Creating a Rudimentary Stardew Valley Save File Analyser Using the Boyer-Moore Algorithm

Jeremy Syaloom Okey Nathanael Simbolon - 13520042[1]
Informatics Study Program
School of Electrical Engineering and Informatics
Bandung Institute of Technology, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13520042@std.stei.itb.ac.id

***Abstract*—Pattern matching, especially those that involves string, is an important topic in computer science to this day. Decades of research has been done to tackle this problem in an efficient manner. The Boyer-Moore string-search algorithm is one of the most well-known string-search algorithms with broad usage, various implementations, and highly respected as one of the standard benchmarks in string-search literature. In this paper, I would like to present a simple use case for the algorithm, a rudimentary save file analyser for Stardew Valley, a well-known video game played by millions of players around the world**

***Keywords*—Boyer-Moore, Stardew Valley, save file analyser**

I. Introduction

For millennia, we have come to an understanding that human's advancement in science and technology cannot be separated from our understanding of patterns. Patterns exist around us in the form of regularity in nature, man-made designs, and abstract concepts. Our senses allow us to observe these regularities and open a new door to thorough analysis in hopes for increasing our understanding. These noticeable arrangements are scattered throughout our scientific idea, from mathematics, to language. The field of computer science has been spearheading our efforts to further expand our knowledge in this area.

Pattern matching and recognition is one of the most celebrated topics in computer science, due to our advancements since the 1950-s during which regular expressions are first described [1]. In modern society, pattern matching and recognition see wide applications from computer-aided diagnosis in medical science [2] to image processing [3]. In computer science, extensive efforts around pattern matching and recognition have been made, especially in those involving strings.

Over the course of seven decades, several string-matching algorithms have been developed to fulfill scientists' needs in efficiency. One of those is the Boyer-Moore algorithm, first described by Robert S. Boyer and J. Strother Moore in 1977 [4]. Over the following years, several optimisations have been made to the algorithm, notably by Galil in 1979 [5], then by Apostolico and Giancarlo in 1986 [6].

In this paper, I shall present a simple use case for the well-known algorithm in the form of a rudimentary save file analyser for the popular video game Stardew Valley. Hopefully, this paper will induce more discussion and interest in the topic, especially for newcomers in the field of string matching.

II. Theoretical Foundation

A. Pattern Matching

Pattern matching is an area of research in computer science that is devoted to the act of checking the presence of a pattern in a given sequence. The history of pattern matching dates back to 1951 when regular expressions is first described by mathematician Stephen Cole Kleene [1]. Following that, several programming languages are developed to aide scientists in pattern-matching activities, especially those involving strings. These languages include COMIT in 1957 [7], SNOBOL in 1962 [8], and Refal in 1968 [9].

In 1970, James H. Morris and Vaughan Pratt published a technical report in the 1970 regarding a linear fast string-matching algorithm [10]. The algorithm is now referred to as the Knutt-Morris-Pratt algorithm, owing to the fact that Donald E. Knutt discovered the algorithm independently and them releasing a joint paper in 1977 [11]. Since then, several other string-matching algorithms have been born. A simplified timeline can be seen below.

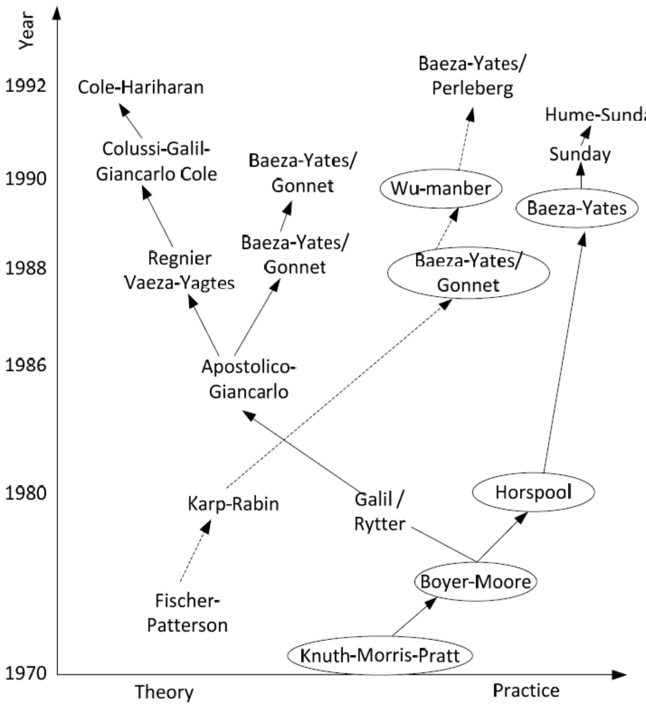


Fig. 1. The development history of pattern matching algorithms [12]

B. Boyer-Moore Algorithm

The Boyer-Moore algorithm is a string-matching algorithm first described by Robert S. Boyer and J. Strother Moore in 1977 [4]. The algorithm has a worst-case running time of $O(m+n)$ if the pattern does not exist in the text [11]. If the pattern does exist, however, the algorithm has a worst-case running time of $O(mn)$ [5]. The algorithm compares characters from right to left, unlike most string-matching algorithms. In addition, the algorithm preprocesses the pattern to obtain a last occurrence table, that is a table that stored the last occurrence of a given alphabet in the pattern.

The algorithm can be described in pseudocode as follows [13].

```
Boyer-Moore Algorithm

1.  For a given pattern and the alphabet used in
    both the pattern and the text, construct the
    bad-symbol shift table
2.  Using the pattern, construct the good-suffix
    shift table
3.  Align the pattern against the beginning of the
    text
4.  Repeat the following step until either a
    matching substring is found or the pattern
    reaches beyond the last character of the text.
    Starting with the last character in the
    pattern, compare the corresponding characters
    in the pattern and the text until either all m
    character pairs are matched (then stop) or a
    mismatching pair is encountered after k ≥ 0
    character pairs are matched successfully. In
    the latter case, retrieve the entry t1(c) from
    the c's column of the bad-symbol table where c
    is the text's mismatched character. If k > 0,
    also retrieve the corresponding d2 entry from
    the good-suffix table. Shift the pattern to the
    right by max{t1(c) − k, 1}.
```

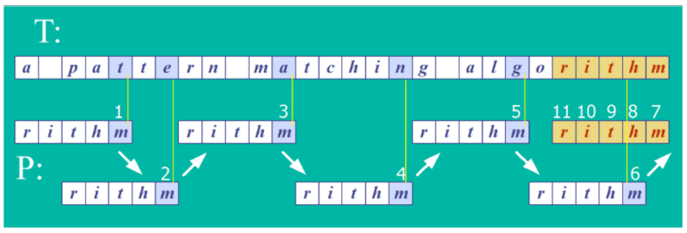


Fig. 2. Illustration on how the Boyer-Moore algorithm works [14]

C. Stardew Valley

Stardew Valley is an open-ended country-life role-playing game created by game designer Eric Barone under the alias of ConcernedApe. Heavily inspired by the Harvest Moon series, the game sets the player as an overworked corporate worker that has just inherited their grandfather's farm in the countryside. The soft goal of the game is to maintain the inherited farm and build it back to its former glory.

Fig. 3. A view of the in-game farm in the evening [15]

The game is written in C# and was released on PC on February 26th, 2016. Further release was expanded to Mac and Linux, Xbox One, Playstation 4, Nintendo Switch, and iOS over the following two years. As of March 2022, Stardew Valley has sold over 20 million copies across all platforms, with 13 million copies sold on PC alone [15].

III. Methodology

We shall determine the appropriate pattern to be search on the game save file. The Stardew Valley save file is written in the Extensive Markup Language (XML). With this information, we can choose the appropriate XML tag to be search in order to obtain the information we want from the save file.

I have written a short Python code that can be used to achieve our goal. The full source code and a copy of this paper can be found in https://github.com/tastytypist/save-file-analyser.

Our strategy to obtain the wanted information from the save file can be divided into three sub-strategies. The first sub-strategy involves simply finding the appropriate tag for the information. Then traversing the save file can be done until a closing tag is found. The information parsed can then be stored into the program. The code snippet can be constructed as follows.

```
def analyse(self):
    self.import_save_file()

    self.name = self.find("name")
    self.farm = self.find("farmName")
    self.farm_type = int(self.find("whichFarm"))
    self.time = int(self.find("millisecondsPlayed"))
    self.version = self.find("gameVersion")
    self.money = int(self.find("totalMoneyEarned"))
    self.spouse = self.find("spouse")
```

In the code snippet above, I pass the appropriate XML tag to the method `find`. The method `find` implements the Boyer-Moore algorithm and can be seen in the code snippet below.

```
def find(self, attr, child=False, data=None):
    if not data:
        data = self.data
    self.finder = str_find.StringFinder(attr, data)
    self.finder.find_string()

    if self.finder.index == -1:
        return None

    character = data[self.finder.index]
    start_index = self.finder.index
    stop_index = self.finder.index

    if not child:
        while character != "<":
            if character == ">":
                start_index = stop_index + 1
            stop_index += 1
            character = data[stop_index]
    else:
        while character != "/":
            if character == ">":
                start_index = stop_index + 1
            stop_index += 1
            character = data[stop_index]
        stop_index -= 1

    return data[start_index:stop_index]
```

The second sub-strategy involves finding the existence of multiple tags. If a tag being searched is found, a counter depicting how many of those tags have been found is stored. This strategy is used to count how many stardrops a player has obtained. The code snippet can be seen as follows.

```
self.stardrop_id = ["CF_Fair", "CF_Fish",
                    "CF_Mines", "CF_Sewer",
                    "CF_Spouse", "CF_Statue",
                    "museumComplete"]

def analyse(self):
    for ids in self.stardrop_id:
        if self.find(ids):
            self.stardrops += 1
```

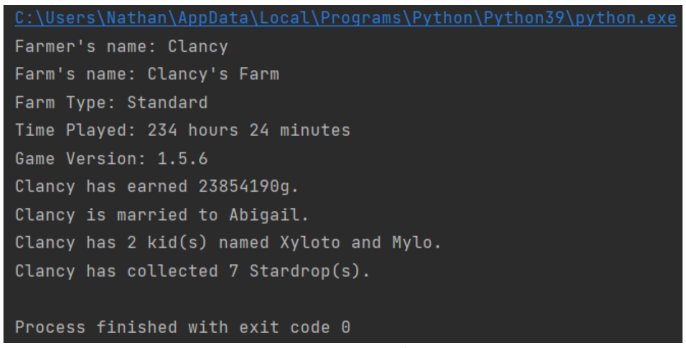
The third sub-strategy involves finding multiple occurrences of XML tags. From decompiling the game source code, the tag being searched only appears twice at maximum. With this knowledge, we can store the occurrence count of the tag and search in tag using a loop with the occurrence count as a condition. This strategy is used to find the name of child(ren) the farmer has. The code snippet can be seen as follows.

```
self.kids = []      # type: "Child"

def analyse(self):
    child = self.find('type="Child"', child=True)
    last_found = 0
    while child and len(self.kids) < 2:
        self.kids.append(child)
        last_found += self.finder.index
        child = self.find('type="Child"',
                          child=True,
                          data=self.data[last_found
                                         + 12:])
```

IV. Results

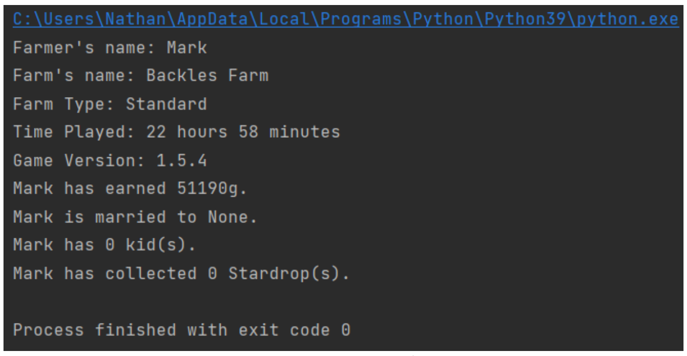
Here are the results of using the program with the provided test save files.



```
C:\Users\Nathan\AppData\Local\Programs\Python\Python39\python.exe
Farmer's name: Clancy
Farm's name: Clancy's Farm
Farm Type: Standard
Time Played: 234 hours 24 minutes
Game Version: 1.5.6
Clancy has earned 23854190g.
Clancy is married to Abigail.
Clancy has 2 kid(s) named Xyloto and Mylo.
Clancy has collected 7 Stardrop(s).

Process finished with exit code 0
```

Fig. 4. Using the program with test file Clancy_241495642



```
C:\Users\Nathan\AppData\Local\Programs\Python\Python39\python.exe
Farmer's name: Mark
Farm's name: Backles Farm
Farm Type: Standard
Time Played: 22 hours 58 minutes
Game Version: 1.5.4
Mark has earned 51190g.
Mark is married to None.
Mark has 0 kid(s).
Mark has collected 0 Stardrop(s).

Process finished with exit code 0
```

Fig. 5. Using the program with test file Mark_256552446

The test save files showcased here are also available at the GitHub repository of this paper, accessible via the following link: https://github.com/tastytypist/save-file-analyser.

V. Conclusion

The Boyer-Moore algorithm is one the most well-known string-matching algorithm with broad implementation. In this paper, I present a simple use case for the algorithm in the form of save file analyser of the game Stardew Valley. I have also showcased that the program works as expected. Unfortunately, the program still has a lot of limitation regarding its ability.

As a suggestion for future researchers who are interested in this topic, it may be advisable to further expand the analyser to support perfection tracking, museum tracking, fish caught tracking, food cooked tracking, item crafted tracking, and social tracking. Such improvement will surely increase the functionality of the analyser, thus helping players to better understand and keep track of their progress during the game.

VI. Acknowledgment

This paper would not have been possible without the opportunities given by my lecturer, Dr. Ir. Rinaldi Munir, M.T. His enthusiasm, curiosity, and ability to bring out the best of his student motivated me to explore and push my limits in my study. I would like to also thank Mr. Eric Barone for creating a stunning video game loved by millions of people around the world.

References

[1] Kleene, S.C., 1956. Representation of Events in Nerve Nets and Finite Automata. *Automata Studies. (AM-34)*, 34, pp.3–42. https://doi.org/10.1515/9781400882618-002.

[2] Milewski, R. and Govindaraju, V., 2008. Binarization and cleanup of handwritten text from carbon copy medical form images. *Pattern Recognition*, 41(4).

[3] Poddar, A., Sahidullah, M. and Saha, G., 2017. Speaker Verification with Short Utterances: A Review of Challenges, Trends and Opportunities. IET Biometrics, 7. https://doi.org/10.1049/iet-bmt.2017.0065.

[4] Boyer, R.S. and Moore, J.S., 1977. A fast string searching algorithm. *Communications of the ACM*, 20(10), pp.762–772. https://doi.org/10.1145/359842.359859.

[5] Galil, Z., 1979. On improving the worst case running time of the Boyer-Moore string matching algorithm. *Communications of the ACM*, 22(9), pp.505–508. https://doi.org/10.1145/359146.359148.

[6] Apostolico, A. and Giancarlo, R., 1986. The Boyer–Moore–Galil String Searching Strategies Revisited. *SIAM Journal on Computing*, 15(1), pp.98–105. https://doi.org/10.1137/0215007.

[7] Yngve, V.H., 1958. A Programming Language for Mechanical Translation. *Mechanical Translation*, 5(1).

[8] Gimpel, J.F., 1973. A theory of discrete patterns and their implementation in SNOBOL4. *Communications of the ACM*, 16(2), pp.91–100. https://doi.org/10.1145/361952.361960.

[9] Turchin, V.F., 1986. The concept of a supercompiler. *ACM Transactions on Programming Languages and Systems*, 8(3), pp.292–325. https://doi.org/10.1145/5956.5957.

[10] Weiner, P., 1973. Linear pattern matching algorithms. In: *14th Annual Symposium on Switching and Automata Theory (swat 1973)*. United States of America: IEEE. https://doi.org/10.1109/swat.1973.13.

[11] Knuth, D.E., Morris, Jr., J.H. and Pratt, V.R., 1977. Fast Pattern Matching in Strings. *SIAM Journal on Computing*, 6(2), pp.323–350. https://doi.org/10.1137/0206024.

[12] Zhang, H., 2011. *Parallelization of a software based intrusion detection system - Snort*. Thesis.

[13] Levitin, A., 2012. *Introduction to the Design & Analysis of Algorithms*. Essex: Pearson..

[14] Davidson, A., 2006. *Pattern Matching*. [online] fivedots. Available at: <http://fivedots.coe.psu.ac.th/Software.coe/LAB/PatMatch>

[15] Barone, E., n.d. *Stardew Valley - Press*. [online] Stardew Valley. Available at: <https://www.stardewvalley.net/press/>.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 23 Mei 2021

[illegible]

Jeremy S.O.N. Simbolon
13520042