# 3100 Homework 04: Spiderman's Workout

| | |
|---|---|
| Created | @October 6, 2025 10:12 PM |
| Class | CSCI 3100 Algorithms |

## Spiderman's Workout

### Problem Decomposition

- Variant of Subset Sum problem, with constraints

**Simply Put:**

Given a list of positive integers m=[m0,m1,...,mn],

choose for each element either "+" (up) or "–" (down), applied sequentially, such that:

1. The cumulative sum never becomes negative at any point (Spiderman can't go below ground).
2. The final cumulative sum equals 0 (he ends back at ground level).
3. Among all such valid sequences, the **maximum cumulative sum** (the highest height he reaches) is **as small as possible**.

## Subproblem Definition

Let

$$f(i, h)$$

be the **minimum possible peak height** (maximum height reached so far) along the path

after taking the first `i` moves from the movement array

$$m = [m_0, m_1, m_2, \ldots, m_{n-1}],$$

and ending at current height `h`.

If it's impossible to reach height `h` after `i` moves without going below ground, then `f(i, h) = ∞`.

# Recursion & Order of Subproblems

## Base case

At the beginning:

$$f(0, 0) = 0$$

He starts on the ground and hasn't climbed anywhere,

so the highest height (peak) reached so far is 0.

All other heights are unreachable:

$$f(0, h > 0) = \infty$$

---

## Recursive step (for i > 0)

For each step `i`, Spiderman chooses to go **Up** or **Down** by distance `m[i-1]` (moves are 1-indexed in the theory).

## Case 1 — Up move

He was previously at height `h - m[i-1]`.

The new height is `h`, and his peak might increase.

$$\text{new peak} = \max(f(i-1,\ h - m_{i-1}),\ h)$$

## Case 2 — Down move

He was previously at height `h + m[i-1]`.

Descending never raises the peak:

$$\text{new peak} = f(i-1,\ h + m_{i-1})$$

---

## Combined recurrence

If both moves are possible, choose the one with the smaller peak:

$$f(i, h) = \min\Big(f(i-1,\ h + m_{i-1}),\ \max(f(i-1,\ h - m_{i-1}),\ h)\Big)$$

With the conditions:

- `h - m[i-1] ≥ 0` (can't go below ground)

- Ignore any terms involving unreachable states ( `∞` ).

---

### Boundary & termination

After processing all `n` moves:

- If `f(n, 0) < ∞`, that's the minimal peak.
- If `f(n, 0) = ∞`, output `"IMPOSSIBLE"`.

---

### summary

| Symbol | Meaning |
|---|---|
| `m_i` | size of the i-th climb (distance) |
| `h` | current height after `i` moves |
| `f(i, h)` | minimal possible *maximum height* after first `i` moves ending at `h` |
| `max()` | ensures we update the peak when climbing up |
| `min()` | chooses the better of up/down options |
| Base | `f(0,0)=0` |
| Answer | `f(n,0)` |

## Conclusion

$$f(i,h) = \begin{cases} 0, & \text{if } i = 0 \text{ and } h = 0; \\ \infty, & \text{if } i = 0 \text{ and } h > 0; \\ \infty, & \text{if both transitions are invalid (unreachable);} \\ f(i-1,\, h + m_{i-1}), & \text{if } h - m_{i-1} < 0 \text{ (can't go below ground);} \\ \min\Big(f(i-1,\, h + m_{i-1}),\ \max(f(i-1,\, h - m_{i-1}),\, h)\Big), & \text{otherwise.} \end{cases}$$

Formally,

$$f(i,h) = \begin{cases} 0, & \text{if } i = 0 \text{ and } h = 0; \\ \infty, & \text{if } i = 0 \text{ and } h > 0; \\ f(i-1,\, h + m[i-1]), & \text{if } h - m[i-1] < 0 \text{ and } f(i-1,\, h + m[i-1]) < \infty; \\ \min\left(f(i-1,\, h + m[i-1]),\ \max(f(i-1,\, h - m[i-1]),\, h)\right), & \text{if } h - m[i-1] \geq 0; \\ \infty, & \text{if } (h - m[i-1] < 0 \text{ or } f(i-1,\, h - m[i-1]) = \infty) \text{ and } f(i-1,\, h + m[i-1]) = \infty. \end{cases}$$

## Using Basic Example [1, 1, 1, 1] (Bottom-up), from i = 0 ~ i = n

### Step 0 — Base case

| i=0 | h=0 | h=1 | h=2 | h=3 | h=4 |
|---|---|---|---|---|---|
| f(0,h) | **0** | ∞ | ∞ | ∞ | ∞ |

Start on the ground, peak = 0.

Indexing is done from 1, 0th index doesn't exist (base case)

## Step 1 — Move 1 ($m_o = 1$)

| h | Came Down from Above (h+1) | Came Up from Below (h−1) | f(1,h) |
|---|---|---|---|
| 0 | f(0,1)=∞ | invalid | ∞ |
| 1 | f(0,2)=∞ | max(f(0,0)=0, 1)=1 | **1** |
| 2 | f(0,3)=∞ | max(f(0,1)=∞,2)=∞ | ∞ |
| 3 | f(0,4)=∞ | ∞ | ∞ |
| 4 | ∞ | ∞ | ∞ |

| i=1 | h=0 | h=1 | h=2 | h=3 | h=4 |
|---|---|---|---|---|---|
| f(1,h) | ∞ | **1** | ∞ | ∞ | ∞ |

## Step 2 — Move 2 ($m_1 = 1$)

Now use row 1 to compute row 2.

| h | Came Down from Above (h+1) | Came Up from Below (h−1) | f(2,h) |
|---|---|---|---|
| 0 | f(1,1)=1 | invalid | **1** |
| 1 | f(1,2)=∞ | max(f(1,0)=∞,1)=∞ | ∞ |
| 2 | f(1,3)=∞ | max(f(1,1)=1,2)=2 | **2** |
| 3 | f(1,4)=∞ | max(f(1,2)=∞,3)=∞ | ∞ |
| 4 | ∞ | ∞ | ∞ |

| i=2 | h=0 | h=1 | h=2 | h=3 | h=4 |
|---|---|---|---|---|---|
| f(2,h) | **1** | ∞ | **2** | ∞ | ∞ |

Reachable heights: 0 (UD) and 2 (UU).

## Step 3 — Move 3 ($m_2 = 1$)

Now from row 2.

| h | Came Down from Above (h+1) | Came Up from Below (h−1) | f(3,h) |
|---|---|---|---|
| 0 | f(2,1)=∞ | invalid | ∞ |
| 1 | f(2,2)=2 | max(f(2,0)=1,1)=1 | **1** |

| h | Came Down from Above (h+1) | Came Up from Below (h−1) | f(3,h) |
|---|---|---|---|
| 2 | f(2,3)=∞ | max(f(2,1)=∞,2)=∞ | ∞ |
| 3 | f(2,4)=∞ | max(f(2,2)=2,3)=3 | **3** |
| 4 | f(2,5)=∞ | ∞ | ∞ |

| i=3 | h=0 | h=1 | h=2 | h=3 | h=4 |
|---|---|---|---|---|---|
| f(3,h) | ∞ | **1** | ∞ | **3** | ∞ |

## Step 4 — Move 4 ($m_3 = 1$)

Now from row 3.

| h | Came Down from Above (h+1) | Came Up from Below (h−1) | f(4,h) |
|---|---|---|---|
| 0 | f(3,1)=1 | invalid | **1** |
| 1 | f(3,2)=∞ | max(f(3,0)=∞,1)=∞ | ∞ |
| 2 | f(3,3)=3 | max(f(3,1)=1,2)=2 | **2** |
| 3 | f(3,4)=∞ | max(f(3,2)=∞,3)=∞ | ∞ |
| 4 | f(3,5)=∞ | max(f(3,3)=3,4)=4 | **4** |

| i=4 | h=0 | h=1 | h=2 | h=3 | h=4 |
|---|---|---|---|---|---|
| f(4,h) | **1** | ∞ | **2** | ∞ | **4** |

## Final Result

- Target cell: `f(4, 0) = 1`
- Meaning: minimal possible peak = **1**
- Optimal path example: **U D U D**

# Runtime Analysis

- Time Complexity = O(n x m)
  - n = number of movements
  - m = sum of all movements
- Space Complexity = O(m)
  - Space Complexity can be optimized by using one row at a time.

# Further Exploration

| Modification | Adaptation required | Change in DP structure |
|---|---|---|
| Allow negative heights | Relax boundary condition | Domain of h expands |
| Minimize total climb distance | Replace `max()` with `+` | Objective changes |
| Add direction-change penalty | Add direction dimension `d` | State space doubles |
| Limit max building height H | Add transition constraint | Same recurrence |
| Change ending condition | Modify final selection | Same table |
| Count valid paths | Use addition instead of min/max | Switch to counting DP |