Application of String Matching in Identifying the Imitation of a Theme in a Fugue

Ahmad Alfani Handoyo – 13520023
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520023@std.stei.itb.ac.id

***Abstract*— A fugue is a style of composition dominantly used in the baroque period where a main theme is introduced at the beginning and is then repeated in imitation frequently in a contrapuntal manner. This paper aims to identify the imitation of the initial theme by applying a brute-force string-matching algorithm of the theme to the rest of the fugal composition. Successive note intervals are used to represent each note in the composition to address the issue of imitation in different pitches.**

Keywords—string matching; fugue; theme and imitation.

I. Introduction

For a lot of people, western classical music holds a special place and feel that cannot be replicated with other genres of music. Whether that person is a casual listener or a music enthusiast, there is something in classical music for everyone. A college undergrad may tune in to a Chopin étude for the purposes of studying due to its calming nature whereas a music professor may crank up to a Bach fugue studying its counterpoint structure. Classical music resonates with a wide range of demographic in such a raw manner due to its unlimited potential in expressing emotions through harmony, beauty, and grandiosity.

Classical music has stood the test of time because of its unsurpassed quality. For many generations, musicians and historians alike have studied it thoroughly and extensively, digging deep down into its roots and structures. Right now, classical music is being taught in many prestigious music schools and conservatoriums all around the world. Although many critics say that the choice of using classical music in everyday music curriculum is considered music whitewashing, the truth be told that classical music provides an outstanding platform for musical students and professionals to delve down into the bare structure and intricacies of music such as harmony, tone, rhythm, and emotion.

Perhaps an aspect of music that are important and go hand in hand especially with classical music is virtuosity. Classical music provides a unique pedestal for musicians of all backgrounds to show off their virtuosity. Whether that someone is a composer or a player, if one is to put the effort into practicing hard enough, that someone can eventually become masterfully skilled in their arts. Sometimes, this display of skill supersedes what is previously thought humanly possible, reaching a level of virtuosity able to bedazzle an outsider into appreciating the effort going in to masterfully craft the art.

Throughout history, there have been many virtuosos that completely changed and revolutionized music. For pianists, besides composers such as Sergei Rachmaninoff and Frédéric Chopin, Franz Liszt may be the figure that many look up to as the piano virtuoso. His compositions break the barriers on what is humanly thought possible on the piano. Extremely advanced and difficult techniques were no longer the oddity but the norm to play his pieces. Yet, even with all these difficulties his works are cherished and frequently played in many recitals because they bring out the best in what a pianist can do. Pianists look up to his work as their virtuosic writing allows them to flex their talent, putting on a performance able to amaze the audience.

Going back further, Liszt took many inspirations from previous virtuosos. Perhaps his most notable inspiration is violinist and composer Niccolò Paganini. Many violinists tremble when they hear the name Paganini due to his extremely virtuosic violin writing and playing. His 24 Caprices for Solo Violin sets the bar for many world class violinists. The techniques required to play these caprices take years to develop and many more to master and perfect. His work not only affects violin playing but also classical music in its entirety. Many other composers take inspiration from Paganini's caprices, setting its themes and variations in new virtuosic settings in many different instruments.

It is undoubtable that in the case of virtuosity in classical music, Johann Sebastian Bach reigns supreme. Many musicians even consider Bach as the father of classical music. Throughout his lifetime, he has composed more than a thousand pieces and works. These compositions are considered sacred, leaving a legacy behind for generations of musicians and composers to come. His works treat harmony and structure in a very virtuosic manner, in a precision like no other composers that has come after him. His use of polyphony, tonality, and counterpoint surpasses many before him in such a beautiful accordance, almost such in a mathematical accuracy.

His ingenuity is perhaps most notably shown in his fugal works. Fugues are well known to be extremely difficult to compose due to its strict counterpoint nature. Any musician who has tried to compose a fugue should know the difficulties that arise in trying to be establish a fugal structure yet

maintaining a harmonious piece that is pleasant to the ear. Yet, throughout his lifetime Bach has written hundreds of fugues almost with such an ease unlike any other composers. One of his works on fugues, fittingly called the Art of the Fugue, has been thoroughly studied by many music experts and professionals in hopes to gain knowledge of this virtuosic style of writing that is the fugue.

This paper aims to provide a better understanding of the fugue by trying to analyze the subject, the imitation of the initial theme, by applying string matching of the theme to the rest of the fugal composition. The automation of identifying the variations of the theme can provide a better experience in trying to understand a fugue and its intricacies. Hopefully, this paper can serve its function as a tool to help music experts and professionals in studying the structure of fugues.

II. Basic Theory

A. String Matching

String matching is a type of algorithm that is used to search the occurrences of a pattern in a text. The pattern is a sequence of characters in a string which has *m* length of characters. The text is also a sequence of characters in a string which has *n* length of characters. What uniquely identifies the pattern from the text is their length. The text has a way longer length of characters than the pattern, or in other words $m << n$, much so that the pattern is tiny compared to the length of the txt. A string is a data structure or type that contains within it a sequence or a combination of letters, numbers, and other characters (whitespace, punctuation marks, etc.). An example of string matching is as shown below with text *T* and pattern *P*.

T = The girl from Ipanema goes walking

P = Ipanema

Here the pattern *"Ipanema"* is searched upon the text *"The girl from Ipanema goes walking"* and is found to start at character of index 14.

There are two very important concepts concerning a string structure. The prefix and the suffix. The prefix is a character, or a sequence of characters that appear at the start of a string. The suffix on the other hand is a character, or a sequence of characters that appear at the end of a string. Say there is a string *S* with total number of characters *m*. A prefix *p* is a substring of S such that *p* has index *S[0...k]*, where *k* is an index anywhere between 0 and *m-1*. On the other hand, a suffix *s* is a substring of *S* such that *s* has index *S[k...m-1]*.

For example, take a string *S* with a sequence of characters "Ipanema". All possible prefixes of *S* are *"I"*, *"Ip"*, *"Ipa"*, *"Ipan"*, *"Ipane"*, *"Ipanem"*, and *"Ipanema"*. On the other hand, all possible suffixes of *S* are *"a"*, *"ma"*, *"ema"*, *"nema"*, *"anema"*, *"panema"*, and *"Ipanema"*.

String matching is used for a variety of problem solving in the computer science world, especially for ones dealing with strings. A few examples include a search engine service such as Google, Bing, and Yahoo, detecting plagiarism in a document, and in the field of bioinformatics where DNA sequences are matched to find patterns.

There is already a dozen of string-matching algorithms currently existing, each with their own pros and cons. In this paper however, only the brute-force algorithm is discussed.

B. Brute-Force Algorithm

A brute-force algorithm for string matching, like many other brute-force applications will in theory always find the solution to any given problem which in this case is the occurrence of a pattern in a text. However, in searching for this solution, every possible solution must be explored. Thus, the outcome of the solution may not be the most optimal.

In general, a brute-force algorithm for string matching entails this sequence of steps. First, start the string matching from the start of the text, which in this case starts from the left and ends on the right. In this state, match each character one by one from left to right from the pattern to the text in the same position. If all characters on the pattern match up with the ones on the text, then the pattern is considered to have been matched with the text and the algorithm is halted. However, if a character on the pattern at any point does not match with a character on the text in the same position, shift the pattern to the right by one on the text and repeat the matching process previously stated again until the right end of the text.

It is important to note that the matching process and the shifting ensures that with each new shift on the text, the matching begins again at the start index of the text. If the right end of the text is reached and the pattern is still not yet matched to the text, it can be concluded that the pattern does not exist on the text.

```
  NOBODY NOTICED HIM
1 NOT
2  NOT
3   NOT
4    NOT
5     NOT
6      NOT
7       NOT
8        NOT
```

Fig. 2.1. Example of brute-force algorithm string matching.

An example of the algorithm is shown above with the pattern *"NOT"* and text *"NOBODY NOTICED HIM"*. The algorithm starts at the first index of the text and compares the pattern to the text. At the first index the letter *N* and *O* matches with the text, but the *T* does not and so the pattern shifts to the right by one. At the second index, *N* already does not match with the second index of the text with letter *O* and so the pattern shifts again. Repeat this a couple of times. When reaching the eighth index, the letter *N*, *O*, and *T* in the pattern all matches with the sequence of letters on the text starting on the eighth index. Thus, a conclusion can be reached that the pattern exists on the text starting on the eighth index.

With a pattern of length *m* and text of length *n*, a few scenarios arise regarding the time complexity of the brute-force algorithm for string matching.

A worst-case scenario in which a majority of the pattern prefix matches with a lot of characters in the text and/or the pattern is found at the end of the text yields a maximum comparison of *m(n-m+1)* characters, which in turn has a time complexity of *O(mn)*. An example of this is with a text *"aaaaaaaaaaaaac"* matched with a pattern *"aaac"*.

A best-case scenario in which the first character of the pattern never matches with any character of the text yields a time complexity of *O(n)* where the comparison is made once for every character of the text. An example of this is with a text *"Hey Jude don't make it bad zzz"* with a pattern *"zzz"*.

An average-case scenario has the time complexity *O(m+n)*. A time complexity of *O(m+n)* means that for average cases, the brute-force algorithm can be categorized as being quick.

It is important to note that due to its characteristics the brute-force algorithm is fast if the alphabet of the text is large. For example, an everyday alphabet of 26 characters of A-Z. On the other hand, the algorithm becomes slower when the alphabet of the text is small. For example, in the case of binary numbers of between 0 and 1.

C. Music Notation

Music has always been a part of human life. Throughout all cultures across the world, there is also a culture of music. Through the ages, back in prehistoric times until now, humans have always found a use for music. Whether it be just for entertainment or for higher purposes such as religion. However, through time it has become apparent that musical ideas gradually became more complicated and structured. Thus, to communicate these musical ideas between musicians, there needs to be a formalization and standardization of music so as not to give confusion and miscommunication. Each culture has their own way of communicating and expressing musical ideas, each with their own method and purposes. But for the sake of universality, western musical notation has become the standard for expressing musical ideas worldwide.

An octave is an interval between a note and another that has double its vibrating frequency. Western music theory uses a system of twelve-tone equal temperament. This means that in an octave, there are twelve tones that are equally spaced out with a ratio of $\sqrt[12]{2}$ between one tone and the next. The smallest interval between a tone and its neighboring tone is called a semitone. To identify these tones, letters A to G are labelled as shown in the piano octave below.

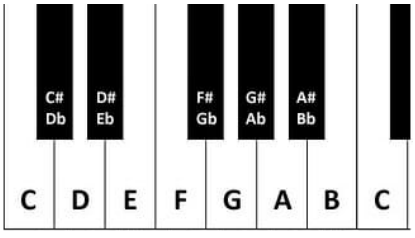


Fig. 2.2. Twelve-tone equal temperament in an octave on a piano.

Notice that letters A to G are not enough to label all the twelve notes in an octave. Therefore, two symbols are introduced to further distinguish these notes. A sharp, ♯, takes a note and transposes it higher by one semitone. For example, a C♯ is a semitone higher than its normal counterpart C. A flat, ♭, takes a note and transposes it lower by one semitone. For example, a B♭ is a semitone lower than its normal counterpart B. As an additional note, a natural, ♮, is used to bring back the pitch of a previously sharpened or flattened note back to its normal pitch. For example, a C♮ is essentially the same as a C.

To write music, a staff acts like a line in normal everyday writing. The staff comprises of five horizontal lines with four spaces between them. Notes are written on these horizontal lines and spaces to represent different pitches or tones.

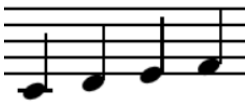


Fig. 2.3. A staff with notes.

To represent time and rhythm, the western music notation uses different note and rest symbols denoting a specific time value.

Fig. 2.4. Semibreve, minim, crotchet, quaver, and semiquaver notes.

Fig. 2.5. Semibreve, minim, crotchet, quaver, and semiquaver rests.

A semibreve holds a beat value of 4. A minim holds a beat value of 2. A crotchet holds a beat value of 1. A quaver holds a beat value of ½. A semiquaver holds a beat value of ¼. The difference between a note and a rest is that during a note, the player of the instrument plays the note with the corresponding pitch, and during a rest, the player does not play anything. Keep in mind there is also a dotted note, which means to add another half value to the note dotted. For example, a dotted semibreve note has a beat value of 6.

These beat values loosely define the time required to hold these notes and rests. The length of a beat is defined by the tempo of the composition where in modern music is represented by how many beats per minute (BPM) there are.

D. Theme

A theme or a subject is the main recognizable material of composition. Usually, it is in the form of a melody that can be heard throughout the composition. The theme serves as the building block to which the entire composition is built upon. Throughout a composition, the theme may be changed to give its variations as an exploration of musical ideas.

Perhaps the most famous example of a theme in the western classical music repertoire is the Ode to Joy theme in the fourth movement of Ludwig van Beethoven's Symphony No. 9 in D minor, Op. 125, first introduced in bar 92 in the strings section.

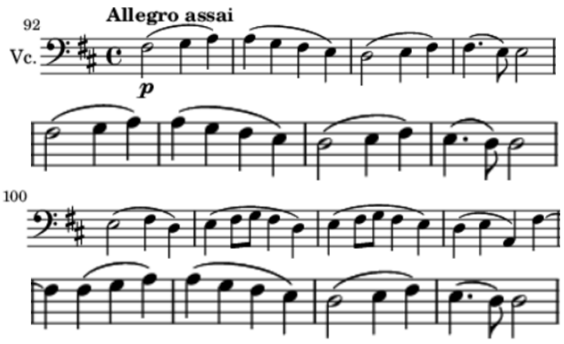


Fig. 2.6. Ode to Joy theme from Beethoven's Symphony No. 9.

This theme, like a good theme is supposed to function, stands out and is memorable. It is then explored intensively all throughout the symphony's fourth movement through a variety of variations, including a turkish march section, a double fugue with another theme, and several codas near the end.

E. Fugue

A fugue is a style of composition dominantly used in the baroque period of western classical music. It comprises of a main theme that is introduced at the beginning of the composition which is then repeated successively in imitation frequently throughout the whole composition. Each voice or each melody line imitates the initial theme in a contrapuntal manner through a variety of techniques from transposing the theme to a different pitch all the way to the inversion of the theme.

An example of a fugue with its theme and imitations can be found on the fugue section of Johann Sebastian Bach's Prelude and Fugue in C major, BWV 846 from his Well-Tempered Clavier, Book I.

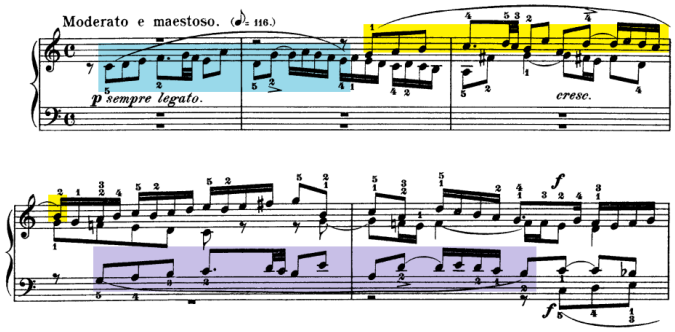


Fig. 2.7. Main theme (light blue) with imitations (yellow and light purple).

Fig. 2.7 shows how in the fugue section of Prelude and Fugue in C major the theme is introduced in the start highlighted in light blue. Then, the theme is imitated in another voice entering bar 3, transposed up to a higher pitch highlighted in yellow. The theme is imitated yet again in bar 4 in another voice, in a lower pitch as shown in light purple. This imitation of the theme continues to happen all throughout the composition.

What makes a fugue difficult to compose is the challenge to keep the imitation as if to sound independent from each other, acting as their own melodic lines. In other words, to maintain its counterpoint nature. An imitation of a theme is not enough, but how the imitation of the theme sounds pleasant to the ear harmonically proves to be a challenge.

III. Implementation

The identification of the theme imitation in a fugue will be found by using the brute-force string-matching algorithm. As the theme is a short passage that is being matched and searched in the composition in hopes of finding its imitations, it takes the place of the pattern. The composition itself will represent the text being matched upon by the pattern. The result of the string-matching algorithm will be the imitations of the theme found in the composition.

It is obvious that the notes making up the composition represent the characters in a string for either the pattern or text. However, simply representing a character by their note (for example, C, D, B♭, F♯) becomes a problem. If the imitation starts and ends in the same pitch as the theme, then this approach is safe. For example, if we have a theme with the notes C-D-E-F-G which is then imitated yet again with the same notes C-D-E-F-G. The imitation will surely be identified by the approach of representing a character with the note.

This approach becomes a problem if the imitation is in a different pitch. Yet, it is fundamental to composing a fugue that the imitation starts on a different pitch, so that the composition stays harmonically pleasant and moving. For example, the same previous theme example with notes C-D-E-F-G and is then repeated in imitation in the composition with the notes G-A-B-C-D. Musically, this succession is completely legal to be considered an imitation of a theme in a fugue as they both have the same intervals harmonically. But because the previous approach string matches the note name, this imitation will not be identified. This is already apparent from the first note of the theme C and imitation G which are clearly different.

To fix this issue, a different approach is chosen to represent the notes in the pattern and the text. Rather than simply using the symbols of the notes, the interval between a note and the next is used instead. These intervals between notes then make up an array of intervals. An increase in one semitone is represented by the value 1, whereas a decrease in one semitone is represented by -1. Other intervals follows and no change in pitch equals 0. For example, take the simple melody below as a theme.

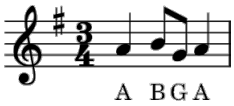


Fig. 3.1. A simple theme.

The simple theme shown in Fig. 3.1 with notes A-B-G-A can be represented with the interval 2 (from A to B), -4 (from B to G), and 2 (from G to A). Then, follows an imitation as below.

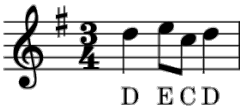


Fig. 3.2. An imitation of previous theme.

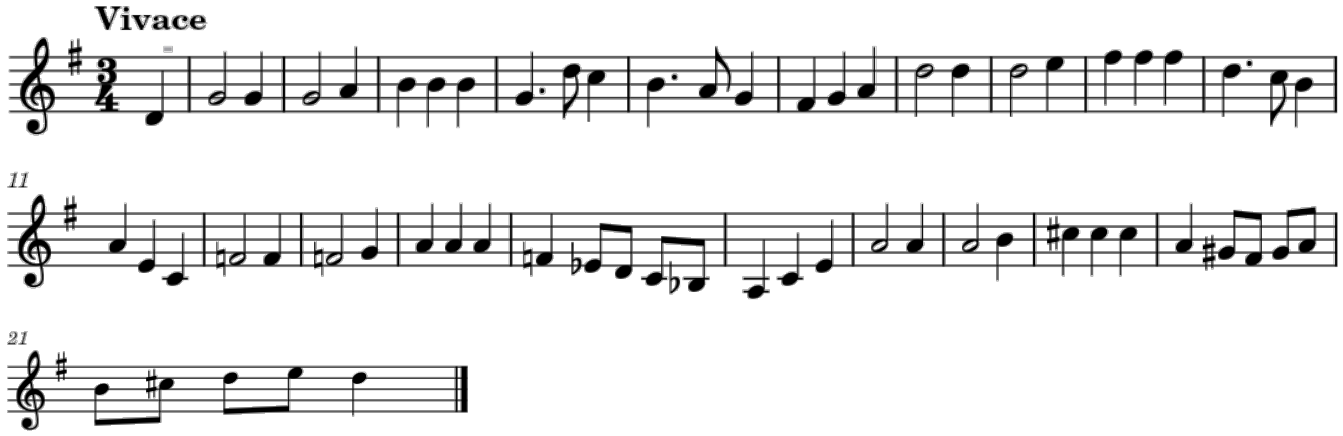


Fig. 3.3. A short fugal excerpt based on the melody of Naik Naik ke Puncak Gunung.

The imitation shown in Fig. 3.2 with notes D-E-C-D can be represented with the interval 2 (from D to E), -4 (from E to C), and 2 (from C to D). This sequence of interval is the same as the theme. And so, this approach allows the identification of the imitation even when having different pitches, but still maintaining the same semitone intervals which covers a lot of scenarios in the imitation of a theme in a fugue.

Further assumptions are made to model the problem at hand. First, because the approach to use the interval between two notes are used, this means any rests between two notes are ignored completely. For example, say there is a D crotchet note followed by a minim rest and finally an E crochet note. Through this approach, there is no way to represent the minim rest and so the array is just filled with the element 2 to represent the D note going to E.

Secondly, although timing and rhythm is important to the identity of a theme, as only the interval between notes are measured there is no way to distinguish between the different types of notes and timings there are. For example, a D crotchet note followed by an E minim note is no different than a D minim followed by an E crotchet note. Although rhythm analysis is important to identifying the theme and its imitation, for the sake of simplicity it is ignored. With those assumptions made, therefore the only factor in determining the imitation of the theme is purely from the intervals of its notes.

The brute-force string matching algorithm used to identify the imitation of the theme follows the general implementation of the pseudocode below.

```
procedure BruteForce (input P : Array of integer, input
T : Array of integer, output occurrence : Array of
integer)
{ P is list for Pattern, T is list for Text.
  M is length of Pattern, N is length of Text.
  occurrence is list for occurrence of Pattern in Text,
  occurrence is initialized empty. }
VARIABLES
  M, N : integer
  i, j : integer
ALGORITHM
  M ← { length of P }
  N ← { length of T }
  i traversal [0..N-M]
    j ← 0
    while (j < M) and (P[j] = T[i+j]) do
      j ← j + 1
    if (j = M) then
      { add index where Pattern is found to list of
        occurrences}
      occurrence.add(i)
```

To use the algorithm, the user must input an array of integers for the text which in this case is the note intervals of the text, and an array of integers for the pattern which is the note intervals of the theme.

A thing to keep in mind with the algorithm above is that it will take note every index in which the occurrence of the pattern exists on the text. To put it back to the perspective of the theme and its imitation, the list of occurrences will take note in which note intervals the imitation of the theme starts.

Although the theme is guaranteed to be found at the start of the fugue, for the purpose of generality the whole composition from the start including the theme is included in the text. This means that the interval at index 0 is bound to have the theme as an imitation, only if the theme pattern inputted is the same theme in the start of the composition.

For the purposes of writing this paper, the author has included a short fugal excerpt as shown in Fig. 3.3 that demonstrates the algorithm. The excerpt contains a theme based on the opening melody of a classic Indonesian children song, Naik Naik ke Puncak Gunung.

The theme of the fugue as will be inputted into the algorithm is as shown below.

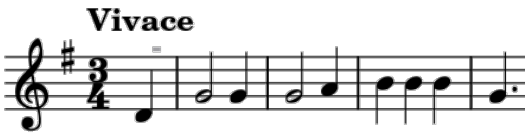


Fig. 3.4. Theme of fugal excerpt.

This theme serves as the pattern for the algorithm. Next, the theme must be converted to an array of integers which represent note intervals.

P = [5, 0, 0, 2, 2, 0, 0, -4]

The text inputted is the entire note intervals of the whole composition.

T = [5, 0, 0, 2, 2, 0, 0, -4, 7, -2, -1, -2, -2, -1, 1, 2, 5, 0, 0, 2, 2, 0, 0, -4, -2, -1, -2, -5, -4, 5, 0, 0, 2, 2, 0, 0, -4, -2, -1, -2, -2, -1, 3, 4, 5, 0, 0, 2, 2, 0, 0, -4, -1, -2, 2, 1, 2, 2, 1, 2, -2]

Applying the brute-force string algorithm will take on the steps as shown in Fig. 3.5 where the first row of the table is the text or composition, and the successive rows are the string matching of the pattern or theme to each note interval on the composition. A total of 82 character comparisons are made throughout the search. The result is that the imitation of the theme starts at interval of index 0, 16, 29, and 44. Other than the theme itself at index 0, this means that in the composition there are 3 imitations of the theme.

The corresponding imitations in the resulting interval indices are the equivalent of the highlighted musical notation as shown in Fig. 3.6. It shows that the theme begins at the start of the composition in the key of G major. It is then followed by a couple of notes before reaching the first imitation transposed to the key of D major in a higher register on the pickup to bar 7. A series of notes then follows again until the second imitation in the key of F major in a lower register on the pickup to bar 12. The third and last imitation is heard in the key of A major on the pickup to bar 17.

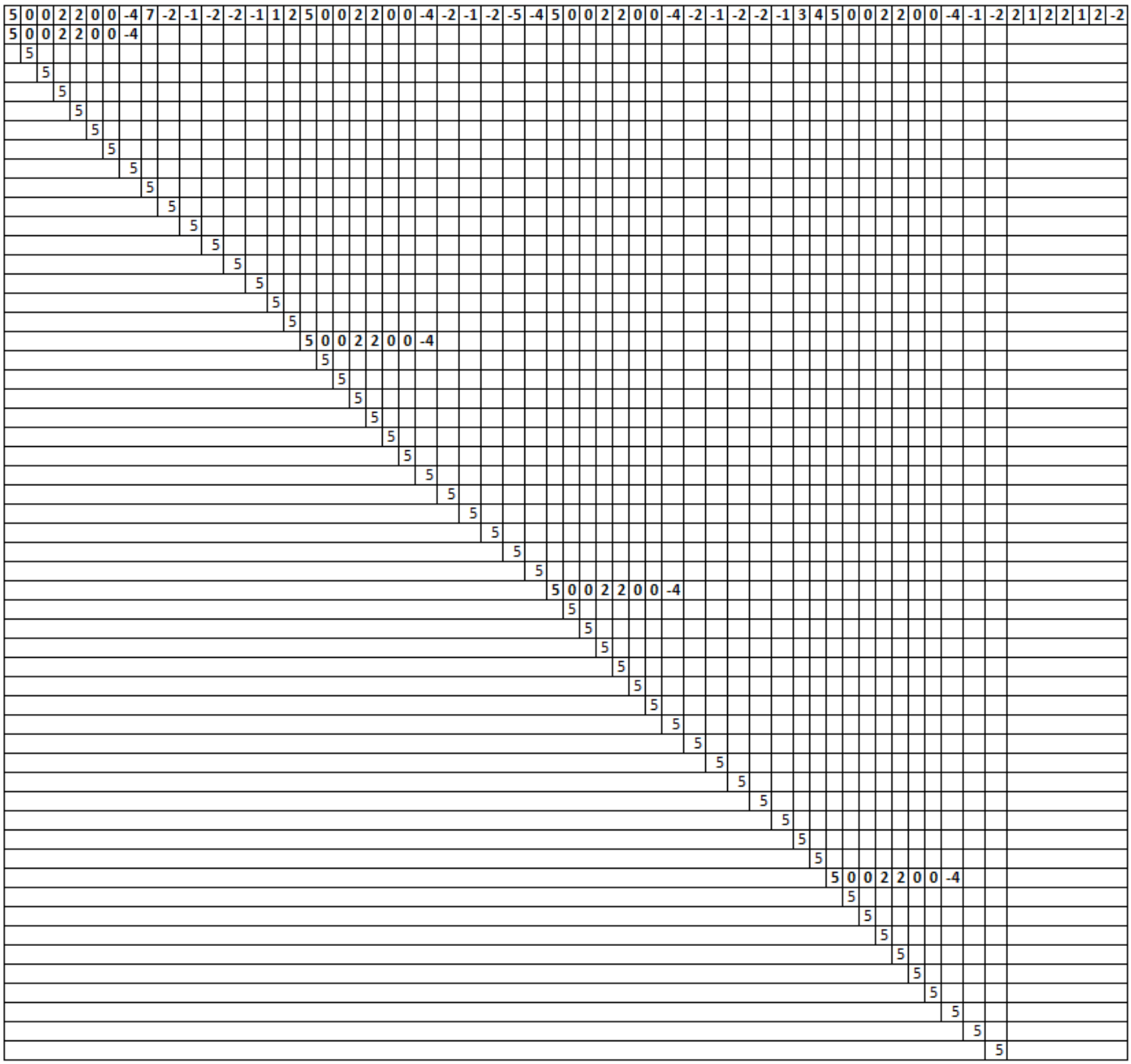


Fig. 3.5. Steps on the brute-force algorithm on the theme and composition of the fugal excerpt.

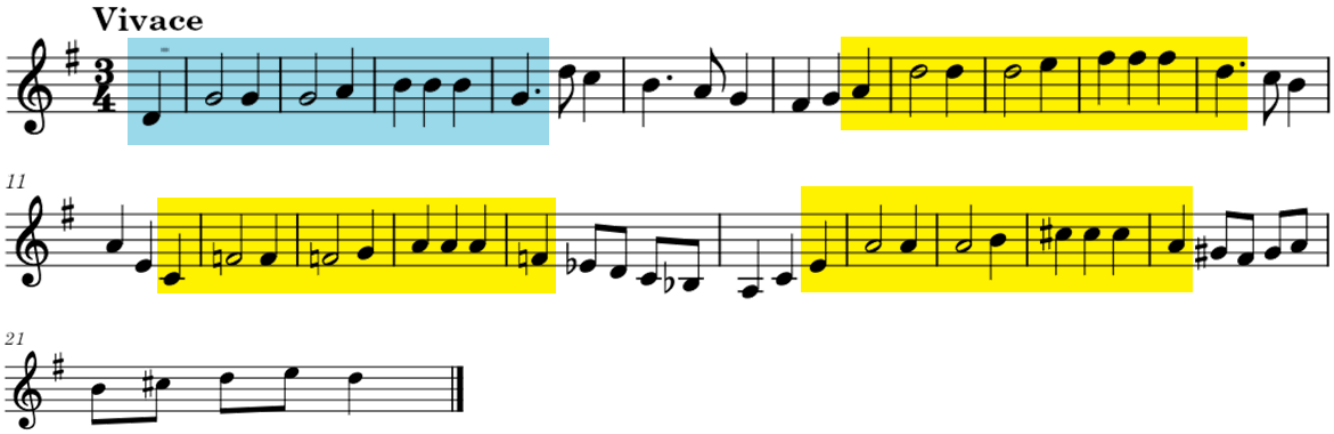


Fig. 3.6. Theme (light blue) and imitation (yellow) of the short fugal excerpt.

IV. Analysis

The fugal excerpt incidentally acts as a best-case scenario for the brute-force string matching algorithm as the theme has a beginning jump from a fifth to a root note in a higher octave represented by the first interval at index 0 with a value 5. This interval of value 5 is not seen throughout the composition or text other than the case of the theme imitations. Therefore, it only takes the first index of the pattern to check that the pattern does not match the text except where an imitation occurs. However, this best-scenario where the first note interval is distinct enough from the rest of the intervals in the composition realistically will not always happen.

It becomes apparent that as the size of the composition grows, many more string matching needs to be done. Also, the fugal excerpt does not reflect the condition of a composition as per usual in the real world. A real composition will take the form of many instruments, and each with their own clefs and voices. And so, these different clefs and voices need to be checked one by one to see if they contain the imitation to the theme.

Due to the assumptions previously made, only using the intervals of successive notes becomes the bottleneck of this approach. If a passage has the same note intervals as the theme but having different timings, the algorithm will pass this passage as an imitation of the theme. However, musically this is not correct as both the timing and rhythm makes up the identity of the theme.

Another problem occurs in how each successive imitations derive from the initial theme of the fugue. This approach only works if the intervals of the imitations are identical to the theme. However, legally an imitation does not need to have the same intervals as the theme. It only needs to take the general melodic direction of the theme. This problem is further magnified if the imitation derives from the theme in an inverted manner. This means the general direction of the imitation is upside down, where notes that move up in the theme move down in the imitation, and in turn notes that go down in the theme go up in the imitation.

V. Conclusion

The brute-force string matching algorithm successfully identified the imitation of a theme in a fugue. Overall, the approach of representing the composition, theme and imitations using their note intervals is passable for simple to intermediate fugue examples. A best-case scenario happens when the interval between the first and second note is unique enough that it does not appear frequently in the whole composition except when the imitations occur.

However, problems arise if the imitations derive too much from the theme in a manner chaotic so that the intervals change. Problems also arise as the approach used does not account for the timing and rhythm of each note which is essential to the identity of the initial theme.

The author would like to give a few recommendations to help future research in this topic. First, more efficient algorithms can be used to do the string matching such as Knuth–Morris–Pratt algorithm or the Boyer–Moore algorithm. Secondly, a better representation of the notes other than the one used in this paper could be used to also represent the time values of each note. Lastly, rather than using exact string-matching algorithms, approximate string-matching algorithms such as Levenshtein distance could be used instead to solve the problem of similar yet unidentical intervals in the imitation to the theme.

Video Link at Youtube

https://youtu.be/GxJuw6K6EwQ

Acknowledgment

The author would like to bestow the highest gratitude to Dr. Nur Ulfa Maulidevi, S.T., M.Sc. as the lecturer of IF2211 Algorithm Strategies along with other Algorithm Strategies lecturers for having taught all their knowledge as well as giving the author the opportunity to write this paper. The author would also like warmly thank friends and family that has played an integral role in helping the author in any way to finish this paper.

REFERENCES

[1] Munir, Rinaldi. 2021. *Pencocokan String (String/Pattern Matching).* [online] Available at: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf [Accessed 20 May 2022]

[2] Cormen, T., 2009. *Introduction to algorithms*. Cambridge, Mass.: MIT Press.

[3] Strayer, H.R., 2013. *From neumes to notes: The evolution of music notation.*

[4] *Mann, A., 1987. The study of fugue. Courier Corporation.*

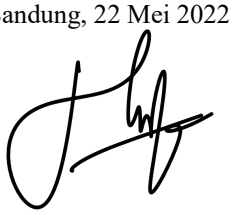
[5] Beethoven, L.V., 1824. *Symphony No. 9 in D minor*, Op. 125.

[6] Bach, J.S., 1722. *Prelude and Fugue in C major*, BWV 846. Das wohltemperierte Klavier I.

DECLARATION

I hereby declare that this paper is of my own writing, nor is it an adaptation or translation of another paper, nor a result of plagiarism.

Bandung, 22 Mei 2022



Ahmad Alfani Handoyo
13520023