Asian Journal of Information Technology 4 (10): 890-894


# An Approach to Sort Unicode Bengali Text Using Ancillary Maps

Shah Md. Emrul Islam and Muhammad Masroor Ali
Institute of Information and Communication Technology,
Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh

**Abstract:** This paper presents an approach for sorting Unicode represented Bengali text. Unicode provides a unique number for internal representation inside the computer for every character of almost every language of the world, irrespective of the platform and software. The emergence of the Unicode Standard and the availability of tools supporting it are among the most significant recent global software technology trends. Our concern is the character order in Unicode for Bengali is different from the sorting order suggested by the governing authority. The presence of modifiers in Bengali has made the problem different from conventional lexicographical sorting. The objective of this study is to adapt the suggested sort order for Bengali text, when Unicode represents it. The method is open for future modification so that rearranging the sort order does not require the algorithm to be changed. The method is based on the use of an ancillary table that specifies the desired sorting order by a Unicode to dictionary significance value (weight) for the Unicode characters. By the use of relative positional dual indexes for the characters in the input text the weights are aggregated to a single value for each Unicode enabled word Then a simple sorting of these values will sort the text in the desired sorting order.

**Key words:** Bengali word sorting, bengali unicode sorting

## INTRODUCTION

Bengali [1] is a very rich language and approximately 10% of world's populations speak in Bengali. Bangla Academy [2] is the organization looking after the standardization, development and publicity of Bengali. On the other hand, Unicode [3,4] is the recent technology trend and now almost all the languages of the world are using Unicode for their computer representation. However, the problem is that Unicode sort order does not follow the sort order of Bangla Academy as well as the order that Bengali people use. The afore dissimilarity, use of compound characters and modifiers in Bengali has created a problem and restricted the straightforward computerization such as internal representation, sorting searching, etc. of Bengali. present study, we have tried to analyze an idea about the implementation and dual indexing for Unicode Bengali text so that we can change the sort order for Bengali without causing the Unicode standard to be changed. If we can adapt Bengali sort order with Unicode, then we will be able to adapt Bengali sort order for all the products using Unicode Bengali text.

**Difficulties of sorting in bengali:** If compare Bengali with English, it can say that English words are composed of individual alphabets and sorting of English word is moderately simple. To sort English words we proceed the comparison from the first letters of both the words till the end of the words comparing character by character pair e.g. (Fig. 1). Based on the first contradictory pair of characters, a comparison is made. There is no modifier or compound character forms for English alphabet. Therefore, it is feasible to compare English word pair to pair.

In case of Bengali the scenario is pretty different. Sorting of Bengali word is not possible using such a regular algorithm[5-8]. In Bengali words, vowel and consonant modifiers are placed before and after, above of any character. Moreover, there are frequent uses of compound characters. The actual Bengali sorting is based on the base letter value. In logical Bengali sorting, a modifier comes after the base letter and the comparison is made according to the logic that is apart from the traditional typing style. In Bengali, a base letter that is not modified by a vowel or consonant modifier has a hidden modifier in the sequence after that. That basic form of that modifier is A (A). It comes after any base letter. For example, K (Ka) is composed of K (Ka) and A (A) and i (Ra) is composed of i (Ra) and A (A). Therefore, while doing Bengali sorting this important property of Bengali alphabet has to be considered.

**Corresponding Author:** Shah Md. Emrul Islam, Institute of Information and Communication Technology, Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh
Tel: 880-189-528859, 880-189-263263

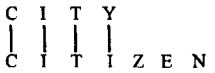


Fig. 1:English word: Pair to pair sorting

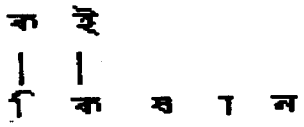

Fig. 2: Bengali word: Pair to pair sorting

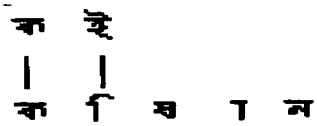

Fig. 3: Bengali Word: Pair to Pair Sorting after the Placement of Modifiers into its Right PositionAPPENDIX

## MATERIAL AND METHOD

**Character formats in Bengali:** If we do pair to pair sorting for Bengali, actual order will not come in result, e.g: Figure 2 so we need to place the vowel and consonant modifiers into its right place before sorting it. e.g Figure 3. Therefore, we need to apply proper sorting placement before sorting the Bengali text.

If a left biased modifier comes before any base letter in the sorting order, we cannot simply place it after the base letter while doing the actual sorting. We need to place it after the character following it. The letter after the left biased modifier can be a one, two or three level compound character. As Unicode does not support a single Unicode value for a compound character, we need to consider how many characters are following the left biased modifier

Like most of the languages, Bengali alphabet is divided into vowels and consonants. Like many South Asian languages, all of the vowels in Bengali have two forms. The selection of forms depends on the purpose of using a vowel in a word. If a vowel sounds independently, the principal form of vowel is used and if a vowel guides the sounds of a consonant or a group of a consonant, a different non-principal form of vowel is used. Let the non-principal form of vowels be called the half vowels. Table I. Among the half vowels are written on the left and some are on the right and some are on the bottom of the associate consonants. As the right placed half vowels, bottom placed half-vowels comes after the base letter in the type sequence. There are few half vowels, which are written in two parts in the traditional Bengali typing system, but according to Unicode rule, a vowel modifier should not be distinguished into parts. These kinds of

Table 1: Principal and modified forms of vowel

| Principal form of vowel | Non-principal form of vowel |
|---|---|
| Av | v |
| B | w |
| C | x |
| D | Y |
| E | ~ |
| F | „ |
| G | ‡ |
| H | % |
| I | ‡ v |
| J | ‡ Š |

Table 2: Principal and modified forms of consonant

| Principal form of consonant | Non-principal form of Consonant |
|---|---|
| b | Œœ |
| Y | § |
| R | ¨ |
| i | ª |
| j | ¬ |
| e | ¡ |

Table 3: Compound characters and their level

| Compound Character | No of embedded characters | Decomposed form | Example |
|---|---|---|---|
| ß | 2 | c + Z | Jyß |
| ¾¡ | 3 | R+R+e | D¾¡j |
| §¡¨ | 4 | b+Z+i+h | ¨^vZ§¡¨ |

biasing of vowels have made Bengali the complex one among the south Asian languages.

Like many Asian languages, here also two or more consonants might form a group, a compound consonant. A compound consonant is not always just the juxtaposition of the group of consonants in their principal forms. By introducing one or two different forms of some consonants besides the principal form, it is possible to display a number of compound consonants by concatenation. Let these non-principal forms of consonants be called half consonants (Table 2). But this technique of using half consonants will not work to display a few compound consonants that have no resemblance with the group of consonants they represent. In such case a distinct letter has to be used which also demands distinct position in alphabet list. Thus occupy a large numbers of distinct character values to represent all of them. Bengali compound characters can also have some different levels of comparison. A compound character can have two, three and four characters embedded into it. (Table 3)

**Sorting Method:** The sorting method is supposed to sort a list of Unicode Bengali words using the ancillary table where the proper weight of a Bengali Unicode character and type of the character is predefined. Then the algorithm works according to the predefined values. An important property of the interface is it shifts the modifiers

Table 4: Ancillary table structure

| Unicode | Sort weight | Remarks |
|---|---|---|
| 0985 | 01 | |
| 0986 | 02 | |
| 09BE | 03 | RM |
| 0987 | 04 | |
| 09BF | 05 | LM |
| . | . | |
| . | . | |
| 09B6 | 56 | BL |
| 09B7 | 57 | BL |
| . | . | |
| . | . | |
| 09FA | 89 | |
| 09BD | 90 | |

that are left biased or comes before a base letter to right of the base letter. After the shifting of the left placed modifiers, it determines value for each of the words in list. The sort interface picks the sort weight from an ancillary table and determinates a number for each of the words of the word list. Then it simply compares words according to the number determined.

The ancillary table is a simple ASCII text file that contains three columns. (Table 4). This table being independent of the main program can be easily modified. This table is thus can be adapted to any change in sort order. The first column contains the Unicode value for a character. The second column contains the sort weight value suggested by Bangla Academy dictionary. The third column contains remarks that describe whether it is a base letter or left modifier or right modifier or a joint character.

The algorithm distinguishes each Unicode value into four categories. (i) Base letter, (ii) left placed Modifier, (iii) Right placed modifiers and (iv) Joint characters. The algorithm groups a word into several pieces leaded by a base letter. As there are ninety characters listed in the Unicode-sort order value list, each entity requires two-digit value to represent its sort order value.

Therefore, a group contains a four-digit value (two digit for base letter and two digit for modifier) to represent the group value. As the shift_modifier function shifts a modifier to the right of the base letter, sort value of modifier comes after the sort value of base letter. If there is no modifier following the base letter, two zeros are inserted to fulfill the four-digit value of a group. To store the value of a word, a single real number is used and except the first letter, the values come after point.

So, according to the length of a word, the sort value after decimal point of the word becomes longer in size. For example, the word Kvb‡Kv(Kanko) (099509BE09A8099509CB) gets the value25.0346002519

0995 - 25 (Base Letter)
09BE - 03 (Modifier)
09A8 - 46 (Base Letter)
- 00 (Base letter not followed by any modifier)
0995 - 25 (Base Letter)
09CB - 19 (Modifier)

And the word Kvb‡KvUv (Kankota)(099509BE09A8099509CB099F09BE) gets the value 25.03460025193503

0995 - 25 (Base Letter)
09BE - 03 (Modifier)
09A8 - 46 (Base Letter)
- 00 (Base letter not followed by any modifier)
0995 - 25 (Base Letter)
09CB - 19 (Modifier)
099F - 35 (Base Letter)
09BE - 03 (Modifier)

The algorithm uses the decimal number system for determination of the value of a Bengali Word. While sorting, we cannot compare two Bengali words directly. We cannot even compare the Unicode value of two characters, as the sorting order of the Unicode for Bengali is not ordered. So for the comparison of two Bengali words, the algorithm works as following. The pseudo code for sorting is presented in Appendix A

Lets we have two Bengali Words Kg©Pvix (Karmochari) (099509B009CD09AE099A09BE09B009C0) Kvh©Kifv‡e(Karjokorbhabe) (099509BE09B009CD09AF099509B009AD09BE09C709AC)

Analysing the value of Kg©Pvix(Karmochari) (099509B009CD09AE099A09BE09B009C0) we get the following result.

| | |
|---|---|
| K | 0995 |
| I | 09B0 |
| & | 09CD |
| g | 09AE |
| P | 099A |
| v | 09BE |
| I | 09B0 |
| x | 09C0 |

Applying the procedure for shifting the modifier, we get the following order:

| | |
|---|---|
| K | 0995 |
| I | 09B0 |
| & | 09CD |
| g | 09AE |
| P | 099A |
| v | 09BE |
| I | 09B0 |
| x | 09C0 |

According to the ancillary table, we get the following values:

| | | |
|---|---|---|
| K | 0995 | 25 |
| I | 09B0 | 54 |
| & | 09CD | 63 |
| g | 09AE | 51 |
| P | 099A | 30 |
| v | 09BE | 03 |
| I | 09B0 | 54 |
| x | 09C0 | 07 |

Considering the ancillary table value and the type of the character, we get the following values:

| | | |
|---|---|---|
| K+ 0 | 0995+0000 | 2500 |
| I+ & | 09B0+09CD | 5463 |
| g+0 | 09AE+0000 | 5100 |
| P+ v | 099A+09BE | 3003 |
| I+ x | 09B0+09C0 | 5407 |

So we get the value 25.005463510030035407 for the word Kg©Pvix (Karmochari)
(099509B009CD09AE099A09BE09B009C0)

On the other hand, analyzing the value of Kvh©Kifv‡e (Karjokorbhabe)
(099509BE09B009CD09AF099509B009AD09BE09C709AC) we get the following result

| | |
|---|---|
| K | 0995 |
| v | 09BE |
| I | 09BO |
| & | 09CD |
| h | 09AF |
| K | 0995 |
| I | 09B0 |
| f | 09AD |
| v | 09BE |
| ‡ | 09C7 |
| `e | 09AC |

Applying the procedure for shifting the modifier, we get the following order:

| | |
|---|---|
| K | 0995 |
| v | 09BE |
| I | 09BO |
| & | 09CD |
| h | 09AF |
| K | 0995 |
| I | 09B0 |
| f | 09AD |
| v | 09BE |
| e | 09AC |
| ‡ | 09C7 |

According to the ancillary table, we get the following values:

| | | |
|---|---|---|
| K | 0995 | 25 |
| v | 09BE | 03 |
| I | 09B0 | 54 |
| & | 09CD | 63 |
| h | 09AF | 52 |
| K | 0995 | 25 |
| I | 09B0 | 54 |
| f | 09AD | 50 |
| v | 09BE | 03 |
| e | 09AC | 49 |
| ‡ | 09C7 | 15 |

Considering the ancillary table value and the type of the character, we get the following values:

| | | |
|---|---|---|
| K+v | 0995+09BE | 2503 |
| I+& | 09B0+09CD | 5463 |
| h+0 | 09AF+0000 | 5200 |
| K+0 | 0995+0000 | 2500 |
| I+0 | 09B0+0000 | 5400 |
| f+v | 09AD+09BE | 5003 |
| e+‡ | 09AC+09C7 | 4915 |

So we get the value 25.035463520025005400500349l5 for the word Kvh©Kifv‡e (Karjokorbhabe)
(099509BE09B009CD09AF099509B009AD09BE09C709AC)

So, after the determination of a single value for a Bengali word, we can just simply compare them according to their extracted value.

## UNIQUENESS OF THE DETERMINED VALUE

Let us say that A and B are two Unicode words and their generated sort values are X and Y respectively. In addition, we assume that A<>B. We want to prove that, the generated sort values are distinct.

Now from the first fraction (base letter pair with modifier and blank modifier) of A and B let we get two values x1 and y1. As the ancillary table returns a unique value for an alphabet, there is no possibility that these two values are same unless the first fractions of the two words are same. From the second fraction of A and B let we get two values $x_2$ and $y_2$. The ancillary table returns a unique value for an alphabet and there is no possibility that these two values are same unless the second fractions of the two words are same.

Let n is the last number of fraction of the shortest word. From this nth fraction of A and B let we get two values $x_n$ and $y_n$. These two values will not be the same one if the nth fraction of A and B are same. After that the shortest word will not produce any value as n is the end fraction of that and for the other word we will get values that will cause catenation to the end of the nth fraction value of that word.

Now, from our analysis we can get that

$A_1 <> B_1$ unless they are same
$A_2 <> B_2$ unless they are same
. .
. .
$A_n <> B_n$ unless they are same
$A_{n+1}$ If A is the longest word
$A_{n+2}$ If A is the longest word
.
.
$A_t$ If A is the longest word and t is the last number of fraction of A

$X=x_1$ and $x_2$ and $x_3$..... and $x_t$ (t is the last number of fraction of A)

$Y=y_1$ and $y_2$ and $y_3$..... and $y_n$ (n is the last number of fraction of B)

Now, X and Y can only be the same if and only if the lengths of the words are same and every value of the pair fractions are same. But it is only possible when these two words are same because there is only one value mapped for a Unicode symbol in the ancillary table. Thus, we get that every generated value is unique.

## DISCUSSION

The algorithm is not based upon any proprietary solution but free to change the order of the sorting. We cannot only sort our words according to Bangla Academy but it can be changed if any future amendment is made without causing the algorithm to be changed. Moreover, there is future probation for the Unicode extension for Bengali. As we are shifting the modifiers before sending a word to the value detection function, we are getting the right sorting result. Future suggestion for it is to consider the three-digit value for a sort weight value in the ancillary table and to consider alphabet frequency for the arrangement of Bengali alphabets in the ancillary table

Appendix A

Sorting algorithm

```
1     Start
2     i = 0 ;
3     j = 0 ; passweight = 1; weight=0
4     Read the Unicode Value of a word Wi from the list
5     Shift the left modifiers of Wi to right of the base letter
6     Select a single letter Li from Wi
7     If Type (Lj) < > Modifier then
8     If (Lj-1) < > Modifier then
9     passweight = passweight / 100
10    Else
11    weight = weight + Value (Lj) * passweight
12    passweight = passweight / 100
13    End if
14    End if
15    j = j + 1
16    If Lj <> NULL then
17    Goto 4
18    Else
19    i = i + 1
20    End If
21    If all words of the list are not covered then
22    Goto 3
23    Else
24    Sort the list using the Generated Values
25    End If
26    End
```

## REFERENCES


1. Miscellaneous topicsonBengalihttp://www.world language.com/ Languages/Bengali.htm.
2. Hoque, M.E., S. Lahidi and S. Sarker, 1992. e¨envwiK evsjv Awfavb (Baboharik BanglaObhidhan), Bangla Academy, Dhaka.
3. Unicode Home Page http://www.unicode.org.
4. Aliprand, J., J. Allen, J. Becker, M. Davis and M. Everson, 2002. The Unicode Standard Version 4.0, Addison-Wesley Pub Co, Boston.
5. Horowitz, E., S. Sahni and S. Rajasekaran, 1999. Fundamentals of computer algorithms. Galgotia Publications Pvt. Ltd, New Delhi.
6. Knuth Donald, E., 1997. The Art of Computer Programming, Addison-Wesley Pub Co, Boston.
7. Rahman, Md. Shahidur and Iqbal Md. Zafar, 1998. Bangla sorting algorithm: A linguistic approach. Proceedings of International Conference on Computer and Information Technology, Dhaka, pp. 204-208.
8. Zibran, M.F., A. Tanvir, R. Shammi and M. A. Sattar,. 2002. Computer representation of Bangla characters and sorting of Bangla words Proceedings of International Conference on Computer and Information Technology, Dhaka, pp. 191–195.