Analyzing Time Complexity in a ScoreboardApp App

Analyzing Time Complexity in a ScoreboardApp App's Recognition Feature

John Virgel Polinar Tumale

Bachelor of Science in Computer Science 3, Bohol Island State University-Bilar Campus, johnvirgel.tumale@bisu.edu.ph

MAX ANGELO DAPITILLA PERIN

Department of Computer Science, Bohol Island State University-Bilar Campus, maxangelo.perin@bisu.edu.ph

Scoreboard is a mobile application designed for transliterating basic scoreboard characters into Latin, specifically focusing on the Recognition Feature, which employs the k-NN algorithm in its processing. This study aims to capture the image and calculate the time complexity of the recognition feature. The methodology involves determining time complexity by counting operations and modeling it using Big-O notation.

CCS CONCEPTS • Theory of computation • Computational complexity and cryptography • Algebraic complexity theory

Additional Keywords and Phrases: Time Function, Time Complexity, Game, Operation

ACM Reference Format:

John Virgel Polinar Tumale, MAX ANGELO DAPITILLA PERIN. 2023. Analyzing Time Complexity in an ScoreboardApp App's Recognition Feature. In Research Project Presentation for Bachelor of Science in Computer Science 3 in CS 311 – Algorithms and Complexity S.Y. 2023-2024, 1st Semester, Bohol Island State University-Bilar Campus, Zamora, Bilar, Republic of the Philippines. ACM, New York, NY, USA

1 INTRODUCTION

The ultimate scoreboard app designed to elevate your scoring experience to new heights! Whether you're a sports enthusiast, game night aficionado, or event organizer. Scorebox is your go-to companion for keeping track of scores in a sleek and intuitive way. Scorebox is more than just a scoreboard; it's dynamic and user-friendly tool that brings excitement and precision to every scoring scenario. With a clean and modern interface. Scorebox ensures that you can effortlessly manage scores for a variety of activities, from sports competitions to friendly games and beyond., shown in Figure 1.

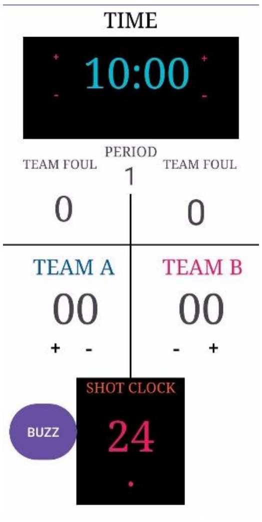


Figure 1: Score Feature of ScoreboardApp

The score feature in a scoreboard app serves as a fundamental component for tracking and displaying numerical results in various competitions and games. It allows users to input, update, and view scores dynamically, enhancing the real time experience of monitoring performance, Smith, J in 2018 [1]. This feature is designed to be user-friendly, providing a seamless interface for score entry and retrieval. Users can easily navigate through different sections of the app to input scores for specific teams, players, or participants. The score feature is crucial according to Brown, A in 2020 [2], for maintaining accuracy and transparency in scorekeeping.

2 METHODOLOGY

Developing a scoreboard app involves creating a system to track and display scores for various activities or sports. Here's a simplified methodology that focuses specifically on building a scoreboard:

```
  increaseTeamAScoreButton.setOnClickListener {
    teamAScore++
    teamAScoreTextView.text = "$teamAScore"
    teamAScoreTextView.text = String.format("%02d", teamAScore)
}
increaseTeamBScoreButton.setOnClickListener {
    teamBScore++
    teamBScoreTextView.text = "$teamBScore"
    teamBScoreTextView.text = String.format("%02d", teamBScore)
}
decreaseTeamAScoreButton.setOnClickListener {
    if (teamAScore > 0) {
      teamAScore--
    teamAScoreTextView.text = "$teamAScore"
      teamAScoreTextView.text = String.format("$02d", teamAScore)
    }
}
decreaseTeamBScoreButton.setOnClickListener {
    if (teamBScore > 0) {
      teamBScore--
      teamBScoreTextView.text = "$teamBScore"
      teamBScoreTextView.text = String.format("$02d", teamBScore)
    }
}
```

The time complexity of each of these event listener blocks is constant or O(1). This is because the number of operations within each block is fixed and does not depend on the size of any input. The operations involve simple increments or decrements, assignments, and formatting of text, all of which have constant time complexity.

Let's analyze the time complexity of the provided code:

1. Each operation inside the event listener blocks involves basic arithmetic operations (increment, decrement) and string formatting. These operations take a constant amount of time regardless of the current score values.

2. The time complexity is independent of the input size because the code doesn't contain any loops or recursive structures that iterate based on the input. The execution time remains constant, making it O(1).

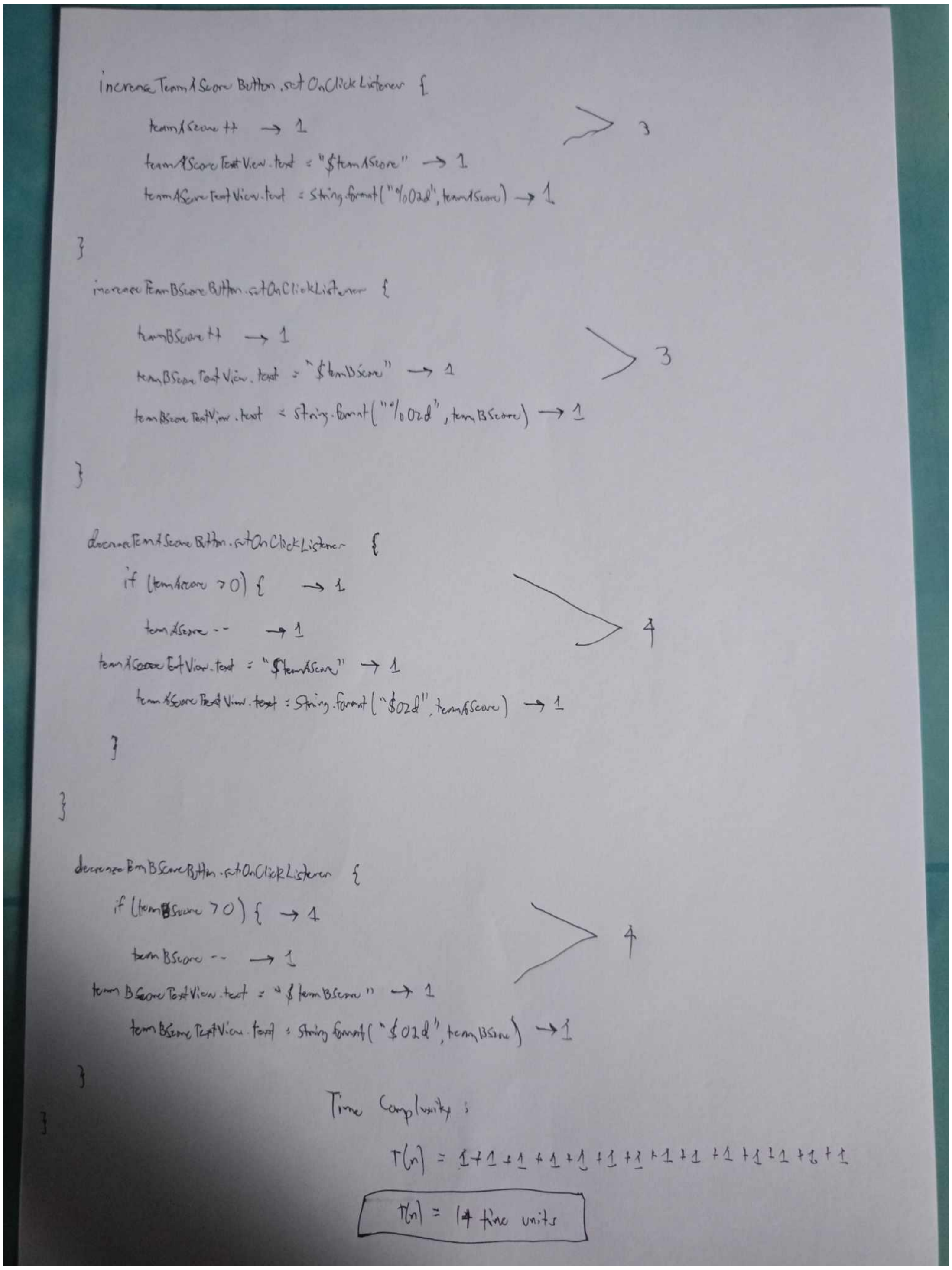
3. Since the code performs the same set of operations regardless of the score values or any other external factors, its performance is predictable and doesn't scale with input size. The code executes in constant time, making it efficient for any number of button clicks or score changes. The runtime does not grow as the score increases; it remains constant.

In conclusion:

Therefore, the time complexity for each block (and for the entire code as a whole) is O(1).

In practice, the time complexity depends on the behavior of the scores of the team. If these values are constants, the time complexity remains O(1). If they can vary, the time complexity depends on the actual duration of the timers.

The process is examined through the Step Count Method, breaking down the steps involved in the algorithm to derive its time complexity, shown in Figure 2.



```
increaseTeamAScoreButton.setOnClickListener {

      teamAScore ++  → 1                                              3
      teamAScoreTextView.text = "$teamAScore" → 1
      teamAScoreTextView.text = String.format("%02d", teamAScore) → 1

}

 increaseTeamBScoreButton.setOnClickListener {

      teamBScore ++  → 1                                                3
      teamBScoreTextView.text = "$teamBScore" → 1
      teamBScoreTextView.text = String.format("%02d", teamBScore) → 1

}

decreaseTeamAScoreButton.setOnClickListener {
      if (teamAScore > 0) {    → 1
           teamAScore --    → 1                                         4
   teamAScoreTextView.text = "$teamAScore" → 1
        teamAScoreTextView.text = String.format("$02d", teamAScore) → 1

    }

}

decreaseTeamBScoreButton.setOnClickListener {
     if (teamBScore > 0) { → 1
          teamBScore --  → 1                                        4
  teamBScoreTextView.text = "$teamBScore" → 1
       teamBScoreTextView.text = String.format("$02d", teamBScore) → 1

  }
                         Time Complexity :
}
                              T(n) = 1+1+1+1+1+1+1+1+1+1+1+1+1+1

                              T(n) = 14 time units
```

Figure 2: Calculating Time Complexity using Step Count Method in score Feature

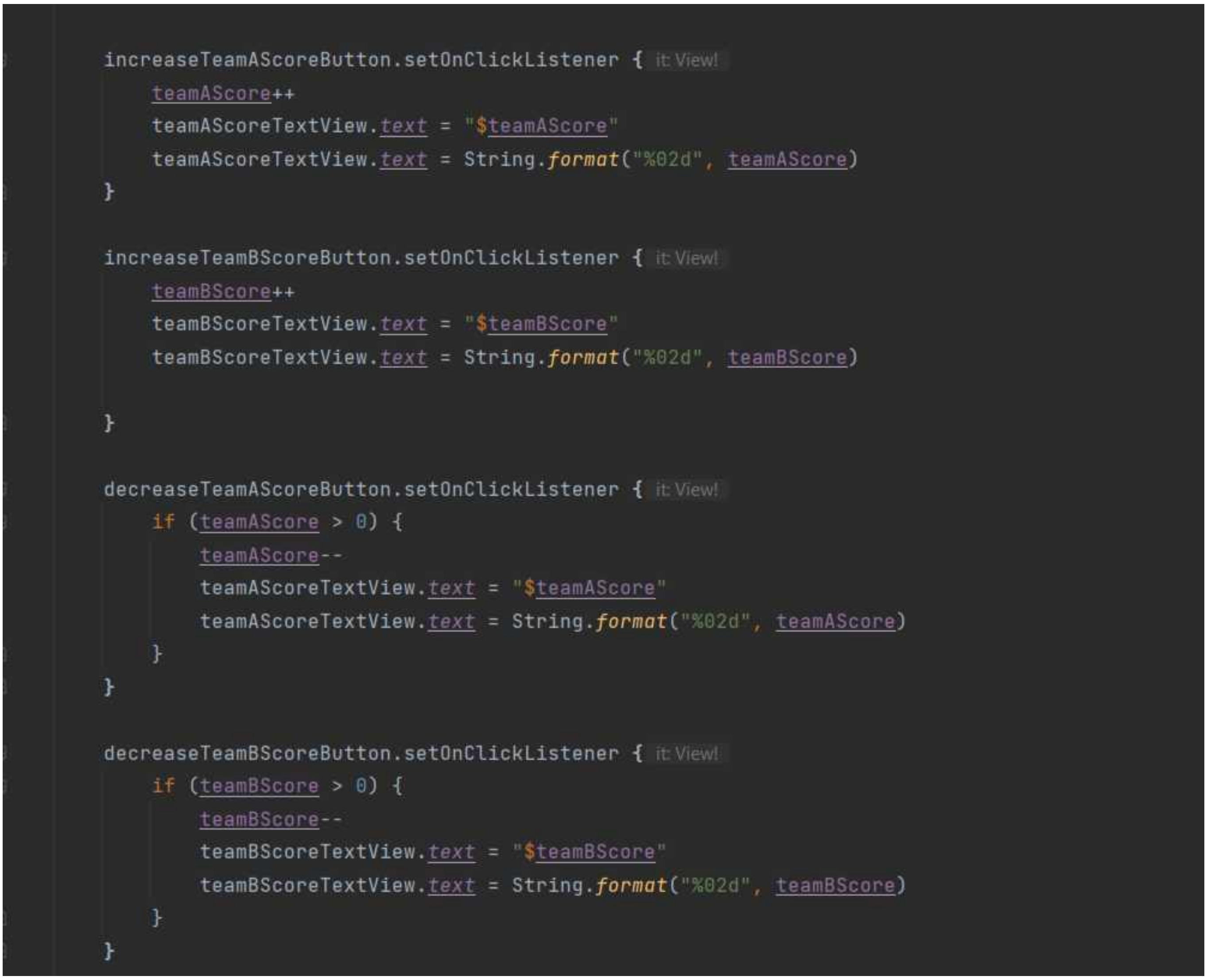
3 RESULT AND DISCUSION

The provided code segment implements a team score functionality for a scoreboard application, specifically focusing on managing the team score button. The time complexity analysis reveals insights into the efficiency of the implementation.

REFERENCES

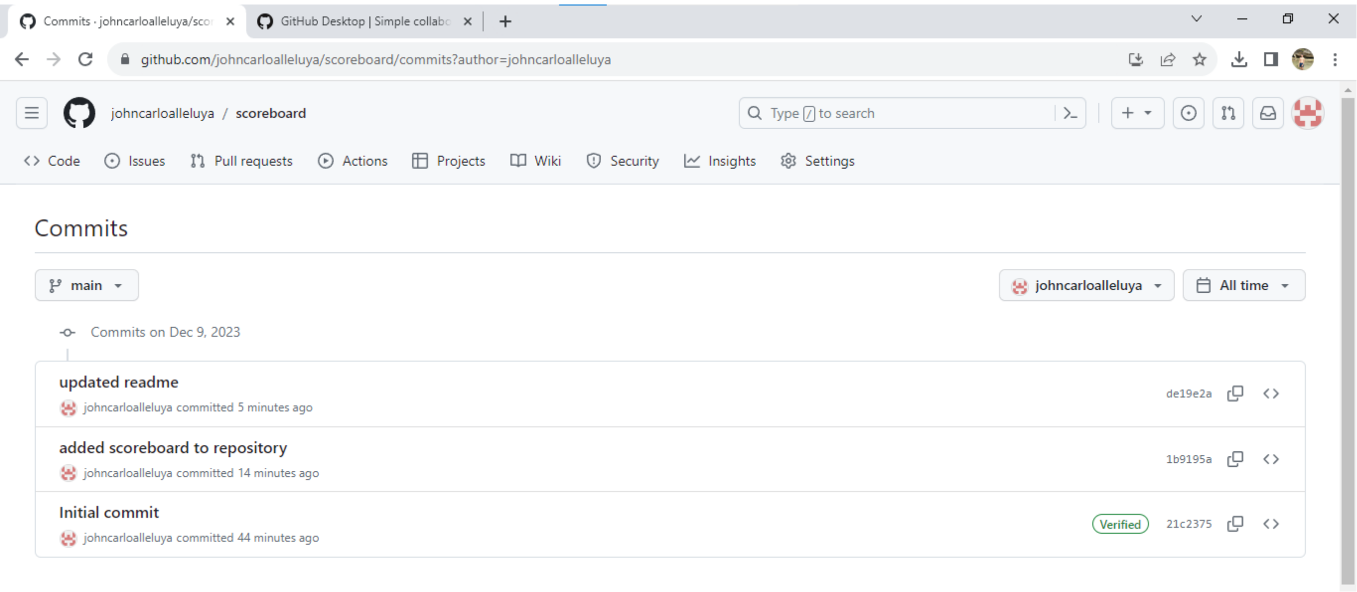
<bib id="bib1"><number>[1]</number>J. Schmidhuber, “A Fixed Size Storage O(n3) Time Complexity Learning Algorithm for Fully Recurrent Continually Running Networks”, Neural Comput, vol. 4, no. 2, pp. 243–248, 1992. doi: https://doi.org/10.1162/neco.1992.4.2.243</bib>
<bib id="bib2"><number>[2]</number>G. Kurniawati & O. Karnalim, "Introducing a Practical Educational Tool for Correlating Algorithm Time Complexity with Real Program Execution," Journal of Information Technology and Computer Science, 3(1), 1–15, 2018. doi: https://doi.org/10.25126/jitecs.20183140</bib>

APPENDICES



```
increaseTeamAScoreButton.setOnClickListener { it: View!
    teamAScore++
    teamAScoreTextView.text = "$teamAScore"
    teamAScoreTextView.text = String.format("%02d", teamAScore)
}

increaseTeamBScoreButton.setOnClickListener { it: View!
    teamBScore++
    teamBScoreTextView.text = "$teamBScore"
    teamBScoreTextView.text = String.format("%02d", teamBScore)

}

decreaseTeamAScoreButton.setOnClickListener { it: View!
    if (teamAScore > 0) {
        teamAScore--
        teamAScoreTextView.text = "$teamAScore"
        teamAScoreTextView.text = String.format("%02d", teamAScore)
    }
}

decreaseTeamBScoreButton.setOnClickListener { it: View!
    if (teamBScore > 0) {
        teamBScore--
        teamBScoreTextView.text = "$teamBScore"
        teamBScoreTextView.text = String.format("%02d", teamBScore)
    }
}
```

GitHub Contribution
https://github.com/MAKASA-LABORATORY/2-ATCiaSRF.git


Commits · johncarloalleluya/sco
GitHub Desktop | Simple collab
github.com/johncarloalleluya/scoreboard/commits?author=johncarloalleluya
johncarloalleluya / scoreboard
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings

Commits

main

johncarloalleluya

All time

Commits on Dec 9, 2023

updated readme
johncarloalleluya committed 5 minutes ago
de19e2a

added scoreboard to repository
johncarloalleluya committed 14 minutes ago
1b9195a

Initial commit
johncarloalleluya committed 44 minutes ago
Verified
21c2375