# 🪜 746. Min Cost Climbing Stairs (Iterative DP)

## 💻 Code

```
class Solution {
public:
    int minCostClimbingStairs(vector<int>& arr) {
        for (int i = 2; i < arr.size(); i++) {
            arr[i] += min(arr[i - 1], arr[i - 2]);
        }
        return min(arr[arr.size() - 1], arr[arr.size() - 2]);
    }
};
```

## 🧩 Step-by-Step Explanation

| Step | Description |
|---|---|
| 1️⃣ | Start from the **2nd index (i = 2)** because the first two steps (0 and 1) are your starting options. |
| 2️⃣ | For each step `i`, compute the **minimum cost** to reach it by taking the cheaper of the two previous steps: `arr[i] += min(arr[i-1], arr[i-2])`. |
| 3️⃣ | This means at every index, you're storing the **minimum total cost to reach that step**. |
| 4️⃣ | Finally, you can reach the top either from the **last step (n-1)** or the **second last step (n-2)**, so return `min(arr[n-1], arr[n-2])`. |

## ⚙️ Intuition

At each step, you decide whether it's cheaper to come from one step below or two steps below.

By the end of the loop, each element of `arr` represents the **minimum cumulative cost** to reach that step.

---

## 🧠 Dry Run Example

**Input:** `[1,100,1,1,1,100,1,1,100,1]`

| i | arr before update | Operation | arr after update |
|---|---|---|---|
| 2 | `[1,100,1,...]` | `arr[2] += min(100,1)` → `arr[2]=1+1` | `[1,100,2,...]` |
| 3 | `[1,100,2,...]` | `arr[3] += min(100,2)` → `arr[3]=1+2` | `[1,100,2,3,...]` |
| 4 | `[1,100,2,3,...]` | `arr[4] += min(2,3)` → `arr[4]=1+2` | `[1,100,2,3,3,...]` |
| ... | continue | ... | ... |
| Final | `[1,100,2,3,3,103,4,4,104,5]` | `min(5,104)` → ✅ **6** | |

---

## ⏱️ Time Complexity

- **O(n)** → Single pass through the array.

## 💾 Space Complexity

- **O(1)** → In-place update of the input array.

---

## 🧭 Key Idea

- Modify the input array to act as a **DP table**.
- Each cell represents the **minimum cost** to reach that step.
- Answer = `min(last, second_last)`.

---

## ✅ Quick Summary

| Concept | Description |
|---|---|
| **Approach** | Bottom-Up Dynamic Programming |
| **Transition** | `arr[i] += min(arr[i-1], arr[i-2])` |
| **Base Case** | Start from step `0` or `1` |

| Concept | Description |
|---|---|
| **Answer** | `min(arr[n-1], arr[n-2])` |
| **Complexity** | Time: O(n), Space: O(1) |