TreeSort

Kamil Landa

http://github.com/KamilLanda/TreeSort

in **LibreOffice** http://libreoffice.org 12/2022

TreeSort is an algorithm for sorting, removing a duplicities and finding an uniques – all at once. It is possible to extend it also for fast searching and also for searching that ignores the lower/upper-cases or diacritic.

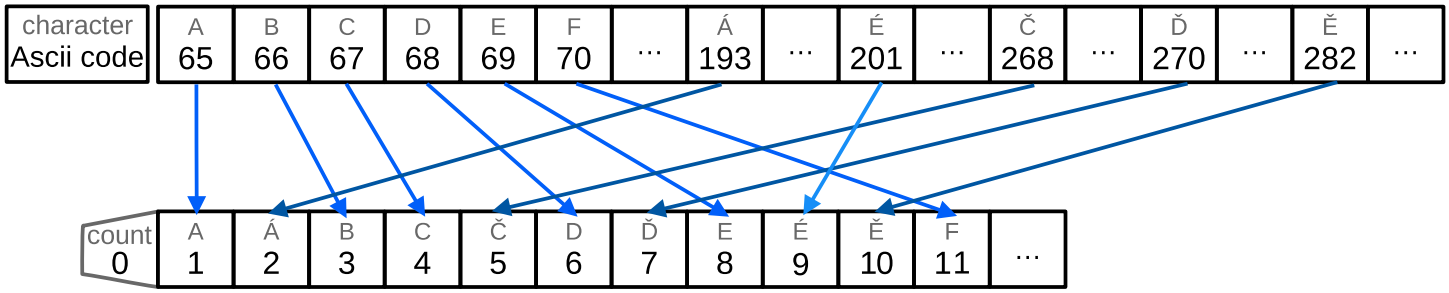
There isn't any comparison of strings nor swapping ones. The algorithm detects used characters in 1st pass, then it makes the statistics about characters in 2nd pass (this statistics has form of tree), and one reading of this tree returns the result.

The biggest problem could be the size of used RAM (primarily if input data will full of uniques and their count will in millions or more), but I didn't research how to outguess or compute it beforehand. The Complex variant could be good for scrimping the RAM, but the examples in LibreOffice Basic are finished for Simple and Extended variant, and Simple variant in Python.

Simple variant

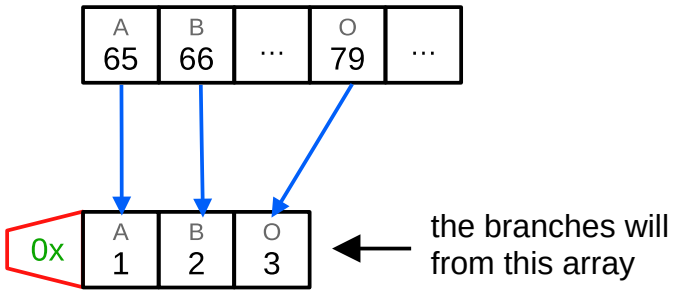
Simple variant doesn't mean inferior, but it is simplest to programm and simultaneously the fastest variant. There are only one-letter characters, there isn't possibility for example to sort according to Czech two-letter **ch**.

Every character has own Ascii code and it need next array with sorted characters. But these characters are only as indexes in this array.

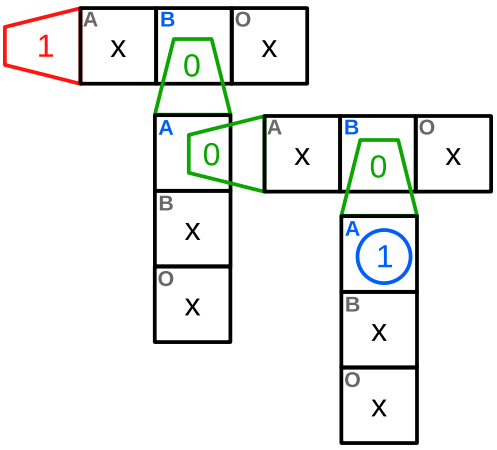


The branches in tree are made from these indexed arrays – according to the order of letters in words (characters in items). There will be new sub-branch for every next letter (instead last letter that has only count for word).

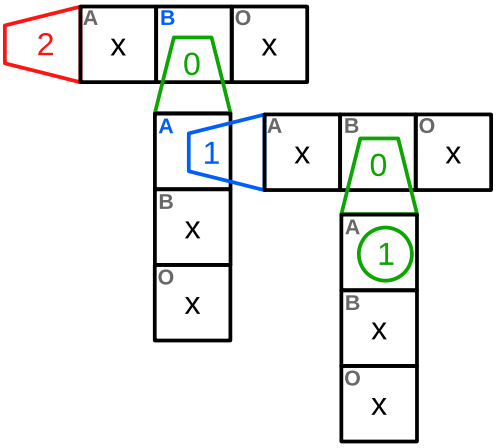
For example the words: BABA, BA, OBA, BAOBAB, OB, BOB, BOA. There will be the array with four items, zero index is for count of word, next indexes are for letters. So it detect the index for every Acii code and then it builds a branches from the array of indexes.



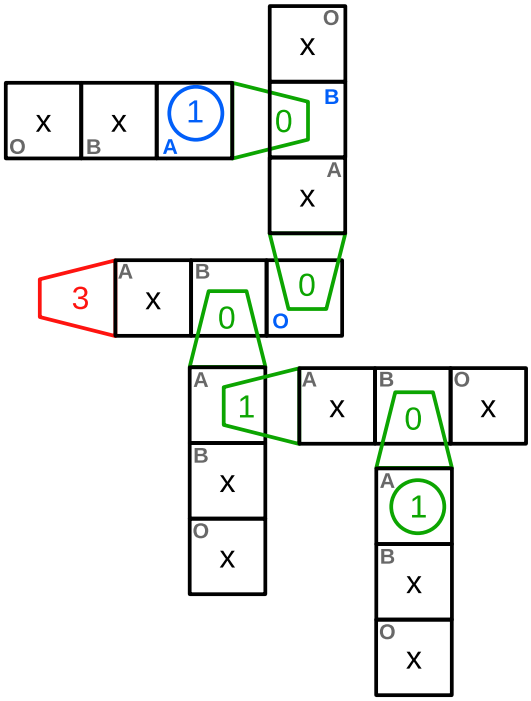
The tree for 1st word **BABA**. The zero index that means total count of all word is red, every next branch is for next letter. There is a blue count for current word at the end of word. **x** means the empty cell in array.



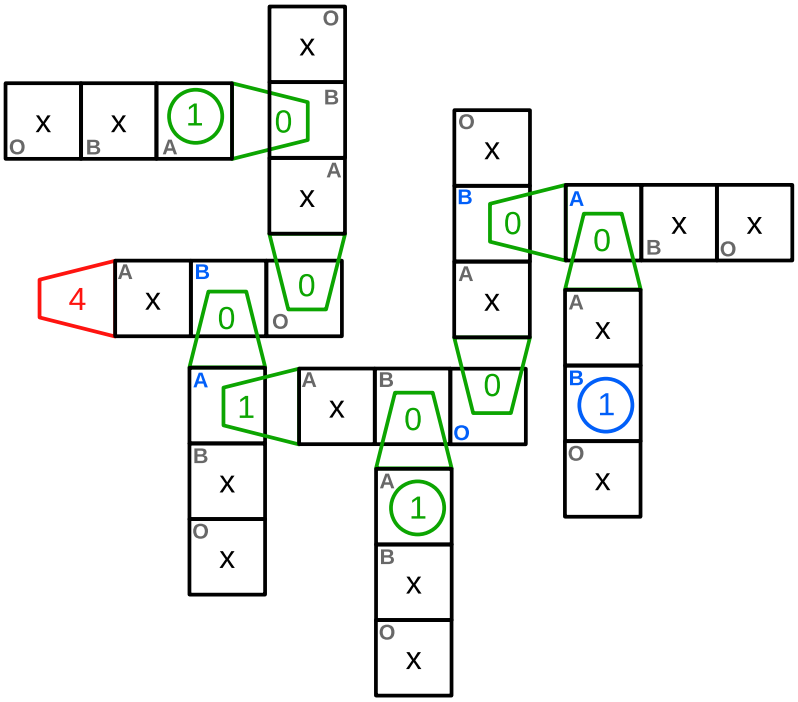
Next word is **BA**. Zero index in sub-branchs means the end of word and count for this word. There is other blue **1** for it.



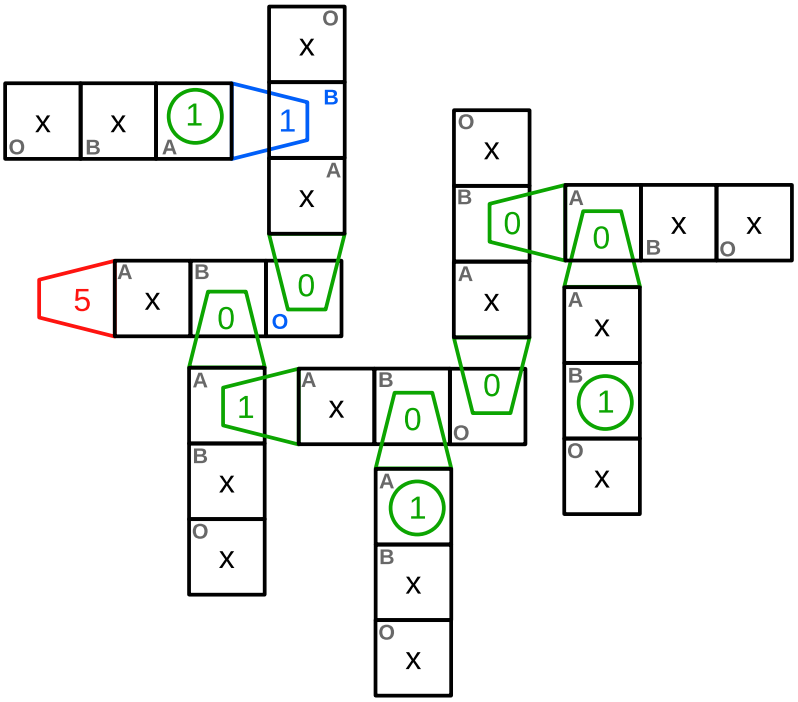
For **OBA**.



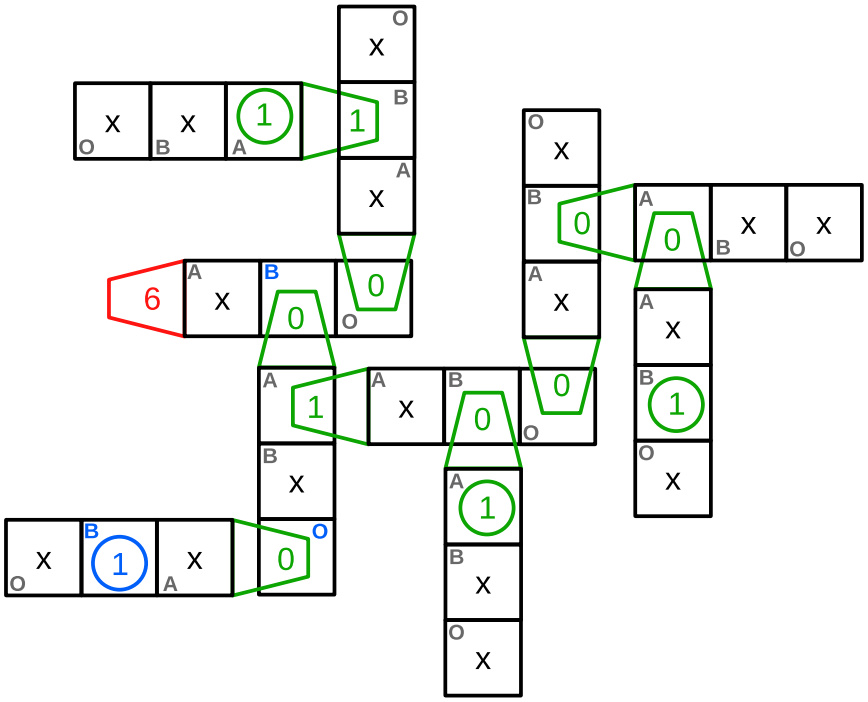
And **BAOBAB**.



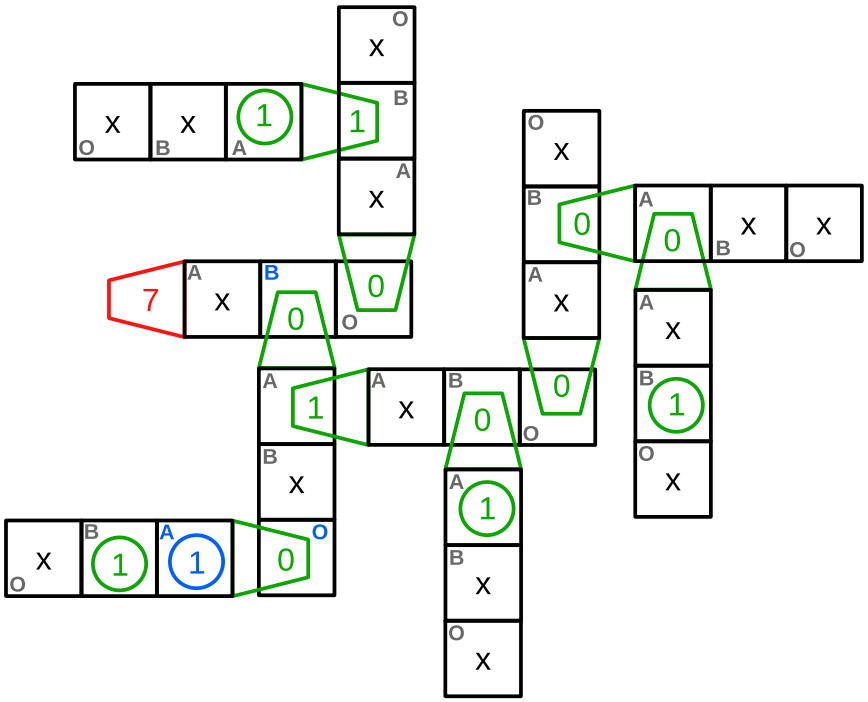
Of course **OB**.



Penultimate **BOB**.

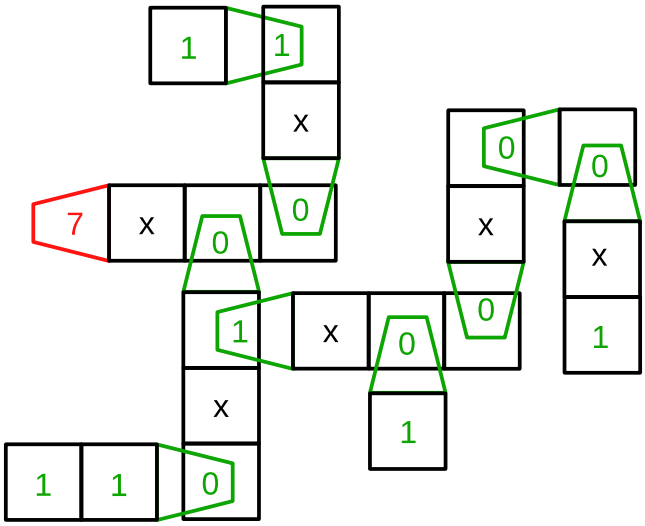


And last **BOA**.



Systematic reading of tree (I used a recursion) will give the result → sorted words with their counts. The uniques are only once; the list without counts is the list without duplicities; and for sorted list is needful to write out every word accordind to its count.

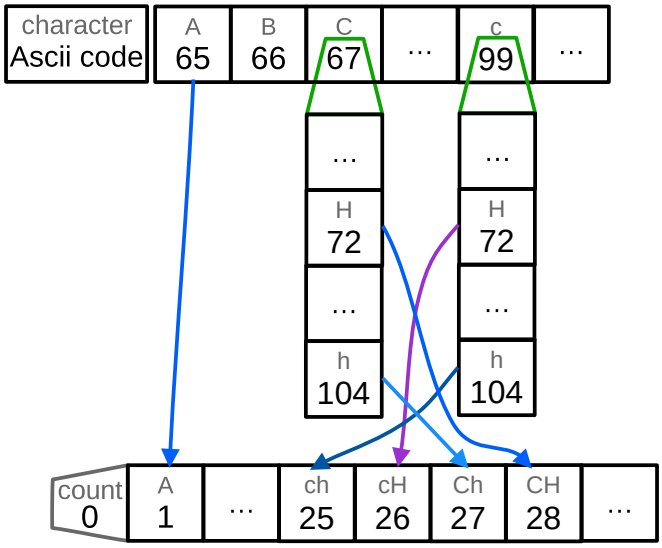
It is possible to truncate some branches, so some RAM will be scrimped. The tree looks like this – but it is without the helped letters because there is everything only in according to indexes.



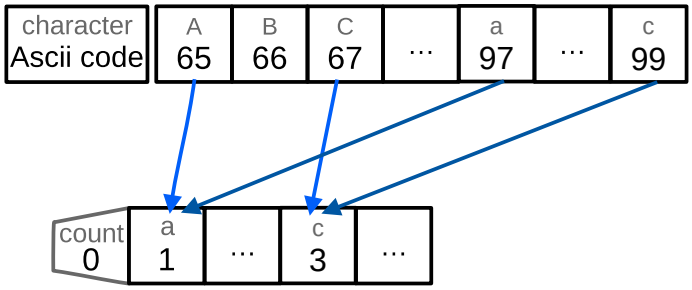
Extended variant

There is added the possibility to sort according to two-letters characters like Czech **ch** in the example. Also with the detection of unknown characters that will be added to the end and sorted according to their Ascii codes. There is also **Hewn sorting** with ignoring lower/upper-cases – but it means the result will be only in lowercases or only in uppercases.

The tree is built by the same way, according to indexes. There is only control during reading the characters if current character is part of two-letter one or not. There is used the function **IsArray()** for it. So if the variable for current Ascii code isn't the number but it is the array, it means there is the subarray with Ascii codes for 2nd letter and afterward the index is taken.

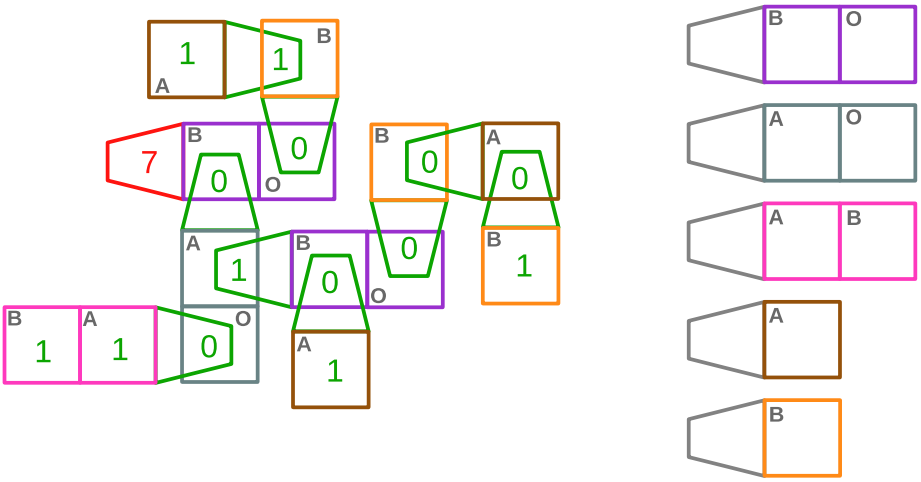


The Hewn sorting with the ignoring of lower/upper-cases will index the Ascii codes for lowercases and also the uppercases to same index. Then the result will be only in lowercases or only in uppercases.

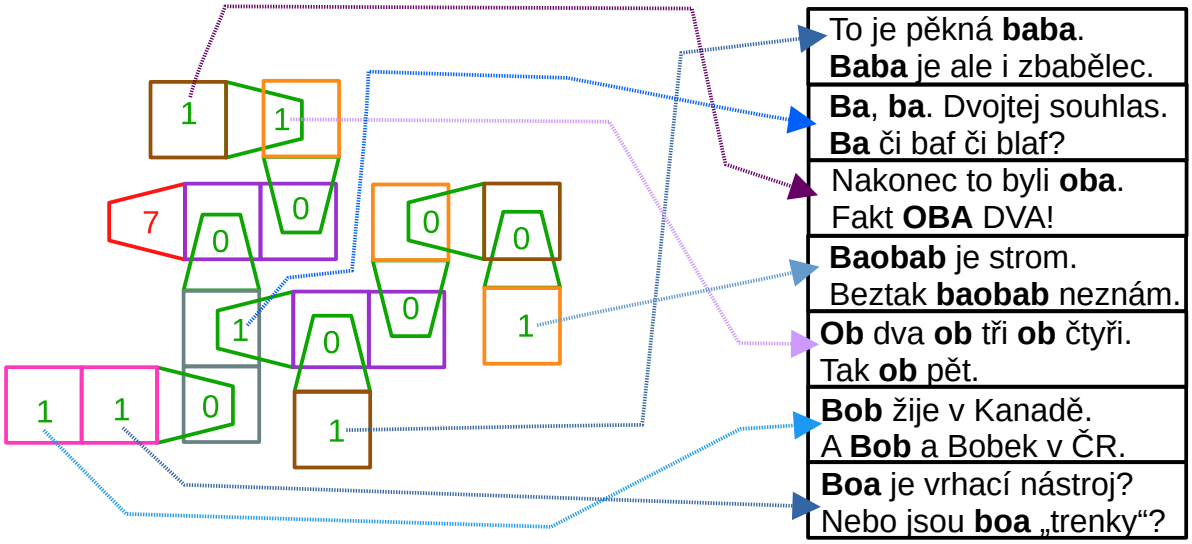


Complex variant

It is possible to use different branches. It can look like this – different branches are colored by different colors. It need add the information to distinguish the branches, but there aren't any superfluous empty cells.



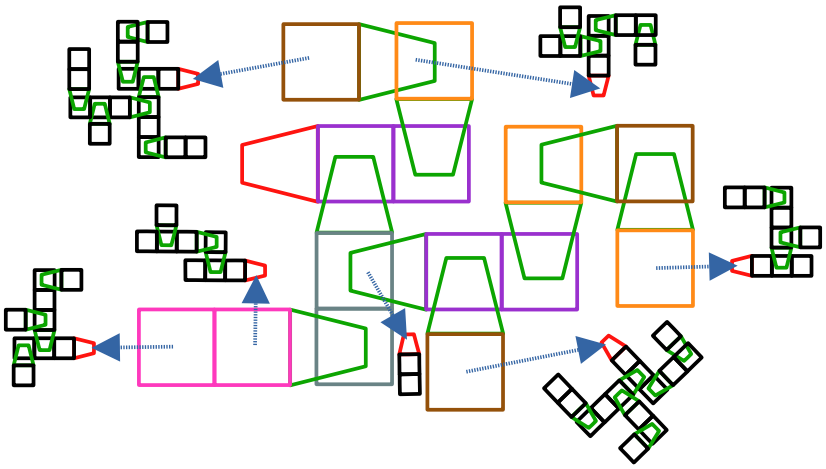
The complex variant envisages it is also possible to add the links to primal data. It enables fast searching of primal data. The advantage is for searching of phrases → it is possible to build the tree for all words in all phrases, but final indexes will be linked to original phrases. So the searching of some word will show all phrases including the word.



It shows it is possible to build the tree from Hewn variant and use only lowercases, but with the links to the primal data where are also the uppercases. But it is also posisble to use it for searching without diacritic. For example the index for letter **a** is also for letters **a A á Á**, or the **e** is indexed from **e E é É ě Ě**. Or there can be the word **rad** in the tree, but the links will be to the phrases with words **rad, rad’, rád, řád, řad’**. The linking takes more RAM, but can be very good for repeated searching :-).

Multitree

The final links aren't to the primal data, but to the other trees that are builded from used phrases. Main tree can be built only in lowercases and without diacritic, but subtrees can be built from original letters. So it enables fast searching without diacritic in data with diacritic. For example you will try to find the word **řad’**, but it will be searched as **rad** in main tree and as **řad’** in sub-tree.



Woodland?

Why to have only one tree if it is posisble to add all parts of all words to the tree? For example create the links to word HELLO from all parts: HELL, HEL, HE, H, ELLO, ELL, EL, E, LLO, LL, L, LO, O. A also make all combinations for all lower-cases and uppercases and also for all diacritic variants. It can take many RAM, but be good for searching according to parts of words. And what to create a little forest for it :-)?

Testing (speed and RAM)

LibreOffice Basic

I made a few tests with some TXT files for comparison in LibreBasic for: TreeSort (Extended variant), HeapSort from ScriptForge library, and QuickSort from older version of this library (LibO primitives). And TreeSort won :-). Simple variant is faster, but I kept it simple for easier studying how it creates the tree. And I didn't want to ignore Czech ***ch**-es* in sorting :-).

Count of words	TreeSort	QuickSort	HeapSort
2,014	0*m*:01*s*	0:00	0:08
6,897	0:04	0:01	0:30
58,613	0:27	0:46	5:33
121,494	0:53	2:44	12:15
340,094	2:40	*	**
1,015,803	7:54	**	**

** maybe about 4 minutes it showed error message like: Not enough memory in stack.*

*** Greendeal is greendeal; and I was kind to fan in laptop :-).*

About the RAM. I tested the Extended variant without the hewned branches and the tree for biggest tested file with about 1 million words took about 300MB RAM. The tree for Czech dictionary of synonyms that has about 260,000 nearly unique words took about 2.2GB RAM.

Python in editor Pyzo

I tried the Simple variant. Tested on generated array with 6-letters words from A-Z. The most combinations was 20^6 = 64 millions items. I used the algorithm Timsort for comparison, where the input data was generated from Z to A to give the work to the algorithm :-). There isn't any advantage or disadvantage that input data are sorted or not for TreeSort, it doesn't bring easiment during building a tree.

Count of words	TreeSort	Timsort
1,000,000	5*s*	8.2
2,985,984	15	25.56
11,390,625	62.11	116.43
64,000,000	333.27	694.27

The biggest sorted array had about 4.6 GB, it took about 8.8 GB with Timsort but about 15GB RAM with TreeSort, because the creating of new output array and recursion took it.

Bugs in LibreOffice for LibreBasic

I discovered two bugs during testing *(12/2022 LibreOffice 7.4.3.2 Win10x64)*.

Bug https://bugs.documentfoundation.org/show_bug.cgi?id=152613

During the recursion for reading the tree, LibreOffice doesn't deallocate the RAM. I tested about 200k words from Czech dictionary for synonyms from LibreOffice and after few minutes it took 24GB RAM and Windows showed picturesque blue death (there wans't only a text but also some picture) :-(.

Bug https://bugs.documentfoundation.org/show_bug.cgi?id=152466

There isn't refresh of array of array in some case in Watch in Basic Editor IDE.

Initial idea

My English isn't so good, and I like use the small offline cs/en dictionary **ACS Slovník**. But when I search the word, it shows more words from other language many times. For example for Czech word ***příklad*** it shows: ***exemplar***, ***instance***, ***paragon***, ***example***. And how to choose competent word? So I rewrite the showed words one by one and see the other retranslated variants:

exemplar → *příklad*
instance → *žádost, případ, instance, výzva, příklad, prosba, situace*
paragon → *ideál, příklad, vzor*
example → *ukázka, vzor, příklad*

And then it is possible to choose more competent word :-).

Resumé

I programmed the Simple and Extended variant during about 10 days (12/2022) in LibreOffice Basic because I've programm in this language last years. I deem the algorithm is trivial, but the programming for two-letters wasn't so trivial for me and I contended with the indexes in arrays of arrays some days.

Penultimate day of year 2022 I created and tested the Simple variant in Python.

I had decided if to make also the Complex and Multi variant and then to publish it, but I decided I will share already the variants I made, even though it isn't finished to final form, although there is a place for some optimalization in made examples.

LibreBasic isn't quiet felicitous programming language for it, and of course Python also isn't miracle for it. The creating a tree and reading one could be much faster in some compiled language. But it isn't in my knowledge to program it for example in C++ or Rust.

The question ergo is, if the creation of tree and reading one could be faster than miscellaneous comparison of strings with swapping.

So the imagination about sharing with the others was more pleasant – maybe other programmers will play with it, cope it, improve it, control it etc. :-).

But I nevertheless hope it will be for good and it will be expanded already from this form :-).