Scientific Journal of Impact Factor(SJIF): 3.134

e-ISSN(O): 2348-4470
p-ISSN(P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 2,Issue 5, May -2015

Different Data Structures Used For Playing With Anagram

JinalPatel

Computer And Science Department,Saffrony Institute of Technology ,Linch,Mehsana

Abstract: *In this paper, we are presenting afyndamentals of data structure, explains what exactly is the anagram . Then different Data Structures used for Anagram. Comparing them with each other. Explains the data structure which stores the given dictionary data in a hash table called PRIME by usingfundamental theorem on Arithmetic to generate a key for each dictionary word, and stores the word in the hash table based on the key. Ascompared to tree-based techniques PRIME table generates anagram for the given random word in O(1) time, time to construct a PRIME tabled epends on the number of words in the dictionary. If dictionary has „n" words then the time to develop the PRIME table is O(n).*

Keywords: *Anagrams, Data Structure, Anatree, Algorithms,performance, Experimentation, fundamental theorem on Arithmetic, Hash map, Prime Table.*

I. INTRODUCTION

[1] Anagram:

1.1What is Anagram?

An **anagram** is a type of word play, the result of rearranging the letters of a word or phrase to produce a new word or phrase, using all the original letters exactly once; forexample Torchwood can be rearranged into Doctor Who. Someone who creates anagrams may be called an "anagrammatist". The original word or phrase is known as thesubject of the anagram. Anagrams are often used as a form of mnemonic device as well.

Any word or phrase that exactly reproduces the letters in another order is an anagram. However, the goal of serious or skilled anagrammatist is to produce anagrams that in some way reflect or comment on the subject. Such an anagram may be a synonym or antonym of its subject, a parody, a criticism, or praise; e.g. William Shakespeare = I am a weafish speller

Another example is "silent" which can be rearranged to "listen". The two can be used in the phrase, "Think about it, SILENT and LISTEN are spelled with the same letters". (To mean "the quieter you become, the more you can hear").

1.2 Conditions for Anagram:

The creation of anagrams assumes an alphabet, the symbols which are to be permuted. In a perfect anagram, every letter must be used, with exactly the same number of occurrences as in the anagrammed word or phrase.

II. DATA STRUCTURE

2.1What is Data Structure?

In computer science, a data structure is a particular way of organizing data in a computer so that it can be usedefficiently.

Data structures provide a means to manage large amounts of data efficiently for uses such as large databases and internet indexing services. Usually, efficient data structures are key to designing efficient algorithms. Some formal design methods and programming languages emphasize data structures, rather than algorithms, as the key organizing factor in software design. Storing and retrieving can be carried out on data stored in both main memory and in secondary memory.

2.2Data Structure for Anagrams:

There are many Data Structure used to solve Anagrams. Let us discuss some possible Data Structure used

2.2.1 Simple Data Structures :

2.2.1.1 Alphabetic Map:

A number of data structures have been proposed to solve anagrams in constant time. Two of the most commonly used data structures are the Alphabetic map and theFrequency map. The Alphabetic map maintains a hash table of all the possible words that can be in the language (this is referred to as the lexicon). For a given input sting, sort the letters in alphabetic order. This sorted string maps onto a word in the hash table. Hence finding the anagram requires sorting the letters and a looking up the word in the hash table. The sorting can be done in linear time by the counting sort and hash table look up can be done in constant time.

For example for the word ANATREE, the alphabetic map would produce a mapping of $\{AAEENRT- > \{''anatree''\}\}$.

2.2.1.2 Frequency Map:

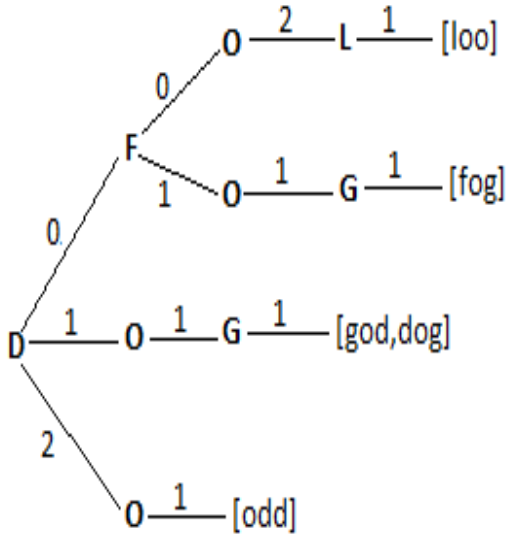
A Frequency map also stores the list of all possible words in the lexicon in a hash table. For a given input string, the frequency map maintains the frequencies (number of appearances) of all the letters and uses this count to perform a look up in the hash table. The worst case execution time is found to be linear in size of the lexicon.

2.2.2 Anatree Data Structure:

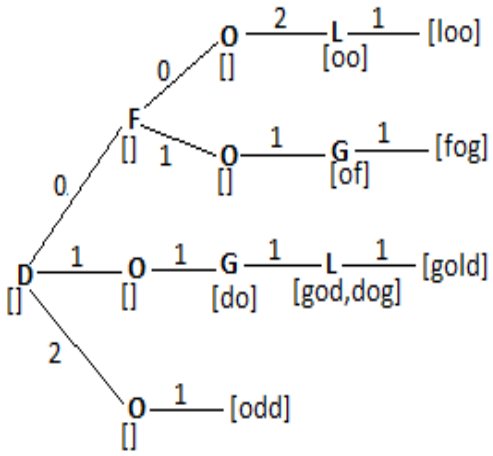
An Anatree is a data structure designed to solve anagrams. Solving an anagram is the problem of finding a word from a given list of letters. These problems are commonly encountered in word games like wordwheel, scrabble or crosswordpuzzles that we find in newspapers. The problem for the wordwheel also has the condition that the central letter appear in all the words framed with the given set. Some other conditions may be introduced regarding the frequency (number of appearances) of each of the letters in the given input string. These problems are classified as Constraint satisfaction problem in computer science literature.

An Anatree is represented as a directed tree which contains a set of words (W) encoded as strings in some alphabet. The internal vertices are labeled with some letter in the alphabet and the leaves contain words. The edges are labeled with non-negative integers. An Anatree has the property that the sum of the edge labels from the root to the leaf is the length of the word stored at the leaf. If the internal vertices are labeled as α_1, α_2 ... α_l, and the edge labels are n_1, n_2 ... n_l, then the path from the root to the leaf along these vertices and edges are a list of words that contain n_1 α_1s, n_2 α_2 s and so on. Anatrees are intended to be read only data structures with all the words available at construction time.

A Mixed Anatree is an anatree where the internal vertices also store words. A mixed anatree can have words of varying lengths, where as in a regular anatree, all words are of the same length.



SIMPLE ANATREE



ANATREEMIXED ANATREE

2.2.2.1 Construction of an Anatree :

The construction of an Anatree begins by selecting a label for the root and partitioning words based on the label chosen for the root. This process is repeated recursively for all the labels of the tree. Anatree construction is non-canonical for a given set of words, depending on the label chosen for the root, the anatree will differ accordingly. The performance of the anatree is greatly impacted by the choice of labels.

The following are some heuristics for choosing labels:

- Start labeling vertices in alphabetical order from the root. This approach reduces construction overhead.
- Start labeling vertices based on the relative frequency. A probabilistic approach is used to assign labels to vertices. If W_n^{α} is the set of words that contain $n\alpha s$, then we label the vertex with α if it maximizes the expected distance to the leaf. This approach has the most frequently appearing characters (like E) labeled at the root and the least frequently

appearing characters labeled at the leaves. The following equation is maximized $D_\alpha = \sum_n n\frac{|W_n^\alpha|}{|W|}$. This approach prevents long sequences of zero labeled edges since they do not contribute letters to the words generated by the anatree.

2.2.2.2 Finding Anagrams from Anatree :

To find a word in an anatree, start at the root, depending on the frequency of the label in the given input string, follow the edge that has that frequency till the leaf. The leaf contains the required word. For example, consider the anatree in the figure, to find the word dog, the given string may be ogd. Start at the root and follow the edge that has 1as the label. We follow this label since the given input string has $1\ d$. Traverse this edge until the leaf is encountered. That gives the required word.

2.2.2.3 Space and Time Requirements :

A lexicon that stores w words (each word can be l characters long) in an alphabet A has the following space requirements.

Data Structure	Space Requirements
Alphabetic Map	$O(wl(log\|A\|)$
Frequency Map	$O(w(\|A\|logl + llog\|A\|))$
Anatree	$O(w\|A\|(log\|A\| + l))$

The worst case execution time of an anatree is $O(|w|(l + w|A|^2))$

2.2.3 PRIME TABLE :

PRIME table is a Hashmap, each key in the hash map is the product of prime numbers that are related to each character of a given word. Construction of PRIME table.

Step 1:

Find the key value for a given word.To find key value for the given word, first all the alphabets in the given language should assigned with weights of unique prime numbers.For every word in the dictionary the key is constructed by multiplying all the weighted values of each character in the given word.

A	B	C	D	E	F	G	H	I	J	K	L	M
2	3	5	7	11	13	17	19	23	29	31	37	41
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
43	47	53	59	61	67	71	73	79	83	89	97	101

Table 1: Indextable for English Alphabet

As shown in table 1 each character in Englishalphabet is assigned with distinct prime number. Englishalphabet has 26 characters so we have taken first 26 distinctprime numbers.

For the word: god
The key construction is
getKey (god) = 17 * 47 * 7 = 5593
Since the prime weight for the letter 'g ' is 17
prime weight for the letter 'o ' is 47
prime weight for the letter 'd' is 7

Step 2:
Store the word in hash map based on the key value.Once the key generated for the given word theword is store in the Hash map based on the key value. Thereis possibility that more than one word has same key value,in that case key value is pointed to all the words that havesame key value.

Example :
god and dog has same key value 5593,
thenHashMap(5593) points to both the strings god, dog.
I.e., HashMap (5593) returns both the words.

2.2.3.1 Algorithm for Prime Table:

Terminology used in algorithm:

DictFile : File containing all the words.
readNextWord() : Reads next word in the givenfile.
getKey() : Returns the key value for thegiven string.
str1.append(str2) : Appends string str2 to thestring str1.
put(key, string) : Associates the specifiedstring with the specified key in this map.
get(key) : Returns the value to whichthe specified key is mapped in this identityhash map, or null if the map contains nomapping for this key.
alphaWeight array: Array that has prime number valuesfor each character in thealphabet of given language.
findIndexValue() : Returns the prime number weight ofgiven character.
Length() : Returns the number of characters in given string.
search() : Returns the anagrams for the given random word.
charAt() : Returns a character at particular position thestring.

HashMap<Integer Key,String Value>dictMap :

A Hash structure with fields key of type integer and value oftype string.

```
/* GenerateHashTable for given set of words*/
processDictionary(DictFile)
        {
                do
                {
                        String = readNextWord(DictFile);
                        Key = getKey(String);
                        if((value = dictMap.get(key)) != null)
                        {
                                value.append(","+string);
                                dictMap.put(key, value);
                        }
                        else
                        {
```



```
                                        dictMap.put(key, string);
                                }
                        }while(End Of File reached)
                }
/* Find the value of the alphabet from table/Array Index*/
IntgerfindIndexValue()
{
        return (value of the alphabet from thealphaWeight array);
        //value for A is 2 and H is 19
}
/* Return the key value of given string */
Integer getkey(String)
{
        mulValue = 1
        for I = 1 to length(string)
        {
                indexValue =findIndex Value (string.charAt(i));
                mulValue = mulValue * indexValue
        }
        returnmulValue
        }
String search (findAnagram)
{
        keyValue = getkey (findAnagram)
        returndictMap.get(key)
}
```

2.2.3.2 Working with Prime Table:

Let us illustrate the use of PRIME table with anexample algorithm. A typical query is to find the anagramsof given string. This can be solved directly with PRIMEtable. For example to find the anagram of "ptrulias"we firstfind the key value of the string "ptrulias", and retrieve thestring mapped with this key value from the Hash Map. Herewe are using Hashmap, so to find the anagram of any givenstring will takes constant time, that is, O(1). PRIME tablealso supports the complex queries.

[3]Different Types of Games that can be played using Anagrams:

3.1 Anagram True Or False:
Determine whether a given string of letters hasan anagram. For example, the input dgo should return true,since it is an anagram of god, while alias should return false,since it has no anagram.

3.2 All-words:
List all words, which use only (but notnecessarily all) of the letters of the input string.
Forexample, from the input FOOT, we can make foot, oft, oof,oot, too, of, oo and to. To solve those kind of queries timetaken by the PRIME table is equal to total number ofcombinations of given string. Let 'C' be the total number ofcombinations of given string, then the time required to getall the anagrams of given word is O(C).

3.3 Wildcard:
Determine whether there is any letter than can beadded to the input to produce an anagram. For example, FXshould return a positive result, because the letter O can beadded to produce fox, I can be added to produce Fix.

[4]Conclusion :

4.1 Simple Data Structure is easy and simple but not efficient for too long string as it sort the all data and then does its anagram which is too time consuming process. So it is not good data structure for 1,00,000 words.

4.2 Anatree is efficientData Structure and gives efficiency of 97%, but for 1,00,000 words it becomes complicated to handle too many words using anatree as we have to generate the tree of each and every word and the compare the anagramic words.

4.3 Prime Table:
For PRIME table, let us assume 'N' be the numberof words in dictionary. Let 'K' be the total number of keysrequired to store the 'N' words in hash map. So the totalspace required for PRIME table is O(N+K).

PRIME table is apowerful data structure for answering letter level equalityand inequality queries. It gives the anagram of given randomword in constant time. Its primary disadvantage is it won'tgive the wildcard of anagram efficiently.

We believe thatfurther optimization in this area is fruitful. And main advantage for 1,00,000 words this data structure will work as it store the letters in table and to compare 1,00,000 words we have just check the product of words and that's it. So prime table datastructure is fast and efficient for our purpose.

Let's see how this data structure works for 1,00,000 it's not possible to feasible to try for all 1,00,000 but we can see how it works.
Eg :"God is Nowhere. Dog is No Where."

Now consider the table of alphabet as above given and make the word table for prime table data structure.

GOD	IS	NOW	HERE	DOG	IS	NO	WHERE
5593	1541	167743	140239	5593	1541	2021	11639837

As we can see the above table and compare the numerical values then the value 5593 is similar in "DOG" and "GOD" both are anagrams. Also we get the similar words withvalues 1541 of "IS".

PRIME table supports solving following queries in constanttime.

III. REFRENCES

[1]Sk. Mohiddin Shaw, Hari Krishna Gurram, Rama Krishna Gurram, DharmaiahGurram, "A Fast Data Structure for Anagrams", IJRITCC,September 2014, ISSN: 2321-8169Volume: 2 Issue: 9 2954 – 2956.
[2] AHO, A. AND ULLMAN, J. "Foundations of Computer Science". W. H. Freeman, New York,1992, PP 542–545.
[3] AKERS, S. B. "Binary decision diagrams", IEEETrans. Comput. 27, 6, (1978) PP 509–516.
[4] APPEL, A. W. AND JACOBSON, G. J, "Theworld"s fastest Scrabble program", Comm. ACM31, (1988) 572–579.
[5] CURRAN, K.,WOODS, D., AND RIORDAN, B."Investigating text input methods for mobilephones", Telemat. Inf. 23, 1, (2006) PP1–21.
[6] DICKSON, L. E. "Diophantine Analysis ". Historyof the Theory of Numbers Series, vol. 2. Chelsea,New York.
[7] GORDON, S. A, "A faster Scrabble movegeneration algorithm", Softw. ract. Exp 24, (1994)PP219–232.
[8] SHANNON, C. E, "Prediction and entropy ofprinted English", Bell Syst. Techn. J. 30, (1951) 50–64.
[9] WARD, D. J. AND MACKAY, D. J. C, "Artificialintelligence: Fast hands-free writing by gazedirection", Nature 418, (2002) 838–840.
[10] CHARLES REAMS, "Anatree: A Fast DataStructure for Anagrams", ACM Journal ofExperimental Algorithmics, Vol. 17, No. 1, (2012).

