قورباغه جادوگر

مساله مانند مساله اول همین تمرین است ولی حالت‌بندی برای پرکردن آرایه متفاوت است.
مانند مساله پله‌ها برای رسیدن به هر سنگ، نیاز به دانستن کم‌ترین انرژی برای رسیدن به سنگ‌های قبلی‌اش داریم و برای تعریف زیرمساله به تنها یک متغیر i به عنوان آخرین سنگ نیاز داریم. آرایه dp[i] را کم‌ترین انرژی برای رسیدن به سنگ i ام تعریف می‌کنیم.
برای پیدا کردن رابطه بازگشتی روی طول آخرین قدم حالت‌بندی می‌کنیم و چک می‌کنیم که آیا سنگ قبلی جادویی هست یا نه.

```
dp[i] = min(dp[i-1] + (if i-1 % k == 0 : a/2 else a),
            dp[i-2] + (if i-2 % k == 0 : b/2 else b,
            dp[i-3] + (if i-3 % k == 0 : c/2 else c))
```

البته نامنفی بودن اندیس را باید چک کنیم.
حالت پایه:

```
dp[0] = 0
```

```
function rock_climbing(n, a, b, c, k):
    dp = 1D array with size n+1

    dp[0] = 0
    for i from 1 to n:
        one_rock_cost = a
        if (i-1)%k == 0 and (i-1) != 0:
            one_rock_cost = one_rock_cost / 2
        dp[i] = dp[i-1] + one_rock_cost

        if (i-2) >= 0:
            two_rock_cost = b
            if (i-2)%k == 0 and (i-2) != 0:
                two_rock_cost = two_rock_cost / 2
```

```
            dp[i] = min(dp[i], dp[i-2] + two_rock_cost)
        if (i-3) >= 0:
            three_rock_cost = c
            if (i-3)%k == 0 and (i-3) != 0:
                three_rock_cost = three_rock_cost / 2
            dp[i] = min(dp[i], dp[i-3] + three_rock_cost)

    return dp[n]
```