以下の二つ
  - 各 S について、$\mathrm{R}_{in}(S)$ から $R(S)$ を求める　← これは？
  - 各 S について、$R(S)$ からの遷移を処理する　← $O(2^L \cdot L)$

考察 4: DP 高速化

- $R_{in}(S)$ から同じ行内で右に進めるだけ進んだ位置が $R(S)$
- 「同じ行内で右に進める」とは？
  - S の値を変えない遷移が存在する
  - ＝ 1 個先 / 2 個先の要素の少なくとも一方が集合 S に含まれる
- $R(S)$：$\mathrm{A}[\mathrm{R_{in}}(\mathrm{S})], \mathrm{A}[\mathrm{R_{in}}(\mathrm{S})+1], \ldots, A[N]$ の中で、

  S に**含まれない**要素が最初に連続して現れる場所

以下のクエリが $O(2\text{^}L)$ 回飛んでくるので、オンラインで処理せよ：

$Query(S, p)$：$A_p, A_{p+1}, \ldots, A_N$ の中で、集合 S に含まれない要素が

連続して現れる最初の場所はどこか？

- あとはこの問題を解けばよい
- 様々な方針が存在し、速さに応じてどの小課題まで取れるか決まる

以下のクエリが $O(2\text{^}L)$ 回飛んでくるので、オンラインで処理せよ：

$Query(S, p)$：$A_p, A_{p+1}, \ldots, A_N$ の中で、集合 S に含まれない要素が

連続して現れる最初の場所はどこか？

- 愚直に探す：$O(2^L N)$
  - 実測上ではかなり高速に動作する

以下のクエリが $O(2\text{^}L)$ 回飛んでくるので、オンラインで処理せよ：

$Query(S,p)$：$A_p, A_{p+1}, \ldots, A_N$ の中で、集合 S に含まれない要素が

連続して現れる最初の場所はどこか？

以下のクエリが $O(2^L \cdot L^2)$ 回飛んでくるので、オンラインで処理せよ：

$Query(i,j,p)$：$A_p, A_{p+1}, \ldots, A_N$ の中で、i,j がこの順に連続して現れる

最初の場所はどこか？

以下のクエリが $O(2^L \cdot L^2)$ 回飛んでくるので、オンラインで処理せよ：

$Query(i, j, p)$：$A_p, A_{p+1}, \ldots, A_N$ の中で、i, j がこの順に連続して現れる最初の場所はどこか？

- 二分探索：$O(N + 2^L L^2 \log N)$
  - 各 i, j について、i, j が連続する場所を事前に列挙しておく

以下のクエリが $O(2^L \cdot L^2)$ 回飛んでくるので、オンラインで処理せよ：

$Query(i, j, p)$：$A_p, A_{p+1}, \ldots, A_N$ の中で、i, j がこの順に連続して現れる最初の場所はどこか？

- 前計算：$O(NL^2 + 2^L L^2)$
  - クエリの入力は $O(NL^2)$ 通り存在する
  - i の降順にすべて前計算しておく

以下のクエリが $O(2^L \cdot L^2)$ 回飛んでくるので、オンラインで処理せよ：

$Query(i, j, p)$：$A_p, A_{p+1}, \ldots, A_N$ の中で、i, j がこの順に連続して現れる最初の場所はどこか？

- バケット法：$O(N + 2^L L^2)$
  - **p が L^2 の倍数**であるような入力についてのみ前計算
    - そのような入力は $O(L \cdot L \cdot \frac{N}{L^2}) = O(N)$ 通り存在する
  - $Query(i, j, p)$ に答えるときは、$L^2\lceil\frac{p}{L^2}\rceil$ 番目までは愚直に見て、それ以降は前計算の値を用いる

まとめ

subtask 5 $N \leq 5 \times 10^5, A_i \leq 15$

subtask 6 $N \leq 5 \times 10^5, A_i \leq 21$

subtask 7 $N \leq 5 \times 10^6, A_i \leq 21$

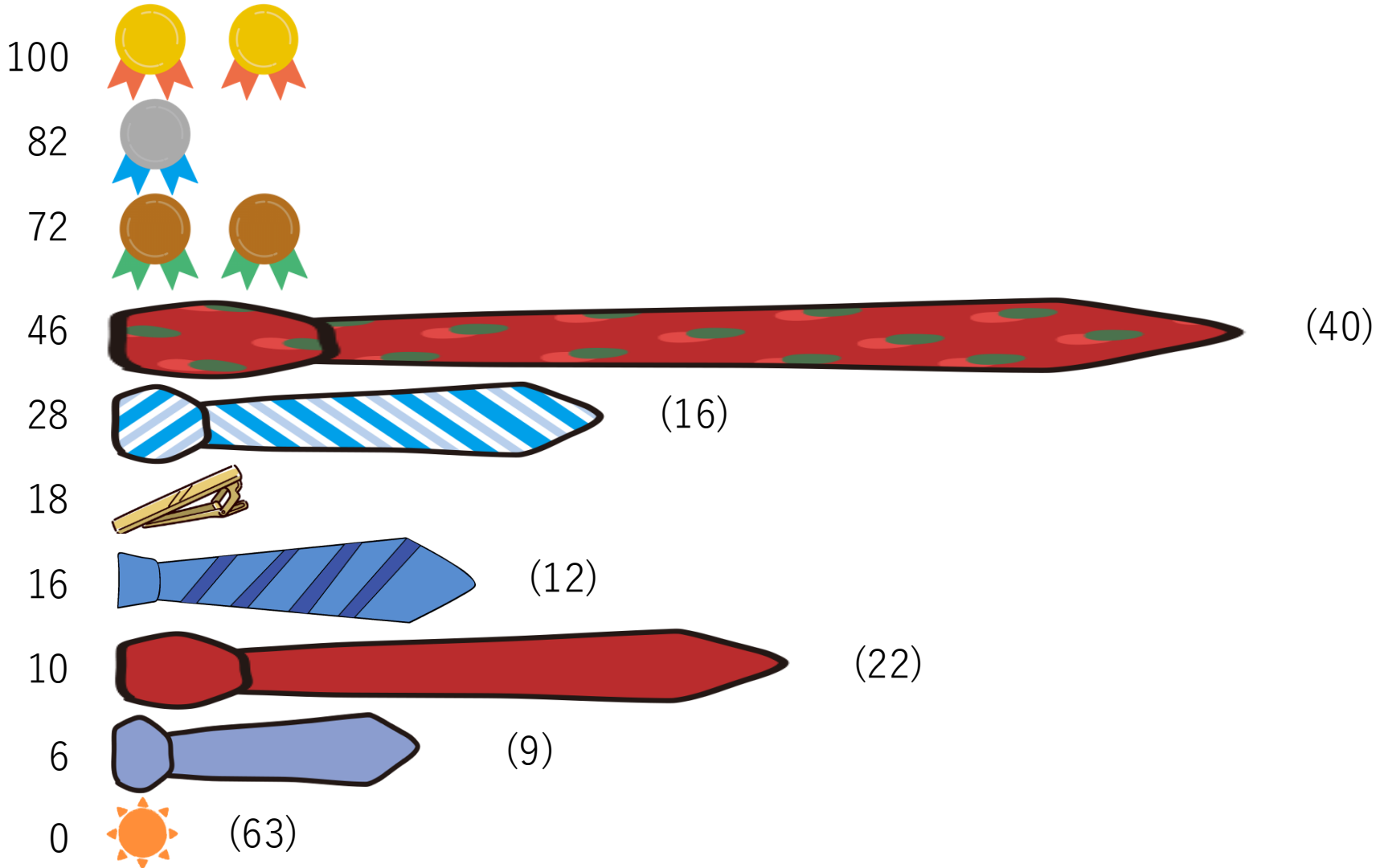
	時間計算量	空間計算量	想定得点
愚直に探す	$O(2^L N)$	$O(N + 2^L)$	72 (~sub5)
二分探索	$O(N + 2^L L^2 \log N)$	$O(N + 2^L)$	72 (~sub5)
前計算	$O(NL^2 + 2^L L^2)$	$O(NL^2 + 2^L)$	82 (~sub6)
バケット法	$O(N + 2^L L^2)$	$O(N + 2^L)$	100 (~sub7)

別解

- Dijkstra 法の要領で、$R_{in}(S)$ の昇順に $R_{in}(S), R(S)$ を求める
    - $Query(i, j, p)$ が p の昇順に飛んでくる ので、

      高速に処理することができ、満点を取れる
- Dilworth の定理を用いて考察することもできる
    - 結局同じ問題に帰着される

得点分布


100
82
72
46
28
18
16
10
6
0
(40)
(16)
(12)
(22)
(9)
(63)