Thomas RABIAN
2019

Search Engine

I. Data structures & functions

1. General

This search engine program is composed of 4 methods and a *main()* function. The inverted list is made is set up using an *unordered_map*, allowing us to be faster and more efficient because it uses a hash table. The unordered_map takes a string as the key and an array of integers as the value. The string represents the word we search, and the integers represent the index numbers of the lines in which they are.

2. openFile()

The method *openFile()* allows us to initialize our unordered_map. We give a sting which is the path to a .txt file as the first parameter, and the empty unordered_map as the second parameter. Then, the function fills the unordered_map with the data in the file. At the end, it prints on the screen the time it took for the method to create the inverted list.

3. printMap()

This method takes the unordered_map as parameter and print everything in the format: "word: list of lines the word is in".

4. query() and skipTo()

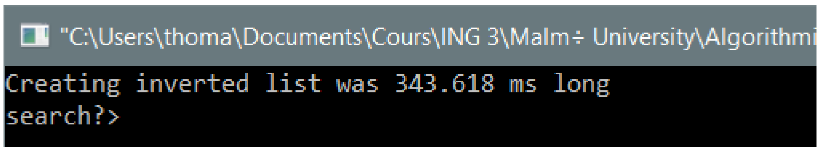
The method *query()* is the central method of the program. It is the one that allow us to search and find the lines in which the word(s) we want is/are situated. It takes the inverted list (unordered_map) as parameter. Then the user needs to input the word/bag of words he wants to find. Each word of the input will then be searched in the inverted list and their data copied in another inverted list (this could have been skipped, but it makes the program easier to understand and the speed results are still very good). If wanted, it is possible to show the results of every words by uncommenting blocs in the if/else. Then, the program search for the word that is in the smaller number of lines. The final results are stored in an array of integers, this array is filled by the content of the array of integers of the word precedingly found (with the smaller number of lines). Then, we run through the arrays of the other words entered by the user and we keep those that are in every word's list. For that we compare the *results* array with the array of each words (that are not the word with the smaller number of lines), and we put every word that is in both in a temporary array. When we have analyzed all the array, we copy the content of the temporary array in the results one and we go to the next word.

To do this efficiently we use the *skipTo()* function. It takes an int vector, a number to find (integer) and a start (unsigned int). Its purpose is to find the "number to find" in the vector. The start is a number, and we are sure that the number we are looking for is not before it. Finally, we browse the vector one by one in order to find or pass the number we want. When we find it, we return its position in the vector. If we didn't find or pass it, we return the size of the vector. We also could have used interesting methods we saw in class such as skipping 2 times more number each time (1->2->4->8…) using the "< 1". But as the results with this technique were very good, I decided that it would not be useful to implement it.

II. Observations

1. Results and displayed content

When we launch the program, we have a first sentence displayed telling us how much time it took for the program to create the inverted list. It also writes "search?>" and give you the right to write something.



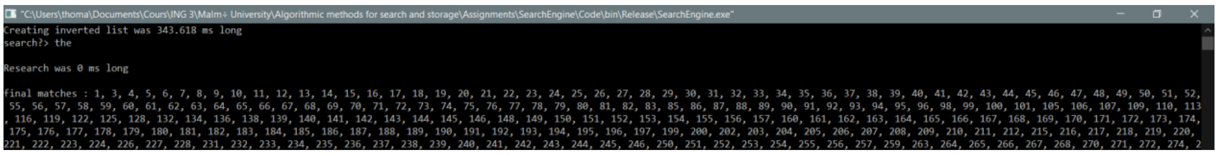
```
"C:\Users\thoma\Documents\Cours\ING 3\Malm÷ University\Algorithmi
Creating inverted list was 343.618 ms long
search?>
```

On this screen you can enter your query. 3 things will be displayed:

- The time it took to find the results
- The word and its results
- The time it took to find and print the results

Here are some examples of results:

- For "the"

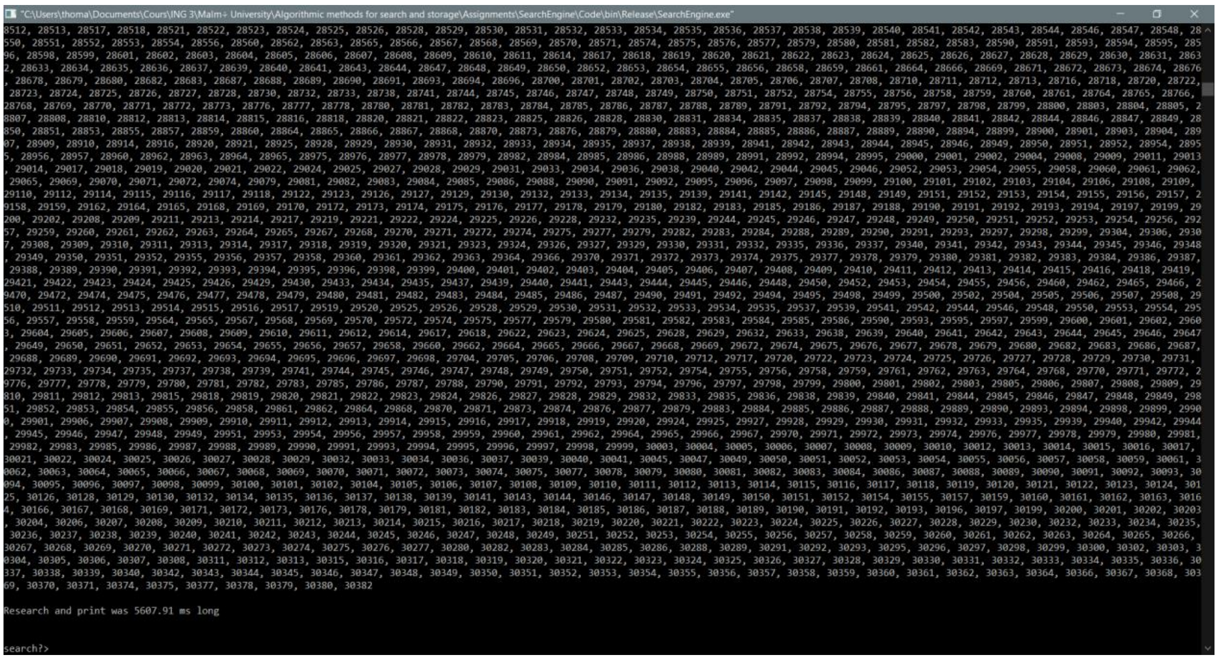


```
"C:\Users\thoma\Documents\Cours\ING 3\Malm÷ University\Algorithmic methods for search and storage\Assignments\SearchEngine\Code\bin\Release\SearchEngine.exe"
Creating inverted list was 343.618 ms long
search?> the

Research was 0 ms long

final matches : 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52,
 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 98, 99, 100, 101, 105, 106, 107, 109, 110, 113
, 116, 119, 122, 125, 128, 132, 134, 136, 138, 139, 140, 141, 142, 143, 144, 145, 146, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174,
 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 199, 200, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 215, 216, 217, 218, 219, 220,
221, 222, 223, 224, 226, 227, 228, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 250, 251, 252, 253, 254, 255, 256, 257, 259, 263, 264, 265, 266, 267, 268, 270, 271, 272, 274, 2
```

…

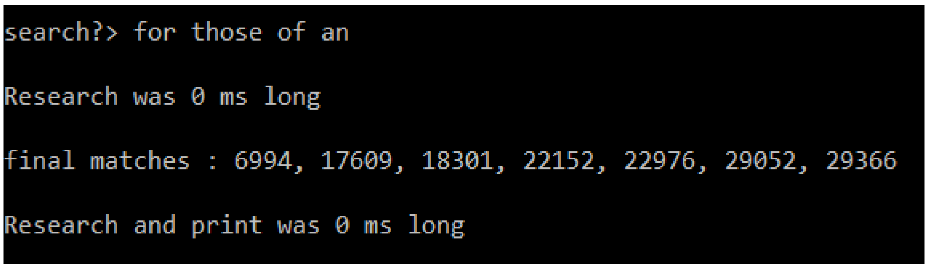


```
"C:\Users\thoma\Documents\Cours\ING 3\Malm÷ University\Algorithmic methods for search and storage\Assignments\SearchEngine\Code\bin\Release\SearchEngine.exe"
8512, 28513, 28517, 28518, 28521, 28522, 28523, 28524, 28525, 28526, 28528, 28529, 28530, 28531, 28532, 28533, 28534, 28535, 28536, 28537, 28538, 28539, 28540, 28541, 28542, 28543, 28544, 28546, 28547, 28548, 28
550, 28551, 28552, 28553, 28554, 28556, 28560, 28562, 28563, 28565, 28566, 28567, 28568, 28569, 28570, 28571, 28574, 28575, 28576, 28577, 28579, 28580, 28581, 28582, 28583, 28590, 28591, 28593, 28594, 28595, 285
96, 28598, 28599, 28601, 28602, 28603, 28604, 28605, 28606, 28607, 28608, 28609, 28610, 28611, 28614, 28617, 28618, 28619, 28620, 28621, 28622, 28623, 28624, 28625, 28626, 28627, 28628, 28629, 28630, 28631, 2863
2, 28633, 28634, 28635, 28636, 28637, 28639, 28640, 28641, 28643, 28644, 28647, 28648, 28649, 28650, 28652, 28653, 28654, 28655, 28656, 28658, 28659, 28661, 28664, 28666, 28669, 28671, 28672, 28673, 28674, 28676
, 28678, 28679, 28680, 28682, 28683, 28687, 28688, 28689, 28690, 28691, 28693, 28694, 28696, 28700, 28701, 28702, 28703, 28704, 28705, 28706, 28707, 28708, 28710, 28711, 28712, 28713, 28716, 28718, 28720, 28722,
 28723, 28724, 28725, 28726, 28727, 28728, 28730, 28732, 28733, 28738, 28741, 28744, 28745, 28746, 28747, 28748, 28749, 28750, 28751, 28752, 28754, 28755, 28756, 28758, 28759, 28760, 28761, 28764, 28765, 28766,
28768, 28769, 28770, 28771, 28772, 28773, 28776, 28777, 28778, 28780, 28781, 28782, 28783, 28784, 28785, 28786, 28787, 28788, 28789, 28791, 28792, 28794, 28795, 28797, 28798, 28799, 28800, 28803, 28804, 28805, 2
8807, 28808, 28810, 28812, 28813, 28814, 28815, 28816, 28818, 28820, 28821, 28822, 28823, 28825, 28826, 28828, 28830, 28831, 28834, 28835, 28837, 28838, 28839, 28840, 28841, 28842, 28844, 28846, 28847, 28849, 28
850, 28851, 28853, 28855, 28857, 28859, 28860, 28864, 28865, 28866, 28867, 28868, 28870, 28873, 28876, 28879, 28880, 28883, 28884, 28885, 28886, 28887, 28889, 28890, 28894, 28899, 28900, 28901, 28903, 28904, 289
07, 28909, 28910, 28914, 28916, 28920, 28921, 28925, 28928, 28929, 28930, 28931, 28932, 28933, 28934, 28935, 28937, 28938, 28939, 28941, 28942, 28943, 28944, 28945, 28946, 28949, 28950, 28951, 28952, 28954, 2895
5, 28956, 28957, 28960, 28962, 28963, 28964, 28965, 28975, 28976, 28977, 28978, 28979, 28982, 28984, 28985, 28986, 28988, 28989, 28991, 28992, 28994, 28995, 29000, 29001, 29002, 29004, 29008, 29009, 29011, 29013
, 29014, 29017, 29018, 29019, 29020, 29021, 29022, 29024, 29025, 29027, 29028, 29029, 29031, 29033, 29034, 29036, 29038, 29040, 29042, 29044, 29045, 29046, 29052, 29053, 29054, 29055, 29058, 29060, 29061, 29062,
 29065, 29069, 29070, 29071, 29072, 29074, 29079, 29081, 29082, 29083, 29084, 29085, 29086, 29088, 29090, 29091, 29092, 29095, 29096, 29097, 29098, 29099, 29100, 29101, 29102, 29103, 29104, 29106, 29108, 29109,
29110, 29112, 29114, 29115, 29116, 29117, 29118, 29122, 29123, 29126, 29127, 29129, 29130, 29132, 29133, 29134, 29135, 29139, 29141, 29142, 29145, 29148, 29149, 29151, 29152, 29153, 29154, 29155, 29156, 29157, 2
9158, 29159, 29162, 29164, 29165, 29168, 29169, 29170, 29172, 29173, 29174, 29175, 29176, 29177, 29178, 29179, 29180, 29182, 29183, 29185, 29186, 29187, 29188, 29190, 29191, 29192, 29193, 29194, 29197, 29199, 29
200, 29202, 29208, 29209, 29211, 29213, 29214, 29217, 29219, 29221, 29222, 29224, 29225, 29226, 29228, 29232, 29235, 29239, 29244, 29245, 29246, 29247, 29248, 29249, 29250, 29251, 29252, 29253, 29254, 29256, 292
57, 29259, 29260, 29261, 29262, 29263, 29264, 29265, 29267, 29268, 29270, 29271, 29272, 29274, 29275, 29277, 29279, 29282, 29283, 29284, 29288, 29289, 29290, 29291, 29293, 29297, 29298, 29299, 29304, 29306, 2930
7, 29308, 29309, 29310, 29311, 29313, 29314, 29317, 29318, 29319, 29320, 29321, 29323, 29324, 29326, 29327, 29329, 29330, 29331, 29332, 29335, 29336, 29337, 29340, 29341, 29342, 29343, 29344, 29345, 29346, 29348
, 29349, 29350, 29351, 29352, 29355, 29356, 29357, 29358, 29360, 29361, 29362, 29363, 29364, 29366, 29370, 29371, 29372, 29373, 29374, 29375, 29377, 29378, 29379, 29380, 29381, 29382, 29383, 29384, 29386, 29387,
 29388, 29389, 29390, 29391, 29392, 29393, 29394, 29395, 29396, 29398, 29399, 29400, 29401, 29402, 29403, 29404, 29405, 29406, 29407, 29408, 29409, 29410, 29411, 29412, 29413, 29414, 29415, 29416, 29418, 29419,
29421, 29422, 29423, 29424, 29425, 29426, 29429, 29430, 29433, 29434, 29435, 29437, 29439, 29440, 29441, 29443, 29444, 29445, 29446, 29448, 29450, 29452, 29453, 29454, 29455, 29456, 29460, 29462, 29465, 29466, 2
9470, 29472, 29474, 29475, 29476, 29477, 29478, 29479, 29480, 29481, 29482, 29483, 29484, 29485, 29486, 29487, 29490, 29491, 29492, 29494, 29495, 29498, 29499, 29500, 29502, 29504, 29505, 29506, 29507, 29508, 29
510, 29511, 29512, 29513, 29514, 29515, 29516, 29517, 29519, 29520, 29525, 29526, 29528, 29529, 29530, 29531, 29532, 29533, 29534, 29535, 29537, 29539, 29541, 29542, 29544, 29546, 29548, 29550, 29553, 29554, 295
56, 29557, 29558, 29559, 29564, 29565, 29567, 29568, 29569, 29570, 29572, 29574, 29575, 29577, 29579, 29580, 29581, 29582, 29583, 29584, 29585, 29586, 29590, 29593, 29595, 29597, 29599, 29600, 29601, 29602, 2960
3, 29604, 29605, 29606, 29607, 29608, 29609, 29610, 29611, 29612, 29614, 29617, 29618, 29622, 29623, 29624, 29625, 29628, 29629, 29632, 29633, 29638, 29639, 29640, 29641, 29642, 29643, 29644, 29645, 29646, 29647
, 29649, 29650, 29651, 29652, 29653, 29654, 29655, 29656, 29657, 29658, 29660, 29662, 29664, 29665, 29666, 29667, 29668, 29669, 29672, 29674, 29675, 29676, 29677, 29678, 29679, 29680, 29682, 29683, 29686, 29687,
 29688, 29689, 29690, 29691, 29692, 29693, 29694, 29695, 29696, 29697, 29698, 29704, 29705, 29706, 29708, 29709, 29710, 29712, 29717, 29720, 29722, 29723, 29724, 29725, 29726, 29727, 29728, 29729, 29730, 29731,
29732, 29733, 29734, 29735, 29737, 29738, 29739, 29741, 29744, 29745, 29746, 29747, 29748, 29749, 29750, 29751, 29752, 29754, 29755, 29756, 29758, 29759, 29761, 29762, 29763, 29764, 29768, 29770, 29771, 29772, 2
9776, 29777, 29778, 29779, 29780, 29781, 29782, 29783, 29785, 29786, 29787, 29788, 29790, 29791, 29792, 29793, 29794, 29796, 29797, 29798, 29799, 29800, 29801, 29802, 29803, 29805, 29806, 29807, 29808, 29809, 29
810, 29811, 29812, 29813, 29815, 29818, 29819, 29820, 29821, 29822, 29823, 29824, 29826, 29827, 29828, 29829, 29832, 29833, 29835, 29836, 29838, 29839, 29840, 29841, 29844, 29845, 29846, 29847, 29848, 29849, 298
51, 29852, 29853, 29854, 29855, 29856, 29858, 29861, 29862, 29864, 29868, 29870, 29871, 29873, 29874, 29876, 29877, 29879, 29883, 29884, 29885, 29886, 29887, 29888, 29889, 29890, 29893, 29894, 29898, 29899, 2990
0, 29901, 29906, 29907, 29908, 29909, 29910, 29911, 29912, 29913, 29914, 29915, 29916, 29917, 29918, 29919, 29920, 29924, 29925, 29927, 29928, 29929, 29930, 29931, 29932, 29933, 29935, 29939, 29940, 29942, 29944
, 29945, 29946, 29947, 29948, 29949, 29951, 29953, 29954, 29956, 29957, 29958, 29959, 29960, 29961, 29962, 29964, 29965, 29966, 29967, 29970, 29971, 29972, 29973, 29974, 29976, 29977, 29978, 29979, 29980, 29981,
 29982, 29983, 29985, 29986, 29987, 29988, 29989, 29990, 29991, 29993, 29994, 29995, 29996, 29997, 29998, 29999, 30003, 30004, 30005, 30006, 30007, 30008, 30009, 30010, 30012, 30013, 30014, 30015, 30016, 30017,
30021, 30022, 30024, 30025, 30026, 30027, 30028, 30029, 30032, 30033, 30034, 30036, 30037, 30039, 30040, 30041, 30045, 30047, 30049, 30050, 30051, 30052, 30053, 30054, 30055, 30056, 30057, 30058, 30059, 30061, 3
0062, 30063, 30064, 30065, 30066, 30067, 30068, 30069, 30070, 30071, 30072, 30073, 30074, 30075, 30077, 30078, 30079, 30080, 30081, 30082, 30083, 30084, 30086, 30087, 30088, 30089, 30090, 30091, 30092, 30093, 30
094, 30095, 30096, 30097, 30098, 30099, 30100, 30101, 30102, 30104, 30105, 30106, 30107, 30108, 30109, 30110, 30111, 30112, 30113, 30114, 30115, 30116, 30117, 30118, 30119, 30120, 30121, 30122, 30123, 30124, 301
25, 30126, 30128, 30129, 30130, 30132, 30134, 30135, 30136, 30137, 30138, 30139, 30141, 30143, 30144, 30146, 30147, 30148, 30149, 30150, 30151, 30152, 30154, 30155, 30157, 30159, 30160, 30161, 30162, 30163, 3016
4, 30166, 30167, 30168, 30169, 30171, 30172, 30173, 30176, 30178, 30179, 30181, 30182, 30183, 30184, 30185, 30186, 30187, 30188, 30189, 30190, 30191, 30192, 30193, 30196, 30197, 30199, 30200, 30201, 30202, 30203
, 30204, 30206, 30207, 30208, 30209, 30210, 30211, 30212, 30213, 30214, 30215, 30216, 30217, 30218, 30219, 30220, 30221, 30222, 30223, 30224, 30225, 30226, 30227, 30228, 30229, 30230, 30232, 30233, 30234, 30235,
 30236, 30237, 30238, 30239, 30240, 30241, 30242, 30243, 30244, 30245, 30246, 30247, 30248, 30249, 30251, 30252, 30253, 30254, 30255, 30256, 30257, 30258, 30259, 30260, 30261, 30262, 30263, 30264, 30265, 30266,
30267, 30268, 30269, 30270, 30271, 30272, 30273, 30274, 30275, 30276, 30277, 30280, 30282, 30283, 30284, 30285, 30286, 30288, 30289, 30291, 30292, 30293, 30295, 30296, 30297, 30298, 30299, 30300, 30302, 30303, 3
0304, 30305, 30306, 30307, 30308, 30311, 30312, 30313, 30315, 30316, 30317, 30318, 30319, 30320, 30321, 30322, 30323, 30324, 30325, 30326, 30327, 30328, 30329, 30330, 30331, 30332, 30333, 30334, 30335, 30336, 30
337, 30338, 30339, 30340, 30342, 30343, 30344, 30345, 30346, 30347, 30348, 30349, 30350, 30351, 30352, 30353, 30354, 30355, 30356, 30357, 30358, 30359, 30360, 30361, 30362, 30363, 30364, 30366, 30367, 30368, 303
69, 30370, 30371, 30374, 30375, 30377, 30378, 30379, 30380, 30382

Research and print was 5607.91 ms long


search?>
```

- For “for those of an”



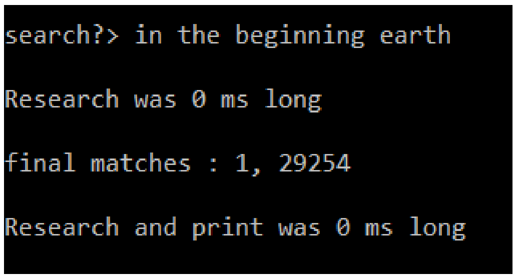
```
search?> for those of an

Research was 0 ms long

final matches : 6994, 17609, 18301, 22152, 22976, 29052, 29366

Research and print was 0 ms long
```

- For “in the beginning earth”



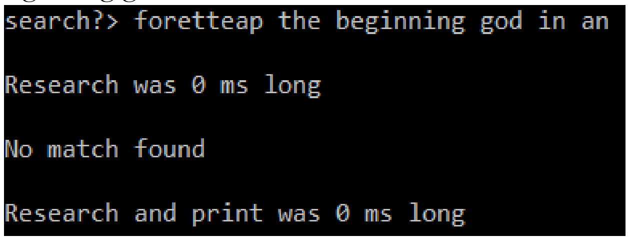
```
search?> in the beginning earth

Research was 0 ms long

final matches : 1, 29254

Research and print was 0 ms long
```

- For “foretteap the beginning god in an”



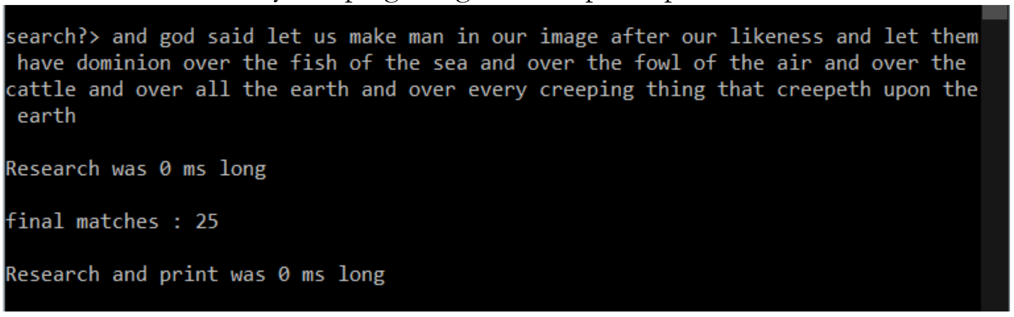
```
search?> foretteap the beginning god in an

Research was 0 ms long

No match found

Research and print was 0 ms long
```

- For “and god said let us make man in our image after our likeness and let them have dominion over the fish of the sea and over the fowl of the air and over the cattle and over all the earth and over every creeping thing that creepeth upon the earth”



```
search?> and god said let us make man in our image after our likeness and let them
 have dominion over the fish of the sea and over the fowl of the air and over the
cattle and over all the earth and over every creeping thing that creepeth upon the
 earth

Research was 0 ms long

final matches : 25

Research and print was 0 ms long
```

2. Analyze of the results

As we can see, all the tests we tried were very fast: less than 1ms to find the results. Even with long sentences the results were very fast. The print time was sometimes a little bit long, but the algorithm in its own is very efficient.

III. How can we be sure that there is no query that would make it perform much worse?

There are a lot of verifications that have been put in place in the query function that permits to make sure everything works well. First of all, if we find a word that has no match at all while browsing to find the word with the smaller number of occurrences, we leave the loop and we go straight to the printing part. Moreover, the loop that compares the results with each inverted list can be ended either when we have looked in every array OR when the results array is empty which allows us to end quickly if we don't have results while there is still 1000 word to look up. Also, the skipTo function starts its research at the index that was used in the calling function which allows us to save a lot of time browsing the array knowing nothing will be there.