

Word Composition Problem Solution

Problem Statement

The given problem statement is all about identifying the

1. longest compounded word
2. second-longest compounded word
3. time taken to process the input file

that given with the problem statement containing alphabetically sorted words. A compounded word is one that can be constructed by concatenating shorter words also found in the same file.

The solution processes two input files,

1. ``Input_01.txt`` (small dataset) and
2. ``Input_02.txt`` (large dataset) .

Approach

- 1) The program uses a "Trie" to store words efficiently and to quickly determine prefixes. This reduces the time complexity of searching for compound words.
- 2) Trie Construction: Words from the input file are inserted into the Trie.
- 3) Queue Processing: A queue is used to store word-suffix pairs for efficient processing of potential compounded words.
- 4) Longest and Second-Longest Identification: Words are evaluated as compounded if their suffixes are either directly present in the Trie or can themselves be further broken into valid words using the Trie.
- 5) By leveraging the Trie and queue, the program ensures efficient prefix lookups and avoids redundant computations.
- 6) Execution time is measured using ``std::chrono`` for precise performance tracking in milliseconds.

Note :

- Before Executing the program please make sure that both the input files are in same directory as the program.
- Make sure you a g++ compiler installed in your system.

Steps to Execute

- Open terminal in same directory where your solution is stored.
- To compile the program write following command in terminal/powershell `"g++ Solution.cpp"` (like `"word_compositon_problem.cpp"`).
- To execute the program write the following command in terminal/powershell `"./a.exe"` or the name of the compiled file.

Sample Output

For the given sample input files:

Input_01.txt

```
word_compositon_problem.cpp > main()
66 class Solution {
87     pair<string, string> findLongestCompoundWords() {
92         while (!q.empty()) {
111             return make_pair(longestWord, secondLongestWord);
112         }
113     }
114 };
115
116 int main() {
117     Solution solinputfile1;
118     Solution solinputfile2;
119     clock_t start = clock();
120
121     solinputfile1.buildfrrie("Input_01.txt");
122     pair<string, string> result = solinputfile1.findLongestCompoundWords();
123     clock_t end = clock();
124     double duration = double(end - start) / double(CLOCKS_PER_SEC);
125     cout << "Longest Compound Word: " << result.first << endl;
126     cout << "Second Longest Compound Word: " << result.second << endl;
127     cout << "Time taken to process file Input_01.txt : " << duration << " seconds" << endl << endl;
128
129     solinputfile2.buildfrrie("Input_02.txt");
130     end = clock();
131     result = solinputfile2.findLongestCompoundWords();
132     duration = double(end - start) / double(CLOCKS_PER_SEC);
133     cout << "Longest Compound Word: " << result.first << endl;
134     cout << "Second Longest Compound Word: " << result.second << endl;
135     cout << "Time taken to process file Input_02.txt : " << duration << " seconds" << endl;
136
137     return 0;
138 }
```

PS D:\CPP> cd "d:\CPA" ; if (\$?) { g++ word_compositon_problem.cpp -o word_compositon_problem } ; if (\$?) { .\word_compositon_problem }

Longest Compound Word: ratcatdogcat
Second Longest Compound Word: catsdogcats
Time taken to process file Input_01.txt : 0 seconds

Input_02.txt

```
word_compositon_problem.cpp > main()
116 int main() {
117     Solution solinputfile2;
118     clock_t start = clock();
119
120     solinputfile1.buildfrrie("Input_01.txt");
121     pair<string, string> result = solinputfile1.findLongestCompoundWords();
122     clock_t end = clock();
123     double duration = double(end - start) / double(CLOCKS_PER_SEC);
124     // cout << "Longest Compound Word: " << result.first << endl;
125     // cout << "Second Longest Compound Word: " << result.second << endl;
126     // cout << "Time taken to process file Input_01.txt : " << duration << " seconds" << endl << endl;
127
128     solinputfile2.buildfrrie("Input_02.txt");
129     end = clock();
130     result = solinputfile2.findLongestCompoundWords();
131     duration = double(end - start) / double(CLOCKS_PER_SEC);
132     cout << "Longest Compound Word: " << result.first << endl;
133     cout << "Second Longest Compound Word: " << result.second << endl;
134     cout << "Time taken to process file Input_02.txt : " << duration << " seconds" << endl;
135
136     return 0;
137 }
```

PS D:\CPP> cd "d:\CPA" ; if (\$?) { g++ word_compositon_problem.cpp -o word_compositon_problem } ; if (\$?) { .\word_compositon_problem }

Longest Compound Word: ethylenediaminetetraacetates
Second Longest Compound Word: electroencephalographically
Time taken to process file Input_02.txt : 1.468 seconds
PS D:\CPP>

Note: Time taken can be different every time and on any system while compile.

SOLUTION BY ETI GARG