For  **./ngram -n 4 shakespeare.txt**

1)

|  |  |  |
| --- | --- | --- |
| ***Comments:*** *original timings* | | |
| **Function** | **% total time** | **time (s)** |
| NgramList::sortByCount() | 69.98 | 6.75 |
| \_\_gnu\_cxx::\_\_enable\_if<std::\_\_is\_char<char>::\_\_value, bool>::\_\_type std::operator==<char>(std::basic\_string<char, std::char\_traits<cha … | 20.2 | 1.95 |
| NgramList::insertNgram(std::string) | 7.48 | 0.72 |
| std::basic\_string<char, std::char\_traits<char>, std::allocator<char> > std::operator+<char, std::char\_traits<char>, std::allocator<char … | 1.3 | 0.13 |
| std::char\_traits<char>::compare(char const\*, char const\*, unsigned long) | 0.73 | 0.07 |
| \_\_gnu\_cxx::\_\_normal\_iterator<std::string const\*, std::vector<std::string, std::allocator<std::string> > >::\_\_normal\_iterator(std::strin … | 0.21 | 0.02 |
| \_ZN9\_\_gnu\_cxx13new\_allocatorISsE9constructISsIRKSsEEEvPT\_DpOT0\_ | 0.1 | 0.01 |
| std::vector<std::string, std::allocator<std::string> >::push\_back(std::string const&) | 0.1 | 0.01 |
| totals -> | 100.1 | 9.66 |

2)

|  |  |  |
| --- | --- | --- |
| ***Comments:*** *implemented sortByCount function with qsort stdlib function instead of bubble sort* | | |
| **Function** | **% total time** | **time (s)** |
| \_\_gnu\_cxx::\_\_enable\_if<std::\_\_is\_char<char>::\_\_value, bool>::\_\_type std::operator==<char>(std::basic\_string<char, std::char\_traits<cha | 73.73 | 1.93 |
| NgramList::insertNgram(std::string) | 22.54 | 0.59 |
| std::char\_traits<char>::compare(char const\*, char const\*, unsigned long) | 3.06 | 0.08 |
| NgramList::compare(void const\*, void const\*) | 0.38 | 0.01 |
| std::move\_iterator<std::string\*>::operator\*() const | 0.19 | 0.01 |
| std::move\_iterator<std::string\*>::operator++() | 0.19 | 0.01 |
| NgramList::sortByCount() | 0 | 0 |
| totals -> | 100.09 | 2.63 |

Notice that sortByCount, which previously took 70% of total time, now takes 0% of total time.

Currently, the speedup is: (9.66s)/(2.63s) = 3.67

3)

|  |  |  |
| --- | --- | --- |
| ***Comments:*** *stored data in a chained hash table as opposed to a linked list* | | |
| **Function** | **% total time** | **time (s)** |
| NgramList::getNextNgram(\_\_gnu\_cxx::\_\_normal\_iterator<std::string const\*, std::vec | 22.24 | 0.02 |
| bool \_\_gnu\_cxx::operator!=<std::string const\*, std::vector<std::string, std::allo | 11.12 | 0.01 |
| void std::\_Destroy<std::string>(std::string\*) | 11.12 | 0.01 |
| operator new(unsigned long, void\*) | 11.12 | 0.01 |
| void std::\_Construct<std::string, std::string>(std::string\*, std::string&&) | 11.12 | 0.01 |
| WordList::addWord(std::string) | 11.12 | 0.01 |
| std::vector<std::string, std::allocator<std::string> >::push\_back(std::string con | 11.12 | 0.01 |
| NgramList::hash(std::string) | 11.12 | 0.01 |
| totals -> | 100.08 | 0.09 |

Total speedup is: (9.66s)/(0.09s) = 107

4)

|  |  |  |
| --- | --- | --- |
| ***Comments:*** *moved to files for my-ngram****;*** *replaced C++ STL qsort function with personal implementation.* | | |
| **Function** | **% total time** | **time (s)** |
| bool \_\_gnu\_cxx::operator!=<std::string const\*, std::vector<std::string, std::allo | 25.02 | 0.01 |
| std::basic\_string<char, std::char\_traits<char>, std::allocator<char> > std::opera | 25.02 | 0.01 |
| WordList::addWord(std::string) | 25.02 | 0.01 |
| myNgramList::hash(std::string) | 25.02 | 0.01 |
| totals -> | 100.08 | 0.04 |

Total speedup is: (9.66)/(0.04) = 242

NOTES:

* The total time taken for table 4 would fluctuate by a few milliseconds and it seemed to effectively be the same as table 3.
* Compiling with optimization level -O2 enabled had negligible effect on final time.

Conclusion: total speedup achieved was in the range: 107 to 242.