CSCI 232:
Data Structures and Algorithms

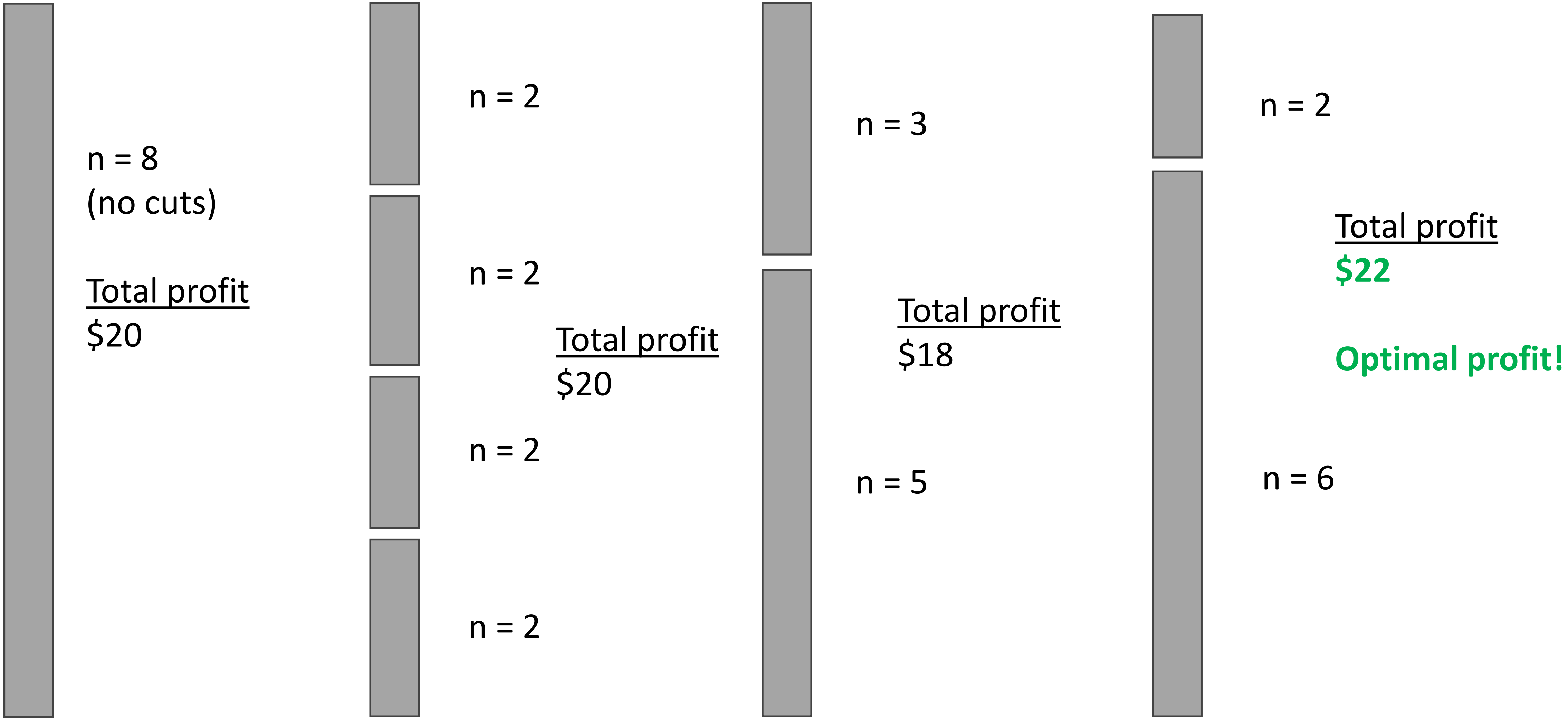
Dynamic Programming (Part 3)

Reese Pearsall
Spring 2025

Rod Cutting

Given a rod of length n inches, and an array of prices that includes prices of all pieces of size smaller than n, determine the maximum value obtainable by cutting up the road and selling the pieces.

Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20



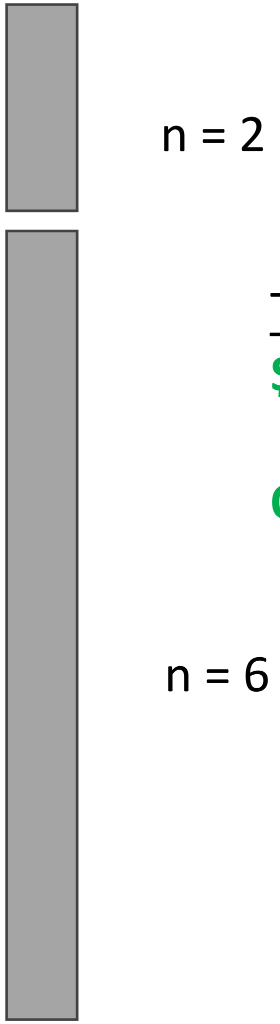
Rod Cutting

Given a rod of length n inches, and an array of prices that includes prices of all pieces of size smaller than n, determine the maximum value obtainable by cutting up the road and selling the pieces.

Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

Optimal Substructure

Our solution for a rod length of n=8, has the optimal solution for rod length of n = 6, and n = 2



Total profit

$22

Optimal profit!

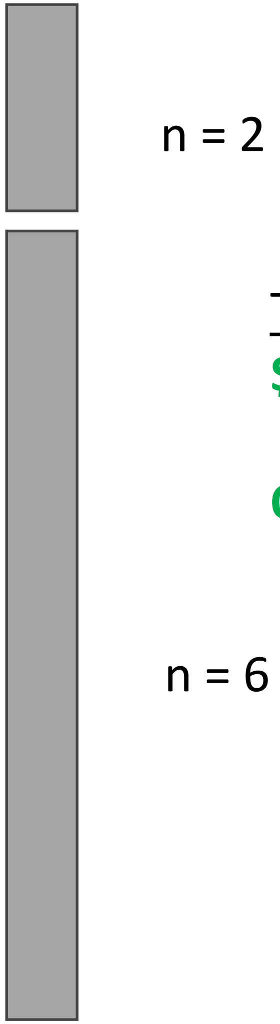
Rod Cutting

Given a rod of length n inches, and an array of prices that includes prices of all pieces of size smaller than n, determine the maximum value obtainable by cutting up the road and selling the pieces.

Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

General Approach:

Compute all possible ways to cut the rod using dynamic programming, and return which one had the highest profit



Total profit

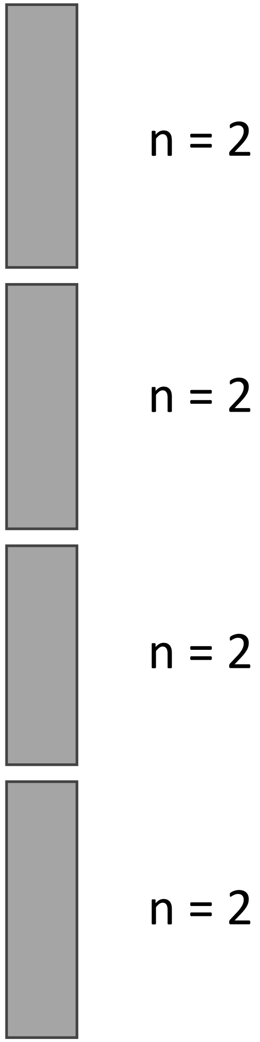
$22

Optimal profit!

Rod Cutting

Given a rod of length n inches, and an array of prices that includes prices of all pieces of size smaller than n, determine the maximum value obtainable by cutting up the road and selling the pieces.

Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20



Total profit
$20

Overlapping subproblems

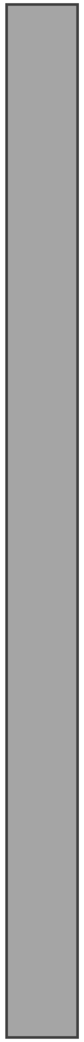
We will compute the optimal way to cut a rod of length n=2 many times. We will use memoization to make sure we don’t compute problems that we have already solved.

Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

index

n = 8



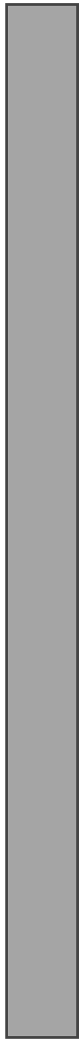
Technically, out algorithm will consider making a cut of length 8 first, but we will skip over this part to avoid confusion

Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

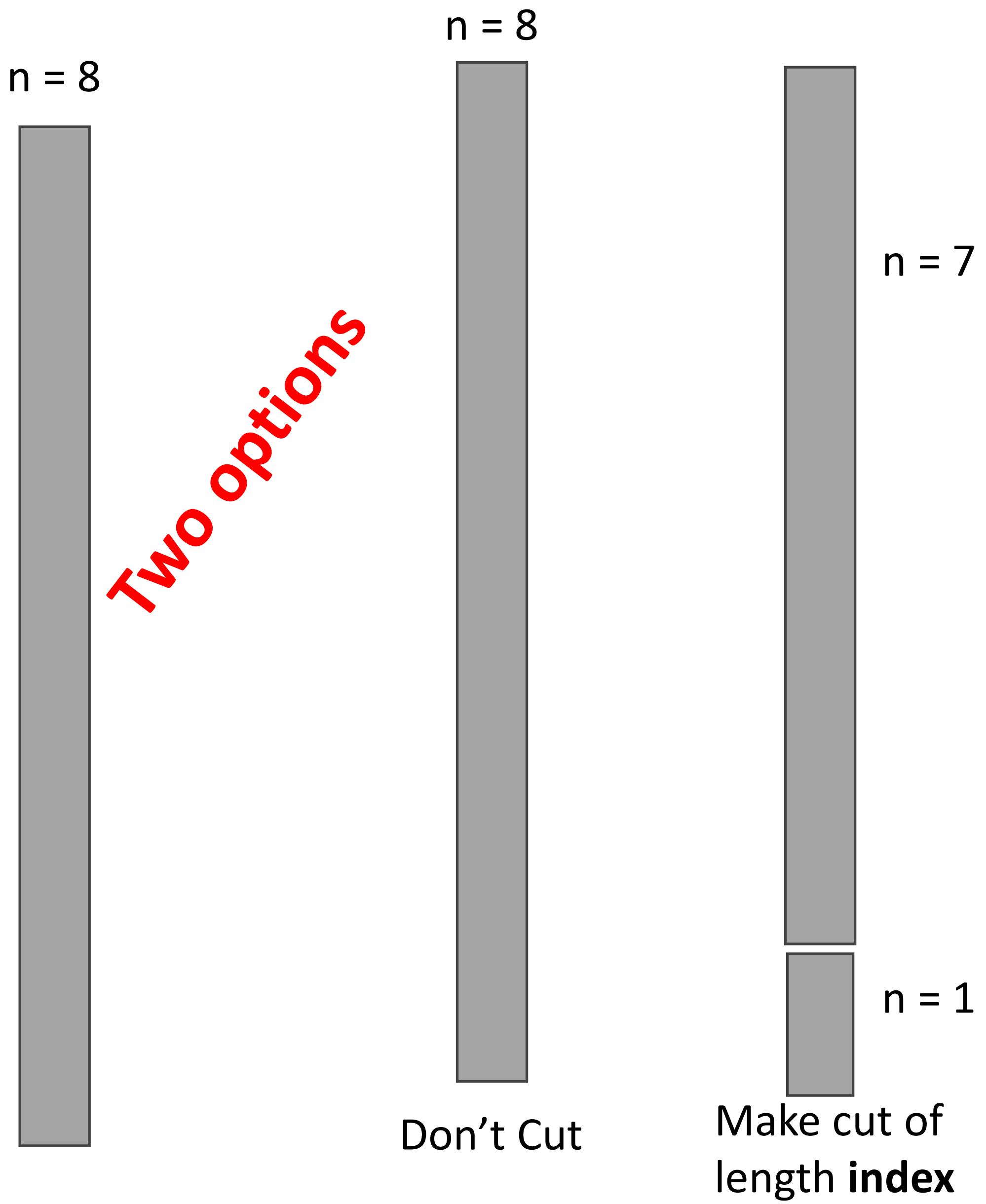
index

n = 8



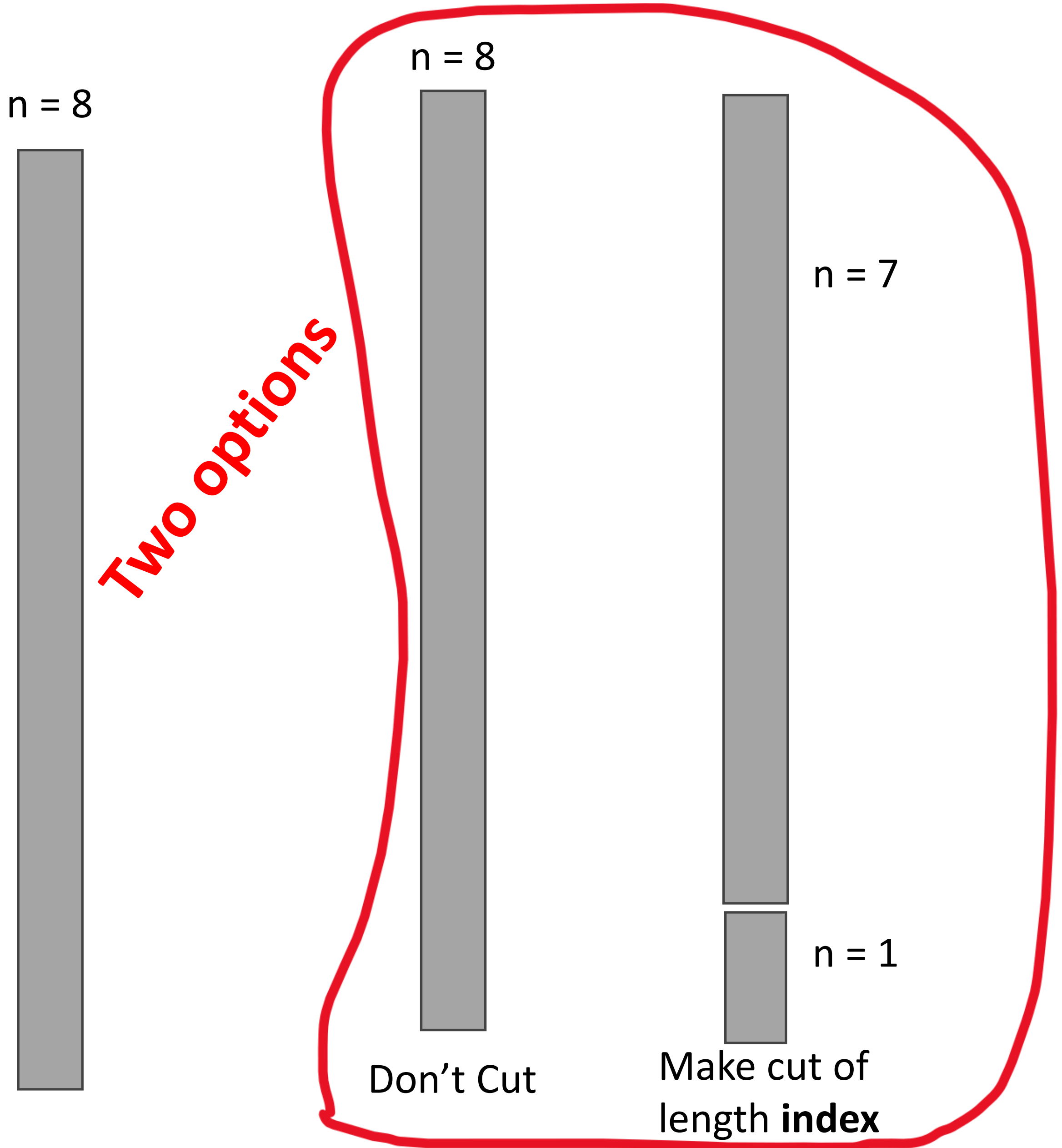
Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20



Rod Cutting

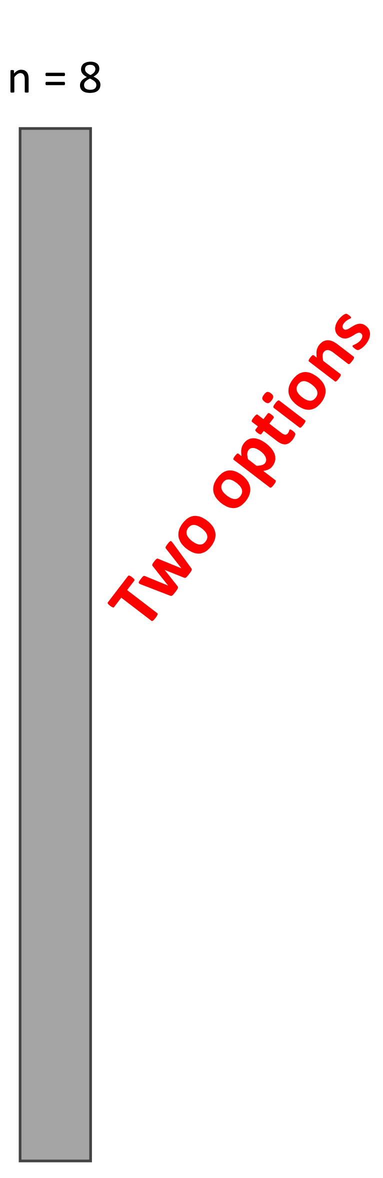
	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20



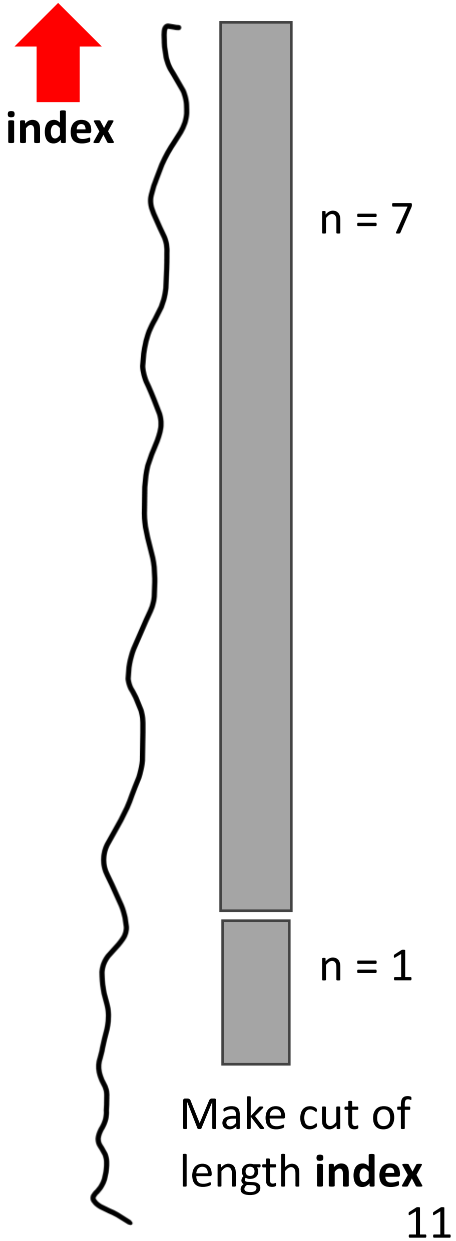
We want to select the option that yield the highest profit

Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20



Now we recurse, and check a new cut value

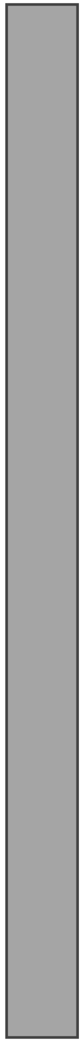


Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

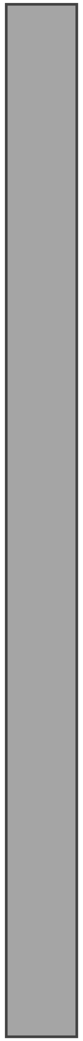
index

n = 8



Two options

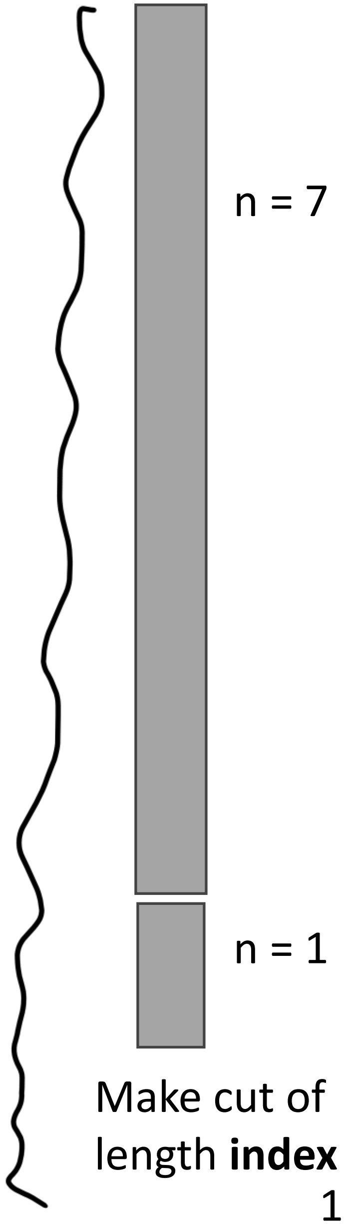
n = 8



Don't Cut

Now we recurse, and check a new cut value

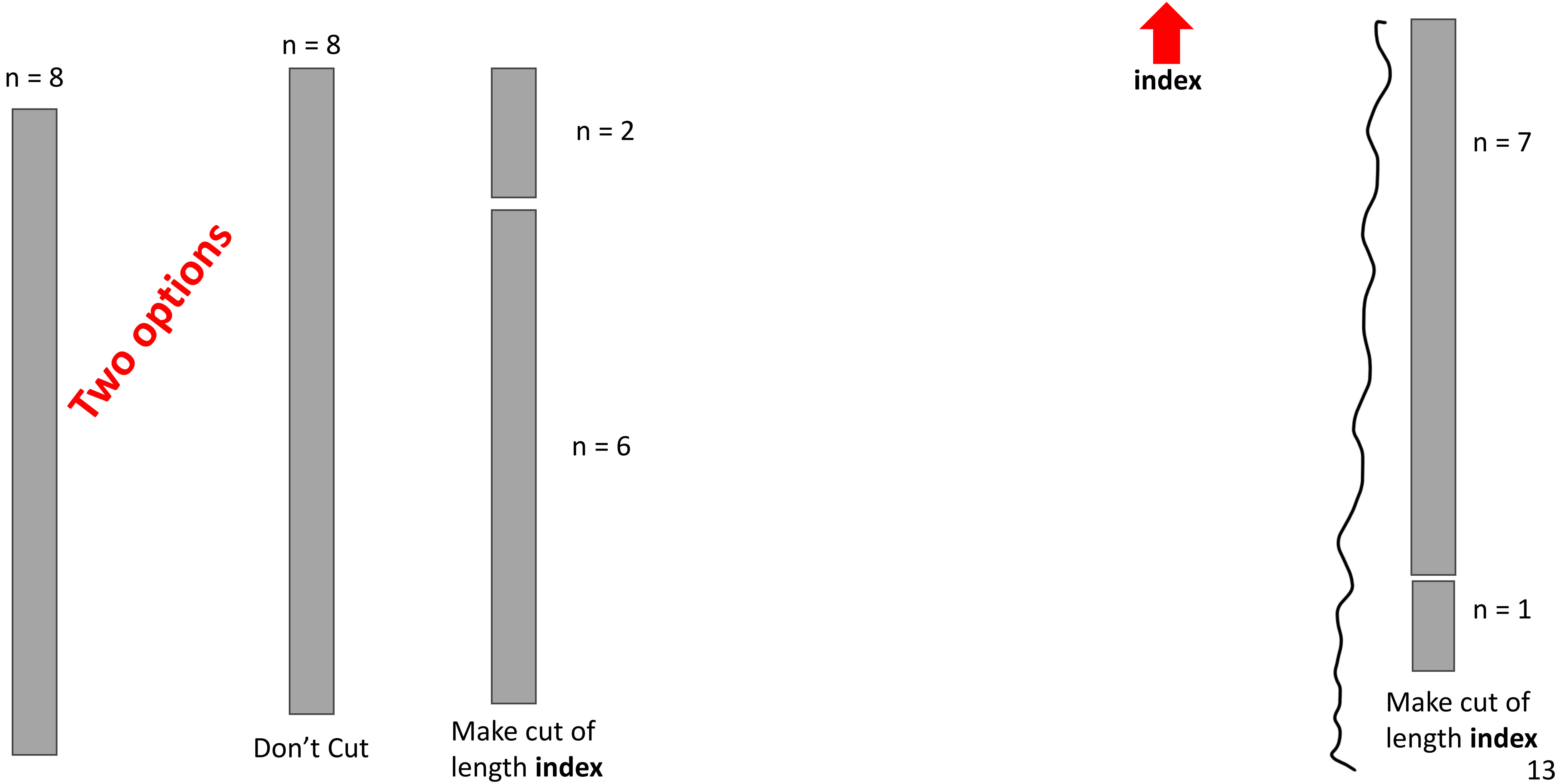
(index – 1)



Make cut of length **index**

Rod Cutting

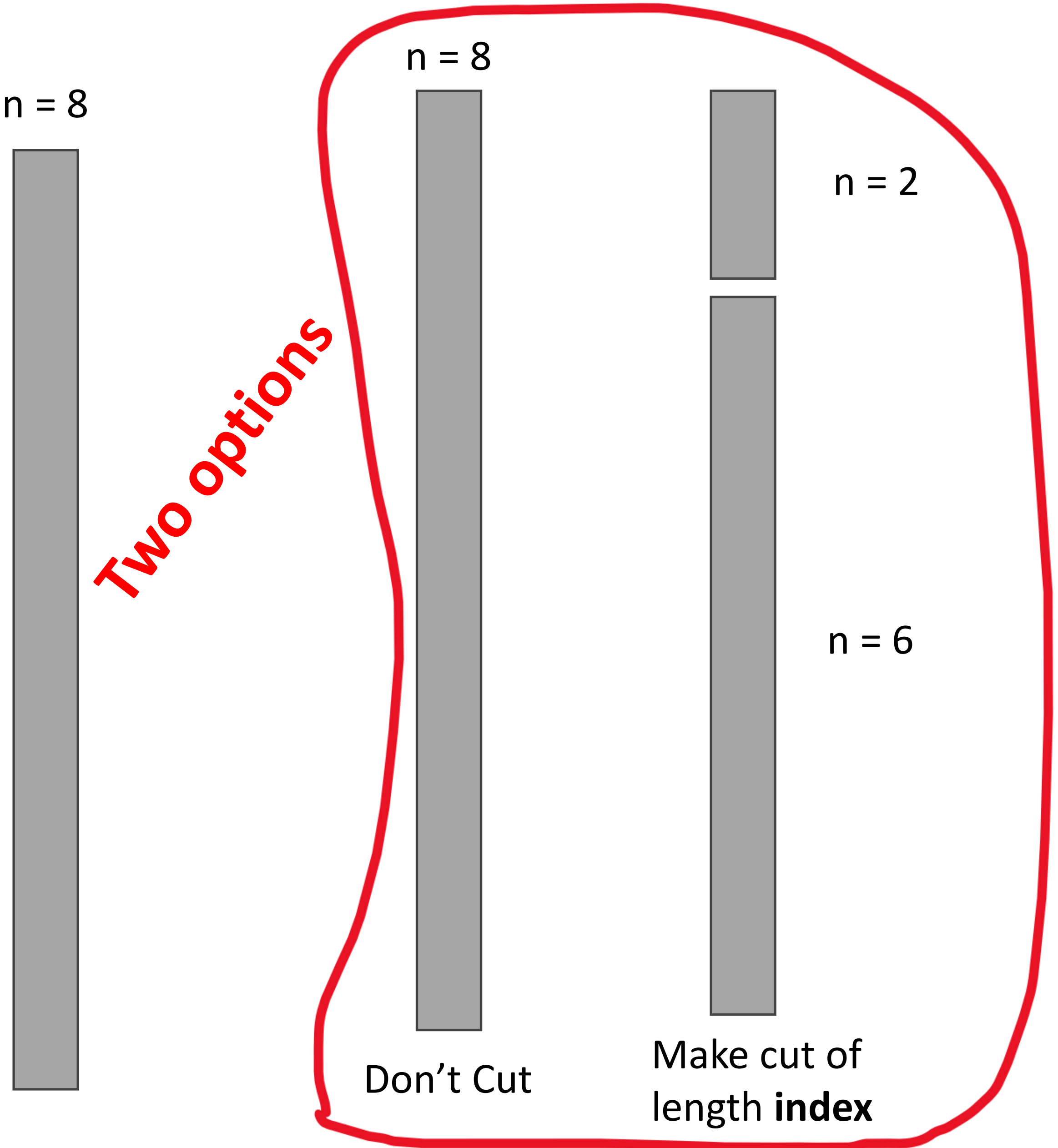
	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20



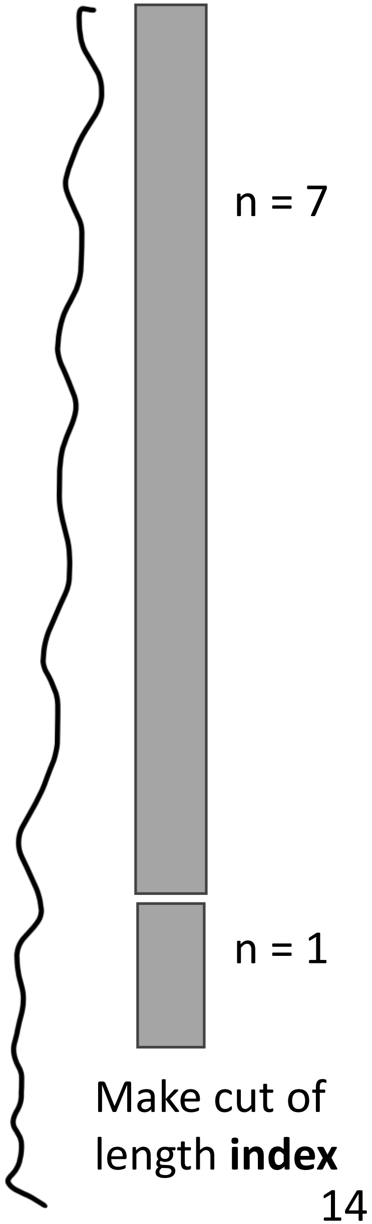
Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

index

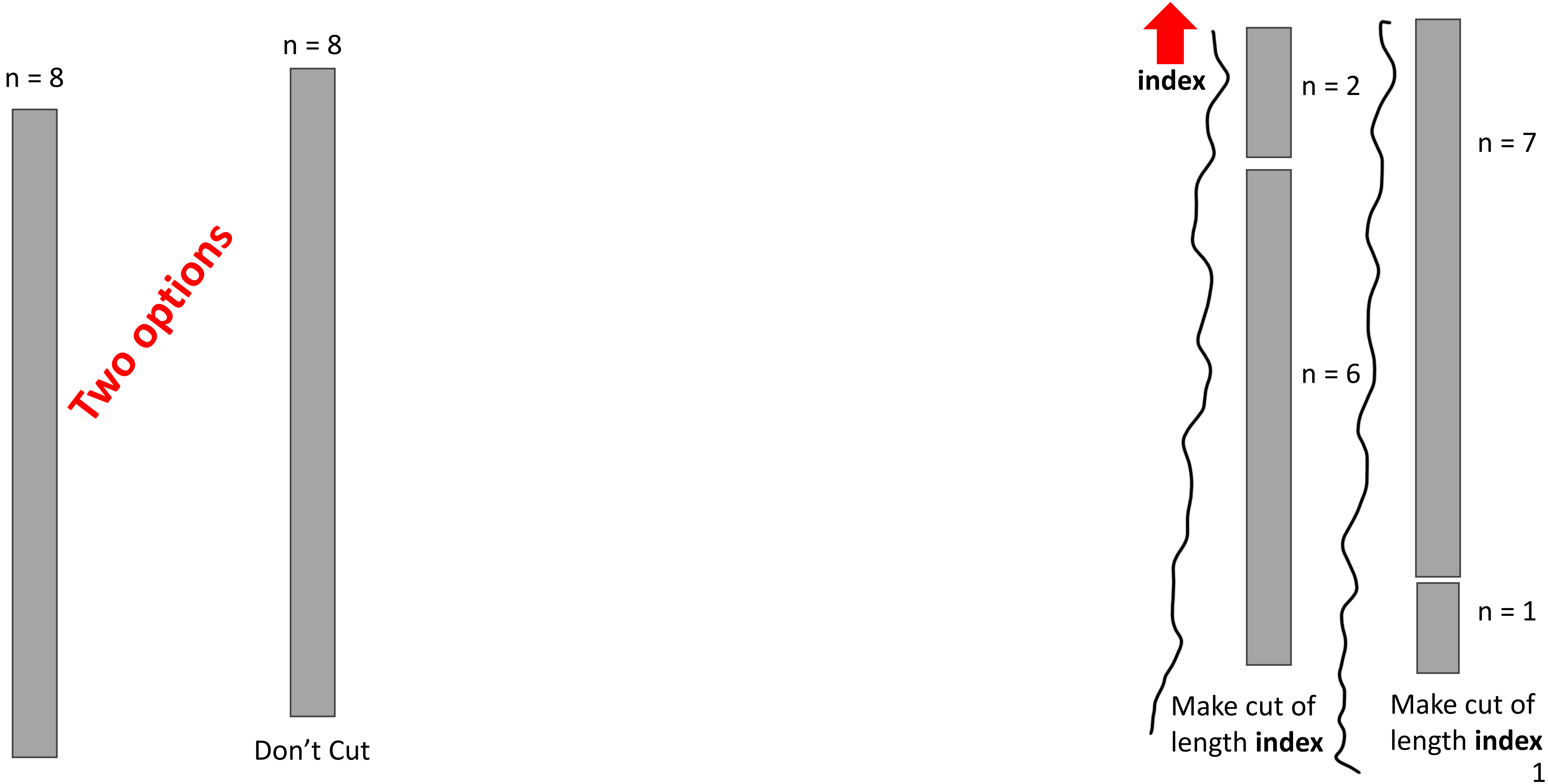


We want to select the option that yield the highest profit



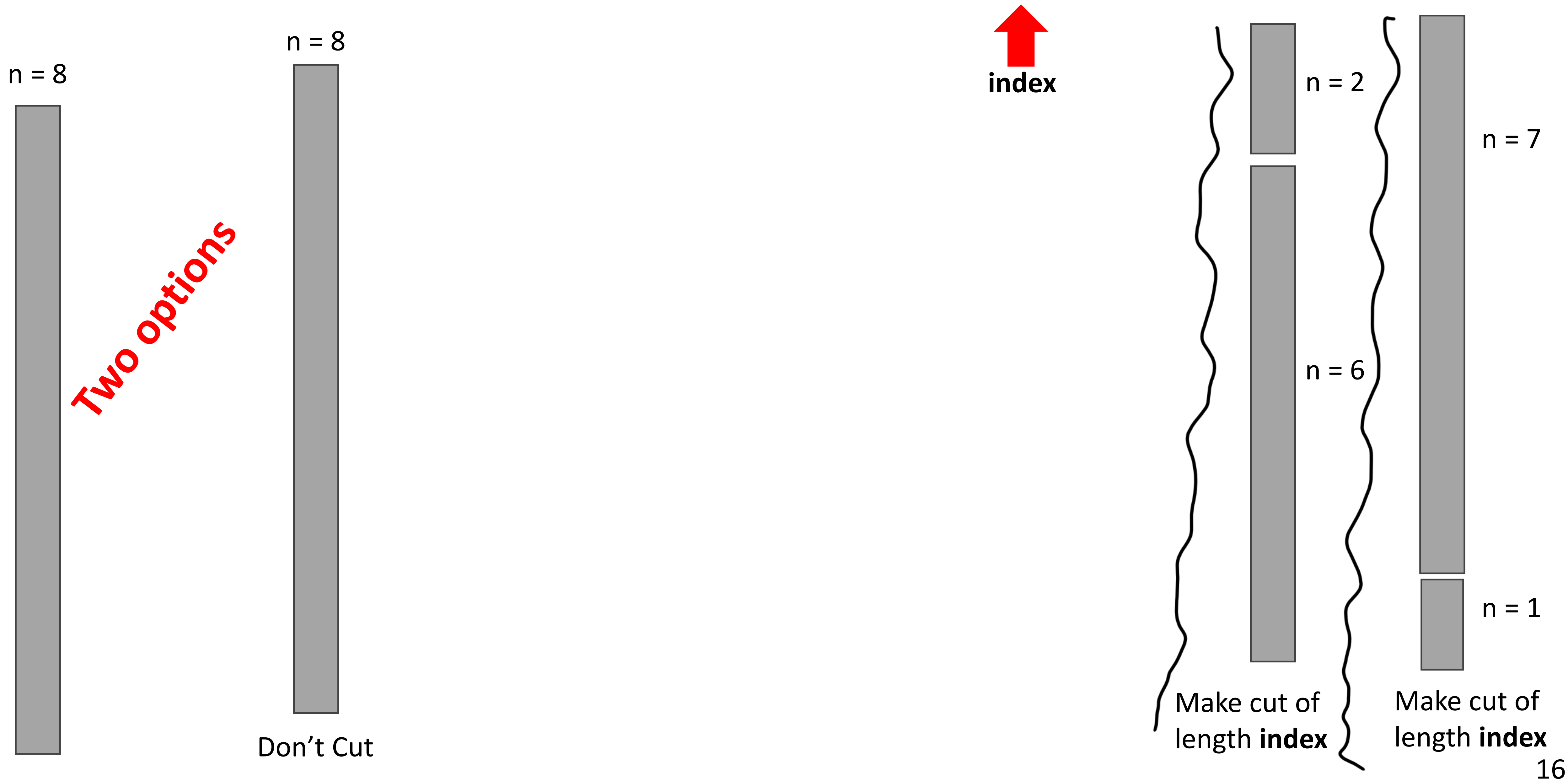
Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20



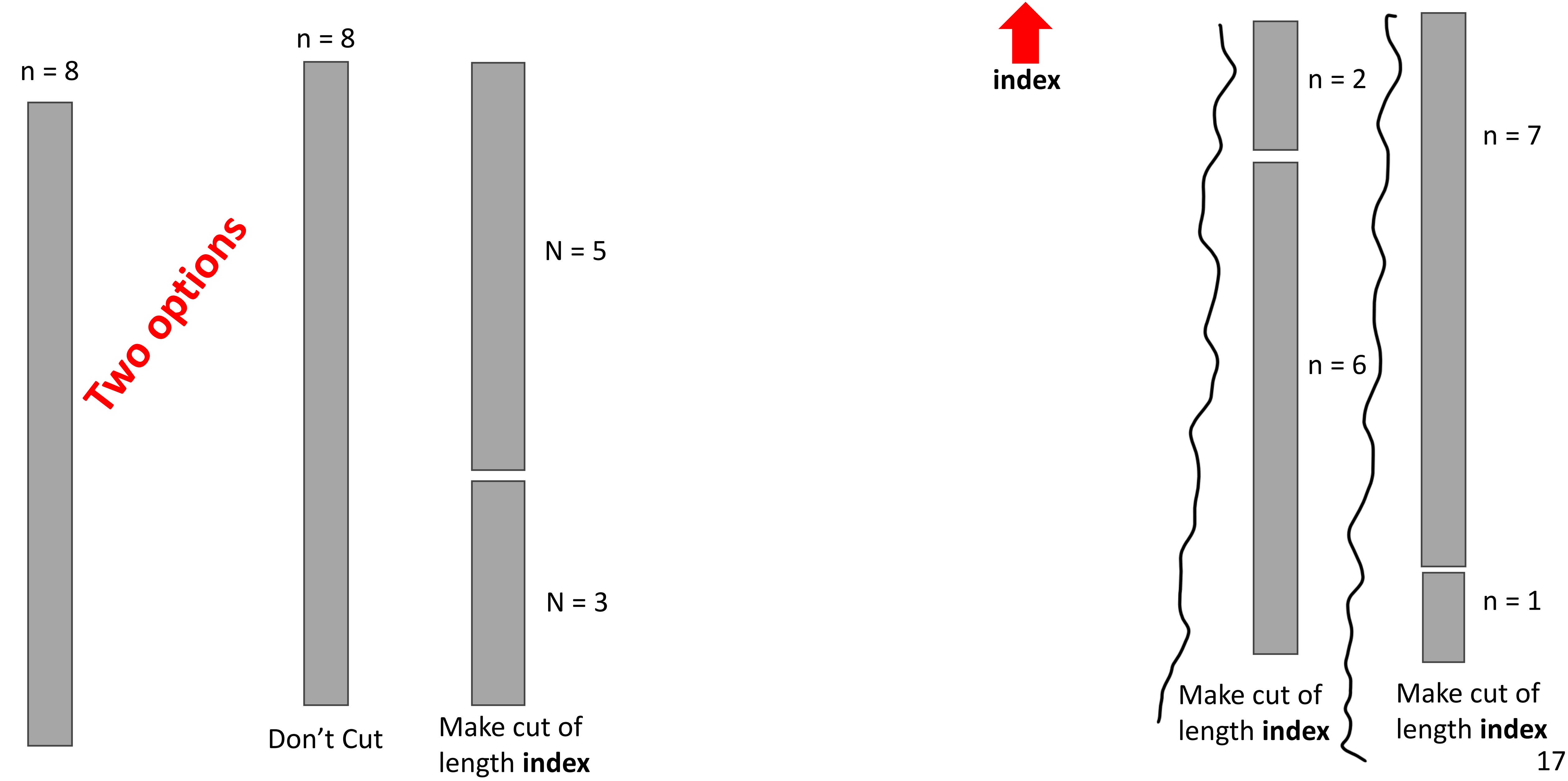
Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20



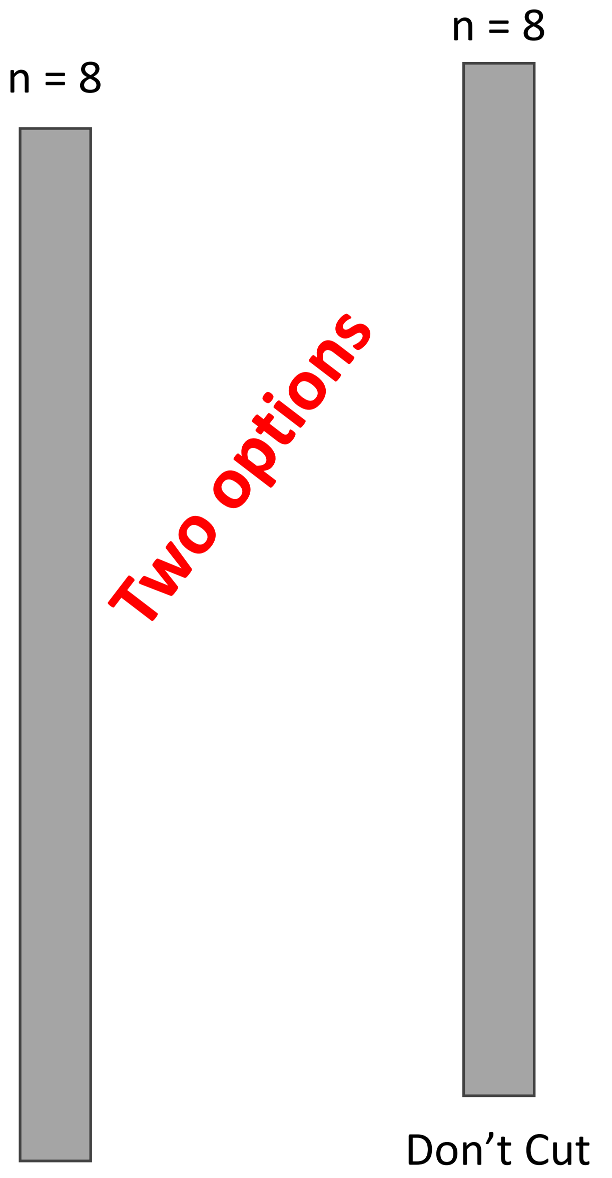
Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

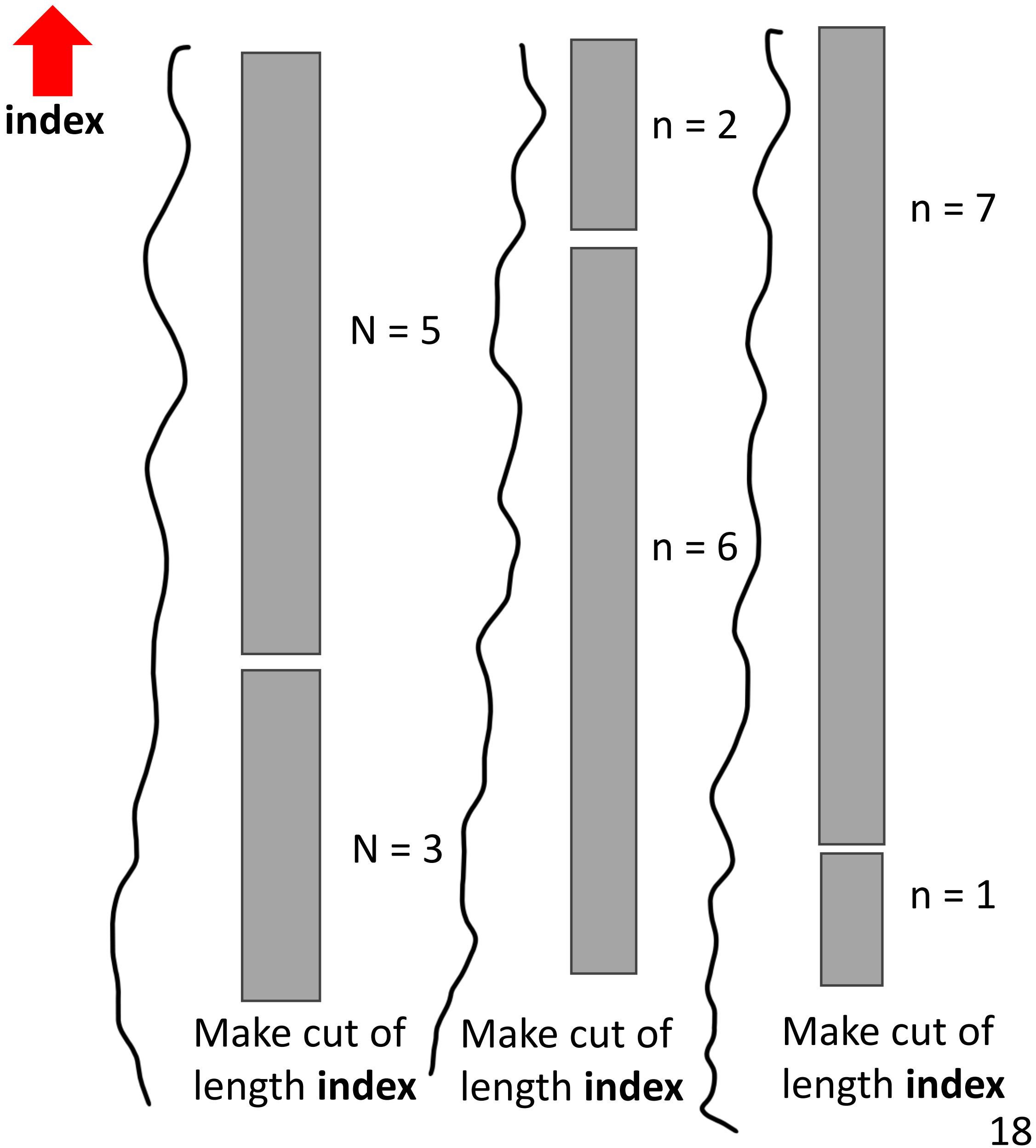


Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

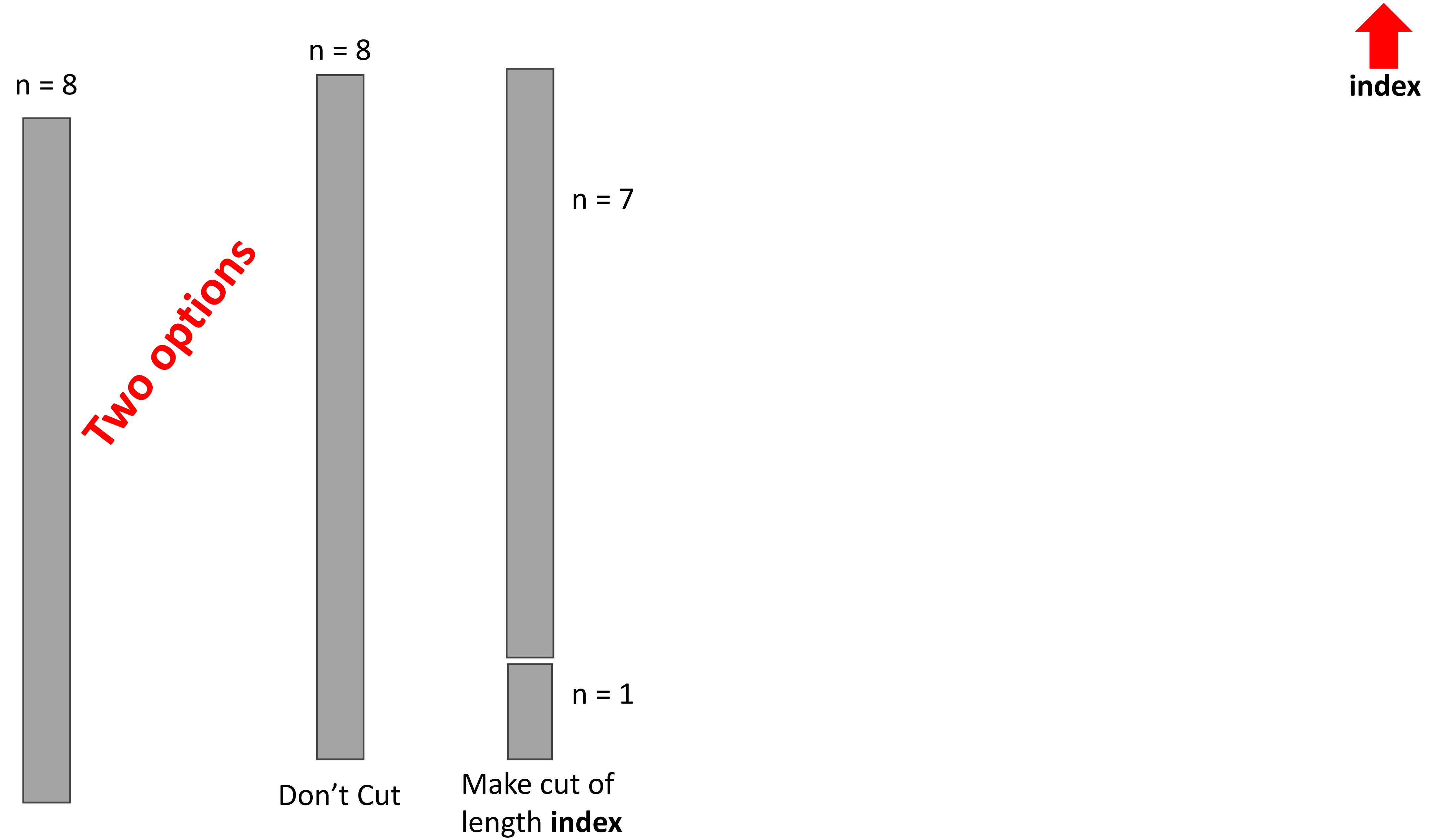


Whenever we don't make the cut, we don't adjust the size of the rod, but we check the next cut length



Rod Cutting

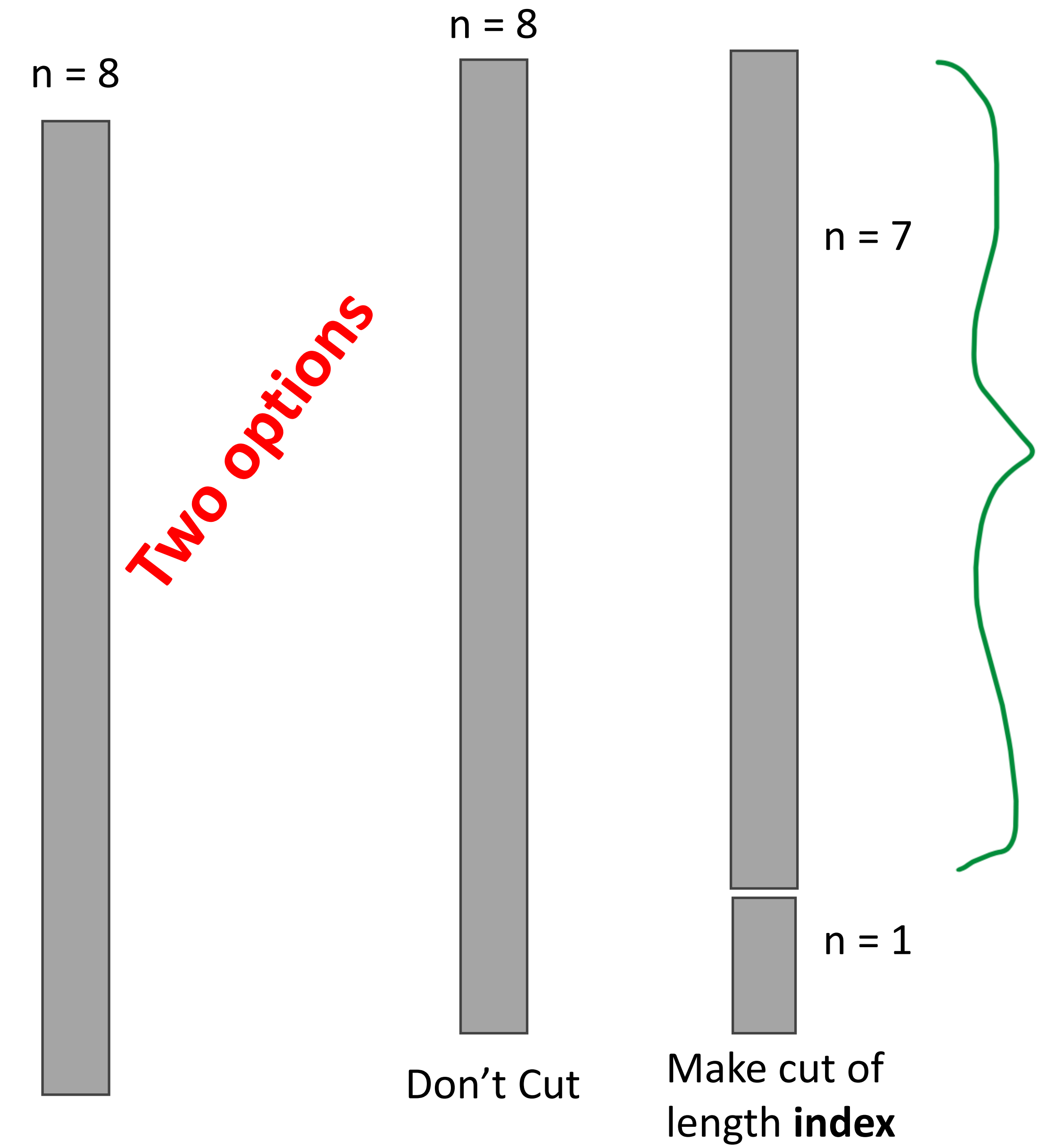
	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20



Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

index



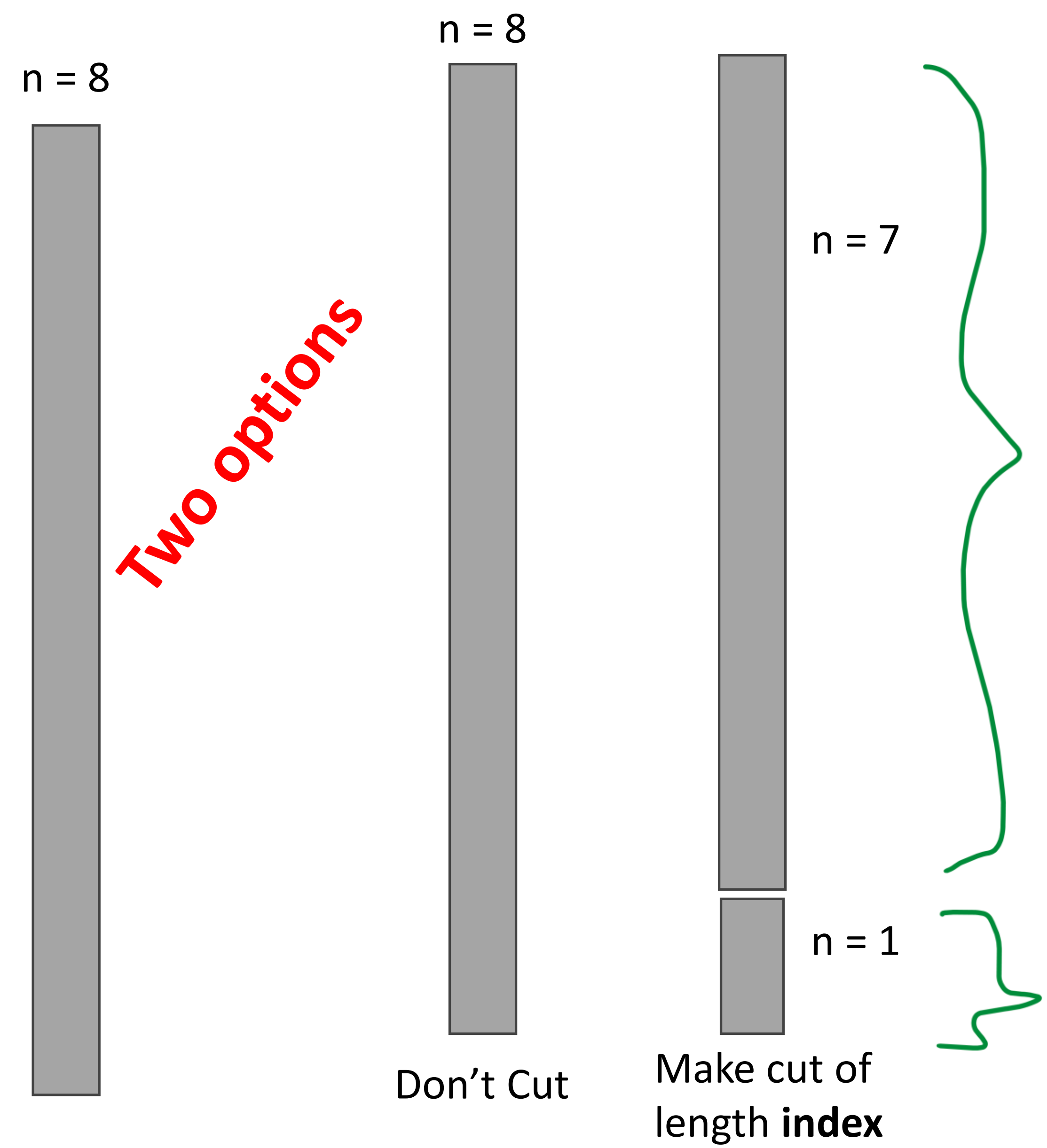
We made a cut of length index, so lets figure out how much that piece is worth!

```
prices[index]
```

Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

index



We made a cut of length index, so lets figure out how much that piece is worth!

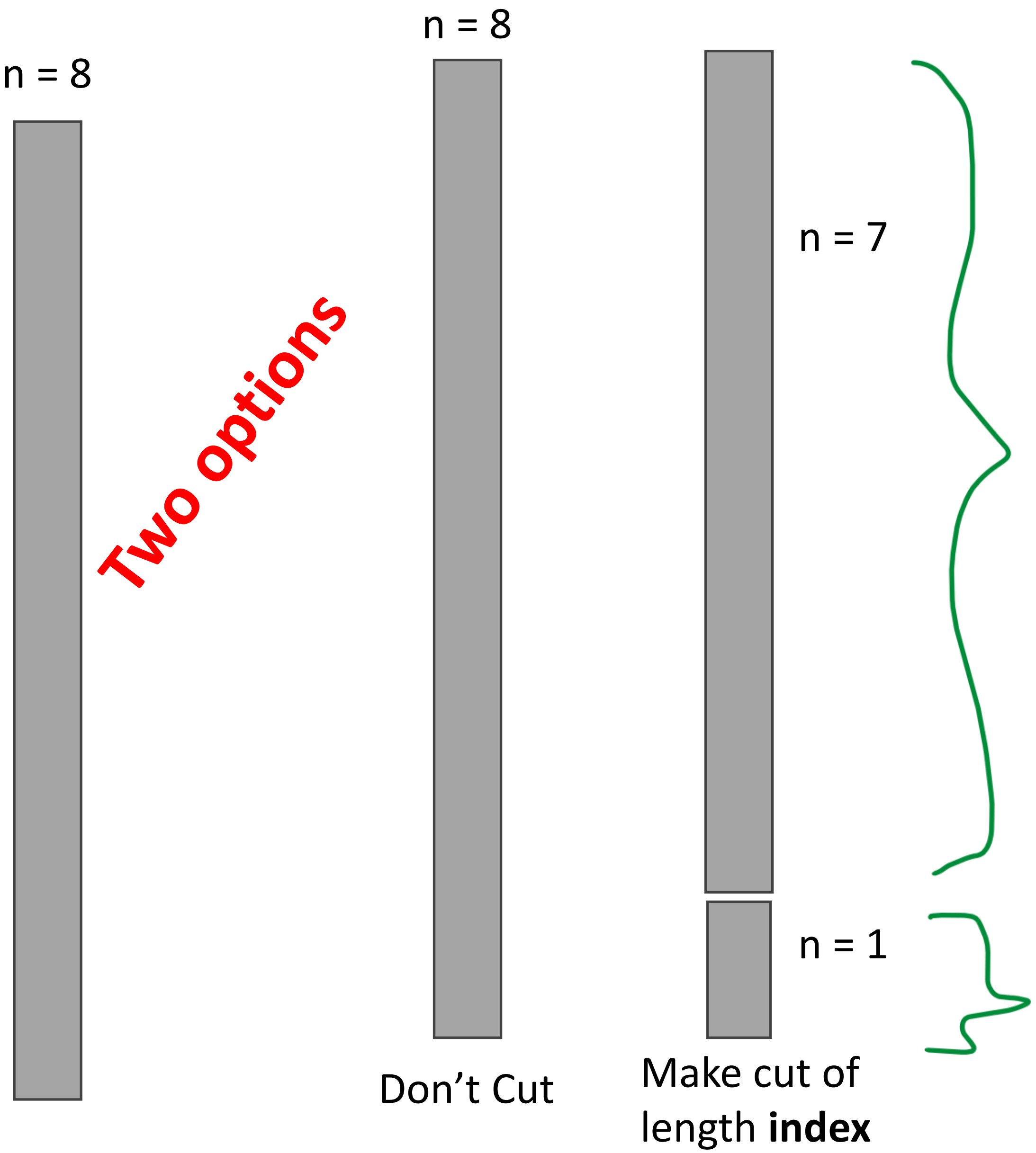
```
prices[index]
```

We have 1 inch of rod left, so we need to now figure out the optimal way to cut this

Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

index



We made a cut of length index, so lets figure out how much that piece is worth!

```
prices[index]
```

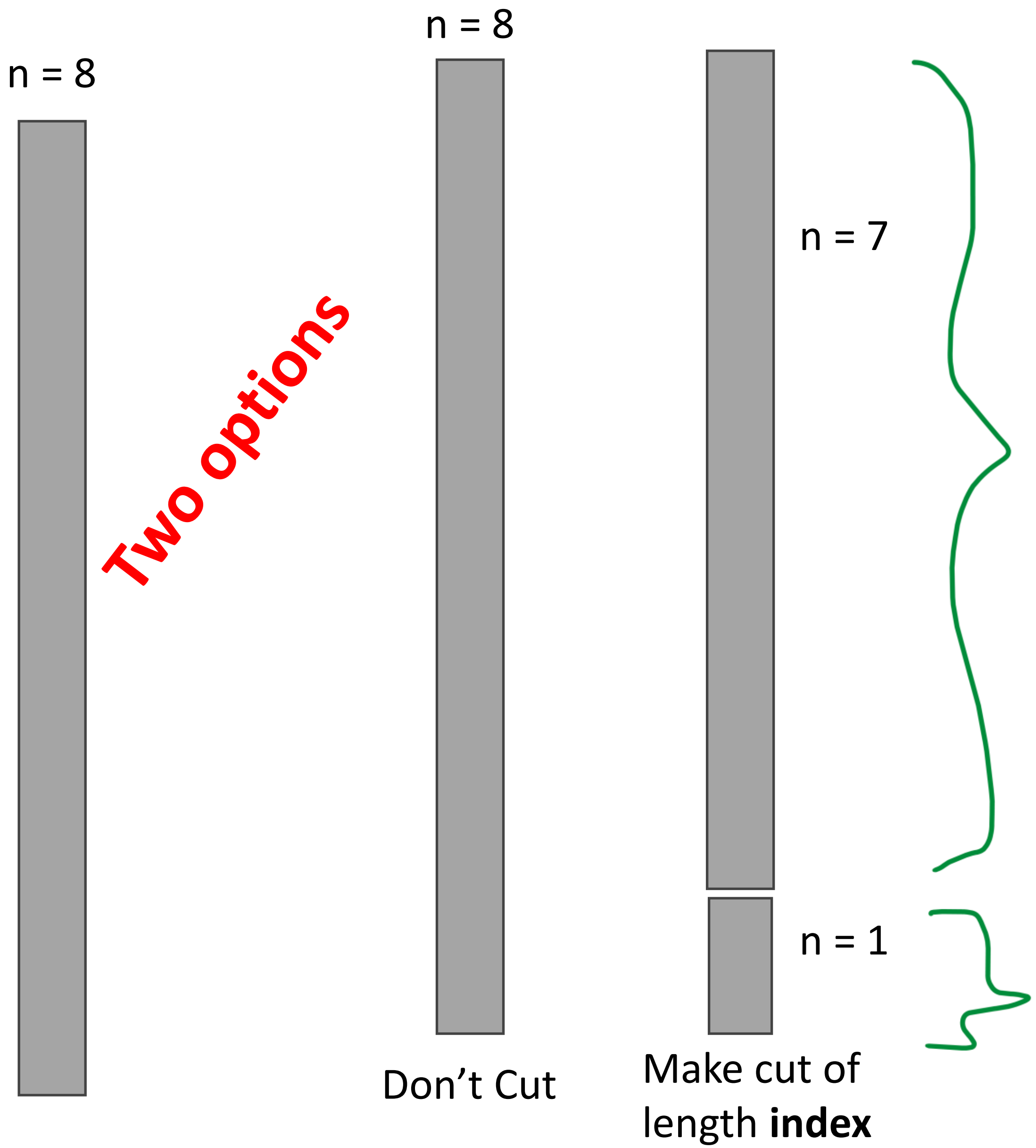
Length of cut made = (index + 1)

We have 1 inch of rod left, so we need to now figure out the optimal way to cut this
--Recurse!

Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

index



We made a cut of length index, so lets figure out how much that piece is worth!

`prices[index]`

Length of cut made = (index + 1)

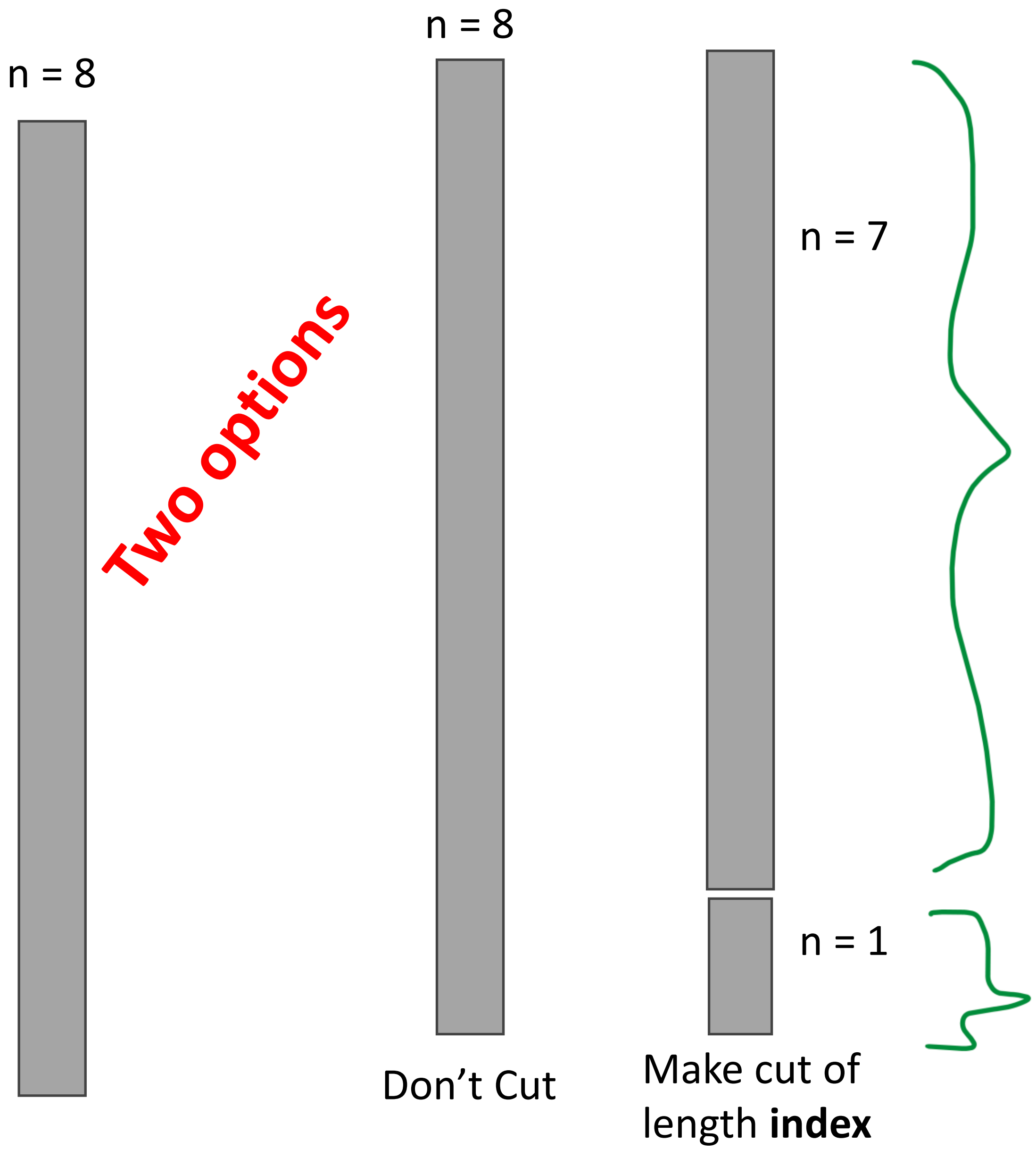
New subproblem = `n- length_of_cut`

We have 1 inch of rod left, so we need to now figure out the optimal way to cut this
--Recurse!

Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

index



We made a cut of length index,
so lets figure out how much that
piece is worth!

`prices[index]`

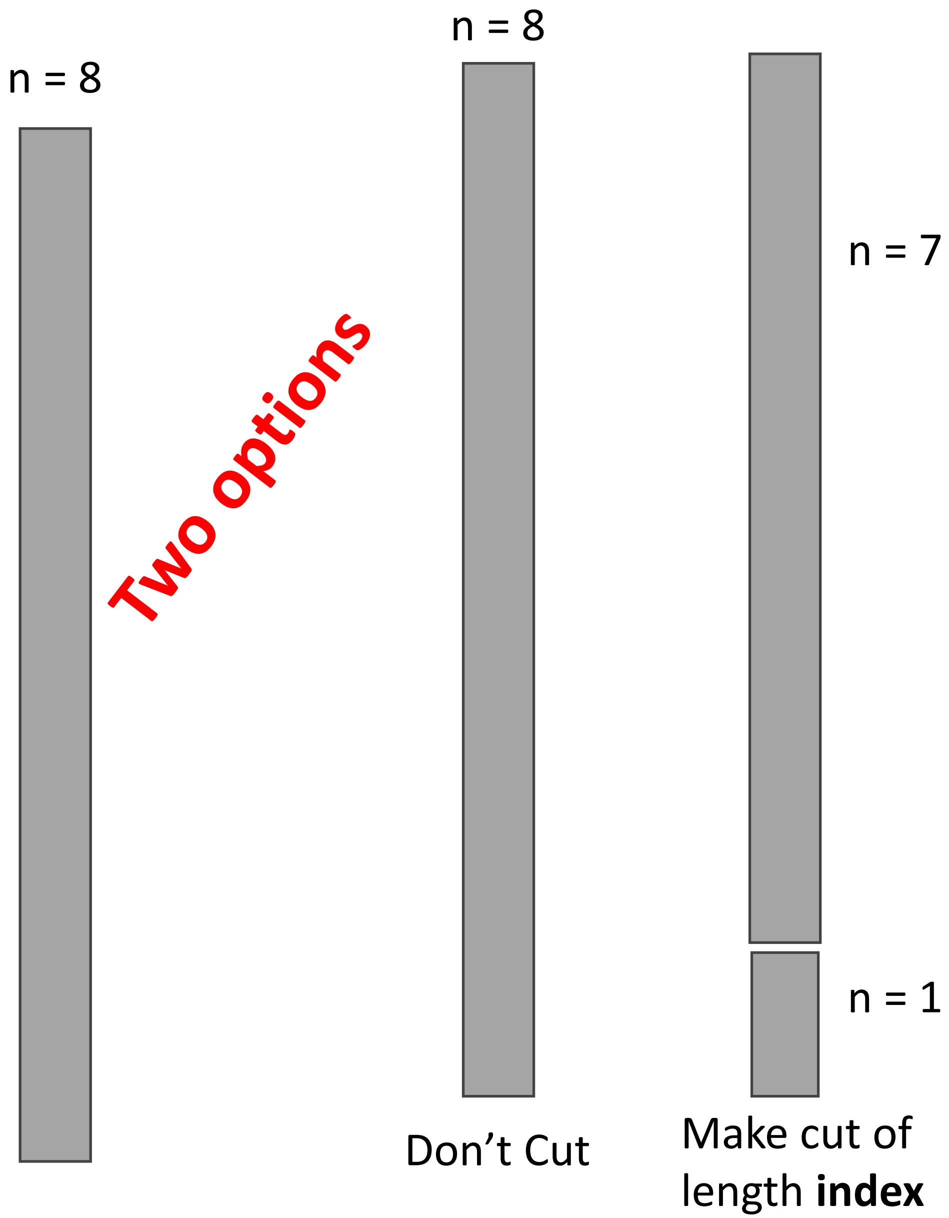
Length of cut made = (index + 1)

New subproblem = `n- length_of_cut`

We have 1 inch of rod left, so we
need to now figure out the
optimal way to cut this
--Recurse!

Rod Cutting

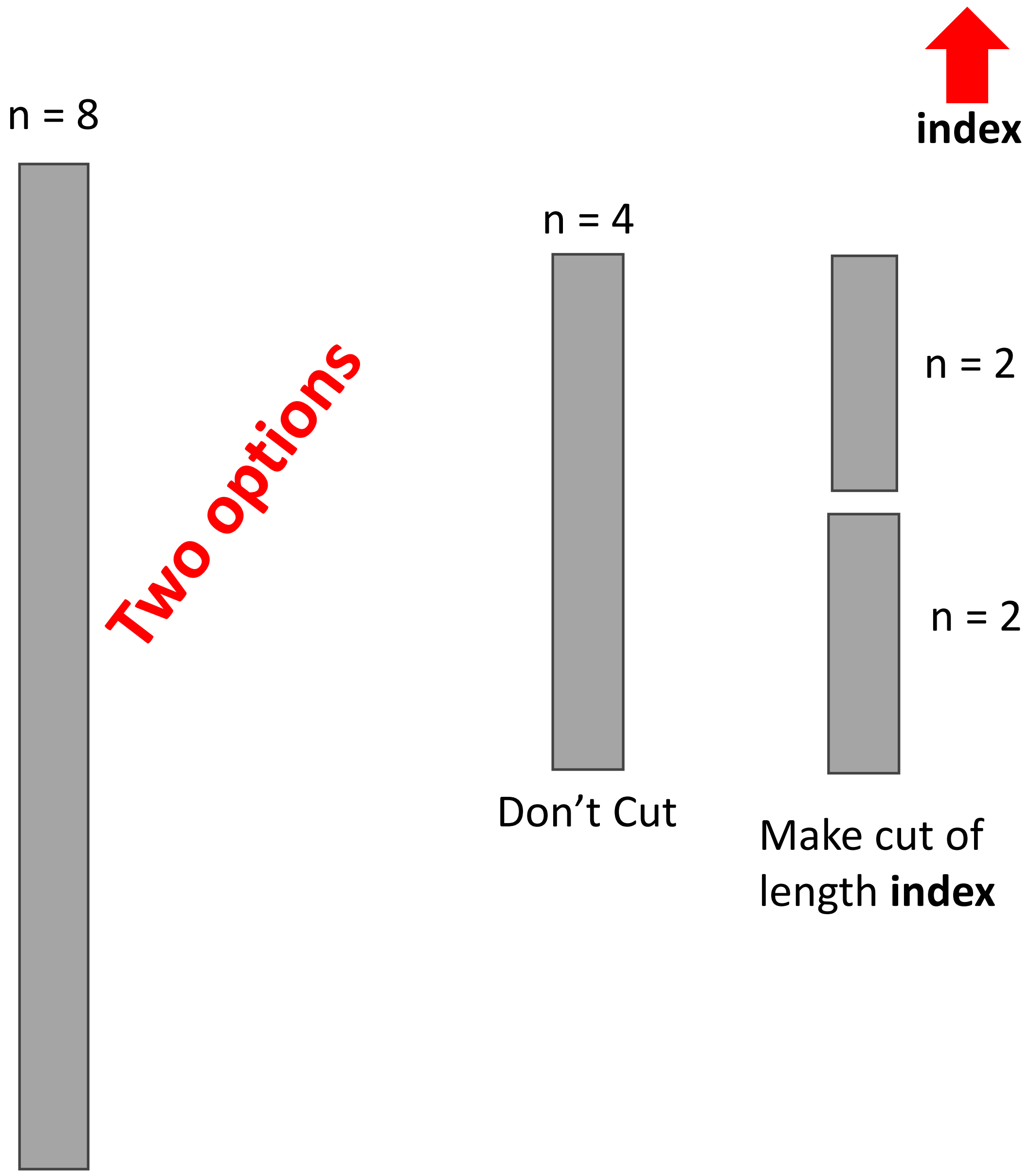
	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20



Whenever we make the cut, we adjust the size of the rod, but keep the same index

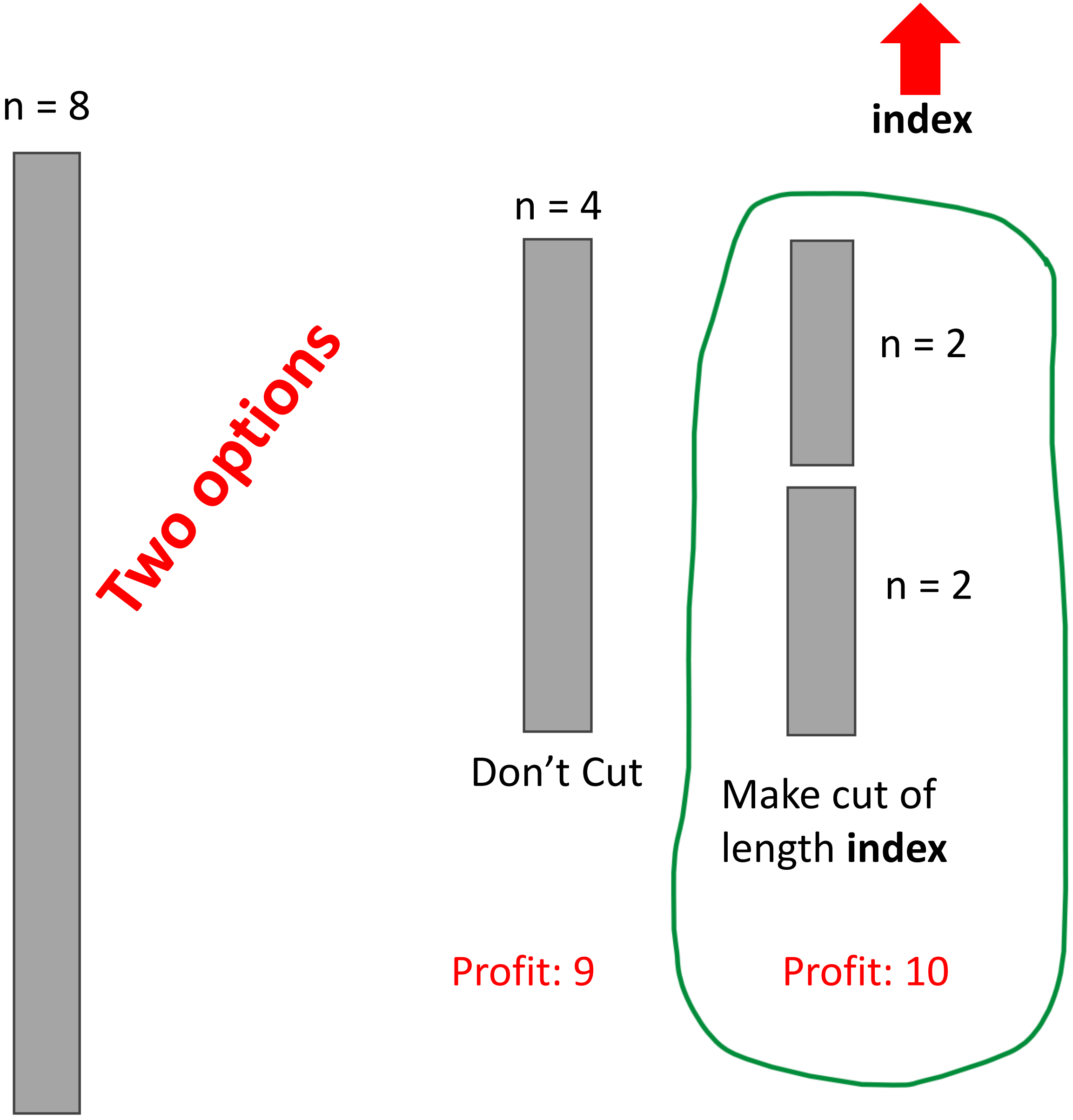
Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20



Rod Cutting

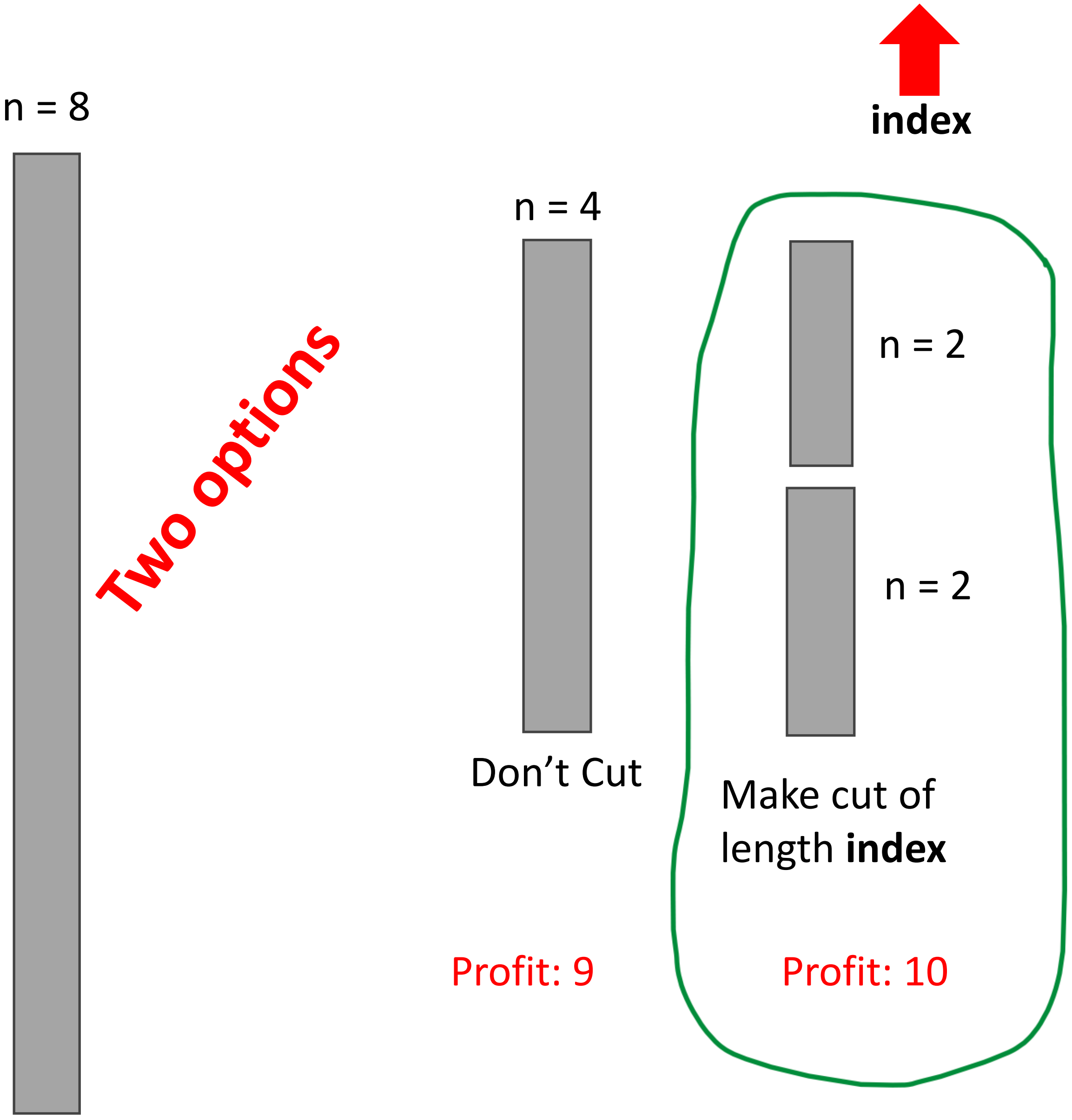
	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20



Given a rod of length 4 and a potential cut value of length 2, the optimal solution is to **make the cut**

Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

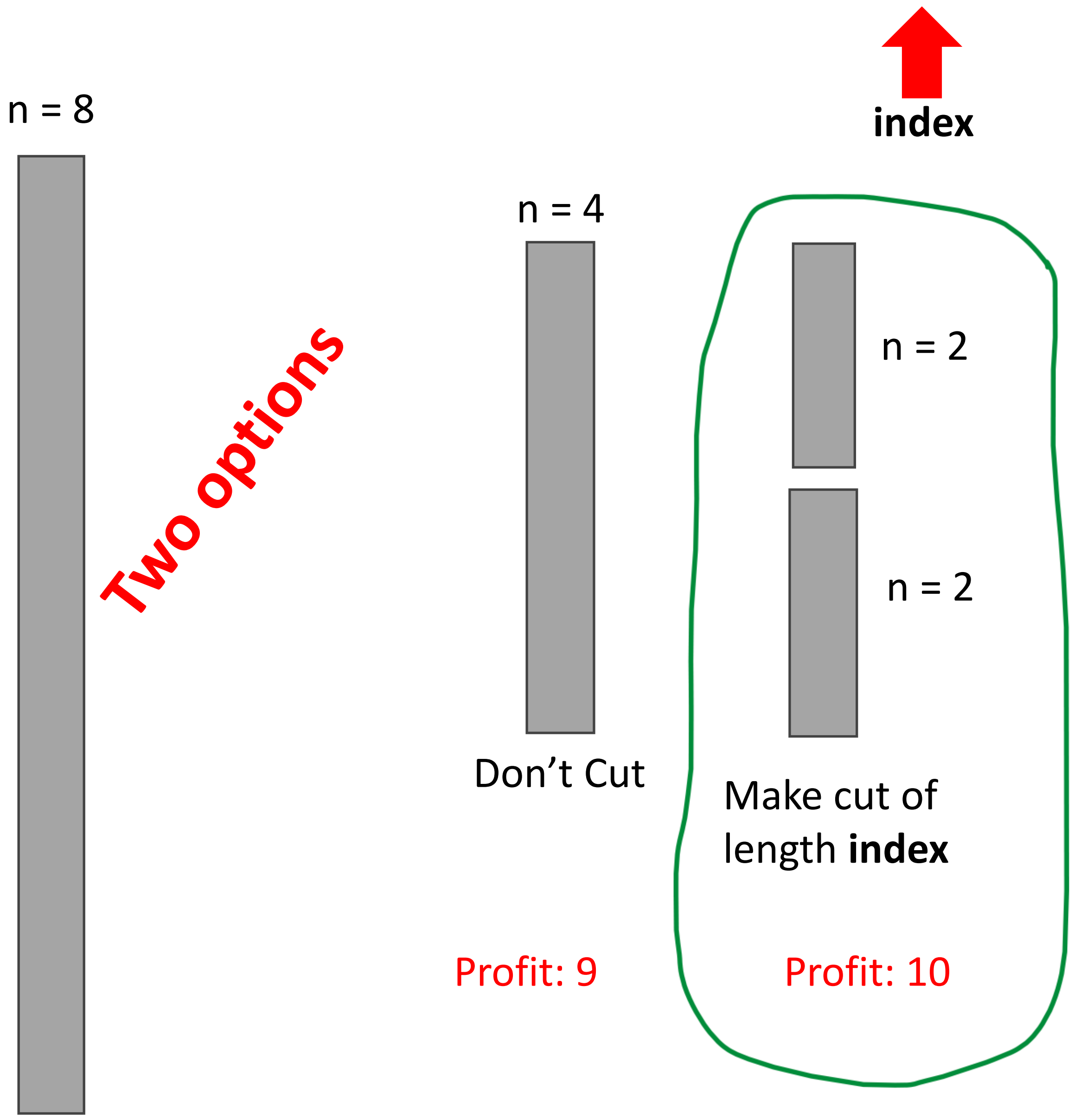


Given a rod of length 4 and a potential cut value of length 2, the optimal solution is to **make the cut**

If we ever encounter this same subproblem again, we want to make sure we don't recompute it

Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

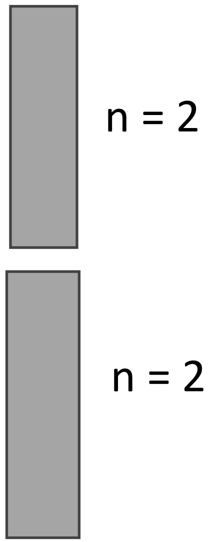


We need to put this solution (10) into our memorization table

Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

index



Make cut of length **index**

Profit: 10

Rod Length

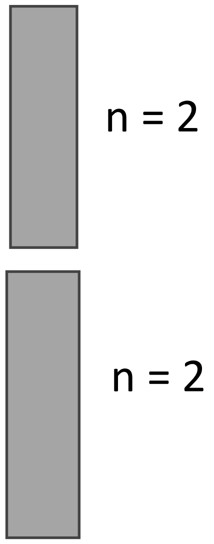
Cut Length

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1								
2								
3								
4								
5								
6								
7								
8								

Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

index



Make cut of length **index**

Profit: 10

Rod Length

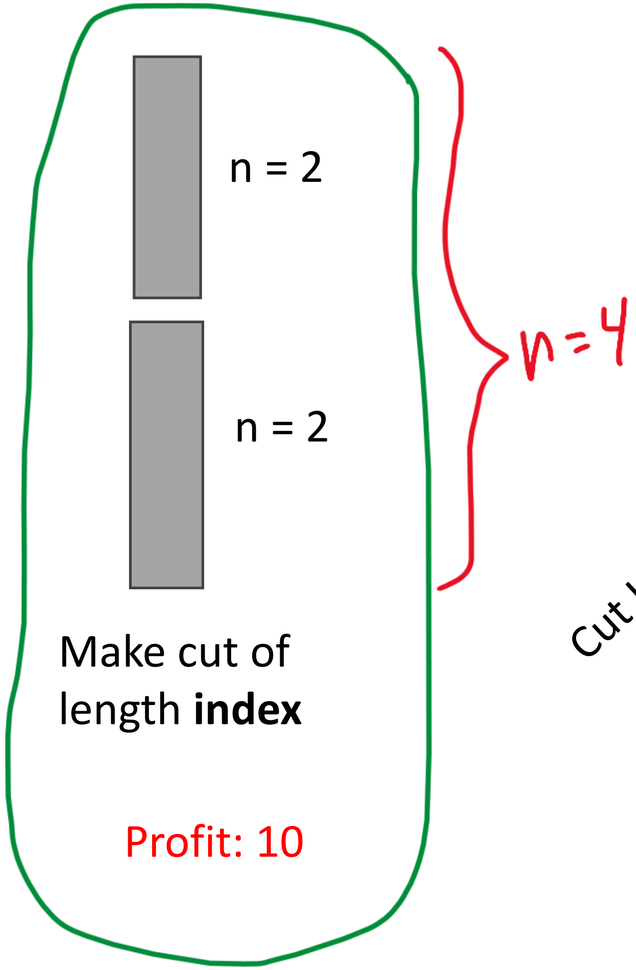
Cut Length

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1								
2								
3								
4								
5								
6								
7								
8								

Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

index



`dp[index][n] = 10`

Rod Length

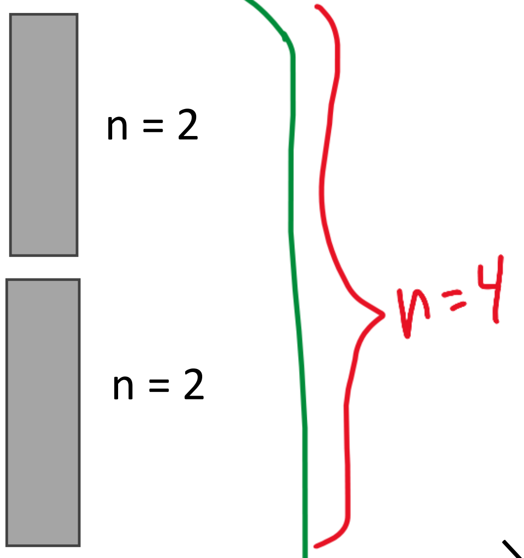
Cut Length

	1	2	3	4	5	6	7	8
1								
2				10				
3								
4								
5								
6								
7								
8								

Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

index



Make cut of length **index**

Profit: 10

dp[index][n] = 10

Rod Length

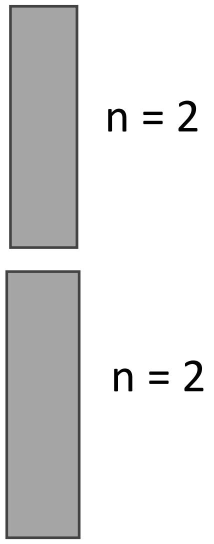
Cut Length

	0	1	2	3	4	5	6	7	8
0									
1					10				
2									
3									
4									
5									
6									
7									

Rod Cutting

	0	1	2	3	4	5	6	7
Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

index



Make cut of length **index**

Profit: 10

```
dp[index][n] = 10
```

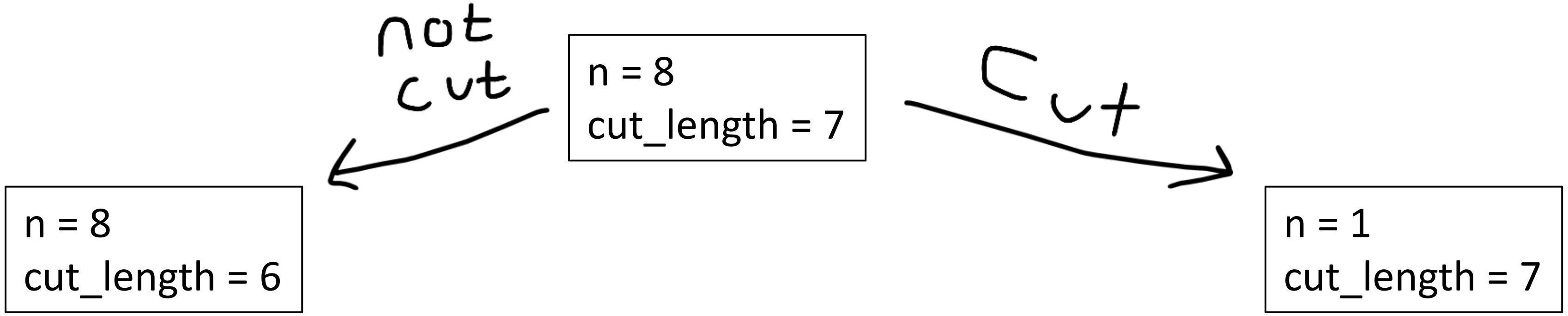
Rod Length

Cut Length

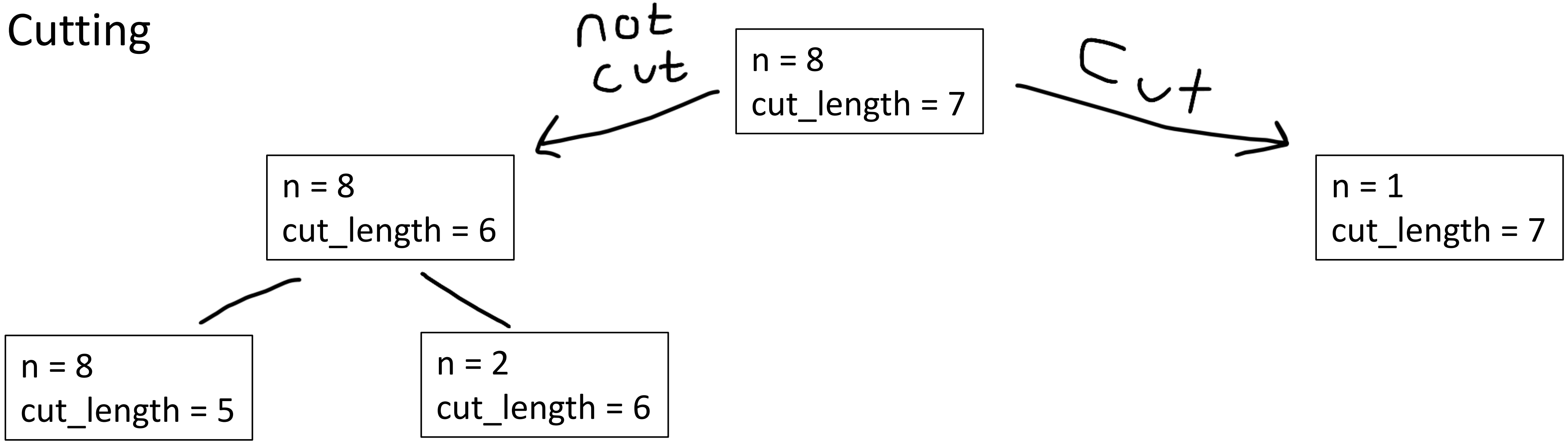
	1	2	3	4	5	6	7	8
1								
2				10				
3								
4								
5								
6								
7								
8								

Whenever we solve a subproblem, remember to place it inside of our memoization table

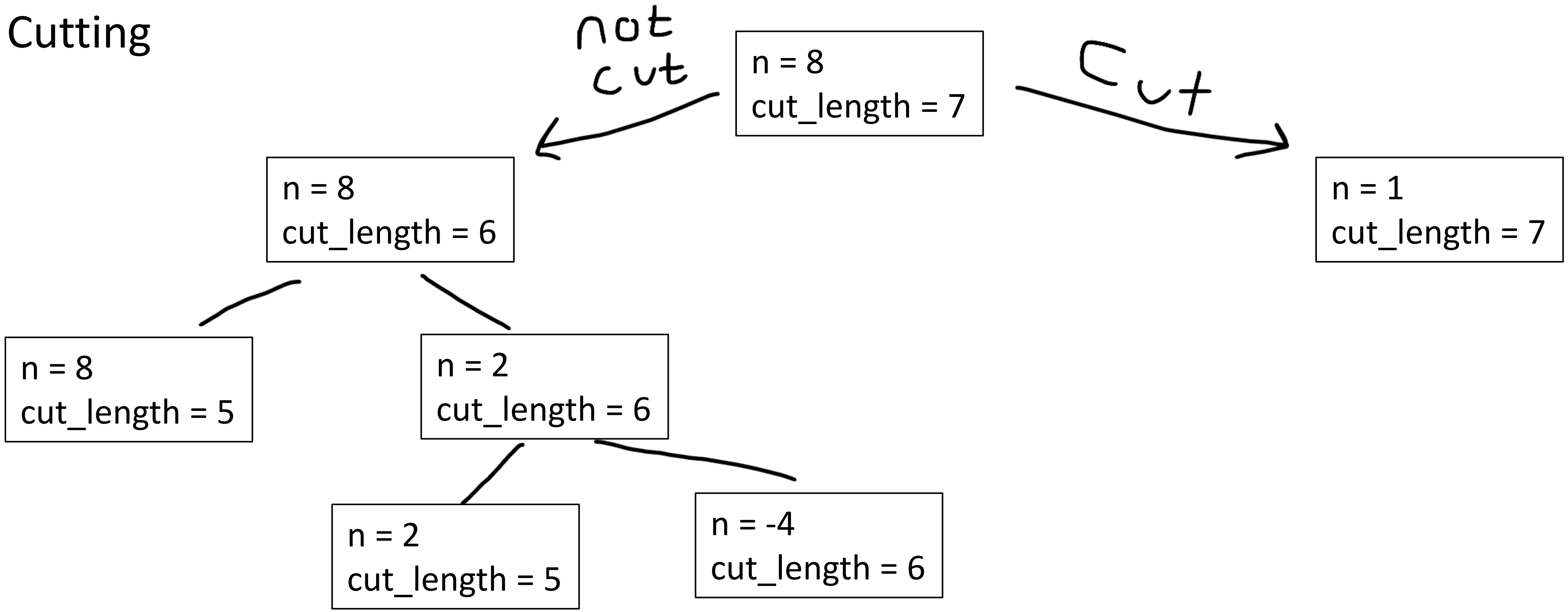
Rod Cutting



Rod Cutting

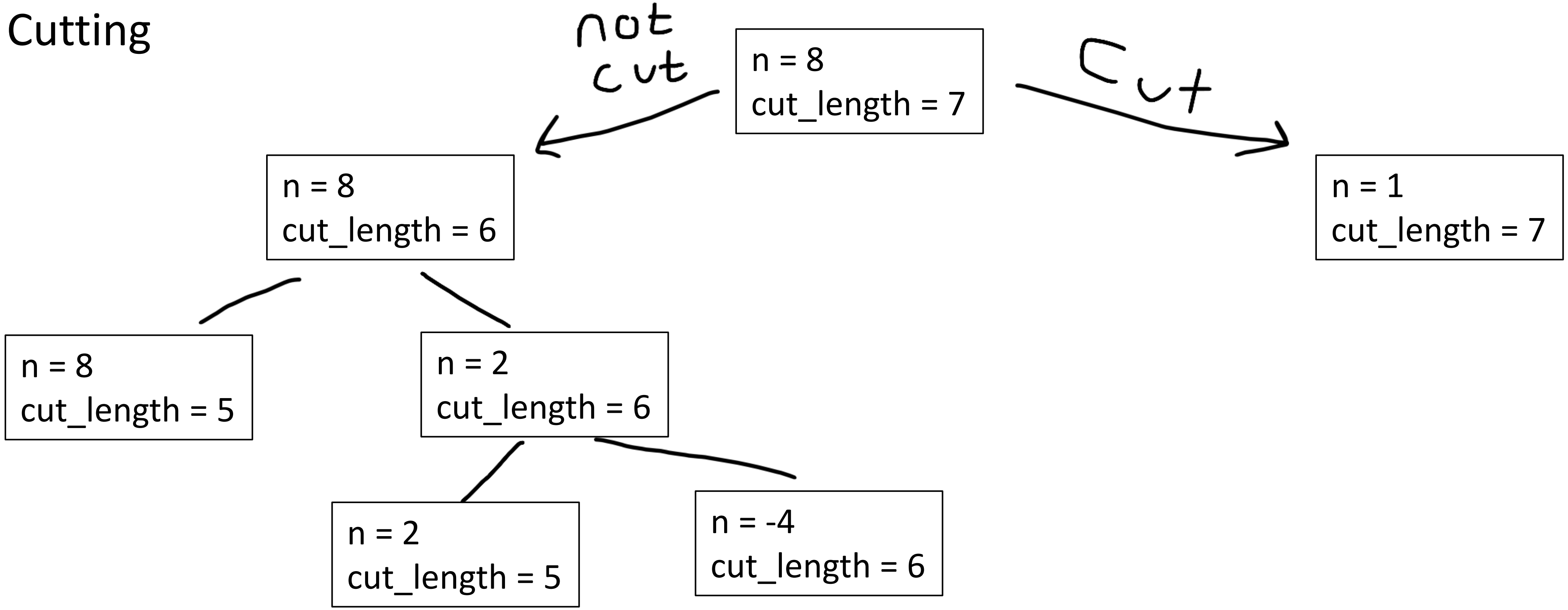

n = 8
cut_length = 7
not cut
Cut
n = 8
cut_length = 6
n = 1
cut_length = 7
n = 8
cut_length = 5
n = 2
cut_length = 6

Rod Cutting

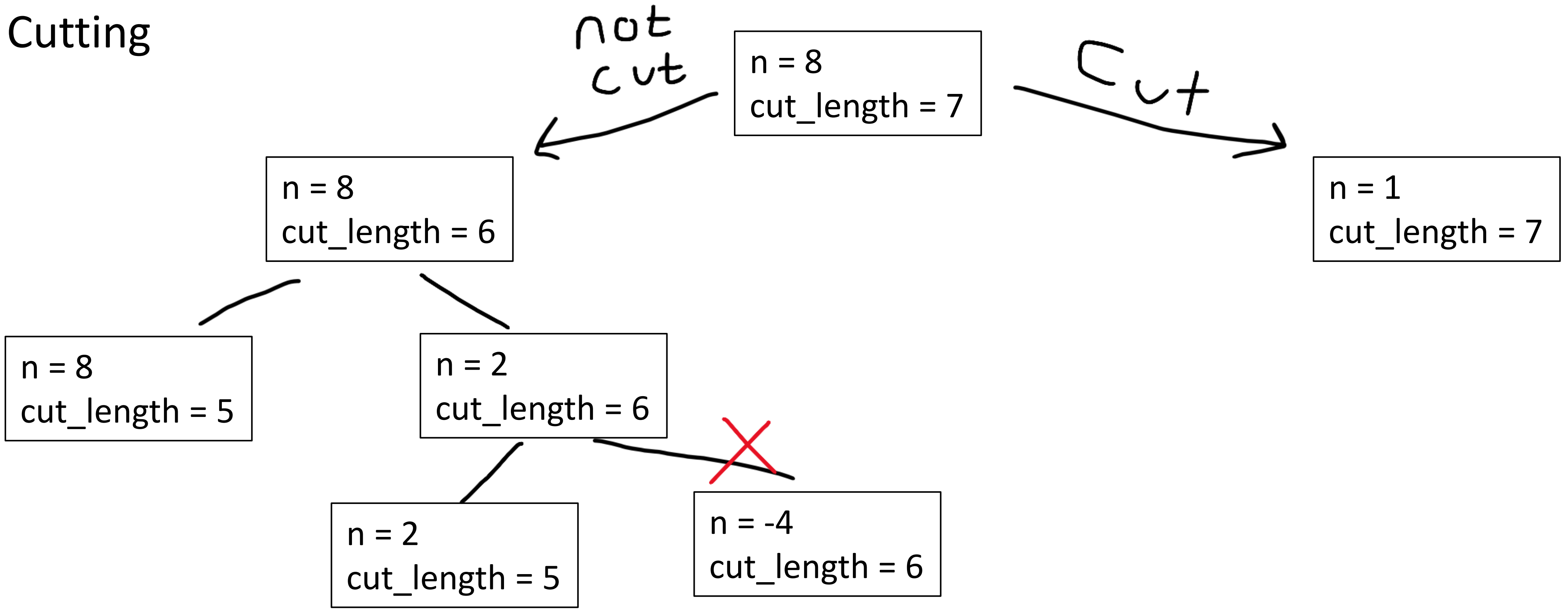


not cut
Cut
n = 8
cut_length = 7
n = 8
cut_length = 6
n = 1
cut_length = 7
n = 8
cut_length = 5
n = 2
cut_length = 6
n = 2
cut_length = 5
n = -4
cut_length = 6

Rod Cutting

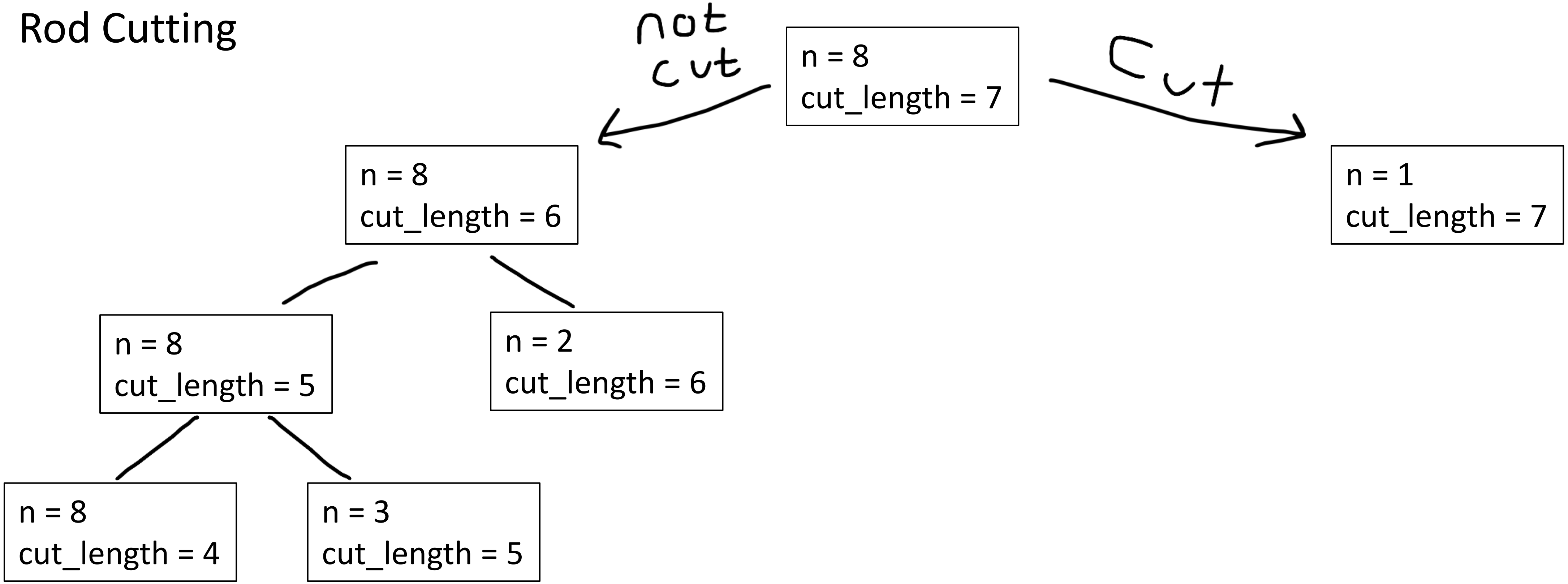

not cut
n = 8
cut_length = 7
Cut
n = 8
cut_length = 6
n = 1
cut_length = 7
n = 8
cut_length = 5
n = 2
cut_length = 6
n = 2
cut_length = 5
n = -4
cut_length = 6

Rod Cutting



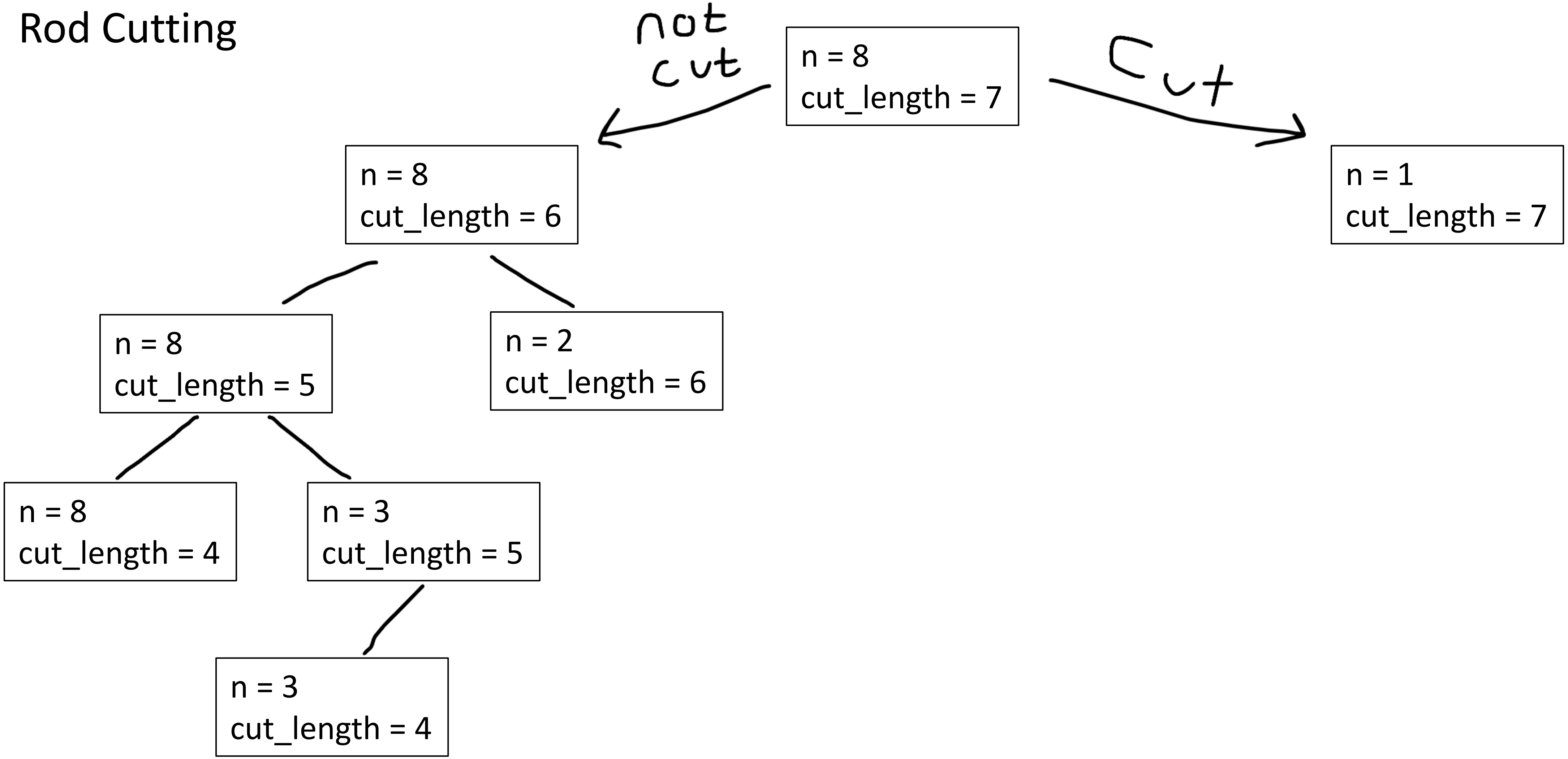
Only make the cut if its possible

Rod Cutting

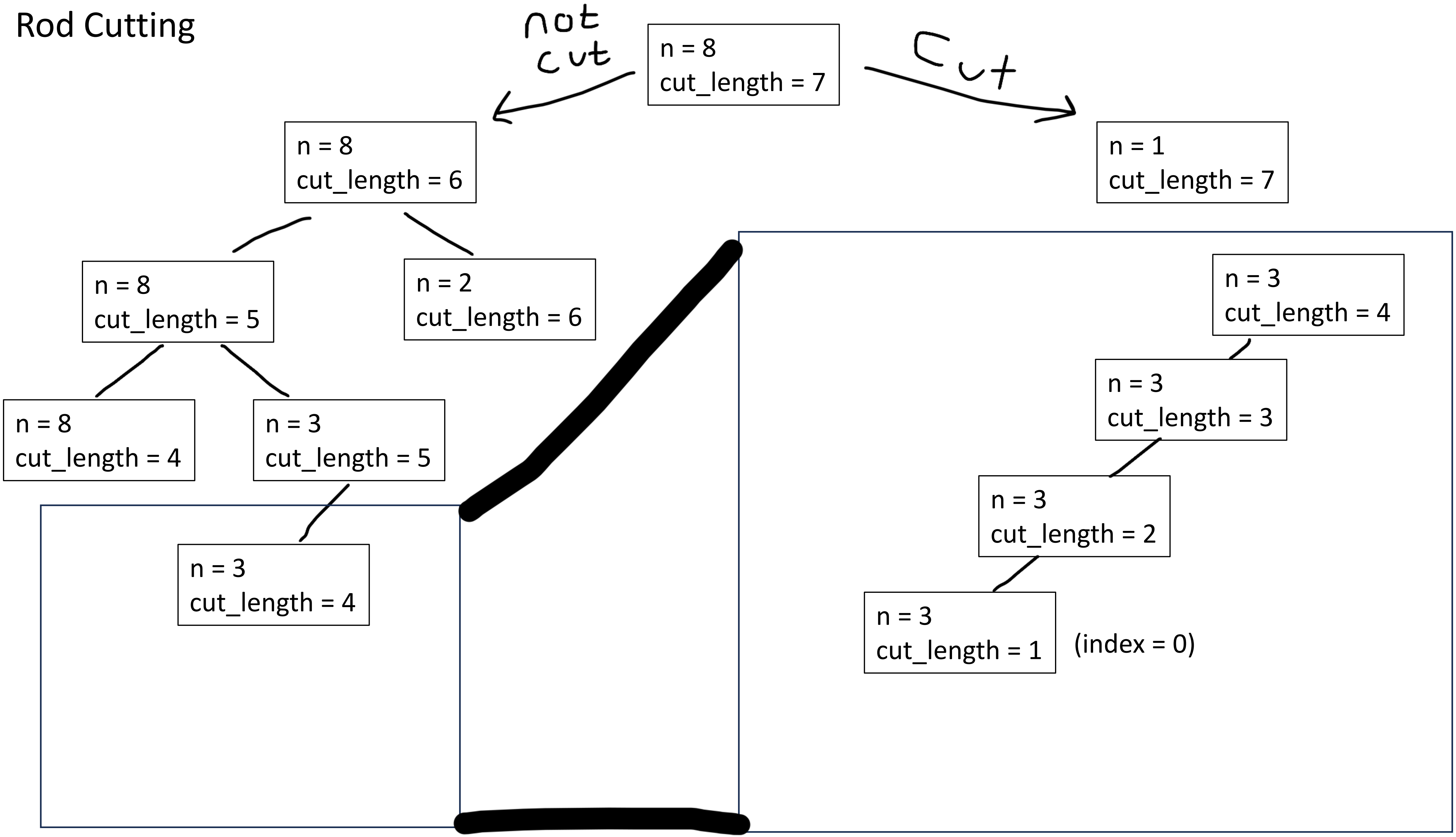


n = 8
cut_length = 7
not cut
Cut
n = 8
cut_length = 6
n = 1
cut_length = 7
n = 8
cut_length = 5
n = 2
cut_length = 6
n = 8
cut_length = 4
n = 3
cut_length = 5

Rod Cutting

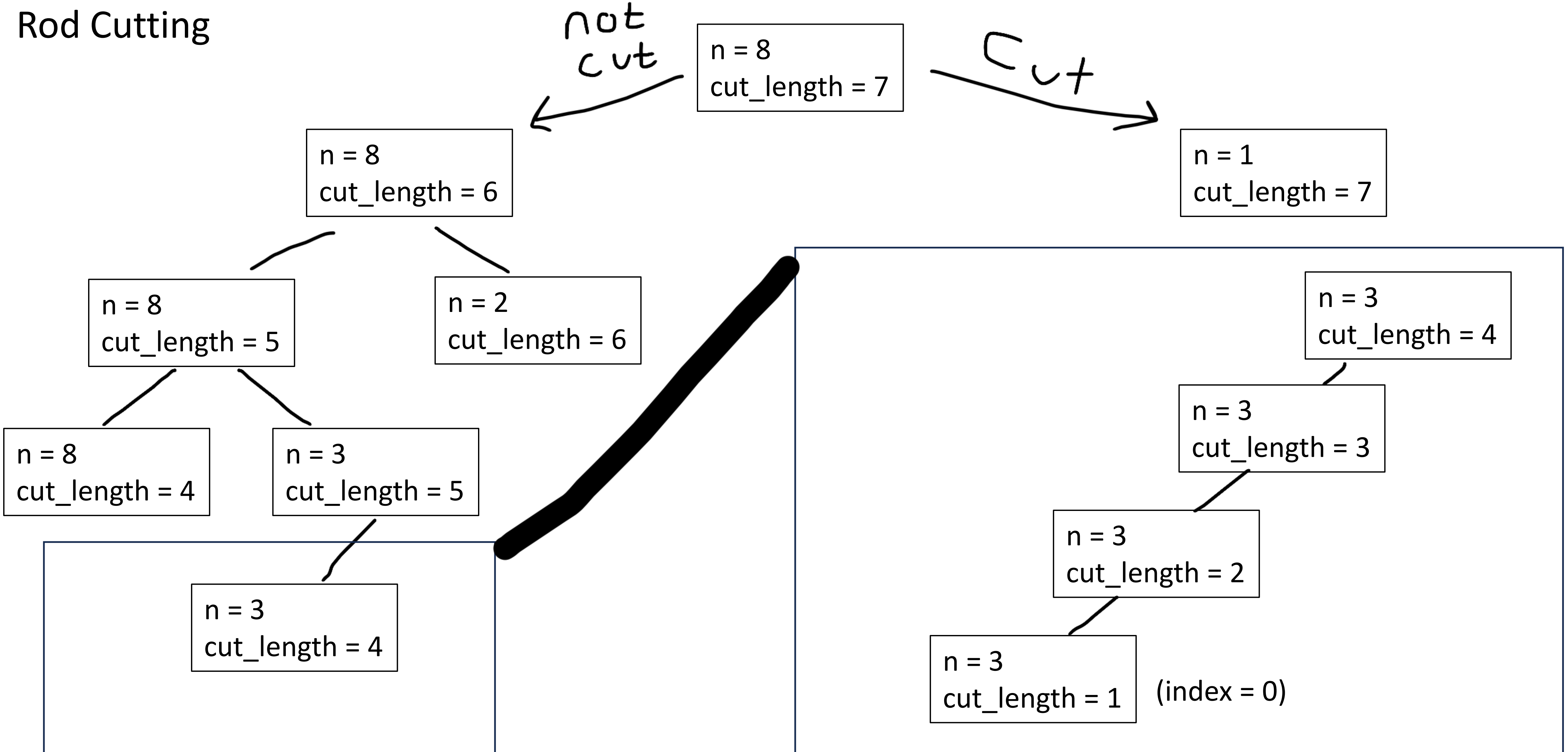

not cut
Cut
n = 8
cut_length = 7
n = 8
cut_length = 6
n = 1
cut_length = 7
n = 8
cut_length = 5
n = 2
cut_length = 6
n = 8
cut_length = 4
n = 3
cut_length = 5
n = 3
cut_length = 4

Rod Cutting



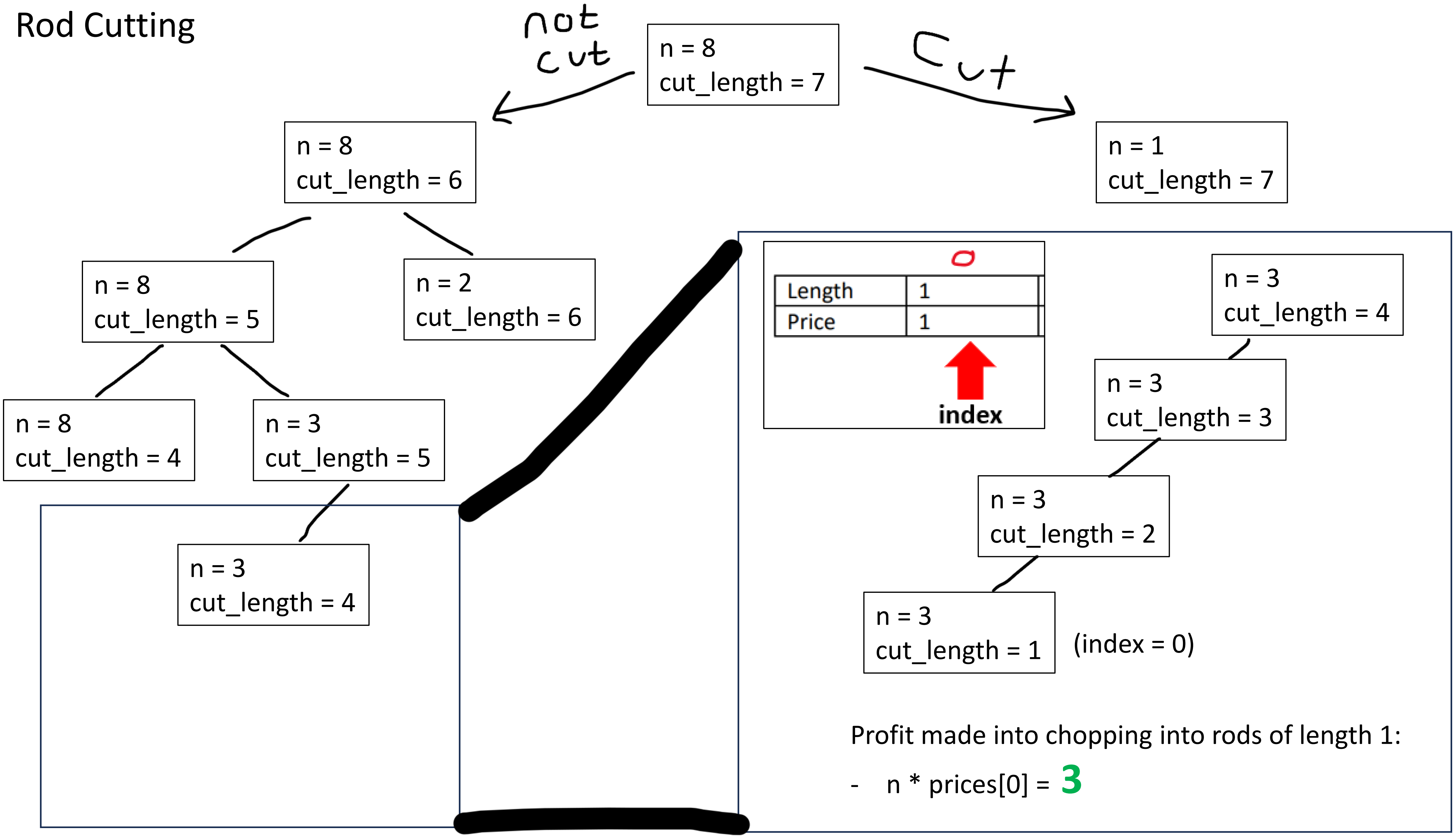
not cut
Cut
n = 8
cut_length = 7
n = 8
cut_length = 6
n = 1
cut_length = 7
n = 8
cut_length = 5
n = 2
cut_length = 6
n = 3
cut_length = 4
n = 8
cut_length = 4
n = 3
cut_length = 5
n = 3
cut_length = 3
n = 3
cut_length = 4
n = 3
cut_length = 2
n = 3
cut_length = 1
(index = 0)

Rod Cutting



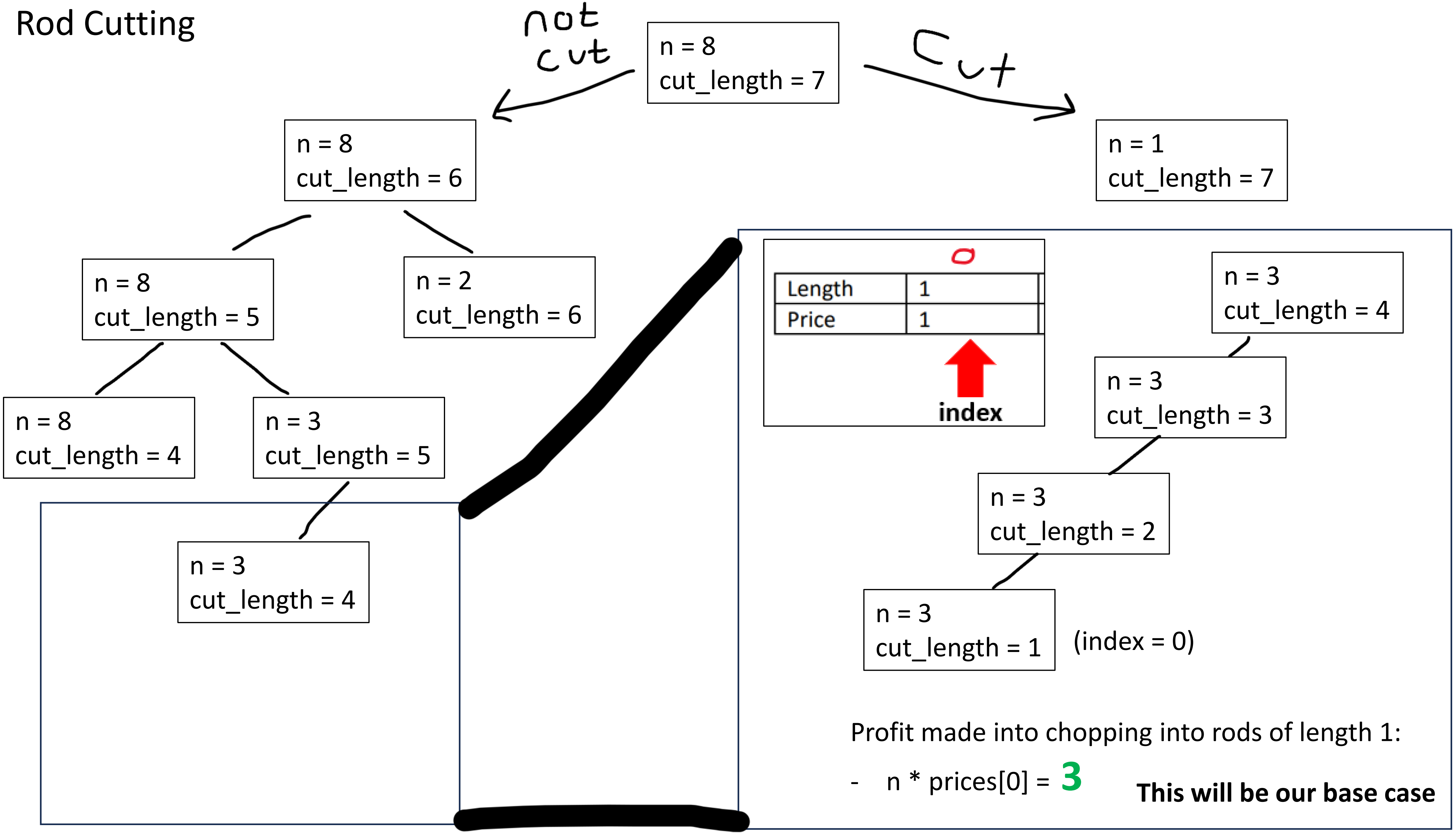
We can’t chop into lengths less than 1, so we can compute a solution right here

Rod Cutting



not cut
Cut
n = 8
cut_length = 7
n = 8
cut_length = 6
n = 1
cut_length = 7
n = 8
cut_length = 5
n = 2
cut_length = 6
n = 8
cut_length = 4
n = 3
cut_length = 5
n = 3
cut_length = 4
0
Length 1
Price 1
index
n = 3
cut_length = 4
n = 3
cut_length = 3
n = 3
cut_length = 2
n = 3
cut_length = 1
(index = 0)
Profit made into chopping into rods of length 1:
- n * prices[0] = 3

Rod Cutting



Length	1
Price	1

Profit made into chopping into rods of length 1:

- n * prices[0] = **3**

This will be our base case

Rod

n = 8
cut_l

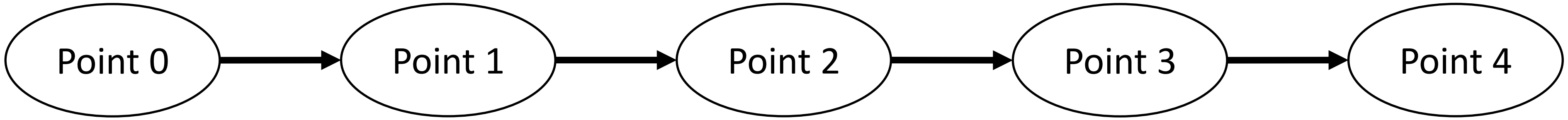
LETS TRY TO CODE THIS

If you are confused are the recursion is set up, don’t stress out about it. Its not a big deal.

The goal here is to show how we are using dynamic programming to solve this problem

Taxi Profit

Given a street that goes from point 1 to point N, we are able to pick up customers on the street and take them to the other end of the street. Our taxi is only able to go one direction



Taxi Profit

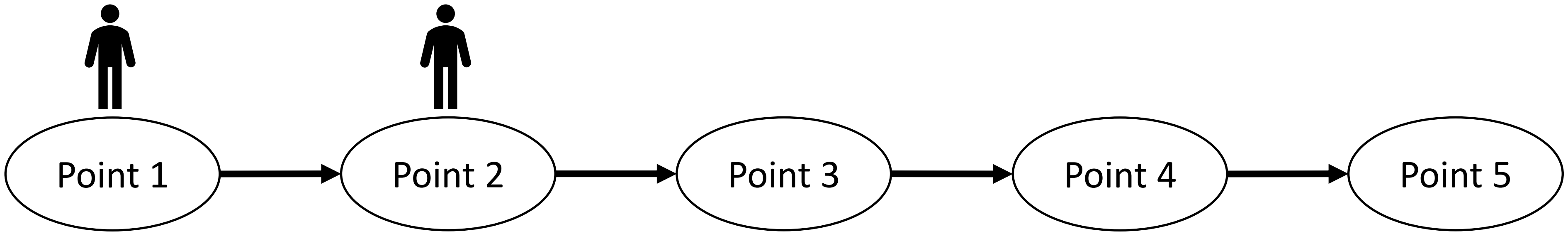
Given a street that goes from point 1 to point N, we are able to pick up customers on the street and take them to the other end of the street. Our taxi is only able to go one direction

Each customer has their starting point (pick-up spot), their ending point (destination), and the tip you will receive if you pick them up

The profit for selecting a customer is (END – START) + TIP

Start: Point 1
End: Point 5
Tip: $1

Start: Point 2
End: Point 5
Tip: $4

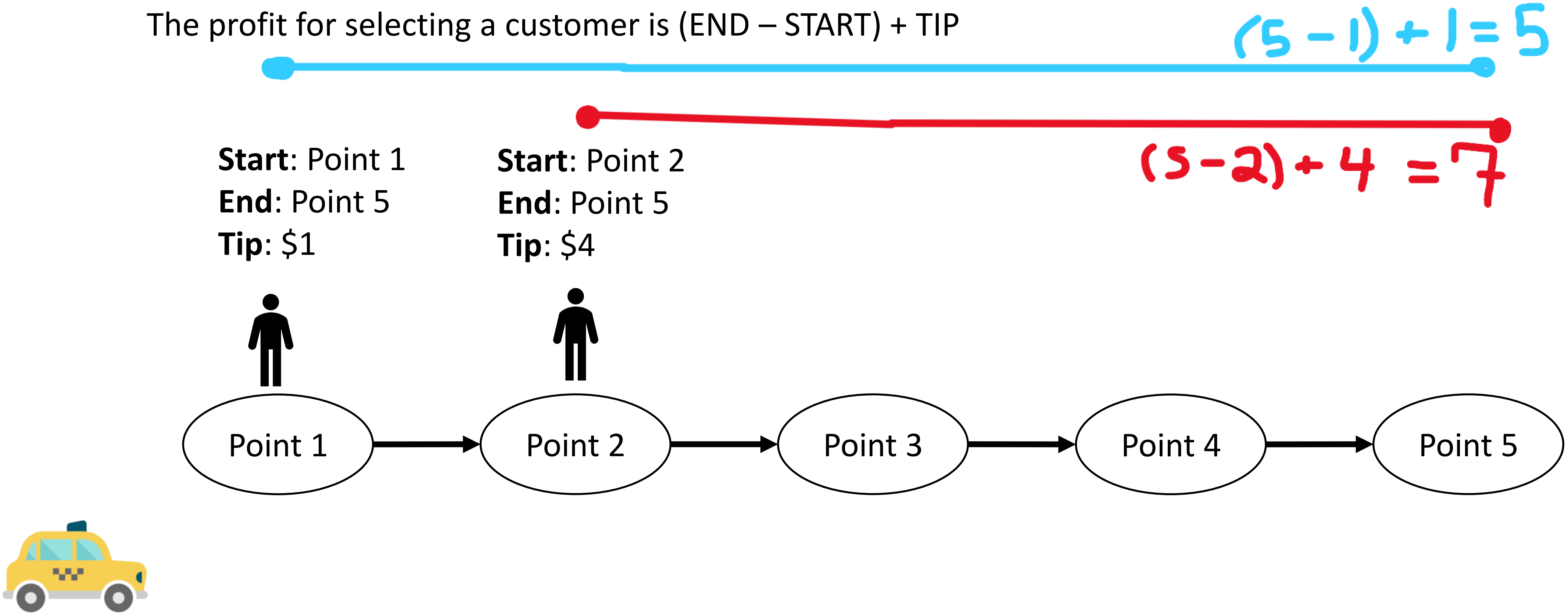


Taxi Profit

Given a street that goes from point 1 to point N, we are able to pick up customers on the street and take them to the other end of the street. Our taxi is only able to go one direction

Each customer has their starting point (pick-up spot), their ending point (destination), and the tip you will receive if you pick them up

The profit for selecting a customer is (END – START) + TIP



Taxi Profit

Given a street that goes from point 1 to point N, we are able to pick up customers on the street and take them to the other end of the street. Our taxi is only able to go one direction

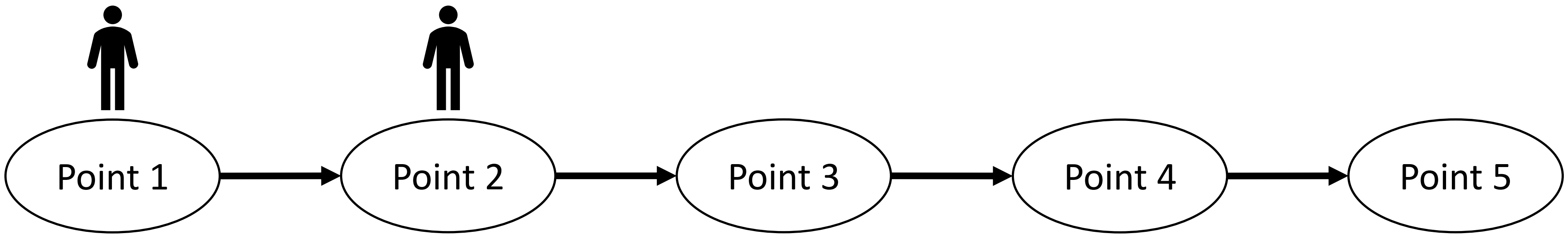
Each customer has their starting point (pick-up spot), their ending point (destination), and the tip you will receive if you pick them up

The profit for selecting a customer is (END – START) + TIP

```
rides = [[1, 5, 1]     [2, 5, 4]                    ]
```

Start: Point 1
End: Point 5
Tip: $1

Start: Point 2
End: Point 5
Tip: $4



What is the **maximum profit** that the taxi can make when going from point 1 to point N?

Taxi Profit

Given a street that goes from point 1 to point N, we are able to pick up customers on the street and take them to the other end of the street. Our taxi is only able to go one direction

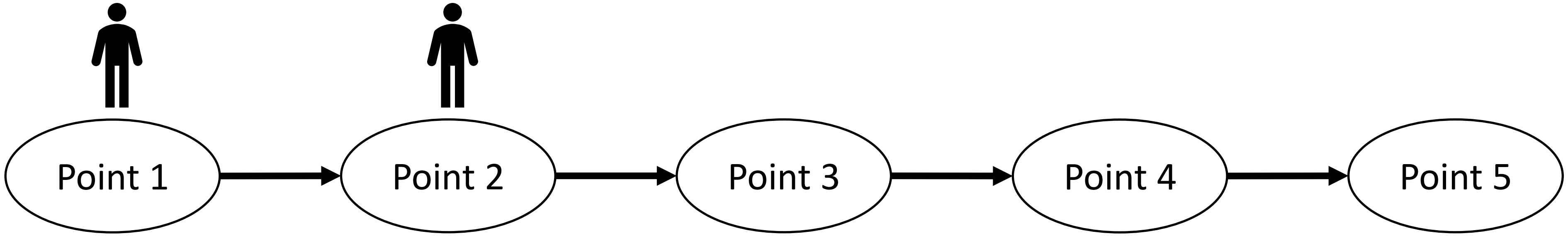
Each customer has their starting point (pick-up spot), their ending point (destination), and the tip you will receive if you pick them up

The profit for selecting a customer is (END – START) + TIP

```
rides = [[1, 5, 1]      [2, 5, 4]                    ]
```

[1, 5, 1]	[2, 5, 4]
Start: Point 1	**Start**: Point 2
End: Point 5	**End**: Point 5
Tip: $1	**Tip**: $4

Is there another way we could represent this problem?

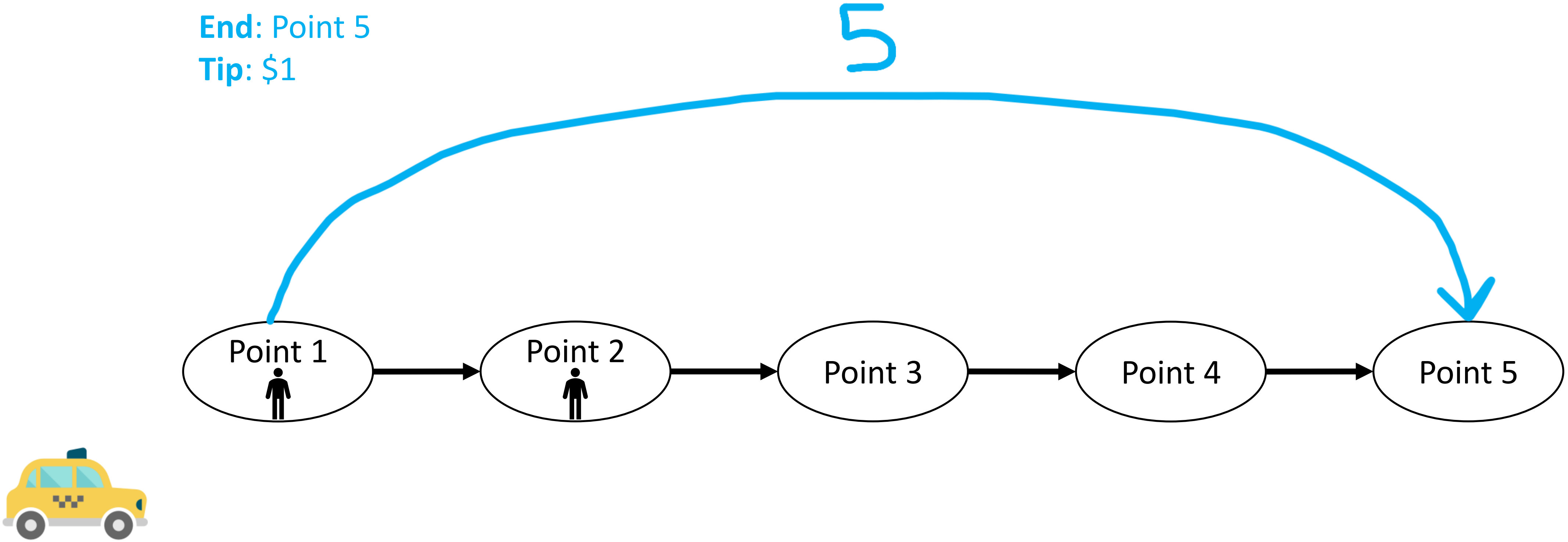


What is the **maximum profit** that the taxi can make when going from point 1 to point N?

Taxi Profit (Graph Representation)

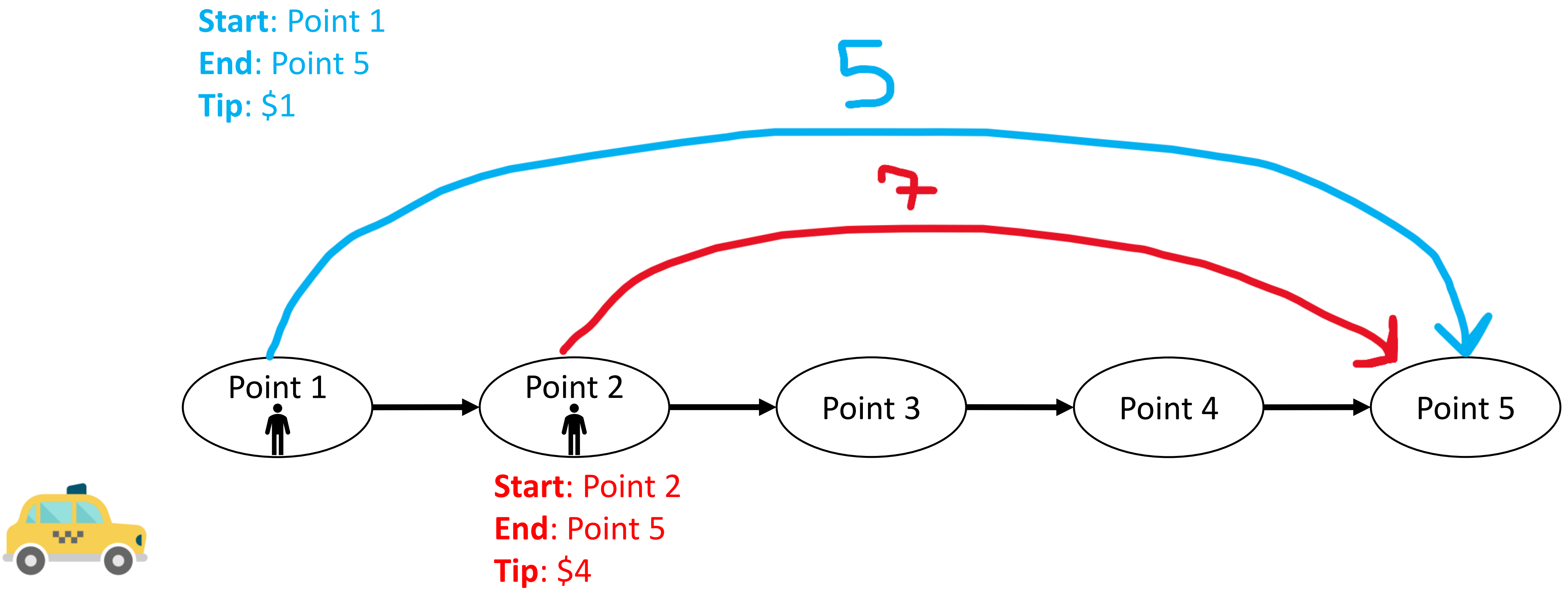
What is the **maximum profit** that the taxi can make when going from point 1 to point N?

Start: Point 1
End: Point 5
Tip: $1



Taxi Profit (Graph Representation)

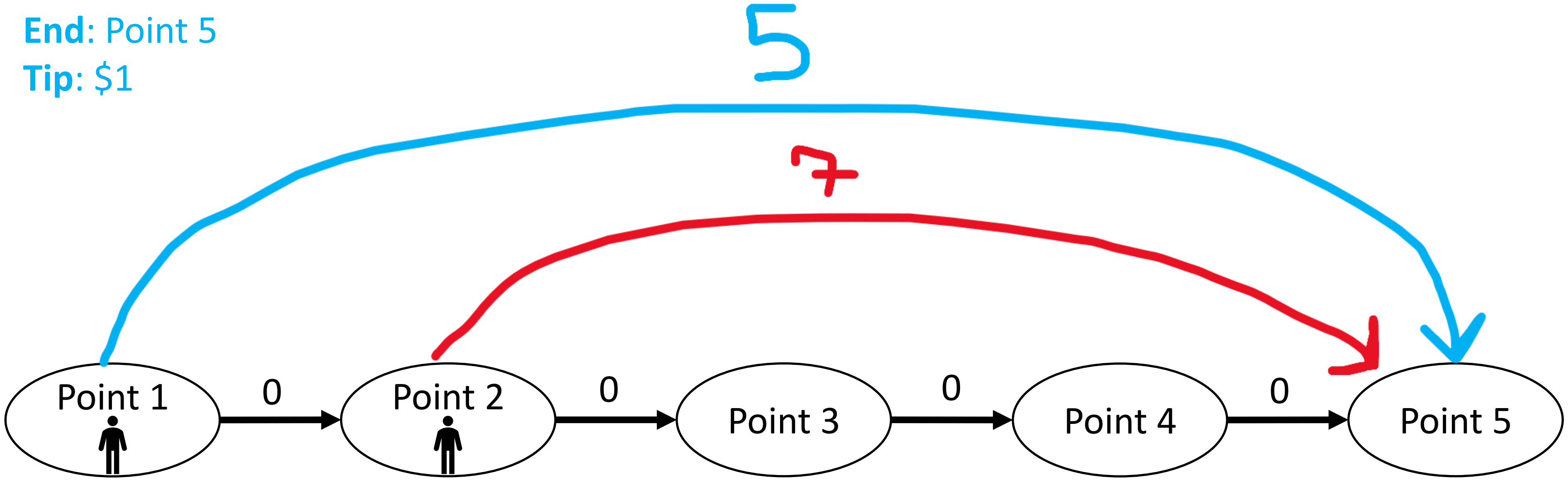
What is the **maximum profit** that the taxi can make when going from point 1 to point N?



Taxi Profit (Graph Representation)

What is the **maximum profit** that the taxi can make when going from point 1 to point N?

Start: Point 1
End: Point 5
Tip: $1



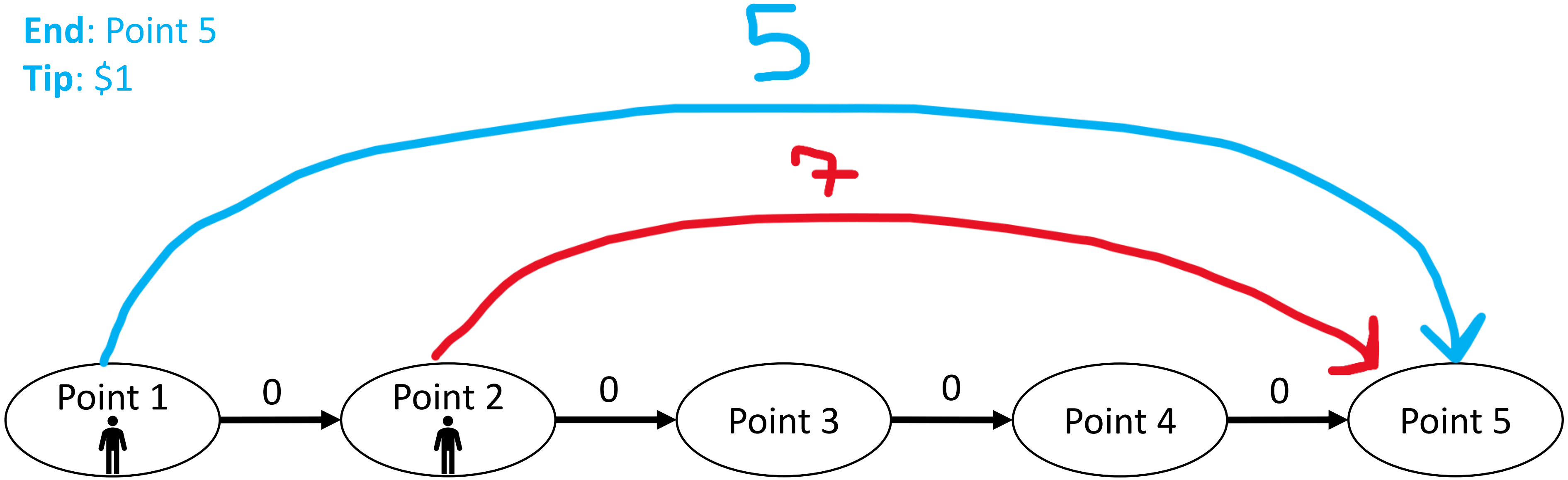
Start: Point 2
End: Point 5
Tip: $4

Taxi Profit (Graph Representation)

What is the *longest* path from point 1 to point 5?

Directed Acyclic Graph (DAG)- a directed graph that contains no cycles

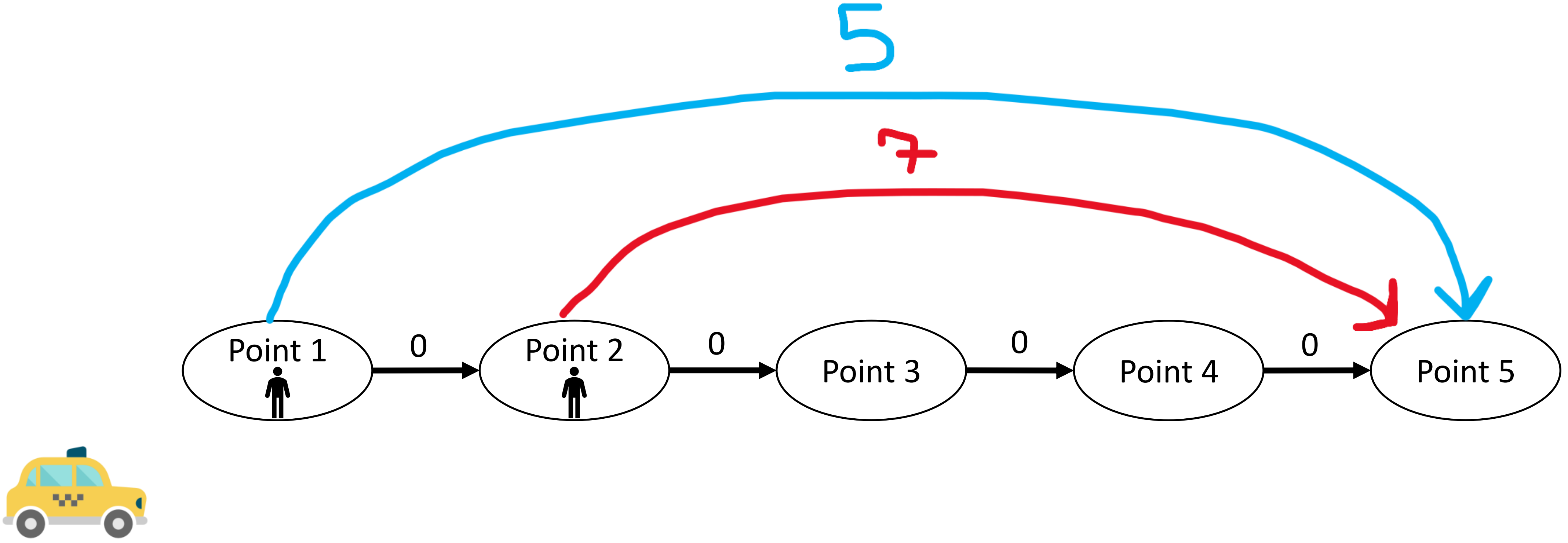
Start: Point 1
End: Point 5
Tip: $1



Start: Point 2
End: Point 5
Tip: $4

Taxi Profit (Graph Representation)

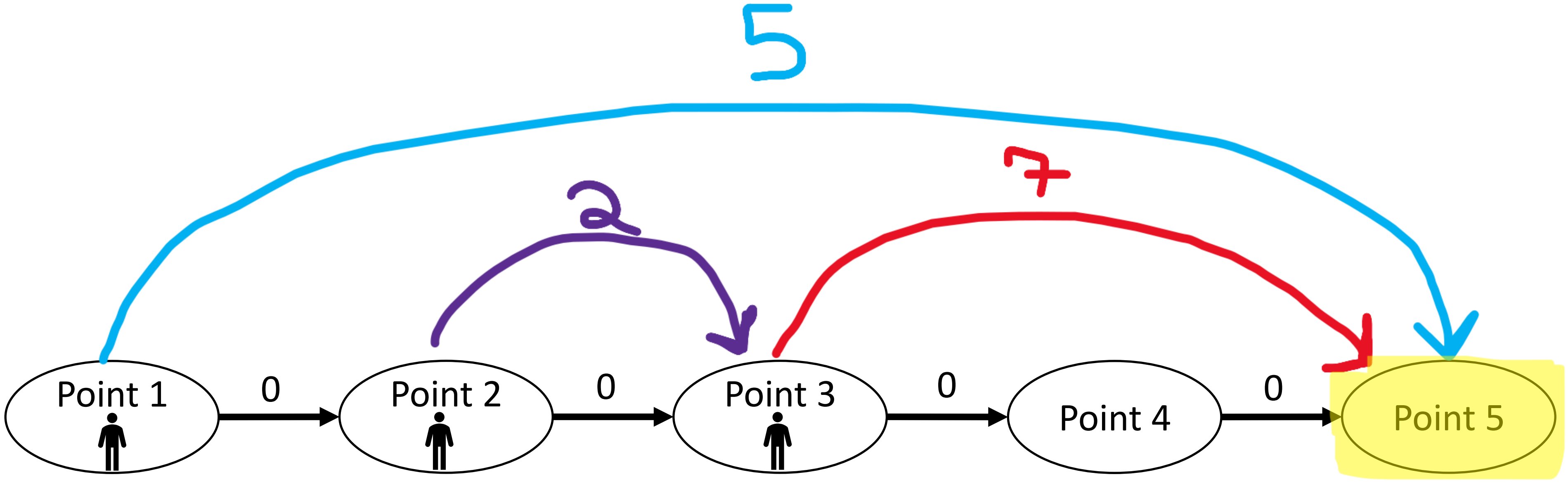
What is the **maximum profit** that the taxi can make when going from point 1 to point N?



Taxi Profit (Graph Representation)

What is the **maximum profit** that the taxi can make when going from point 1 to point N?

Let B(x) be the maximum cost to go from point 1 to point x

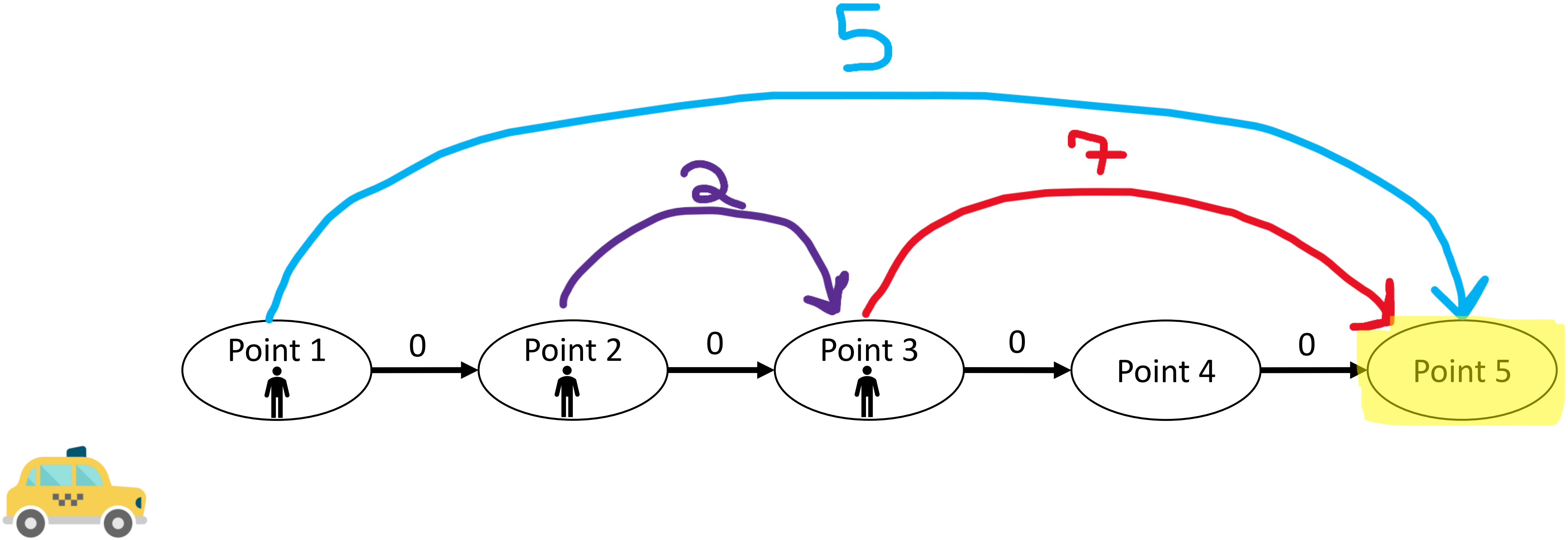


Taxi Profit (Graph Representation)

What is the **maximum profit** that the taxi can make when going from point 1 to point N?

Let B(x) be the maximum cost to go from point 1 to point x

Let P(i, x) be the profit made for a ride that goes from point 1 to point x



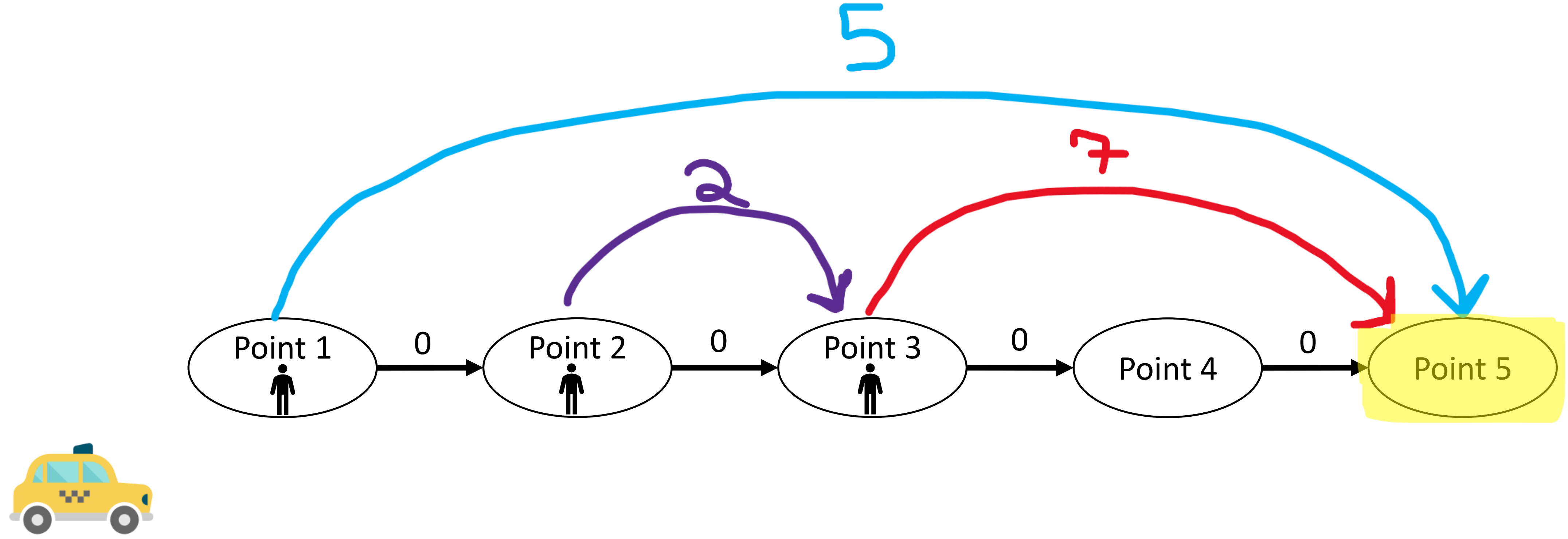
Taxi Profit (Graph Representation)

What is the **maximum profit** that the taxi can make when going from point 1 to point N?

Let B(x) be the maximum cost to go from point 1 to point x

Let P(i, x) be the profit made for a ride that goes from point 1 to point x

B(5) = ?



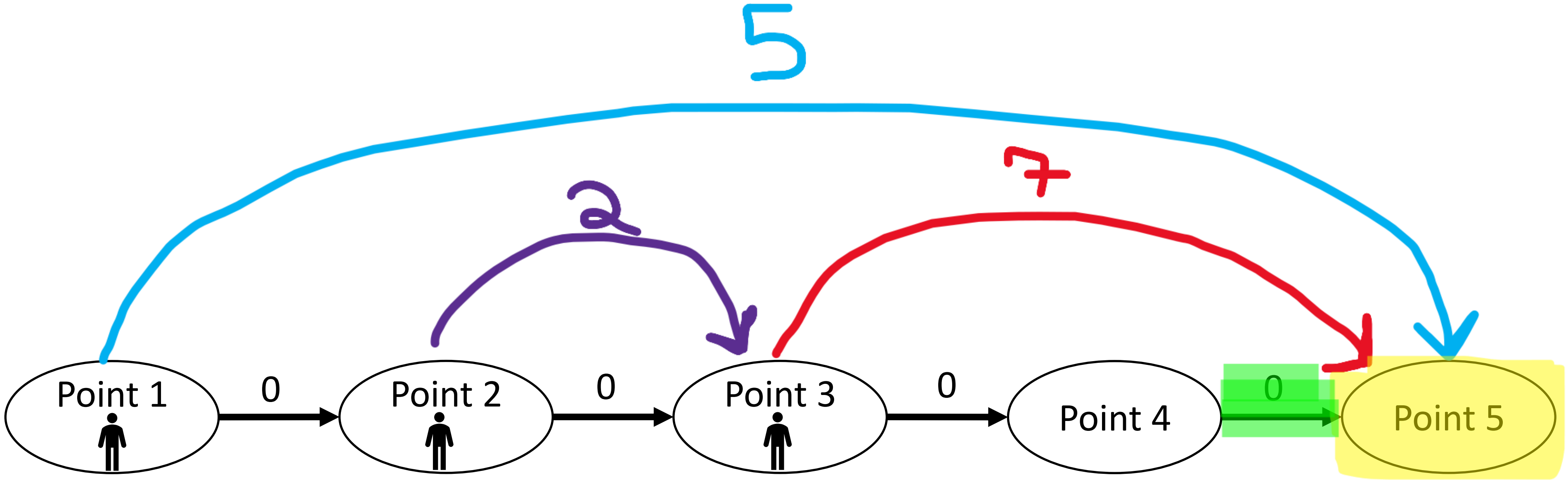
Taxi Profit (Graph Representation)

What is the **maximum profit** that the taxi can make when going from point 1 to point N?

Let B(x) be the maximum cost to go from point 1 to point x

Let P(i, x) be the profit made for a ride that goes from point 1 to point x

$B(5) = B(4) + 0$



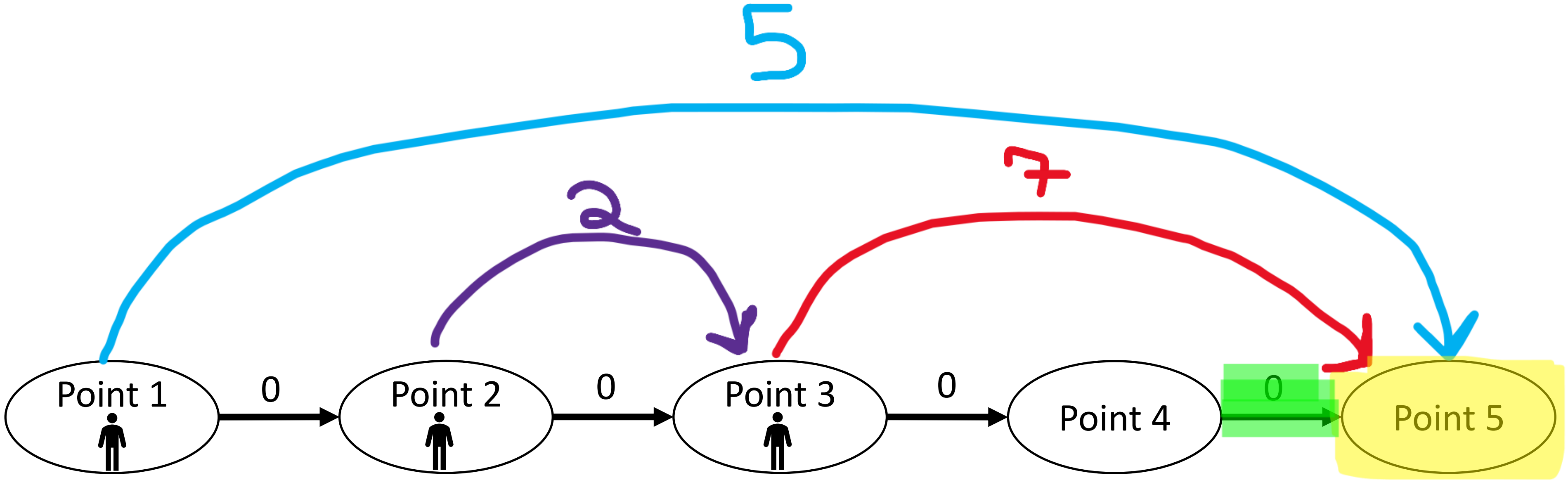
Taxi Profit (Graph Representation)

What is the **maximum profit** that the taxi can make when going from point 1 to point N?

Let B(x) be the maximum cost to go from point 1 to point x

Let P(i, x) be the profit made for a ride that goes from point 1 to point x

B(5) = B(4) + O or B(3) + 7 or B(1) + 5



Taxi Profit (Graph Representation)

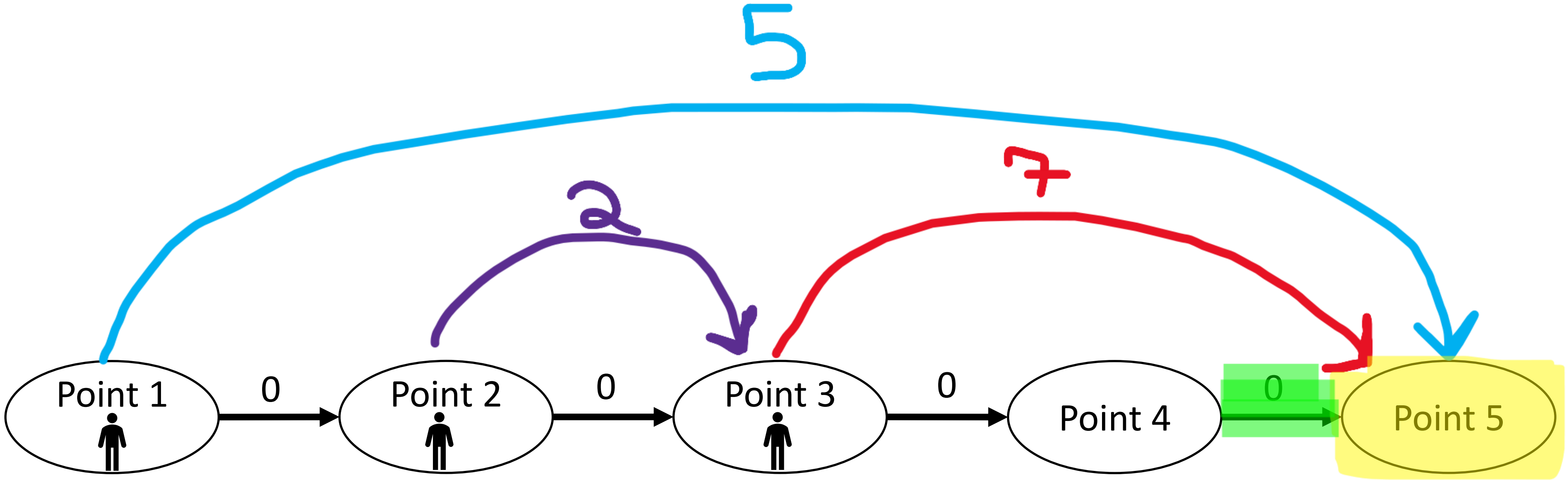
What is the **maximum profit** that the taxi can make when going from point 1 to point N?

Let B(x) be the maximum cost to go from point 1 to point x

Let P(i, x) be the profit made for a ride that goes from point 1 to point x

B(5) = B(4) + O or B(3) + 7 or B(1) + 5

Optimal Substructure



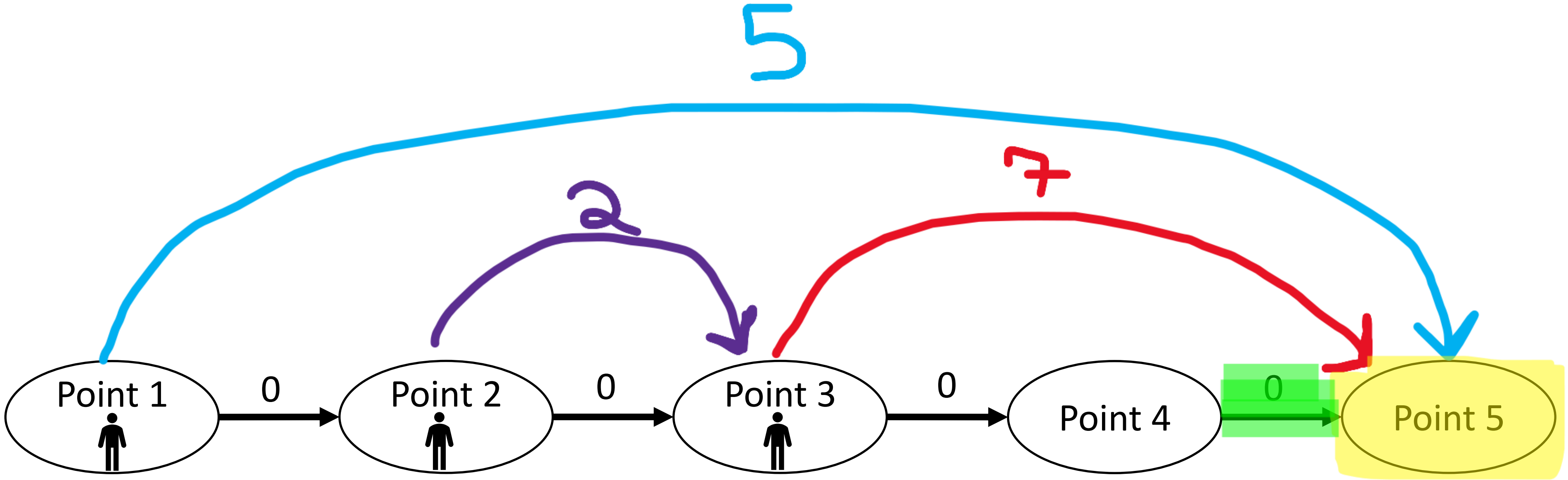
Taxi Profit (Graph Representation)

What is the **maximum profit** that the taxi can make when going from point 1 to point N?

Let B(x) be the maximum cost to go from point 1 to point x

Let P(i, x) be the profit made for a ride that goes from point 1 to point x

$B(5) = B(4) + 0$ or $B(3) + 7$ or $B(1) + 5$



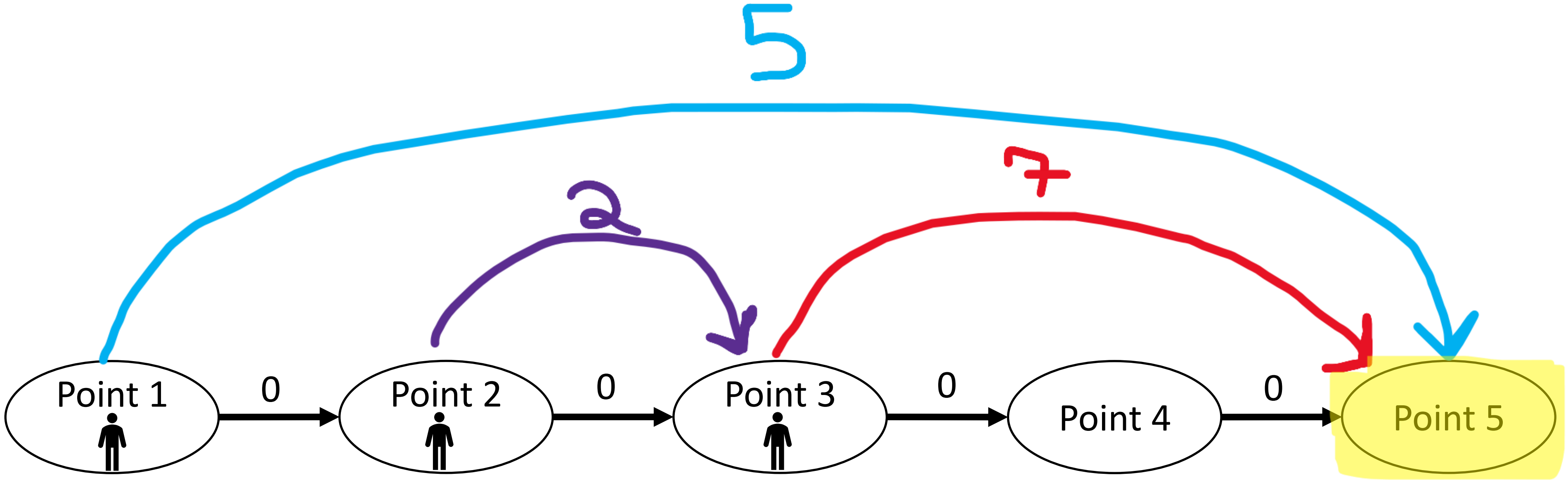
Taxi Profit (Graph Representation)

What is the **maximum profit** that the taxi can make when going from point 1 to point N?

Let B(x) be the maximum cost to go from point 1 to point x

Let P(i, x) be the profit made for a ride that goes from point 1 to point x

$$B(x) = \max \begin{cases} B(x-1) + 0 \\ B(x-i) + P(i,x) \\ \text{(for each edge that leads to point x)} \end{cases}$$



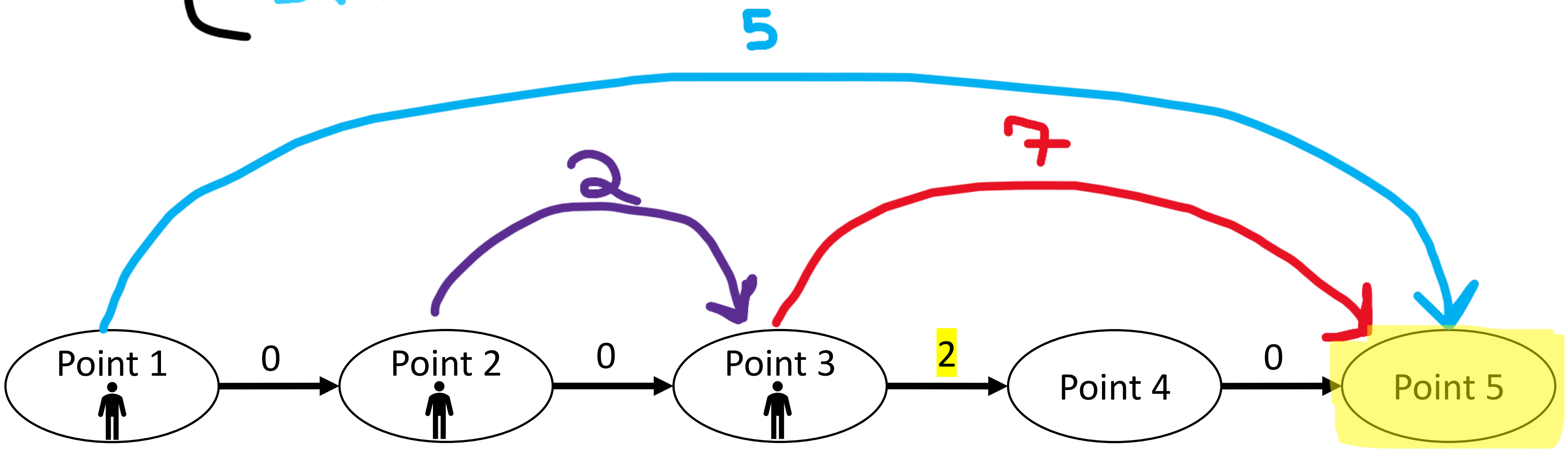
Taxi Profit (Graph Representation)

What is the **maximum profit** that the taxi can make when going from point 1 to point N?

Let B(x) be the maximum cost to go from point 1 to point x

Let P(i, x) be the profit made for a ride that goes from point 1 to point x

$$B(x) = \max \begin{cases} B(4) + 0 = 2 \\ B(3) + 7 = 2 + 7 = 9 \\ B(1) + 5 = 5 \end{cases}$$



Taxi Profit (Graph Representation)

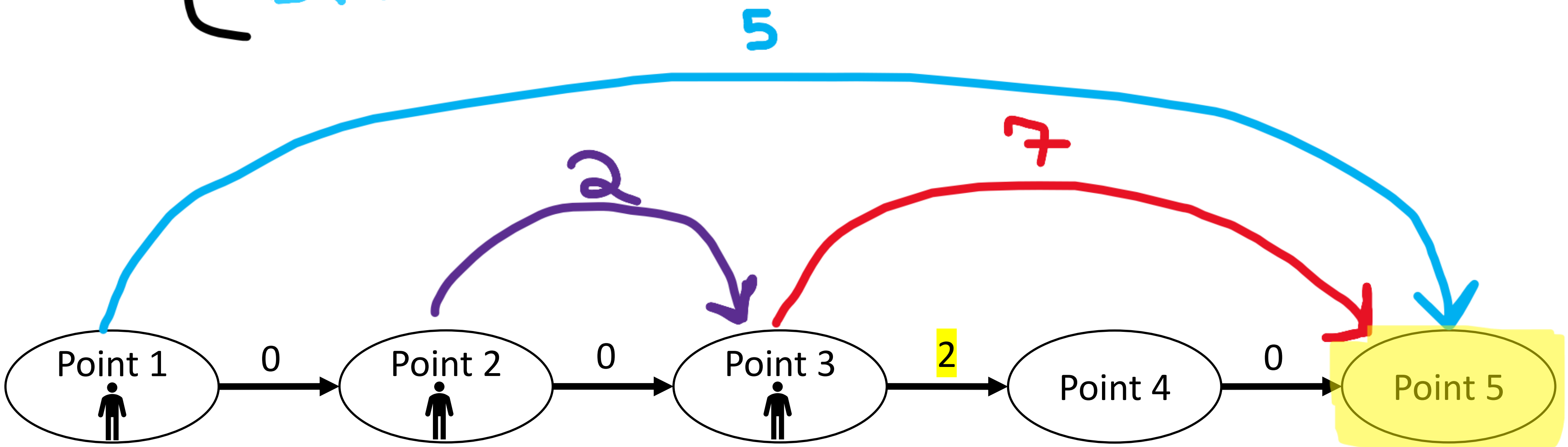
What is the **maximum profit** that the taxi can make when going from point 1 to point N?

Let B(x) be the maximum cost to go from point 1 to point x

Let P(i, x) be the profit made for a ride that goes from point 1 to point x

$$B(x) = \max \begin{cases} B(4) + 0 = 2 \\ B(3) + 7 = 2 + 7 = 9 \\ B(1) + 5 = 5 \end{cases}$$

Maximum profit = $9



Taxi Profit (Graph Representation)

What is the **maximum profit** that the taxi can make when going from point 1 to point N?

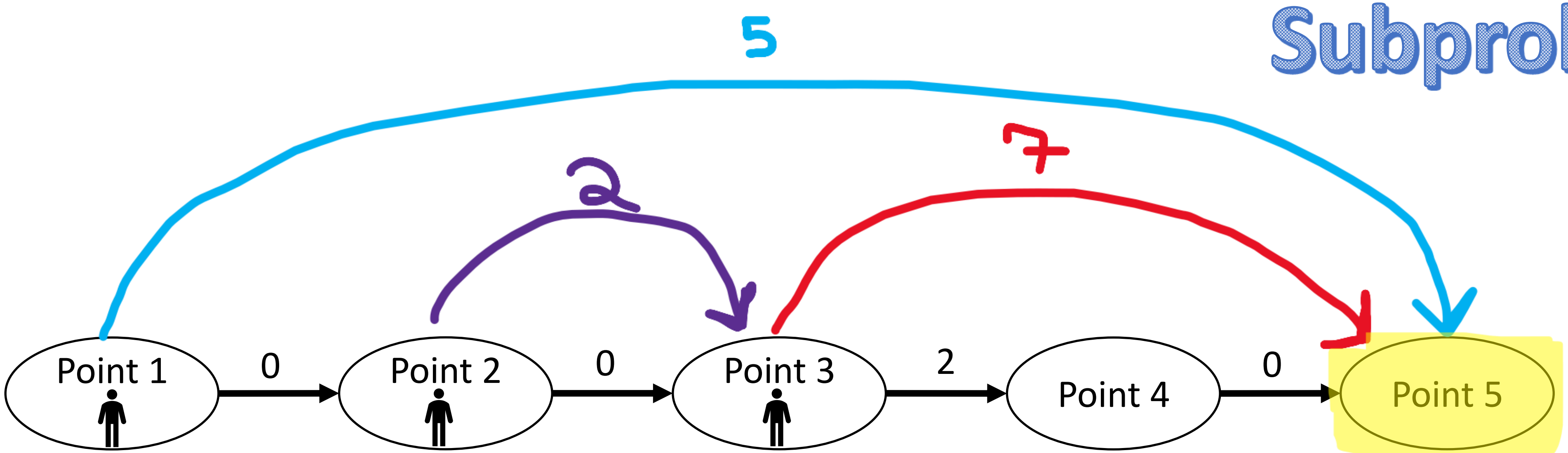
To solve maximum profit from 1 to 5, we will solve :

- 1 to 4
- 1 to 3
- 1 to 2

(This may include *several paths* by picking up customers along the way ie 2 to 3)

2 to 3 may be a subproblem we solve several times

Overlapping Subproblems



Taxi Profit (G

ake when going from point 1 to point N?

To
•
•
•

lve

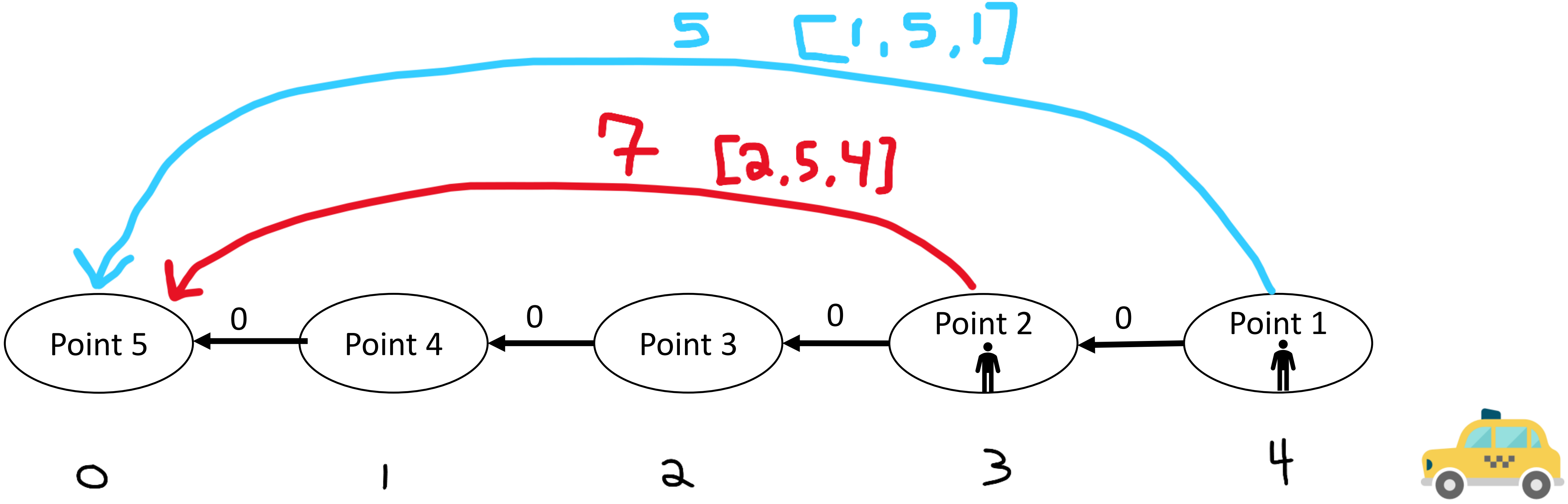
verlapping
bproblems

Poi

Point 5

Taxi Profit

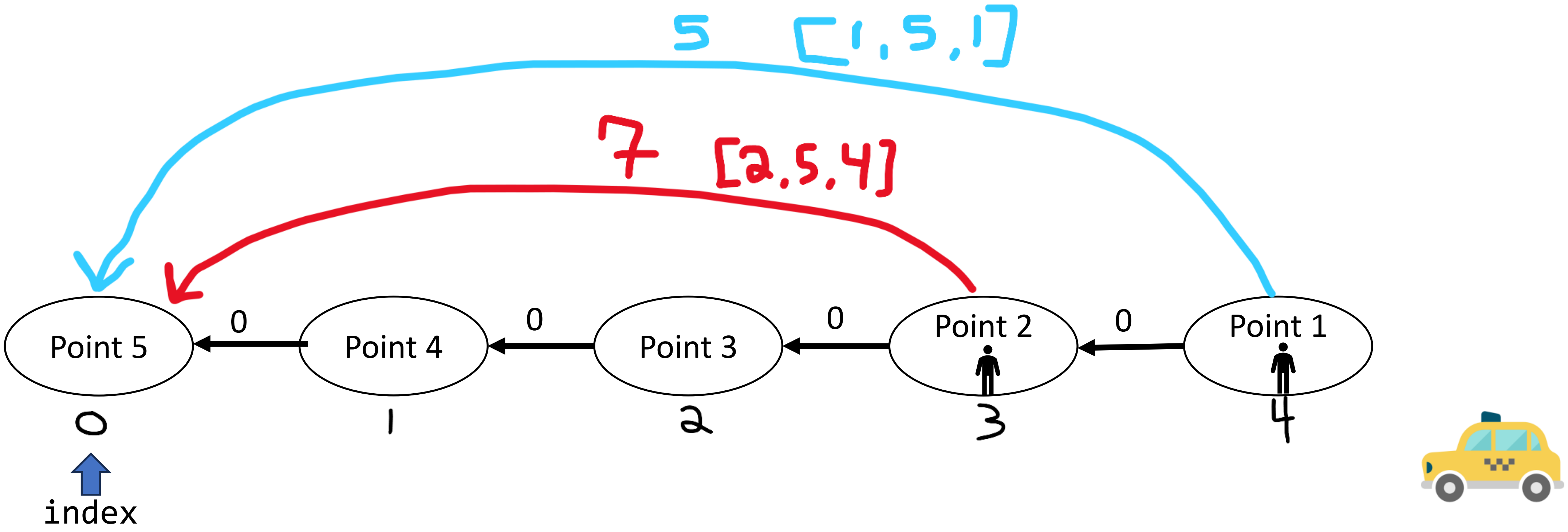
What is the **maximum profit** that the taxi can make when going from point 1 to point N?



Taxi Profit

What is the **maximum profit** that the taxi can make when going from point 1 to point N?

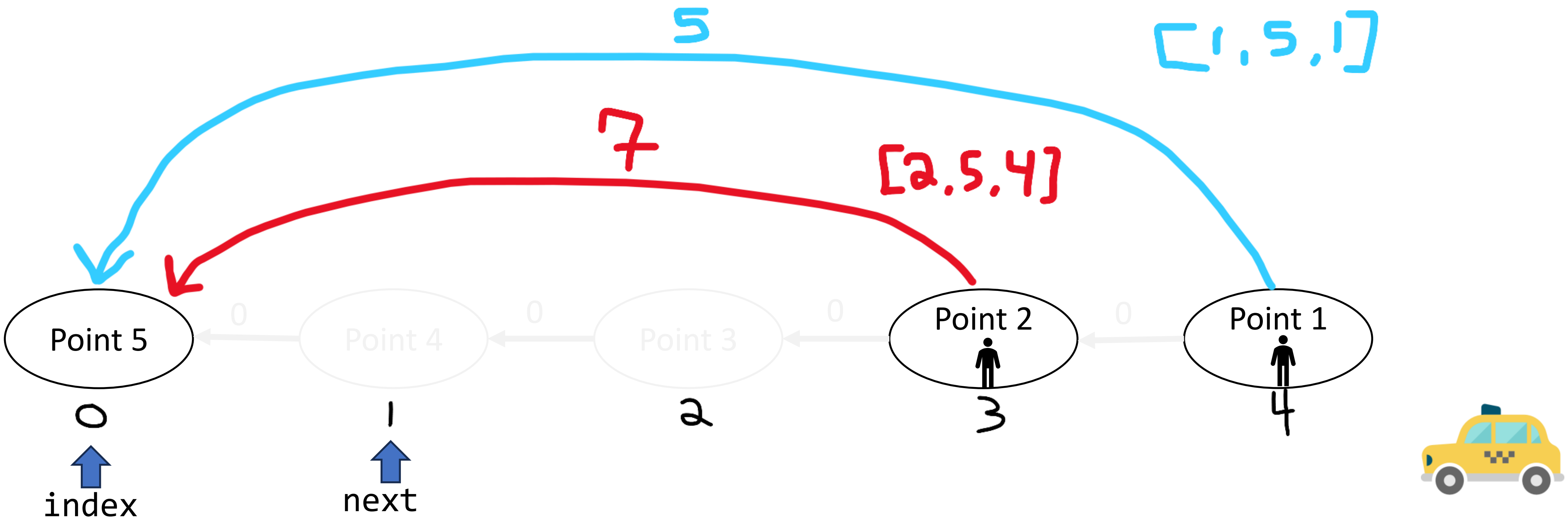
We are going to do this **Top-Down**



We are going to do this **Top-Down**

To solve B(5) we must solve B(4). There isn't an entry that starts at point 4, so must find the next closest customer

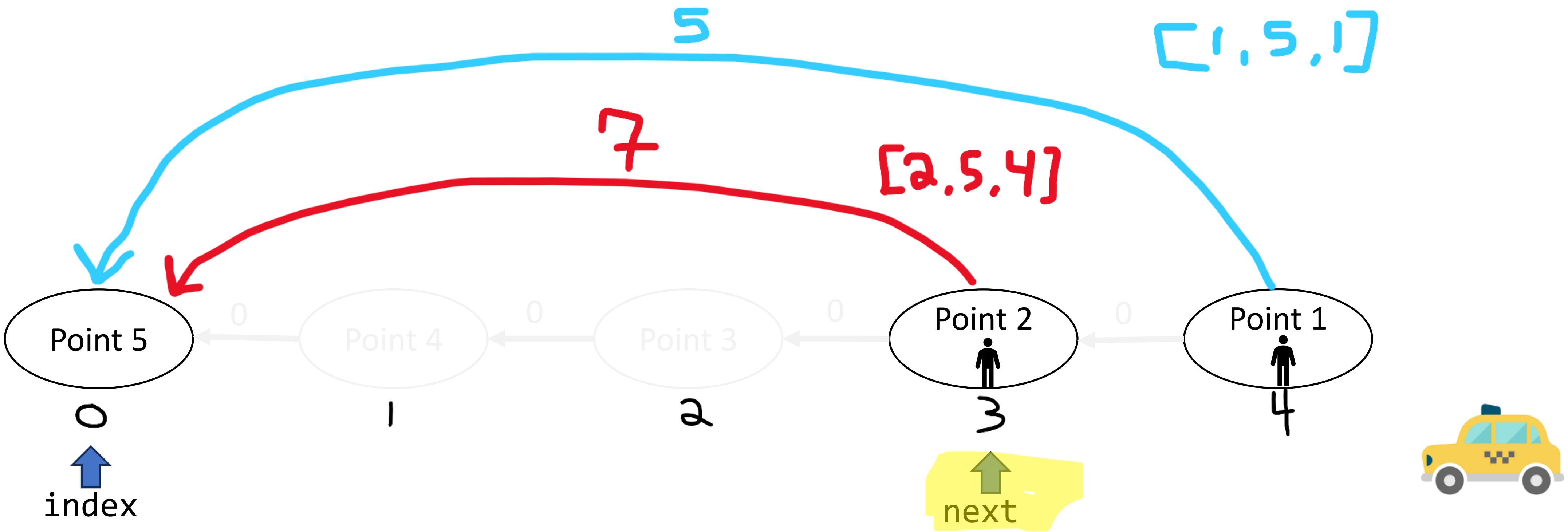
```
int[][] rides = [[2, 5, 4] , [1, 5, 1]]
```



We are going to do this **Top-Down**

To solve B(5) we must solve B(4). There isn't an entry that starts at point 4, so must find the next closest customer

```
int[][] rides = [[2, 5, 4] , [1, 5, 1]]
```



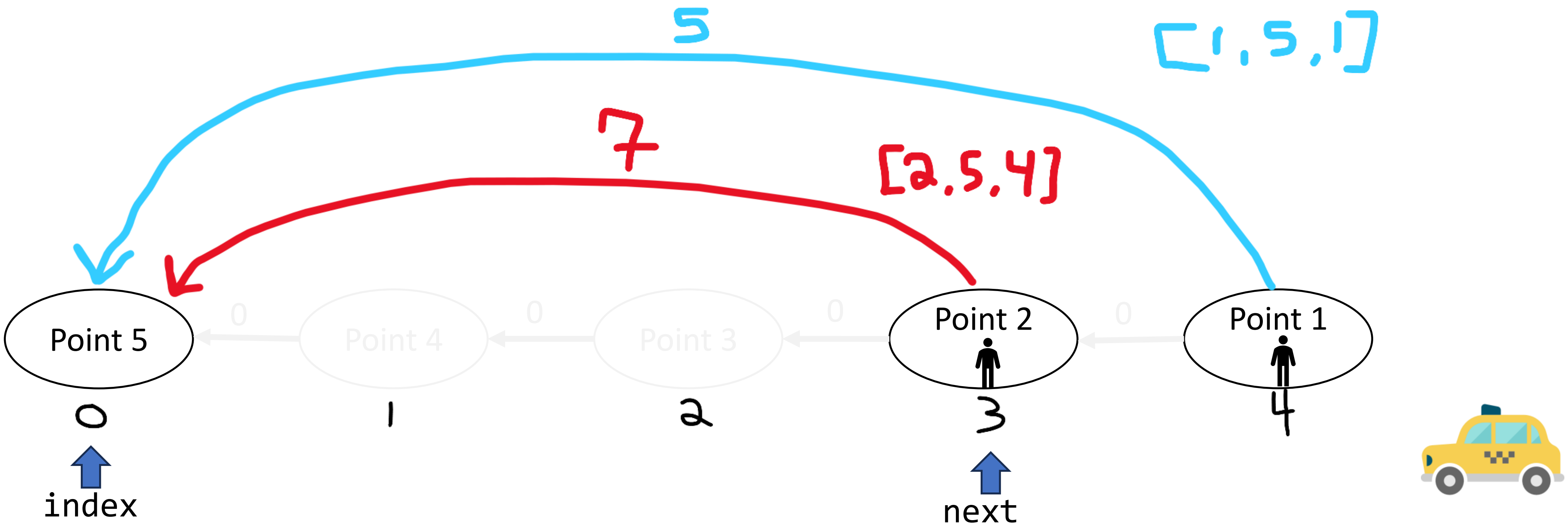
What is the **maximum profit** that the taxi can make when going from point 1 to point N?

Let B(x) be the maximum cost to go from point 1 to point x

We are going to do this **Top-Down**

To solve B(5) we must solve B(4). There isn't an entry that starts at point 4, so must find the next closest customer

```
int[][] rides = [[2, 5, 4] , [1, 5, 1]]
```



Taxi Profit

What is the **maximum profit** that the taxi can make when going from point 1 to point N?

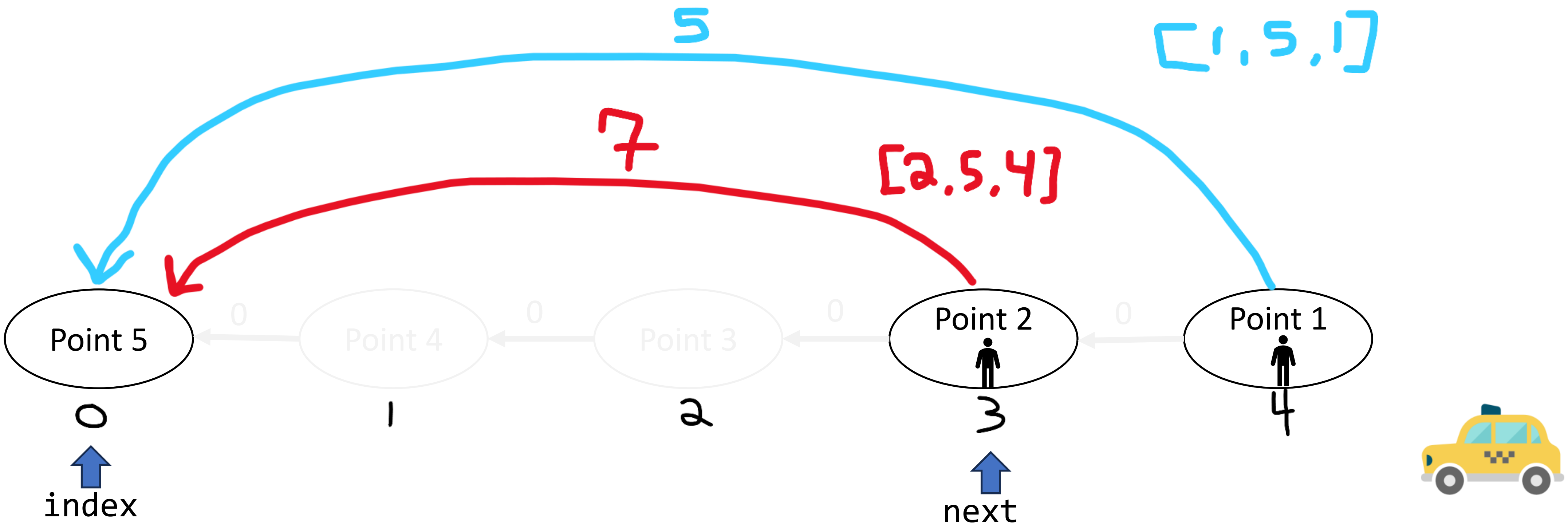
Let B(x) be the maximum cost to go from point 1 to point x

Either we **take** this customer, or we **skip**

Recursive call!

take: (ride.end - ride.start + ride.tip) + B(2)

"We pick up the customer, and add their payment to our profit so far"



Taxi Profit

What is the **maximum profit** that the taxi can make when going from point 1 to point N?

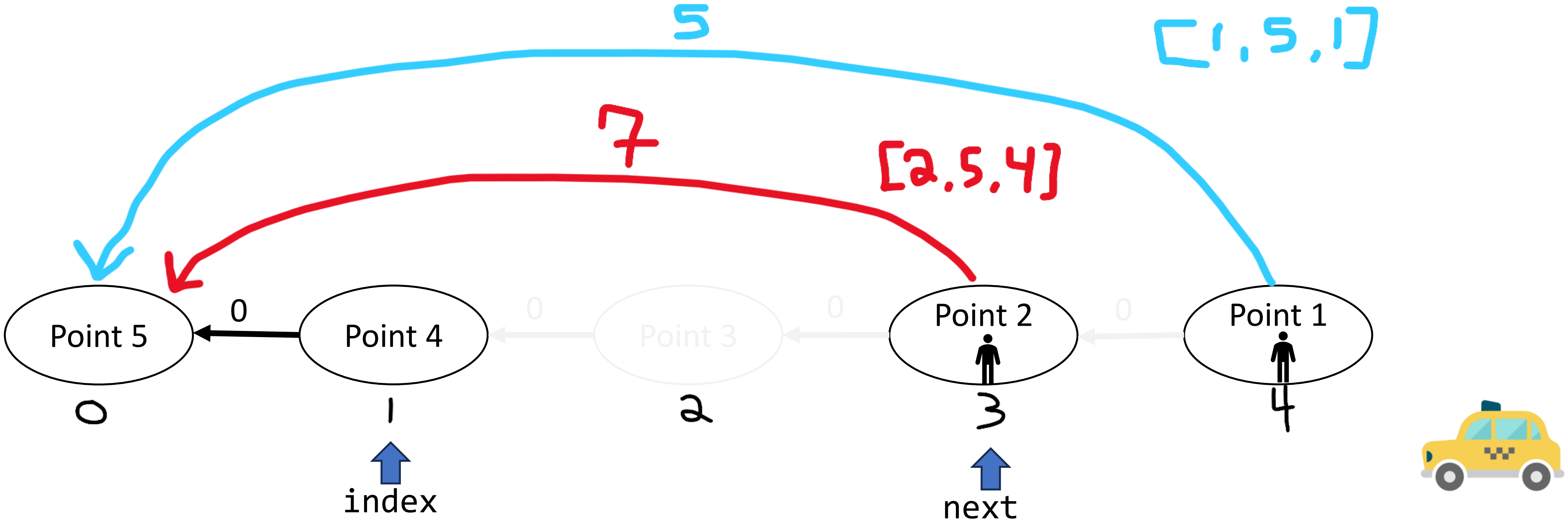
Let B(x) be the maximum cost to go from point 1 to point x

Either we **take** this customer, or we **skip**

take: (ride.end - ride.start + ride.tip) + B(2)

Skip: B(4)

index + 1



Taxi Profit

What is the **maximum profit** that the taxi can make when going from point 1 to point N?

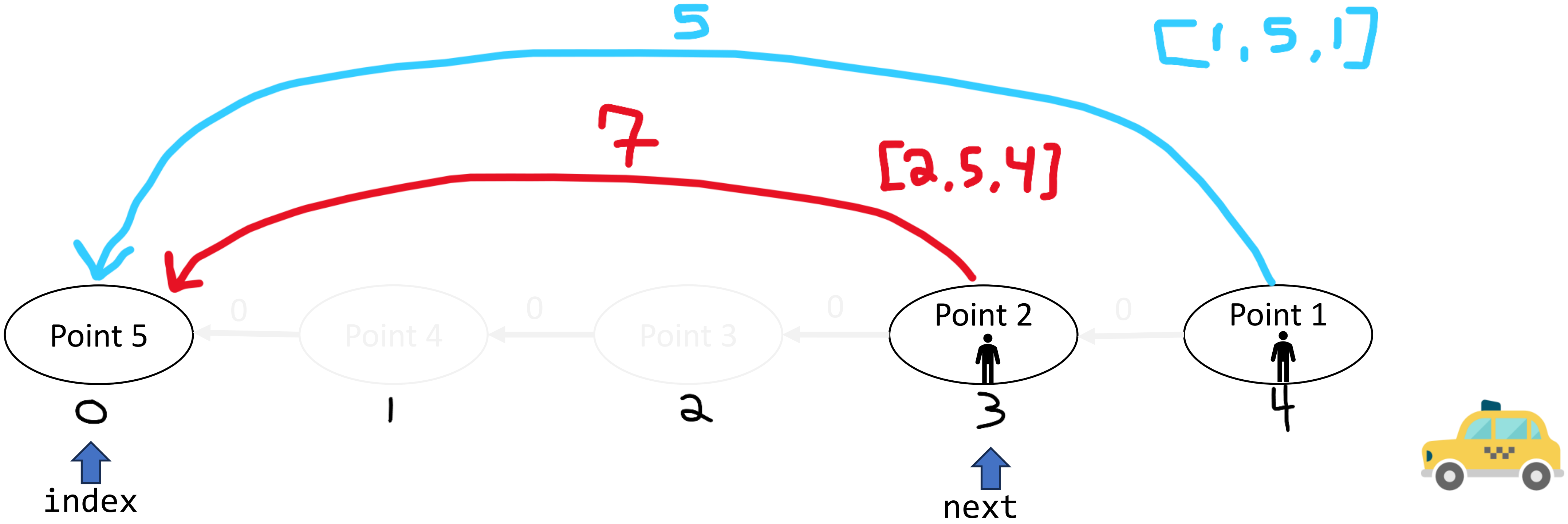
Let B(x) be the maximum cost to go from point 1 to point x

Either we **take** this customer, or we **skip**

take: (ride.end - ride.start + ride.tip) + B(2)

Skip: B(4)

Take the Max choice, and store answer in memorization table

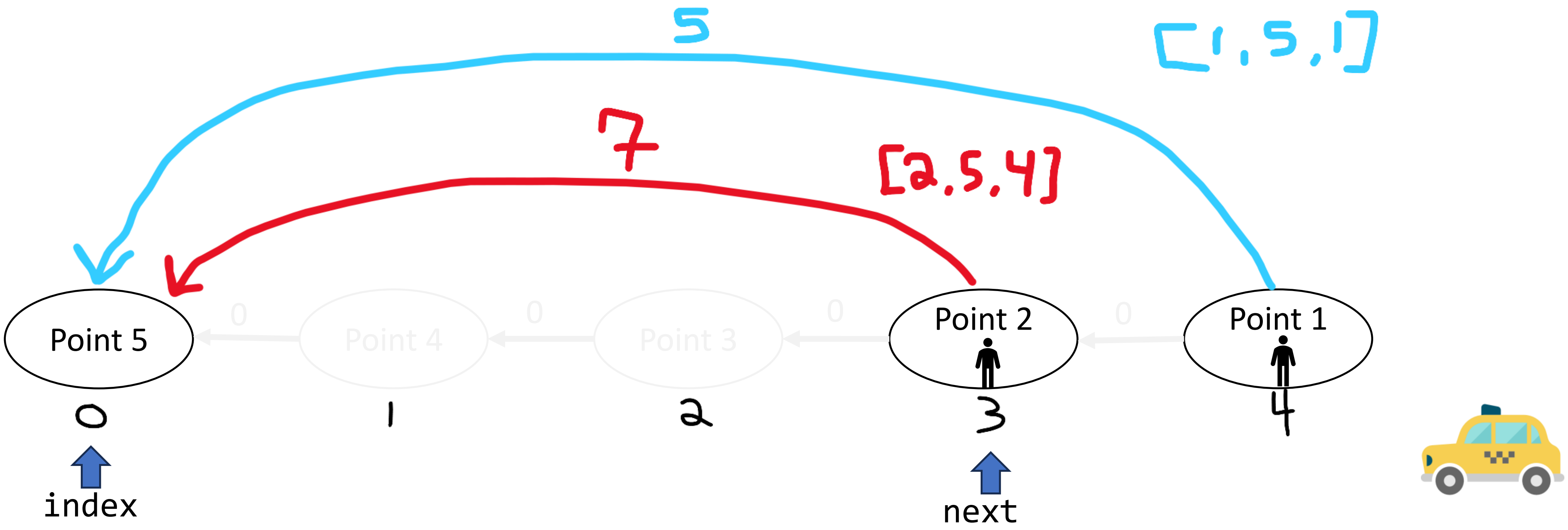


Taxi Profit

What is the **maximum profit** that the taxi can make when going from point 1 to point N?

Let B(x) be the maximum cost to go from point 1 to point x

	0	1	2	3	4
int[] dp					



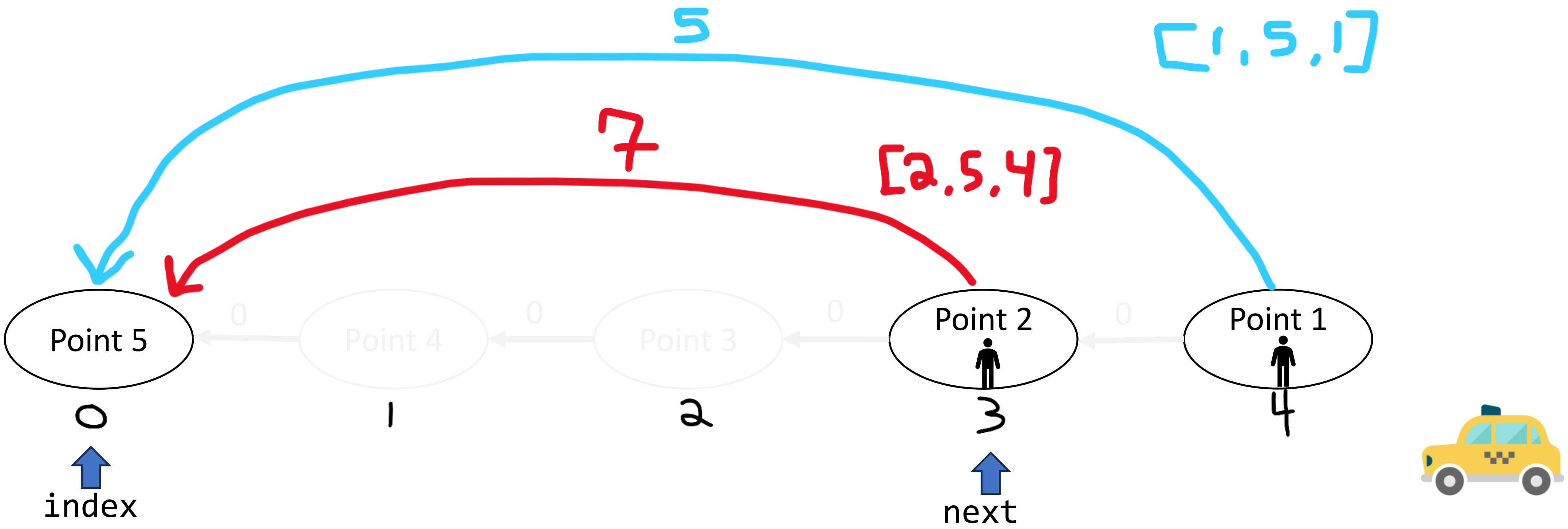
Taxi Profit

What is the **maximum profit** that the taxi can make when going from point 1 to point N?

Let B(x) be the maximum cost to go from point 1 to point x

	0	1	2	3	4
int[] dp	7	0	0	0	0

"The maximum profit to go from point 1 to point 5 (index 0) is $7"



Taxi Profit

What is the **maximum profit** that the taxi can make when going from point 1 to point N?

Let B(x) be the maximum cost to go from point 1 to point x

	0	1	2	3	4
int[] dp	7	0	0	0	0

"The maximum profit to go from point 1 to point 5 (index 0) is $7"