Problem Description*: Ruby Second House (Medium)*

You are a construction manager overseeing the painting of a long row of houses. Each house must be painted one of several available colors. However, adjacent houses **cannot** have the same color to preserve aesthetic appeal.

Each painting option has a different cost. You are given a matrix where `costs[i][j]` represents the cost of painting house `i` with color `j`.

Your task is to paint all houses such that:

- No two adjacent houses share the same color.
- The total painting cost is **minimized**.

Input Format

- First line: Two integers `n` and `k` — the number of houses and the number of colors.
- Next `n` lines: Each line contains `k` space-separated integers, the painting costs for that house using each color.

Output Format

- A single integer — the minimum total cost to paint all the houses such that no two adjacent houses have the same color.

Constraints

- `1 <= n <= 100`
- `2 <= k <= 20`
- `1 <= costs[i][j] <= 20`

Example

Input

```
2 3
1 5 3
2 9 4
```

Output

```
5
```

Explanation

There are two optimal solutions:

- Paint house 0 with color 0 (cost = 1), house 1 with color 2 (cost = 4) → total = 1 + 4 = **5**

- Paint house 0 with color 2 (cost = 3), house 1 with color 0 (cost = 2) → total = 3 + 2 = **5**

20 Additional Test Cases (Updated Format)

S.N	**Input (n k and cost matrix)**	**Output**

1	`2 2`
	`1 2`
	`2 1`
		2

2	`3 3`
	`1 5 3`
	`1 9 1`
	`3 2 4`
		4

3	`3 2`
	`8 2`
	`4 3`
	`1 7`
		12

4	`3 3`
	`1 1 1`
	`1 1 1`
	`1 1 1`
		3

5	`3 6`
	`7 3 8 6 1 2`
	`5 6 7 2 4 3`
	`10 1 4 9 7 6`
		4

6	`2 2`
	`3 4`
	`2 3`
		6

7	`3 3`
	`1 10 3`
	`9 2 8`
	`4 5 6`
		7

8	`3 2`
	`5 5`
	`5 5`
	`5 5`
		15

9	`3 2`
	`1 100`
	`100 1`
	`1 100`
		3

10	`2 4`
	`1 2 3 4`
	`4 3 2 1`
		2

11	`4 2`
	`1 20`
	`20 1`
	`1 20`
		4

S.N	**Input (n k and cost matrix)**	**Output**
	`20 1`	

12	`3 3`
	`2 1 3`
	`6 4 5`
	`7 8 9`
		13

13	`4 2`
	`10 15`
	`1 1`
	`5 6`
	`7 3`
		19

14	`4 2`
	`3 2`
	`1 4`
	`2 3`
	`4 5`
		10

15	`3 2`
	`1 2`
	`1 2`
	`1 2`
		4

16	`4 3`
	`3 2 1`
	`2 1 3`
	`1 3 2`
	`3 1 2`
		4

17	`5 2`
	`20 10`
	`10 20`
	`20 10`
	`10 20`
	`20 10`
		50

18	`3 3`
	`5 6 7`
	`6 5 8`
	`7 6 5`
		15

19	`3 4`
	`9 8 7 6`
	`6 7 8 9`
	`9 6 8 7`
		18

20	`4 2`
	`1 2`
	`2 1`
	`1 2`
	`2 1`
		4

Solution:

```java
public class PaintHouseII {

    public int minCostII(int[][] costs) {
        int n = costs.length;
        if (n == 0) return 0;
        int k = costs[0].length;
        if (k == 0) return 0;

        // Track the least and second least cost from the previous row
        int min1 = -1, min2 = -1;

        for (int i = 0; i < n; i++) {
            int lastMin1 = min1, lastMin2 = min2;
            min1 = -1;
            min2 = -1;

            for (int j = 0; j < k; j++) {
                if (i > 0) {
                    // If previous color was not j, use the min1 cost
                    if (j != lastMin1) {
                        costs[i][j] += costs[i - 1][lastMin1];
                    } else {
                        // If previous color was j, use second min
                        costs[i][j] += costs[i - 1][lastMin2];
                    }
                }

                // Update min1 and min2 for current row
                if (min1 == -1 || costs[i][j] < costs[i][min1]) {
                    min2 = min1;
                    min1 = j;
                } else if (min2 == -1 || costs[i][j] < costs[i][min2]) {
                    min2 = j;
                }
            }
        }

        return costs[n - 1][min1];
    }

    // For testing with console input format
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int k = scanner.nextInt();
        int[][] costs = new int[n][k];

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < k; j++) {
                costs[i][j] = scanner.nextInt();
            }
        }
        System.out.println(minCostII(costs));
    }
    }
}
```