Istanbul Technical University
Faculty of Computer and Informatics
Computer Engineering Department

BLG335E
HW2
Report

Oğuzhan Aybar - 150200077

December 2nd, 2022

Contents

1 Introduction

This is a project of a data storage and processing system for big inputs which aims to make clear the difference between the closest complexities.

2 General Structure

2.1 Main Data Structure

The main data structure is a specialty binary tree. Main idea is that the left branch of a parent is smaller than the parent and right branch is bigger than the parent. This structure has an average search complexity of O(lgn) and insert complexity of O(1). Sadly, because that this is not a self balancing binary tree -which was the intended structure which was cut off due to time limits- this tree has a worst case search complexity of O(n) which is when the input is ordered. This actually creates a double linked list instead of a binary tree.

2.2 Nodes

In a project like this with a not self balancing binary tree, in order to save some time and memory, every node has an overlap value which is incremented when the same value enters the tree twice. Also every node knows the branch count on every side of itself which were planned to be used in order to balance the tree.

3 Algorithms

It is essential that we decrease the complexity of the PRINT function while we keep the ADD function as basic as possible. So as long is it is not strictly required, we tried to do everything at ADD function which includes mean, Q1, median, Q3, max and, min. All of these are calculated by functions of O(1) at every ADD which sums up to O(n).

3.1 Algorithms Implemented

ADD: Adds a new node to a tree, updates Nodecount, Sum, Min, Max then calls UpdateQ. O(1)
UpdateQ: Updates the saved Q nodes and Median according to new input and it's value. O(1)
InorderRecursiveDifferenceSum: Essentially is a In Order Traversal. Along the way, saves a sum of differences from mean value. O(n)
GetForwardNode: Returns the next node in in order traversal. O(n)
GetBackwardNode: Returns the before node in in order traversal. O(n)
GoForward: Changes a Node** value to the next node in in order traversal using GetForwardNode. O(n)

GoBackward: Changes a Node** value to the before node in in order traversal using GetBackwardNode. O(n)

3.2 Mean

Sum of all nodes and node count are saved into values called ”sum” and ”NodeCount” at every ADD function in order to lower the overall complexity. Counting the nodes and summing them all at every PRINT would take O(n) * Print Count overall which can be assumed as O(n^2). By saving every ADD with O(1), we generated a complexity O(n).

3.3 STD

As stated before, mean is a value that is easily accessible. In order to find the STD, In order traversal was chosen because of it’s simplicity when used recursively.

Algorithm 1 Recursive InOrder Traversal

```
function InOrderTraversal(Node, mean, *Output)

    if Left child is not null then
        InOrderTraversal(Node.leftChild, mean, output)
    end if

output += (mean - Node.value) * (mean - Node.value)

    if Right child is not null then
        InOrderTraversal(Node.rightChild, mean, output)
    end if

end function
```

Complexity of this code is O(n) which is because it traverses all the nodes which has a count of n. This function is only called on PRINT only when the user asks for it.

3.4 Min Max

Min and Max are compared and decided on every add to save time. this only adds O(n) complexity to code which is actually larger than finding it on the tree which is at worst O(n) but depending on print count, this seemed more logical and fast.

3.5 Q1 Median Q3

As finding these values needed a traversal of the tree, I have decided to create an algorithm to keep track of changes in these values and moves the pointers accordingly. This code single handedly took half of the development time and was made even more difficult by overlapping branches. And it wasn't even that complex.

Every node added to the tree shifts Q1 0.25 node to the end, median 0.5 node to the end, Q3 0.75 node to the end. using the value of the new node and comparing them to the nodes that are just under or the Q's, we can deduce the movement we should cause in these pointers. this takes O(1) at every add which is n times which means the complexity is a singular O(n) instead of an O(n) at every print. As this is not a really accurate statement, this is just an example and simplification of the situation.

Algorithm 2 Q Update

```
function UpdateQ(value)

    if value is lower than QNodeValue then

        if Whole region of QIndex not Changed then
            MoveQBackwards
        end if
    end if
    if Whole region of QIndex Changed then
        MoveQForwards
    end if
end function
```

Because that this function adds a lot of "if" statement to the code, I have chosen not to run it when the user didn't ask for it.

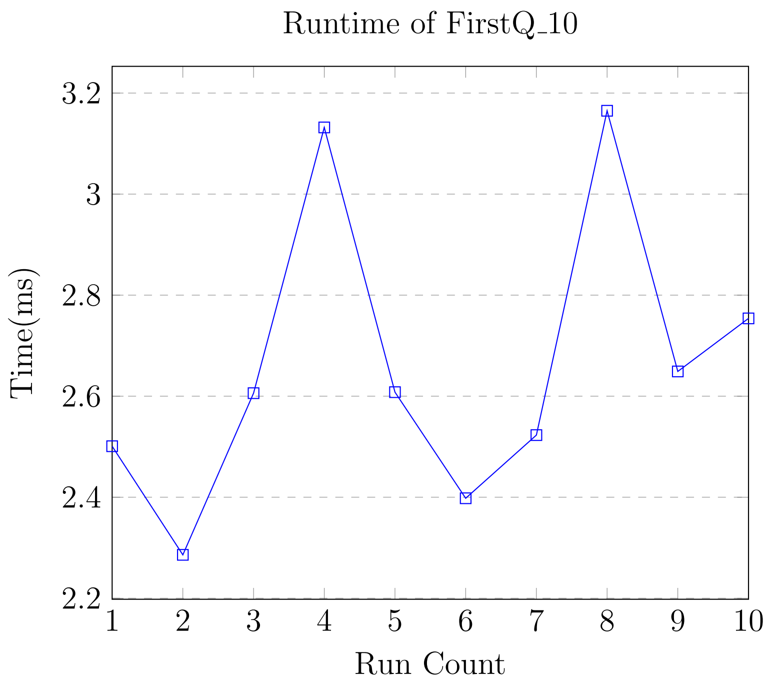
4 Can I Use Sliding Window Based Approach

I have not created any nested loop so I strongly believe that Sliding Window Based Approach is possible which eliminates nested loops and lowers the complexity. It could be said that if PRINT is a loop, Std finding algorithm would be a nested loop. In this case I don not believe there is an easy way to eliminate this nested loop.

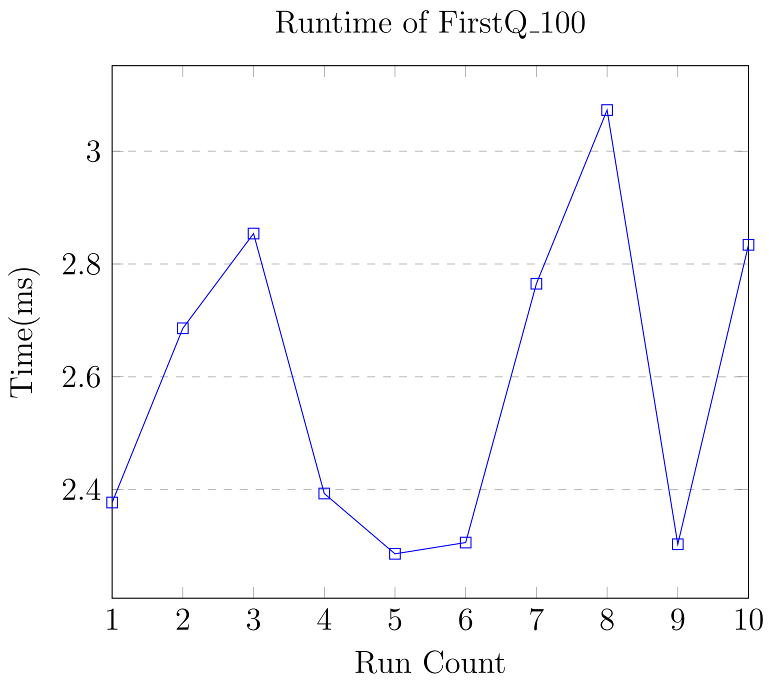
5 Plots and Runtime Values

The functions and how many times they have been called are given after the plots.

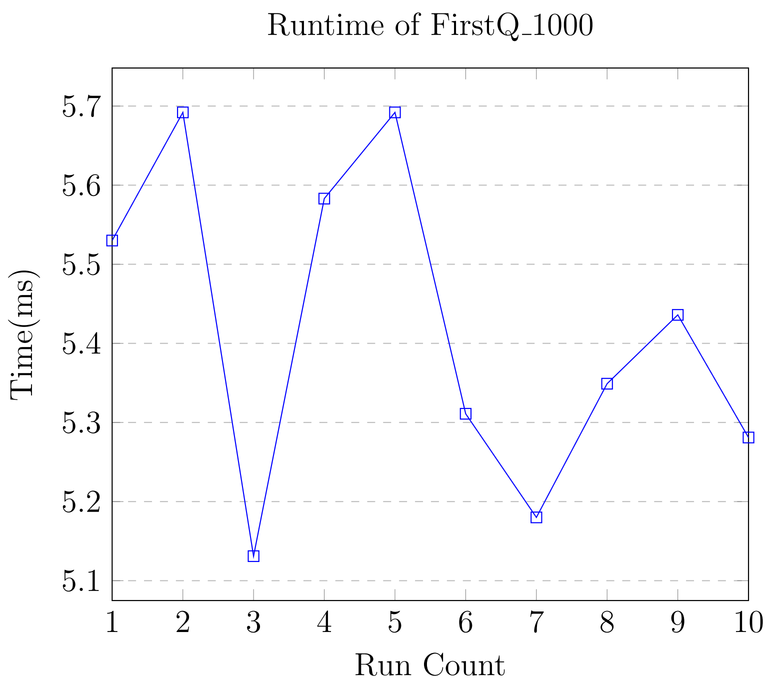
5.1 FirstQ Plots



for 10 inputs;
add : 9
InorderRecursiveDifferenceSum : 0
GetForwardNode : 33
GetBackwardsNode : 9
GoForward : 6
GoBackward : 9
UpdateQ : 9

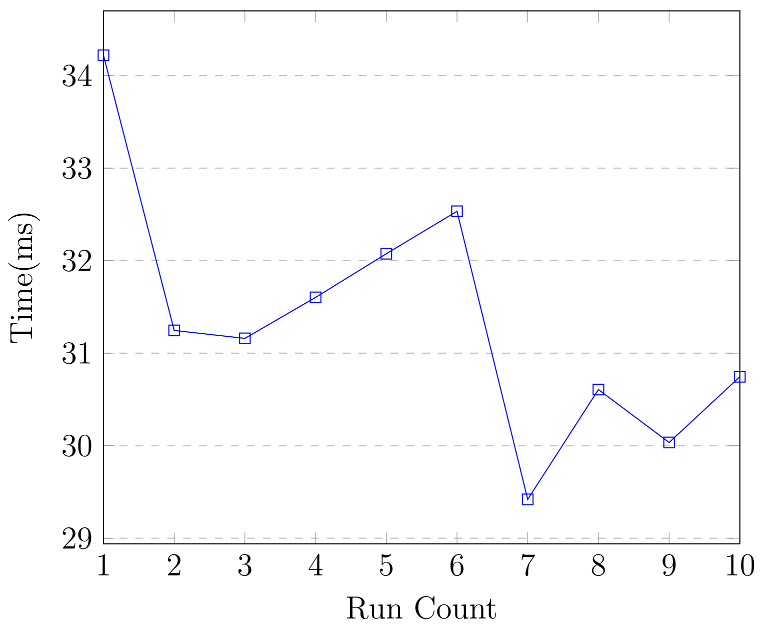


for 100 inputs;
add : 99
InorderRecursiveDifferenceSum : 0
GetForwardNode : 360
GetBackwardsNode : 47
GoForward : 63
GoBackward : 47
UpdateQ : 99



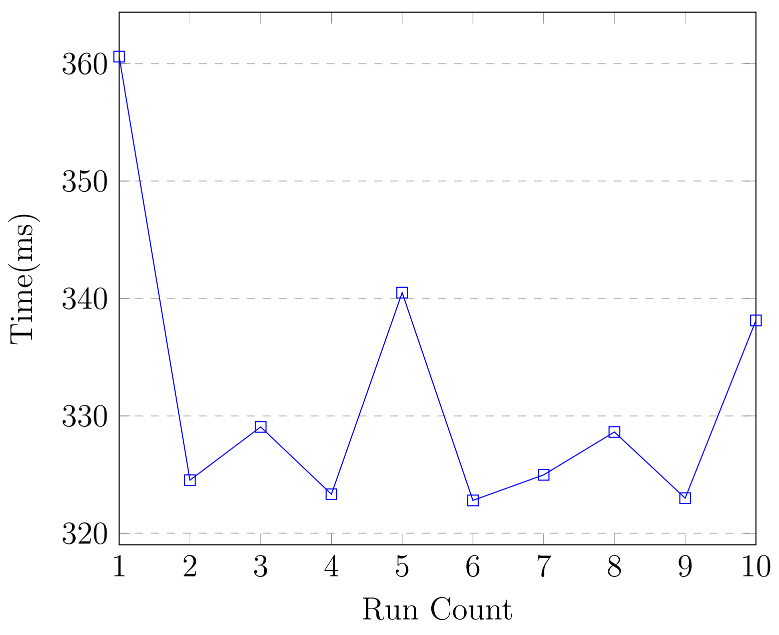
for 1000 inputs;
add : 999
InorderRecursiveDifferenceSum : 0
GetForwardNode : 3630
GetBackwardsNode : 469
GoForward : 633
GoBackward : 469
UpdateQ : 999

Runtime of FirstQ_10000



for 10000 inputs;
add : 9999
InorderRecursiveDifferenceSum : 0
GetForwardNode : 36285
GetBackwardsNode : 5934
GoForward : 6288
GoBackward : 5934
UpdateQ : 9999

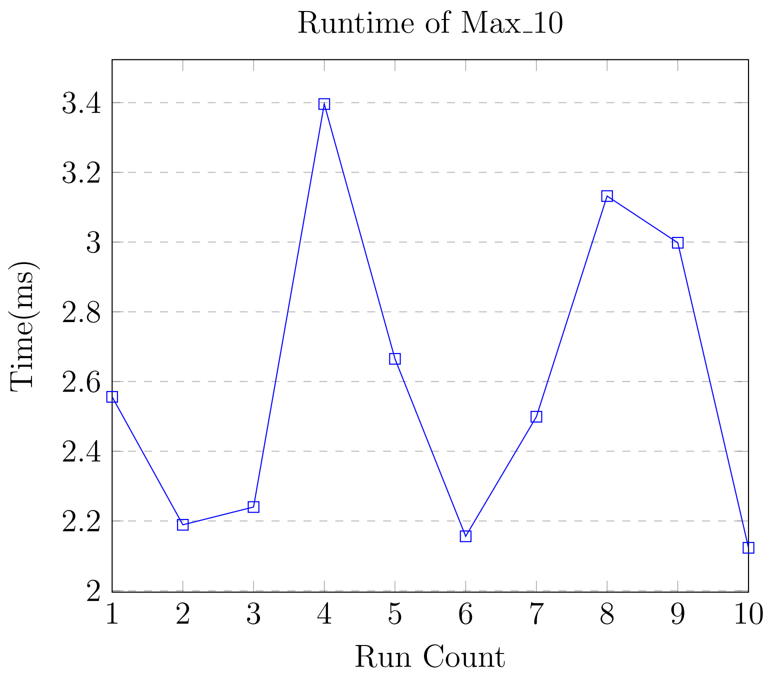
Runtime of FirstQ_100000



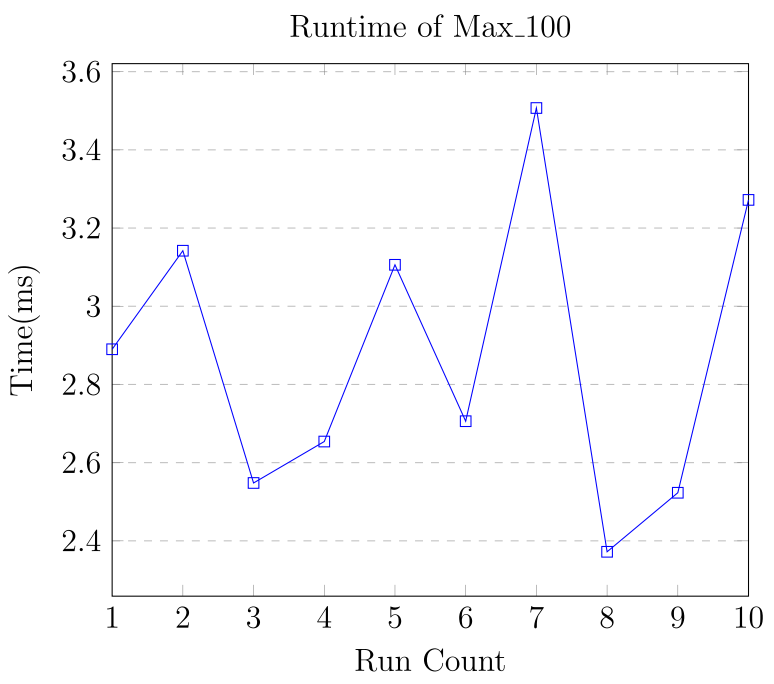
for 100000 inputs;
add : 99995
InorderRecursiveDifferenceSum : 0

GetForwardNode : 363465
GetBackwardsNode : 51069
GoForward : 63480
GoBackward : 51069
UpdateQ : 99995

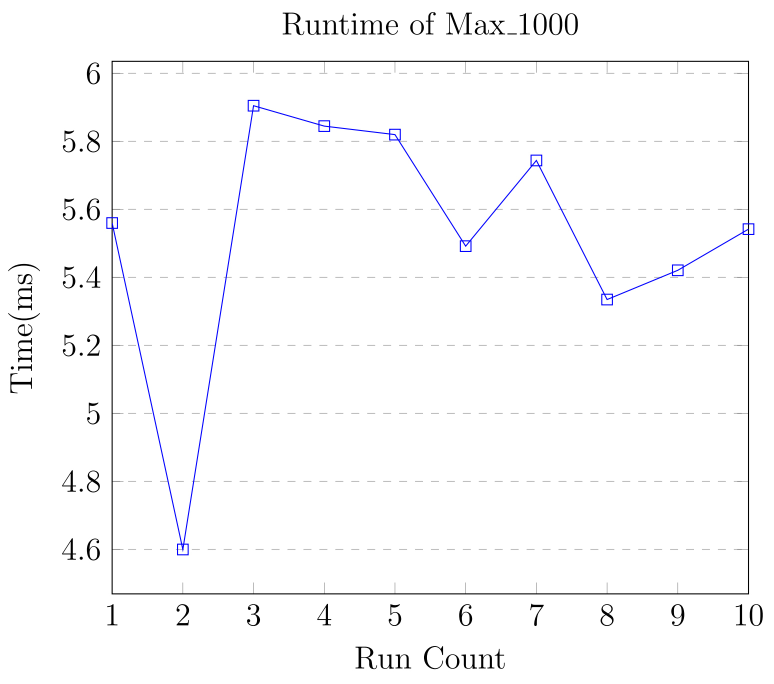
5.2 Max Plots



for 10 inputs;
add : 9
InorderRecursiveDifferenceSum : 0
GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0

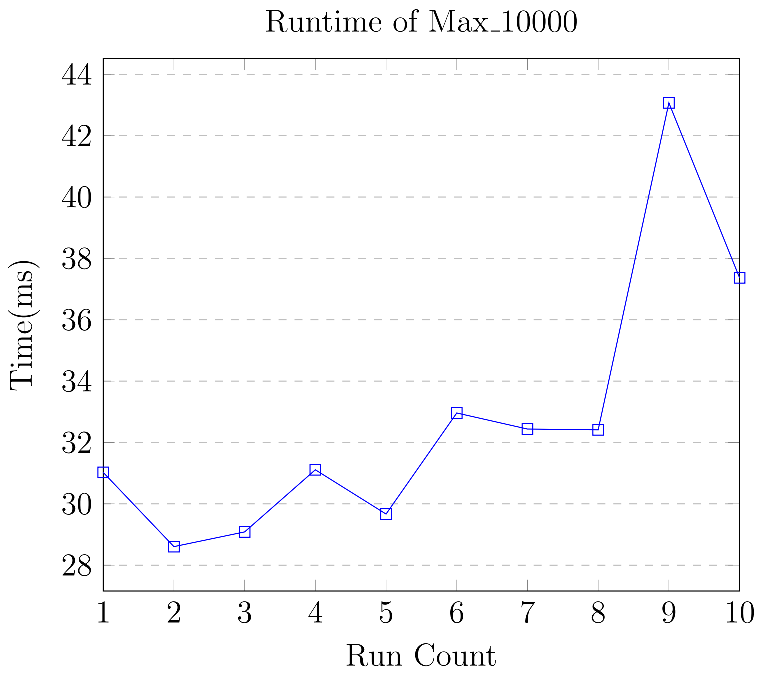


for 100 inputs;
add : 99
InorderRecursiveDifferenceSum : 0
GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0

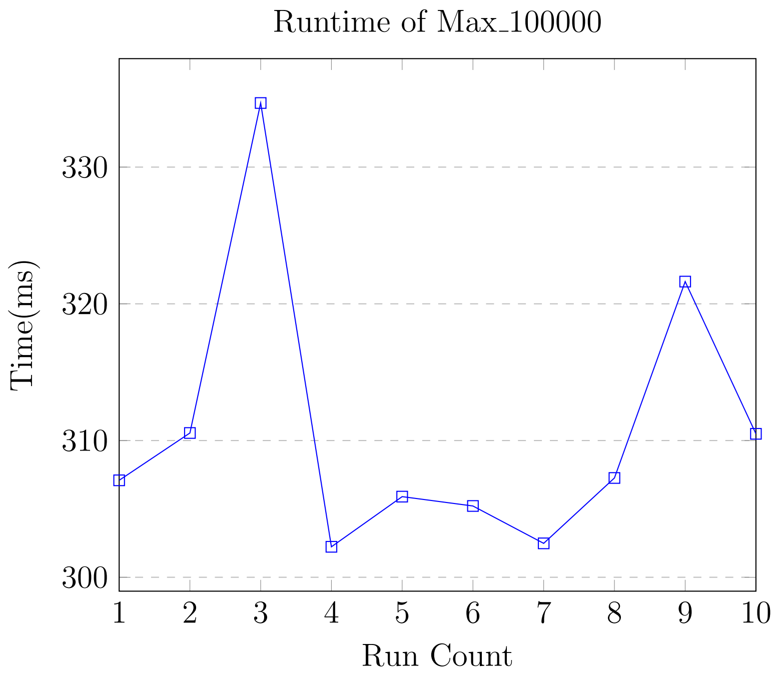


for 1000 inputs;
add : 999
InorderRecursiveDifferenceSum : 0

GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0

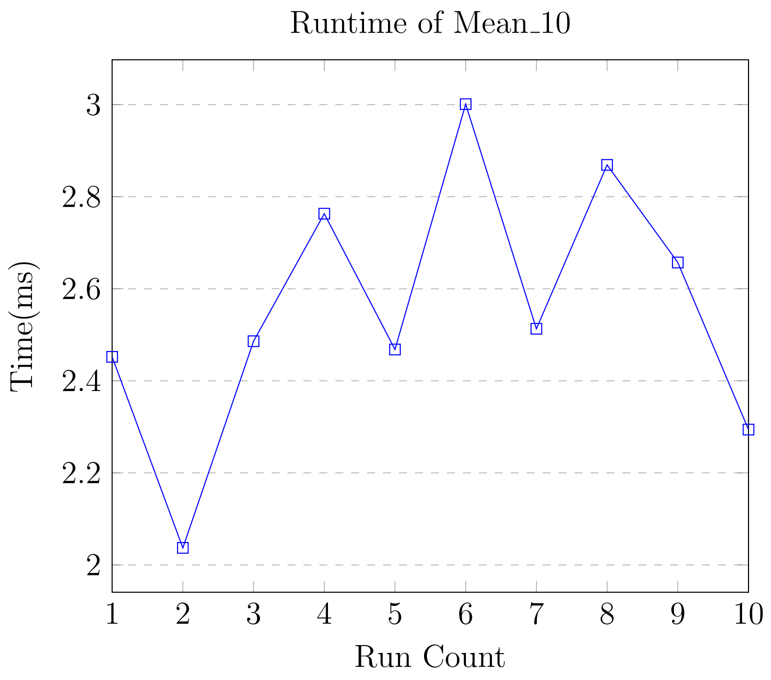


for 10000 inputs;
add : 9999
InorderRecursiveDifferenceSum : 0
GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0

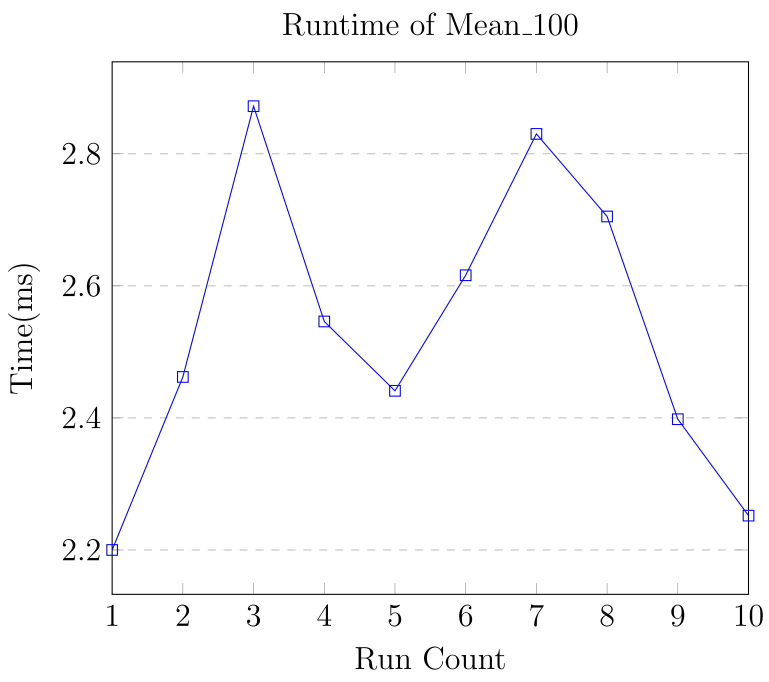


for 100000 inputs;
add : 99995
InorderRecursiveDifferenceSum : 0
GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0

5.3 Mean Plots

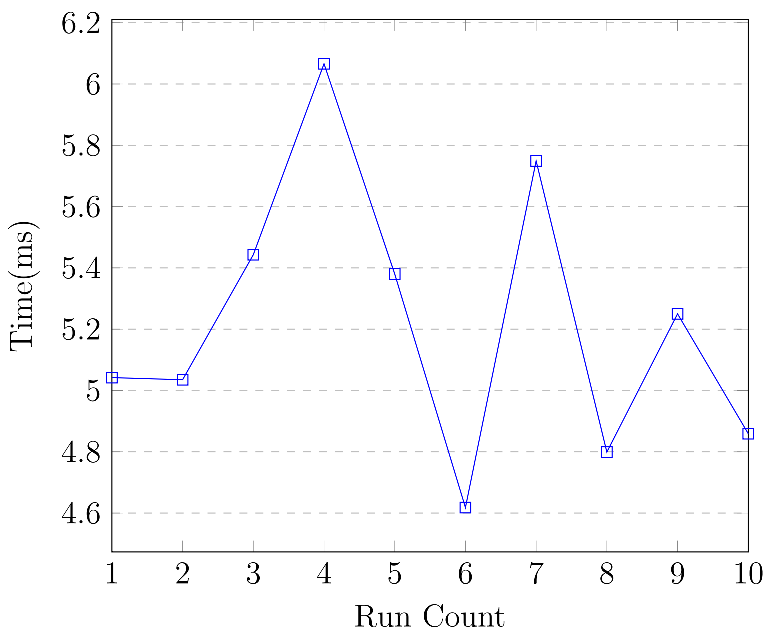


for 10 inputs;
add : 9
InorderRecursiveDifferenceSum : 0
GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0



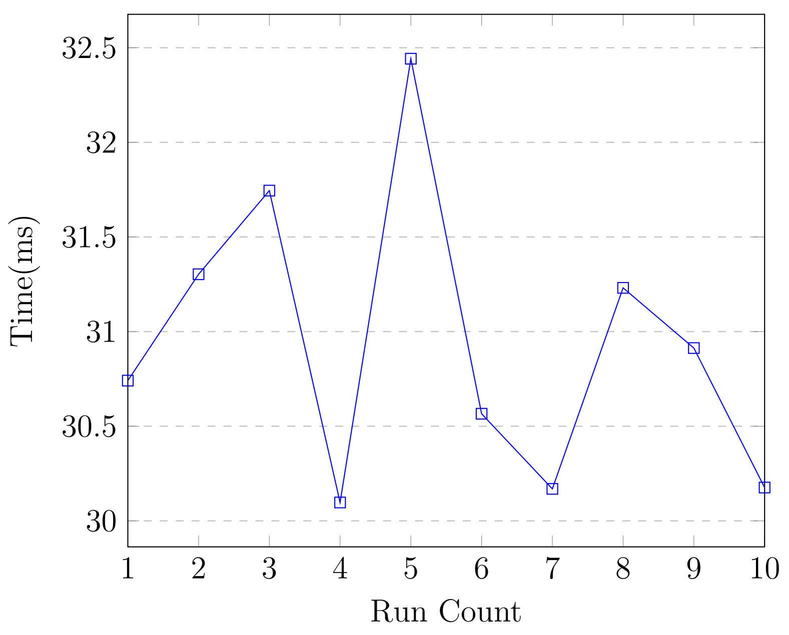
for 100 inputs;
add : 99
InorderRecursiveDifferenceSum : 0
GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0

Runtime of Mean_1000



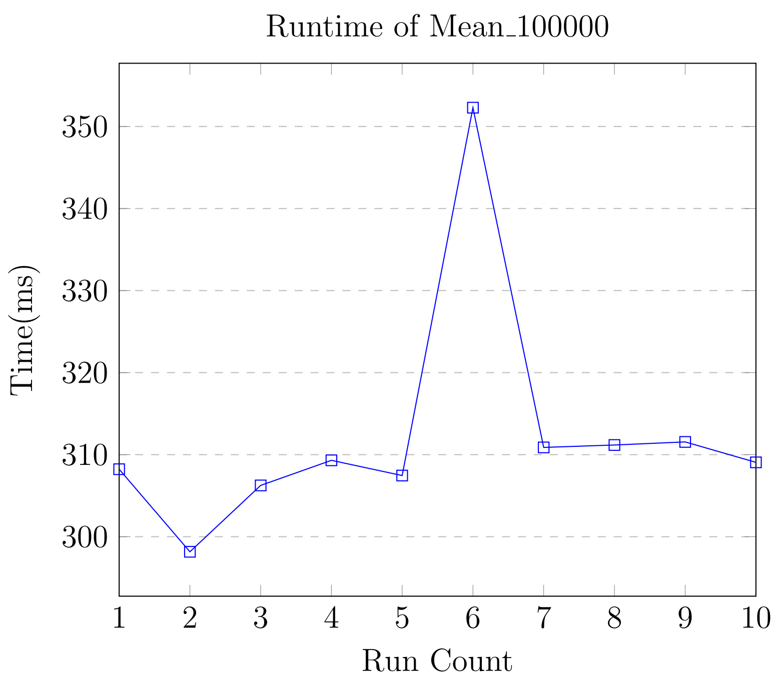
for 1000 inputs;
add : 999
InorderRecursiveDifferenceSum : 0
GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0

Runtime of Mean_10000



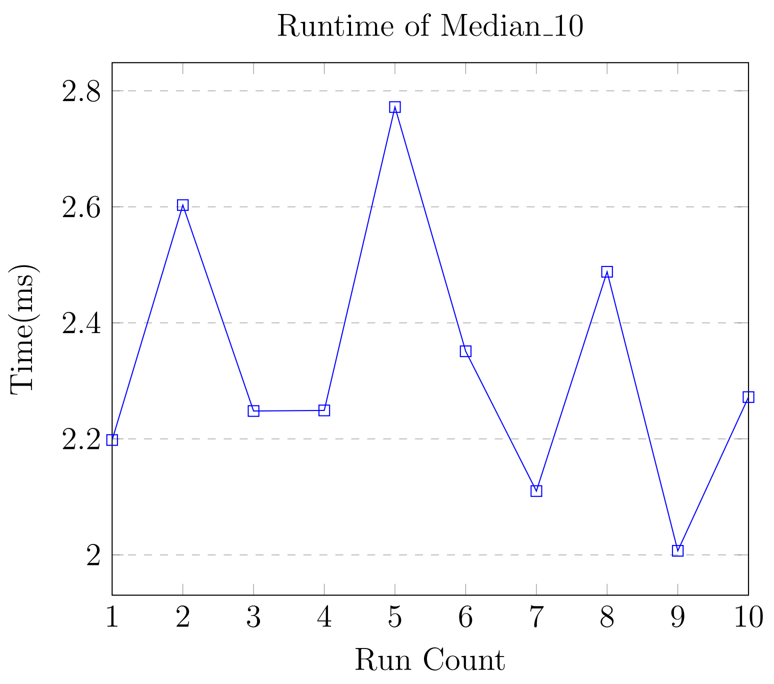
for 10000 inputs;
add : 9999
InorderRecursiveDifferenceSum : 0

GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0

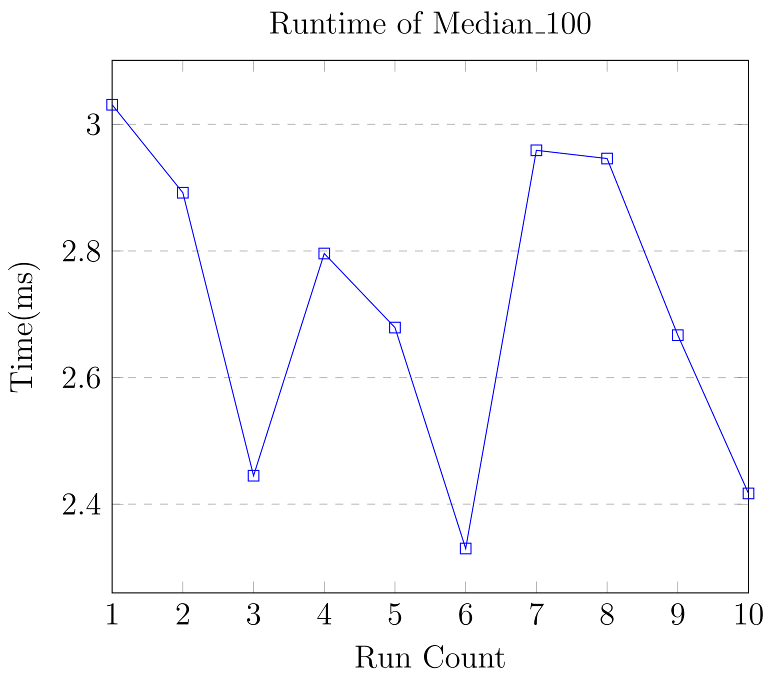


for 100000 inputs;
add : 99995
InorderRecursiveDifferenceSum : 0
GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0

5.4 Median Plots

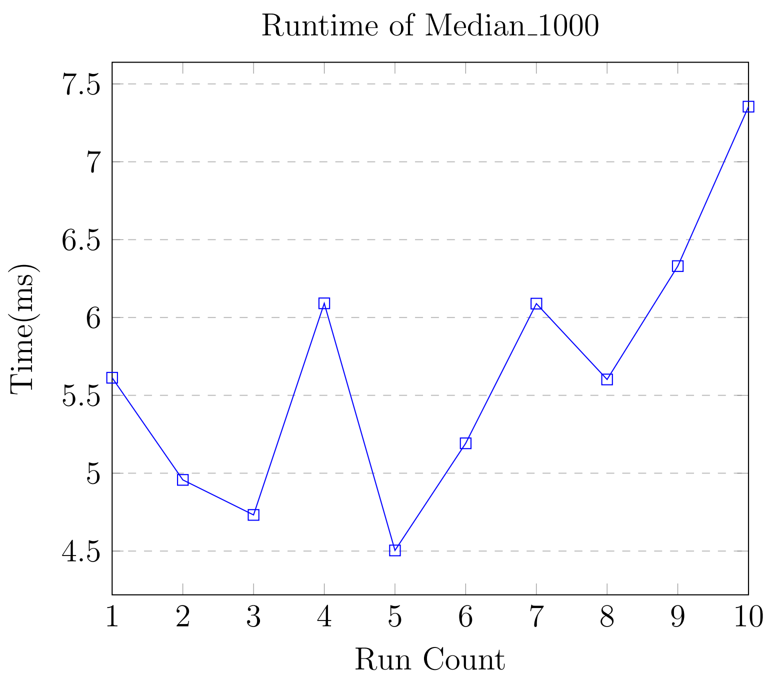


for 10 inputs;
add : 9
InorderRecursiveDifferenceSum : 0
GetForwardNode : 33
GetBackwardsNode : 9
GoForward : 6
GoBackward : 9
UpdateQ : 9

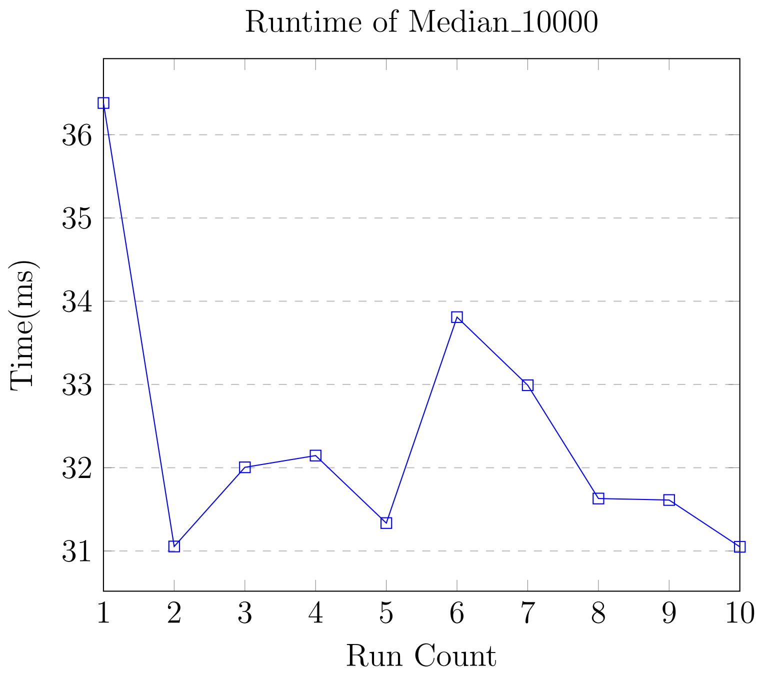


for 100 inputs;

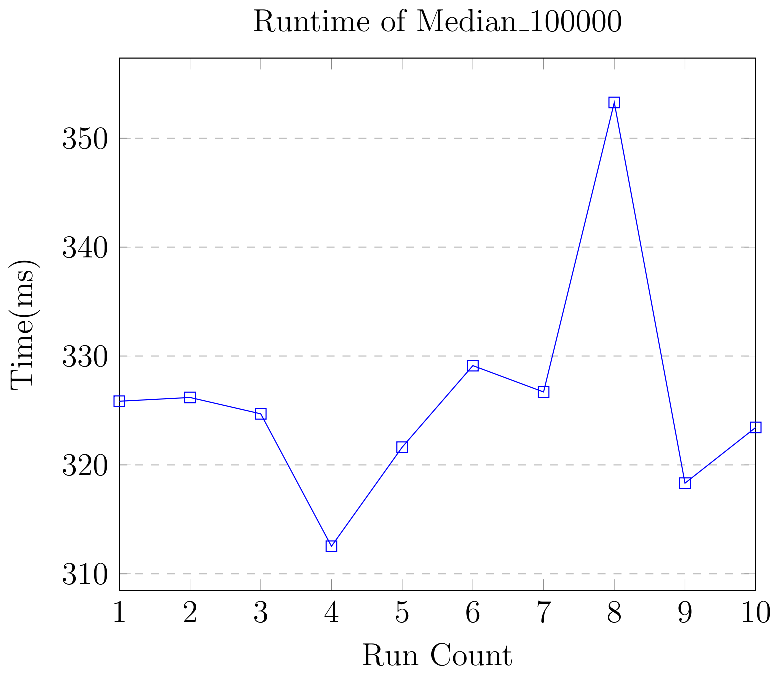
add : 99
InorderRecursiveDifferenceSum : 0
GetForwardNode : 360
GetBackwardsNode : 47
GoForward : 63
GoBackward : 47
UpdateQ : 99



for 1000 inputs;
add : 999
InorderRecursiveDifferenceSum : 0
GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0



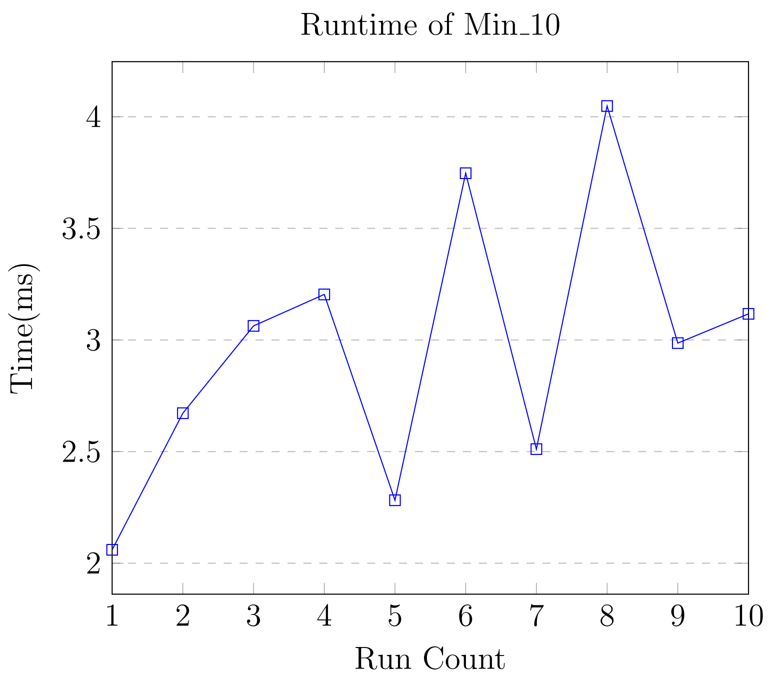
for 10000 inputs;
add : 9999
InorderRecursiveDifferenceSum : 0
GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0



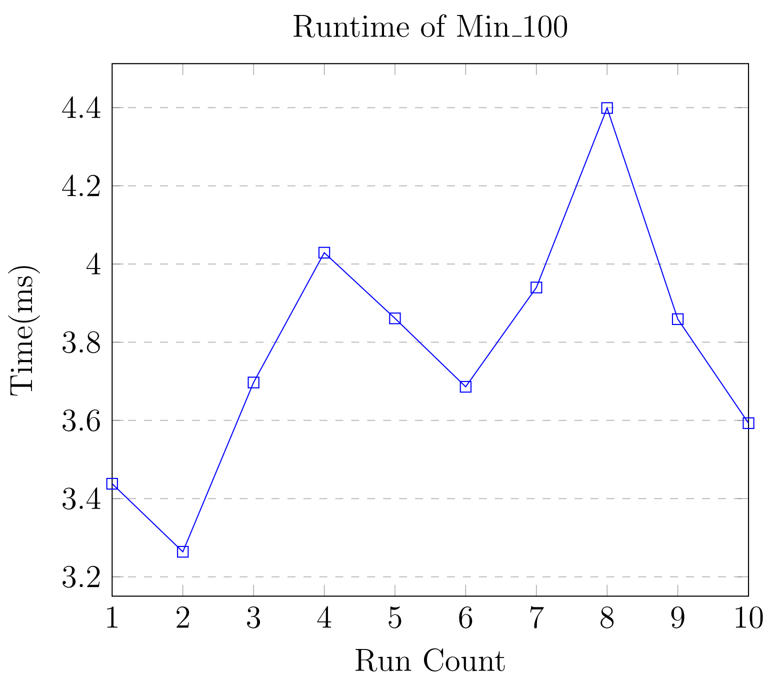
for 100000 inputs;
add : 99995
InorderRecursiveDifferenceSum : 0

GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0

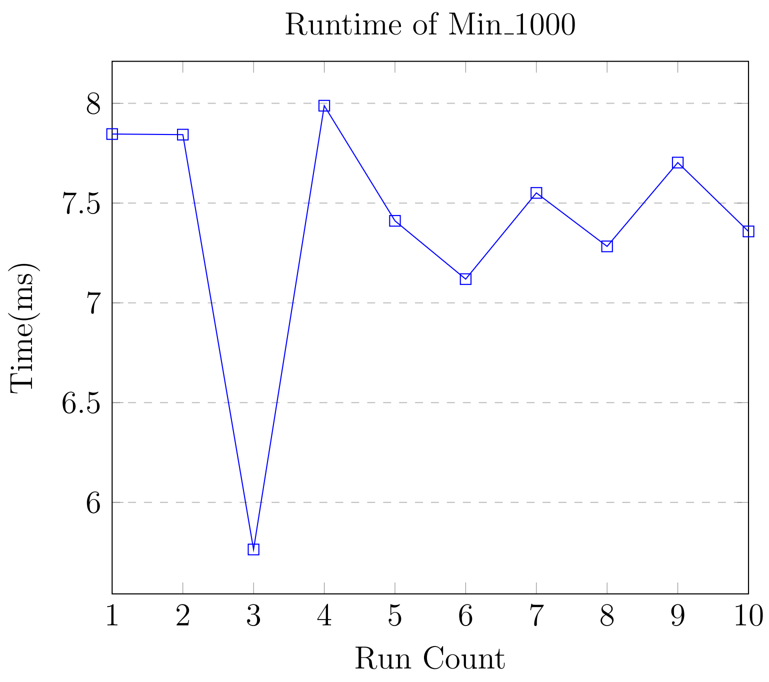
5.5 Min Plots



for 10 inputs;
add : 9
InorderRecursiveDifferenceSum : 0
GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0

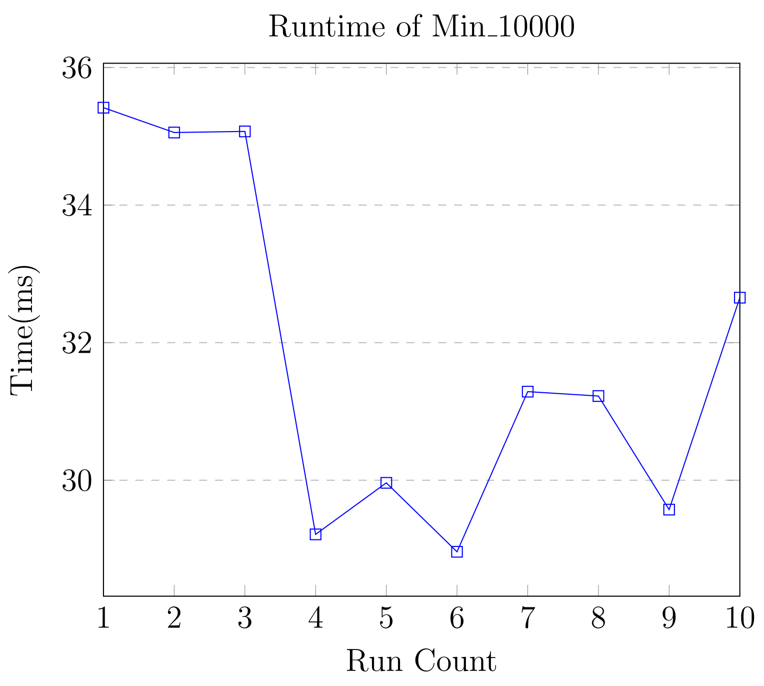


for 100 inputs;
add : 99
InorderRecursiveDifferenceSum : 0
GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0

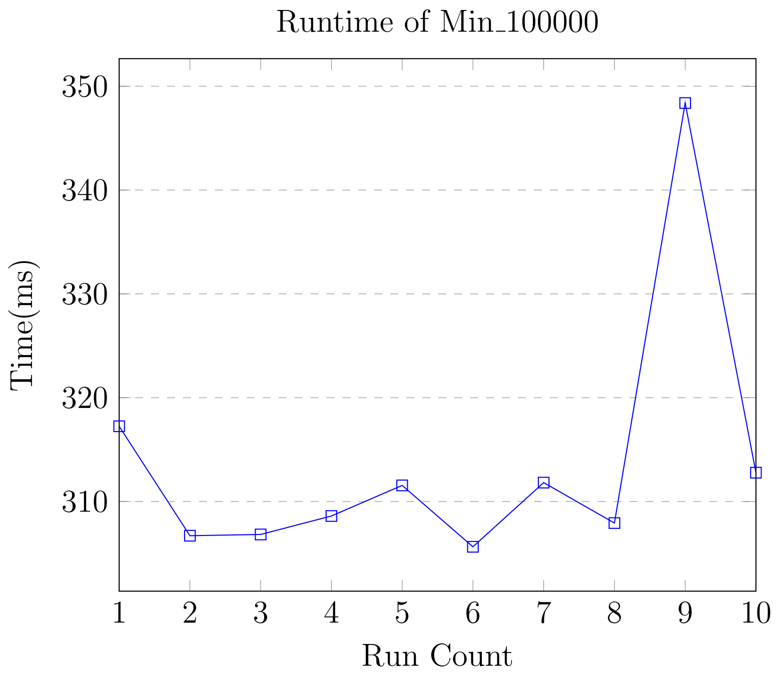


for 1000 inputs;
add : 999
InorderRecursiveDifferenceSum : 0

GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0

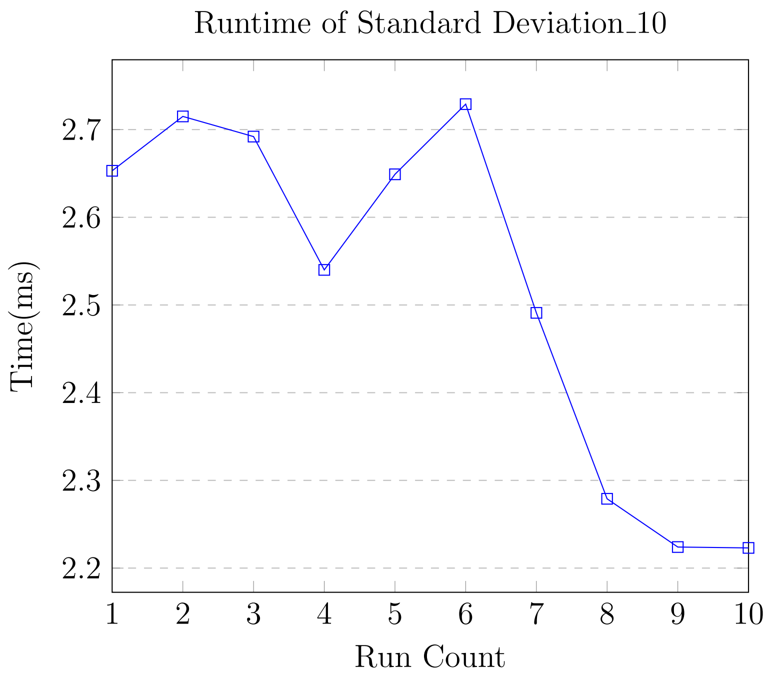


for 10000 inputs;
add : 9999
InorderRecursiveDifferenceSum : 0
GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0

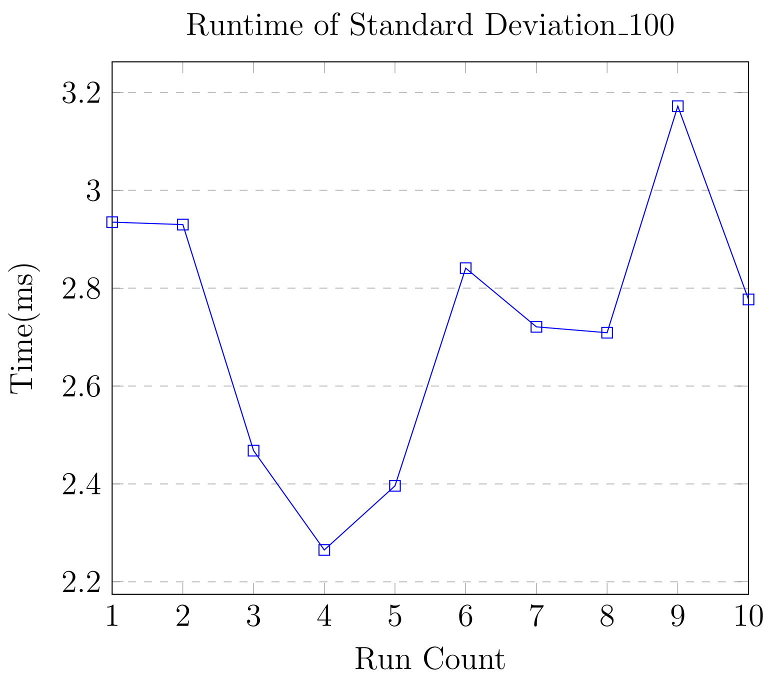


for 100000 inputs;
add : 99995
InorderRecursiveDifferenceSum : 0
GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0

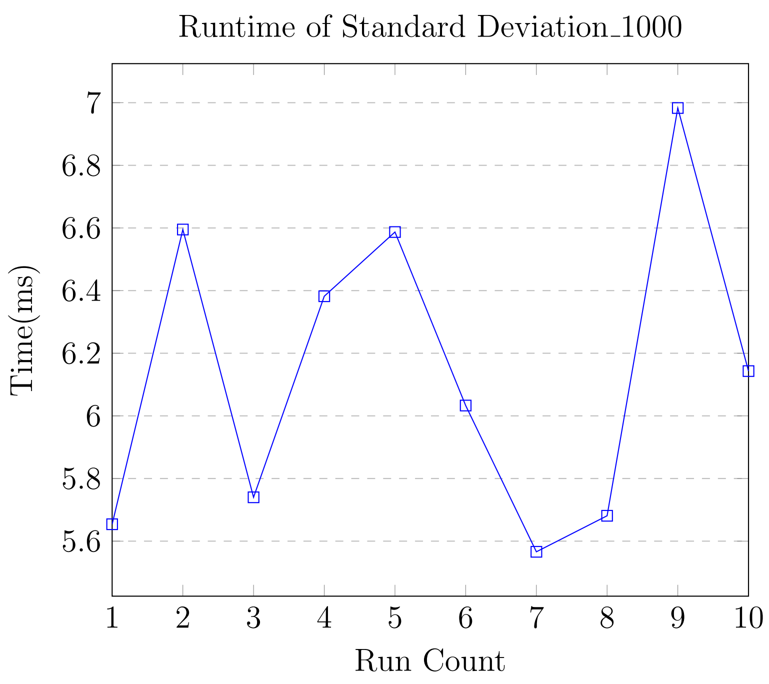
5.6 Standard Deviation Plots



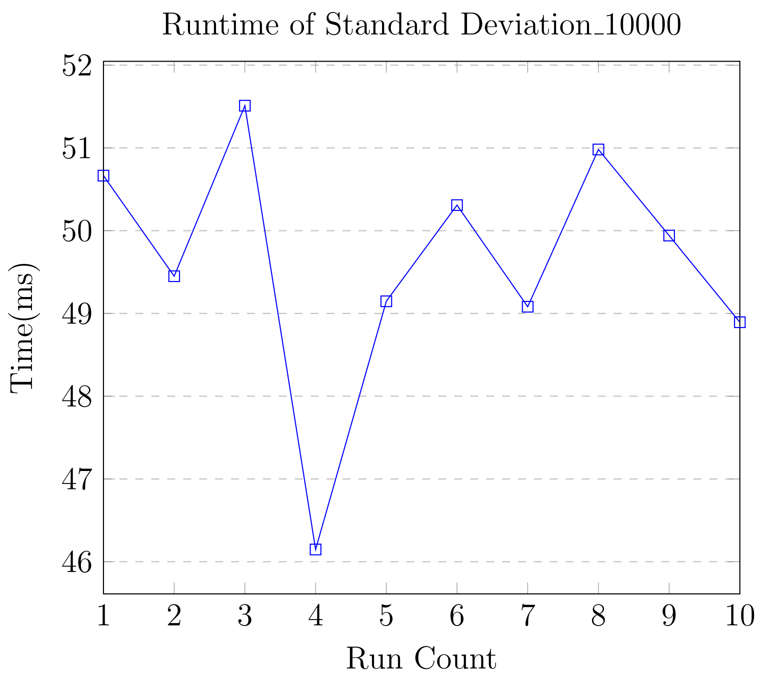
for 10 inputs;
add : 9
InorderRecursiveDifferenceSum : 4
GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0



for 100 inputs;
add : 99
InorderRecursiveDifferenceSum : 475
GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0

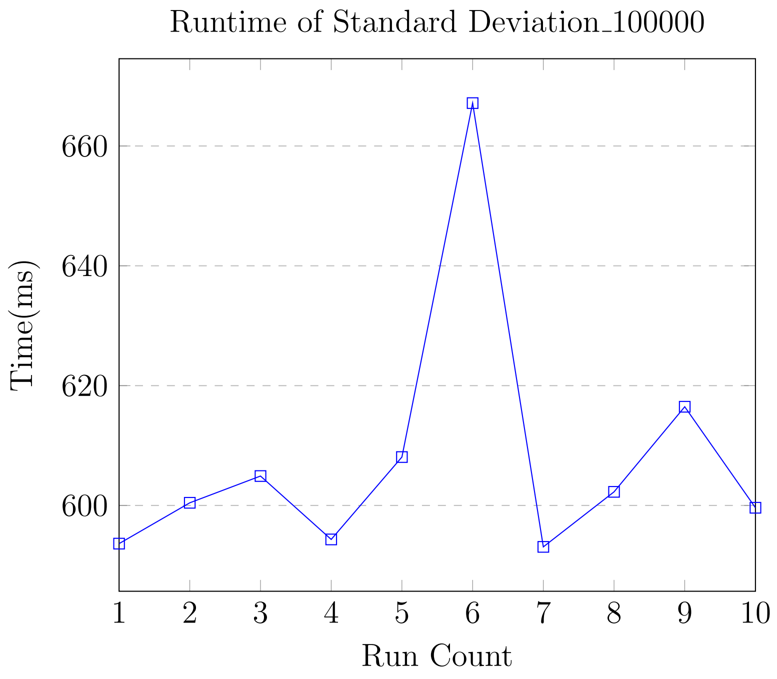


for 1000 inputs;
add : 999
InorderRecursiveDifferenceSum : 24133
GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0



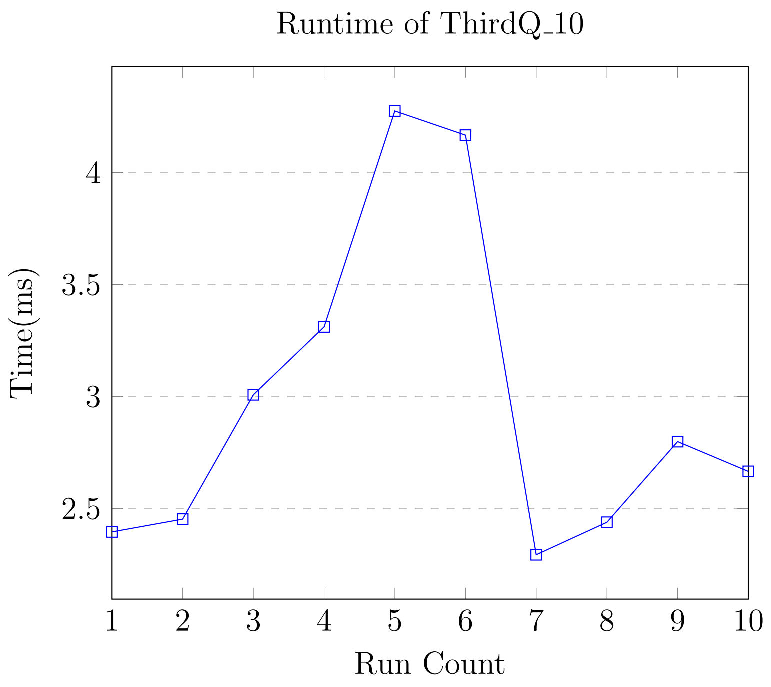
for 10000 inputs;
add : 9999
InorderRecursiveDifferenceSum : 638276

GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0

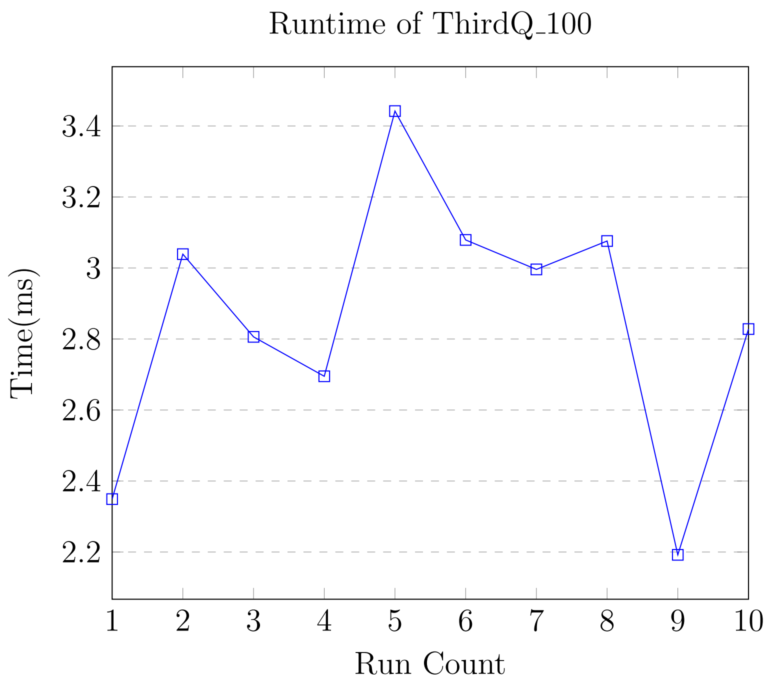


for 100000 inputs;
add : 99995
InorderRecursiveDifferenceSum : 10078329
GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0

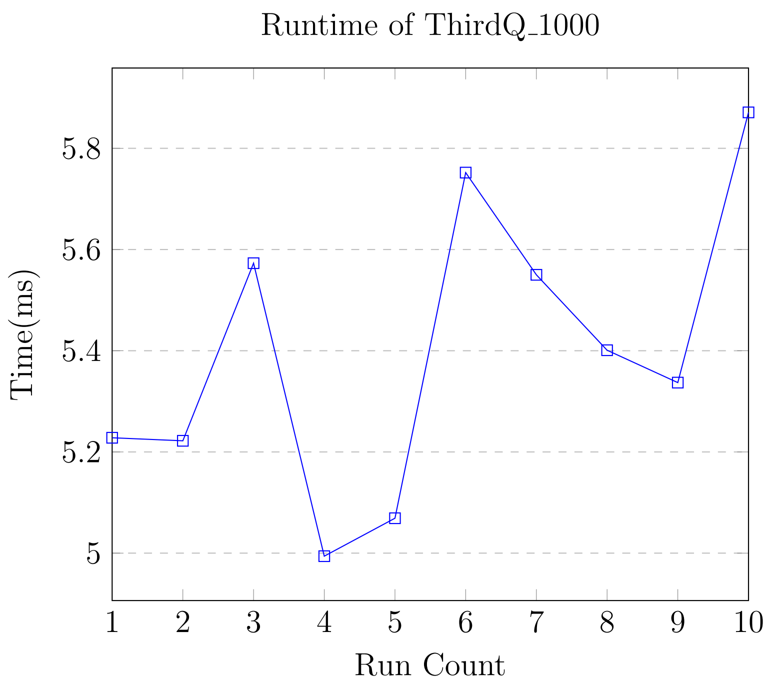
5.7 ThirdQ



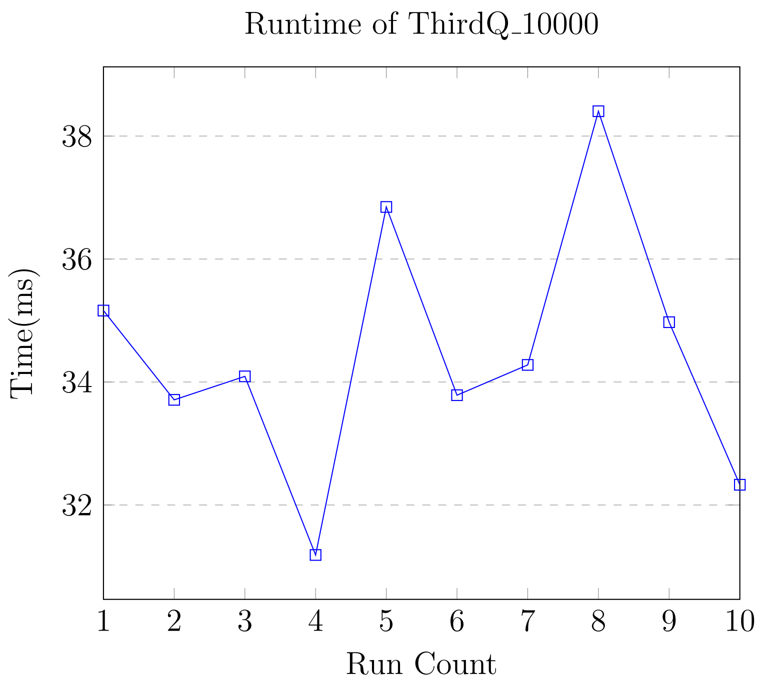
for 10 inputs;
add : 9
InorderRecursiveDifferenceSum : 0
GetForwardNode : 33
GetBackwardsNode : 9
GoForward : 6
GoBackward : 9
UpdateQ : 9



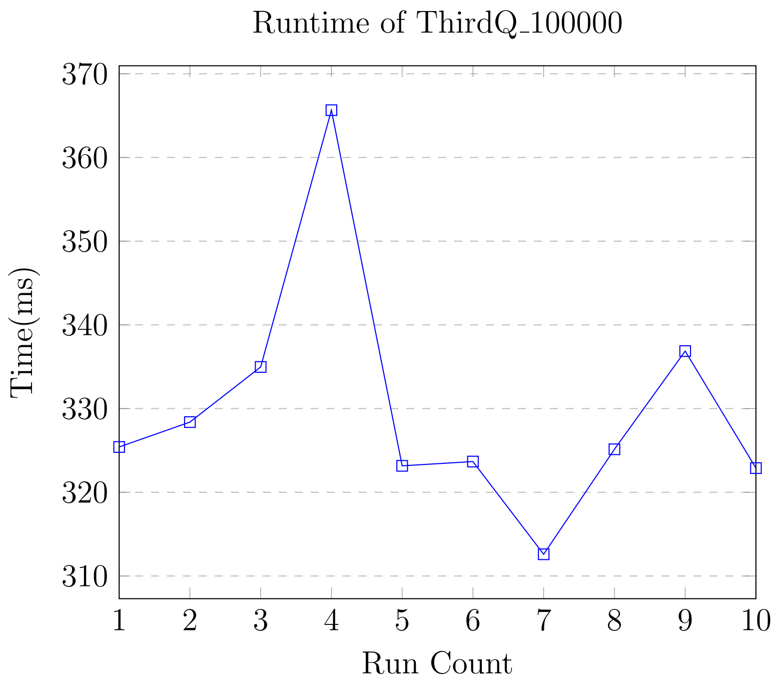
for 100 inputs;
add : 99
InorderRecursiveDifferenceSum : 0
GetForwardNode : 360
GetBackwardsNode : 47
GoForward : 63
GoBackward : 47
UpdateQ : 99



for 1000 inputs;
add : 999
InorderRecursiveDifferenceSum : 0
GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0



for 10000 inputs;
add : 9999
InorderRecursiveDifferenceSum : 0
GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0



for 100000 inputs;
add : 99995
InorderRecursiveDifferenceSum : 0

GetForwardNode : 0
GetBackwardsNode : 0
GoForward : 0
GoBackward : 0
UpdateQ : 0