IOT PHASE 3

```
# Python3 program to implement
# the above approach

# Function to find minimum
# number of fountains to be
# activated



def minCntFoun(a, N):


    # dp[i]: Stores the position of

    # rightmost fountain that can

    # be covered by water of leftmost

    # fountain of the i-th fountain

    dp = [0] * N

    for i in range(N):

      dp[i] = -1


    # Traverse the array
```

```
    for i in range(N):

        idxLeft = max(i - a[i], 0)

        idxRight = min(i + (a[i] + 1), N)

        dp[idxLeft] = max(dp[idxLeft],

                          idxRight)
```

```
# Traverse the array
```

```
for i in range(N):
    idxLeft = max(i - a[i], 0)
    idxRight = min(i + (a[i] + 1), N)
    dp[idxLeft] = max(dp[idxLeft],
                      idxRight)

# Stores count of fountains
# needed to be activated
cntfount = 1

idxRight = dp[0]

# Stores index of next fountain
# that needed to be activated
idxNext = 0

# Traverse dp[] array
```

```
for i in range(N):
    idxNext = max(idxNext,
                  dp[i])

    # If left most fountain
    # cover all its range
    if (i == idxRight):
        cntfount += 1
        idxRight = idxNext
```

```
    # Stores index of next fountain
    # that needed to be activated
    idxNext = 0

    # Traverse dp[] array
```

```
    for i in range(N):
        idxNext = max(idxNext,
                      dp[i])

        # If left most fountain
        # cover all its range
        if (i == idxRight):
            cntfount += 1
            idxRight = idxNext

    return cntfount

# Driver code
if __name__ == '__main__':

    a = [1, 2, 1]
```

```
    N = len(a)

    print(minCntFoun(a, N))

# This code is contributed by Shivam
```

```
        if (i == idxRight):
            cntfount += 1
            idxRight = idxNext

    return cntfount

# Driver code
if __name__ == '__main__':

    a = [1, 2, 1]
```

```
    N = len(a)

    print(minCntFoun(a, N))

# This code is contributed by Shivam
```