International Journal of Computer Applications (0975 – 8887)
Volume 183 – No. 52, February 2022

Pattern Matching in File System

Kuldeep Vayadande
Department of Artificial Intelligence and Data Science
Vishwakarma Institute of Technology
Pune, India

Ram Mandhana
Department of Artificial Intelligence and Data Science
Vishwakarma Institute of Technology
Pune, India

Kaustubh Paralkar
Department of Artificial Intelligence and Data Science
Vishwakarma In3.stitute of Technology
Pune, India

Dhananjay Pawal
Department of Artificial Intelligence and Data Science
Vishwakarma Institute of Technology
Pune, India

Siddhant Deshpande
Department of Artificial Intelligence and Data Science
Vishwakarma Institute of Technology
Pune, India

Vishal Sonkusale
Department of Artificial Intelligence and Data Science
Vishwakarma Institute of Technology
Pune, India

ABSTRACT

String matching algorithms have had a significant impact on computer science and are used to solve a variety of real-world challenges. It aids in the completion of time-saving tasks in a variety of fields. These techniques can be handy when looking for a string within another string. Database schema and network systems both employ string matching. We have designed and used several pattern matching algorithms like Naive, KMP(Knuth Morris Pratt) and Automation Matcher algorithm to find patterns in the file system. The proposed system goes through three main steps; Take the names of the files present in the current directory; Take the pattern to search and type of algorithm from the user; And display the pattern if found in the directory. The program is able to accurately detect the patterns in the file system using the mentioned algorithms successfully.

Keywords

Pattern, String, Text-Editing, Pattern-Matching, KMP, Naive algorithm

1. INTRODUCTION

In today's connected world we need to search different things everywhere on our computers. Pattern matching is the process of checking if a sequence of characters is present in the given data. For this various string matching or searching algorithms are used. These algorithms are very useful as they play a vital role in many fields to perform these tasks in an efficient time. They are used in real life applications like Plagiarism Detection, DNA sequencing, Spell Checker, Spam Filters and many more.

In this paper, we study three pattern matching algorithms namely Naive Algorithm, KMP Algorithm and FA. The proposed program extracts names of the files from a given directory and uses the mentioned algorithms to find a specific pattern. If that pattern is found in the file system, then we display the name of that file.

[3] Donald T. Campbell stated the term “pattern identification” as a characteristic of qualitative analysis which he defines as comprehensive rather than atomic.

Fabrice Le Fessant [1][10] states that the key feature of functional languages is pattern matching. The paper tells how the compiler code is better than human code based on two facts that are compactness and efficiency. The reason behind this is it considers the matching as a whole and it knows some details of runtime error such as representation of values. The paper mainly focuses on producing faster code in the back tracking framework.

2. LITERATURE REVIEW

F. Lupus Fess ant and L. Marange [1] have created 2 contributions to paper pattern matching. They 1st gave AN upgrade to the standard technique of collecting pattern matching phrases into backtracking automata, that has remained much unchanged for the past fifteen years. Their advancements end in speedier automata, that mitigates the downside of backtracking automata in sensible things. Moreover, the structure of automata is unbroken, conserving the very fascinating attribute of output size being linear in input size. Second, they counsel a way for efficiently assembling or-patterns with variables whereas maintaining output size dimensionality. victimization or-pattern rather than various clauses with similar actions leads in programs that square measure additional compact and simple. These blessings square measure currently on the market to milliliter programmers without worrying of poor runtime performance or code size explosion.

Richard M. Karp Michael zero. Rabin [2] have seen that randomizing a collection of simply computed and updatable fingerprints ends up in comparatively straightforward and economical algorithms for a variety of one-dimensional and multidimensional pattern-matching drawback problems. The necessary purpose is that they will be proved. strategies that cause expected computations being completed in an exceedingly short quantity of your time or in real time with a negligible probability of quality, for every and each pattern text try. several variants and extra uses exist for the ideas and strategies given here. The author, particularly, has discovered a brand-new kind of Polynomials over finite domains square measure accustomed to produce fingerprint functions rather than numbers, use fields.

DONALD E. Knuth, JAMES H. MORRIS, JR.: l: AND vocalizer R. PRATT [3] investigated text-editing programs that square measure oft needed to look through a string of characters for instances of a given pattern string, likewise as all positions, or probably solely the left position, within which the pattern happens as a contiguous substring of the text. the foremost apparent technique for locating an identical pattern is to look at each starting purpose within the text then quit the search once an accurate letter is found. during this article, they supply a pattern matching technique that finds all instances of a pattern of length m within a text of length n in O(m+n) units of your time. If the text is scan from AN external file, this method solely needs O(m) internal memory locations and O(log m) units of your time between single character inputs.[11]

Donald T. Campbell [4] fictional the phrase "pattern identification" to explain the holistic (i.e., evaluating the pattern) instead of atomistic (i.e., examining its elements) nature of chemical analysis. He declared that one case study style could supply a sturdy foundation for analysis. A theory is place to the take a look at if a whole set of expectations is drawn from it (all of which might be true if they were all true). In such instance, an "anticipated pattern") could also be incontestable to be true. This was conjointly dubbed by Joseph Campbell a "configurational strategy" during this arrangement, he claimed, chemical analysis tends to as a result of every sequential take a look at should negate instead of making sure past beliefs, the take a look at should negate instead of making sure previous beliefs. The determined constituent of a pattern or arrangement is precisely as expected. As Thomas has found out, the strength of this "non-equivalent, dependent variables" hypothesis was incontestable by D. Cook and Donald T. Campbell. The term "design" refers to the fact that the variables that conjure the pattern or configuration don't seem to be comparable.[12].

3. METHODOLOGY

For this project, to find patterns in the file system, several pattern matching algorithms such as Naive, KMP (Knuth Morris Pratt), and Finite Automata were used. Following are three main steps in the proposed system: Take the names of the files in the current directory; get the pattern to search for and the algorithm type from the user; and, if the pattern is found in the directory, display it. Using the algorithms mentioned, the program is able to accurately detect patterns in the file system.

Naive algorithm for Pattern Searching:The Naive algorithm [1], checks for a match by sliding the pattern over the text one by one. If a match is found, slide the slider by 1 to check for more matches. The best-case scenario is when the pattern's first character does not appear in the text at all.

Naive algorithm for Pattern Matching:This algorithm [1], checks for a match by gliding the pattern over the text singly. If a complement is discovered, slide the slider by 1 to examine for more similarities. The most satisfactory case scenario is when the pattern's initial character is not present in the text at all.

KMP Algorithm for Pattern Searching: When there are many complementing characters followed by an ill-matched character, the above pattern searching algorithm fails. The KMP matching rule takes advantage of the pattern's degenerating property (patterns with constant sub-patterns showing quite once within the pattern) to cut back the worst-case complexity to O(n). The fundamental principle of KMP's algorithm is that by the time it identifies a discrepancy (a few matches later), few of the characters in next windows text are already familiar to us. We access this data to keep away from matching texts that we are aware will be the same in any case. We differentiate all characters at every shift and shift the design in succession, then utilize a value from the array to determine which characters to match next.

Pattern search algorithm based on Finite Automata (FA):It processes the pattern beforehand and builds a two-dimensional array representing a Finite Automata in the FA-based algorithm. The most difficult component of this algorithm is constructing the FA. Once the FA is in place, searching is a breeze. To find the first state of the automata and the first character of the text, we simply need to start at the beginning. We consider the next character of text at each step, look for the next state in the built FA, and then move to a new state. If we get to the end, the pattern will be found in the text.

3.1 Proposed System

This Pattern matching in file system project helps to search a pattern from a larger string that is the filename. Various types of algorithms can be implemented for this. Their main objective is to improve the time complexity. The traditional way may take a longer time to do the same task.

We have created a menu driven program for pattern matching, where the menu consists of the three types of algorithms we had implemented - Naive Algorithm, KMP Algorithm and Finite Automata Algorithm. Then according to the algorithm selected, the working will be decided but overall, the objective is to take the names of the files present in the current directory, then take the pattern to search and type of algorithm from the user and finally display the pattern if found in the directory.

Steps for execution:

Step 1: The files present in the given directory will be displayed.

Step 2: Then, the user must enter a pattern of their choice to search in the file system.

Step 3: The user will then have to enter their choice of algorithm.

Step 4: If the pattern is found in the file, then it will be displayed.

3.2 Flowchart

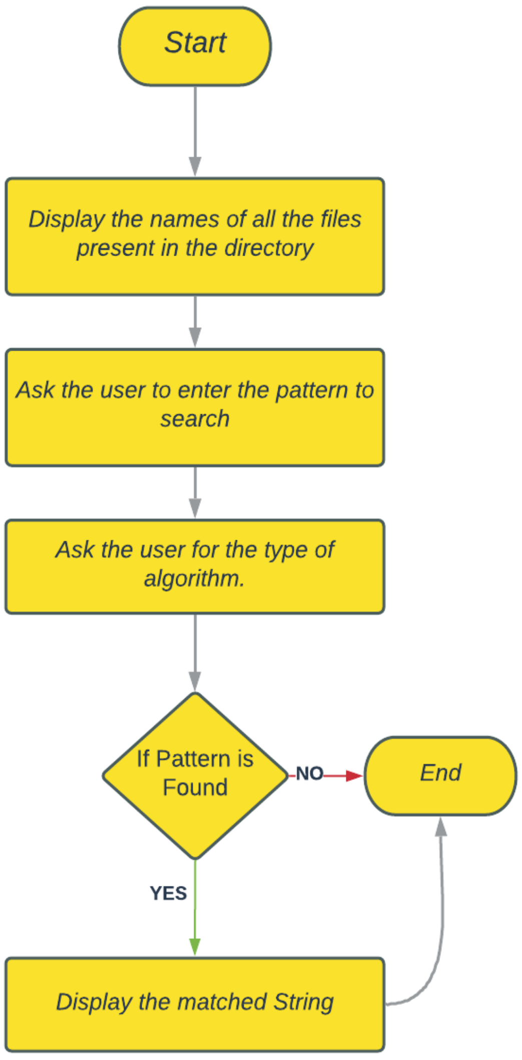

Start
Display the names of all the files present in the directory
Ask the user to enter the pattern to search
Ask the user for the type of algorithm.
If Pattern is Found
NO
End
YES
Display the matched String

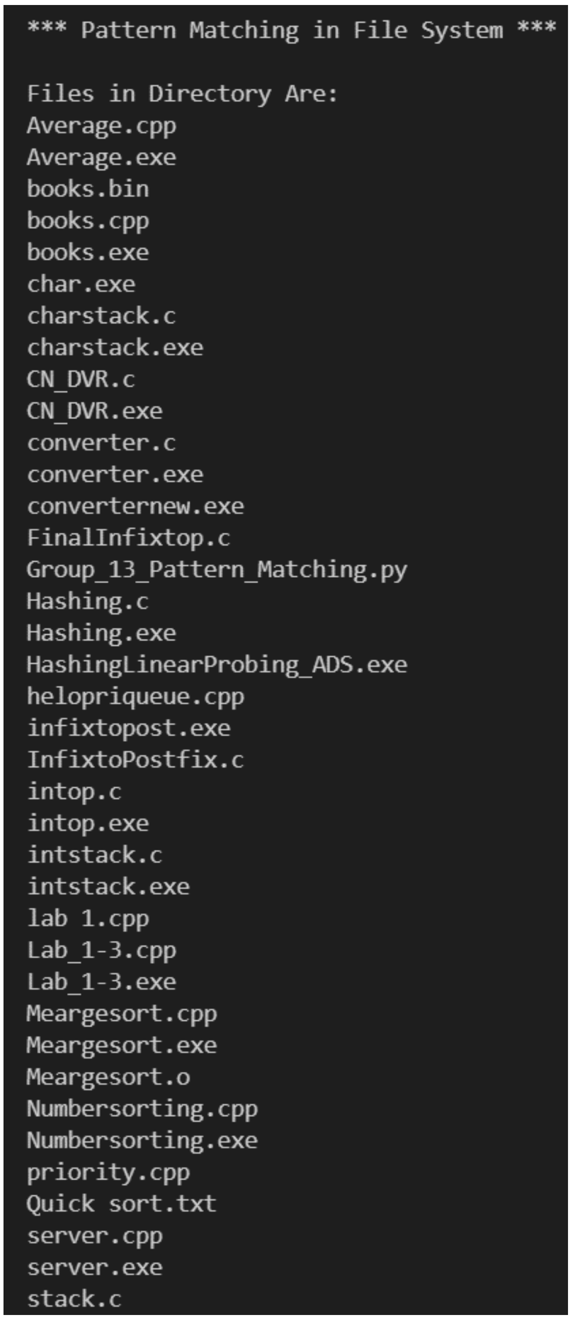
Fig 1: Flowchart of the project

4. RESULTS AND DISCUSSIONS

The three pattern searching algorithms used in our project can accurately find the matching pattern in the given file directory.

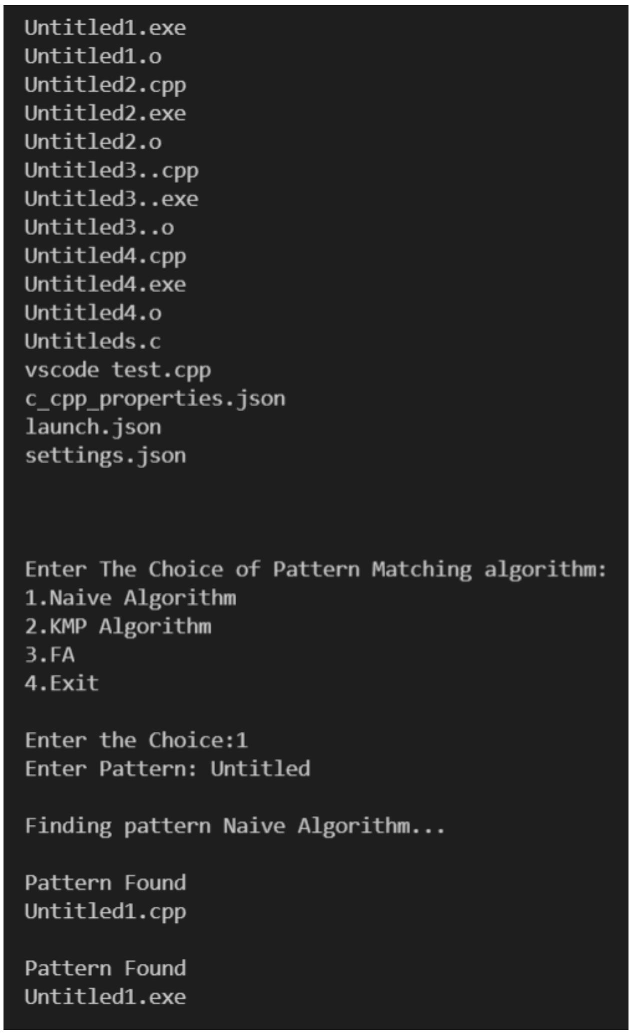
The user needs to first specify the directory in which the searching will take place. After that, the program will display the names of all the files present in the specified directory. Then the user has to enter his/her choice of algorithm from the list. Then, the user has to enter the pattern they have to search in the file system.

After doing this, if the given pattern is found in the file, it will be displayed.



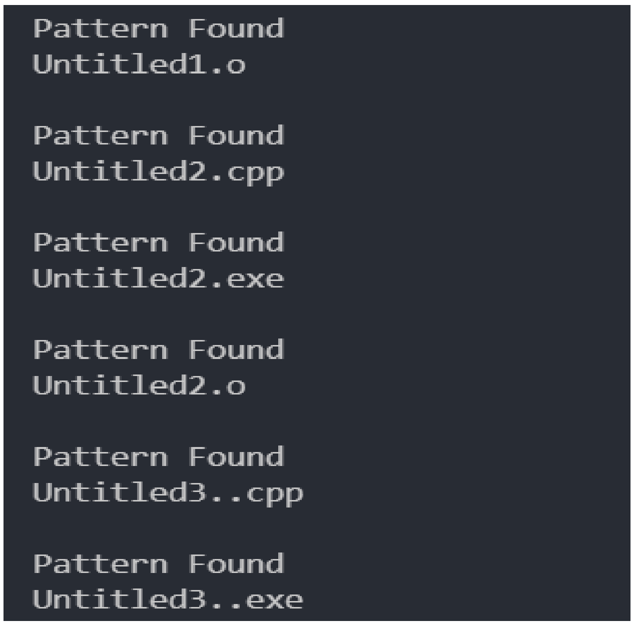
```
*** Pattern Matching in File System ***

Files in Directory Are:
Average.cpp
Average.exe
books.bin
books.cpp
books.exe
char.exe
charstack.c
charstack.exe
CN_DVR.c
CN_DVR.exe
converter.c
converter.exe
converternew.exe
FinalInfixtop.c
Group_13_Pattern_Matching.py
Hashing.c
Hashing.exe
HashingLinearProbing_ADS.exe
helopriqueue.cpp
infixtopost.exe
InfixtoPostfix.c
intop.c
intop.exe
intstack.c
intstack.exe
lab 1.cpp
Lab_1-3.cpp
Lab_1-3.exe
Meargesort.cpp
Meargesort.exe
Meargesort.o
Numbersorting.cpp
Numbersorting.exe
priority.cpp
Quick sort.txt
server.cpp
server.exe
stack.c
```

Fig 2: Files present in the directory are displayed



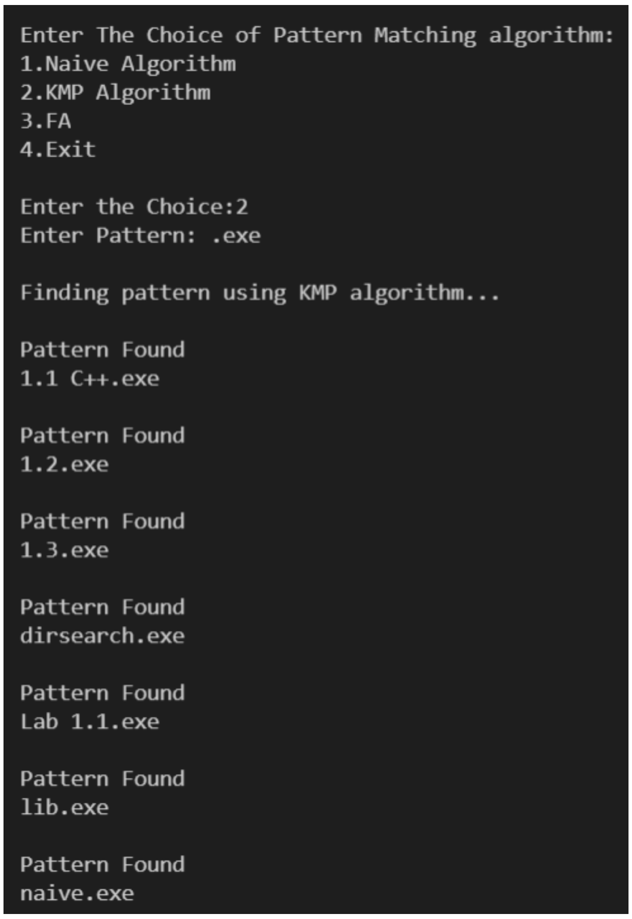
```
Untitled1.exe
Untitled1.o
Untitled2.cpp
Untitled2.exe
Untitled2.o
Untitled3..cpp
Untitled3..exe
Untitled3..o
Untitled4.cpp
Untitled4.exe
Untitled4.o
Untitleds.c
vscode test.cpp
c_cpp_properties.json
launch.json
settings.json



Enter The Choice of Pattern Matching algorithm:
1.Naive Algorithm
2.KMP Algorithm
3.FA
4.Exit

Enter the Choice:1
Enter Pattern: Untitled

Finding pattern Naive Algorithm...

Pattern Found
Untitled1.cpp

Pattern Found
Untitled1.exe
```

Fig 3: Pattern is found using Naive Algorithm.



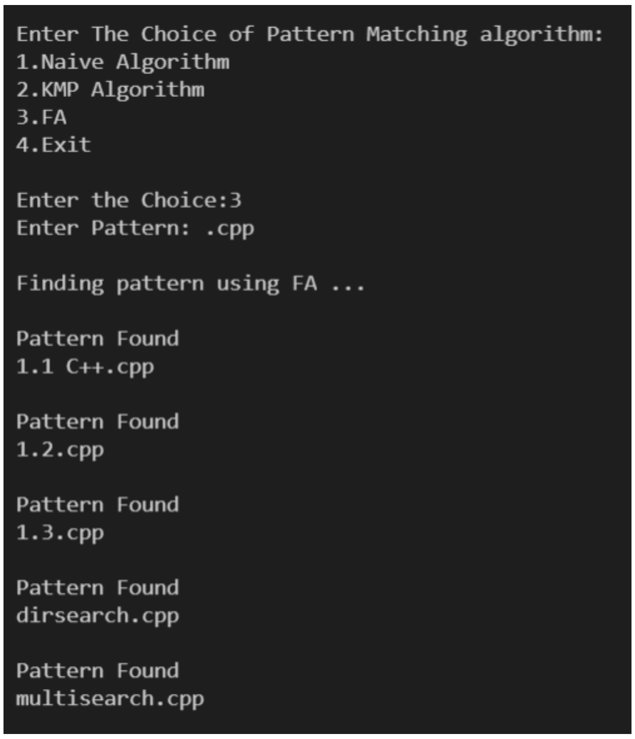
```
Pattern Found
Untitled1.o

Pattern Found
Untitled2.cpp

Pattern Found
Untitled2.exe

Pattern Found
Untitled2.o

Pattern Found
Untitled3..cpp

Pattern Found
Untitled3..exe
```

Fig 4: 'Untitled' Pattern Is Found



```
Enter The Choice of Pattern Matching algorithm:
1.Naive Algorithm
2.KMP Algorithm
3.FA
4.Exit

Enter the Choice:2
Enter Pattern: .exe

Finding pattern using KMP algorithm...

Pattern Found
1.1 C++.exe

Pattern Found
1.2.exe

Pattern Found
1.3.exe

Pattern Found
dirsearch.exe

Pattern Found
Lab 1.1.exe

Pattern Found
lib.exe

Pattern Found
naive.exe
```

Fig 5: Pattern is found using KMP algorithm.



```
Enter The Choice of Pattern Matching algorithm:
1.Naive Algorithm
2.KMP Algorithm
3.FA
4.Exit

Enter the Choice:3
Enter Pattern: .cpp

Finding pattern using FA ...

Pattern Found
1.1 C++.cpp

Pattern Found
1.2.cpp

Pattern Found
1.3.cpp

Pattern Found
dirsearch.cpp

Pattern Found
multisearch.cpp
```

Fig 6: Pattern is found using FA

5. LIMITATIONS

- Only three algorithms are implemented in this project. More algorithms can be implemented.
- The program will display the whole name of the matched pattern filename.

6. CONCLUSION

In this project we investigated how different pattern searching algorithms work and how they are used in our day-to-day life. We started from looking at simple algorithms like naive to more advanced algorithms like KMP. We demonstrated the matching of patterns in a given file system. All the three algorithms were able to successfully find the matching pattern in the file system accurately.

7. FUTURE SCOPE

- Making it more user friendly by adding a graphical user interface to the project.
- Implementing more pattern matching algorithms into the program

8. REFERENCES

[1] F. Le Fessant and L. Maranget, “Optimizing pattern matching,” ACM SIGPLAN Not., vol. 36, no. 10, pp. 26–37, Oct. 2001, Doi: 10.1145/507669.507641.

[2] R. M. Karp and M. O. Rabin, “EFFICIENT RANDOMIZED PATTERN-MATCHING ALGORITHMS.,” IBM J. Res. Dev., vol. 31, no. 2, pp. 249–260, 1987, Doi: 10.1147/RD.312.0249.

[3] D. E. Knuth, J. James H. Morris, and V. R. Pratt, “Fast Pattern Matching in Strings,” http://dx.doi.org/10.1137/0206024, vol. 6, no. 2, pp. 323–350, Jul. 2006, Doi: 10.1137/0206024.

[4] D. T. Campbell, “Pattern Matching,” Comp. Polit. Stud., vol. 8, no. 2, pp. 178–193, 2009, Doi: 10.1177/001041407500800204.

[5] M. C. Johannes Meyer, A. Singhal, and D. E. Seaborg, “Pattern matching in historical data,” AIChE J., vol. 48, no. 9, pp. 2022–2038, Sep. 2002, Doi: 10.1002/AIC.690480916.

[6] N. Sinkovics, “Pattern Matching in Qualitative Analysis,” SAGE Handb. Qual. Bus. Manag. Res. Methods Challenges, pp. 468–484, May 2018, Doi: 10.4135/9781526430236.N28.

[7] W. M. K. Trochim, “Outcome pattern matching and program theory,” Eval. Program Plan., vol. 12, no. 4, pp. 355–366, Jan. 1989, Doi: 10.1016/0149-7189(89)90052-9.

[8] P. Weiner, “Linear pattern matching algorithms,” pp. 1–11, Jul. 2008, Doi: 10.1109/SWAT.1973.13.

[9] “Pattern Matching by Tony Hack, Jan Dull: SSRN.” https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1433934 (accessed Dec. 28, 2021).

[10] Raza, Mir Adil, Kuldeep BabanVayadande, and H. D. Preetham. *"DJANGO MANAGEMENT OF MEDICAL STORE."*, International Research Journal of Modernization in Engineering Technology and Science, Volume:02/Issue:11/November -2020

[11] K.B. Vayadande, Nikhil D. Karande,” *Automatic Detection and Correction of Software Faults: A Review Paper*”, International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321-9653, Volume 8 Issue IV Apr 2020.

[12] KuldeepVayadande, RiteshPokarne, Mahalaxmi Phaldesai, TanushriBhuruk, Tanmai Patil, Prachi Kumar, “*SIMULATION OF CONWAY’S GAME OF LIFE USING CELLULAR AUTOMATA”* International Research Journal of Engineering and Technology (IRJET), Volume: 09 Issue: 01 | Jan 2022, e-ISSN: 2395-0056, p-ISSN: 2395-0072

[13] VaradIngale, Kuldeep Vayadande, Vivek Verma, Abhishek Yeole, Sahil Zawar, Zoya Jamadar. **Lexical analyzer using DFA**, International Journal of Advance Research, Ideas and Innovations in Technology, www.IJARIIT.com.

[14] Kuldeep Vayadande, Harshwardhan More, Omkar More, Shubham Mulay, Atharva Pathak, VishwamTalnikar, “ Pac Man: Game Development using PDA and OOP”, International Research Journal of Engineering and Technology (IRJET), Volume: 09 Issue: 01 | Jan 2022, e-ISSN: 2395-0056, p-ISSN: 2395-0072.

[15] Rohit Gurav, Sakshi Suryawanshi, ParthNarkhede, Sankalp Patil, SejalHukare, Kuldeep Vayadande,” Universal Turing machine simulator”, International Journal of Advance Research, Ideas and Innovations in Technology, (Volume 8, Issue 1 - V8I1-1268, ISSN: 2454-132X