## Task 3 : Security Switches

### 3.1 Problem Description

There is a row of n security switches protecting a military installation entrance. The switches can
be manipulated as follows:
(i) The rightmost switch may be turned on or off at will.
(ii) Any other switch may be turned on or off only if the switch to its immediate right is on and all the other switches to its right, if any, are off.
(iii) Only one switch may be toggled at a time.
Design a Dynamic Programing algorithm to turn off all the switches, which are initially all on, in the minimum number of moves. (Toggling one switch is considered one move.) Also find the minimum number of moves.

### 3.2 Comment

It is an optimization problem as we want the minimum number of moves using dynamic programming, we will go with a forward approach backward reasoning and as there are lots of possibilities it will be hard to draw the multistage graph so we will settle only with the recurrence relation.

## 3.3 Detailed Solution

### 3.3.1 Description of Solution

First analyze the problem and understand the recursion in it as it has overlapping problems, we can see that to turn off a switch we need its immediate right to be on and all the others to the right off

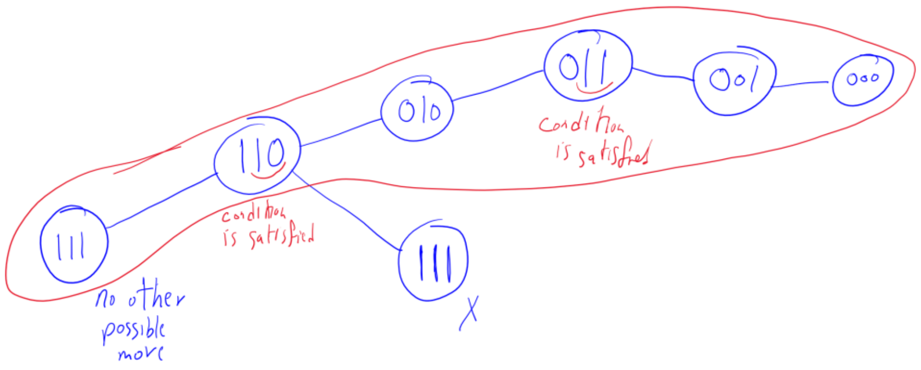


FIGURE 1

We can see the pattern to solve a switch (i) we need to solve for the switches i+2 and the rest on the right to be off and then toggle switch(i) getting 0100…. To get it to be 011111… we need M(n-2) moves again leading to 2 M(n-2) till now then we have to solve M(N-1) of the original problem +1 for the first switch that is a base case leading to the recurrence relation below.

**Recurrence Relation:**

$M(n) = M(n - 2) + 1 + M(n - 2) + M(n - 1)$

Or

$M(n) = M(n - 1) + 2M(n - 2) + 1$ for $n \geq 3$, $M(1) = 1$, $M(2) = 2$

- $M(1) = 1$ and $M(2) = 2$ are the base cases
- $M(n - 1)$: Number of moves needed to solve for (n - 1) switches.
- $M(n - 2)$: Number of moves needed to solve for (n - 2) switches, and we need to double it because we toggle the switches twice in that case.
- 1: Additional move needed to toggle the first switch

## 3.3.2 Pseudo Code

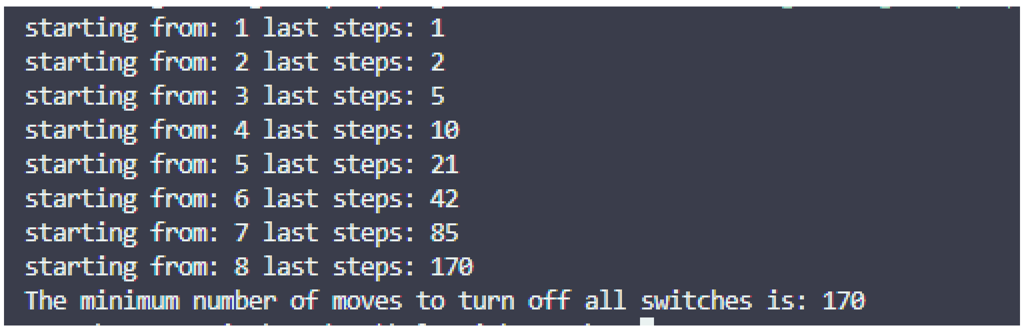
```
Function min_moves(n):
    If n is 1:
        Return 1
    If n is 2:
        Return 2

    Initialize an array dp of size n+1 with zeros
    Set dp[1] to 1
    Set dp[2] to 2

    Print "starting from: 1 last steps: " + dp[1]
    Print "starting from: 2 last steps: " + dp[2]

    For i from 3 to n:
        Set dp[i] to dp[i-1] + 2*dp[i-2] + 1
        Print "starting from: " + i + " last steps: " + dp[i]

    Return dp[n]
End Function
```

## 3.4 Complexity Analysis for the Algorithm

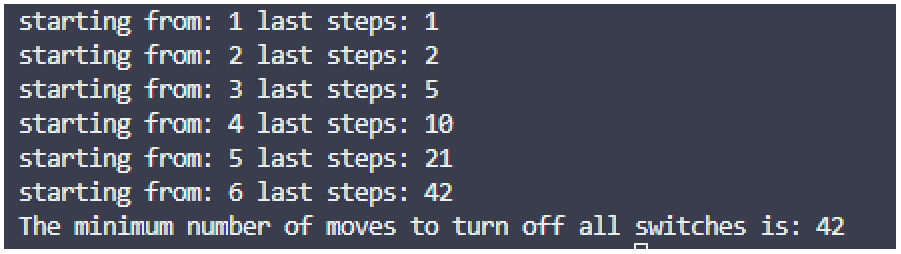
For this code the time complexity will be O(n) using the recurrence relation

The puzzle can be solved in the minimum of $\frac{2}{3} \times 2^n - \frac{1}{6}(-1^n) - \frac{1}{2}$ switch toggles.

## 3.5 Sample Output of the Solution



```
starting from: 1 last steps: 1
starting from: 2 last steps: 2
starting from: 3 last steps: 5
starting from: 4 last steps: 10
starting from: 5 last steps: 21
starting from: 6 last steps: 42
starting from: 7 last steps: 85
starting from: 8 last steps: 170
The minimum number of moves to turn off all switches is: 170
```



```
starting from: 1 last steps: 1
starting from: 2 last steps: 2
starting from: 3 last steps: 5
starting from: 4 last steps: 10
starting from: 5 last steps: 21
starting from: 6 last steps: 42
The minimum number of moves to turn off all switches is: 42
```

### 3.6 Another solution

Another Solution This is another solution that solves the puzzle using decease-and-conquer algorithm. The solution was taken from the book: “Algorithmic Puzzles by Anany Levitin”. We will number the switches left to right from 1 to n and denote the “on” and“off” states of a switch by a 1 and 0, respectively. To help ourselves with solving the general instance of the puzzle, let us start by solving its four smallest instances. Consider now the general instance of the puzzle represented by the bit string of n 1’s: 111. . . 1. Before we can turn off the first (leftmost) switch, the switches should be in the state 110. . . 0. Hence, to begin with, we need to turn off the last n−2 switches and do this in the minimum number of moves if we want to have an optimal algorithm. In other words, we should first solve the same problem as given for the last n−2 switches. This can be done recursively with the n = 1 and n = 2 instances solved directly. After that, we can toggle the first switch to get 010. . . 0. Now, before we can toggle the second switch we will need to pass through the state with all the switches following it “on,” which can be easily proved by mathematical induction. Getting all the switches from the third to the last one “on” can be achieved by reversing the optimal sequence of moves made previously to toggle the last n − 2 switches from the “on” position to the “off” position. This will give us 011. . . 1. Ignoring the first 0, we now have the n − 1 instance of the original puzzle, which can be solved recursively.

| |
|---|
| 1 |
| 0 |

$n = 1$

| | |
|---|---|
| 1 | 1 |
| 0 | 1 |
| 0 | 0 |

$n = 2$

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 0 |
| 0 | 1 | 0 |
| 0 | 1 | 1 |
| 0 | 0 | 1 |
| 0 | 0 | 0 |

$n = 3$

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 |

$n = 4$

FIGURE 2

### 3.7 Conclusion

The approach described in the book "Algorithmic Puzzles by Anany Levitin" uses a decrease-and conquer strategy instead of dynamic programming.

TABLE 1

| Criteria | Dynamic Programming | Decrease-and-Conquer |
|---|---|---|
| **Base Cases** | The base case is when the subproblem size is 1, and the solution for that subproblem is already known | The base cases are the smallest instances of the problem, which are solved directly |
| **Solution Construction** | The solution is constructed by recursively solving subproblems and combining the results | The solution is constructed by solving the smallest instances of the problem first, then using those solutions to solve larger instances |