Threshold Behaviors in Advice Complexity

Knapsack with Removability

Hans-Joachim Böckenhauer, Jan Dreier, **Fabian Frei**, Peter Rossmanith

August 28, 2020 – Virtual satellite workshop of MFCS 2020

Online Problems and Advice

Brief Recapitulation

Recapitulation

Online Problems

- Instance revealed piecewise
- Solution required piecewise
- Algorithm outputs solution parts without full information

Recapitulation

Online Problems: Very Hard

- Online-ness is a severe restriction
- In exchange: Unbounded time and space resources
- Thus no time/space complexity analysis

Recapitulation

Online Problems: Very Hard

- Online-ness is a severe restriction
- In exchange: Unbounded time and space resources
- Thus no time/space complexity analysis

Assessing Online Algorithms

- Compare online solution to an optimal solution
- That is: Can online algorithm *compete* with offline one?
- Next slide: Formal definition of *competitivity*

Recapitulation

Defining Competitivity (Strict and for Maximization Problems)

- Competitivity of an online algorithm A on an instance I:

$$\frac{\text{Gain of an optimal solution to } I}{\text{Gain of } A\text{'s online solution to } I}$$

Recapitulation

Defining Competitivity (Strict and for Maximization Problems)

- Competitivity of an online algorithm A on an instance I:

$$\frac{\text{Gain of an optimal solution to } I}{\text{Gain of } A\text{'s online solution to } I}$$

- Competitivity of an online algorithm: Worst case, i.e.,

$$\max_{I \in \mathcal{I}} \frac{\text{Gain of an optimal solution to } I}{\text{Gain of } A\text{'s online solution to } I}$$

Recapitulation

Defining Competitivity (Strict and for Maximization Problems)

- Competitivity of an online algorithm A on an instance I:

$$\frac{\text{Gain of an optimal solution to } I}{\text{Gain of } A\text{'s online solution to } I}$$

- Competitivity of an online algorithm: Worst case, i.e.,

$$\max_{I \in \mathcal{I}} \frac{\text{Gain of an optimal solution to } I}{\text{Gain of } A\text{'s online solution to } I}$$

- Competitivity of a problem: Best online algorithms, i.e.,

$$\min_{A \in \mathcal{A}} \max_{I \in \mathcal{I}} \frac{\text{Gain of an optimal solution to } I}{\text{Gain of } A\text{'s online solution to } I}$$

This Much for Online Problems

Now Advice

Advice Complexity

Motivation

- Online algorithm lacks information about future
- This makes online problems very hard
- But how much information is lacking exactly?

Advice Complexity

Motivation

- Online algorithm lacks information about future
- This makes online problems very hard
- But how much information is lacking exactly?

Measuring the lack of information

- Size of instance?
- Size of solution?
- Need for better, general measure
- Established measuring tool: Advice

Advice Complexity

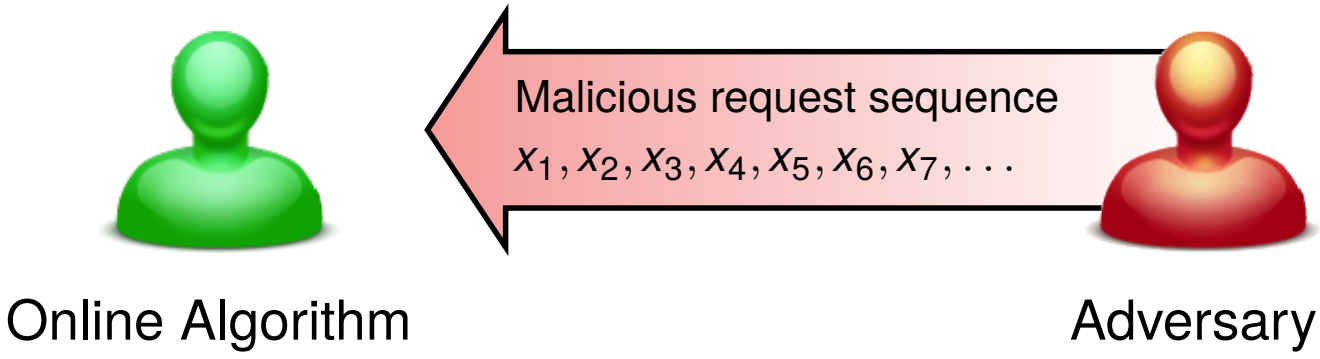
Advice model

- Omniscient oracle provides online algorithm with advice
- Advice has the form of an infinite bit string
- One tailor-made advice string for each instance
- Online algorithm reads as many bits as it wants
- Number of bits read: *Advice complexity*

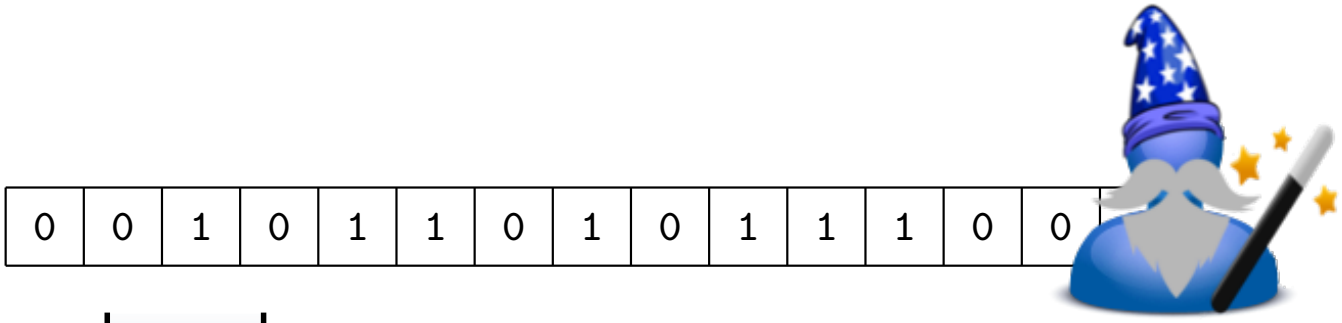
Online Computation

Online Algorithm

Online Computation



Advice Complexity



Oracle

Malicious request sequence

$x_1, x_2, x_3, x_4, x_5, x_6, x_7, \ldots$

Online Algorithm

Adversary

Advice Complexity

Trade-Off

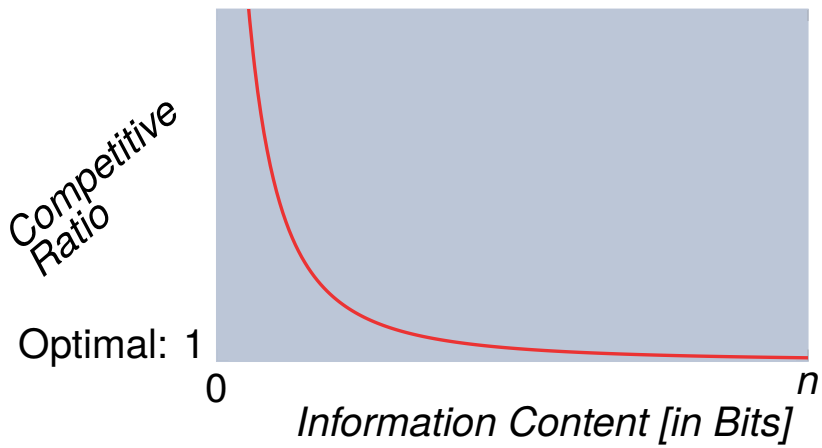
- Improve competitivity, but minimize advice
- Remarkable behavior differences between problems

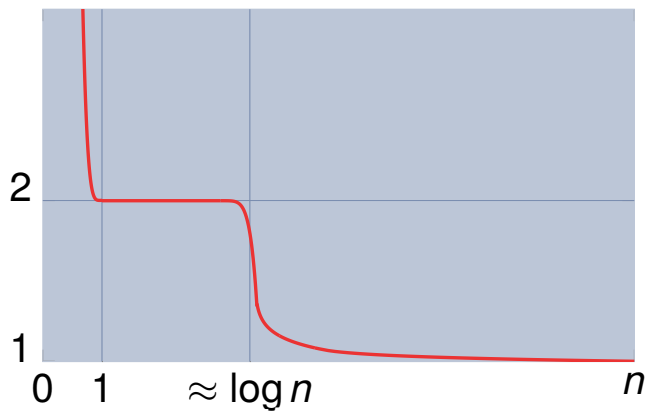
Common: Continuous

Every single additional bit improves the competitivity

Knapsack: Thresholds

Jumps at some thresholds, stagnating elsewhere





Online Knapsack

Classical Version

Classical Online Knapsack

Problem Definition

- Knapsack of capacity 1
- Online Instance: Sequence of n items with $s_1, \ldots, s_n$
- Online Output:
  - Pack or discard each item immediately
  - The decisions are permanent
  - Never exceed the capacity
- Goal: Maximize packed volume

Classical Online Knapsack

Problem Definition

- Knapsack of capacity 1
- Online Instance: Sequence of n items with $s_1, \ldots, s_n$
- Online Output:
  - Pack or discard each item immediately
  - The decisions are permanent
  - Never exceed the capacity
- Goal: Maximize packed volume

Remark

- This is the *proportional*/*simple*/*unweighted* problem
- That is: No size-value distinction for items

Knapsack with Removability

Model by Iwama and Taketomi, ICALP 2002

Knapsack with Removability

Definition of Classical Online Knapsack

- Knapsack of capacity 1
- Online Instance: Sequence of n items with $s_1, \ldots, s_n$
- Online Output:
  - Pack or discard each item immediately
  - The decisions are permanent
  - Never exceed the capacity
- Goal: Maximize packed volume

Knapsack with Removability

Definition of Classical Online Knapsack

- Knapsack of capacity 1
- Online Instance: Sequence of n items with $s_1, \ldots, s_n$
- Online Output:
  - Pack or discard each item immediately
  - The decisions are permanent
  - Never exceed the capacity
- Goal: Maximize packed volume

Knapsack with Removability

Definition of Classical Online Knapsack

- Knapsack of capacity 1
- Online Instance: Sequence of n items with $s_1, \ldots, s_n$
- Online Output:
  - Pack or discard each item immediately
  - The decisions are permanent
  - Never exceed the capacity
- Goal: Maximize packed volume

Knapsack with Removability

Definition of Knapsack with Removability

- Knapsack of capacity 1
- Online Instance: Sequence of n items with $s_1, \ldots, s_n$
- Online Output:
  - Pack or discard each item immediately
  - Packed items can be removed
  - Never exceed the capacity
- Goal: Maximize packed volume

Knapsack with Removability

Clarification on Removability

- Any packed item can be removed from the knapsack
- No restrictions. Algorithm may remove ...
  - ... arbitrarily many items ...
  - ... at arbitrary points in time
- However, once an item is removed, it is gone for good

Knapsack with Removability

Natural Model

- Example: Storage room
- Keep useful things as they come along
- Space will run out
- Need to start disposing
- Goal: Most useful collection
- How much information about future needed?

Knapsack with Removability

Two Results Known So Far

- Without advice, the competitivity is exactly the golden ratio $\Phi \approx 1.618$
  [Iwama and Taketomi, ICALP 2002]
- There is $10/7$-approximative algorithm using a single random bit
  [Han et al., TCS 2015]

Knapsack with Removability

Two Results Known So Far

- Without advice, the competitivity is exactly the golden ratio $\Phi \approx 1.618$
  [Iwama and Taketomi, ICALP 2002]
- There is $10/7$-approximative algorithm using a single random bit
  [Han et al., TCS 2015]
- Thus there is $10/7$-competitive algorithm using a single advice bit

Knapsack with Removability

Competitivity vs. Advice: Our Contribution

- Optimality requires 1 advice bit per item asymptotically
- Near optimality with constant advice
- Improved bounds for one advice bit

Knapsack with Removability

Competitivity vs. Advice: Our Contribution

- Optimality requires 1 advice bit per item asymptotically
- Near optimality with constant advice
- Improved bounds for one advice bit

Knapsack with Removability

Competitivity vs. Advice: Our Contribution

- Optimality requires 1 advice bit per item asymptotically
- Near optimality with constant advice
- Improved bounds for one advice bit

Knapsack with Removability

Competitivity vs. Advice: Our Contribution

- Optimality requires 1 advice bit per item asymptotically
- Near optimality with constant advice
- Improved bounds for one advice bit

A Single Advice Bit

Upper and Lower Bound

1 Advice Bit: Lower Bound

Hard Instance Family

	x_1	x_2	x_3	y_2	y_3
I_1:	ψ	ψ^2	$1-\psi^2+\varepsilon$		
I_2:	ψ	ψ^2	$1-\psi^2+\varepsilon$	$1-\psi^2$	
I_3:	ψ	ψ^2	$1-\psi^2+\varepsilon$		$\psi^2-\varepsilon$

1 Advice Bit: Lower Bound

Hard Instance Family

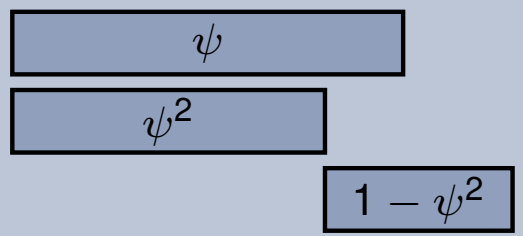
	x_1	x_2	x_3	y_2	y_3
I_1:	ψ	ψ^2	$1-\psi^2+\varepsilon$		
I_2:	ψ	ψ^2	$1-\psi^2+\varepsilon$	$1-\psi^2$	
I_3:	ψ	ψ^2	$1-\psi^2+\varepsilon$		$\psi^2-\varepsilon$

where $\psi \approx 0.78$ is the positive root of $2(1-x^2) = x$

1 Advice Bit: Lower Bound

Hard Instance Family

	x_1	x_2	x_3	y_2	y_3
I_1:	ψ	ψ^2	$1-\psi^2+\varepsilon$		
I_2:	ψ	ψ^2	$1-\psi^2+\varepsilon$	$1-\psi^2$	
I_3:	ψ	ψ^2	$1-\psi^2+\varepsilon$		$\psi^2-\varepsilon$

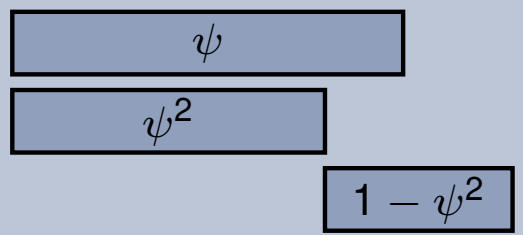


where $\psi \approx 0.78$ is the positive root of $2(1-x^2) = x$

1 Advice Bit: Lower Bound

Hard Instance Family

	x_1	x_2	x_3	y_2	y_3
I_1:	ψ	ψ^2	$1-\psi^2+\varepsilon$		
I_2:	ψ	ψ^2	$1-\psi^2+\varepsilon$	$1-\psi^2$	
I_3:	ψ	ψ^2	$1-\psi^2+\varepsilon$		$\psi^2-\varepsilon$

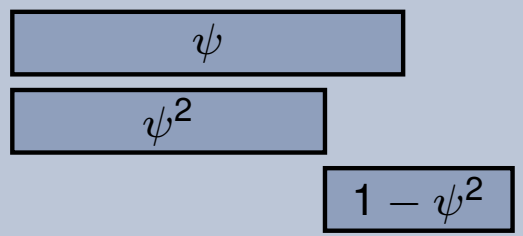


where $\psi \approx 0.78$ is the positive root of $2(1-x^2) = x$

1 Advice Bit: Lower Bound

Hard Instance Family

	x_1	x_2	x_3	y_2	y_3
I_1:	ψ	ψ^2	$1-\psi^2+\varepsilon$		
I_2:	ψ	ψ^2	$1-\psi^2+\varepsilon$	$1-\psi^2$	
I_3:	ψ	ψ^2	$1-\psi^2+\varepsilon$		$\psi^2-\varepsilon$

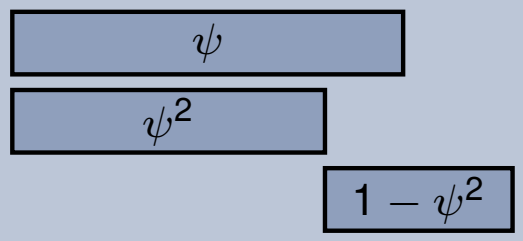


where $\psi \approx 0.78$ is the positive root of $2(1-x^2) = x$

1 Advice Bit: Lower Bound

Hard Instance Family

	x_1	x_2	x_3	y_2	y_3
I_1:	ψ	ψ^2	$1-\psi^2+\varepsilon$		
I_2:	ψ	ψ^2	$1-\psi^2+\varepsilon$	$1-\psi^2$	
I_3:	ψ	ψ^2	$1-\psi^2+\varepsilon$		$\psi^2-\varepsilon$



where $\psi \approx 0.78$ is the positive root of $2(1-x^2) = x$

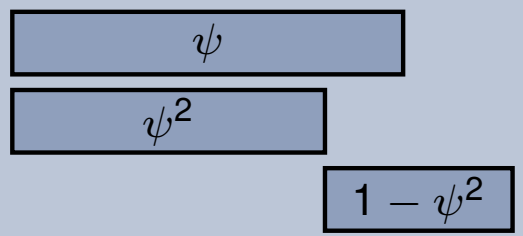
Competitive Analysis

	Unique Optimum	Second Best	Competitivity
I_1:	ψ	ψ^2	ψ/ψ^2
I_2:	1	$2(1-\psi^2)+\varepsilon$	$1/(2(1-\psi^2)+\varepsilon)$
I_3:	1	ψ	$1/\psi$

1 Advice Bit: Lower Bound

Hard Instance Family

	x_1	x_2	x_3	y_2	y_3
I_1:	ψ	ψ^2	$1-\psi^2+\varepsilon$		
I_2:	ψ	ψ^2	$1-\psi^2+\varepsilon$	$1-\psi^2$	
I_3:	ψ	ψ^2	$1-\psi^2+\varepsilon$		$\psi^2-\varepsilon$



where $\psi \approx 0.78$ is the positive root of $2(1-x^2) = x$

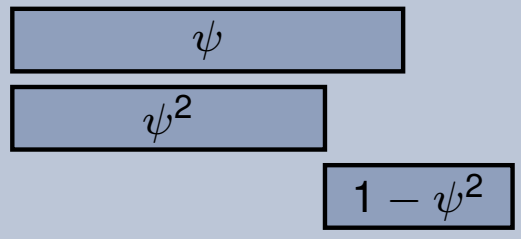
Competitive Analysis

	Unique Optimum	Second Best	Competitivity
I_1:	ψ	ψ^2	ψ/ψ^2
I_2:	1	$2(1-\psi^2)+\varepsilon$	$1/(2(1-\psi^2)+\varepsilon)$
I_3:	1	ψ	$1/\psi$

1 Advice Bit: Lower Bound

Hard Instance Family

	x_1	x_2	x_3	y_2	y_3
I_1:	ψ	ψ^2	$1-\psi^2+\varepsilon$		
I_2:	ψ	ψ^2	$1-\psi^2+\varepsilon$	$1-\psi^2$	
I_3:	ψ	ψ^2	$1-\psi^2+\varepsilon$		$\psi^2-\varepsilon$



where $\psi \approx 0.78$ is the positive root of $2(1-x^2) = x$

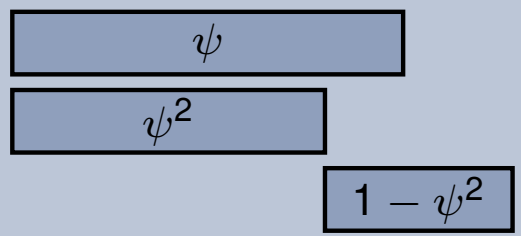
Competitive Analysis

	Unique Optimum	Second Best	Competitivity
I_1:	ψ	ψ^2	ψ/ψ^2
I_2:	1	$2(1-\psi^2)+\varepsilon$	$1/(2(1-\psi^2)+\varepsilon)$
I_3:	1	ψ	$1/\psi$

1 Advice Bit: Lower Bound

Hard Instance Family

	x_1	x_2	x_3	y_2	y_3
I_1:	ψ	ψ^2	$1-\psi^2+\varepsilon$		
I_2:	ψ	ψ^2	$1-\psi^2+\varepsilon$	$1-\psi^2$	
I_3:	ψ	ψ^2	$1-\psi^2+\varepsilon$		$\psi^2-\varepsilon$



where $\psi \approx 0.78$ is the positive root of $2(1-x^2) = x$

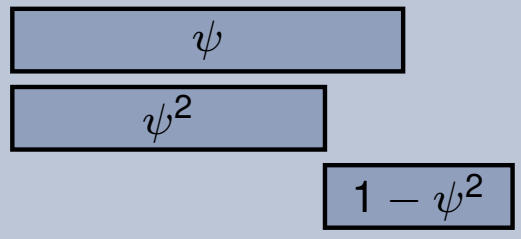
Competitive Analysis

	Unique Optimum	Second Best	Competitivity
I_1:	ψ	ψ^2	ψ/ψ^2
I_2:	1	$2(1-\psi^2)+\varepsilon$	$1/(2(1-\psi^2)+\varepsilon)$
I_3:	1	ψ	$1/\psi$

1 Advice Bit: Lower Bound

Hard Instance Family

	x_1	x_2	x_3	y_2	y_3
I_1:	ψ	ψ^2	$1-\psi^2+\varepsilon$		
I_2:	ψ	ψ^2	$1-\psi^2+\varepsilon$	$1-\psi^2$	
I_3:	ψ	ψ^2	$1-\psi^2+\varepsilon$		$\psi^2-\varepsilon$



where $\psi \approx 0.78$ is the positive root of $2(1-x^2) = x$

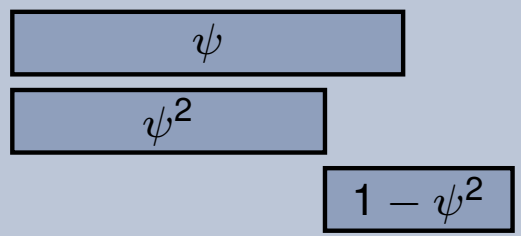
Competitive Analysis

	Unique Optimum	Second Best	Competitivity
I_1:	ψ	ψ^2	$1/\psi$
I_2:	1	$2(1-\psi^2)+\varepsilon$	$1/\psi+\varepsilon$
I_3:	1	ψ	$1/\psi$

1 Advice Bit: Lower Bound

Hard Instance Family

	x_1	x_2	x_3	y_2	y_3
I_1:	ψ	ψ^2	$1-\psi^2+\varepsilon$		
I_2:	ψ	ψ^2	$1-\psi^2+\varepsilon$	$1-\psi^2$	
I_3:	ψ	ψ^2	$1-\psi^2+\varepsilon$		$\psi^2-\varepsilon$



where $\psi \approx 0.78$ is the positive root of $2(1-x^2) = x$

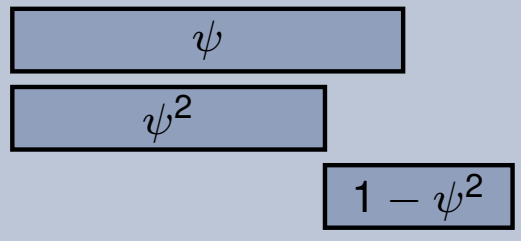
Competitive Analysis

	Unique Optimum	Second Best	Competitivity
I_1:	ψ	ψ^2	$1/\psi$
I_2:	1	$2(1-\psi^2)+\varepsilon$	$1/\psi+\varepsilon$
I_3:	1	ψ	$1/\psi$

1 Advice Bit: Lower Bound

Hard Instance Family

	x_1	x_2	x_3	y_2	y_3
I_1:	ψ	ψ^2	$1-\psi^2+\varepsilon$		
I_2:	ψ	ψ^2	$1-\psi^2+\varepsilon$	$1-\psi^2$	
I_3:	ψ	ψ^2	$1-\psi^2+\varepsilon$		$\psi^2-\varepsilon$



where $\psi \approx 0.78$ is the positive root of $2(1-x^2) = x$

Competitive Analysis

	Unique Optimum	Second Best	Competitivity	
I_1:	ψ	ψ^2		$1/\psi \approx 1.28$
I_2:	1	$2(1-\psi^2)+\varepsilon$		vs.
I_3:	1	ψ	$1/\psi$	$10/7 \approx 1.43$

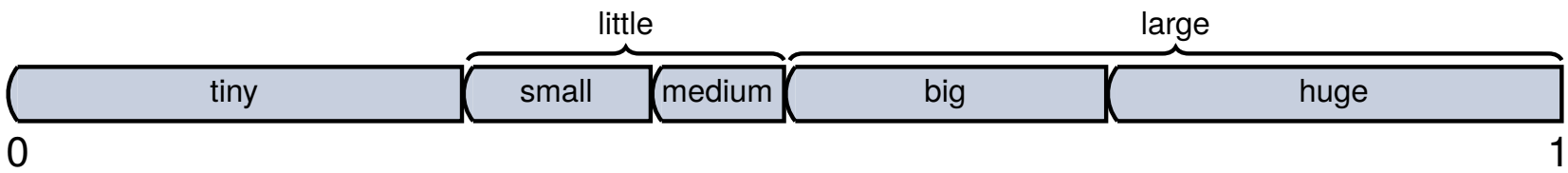
1 Advice Bit

Upper Bound

1 Advice Bit: Improved Upper Bound

High-Level Outline Only

- Divide items into 5 size classes
- Two different packing strategies
- Use advice bit to indicate which one is better
- Yields a $\sqrt{2}$-competitive algorithm



Near-Optimality

Using Constant Advice

Near-Optimality with Constant Advice

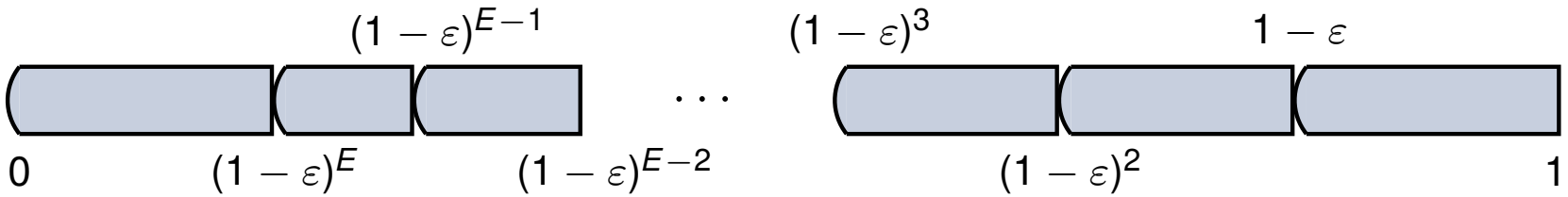
Algorithm Requirements

- Takes parameter $\varepsilon > 0$
- Is $(1 + \varepsilon)$-competitive
- Uses only constant advice

Near-Optimality with Constant Advice

Algorithm Outline

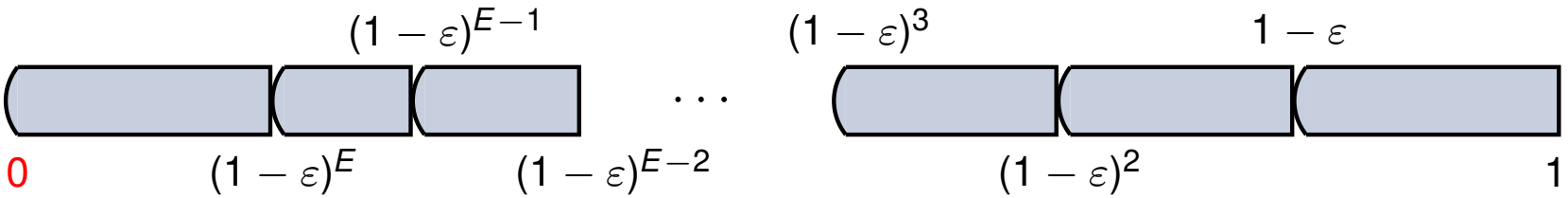
- Given $\varepsilon > 0$
- Let $E = \log_{1-\varepsilon} \varepsilon$
- Divide items into $E + 1$ size classes as shown below



Near-Optimality with Constant Advice

Algorithm Outline

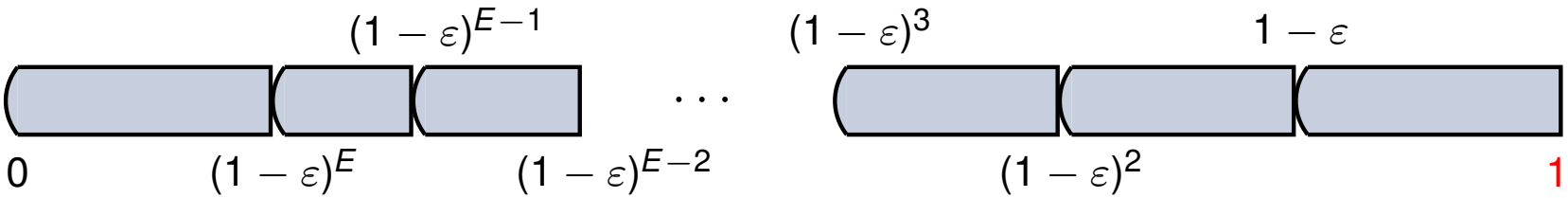
- Given $\varepsilon > 0$
- Let $E = \log_{1-\varepsilon} \varepsilon$
- Divide items into $E + 1$ size classes as shown below



Near-Optimality with Constant Advice

Algorithm Outline

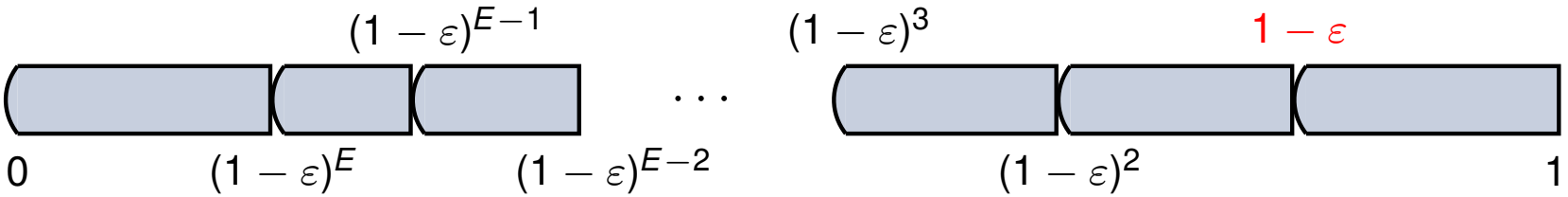
- Given $\varepsilon > 0$
- Let $E = \log_{1-\varepsilon} \varepsilon$
- Divide items into $E + 1$ size classes as shown below



Near-Optimality with Constant Advice

Algorithm Outline

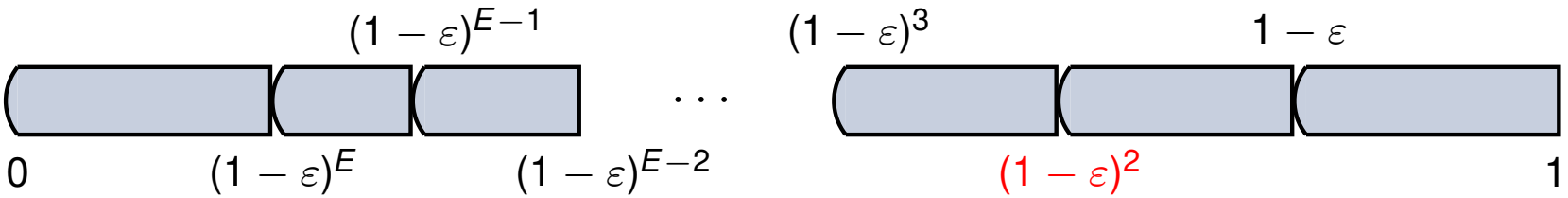
- Given $\varepsilon > 0$
- Let $E = \log_{1-\varepsilon} \varepsilon$
- Divide items into $E + 1$ size classes as shown below



Near-Optimality with Constant Advice

Algorithm Outline

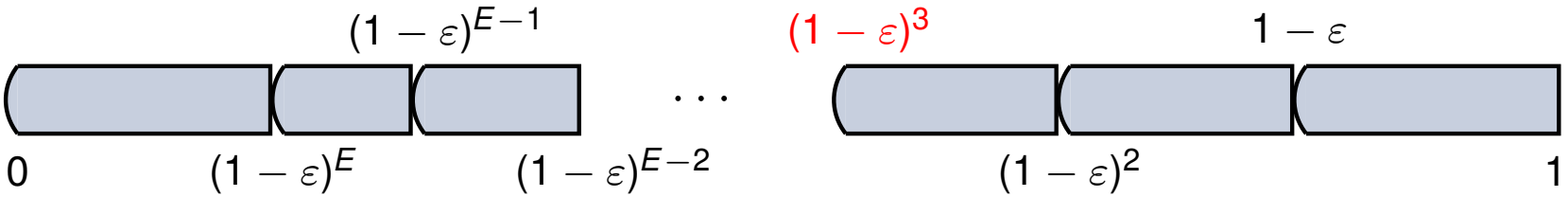
- Given $\varepsilon > 0$
- Let $E = \log_{1-\varepsilon} \varepsilon$
- Divide items into $E + 1$ size classes as shown below



Near-Optimality with Constant Advice

Algorithm Outline

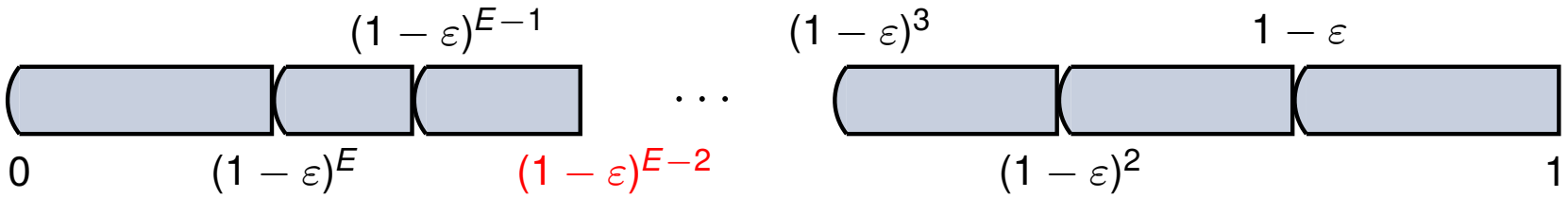
- Given $\varepsilon > 0$
- Let $E = \log_{1-\varepsilon} \varepsilon$
- Divide items into $E + 1$ size classes as shown below



Near-Optimality with Constant Advice

Algorithm Outline

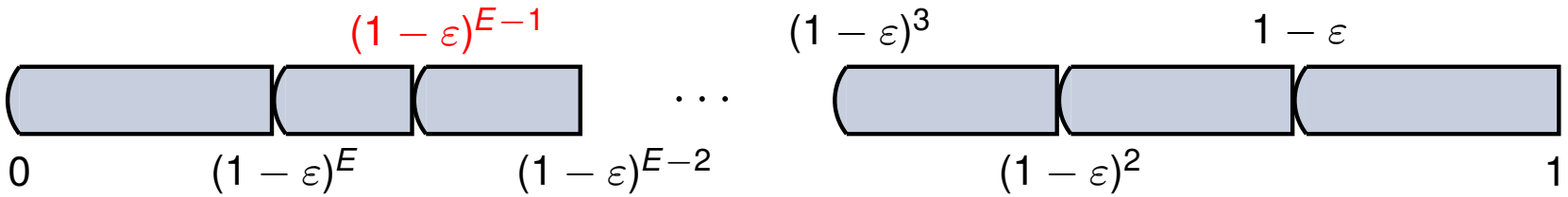
- Given $\varepsilon > 0$
- Let $E = \log_{1-\varepsilon} \varepsilon$
- Divide items into $E + 1$ size classes as shown below



Near-Optimality with Constant Advice

Algorithm Outline

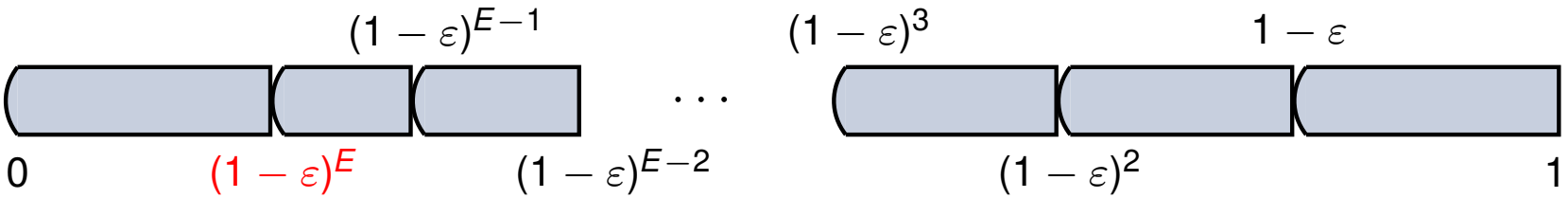
- Given $\varepsilon > 0$
- Let $E = \log_{1-\varepsilon} \varepsilon$
- Divide items into $E + 1$ size classes as shown below



Near-Optimality with Constant Advice

Algorithm Outline

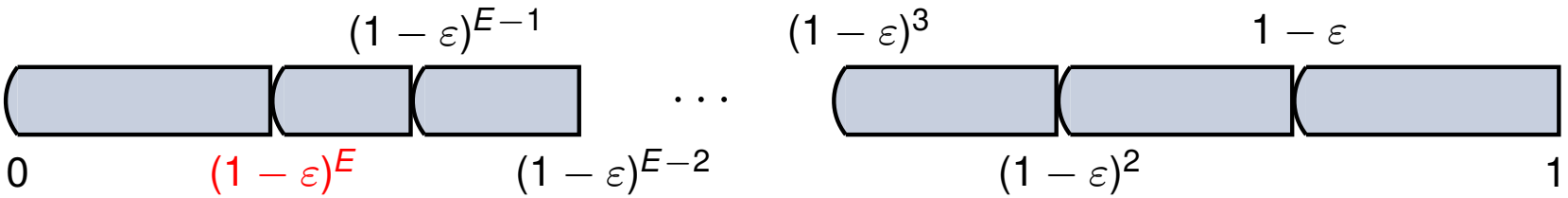
- Given $\varepsilon > 0$
- Let $E = \log_{1-\varepsilon} \varepsilon$
- Divide items into $E + 1$ size classes as shown below



Near-Optimality with Constant Advice

Algorithm Outline

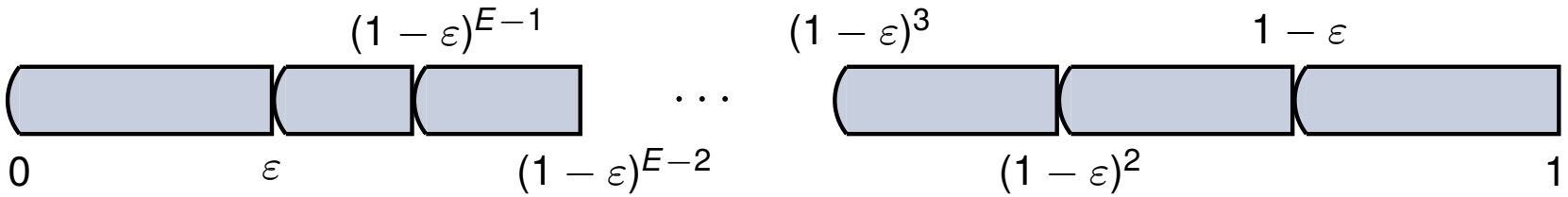
- Given $\varepsilon > 0$
- Let $E = \log_{1-\varepsilon} \varepsilon$
- Divide items into $E + 1$ size classes as shown below



Near-Optimality with Constant Advice

Algorithm Outline

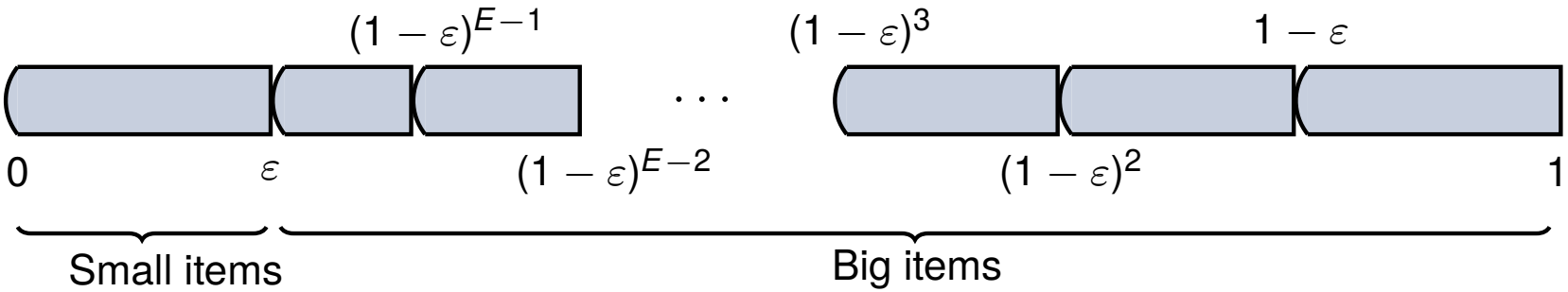
- Given $\varepsilon > 0$
- Let $E = \log_{1-\varepsilon} \varepsilon$
- Divide items into $E + 1$ size classes as shown below



Near-Optimality with Constant Advice

Algorithm Outline

- Given $\varepsilon > 0$
- Let $E = \log_{1-\varepsilon} \varepsilon$
- Divide items into $E + 1$ size classes as shown below



Near-Optimality with Constant Advice

What does the advice encode?

- Oracle fixes arbitrary optimal solution S
- Let $u_1, \ldots, u_k$ denote big items of S in appearance order
- Let c_i denote size class of u_i
- The advice encodes $(c_1, \ldots, c_k)$

Near-Optimality with Constant Advice

Only constant advice

- $2k\lceil \log_2(E+1) \rceil$ bits suffice to encode $(c_1, \ldots, c_k)$
  - There are E classes for big items
  - Thus one class indicated by $\lceil \log_2(E+1) \rceil$ bits
  - A *self-delimiting* encoding: $2\lceil \log_2(E+1) \rceil$ bits

Near-Optimality with Constant Advice

Only constant advice

- $2k\lceil \log_2(E+1) \rceil$ bits suffice to encode $(c_1, \ldots, c_k)$
  - There are E classes for big items
  - Thus one class indicated by $\lceil \log_2(E+1) \rceil$ bits
  - A *self-delimiting* encoding: $2\lceil \log_2(E+1) \rceil$ bits
- This is constant advice:
  - The number k of big elements in S is bounded by $1/\varepsilon$
  - Recall that $E = \log_{1-\varepsilon}(\varepsilon)$

Near-Optimality with Constant Advice

Algorithm Procedure

- Big items: Packed into k virtual *slots*
  - Slot i accommodates items from class c_i exclusively
  - In the beginning, the slots are empty
  - Each slot is filled with exactly one big item
  - The slots are filled strictly in their order
  - Items in filled slots replaced by smaller ones if possible

Near-Optimality with Constant Advice

Algorithm Procedure

- Big items: Packed into k virtual *slots*
  - Slot i accommodates items from class c_i exclusively
  - In the beginning, the slots are empty
  - Each slot is filled with exactly one big item
  - The slots are filled strictly in their order
  - Items in filled slots replaced by smaller ones if possible
- Small items: Packed greedily
  - Removed one by one whenever needed to pack a big one

Example Execution of the Algorithm

Execution Example

Optimal Solution: 

Execution Example

Execution Example

Optimal Solution:

Execution Example

Optimal Solution:

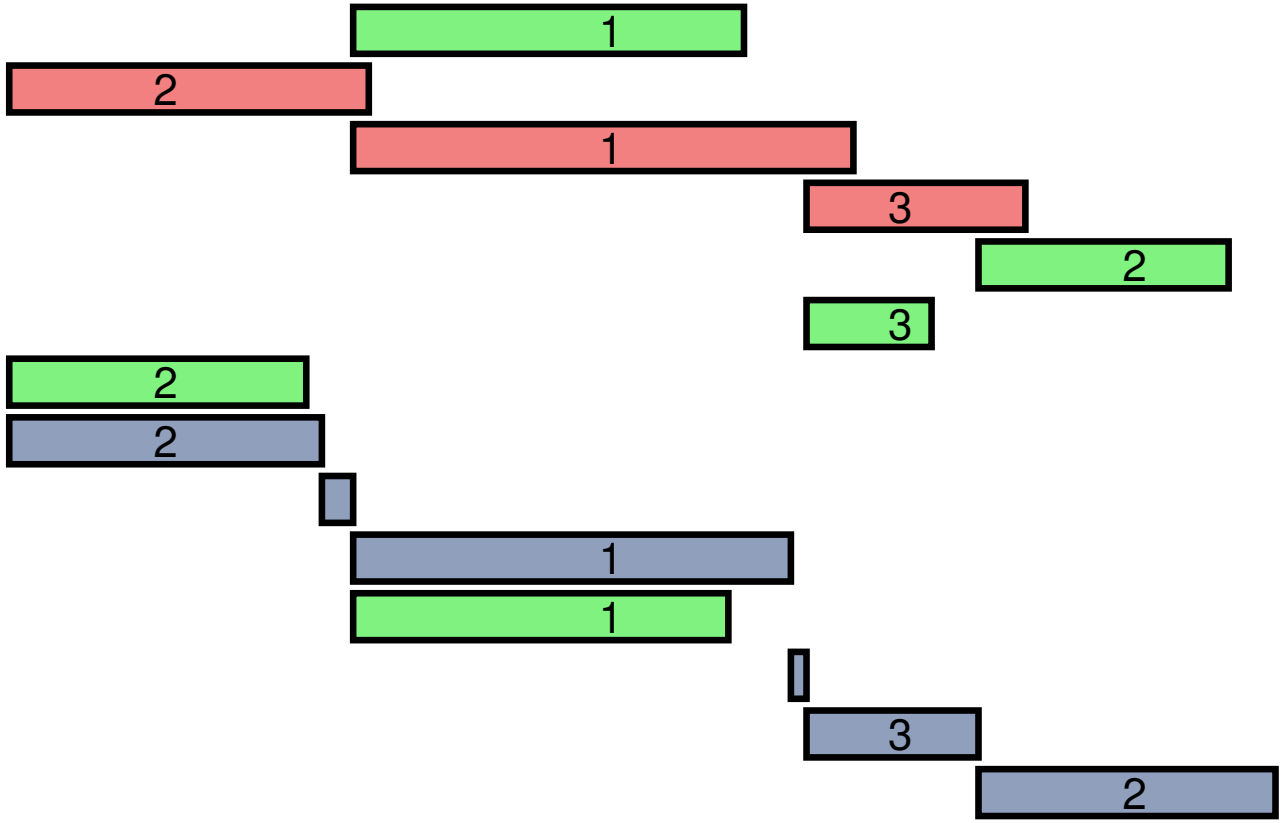
Advice: $(c_1, c_2, c_3, c_4) = (2, 1, 3, 2)$

Execution Example

Optimal Solution:

Advice: $(c_1, c_2, c_3, c_4) = (2, 1, 3, 2)$

Input:



Execution Example

Optimal Solution:

Advice: $(c_1, c_2, c_3, c_4) = (2, 1, 3, 2)$

Input:

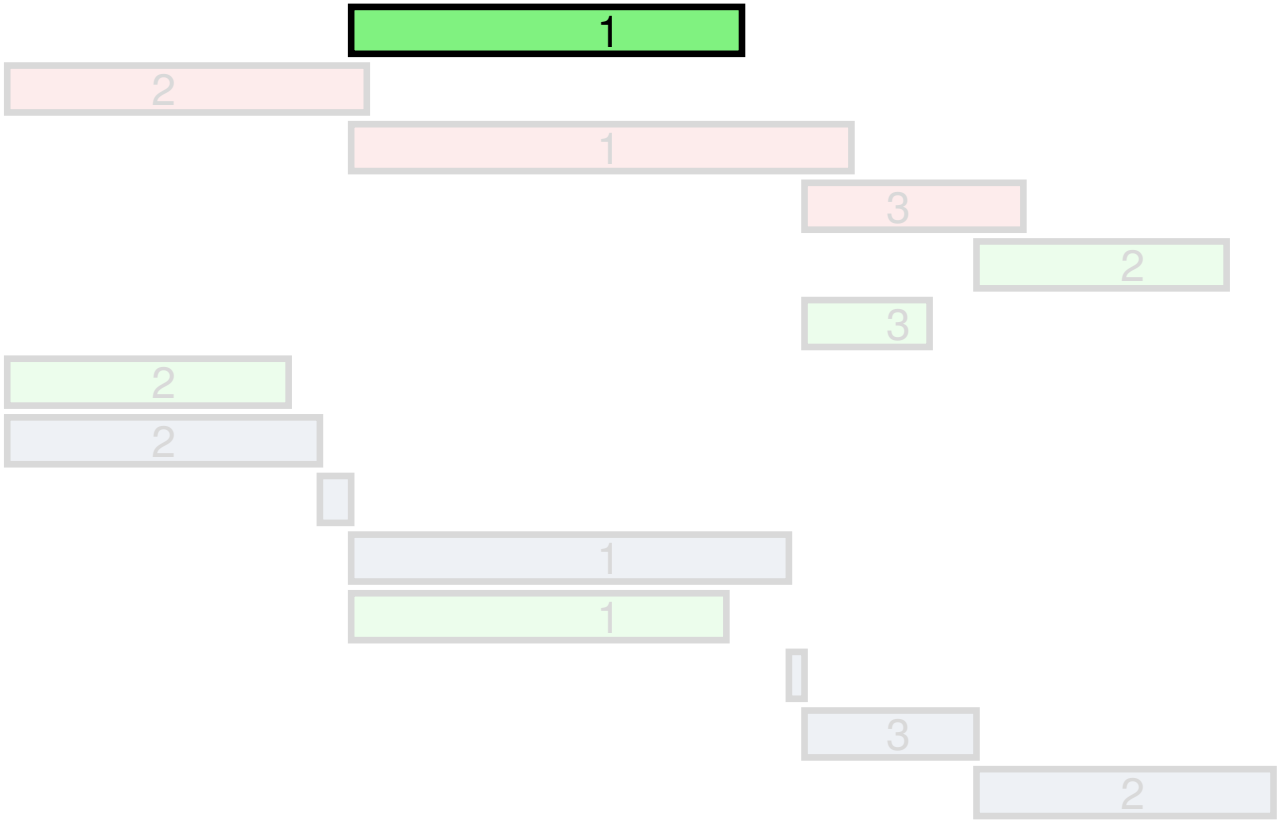
Output:

Execution Example

Optimal Solution:

Advice: $(c_1, c_2, c_3, c_4) = (2, 1, 3, 2)$

Input:



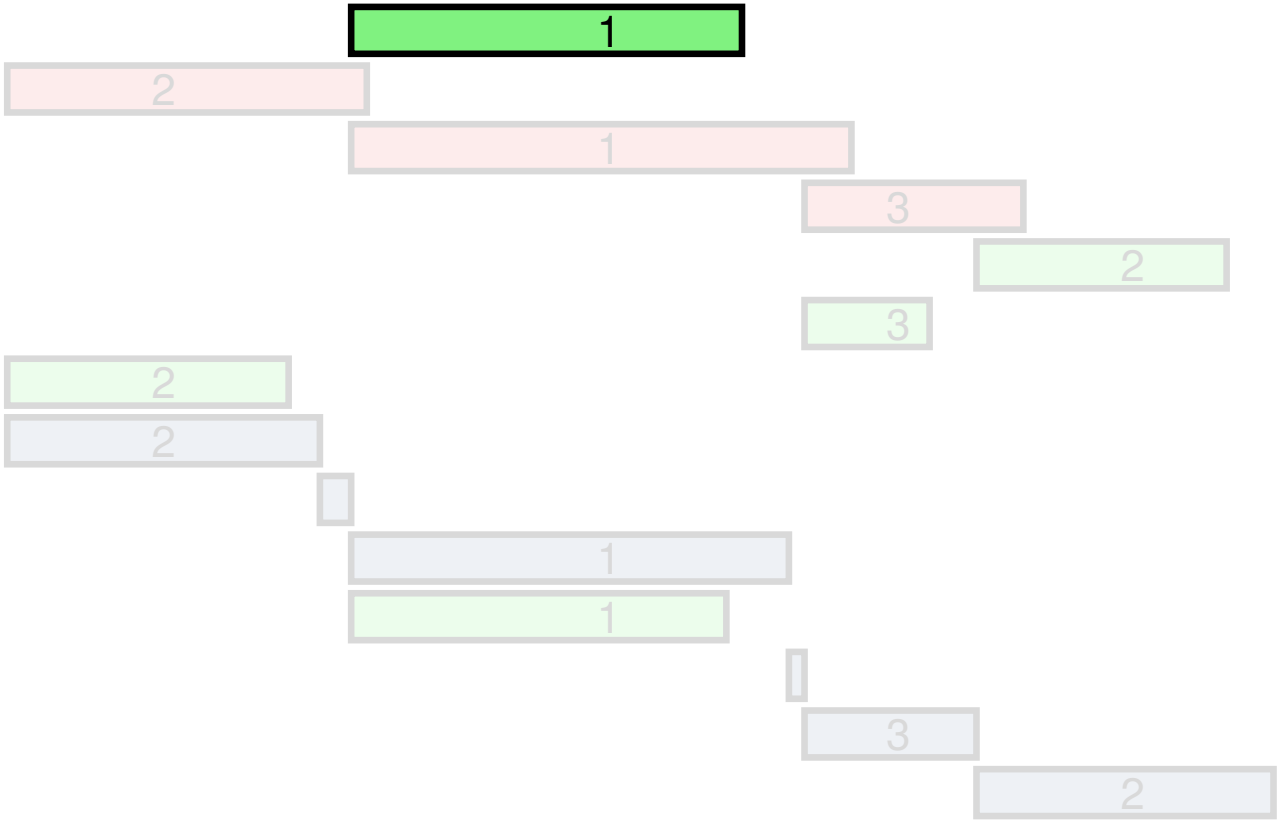
Output:

Execution Example

Optimal Solution:

Advice: $(c_1, c_2, c_3, c_4) = (2, 1, 3, 2)$

Input:



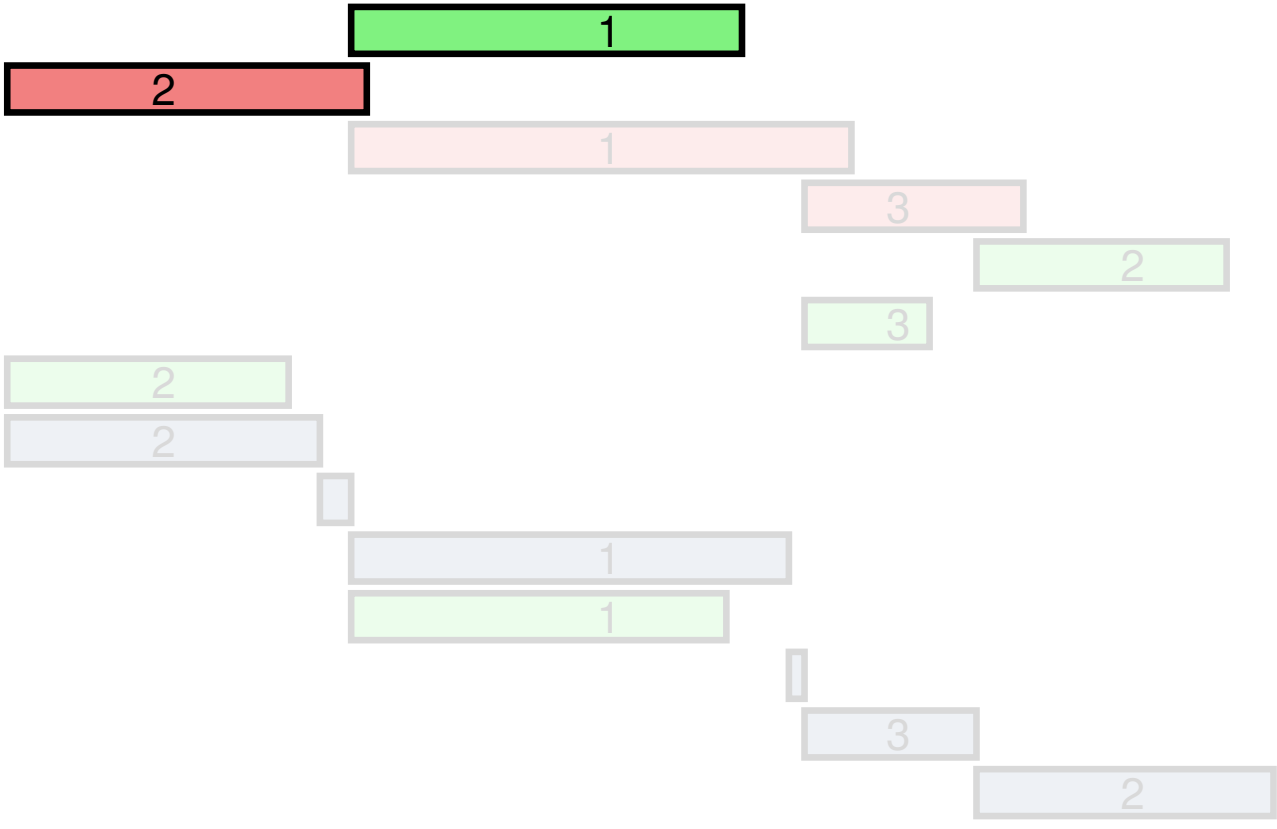
Output:

Execution Example

Optimal Solution:

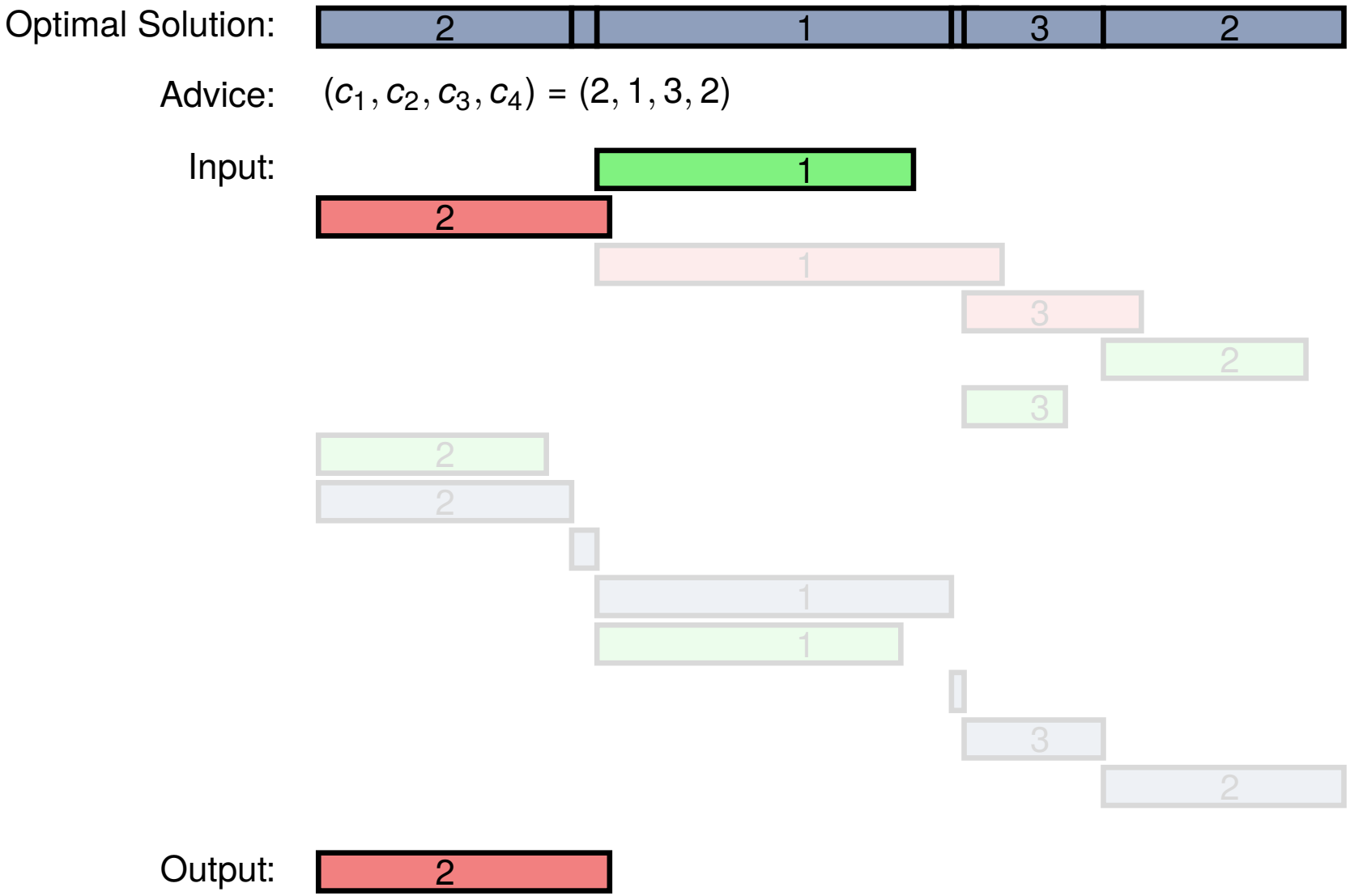
Advice: $(c_1, c_2, c_3, c_4) = (2, 1, 3, 2)$

Input:



Output:

Execution Example

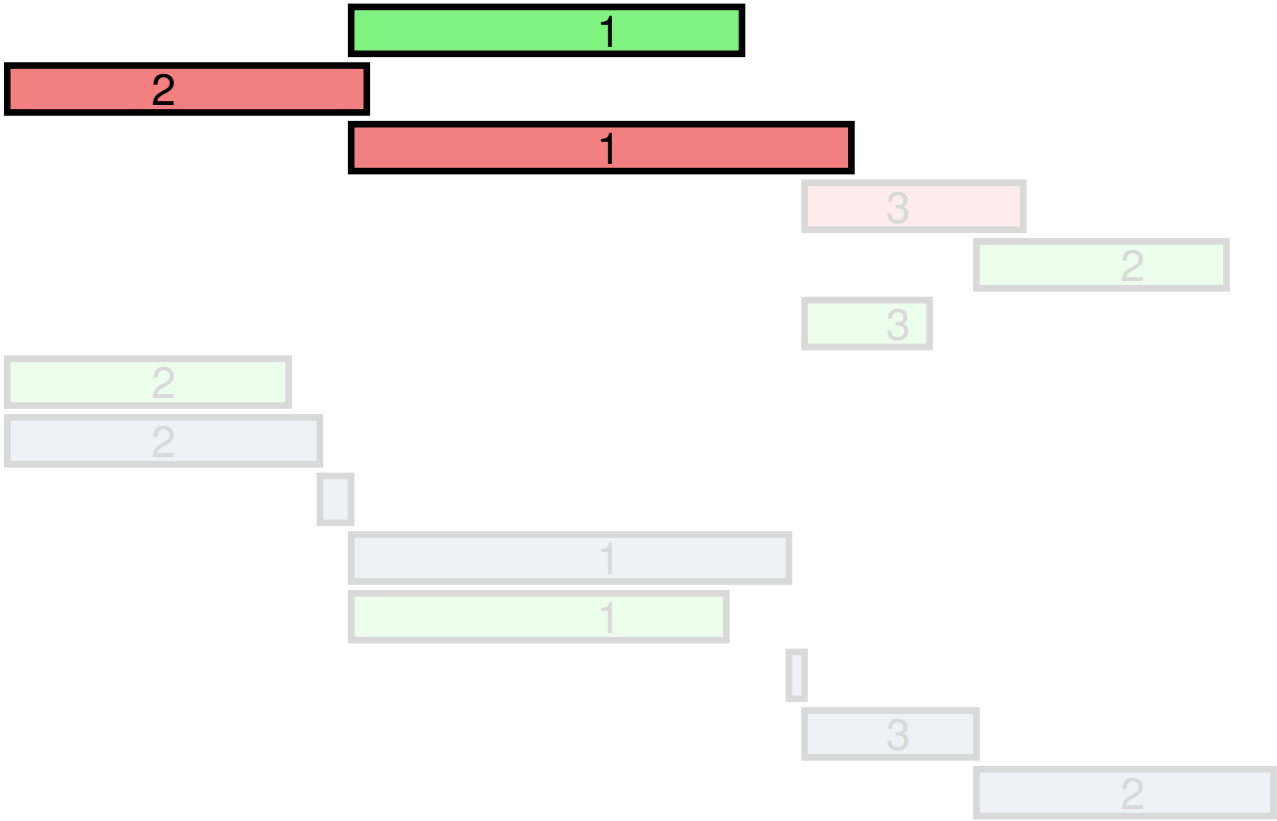


Execution Example

Optimal Solution:

Advice: $(c_1, c_2, c_3, c_4) = (2, 1, 3, 2)$

Input:



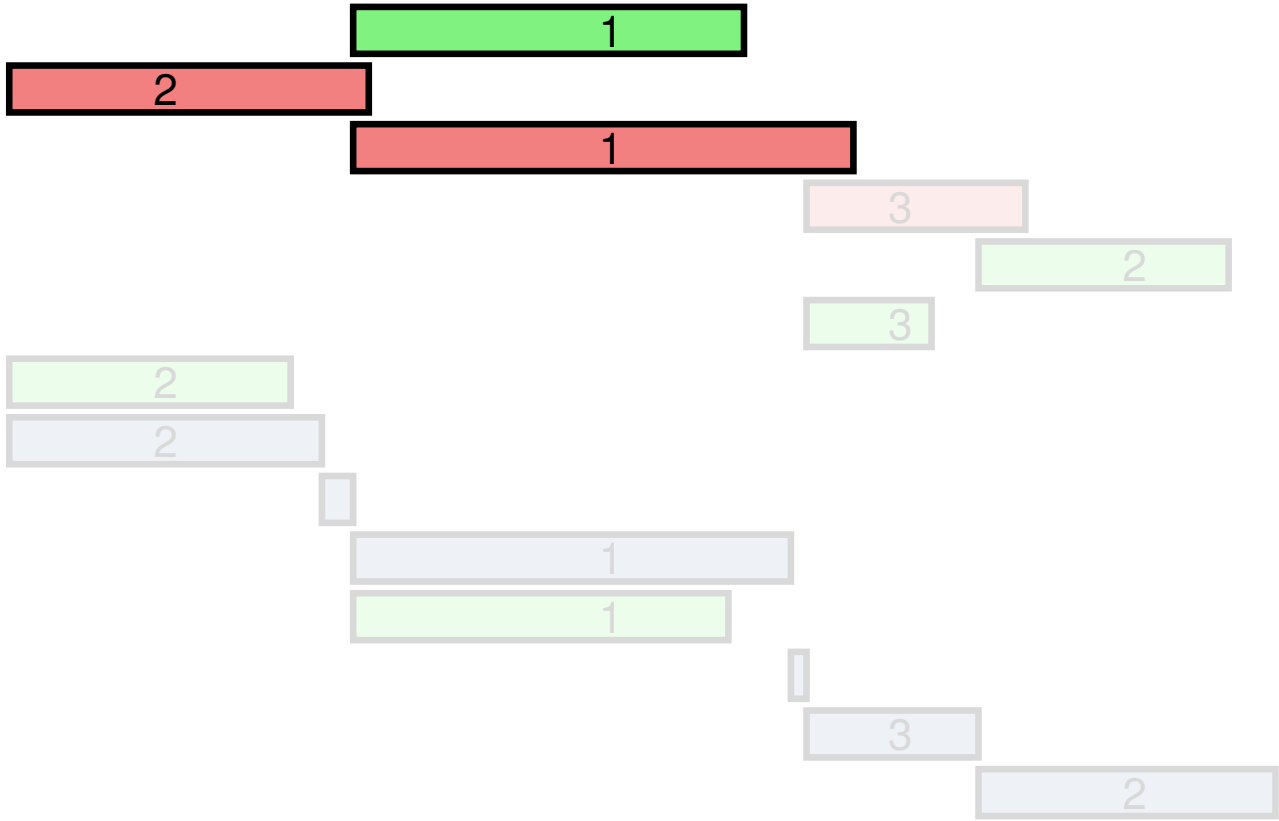
Output:

Execution Example

Optimal Solution:

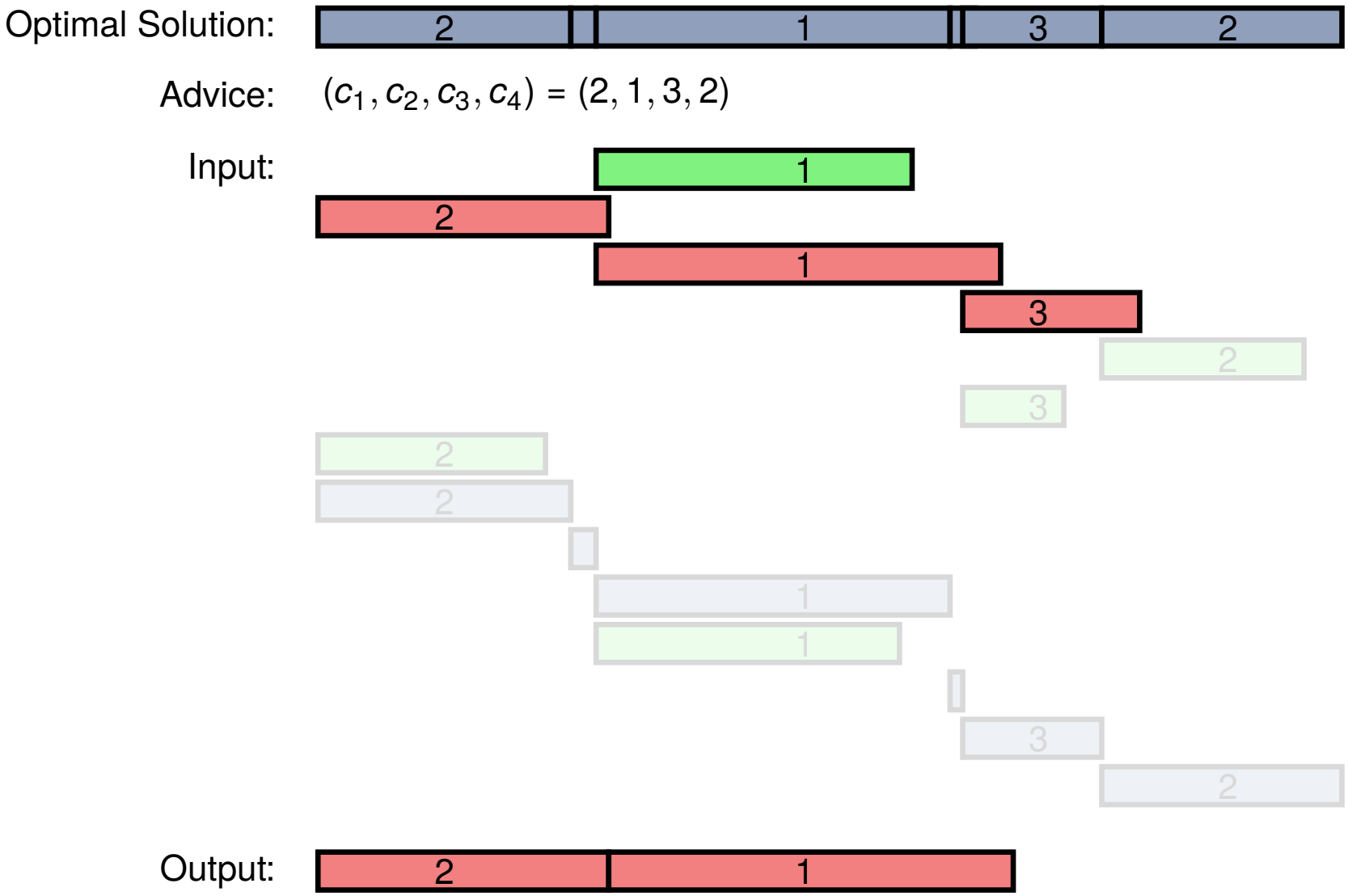
Advice: $(c_1, c_2, c_3, c_4) = (2, 1, 3, 2)$

Input:

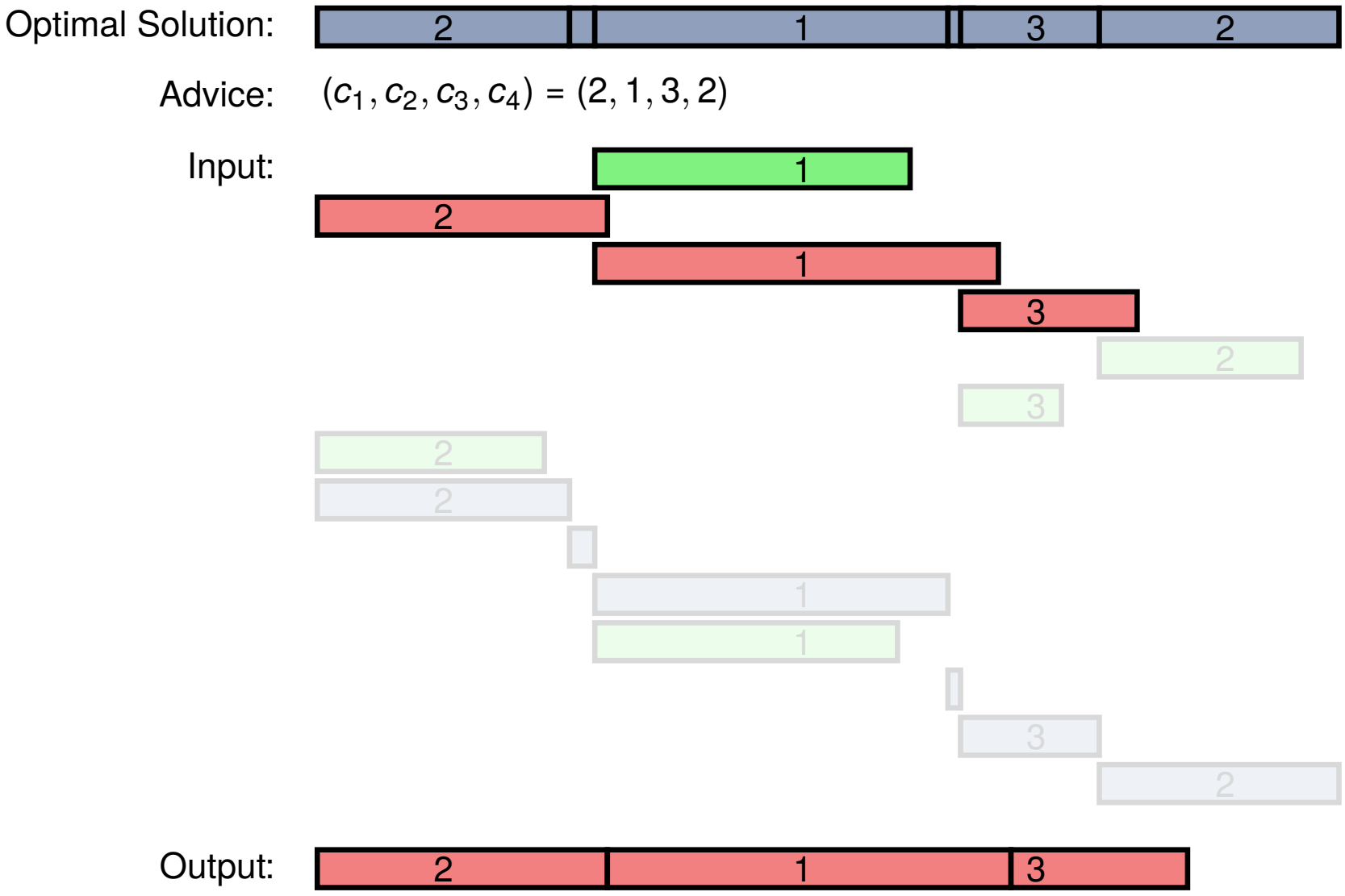


Output:

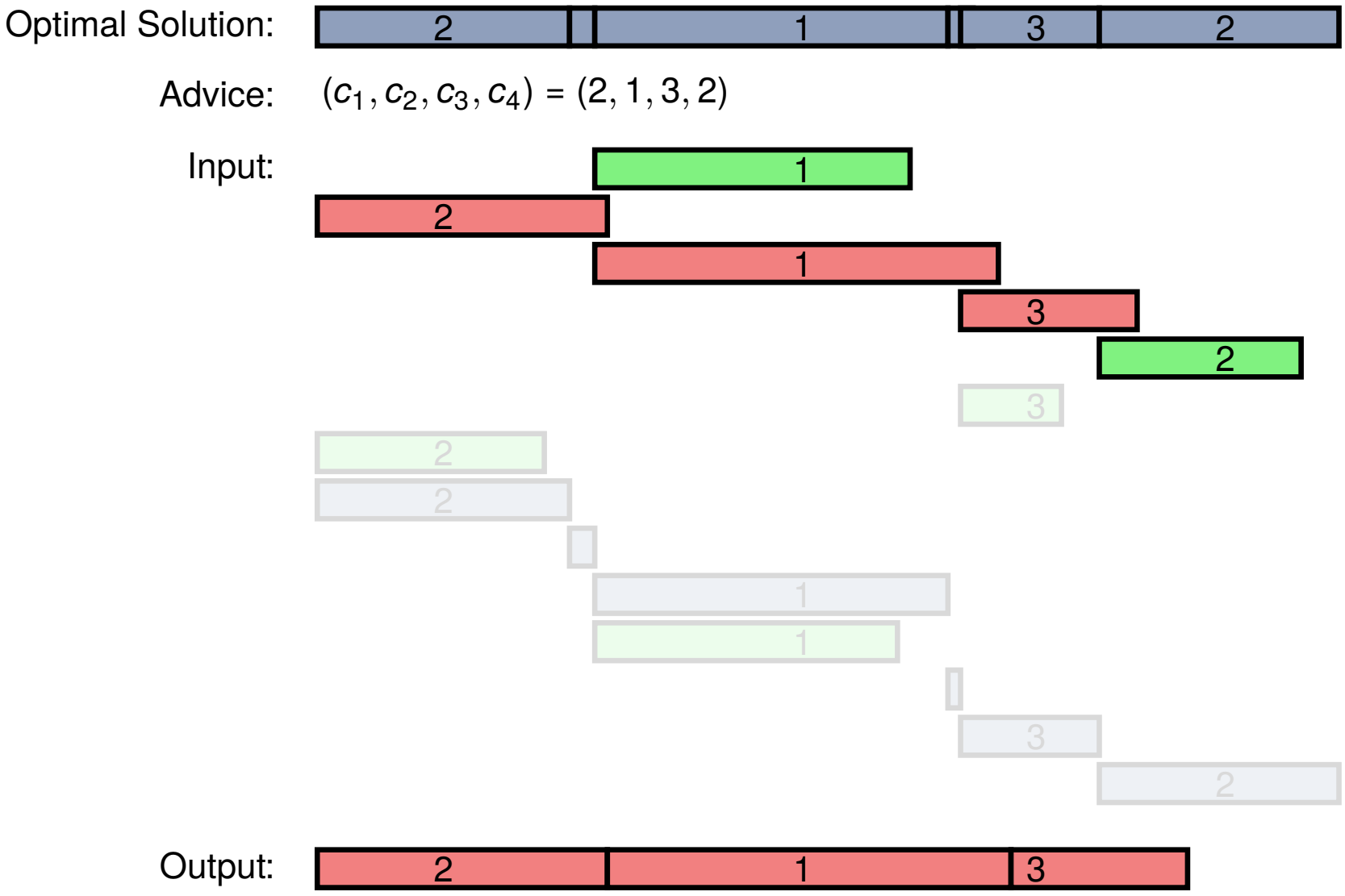
Execution Example



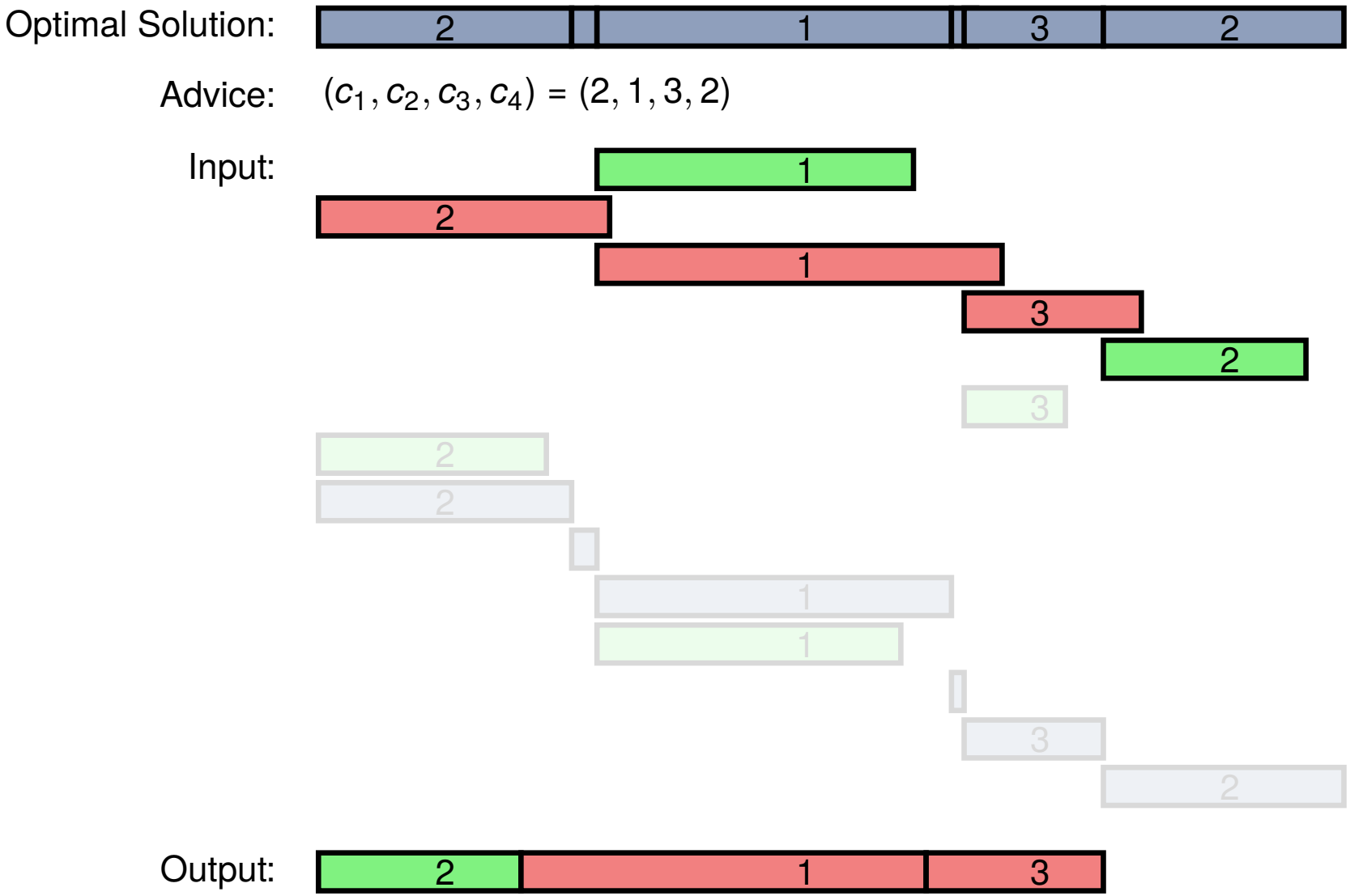
Execution Example



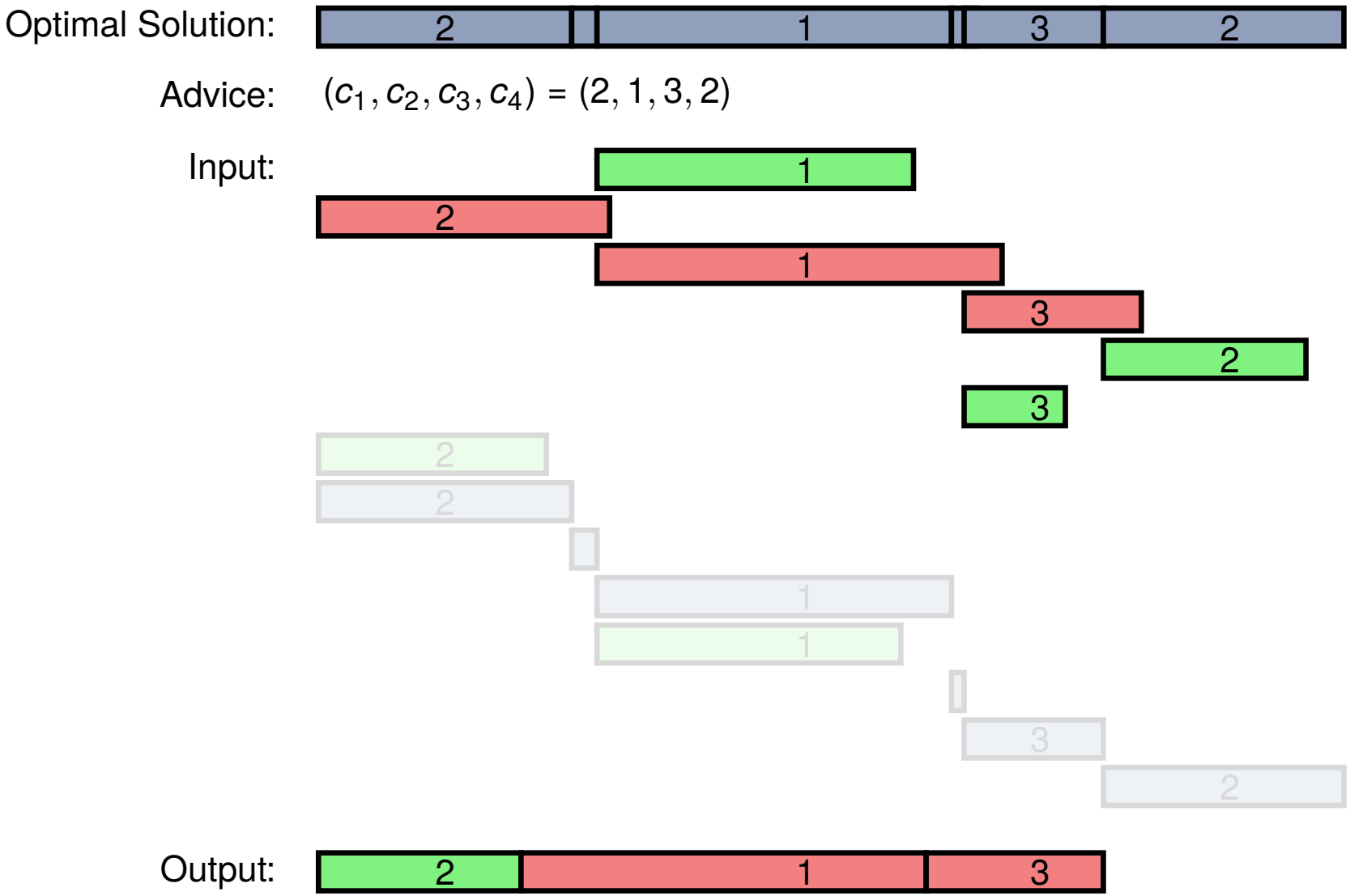
Execution Example



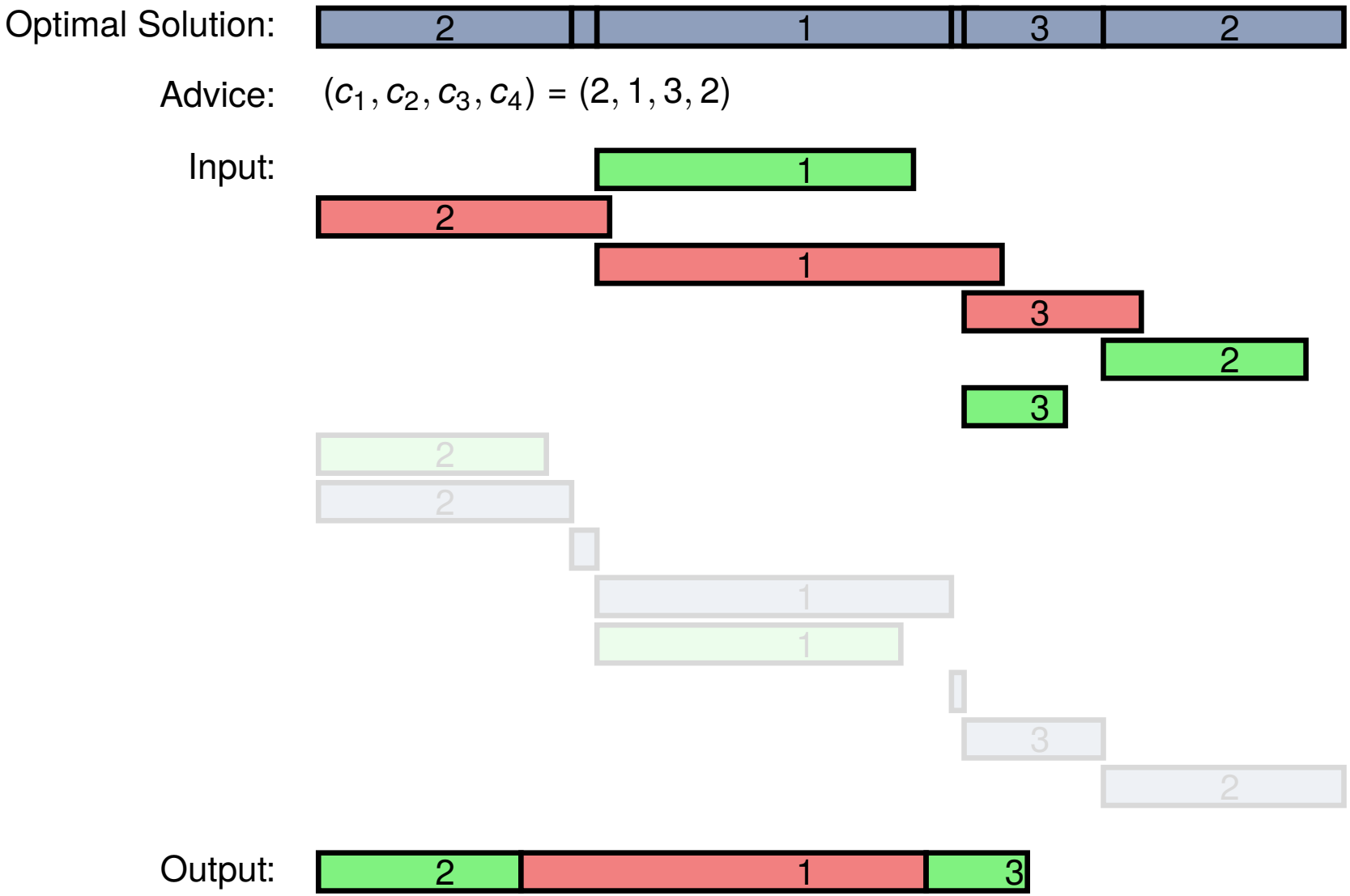
Execution Example



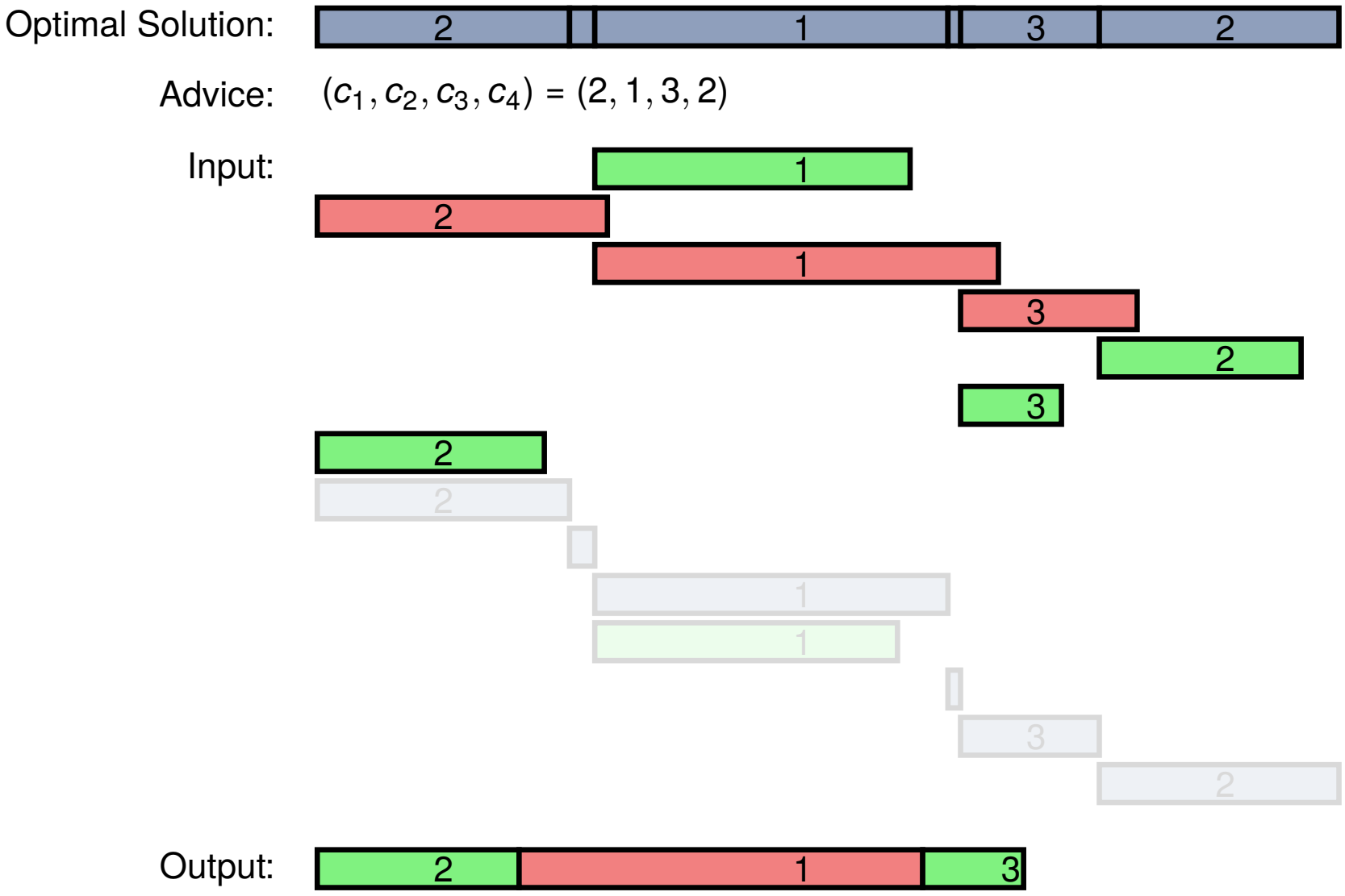
Execution Example



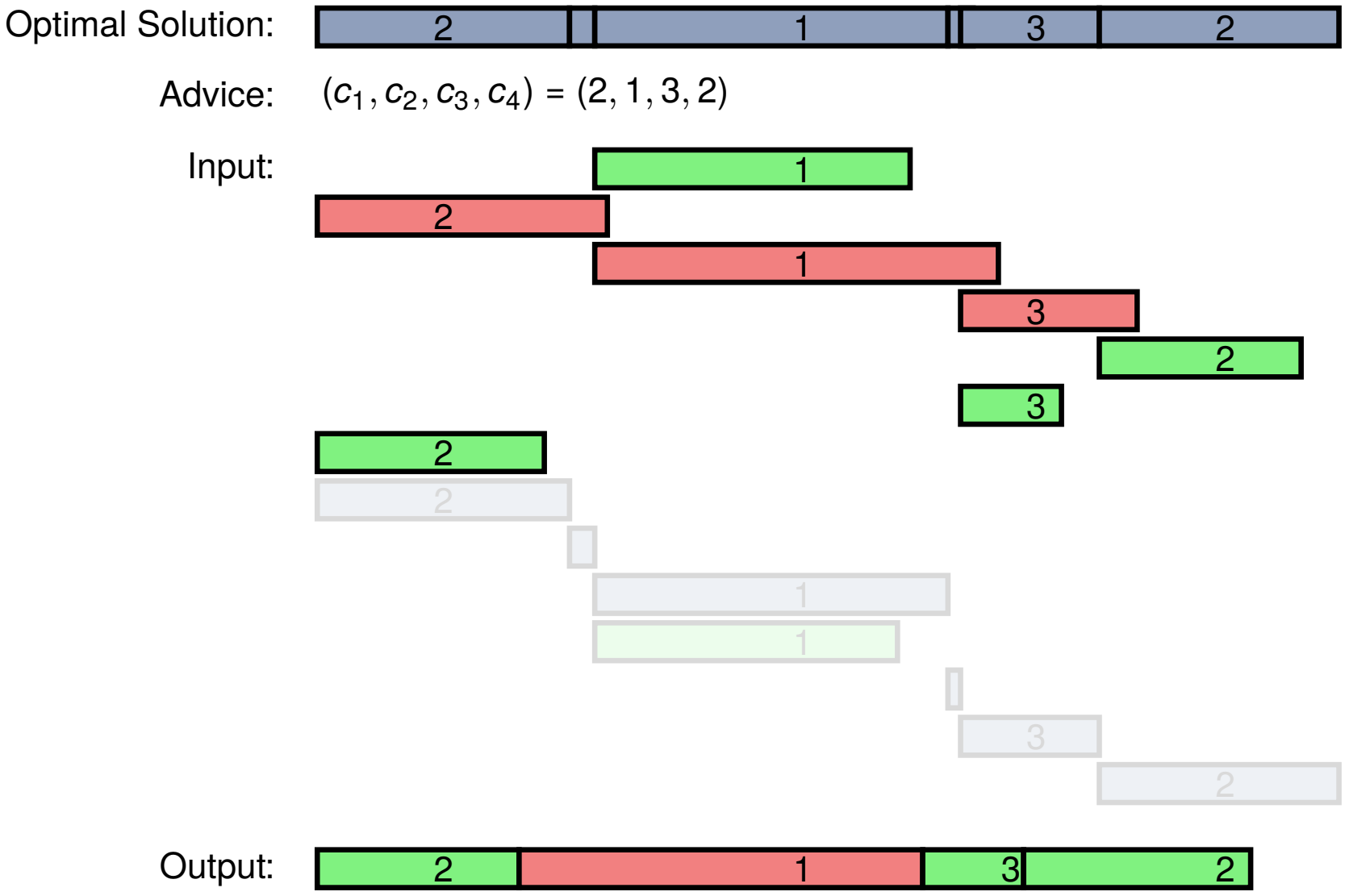
Execution Example



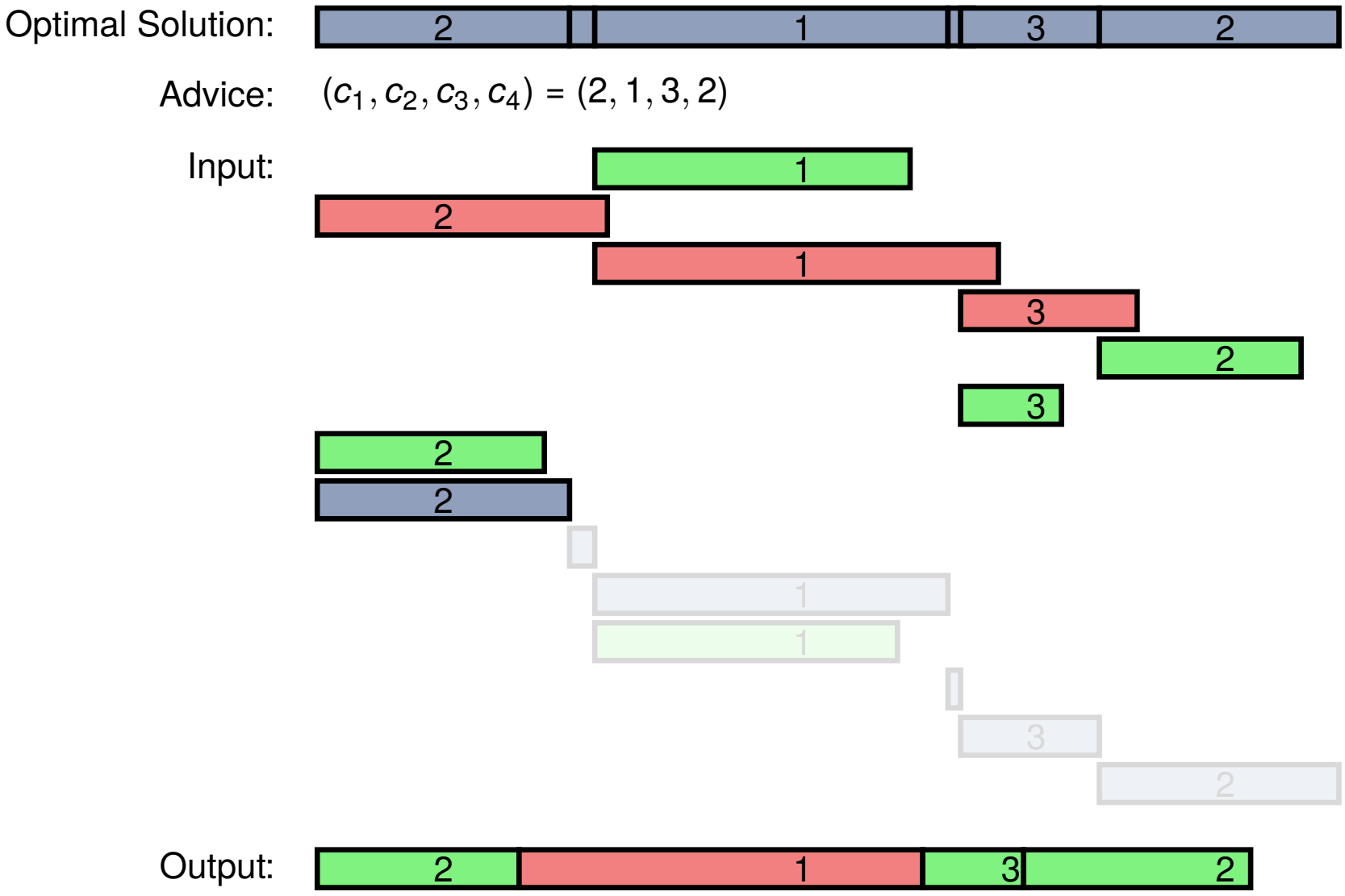
Execution Example



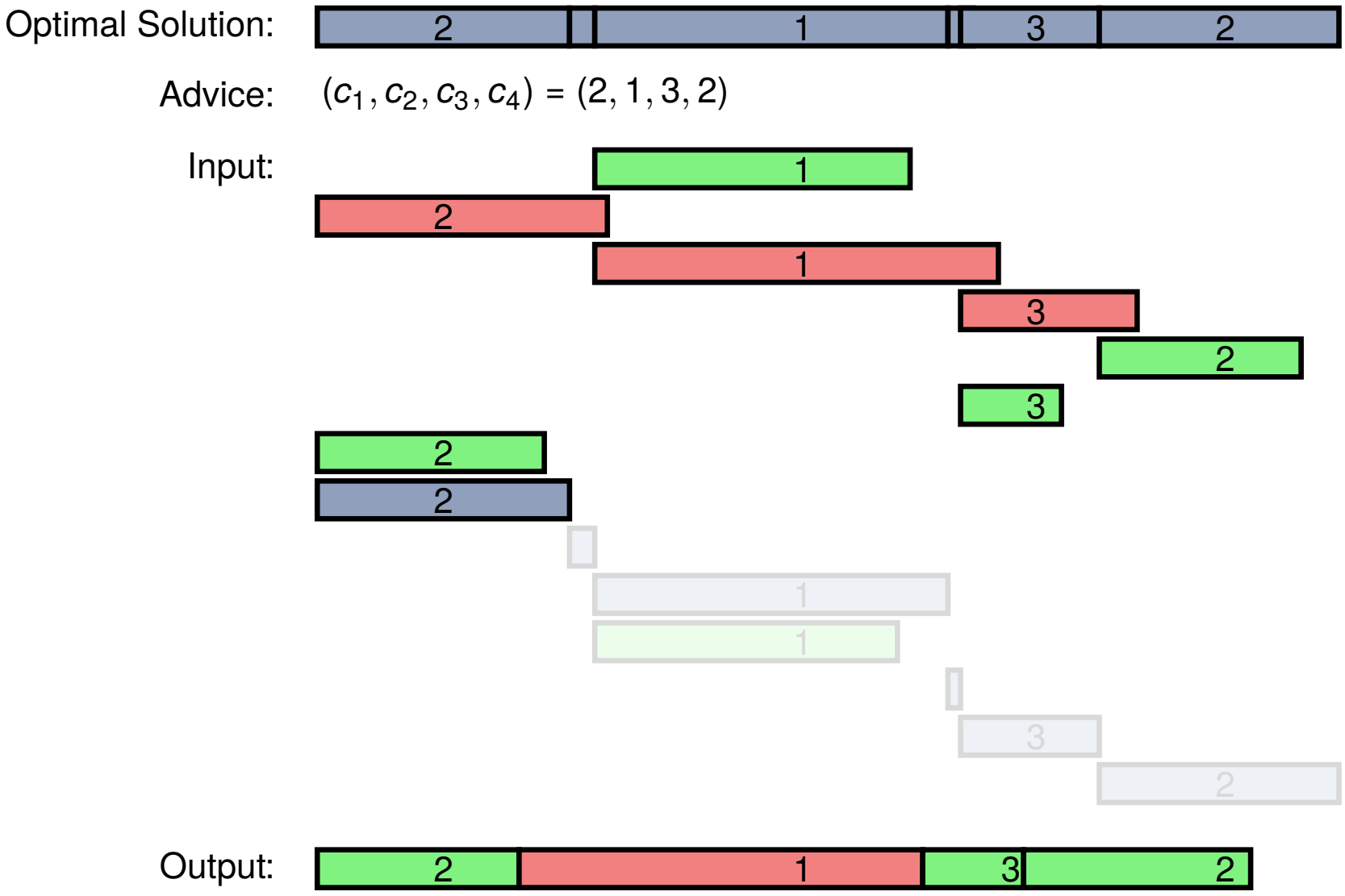
Execution Example



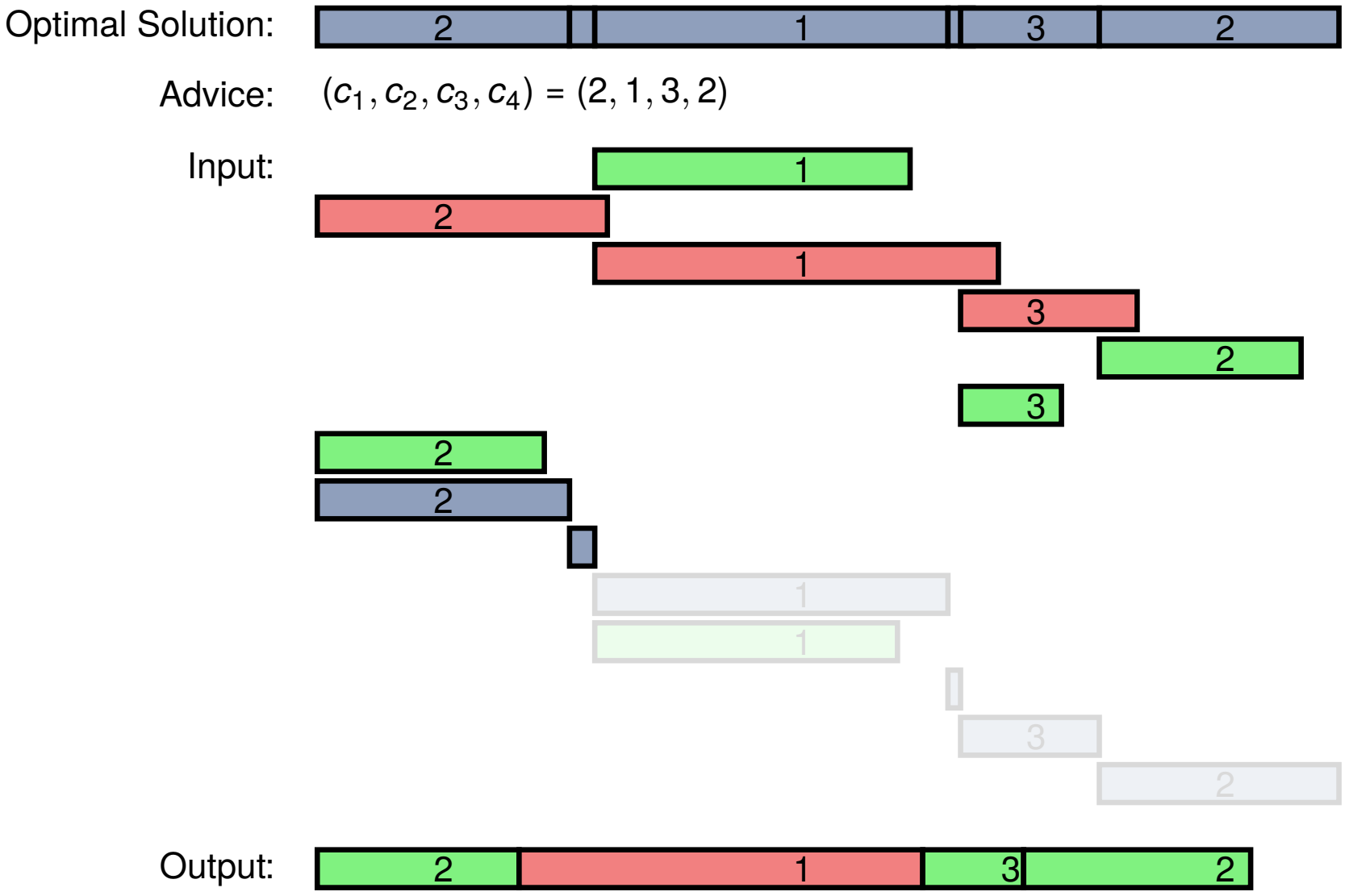
Execution Example



Execution Example



Execution Example

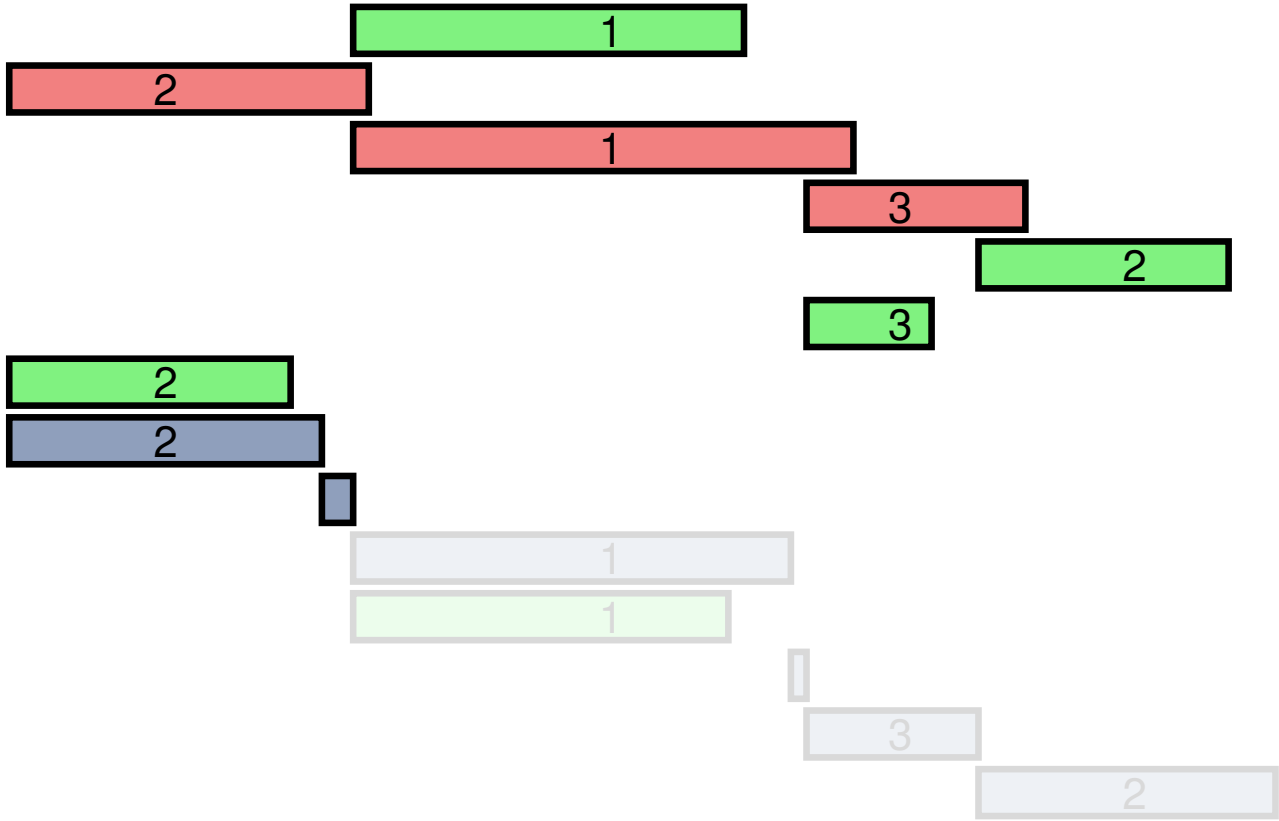


Execution Example

Optimal Solution:

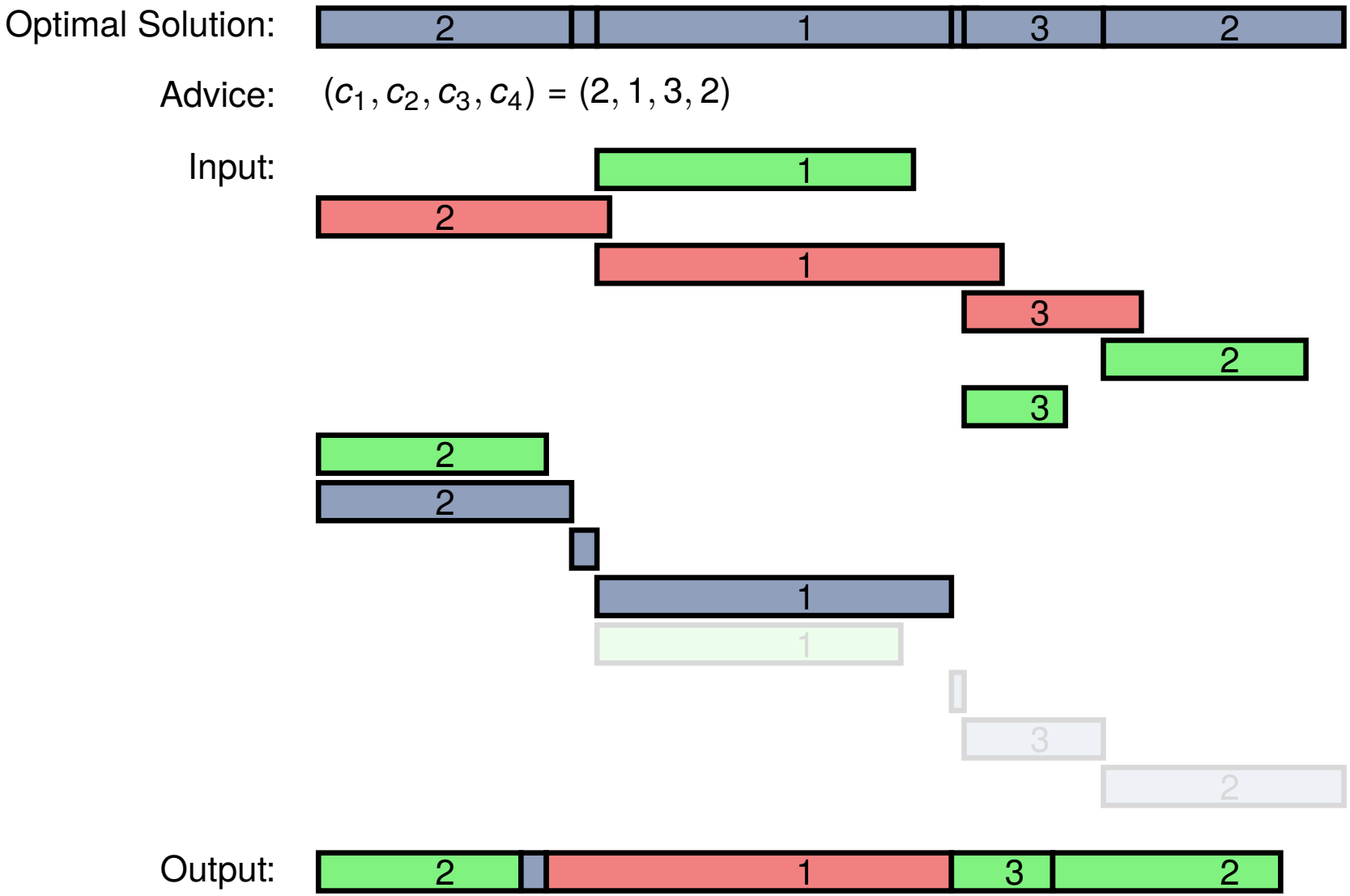
Advice: $(c_1, c_2, c_3, c_4) = (2, 1, 3, 2)$

Input:

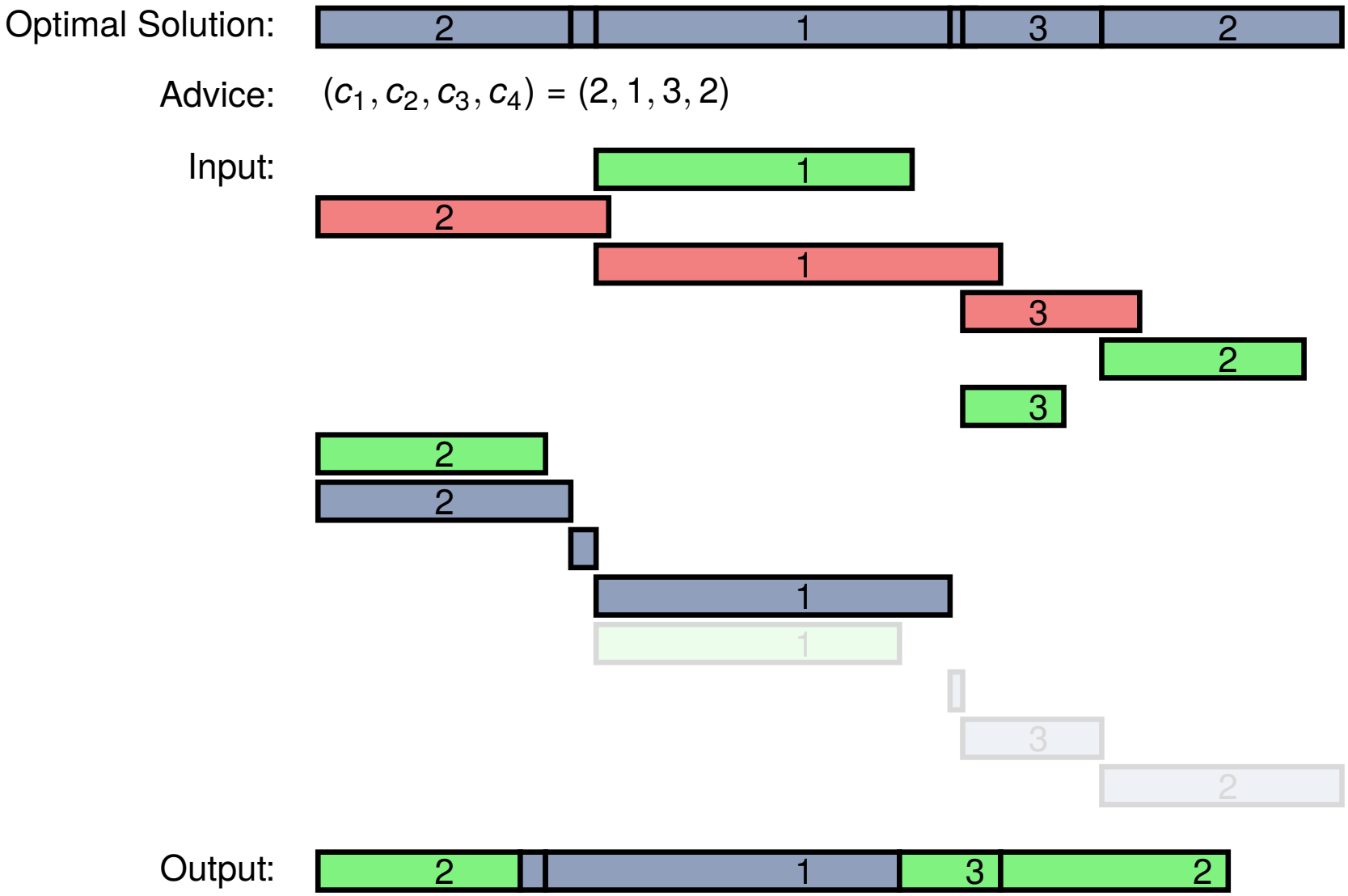


Output:

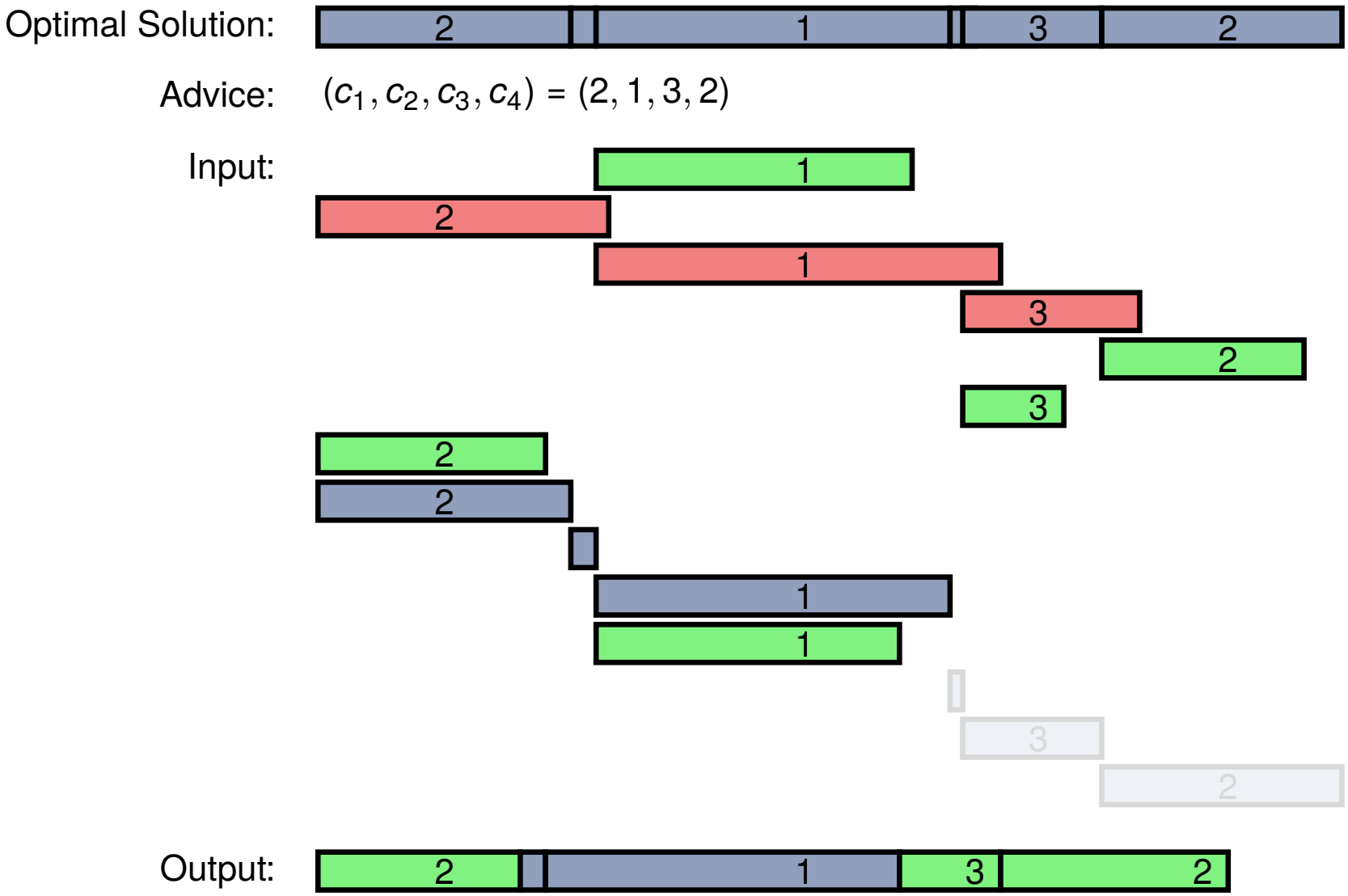
Execution Example



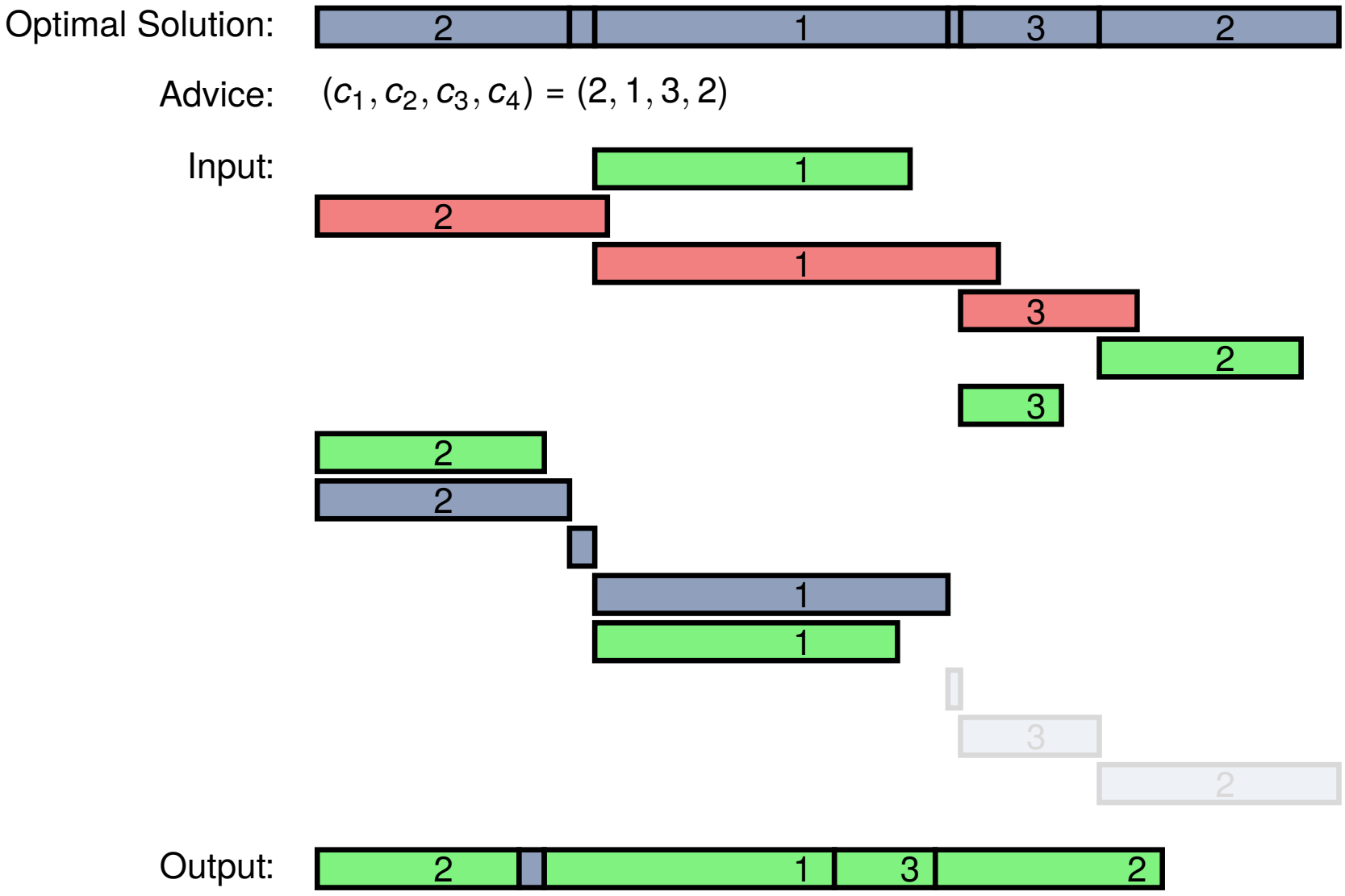
Execution Example



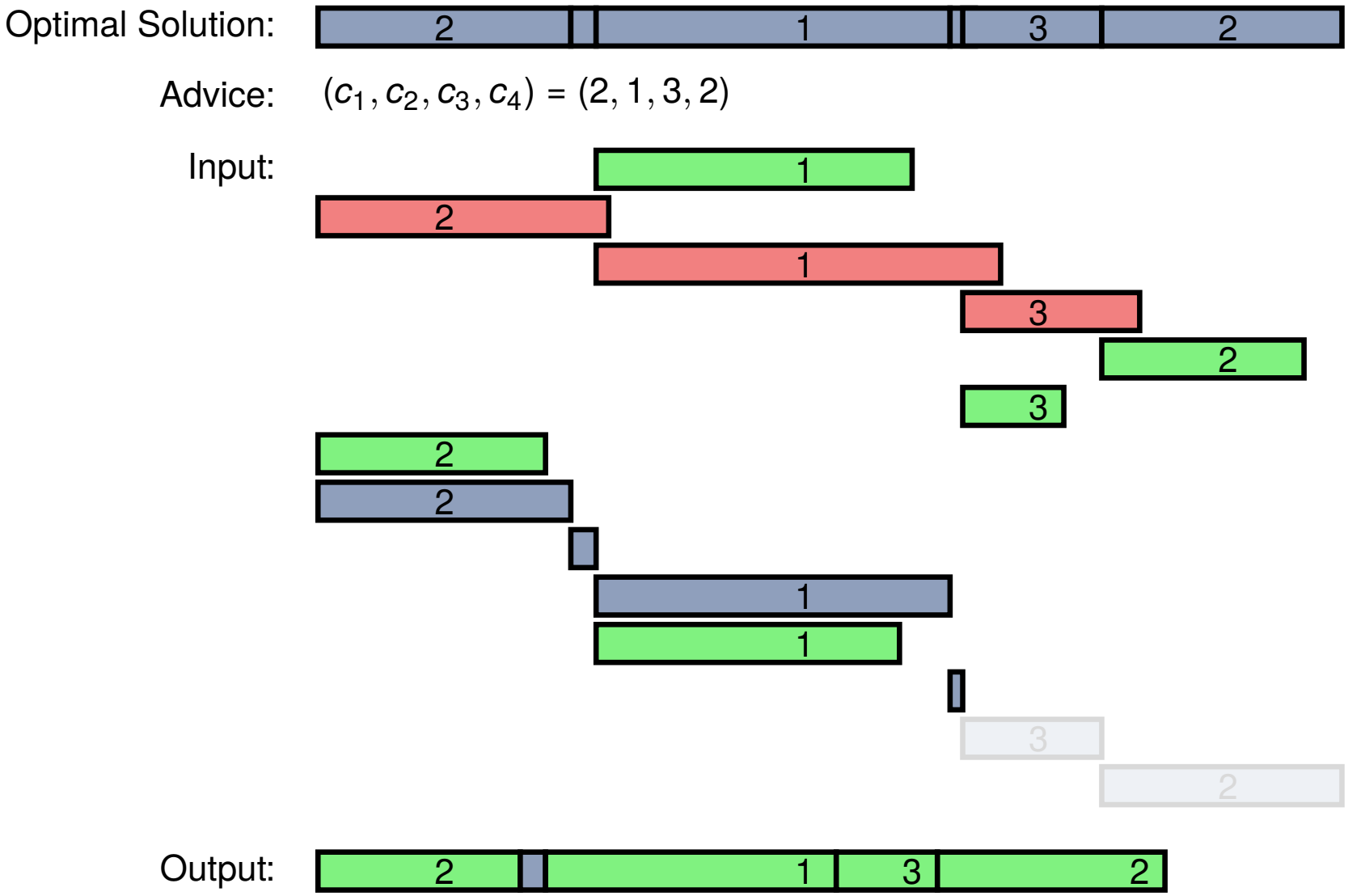
Execution Example



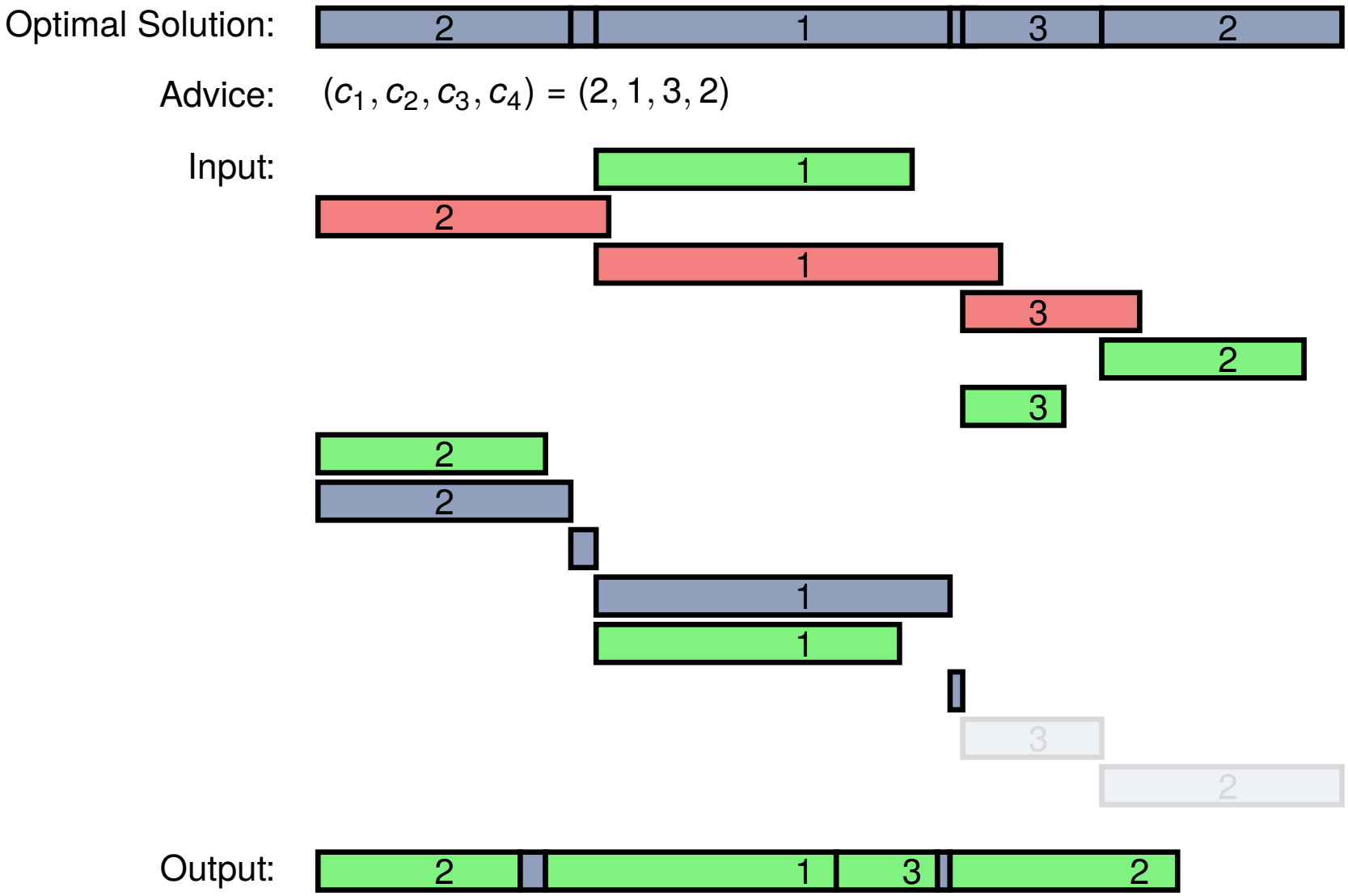
Execution Example



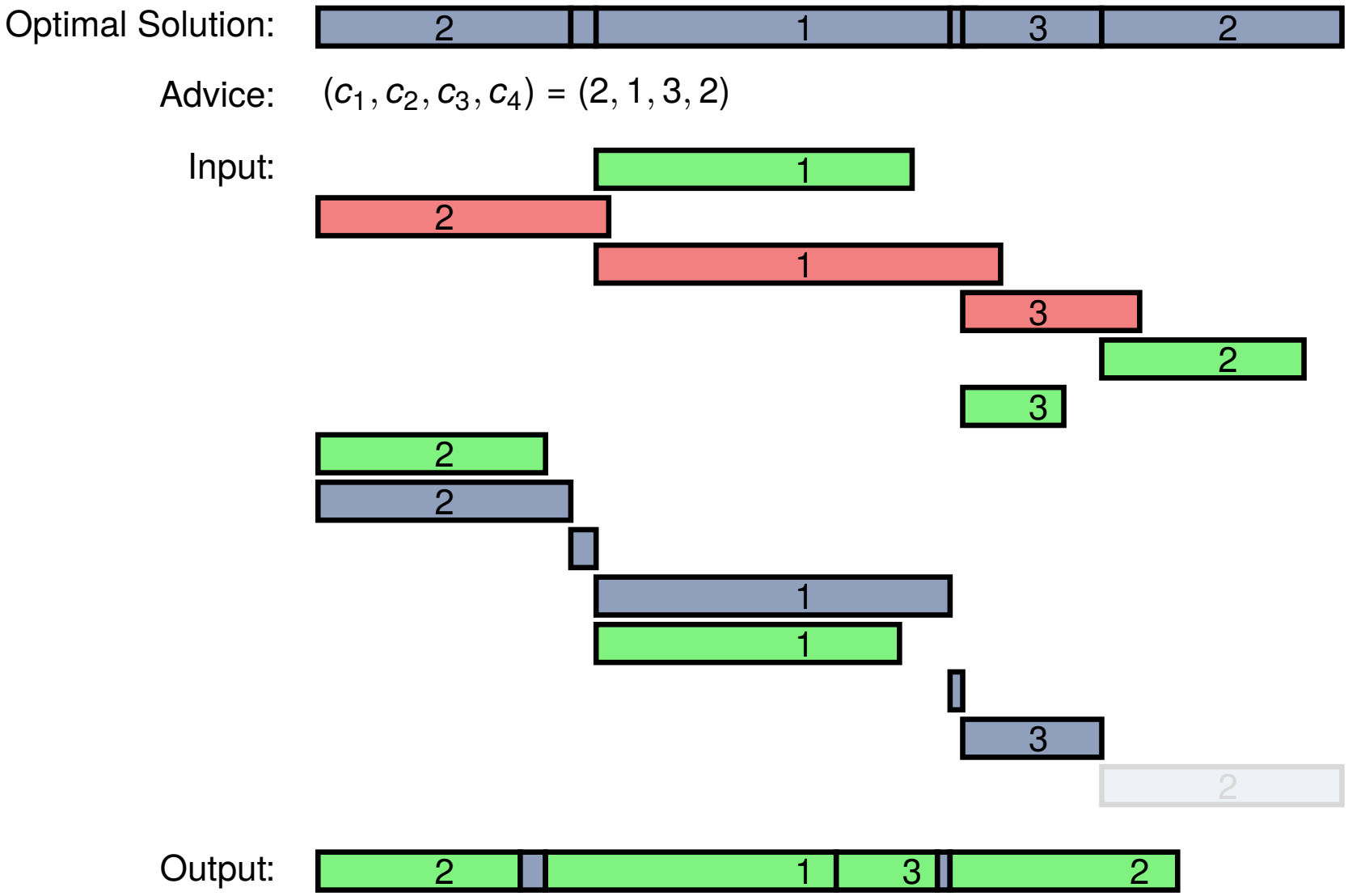
Execution Example



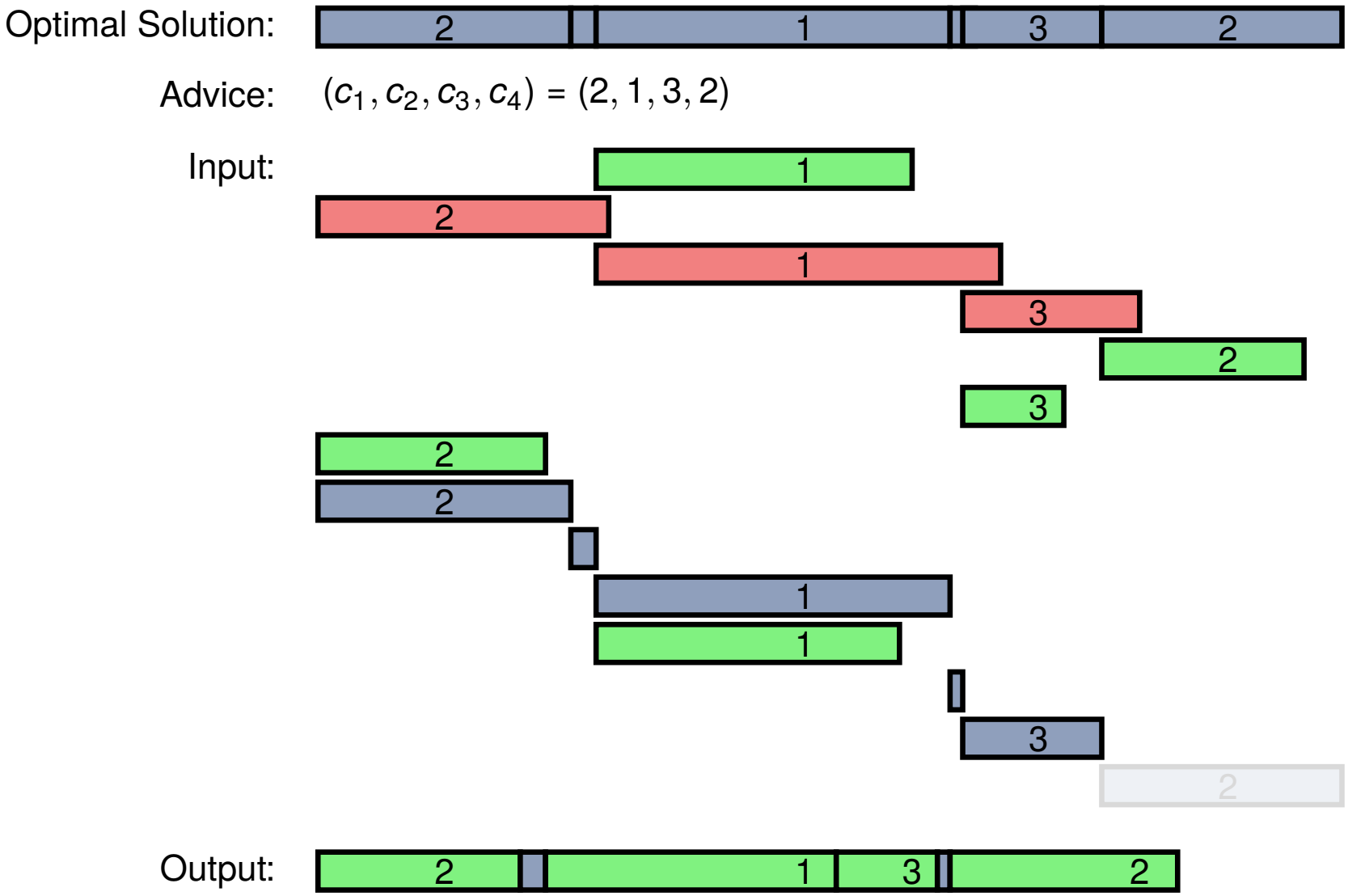
Execution Example



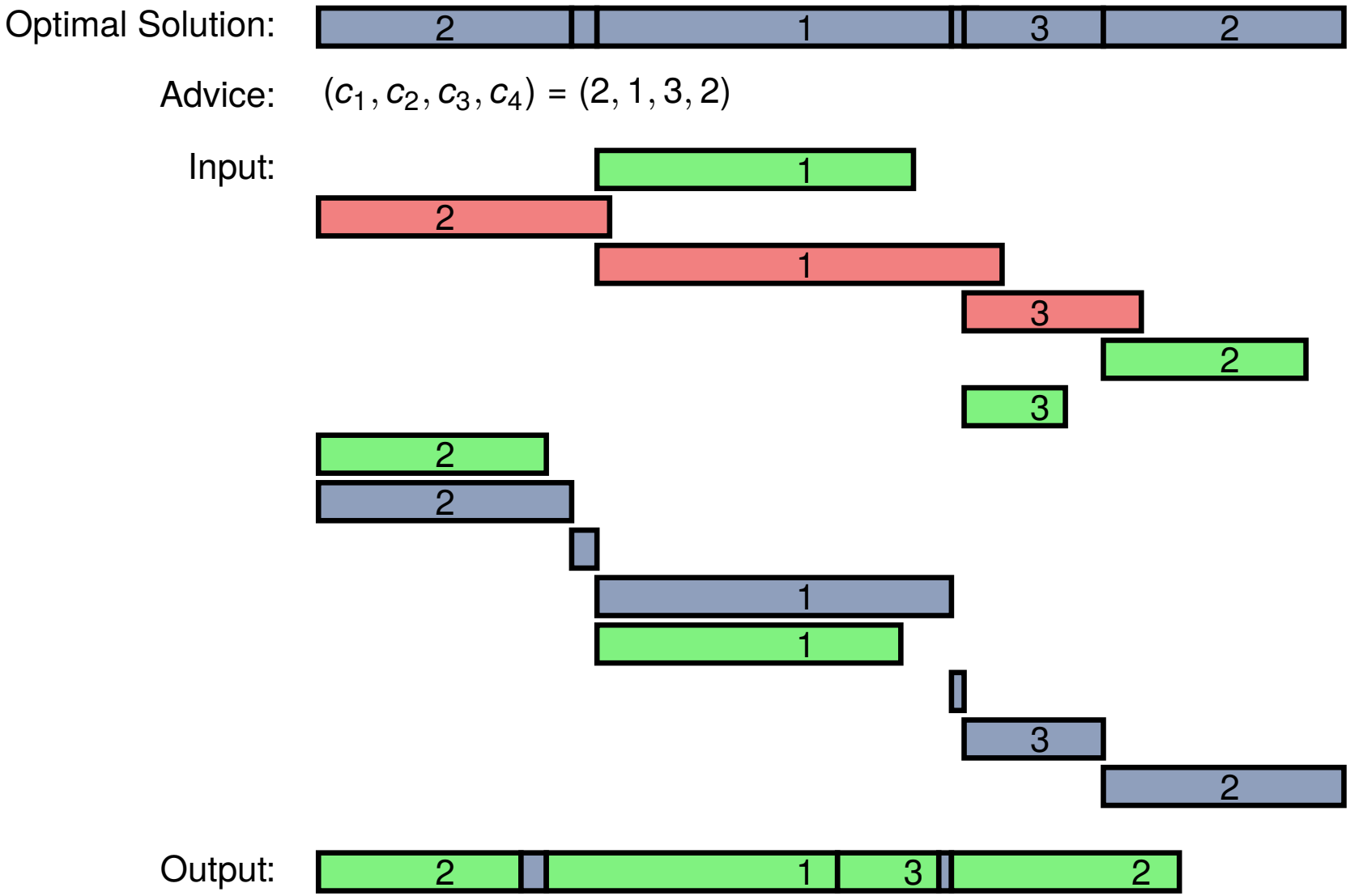
Execution Example



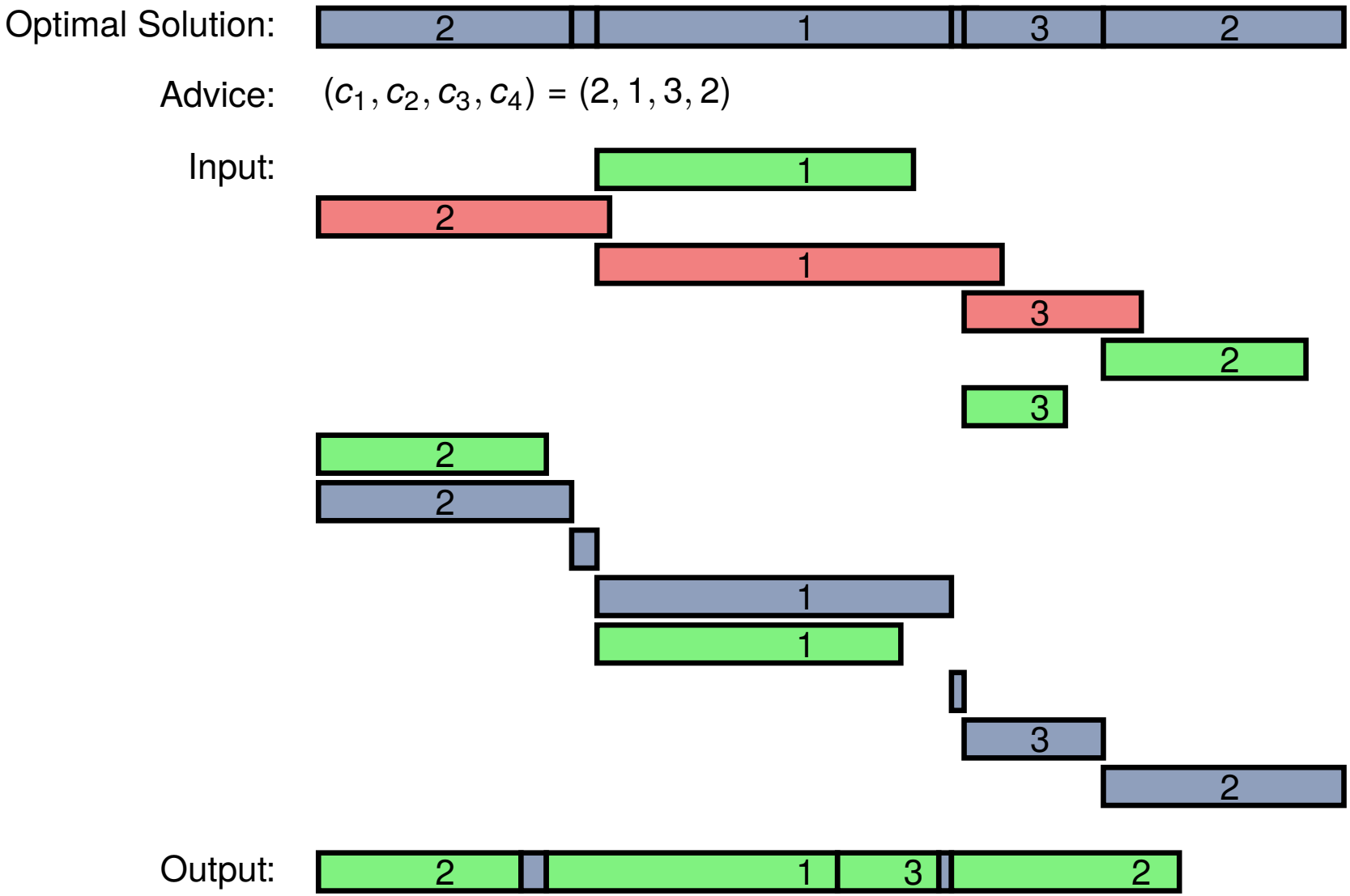
Execution Example



Execution Example



Execution Example



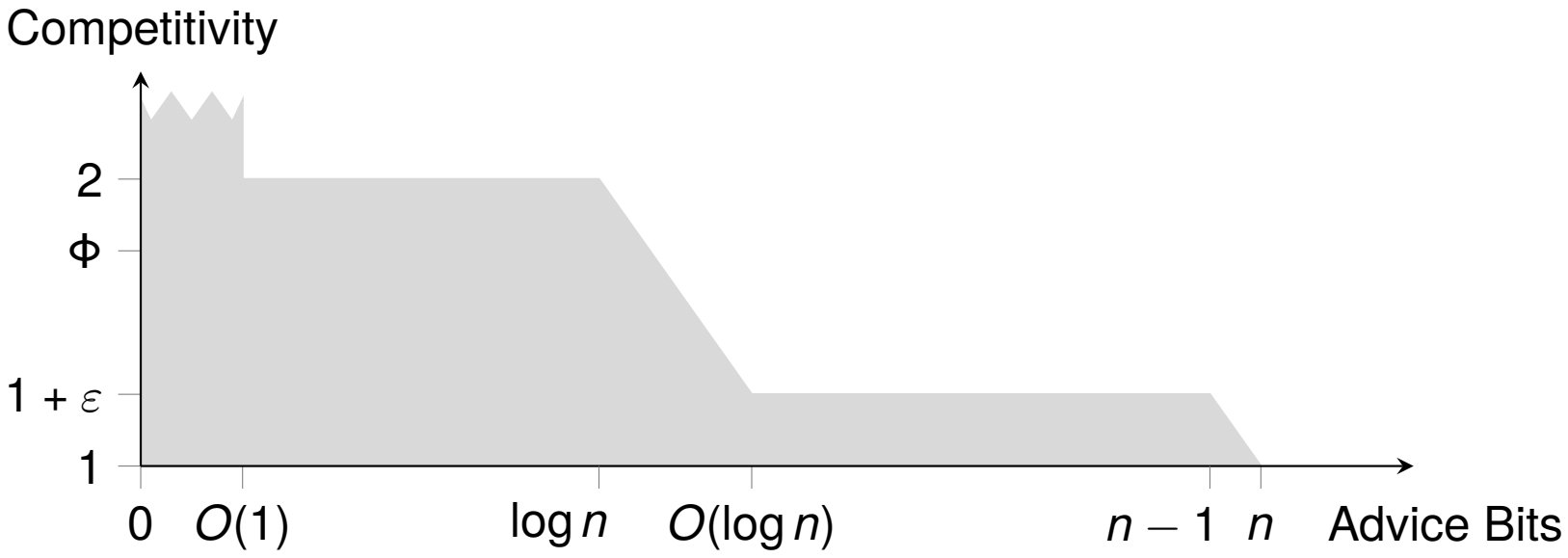
Near-Optimality with Constant Advice

Bounding the Competitivity

- Big items: Losing at most a factor ε
  - All slots are filled in the end (Proof by Induction)
  - As many big items from every class as optimal solution
  - Every class for big item spans factor $1 - \varepsilon$
- Small items: Losing at most a factor ε (Greedy)
- Thus $(1 + 2\varepsilon)$-competitive

Conclusion

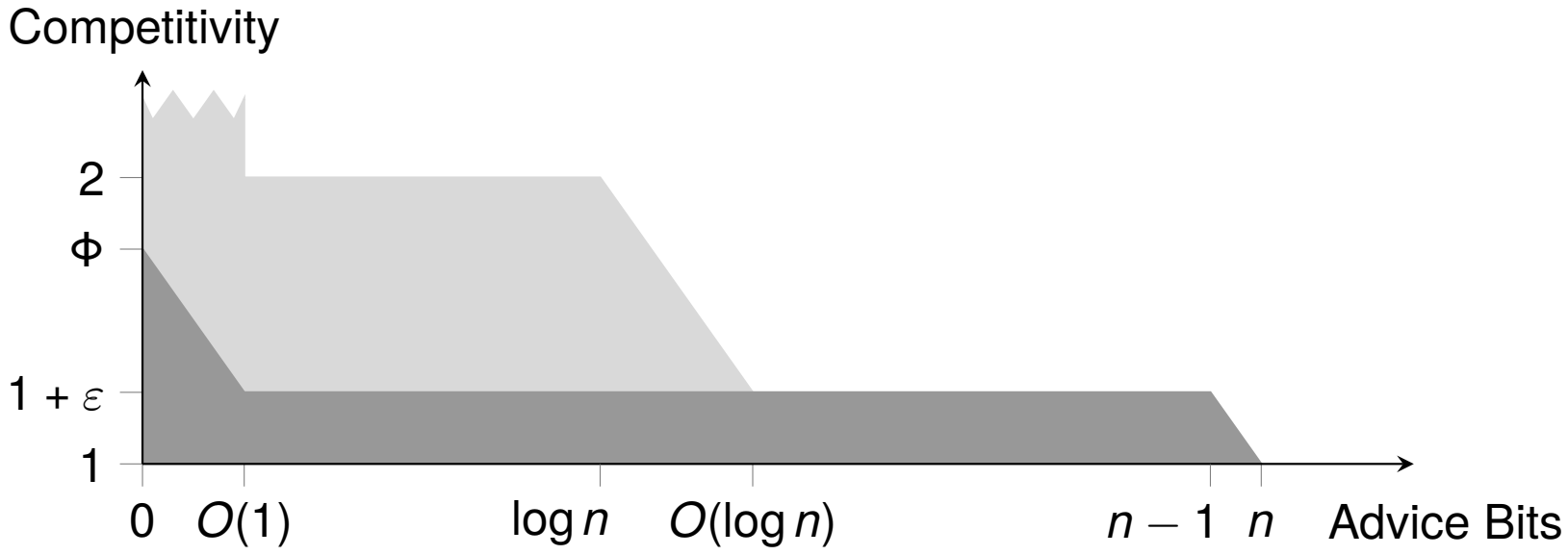
Advice Complexity Behavior



Classical Knapsack: Two thresholds

- No advice: Unbounded competitivity
- Constant advice (a single bit): Drop to 2-competitivity
- Logarithmic advice necessary for any improvement
- Logarithmic advice sufficient for near-optimality
- Linear advice (one bit per item) necessary for optimality

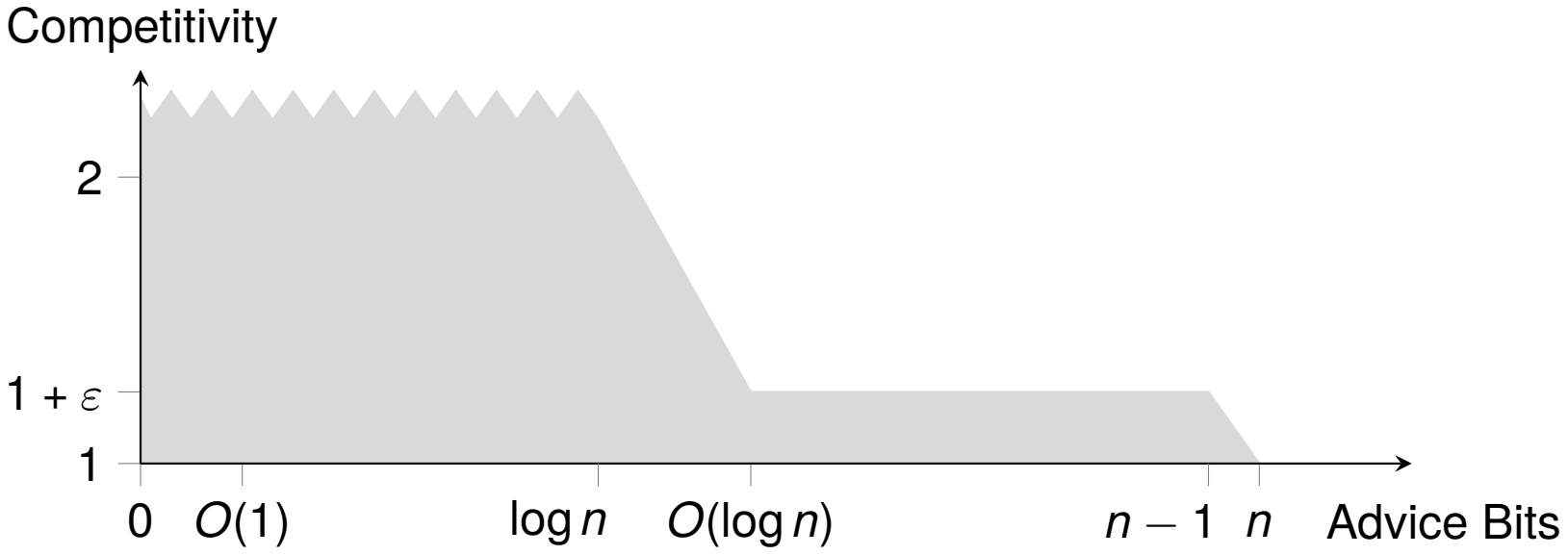
Conclusion: Advice Complexity Behavior



With removability: Collapse to a single threshold

- No advice: Φ-competitive (Golden ratio $\Phi \approx 1.618$)
- Constant advice: Jump down to near-optimality
- Logarithmic advice: Still being near-optimal
- Linear advice (one bit per item) necessary for optimality

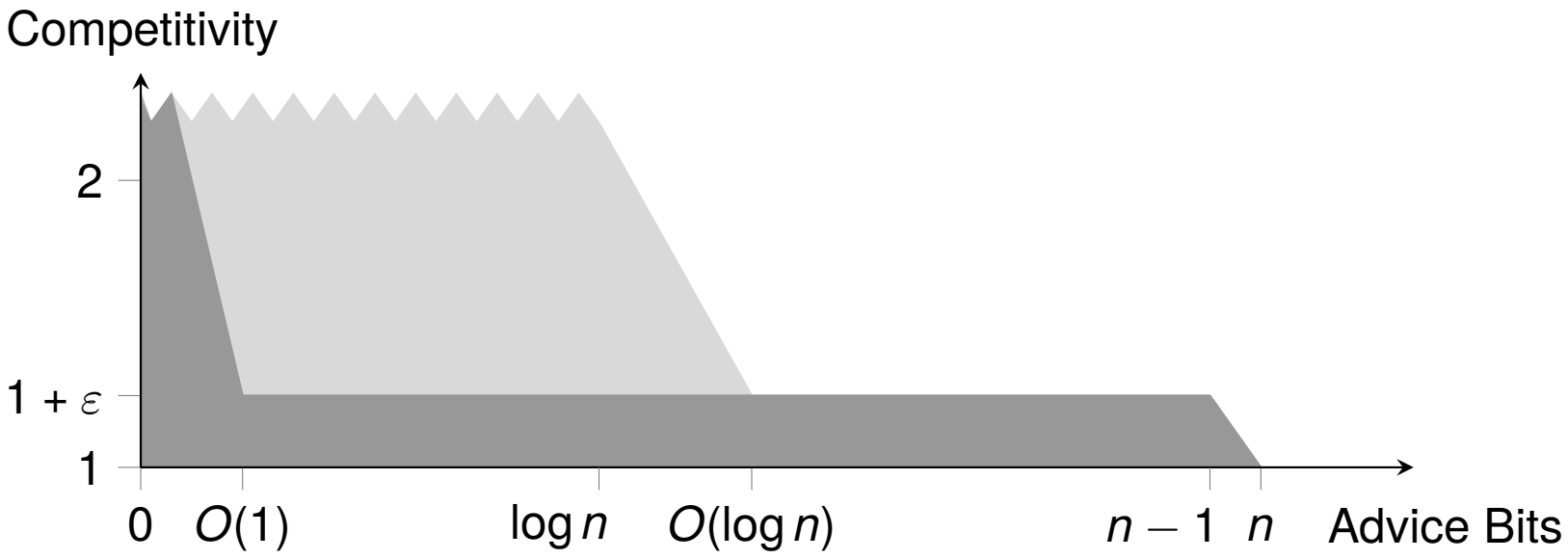
Outlook: Non-Proportional Variant



Non-Proportional Online Knapsack

- Without removability: A single threshold at logarithmic advice, jump from unbounded to near-optimal competitivity

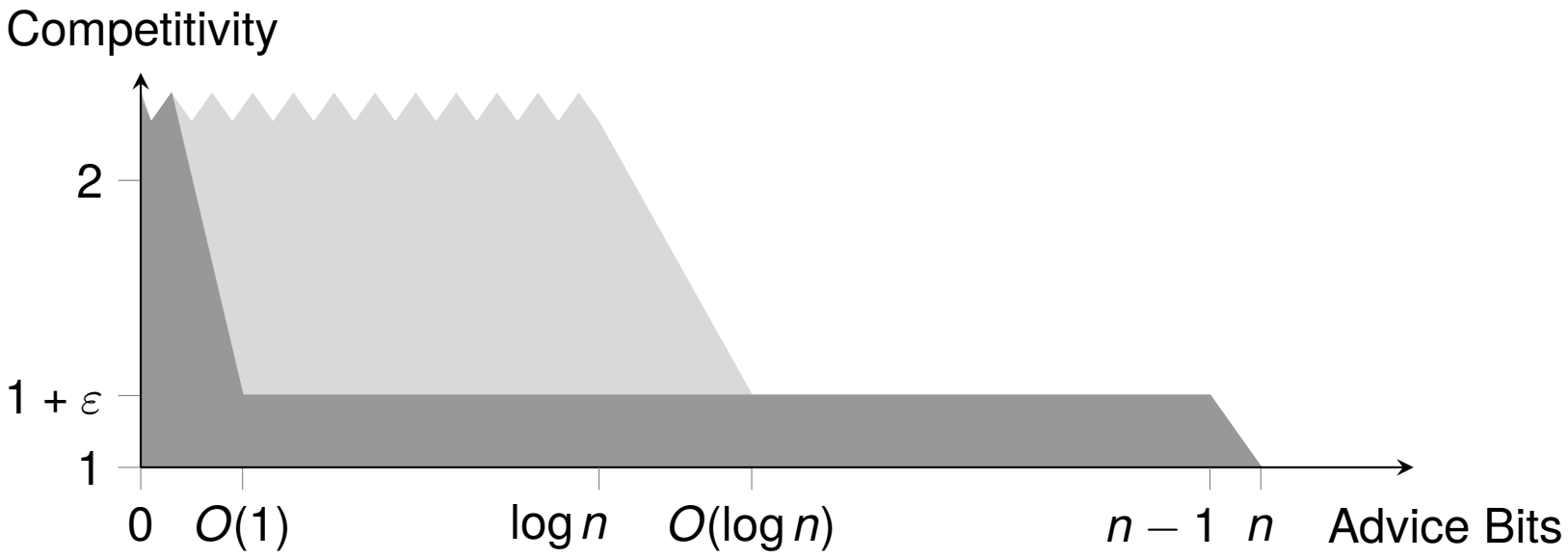
Outlook: Non-Proportional Variant



Non-Proportional Online Knapsack

- Without removability: A single threshold at logarithmic advice, jump from unbounded to near-optimal competitivity
- With removability: Threshold moves to constant advice

Outlook: Non-Proportional Variant



Non-Proportional Online Knapsack

- Without removability: A single threshold at logarithmic advice, jump from unbounded to near-optimal competitivity
- With removability: Threshold moves to constant advice
- Proof starts similarly, but many new tricks required

Questions?