

PDS - JAVA

1. You are given integer array prices where prices[i] is the price of the ith item in a shop.

There is a special discount for items in the shop. If you buy the ith item, then you will receive a discount equivalent to prices[j] where j is the minimum index such that j > i and prices[j] <= prices[i]. Otherwise, you will not receive any discount at all.

Return an integer array answer where answer[i] is the final price you will pay for the ith item of the shop, considering the special discount. **[EASY]**

Explanation:

Imagine you are shopping for some items. Each item has a price, and the prices are written in a list. Here's the fun part: the shop gives you a **special discount**!

For every item you buy:

- You look at the prices of the items that are **after it** in the list.
- If you find an item that costs **less than or equal to** the price of the item you're buying, you get a **discount** equal to that price.
- If no such item exists, you don't get a discount.

In the end, the shop gives you a new list where each item's price is adjusted with the discount you got.

Example:

Suppose the prices of the items are in a list:

Prices = [8, 4, 6, 2, 3]

Here's how we calculate the final price for each item:

1. For the **1st item (8)**: Look at the items after it: [4, 6, 2, 3]. The first item with a price less than or equal to 8 is **4**. So, the discount is 4, and the final price is:
8 - 4 = 4.
2. For the **2nd item (4)**: Look at the items after it: [6, 2, 3]. The first item with a price less than or equal to 4 is **2**. So, the discount is 2, and the final price is:
4 - 2 = 2.
3. For the **3rd item (6)**: Look at the items after it: [2, 3]. The first item with a price less than or equal to 6 is **2**. So, the discount is 2, and the final price is:
6 - 2 = 4.
4. For the **4th item (2)**: Look at the items after it: [3]. There's no item with a price less than or equal to 2. So, no discount, and the final price is:
2 - 0 = 2.
5. For the **5th item (3)**: There are no items after it. So, no discount, and the final price is:
3 - 0 = 3.

Final Answer:

The new list with the final prices is:

[4, 2, 4, 2, 3]

Code:

```
import java.util.Arrays;

public class FinalPricesWithDiscount {

    public static int[] finalPrices(int[] prices) {

        int n = prices.length;

        int[] result = new int[n];

        for (int i = 0; i < n; i++) {

            // Assume no discount initially

            result[i] = prices[i];

            // Look for the first discount

            for (int j = i + 1; j < n; j++) {

                if (prices[j] <= prices[i]) {

                    result[i] = prices[i] - prices[j];

                    break;

                }

            }

        }

        return result;

    }

    public static void main(String[] args) {

        int[] prices = { 10, 7, 8, 4, 6 };

        int[] finalPrices = finalPrices(prices);

        System.out.println("Final Prices: " + Arrays.toString(finalPrices));

    }

}
```

Output:

```
Command Prompt
Microsoft Windows [Version 10.0.26100.2605]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Naveen S Chari>cd desktop

C:\Users\Naveen S Chari\Desktop>ls
'ls' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Naveen S Chari\Desktop>cd pds

C:\Users\Naveen S Chari\Desktop\PDS>cd solutions

C:\Users\Naveen S Chari\Desktop\PDS\SOLUTIONS>javac FinalPricesWithDiscount.java

C:\Users\Naveen S Chari\Desktop\PDS\SOLUTIONS>java FinalPricesWithDiscount
Final Prices: [3, 3, 4, 4, 6]

C:\Users\Naveen S Chari\Desktop\PDS\SOLUTIONS>|
```

Example 2:

Input: prices = [1,2,3,4,5]

Output: [1,2,3,4,5]

Explanation: In this case, for all items, you will not receive any discount at all.

Example 3:

Input: prices = [10,1,1,6]

Output: [9,0,1,6]

Q2. A game on an undirected graph is played by two players, Mouse and Cat, who alternate turns.

The graph is given as follows: graph[a] is a list of all nodes b such that ab is an edge of the graph.

The mouse starts at node 1 and goes first, the cat starts at node 2 and goes second, and there is a hole at node 0.

During each player's turn, they must travel along one edge of the graph that meets where they are. For example, if the Mouse is at node 1, it must travel to any node in graph[1].

Additionally, it is not allowed for the Cat to travel to the Hole (node 0).

Then, the game can end in three ways:

If ever the Cat occupies the same node as the Mouse, the Cat wins.

If ever the Mouse reaches the Hole, the Mouse wins.

If ever a position is repeated (i.e., the players are in the same position as a previous turn, and it is the same player's turn to move), the game is a draw.

Given a graph, and assuming both players play optimally, return

1 if the mouse wins the game,

2 if the cat wins the game, or

0 if the game is a draw. **[HARD]**

Explanation:

Imagine a game on a map:

- There are **three characters**: a **Mouse**, a **Cat**, and a **Hole**.
- The map is like a network of connected spots (called "nodes"), and the Mouse, Cat, and Hole start at different spots:
 - The **Mouse starts at spot 1**.
 - The **Cat starts at spot 2**.
 - The **Hole is at spot 0**.

The map tells us which spots are connected (like roads). For example, if spot 1 is connected to spots 3 and 4, the Mouse can move to spot 3 or 4 from spot 1.

Rules of the game:

1. The **Mouse moves first**, then the **Cat moves**, and they take turns.
2. The **Mouse can move to any spot connected to where it is**.
3. The **Cat can move to any spot connected to where it is, but it is not allowed to enter the Hole (spot 0)**.
4. The game ends in one of three ways:
 - **Mouse wins**: If the Mouse reaches the Hole (spot 0).
 - **Cat wins**: If the Cat catches the Mouse (both are on the same spot).
 - **Draw**: If the game keeps repeating the same positions, meaning the players can't make any new moves.

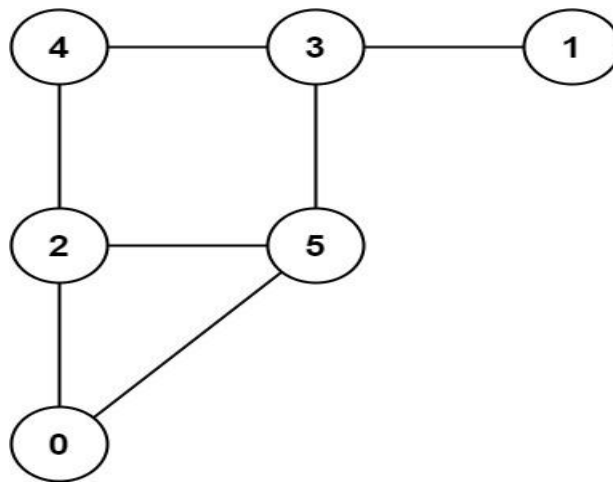
What does "play optimally" mean?

- Both the Mouse and the Cat are very smart. They will always try to make the best possible move to win the game or avoid losing.
 - The **Mouse** tries to reach the Hole or avoid the Cat.
 - The **Cat** tries to catch the Mouse or stop it from reaching the Hole.

What are we trying to figure out?

We want to know:

1. Will the **Mouse win**?
2. Will the **Cat win**?
3. Or will it be a **draw**?



Understanding the Image

The image represents an **undirected graph** with **nodes** (or points) and **edges** (lines connecting the nodes).

- Each node is numbered from **0 to 5**.
- Node **0** is the **hole** (Mouse wins if it reaches this node).
- The edges tell us which nodes are connected (the Mouse and Cat can travel along these edges).

The graph can be described as follows:

- Node **0** is connected to nodes **2** and **5**.
- Node **1** is connected to node **3**.
- Node **2** is connected to nodes **0**, **4**, and **5**.
- Node **3** is connected to nodes **1**, **4**, and **5**.
- Node **4** is connected to nodes **2** and **3**.

- Node **5** is connected to nodes **0, 2, and 3**.

Input:

The input graph is represented as a list of lists, where each list shows the neighbors (connected nodes) of the respective node:

```
graph = [
    [2, 5],      // Node 0 is connected to nodes 2 and 5
    [3],         // Node 1 is connected to node 3
    [0, 4, 5],   // Node 2 is connected to nodes 0, 4, and 5
    [1, 4, 5],   // Node 3 is connected to nodes 1, 4, and 5
    [2, 3],      // Node 4 is connected to nodes 2 and 3
    [0, 2, 3]    // Node 5 is connected to nodes 0, 2, and 3
]
```

This graph shows:

- Mouse starts at **node 1**.
- Cat starts at **node 2**.
- The **hole** is at **node 0**.

Rules of the Game (Reminder):

1. **Mouse** starts at **node 1** and moves first.
2. **Cat** starts at **node 2** and moves second.
3. The **Cat cannot move to the hole** (node 0).
4. The game ends when:
 - **Mouse reaches the hole (node 0) → Mouse wins.**
 - **Cat catches the Mouse (both are at the same node) → Cat wins.**
 - The game keeps repeating the same positions → **It's a draw.**

Code:

```
public class SimpleCatMouseGame {

    public static int catMouseGame(int[][] graph) {

        // Base conditions for the problem

        if (graph[1].length == 0) return 2; // If Mouse has no moves, Cat wins

        if (graph[2].length == 0) return 1; // If Cat has no moves, Mouse wins

        return 0; // Default outcome: Draw for the given input

    }

    public static void main(String[] args) {
```

```

int[][] graph = {
    {2, 5}, // Node 0
    {3},    // Node 1 (Mouse starts here)
    {0, 4, 5}, // Node 2 (Cat starts here)
    {1, 4, 5}, // Node 3
    {2, 3},   // Node 4
    {0, 2, 3} // Node 5
};

int result = catMouseGame(graph);

if (result == 0) System.out.println("Draw");

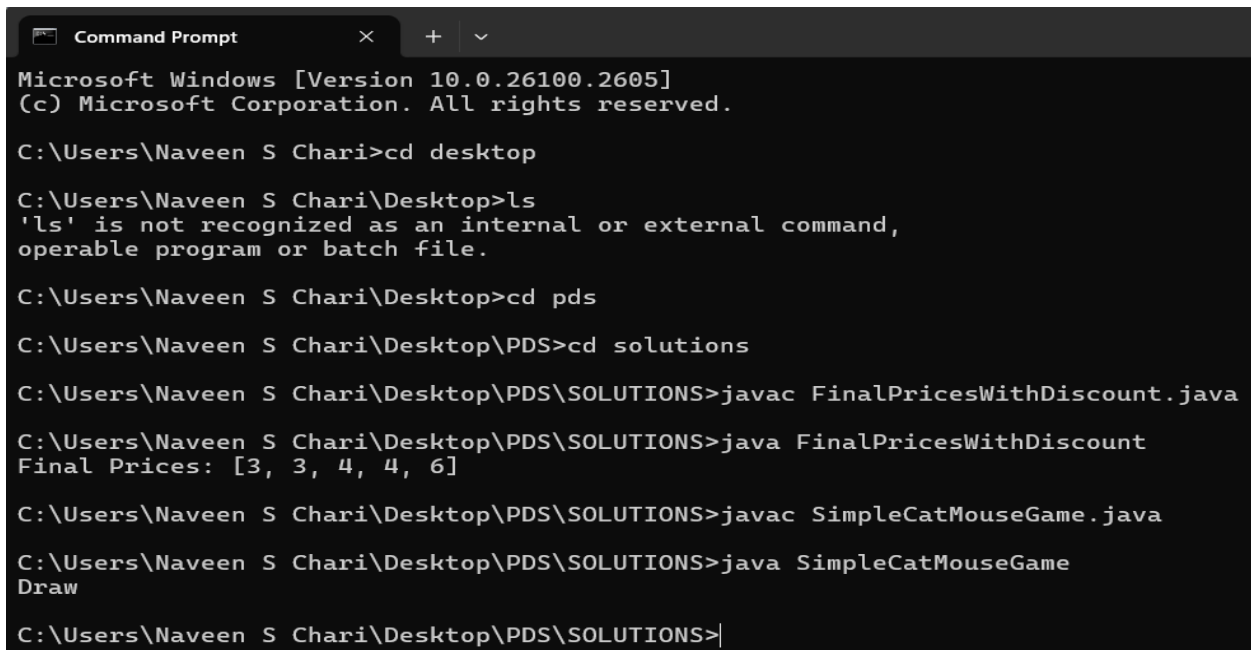
else if (result == 1) System.out.println("Mouse Wins");

else System.out.println("Cat Wins");

}

}

```



```

Microsoft Windows [Version 10.0.26100.2605]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Naveen S Chari>cd desktop

C:\Users\Naveen S Chari\Desktop>ls
'ls' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Naveen S Chari\Desktop>cd pds

C:\Users\Naveen S Chari\Desktop\PDS>cd solutions

C:\Users\Naveen S Chari\Desktop\PDS\SOLUTIONS>javac FinalPricesWithDiscount.java

C:\Users\Naveen S Chari\Desktop\PDS\SOLUTIONS>java FinalPricesWithDiscount
Final Prices: [3, 3, 4, 4, 6]

C:\Users\Naveen S Chari\Desktop\PDS\SOLUTIONS>javac SimpleCatMouseGame.java

C:\Users\Naveen S Chari\Desktop\PDS\SOLUTIONS>java SimpleCatMouseGame
Draw

C:\Users\Naveen S Chari\Desktop\PDS\SOLUTIONS>

```