Island.java – gettingWatter (method) Algorithm

Theoretical Analysis

To know the performance of my code and how effective or fast it is, I put the following inputs:

- **Input Sizes (Array's length):** 0, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192.
- **No. of Tests:** 9,999,999 (I tested this big quantity to make the average time more precise and see which were the effects of looping thousands of times). Moreover, this input was the most my computer could handle.
- It should be noted that the height of the island can have a range of 1 to 10,000 units.

Just observing my code, I think it will not take long to complete the tasks, which are, first, to identify the largest peaks of my arrangement from left to right and from right to left and, second, to begin calculating the volume of water (if possible) found on the island.

I also know that the time for my code to complete the task will depend on the size I choose for the array. For example, if my array has a size of 0, then the process will be instantaneous, but if I choose a very large size, it may even take seconds to complete the task.

Non-repeating (outside for-loop)

- Condition
- Initialize waterCollected
- Initialize possibleWater
- Initialize left
- Initialize right
- Initialize maxBarLeftToRight
- Initialize maxBarRightToLeft
- Return waterCollected

Total: 8

Non-repeating (1st for-loop)

- Initialize i
- Last check

How Many Times (1st for loop)?

- N times

Total: 2 + 6N

Repeating (1st for-loop)

- Condition (i < island.length)
- If Condition (island[i] < left)
- Else Condition
- If Condition (island[island.length – 1 – i] < right)
- Else Condition
- Change in i

Non-repeating (2nd for-loop)

- Initialize i
- Last check

How Many Times (2nd for-loop)?

- N times

Total: 2 + 7N

Repeating (2nd for-loop)

- Condition (i < island.length)
- If Condition (maxBarLeftToRight[i] != 0) &&
- If Condition (maxBarRightToLeft[i] != 0)
- If Condition (maxBarLeftToRight[i] < maxBarRightToLeft[i])
- Else Condition
- Change in waterCollected
- Change in i

```
public static int gettingWater(int[] island){

    if(island.length < 3){ (1)
        return 0;
    }

    int waterCollected = 0; (1)
    int possibleWater = 0; (1)

    int left = island[0]; (1)
    int right = island[island.length - 1]; (1)

    int[] maxBarLeftToRight = new int[island.length]; (1)
    int[] maxBarRightToLeft = new int[island.length]; (1)

    for(int i = 1; i < island.length; i++){      (1) (1) (n) (n)
        if(island[i] < left){                    (n)
            maxBarLeftToRight[i] = left;
        } else{                                  (n)
            left = island[i];
        }
        if(island[island.length - 1 - i] < right){   (n)
            maxBarRightToLeft[island.length - 1 - i] = right;
        } else{                                  (n)
            right = island[island.length - 1 - i];
        }
    }

    for(int i = 0; i < island.length; i++){      (1) (1) (n) (n)
        if( (maxBarLeftToRight[i] != 0 ) && (maxBarRightToLeft[i] != 0) ){   (n) (n)
            if(maxBarLeftToRight[i] < maxBarRightToLeft[i]){   (n)
                possibleWater = maxBarLeftToRight[i];
            } else{                              (n)
                possibleWater = maxBarRightToLeft[i];
            }
            waterCollected += possibleWater - island[i];   (n)
        }
    }
    return waterCollected;   (1)
}
```

These are constants (if there was a for-loop in it, it would change everything)

n times

2 + 6N

This is going to check n times

n times

2+7n

Outside for-loop = 8

Time complexity:

$8 + 6N + 2 + 7n + 2 =$

$13N + 12 =$

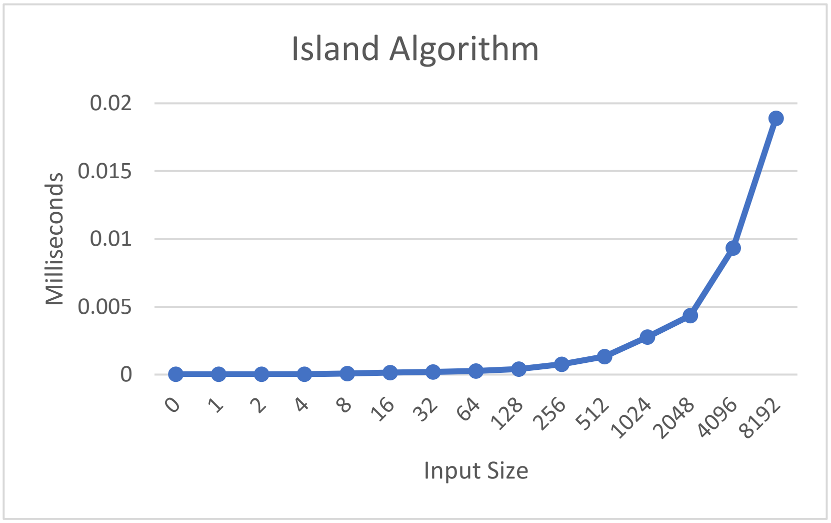
$O(N)$

Experimental Analysis

Input Size: 0	**Average Time:** 0.00002 ms	**No. of Tests:** 9,999,999	**Running Time:** Instant	**Input Size:** 128	**Average Time:** 0.0004 ms	**No. of Tests:** 9,999,999	**Running Time:** 1 min
Input Size: 1	**Average Time:** 0.00002 ms	**No. of Tests:** 9,999,999	**Running Time:** Instant	**Input Size:** 256	**Average Time:** 0.00076 ms	**No. of Tests:** 9,999,999	**Running Time:** 1 min
Input Size: 2	**Average Time:** 0.00002 ms	**No. of Tests:** 9,999,999	**Running Time:** Instant	**Input Size:** 512	**Average Time:** 0.00134 ms	**No. of Tests:** 9,999,999	**Running Time:** 2 min
Input Size: 4	**Average Time:** 0.00004 ms	**No. of Tests:** 9,999,999	**Running Time:** Instant	**Input Size:** 1024	**Average Time:** 0.00277 ms	**No. of Tests:** 9,999,999	**Running Time:** 5 min
Input Size: 8	**Average Time:** 0.00007 ms	**No. of Tests:** 9,999,999	**Running Time:** Instant	**Input Size:** 2048	**Average Time:** 0.00436 ms	**No. of Tests:** 9,999,999	**Running Time:** 9 min
Input Size: 16	**Average Time:** 0.00014 ms	**No. of Tests:** 9,999,999	**Running Time:** Instant	**Input Size:** 4096	**Average Time:** 0.00934 ms	**No. of Tests:** 9,999,999	**Running Time:** 18 min
Input Size: 32	**Average Time:** 0.00019 ms	**No. of Tests:** 9,999,999	**Running Time:** Instant	**Input Size:** 8192	**Average Time:** 0.01891 ms	**No. of Tests:** 9,999,999	**Running Time:** 36 min
Input Size: 64	**Average Time:** 0.00026 ms	**No. of Tests:** 9,999,999	**Running Time:** Instant				

At the same time I ran and did tests with my code (with the previous sizes), I had more than 7 tabs open in two Google Chromes browsers, a PowerPoint presentation, 2 Word documents, and an Excel sheet. Perhaps, that is one of the factors why my code took more than 10 min at the time that the size of the arrangement was very large, in fact, at the time I put the size 8192 in the array I had to wait more than 30 minutes to get the average time.

Moreover, I was also surprised to see the graph of my code on how it was really behaving. Because I made this graph I could know what kind of function my code resembles and from what I can see is that it is similar to a time of complexion of O(n^2). Although I am still not clear about why in my theoretical analysis I came to the conclusion that my code is O (N), while in the experimental analysis and thanks to the graph I could see that my code has an O(N^2) function. I feel that the answer to my previous question is that in my theoretical analysis I had as a result an O (N) + O (N), but to simplify it is O(N).


Island Algorithm
Milliseconds
Input Size