

Name - Bimal Tyagi

UID - 22BCS15995

Section - FL603 - A

Advanced Programming LAB - 2 Assignment - 6

Description | **Code** | Editorial | Solutions | Submissions

Java ▾ Auto

```
1 public class Solution {
2     public int hammingWeight(int n) {
3         int res = 0;
4         for (int i = 0; i < 32; i++) {
5             if (((n >> i) & 1) == 1) {
6                 res += 1;
7             }
8         }
9         return res;
10    }
11 }
```

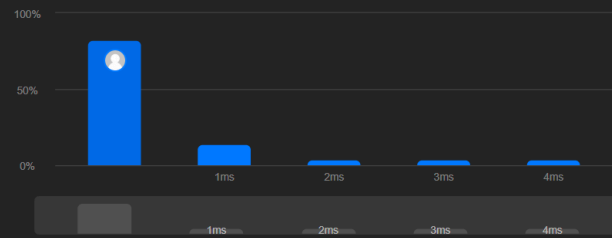
Testcase | **Accepted** | Test Result

All Submissions

Runtime
0 ms | Beats 100.00%

Memory
40.84 MB | Beats 42.31%

Analyze Complexity



Runtime	Memory
0 ms	40.84 MB
1 ms	
2 ms	
3 ms	
4 ms	

Code | Java

```
public class Solution {
    public int hammingWeight(int n) {
        int res = 0;
        for (int i = 0; i < 32; i++) {
            if (((n >> i) & 1) == 1) {
                res += 1;
            }
        }
    }
}
```

View more

Description | **Code** | Editorial | Solutions | Submissions

Python3 ▾ Auto

```
1 class Heap:
2     def __init__(self, nums):
3         self.H = nums
4         self.heapify()
5         print(self.H)
6     def bubbleDown(self, index):
7         if 2 * index <= len(self.H) - 1 and 2 * index + 1 <= len(self.H) -
8             1:
9             if self.H[2 * index] <= self.H[2 * index + 1]:
10                replace = 2 * index
11            else:
12                replace = 2 * index + 1
13            if self.H[index] > self.H[2 * index] or self.H[index] > self.H
14                [2 * index + 1]:
15                self.H[index], self.H[replace] = self.H[replace], self.H
16                [index]
17                self.bubbleDown(replace)
18            elif 2 * index <= len(self.H) - 1:
19                if self.H[index] > self.H[2 * index]:
20                    self.H[index], self.H[2 * index] = self.H[2 * index], self
21                    .H[index]
22                    self.bubbleDown(2 * index)
23
24     def heapify(self):
25         for i in range(len(self.H) // 2, -1, -1):
26             self.bubbleDown(i)
27
28     def sort(self):
29         arr = []
30         while len(self.H) != 0:
31             val = self.H[0]
32             self.H[0] = self.H[len(self.H) - 1]
33             self.H.pop()
34             if self.H:
35                 self.bubbleDown(0)
```

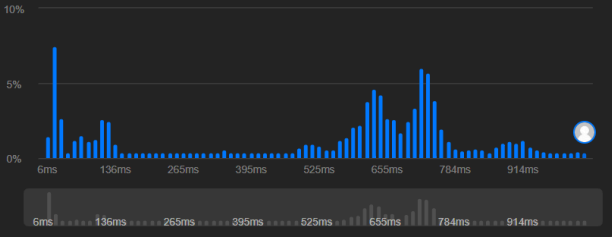
Testcase | **Accepted** | Test Result

All Submissions

Runtime
2011 ms | Beats 5.01%

Memory
24.17 MB | Beats 84.92%

Analyze Complexity

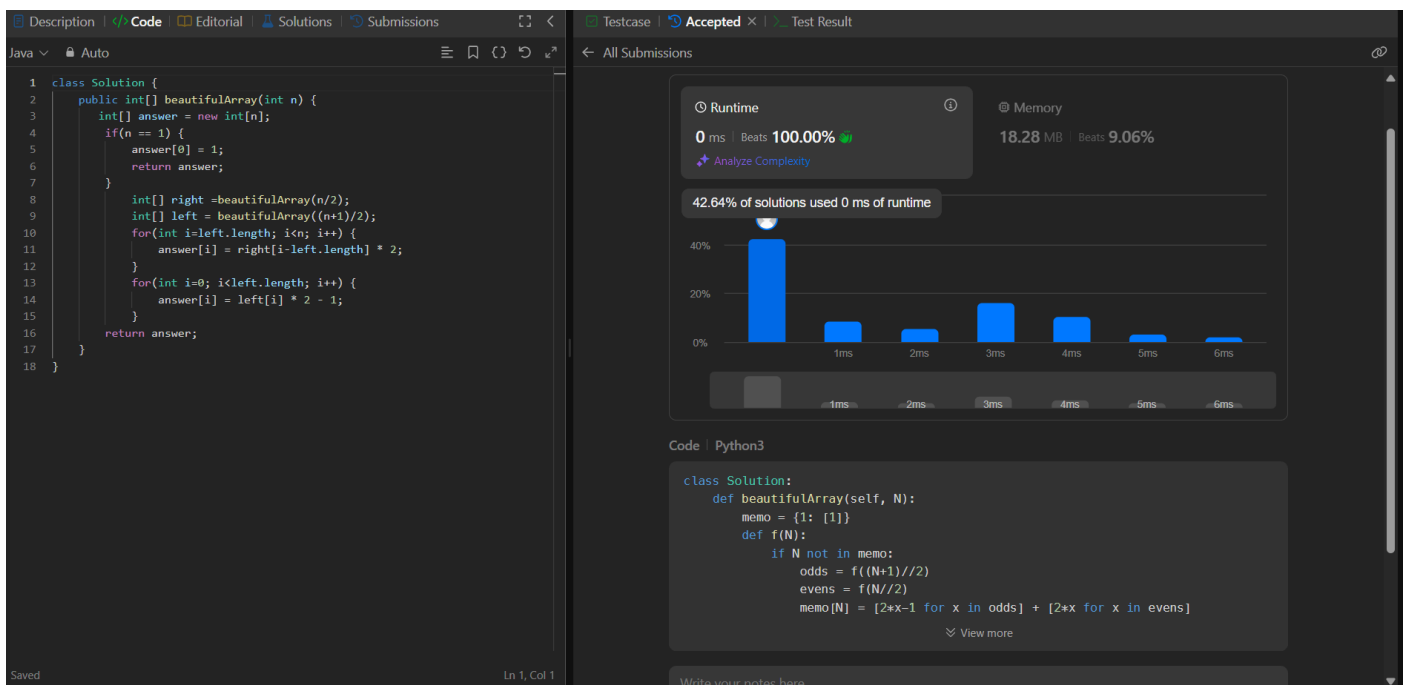
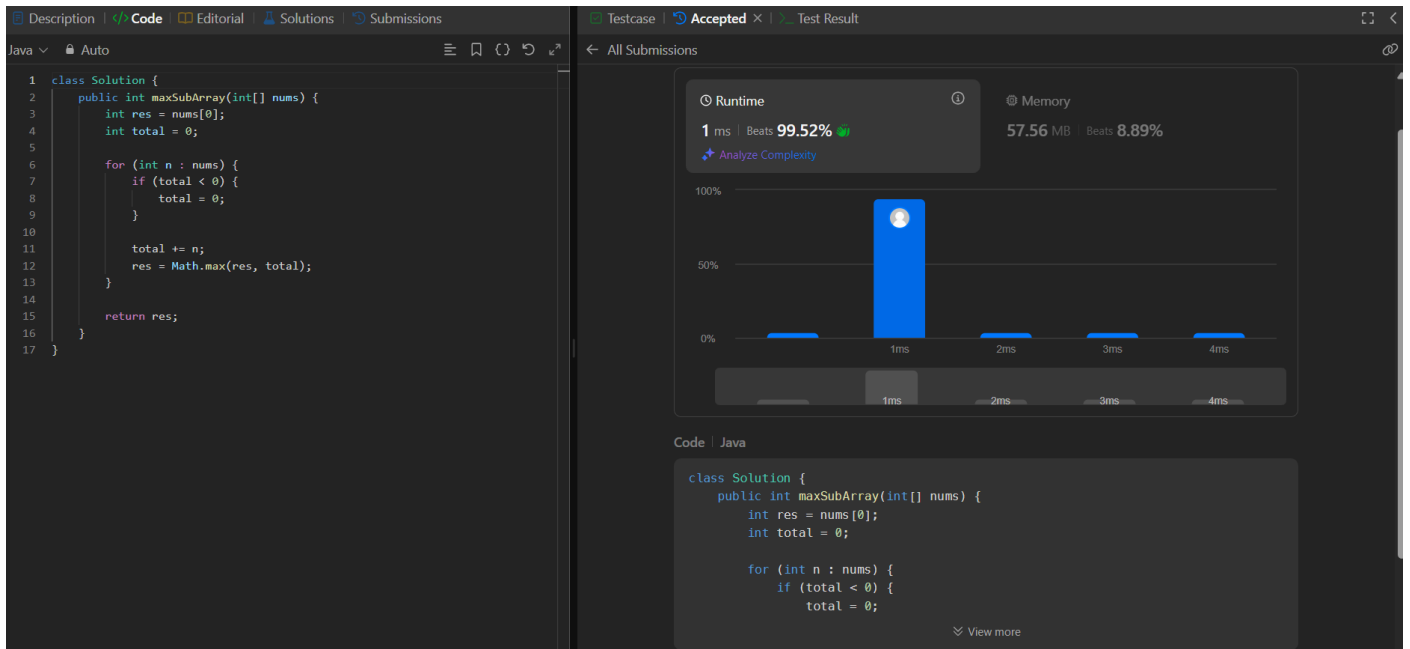


Runtime	Memory
2011 ms	24.17 MB
1 ms	
2 ms	
3 ms	
4 ms	

Code | Python3

```
class Heap:
    def __init__(self, nums):
        self.H = nums
        self.heapify()
        print(self.H)
    def bubbleDown(self, index):
        if 2 * index <= len(self.H) - 1 and 2 * index + 1 <= len(self.H) - 1:
            if self.H[2 * index] <= self.H[2 * index + 1]:
                replace = 2 * index
            else:
                replace = 2 * index + 1
            if self.H[index] > self.H[replace]:
                self.H[index], self.H[replace] = self.H[replace], self.H[index]
                self.bubbleDown(replace)
        elif 2 * index <= len(self.H) - 1:
            if self.H[index] > self.H[2 * index]:
                self.H[index], self.H[2 * index] = self.H[2 * index], self.H[index]
                self.bubbleDown(2 * index)
        elif 2 * index + 1 <= len(self.H) - 1:
            if self.H[index] > self.H[2 * index + 1]:
                self.H[index], self.H[2 * index + 1] = self.H[2 * index + 1], self.H[index]
                self.bubbleDown(2 * index + 1)
        else:
            return
    def heapify(self):
        for i in range(len(self.H) // 2, -1, -1):
            self.bubbleDown(i)
    def sort(self):
        arr = []
        while len(self.H) != 0:
            val = self.H[0]
            self.H[0] = self.H[len(self.H) - 1]
            self.H.pop()
            if self.H:
                self.bubbleDown(0)
```

View more



Accepted 57 / 57 testcases passed

BimalTyagi submitted at Feb 10, 2025 15:16

Solution

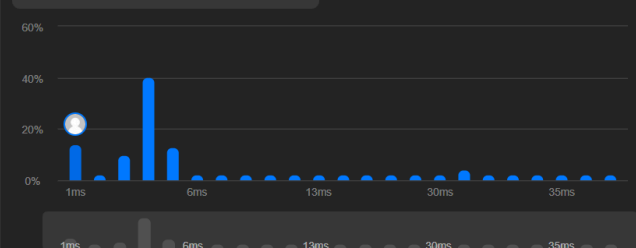
Runtime

1 ms | Beats 100.00%

Analyze Complexity

Memory

44.72 MB | Beats 22.91%



Code | Java

```
class Solution {
    public int superPow(int a, int[] b) {
        if (a % 1337 == 0) return 0;
        int p = 0;
        for (int i : b) p = (p * 10 + i) % 1140;
        if (p == 0) p += 1440;
        return power(a, p, 1337);
    }
    public int power(int a, int n, int mod) {
        a %= mod;
        int ret = 1;
        while (n != 0) {
            if ((n & 1) != 0) ret = ret * a % mod;
            a = a * a % mod;
            n >>= 1;
        }
        return ret;
    }
}
```

Accepted

All Submissions

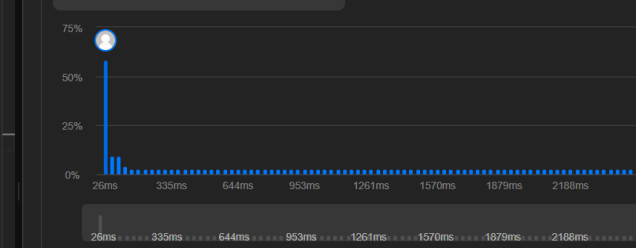
Runtime

23 ms | Beats 89.56%

Analyze Complexity

Memory

25.53 MB | Beats 31.32%



Code | Python3

```
class Solution:
    def getSkyline(self, buildings: List[List[int]]) -> List[List[int]]:
        x_height_right_tuples = sorted([(L, -H, R) for L, R, H in buildings] + [(R, 0, "doesn't
        matter") for _, R, _ in buildings])
        result, max_heap = [[0, 0]], [(0, float('inf'))]
        for x, negative_height, R in x_height_right_tuples:
            while x >= max_heap[0][1]:
                heapq.heappop(max_heap)
            if negative_height:
                heapq.heappush(max_heap, (negative_height, R))
            curr_max_height = -max_heap[0][0]
            if result[-1][1] != curr_max_height:
                result.append([x, curr_max_height])
        return result[1:]
```