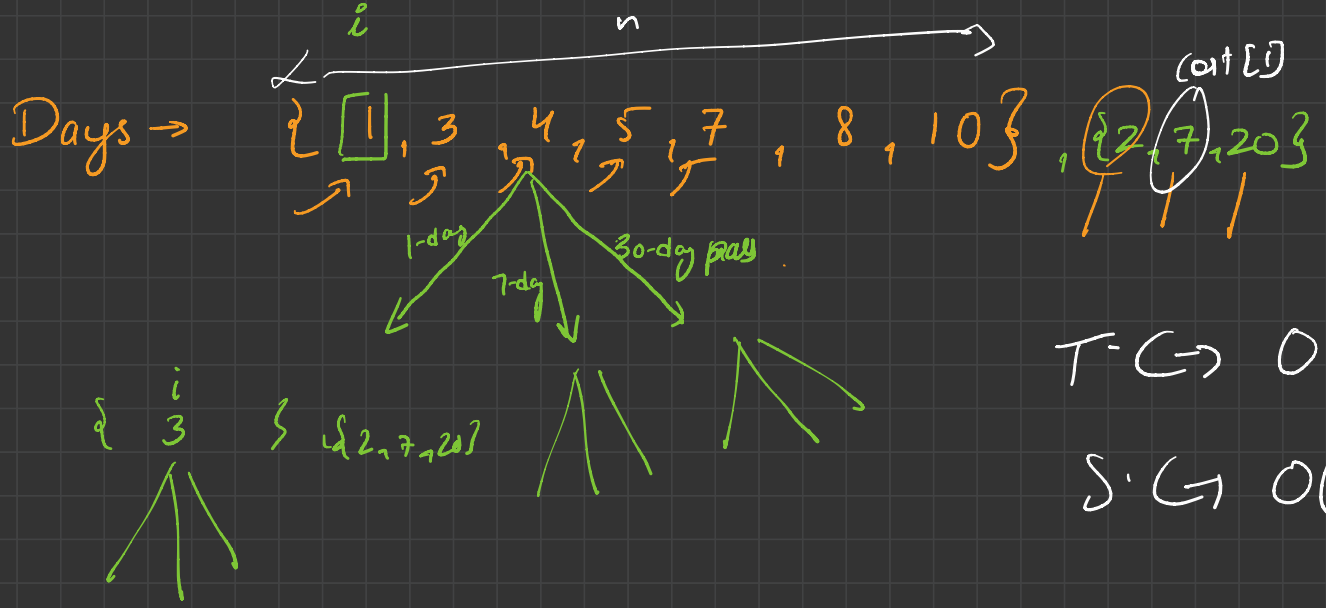
→ Bottom - Up Approach -

T.C → O(n)

S.C → O(n) → O(1)



T.C → O(n)

S.C → O(1)

→ ans → min no of coins

```
int ans = INT_MAX;
```

ans + cost[0]

queue < pair < day, cost till that day > > monthly; → 30 entries

weekly; → 7 entries

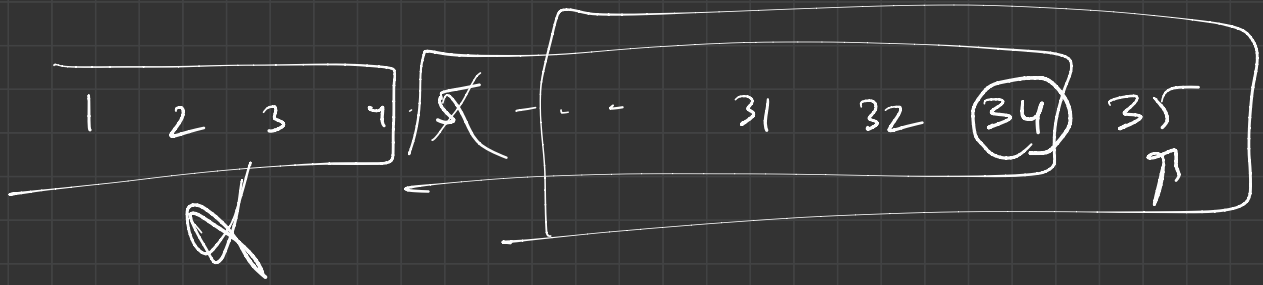
S.C → O(1)

int ans = INT_MAX;

```
for ( day : days)
{
    // 1 step → remove expired days from queue
    while (monthly.size() && monthly.front().first + 30 <= day)
        monthly.pop
    while (weekly.size() && weekly.front().first + 7 <= day)
        weekly.pop()
```

```
// step2 -> push current day's cost
    weekly.push(make_pair(day, ans + cost[1]));
    monthly.push(make_pair(day, ans + cost[2]));

// step3 -> ans update
    ans = min(ans + cost[0], min(monthly.front().second,
                                 weekly.front().second));
  }
  return ans;
}
```


1 2 3 4
31 32 34 35