GREEDY

VIDEO - 28

Leetcode - 1642

Medium

Facebook
Instagram → codestorywithMIK

Twitter → cswithMIK

→ codestorywithMIK

"You'll learn something new"

Microsoft Google

1642. Furthest Building You Can Reach

Medium | Topics | Companies | Hint

You are given an integer array `heights` representing the heights of buildings, some `bricks`, and some `ladders`.

You start your journey from building `0` and move to the next building by possibly using bricks or ladders.

While moving from building `i` to building `i+1` (**0-indexed**),

- If the current building's height is **greater than or equal** to the next building's height, you do **not** need a ladder or bricks.
- If the current building's height is **less than** the next building's height, you can either use **one ladder** or `(h[i+1] - h[i])` **bricks**.

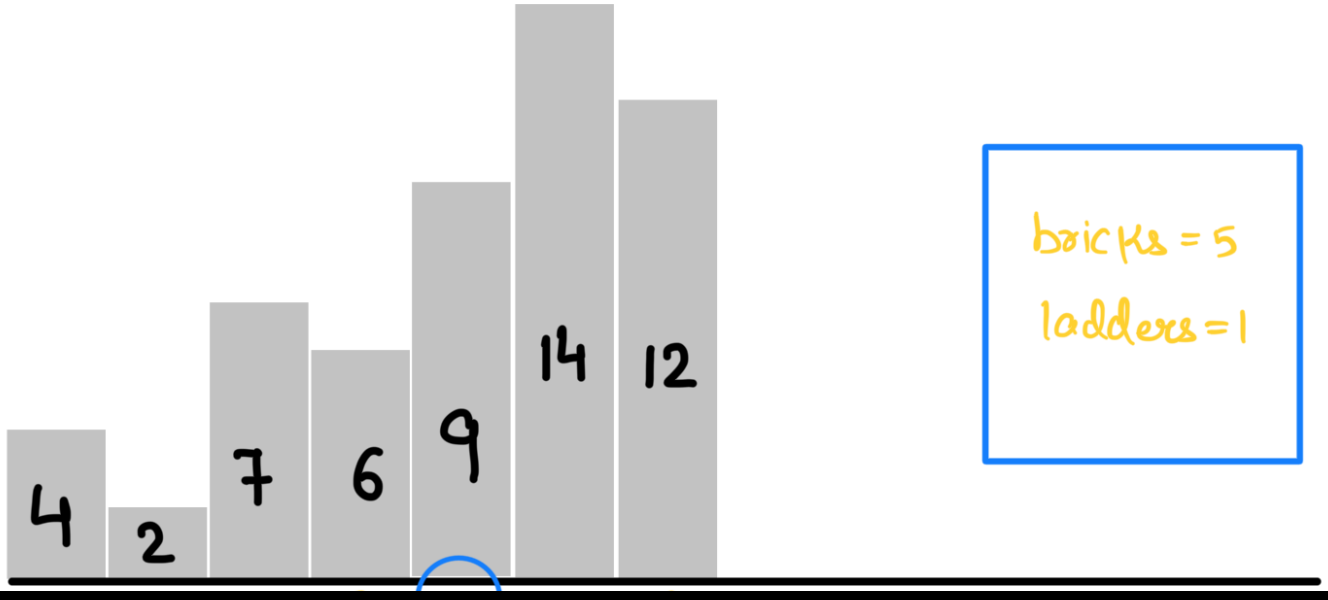
Return the furthest building index (0-indexed) you can reach if you use the given ladders and bricks optimally.

Example :- heights = [4, 2, 7, 6, 9, 14, 12]

bricks = 5

ladders = 1

Output = 4



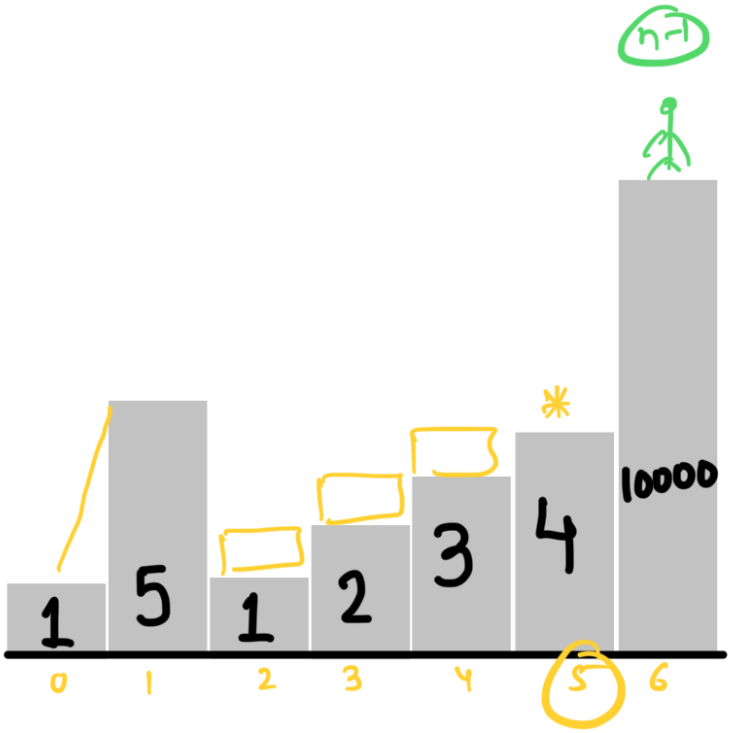
bricks = 5
ladders = 1

0 1 2 3 4 5 6

bricks = (7-2) = 5

return 4

Why Greedy Fails?



Bricks = ~~4~~ ~~3~~ 2 1

ladder = ~~1~~ 0

output = ~~3~~ 5

=> 1 → 5 (4 bricks)

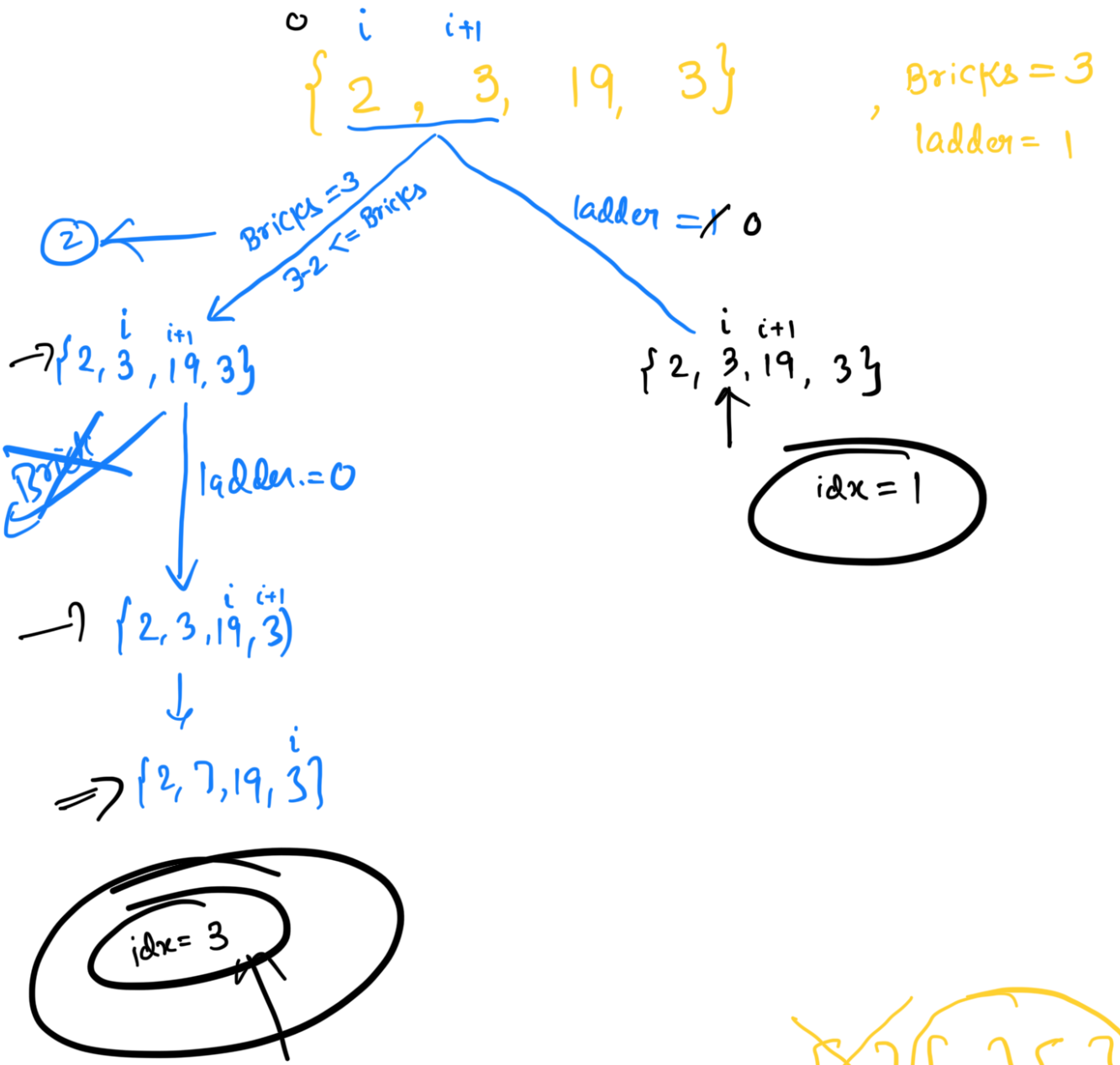
=> 5 → 1

=> 1 → 2 (1 ladder)

} Greedy ✗

Options वाले में Recursion

(Tree Diagram)



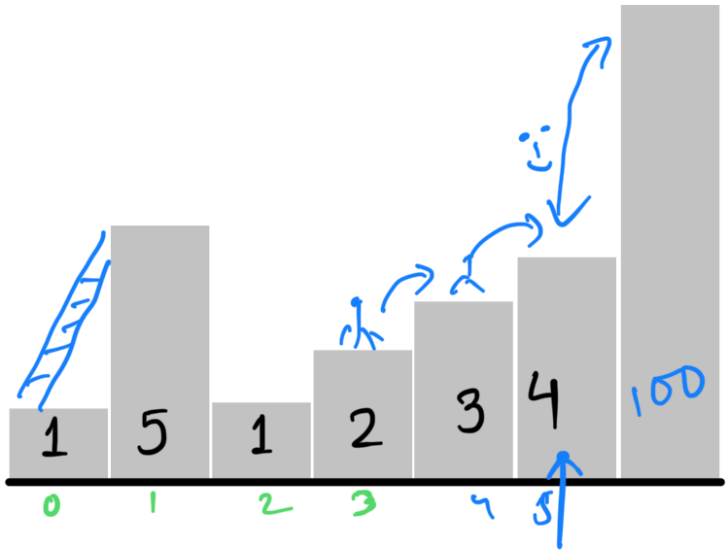

```
int Solve( idx , bricks, ladders) {
    if(idx == n-1) {
        return 0;
    }
```

```
if (heights[idx+1] <= heights[idx]) {
    return 1 + Solve(idx+1, bricks, ladders);
} else { // we need bricks / ladders
    int byBricks = 0;
    int byLadders = 0;
                                  diff
    if (heights[idx+1] - heights[idx] <= bricks) {
        byBricks = 1 + Solve(idx+1, bricks - diff,
                             ladders);
    }

    byLadders = 1 + Solve(idx+1, bricks,
                          ladders-1);
    return max(byBricks, byLadders);
}
}
```

"Lazy Greedy"

Are Bhai Bhai Bhai Bhai Bhai !!!

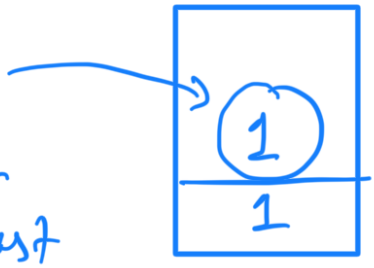


Bricks = ~~4~~ ~~0~~ ~~4~~ ~~2~~ 2

ladder = ~~1~~ 0 2

diff = 100 - 4 = 96

diff > ~~max_past~~



^

Story → code :-

(•) if (heights[i+1] <= hei[i]) {
continue;
}

(•) diff = hei[i+1] - hei[i];

if (bricks >= diff) {
bricks -= diff;
pq.push(diff);
}
else if(ladders > 0) {
max_past_bricks = pq.top();
if (diff < max_past_bricks) {
pq.pop();
bricks += max_past_bricks;
bricks -= diff;
pq.push(diff);
} else {
ladders--;
break;
}
}