Preliminary Analysis of Code Hunt Data Set from a Contest

Pierre McCauley, Brandon Nsiah-Ababio, Joshua Reed, Faramola Isiaka
University of Illinois
Urbana, IL, USA
{pmccaul2,nsiahab2,jbreed3,fisiaka2}@illinois.edu

Tao Xie
University of Illinois
Urbana, IL, USA
taoxie@illinois.edu

ABSTRACT

Code Hunt (https://www.codehunt.com/) from Microsoft Research is a web-based serious gaming platform being popularly used for various programming contests. In this paper, we demonstrate preliminary statistical analysis of a Code Hunt data set that contains the programs written by students (only) worldwide during a contest over 48 hours. There are 259 users, 24 puzzles (organized into 4 sectors), and about 13,000 programs submitted by these users. Our analysis results can help improve the creation of puzzles in a future contest.

CCS Concepts

•Social and professional topics → Software engineering education;

Keywords

Code Hunt; educational software engineering

1. INTRODUCTION

Code Hunt [1] (https://www.codehunt.com/) from Microsoft Research is a web-based serious gaming platform being popularly used for various programming contests. Coding duel [2] is the game type in Code Hunt. A coding duel (written in C# or Java) includes a secret (solution) code segment and a user-visible code segment, both embodied in a `Puzzle` method sharing the same method signature. The user-visible code segment is typically empty in content or an incorrect/incomplete version of the secret code segment. To solve a coding duel, a user iteratively observes clues (i.e., test data and their outputs from both segments) generated by Code Hunt and modifies the user-visible code segment to match the functional behaviors of the secret code segment (the matching is determined based on the generated test data).

Microsoft Research released a Code Hunt data set (https://github.com/Microsoft/Code-Hunt), consisting of programs that were written by students throughout the world for the Imagine Cup 2014. This contest gave students 48 hours to write programs, with their choice of the used programming language as Java or C#. Within the data set, there are 259 users who attempted the given 24 puzzles (named as levels in Code Hunt), being grouped into 4 sectors (6 puzzles in each sector). Each sector typically corresponds to a specific learning topic. Code Hunt imposes a rule that a user may only access a puzzle in a sector if the user has won at least all but one of the puzzles in the previous sector. Therefore, for a contest, contest or puzzle creators typically arrange the sectors or puzzles in an order of increasing difficulty.

In this paper, we present the first preliminary statistical analysis on the data set. The main purpose of our analysis is to better understand how users performed on the puzzles and use the analysis results to help improve the creation of puzzles in a future contest. To analyze the data set, we develop a Python program to traverse the data set and extract information about each user and their performance on each of the 24 puzzles. The extracted information includes the user's self-declared experience level, the number of the user's attempts on a puzzle, the user's completion status of a puzzle, and the programming language used by the user in the attempts. We then analyze the extracted information with R (https://www.r-project.org/), a tool for statistical computing.

2. ANALYZING PUZZLE DIFFICULTY

In this section, we present a statistical analysis on the data set for analyzing puzzle difficulty based on two types of information extracted for each puzzle: the completion status of each user and the number of attempts made by each user. We intend to address two main research questions: **RQ1**. Does the information extracted from the data set generally reflect a trend of increasing difficulty from Puzzles 1 to 24 and from Sectors 1 to 4? **RQ2**. Are there any outliers if a trend of increasing difficulty is indeed generally followed? Given that a trend of increasing difficulty is desirable and expected, answering these research questions can help improve the creation of puzzles in a future contest.

Figure 1 shows a bar chart for the percentage of users who completed (bottom portion), attempted but did not complete (middle portion), and skipped a puzzle (top portion), respectively. Figure 2 shows a boxplot for the distribution of the number of attempts made by each user for a puzzle. The x-axis of both charts denote Puzzles 1-24. For the boxplot, to properly handle cases of users who attempted but did not complete or skipped a puzzle, we post-process the number of these users' attempts: we replace such number with the highest number of attempts (among all the users) for the puzzle. Because there are some cases where the highest number of attempts per puzzle skews the distribution (e.g., 278 or 330 attempts), we make a cutoff of the highest number of attempts as 150.

For RQ1, from the two figures, we can observe the general expected trend of increasing difficulty from Puzzles 1 to 24 and from Sectors 1 to 4. For example, as a user continues further into the

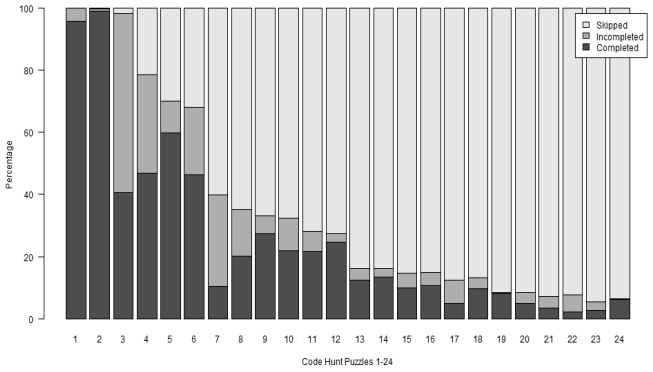

Figure 1: Percentage of users who completed (bottom portion), attempted but did not complete (middle portion), and skipped a puzzle (top portion), respectively.

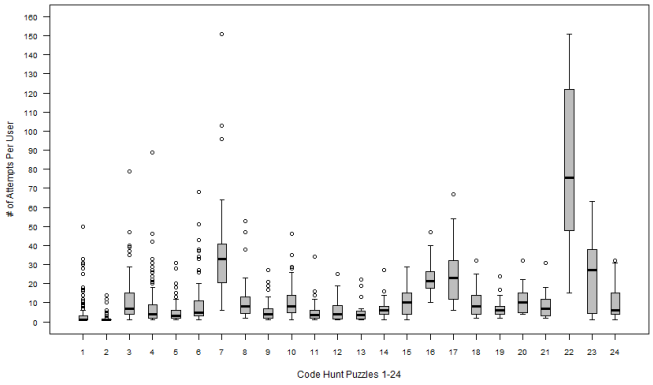

Figure 2: Number of attempts by each user per puzzle.

contest, the puzzles on each sector become more difficult. The percentage of users who completed tends to decrease and the number of attempts tends to increase as users advance from Puzzles 1 to 24 and from Sectors 1-4.

For RQ2, we observe a few outliers while the trend of increasing difficulty is generally followed. From both figures, we notice that at least Puzzle 3 from Sector 1, Puzzle 7 from Sector 2, and Puzzle 17 from Sector 3 are much more difficult than the other puzzles in their respective sector.

Sector 1 (Puzzles 1-6) focuses on the arithmetic portion of programming. Typically the first sector in a contest is expected to be the easiest; however, only about 40% of the users were able to complete Puzzle 3 (whose solution is shown in Figure 3). One possible reason is that Puzzle 3 is based on using bit-wise operations, and users often struggled with using bit-wise operators.

Sector 2 (Puzzles 7-12) focuses on the topic of loops, iterations, and conditional statements. The solution for Puzzle 7 from Sector 2 is shown in Figure 4. Note that API calls `IsNotNull` and `IsTrue` (defined on class `PexAssume`) in Lines 1, 2, and 6 are used to ignore generated test data that violate the specified constraints, and Line 3 is used to cause Code Hunt to generate specific test data. Only about 10% of the users were able to complete this puzzle. In addition, as shown in Figure 2, the average number of attempts to solve Puzzle 7 is about 4 times greater than the other puzzles in Sector 2. The solution to Puzzle 7 iterates through a list of numbers (Lines 5-7) and returns the average rounded to the closest integer (Lines 8-9). We hypothesize that the difficulty faced by many users came from their first inaccurate impression/guess to solve the puzzle. Computing the average of a list does not seem to be a difficult task to many users. However, from the data set, we observe that

```
   public static bool Puzzle(bool x, bool y, bool z) {
1        return x | y & z;
   }
```

Figure 3: Solution to Puzzle 3 (Sector 1)

```
  public static int Puzzle(int[] a) {
1    PexAssume.IsNotNull(a);
2    PexAssume.IsTrue(a.Length > 0);
3    if (a.Length==3 && (a[0]==13 & a[1]==-5 && a[2]==7));
4    int sum = 0;
5    foreach (var n in a) {
6       PexAssume.IsTrue(n>=-100 & n<=100);
7       sum += n;
     }
8    int len = a.Length;
9    return (sum + len/2) / len;
   }
```

Figure 4: Solution for Puzzle 7 (Sector 2)

many of these users were confused on how to compute the proper rounded average from the list. In particular, these users struggled with the code in Lines 8-9 for rounding the average to the closest integer. This struggle resulted in many of these users being frustrated to leave the puzzle incomplete.

Sector 3 (Puzzles 13-18) requires a combination of techniques and algorithms in order to successfully complete the puzzles. The solution to Puzzle 17 in Sector 3 is to emulate the movement of a knight piece in a chess game. We hypothesize that the difficulty faced by many users when solving Puzzle 17 came from requiring the use of bit-wise operations to solve the puzzle, a similar situation for Puzzle 3.

3. CONCLUSION

We have presented the first preliminary statistical analysis on the Code Hunt data set. Our analysis has focused on analyzing puzzle difficulty based on the completion status of each user and the number of attempts made by each user for a puzzle. The analysis results indicate that an expected trend of increasing difficulty is generally followed from Puzzles 1 to 24 and from Sectors 1 to 4. However, at the same time, there exist some outlier cases that a few puzzles are much more difficult than other puzzles in the same sector.

Much of such higher difficulty might be due to the use of bit-wise operations in these puzzles. Such observations have two main implications. First, contest or puzzle creators shall pay special attentions to puzzles involving the use of bit-wise operations and expect these puzzles to be of likely higher difficulty than other puzzles in the same sector. Second, systems and hardware engineering rely on frequent use of bit-wise operations; thus, if a user lacks skills in this topic area, recruiters seeking employees in these fields may want to watch out, and educators may want to enhance student learning on such skills.

4. ACKNOWLEDGMENTS

This material is based upon work supported by the Maryland Procurement Office under Contract No. H98230-14-C-0141. This work is also supported in part by National Science Foundation under grants no. CCF-1409423, CNS-1434582, CCF-1434596, CNS-1513939, CNS-1564274, and a Microsoft Research Award.

5. REFERENCES

[1] J. Bishop, R. N. Horspool, T. Xie, N. Tillmann, and J. de Halleux. Code Hunt: Experience with coding contests at scale. In *Proc. ICSE JSEET*, pages 398–407, 2015.

[2] N. Tillmann, J. D. Halleux, T. Xie, S. Gulwani, and J. Bishop. Teaching and learning programming and software engineering via interactive gaming. In *Proc. ICSE SEE*, pages 1117–1126, 2013.