

Tugas Kecil 1 IF2211 Strategi Algoritma

Semester II Tahun 2021/2022

Laporan Penyelesaian *Word Search Puzzle* dengan Algoritma Brute Force

Oleh:

Adiyansa Prasetya Wicaksana

13520044



PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2022

A. Algoritma *Brute Force*

Algoritma *brute force* merupakan cara penyelesaian masalah dengan program yang meninjau semua kasus yang mungkin muncul berdasarkan deskripsi masalah yang diberikan atau bisa dibilang algoritma diimplemantasikan secara *straight-forward*. Meskipun dengan definisi yang diberikan, tetap ada banyak cara berbeda dalam menyelesaikan satu persoalan yang sama dengan metode *brute force*. Keragaman ini disebabkan pendekatan yang berbeda dalam mengerjakan soal, beberapa observasi yang dapat mengoptimalkan dan membuang kasus yang tidak perlu, atau pemilihan aspek mana yang di-*brute force*-kan. Secara keseluruhan, *brute force* merupakan algoritma yang mudah ditemukan dan diimplementasikan, namun membutuhkan waktu dan memori yang banyak.

Word search puzzle adalah permainan kata dimana pemain harus menemukan beberapa kata tersembunyi dalam kumpulan huruf acak. Kumpulan huruf tersebut biasa diletakkan pada “papan” berbentuk segi empat atau dapat disebut juga matriks huruf. Kata-kata pada matriks huruf ini dapat ditemukan dalam delapan arah yang mungkin, yaitu, vertikal ke atas, vertikal ke bawah, horizontal ke kanan, horizontal ke kiri, diagonal ke kanan atas, diagonal ke kanan bawah, diagonal ke kiri atas, dan diagonal ke kiri bawah.

B. Penyelesaian *Word Search Puzzle* dengan *Brute Force*

Salah satu permasalahan yang dapat diselesaikan dengan implementasi *brute force* adalah permainan *word search puzzle* yang bertujuan untuk mencari kata yang tepat pada urutan tertentu di dalam *grid* yang berisi huruf-huruf. Dalam implementasi yang dilakukan, digunakan Bahasa C++ untuk menyelesaikan problematika ini. Terdapat dua file yang berada dalam *source code*. File pertama bernama `solver.cpp`, yang berisi algoritma untuk menampilkan output dan juga algoritma pencarian kata terhadap puzzlenya. File kedua yang sekaligus menjadi file utama bernama `word_search_puzzle.cpp` yang berisi algoritma utama dalam penyelesaian program.

File `word_search_puzzle.cpp`

Beberapa fungsi atau prosedur yang digunakan di dalam file ini adalah:

- `getPuzzle`

Fungsi `getPuzzle` berfungsi untuk membaca output bagian pertama dalam file yang akan menjadi puzzle huruf-huruf. Puzzle ini diimplemantasikan dalam matriks 2 dimensi yang menggunakan vektor sebagai implementasinya.

- `getWord`

Fungsi `getWord` berfungsi untuk membaca output bagian kedua dalam file yang akan menjadi kata-kata yang dicari dalam puzzle. Kata-kata ini diimplementasikan dalam vektor (array) sehingga dapat dilakukan iterasi untuk pencarian setiap katanya.

- `main`

Fungsi `main` ini merupakan fungsi utama dalam program ini. Di dalam fungsi ini diimplementasikan berbagai fungsi lainnya seperti `getPuzzle` dan `getWord`. Disini dilakukan iterasi terhadap setiap kata yang terdapat dalam vektor `word`, lalu dilakukan search delapan arah ke setiap elemen puzzle.

File solver.cpp

Beberapa fungsi atau prosedur yang digunakan di dalam file ini adalah:

- `printHash`

Fungsi `printHash` mengeluarkan output terhadap hash table yang telah dibuat. Hash table ini merupakan matriks 2 dimensi dengan implementasi dengan besarnya sesuai dengan besar dari puzzlenya. Nilai dari hash table berisi nilai dari 0 sampai 7 sebagai tanda untuk pewarnaan kata yang ditemukan. 0 berarti tidak ditemukannya kata pada huruf di puzzlenya dan 1 sampai 7 berkorespondensi berdasarkan warnanya (ditemukan kata).

- `searchHorizontalRight`
- `searchHorizontalLeft`
- `searchVerticalUp`
- `searchVerticalDown`
- `searchDiagonalRightUp`
- `searchDiagonalRightDown`
- `searchDiagonalLeftUp`
- `searchDiagonalLeftDown`

Delapan fungsi `search` bertujuan dan berstruktur mirip yaitu untuk melakukan pencarian terhadap elemen puzzle untuk menemukan kata tujuan. Setiap fungsi merepresentasikan 8 arah yang ingin dicari. Konsep dari setiap fungsinya adalah dengan melakukan iterasi kepada setiap elemen dan setiap elemennya dilakukan iterasi tergantung dari arahnya sebanyak panjang dari kata yang ingin dicari untuk setiap elemen. Jika setiap huruf cocok dengan setiap huruf dari kata yang dicari maka algoritma akan menyimpan kata yang telah ditemukan ke dalam hash table dengan nilai 1 sampai 7 tergantung dari indeks pada inputnya.

C. Source Program

1. File word_puzzle_search.cpp

```
1  #include <iostream>
2  #include <fstream>
3  #include <vector>
4  #include <string>
5  #include <chrono>
6  #include "solver.cpp"
7  using namespace std::chrono;
8  using namespace std;
9
```

```
1  vector<vector<char>> getPuzzle(ifstream &InputText)
2  {
3      vector<vector<char>> puzzle;
4      vector<char> puzzleLine;
5      char input;
6      int countNewLine = 0;
7
8      while (InputText.get(input))
9      {
10         if (input == '\n')
11         {
12             countNewLine++;
13
14             if (countNewLine == 2)
15             {
16                 break;
17             }
18             puzzle.push_back(puzzleLine);
19             puzzleLine = {};
20
21             continue;
22         }
23
24         countNewLine = 0;
25
26         if (input != ' ')
27         {
28             puzzleLine.push_back(input);
29         }
30     }
31
32     return puzzle;
33 }
```

```
1  vector<string> getWord(ifstream &InputText)
2  {
3      vector<string> word;
4      char input;
5      string text;
6
7      while (InputText.get(input))
8      {
9         if (input == '\n')
10         {
11             word.push_back(text);
12             text = "";
13         }
14
15         else
16         {
17             text.push_back(input);
18         }
19     }
20     return word;
21 }
```

```

1  int main()
2  {
3      string fileName;
4      cout << "Input your file name: ";
5      cin >> fileName;
6
7      // Read from the text file
8      string path;
9      path = "../test/" + fileName;
10     ifstream InputText(path);
11
12     while (!InputText)
13     {
14         cout << "File name doesn't exist, please try again!" << endl;
15         cout << "Input your file name: ";
16         cin >> fileName;
17         path = "../test/" + fileName;
18         InputText.clear();
19         InputText.seekg(0, InputText.beg);
20         InputText.open(path);
21     }
22
23     // Get Puzzle
24     vector<vector<char>> puzzle = getPuzzle(InputText);
25
26     // Get Word
27     vector<string> word = getWord(InputText);
28
29     // Create Boolean Hash Table
30     vector<vector<int>> puzzleHash;
31     for (int i = 0; i < puzzle.size(); i++)
32     {
33         puzzleHash.push_back({});
34         for (int j = 0; j < puzzle[0].size(); j++)
35         {
36             puzzleHash[i].push_back(0);
37         }
38     }
39
40     // Start runtime
41     auto start = high_resolution_clock::now();
42
43     // Check for every iteration for every word
44     int compareCount = 0;
45     for (int i = 0; i < word.size(); i++)
46     {
47         searchHorizontalRight(puzzle, puzzle.size(), puzzle[0].size(), word[i], puzzleHash, i, compareCount);
48         searchHorizontalLeft(puzzle, puzzle.size(), puzzle[0].size(), word[i], puzzleHash, i, compareCount);
49         searchVerticalUp(puzzle, puzzle.size(), puzzle[0].size(), word[i], puzzleHash, i, compareCount);
50         searchVerticalDown(puzzle, puzzle.size(), puzzle[0].size(), word[i], puzzleHash, i, compareCount);
51         searchDiagonalRightUp(puzzle, puzzle.size(), puzzle[0].size(), word[i], puzzleHash, i, compareCount);
52         searchDiagonalRightDown(puzzle, puzzle.size(), puzzle[0].size(), word[i], puzzleHash, i, compareCount);
53         searchDiagonalLeftUp(puzzle, puzzle.size(), puzzle[0].size(), word[i], puzzleHash, i, compareCount);
54         searchDiagonalLeftDown(puzzle, puzzle.size(), puzzle[0].size(), word[i], puzzleHash, i, compareCount);
55     }
56
57     auto stop = high_resolution_clock::now();
58     auto duration = duration_cast<microseconds>(stop - start);
59
60     cout << endl;
61     printHash(puzzle, puzzleHash);
62     cout << "Execution Time : " << (float)(duration.count()) / 1000000 << " seconds" << endl;
63     cout << "Total Comparision : " << compareCount << endl;
64
65     InputText.close();
66 }

```

2. File solver.cpp

```
1 #define RESET "\033[0m"
2 #define BLACK "\033[30m" /* Black */
3 #define RED "\033[31m" /* Red */
4 #define GREEN "\033[32m" /* Green */
5 #define YELLOW "\033[33m" /* Yellow */
6 #define BLUE "\033[34m" /* Blue */
7 #define MAGENTA "\033[35m" /* Magenta */
8 #define CYAN "\033[36m" /* Cyan */
9 #define WHITE "\033[37m" /* White */
10 #include <iostream>
11 #include <fstream>
12 #include <vector>
13 #include <string>
14 using namespace std;
```

```
1 void printHash(vector<vector<char>> puzzle, vector<vector<int>> puzzleHash)
2 {
3     int i, j;
4     int row = puzzle.size();
5     int col = puzzle[0].size();
6
7     for (i = 0; i < row; i++)
8     {
9         for (j = 0; j < col; j++)
10         {
11             if (puzzleHash[i][j] == 1)
12             {
13                 cout << WHITE << puzzle[i][j] << RESET << " ";
14             }
15
16             else if (puzzleHash[i][j] == 2)
17             {
18                 cout << CYAN << puzzle[i][j] << RESET << " ";
19             }
20
21             else if (puzzleHash[i][j] == 3)
22             {
23                 cout << RED << puzzle[i][j] << RESET << " ";
24             }
25
26             else if (puzzleHash[i][j] == 4)
27             {
28                 cout << GREEN << puzzle[i][j] << RESET << " ";
29             }
30
31             else if (puzzleHash[i][j] == 5)
32             {
33                 cout << YELLOW << puzzle[i][j] << RESET << " ";
34             }
35
36             else if (puzzleHash[i][j] == 6)
37             {
38                 cout << BLUE << puzzle[i][j] << RESET << " ";
39             }
40
41             else if (puzzleHash[i][j] == 7)
42             {
43                 cout << MAGENTA << puzzle[i][j] << RESET << " ";
44             }
45
46             else
47             {
48                 cout << BLACK << puzzle[i][j] << RESET << " ";
49             }
50         }
51         cout << endl;
52     }
53
54     cout << endl;
55 }
```

```

1 void searchHorizontalRight(vector<vector<char>> puzzle, int row, int col,
2 string word, vector<vector<int>> &puzzleHash, int wordNumber)
3 {
4     bool found = false;
5     int wordSize = word.size();
6     int wordCount = 0;
7     int i, j, k;
8     int p, q;
9
10    for (i = 0; i < row; i++)
11    {
12        for (j = 0; j < col; j++)
13        {
14            wordCount = 0;
15            for (k = j; k - j < wordSize; k++)
16            {
17                if (k ≥ col)
18                {
19                    break;
20                }
21
22                if (puzzle[i][k] == word[k - j])
23                {
24                    wordCount++;
25                }
26                else
27                {
28                    break;
29                }
30            }
31
32            if (wordCount == wordSize)
33            {
34                found = true;
35                q = j;
36                break;
37            }
38        }
39        if (found)
40        {
41            p = i;
42            break;
43        }
44    }
45
46    if (found)
47    {
48
49        for (j = q; j - q < wordSize; j++)
50        {
51            puzzleHash[p][j] = (wordNumber % 7) + 1;
52        }
53    }
54 }

```



```
1 void searchHorizontalLeft(vector<vector<char>> puzzle, int row, int col,
2 string word, vector<vector<int>> &puzzleHash, int wordNumber)
3 {
4     bool found = false;
5     int wordSize = word.size();
6     int wordCount = 0;
7     int i, j, k;
8     int p, q;
9
10    for (i = 0; i < row; i++)
11    {
12        for (j = 0; j < col; j++)
13        {
14            wordCount = 0;
15            for (k = j; j - k < wordSize; k--)
16            {
17                if (k < 0)
18                {
19                    break;
20                }
21
22                if (puzzle[i][k] == word[j - k])
23                {
24                    wordCount++;
25                }
26                else
27                {
28                    break;
29                }
30            }
31
32            if (wordCount == wordSize)
33            {
34                found = true;
35                q = j;
36                break;
37            }
38        }
39        if (found)
40        {
41            p = i;
42            break;
43        }
44    }
45
46    if (found)
47    {
48
49        for (j = q; q - j < wordSize; j--)
50        {
51            puzzleHash[p][j] = (wordNumber % 7) + 1;
52        }
53    }
54 }
```



```
1 void searchVerticalUp(vector<vector<char>> puzzle, int row, int col,
2 string word, vector<vector<int>> &puzzleHash, int wordNumber)
3 {
4     bool found = false;
5     int wordSize = word.size();
6     int wordCount = 0;
7     int i, j, k;
8     int p, q;
9
10    for (i = 0; i < row; i++)
11    {
12        for (j = 0; j < col; j++)
13        {
14            wordCount = 0;
15            for (k = i; i - k < wordSize; k--)
16            {
17                if (k < 0)
18                {
19                    break;
20                }
21
22                if (puzzle[k][j] == word[i - k])
23                {
24                    wordCount++;
25                }
26                else
27                {
28                    break;
29                }
30            }
31
32            if (wordCount == wordSize)
33            {
34                found = true;
35                q = j;
36                break;
37            }
38        }
39        if (found)
40        {
41            p = i;
42            break;
43        }
44    }
45
46    if (found)
47    {
48
49        for (i = p; p - i < wordSize; i--)
50        {
51            puzzleHash[i][q] = (wordNumber % 7) + 1;
52        }
53    }
54 }
```

```
1 void searchVerticalDown(vector<vector<char>> puzzle, int row, int col,
2 string word, vector<vector<int>> &puzzleHash, int wordNumber)
3 {
4     bool found = false;
5     int wordSize = word.size();
6     int wordCount = 0;
7     int i, j, k;
8     int p, q;
9
10    for (i = 0; i < row; i++)
11    {
12        for (j = 0; j < col; j++)
13        {
14            wordCount = 0;
15            for (k = i; k - i < wordSize; k++)
16            {
17                if (k ≥ row)
18                {
19                    break;
20                }
21
22                if (puzzle[k][j] == word[k - i])
23                {
24                    wordCount++;
25                }
26                else
27                {
28                    break;
29                }
30            }
31
32            if (wordCount == wordSize)
33            {
34                found = true;
35                q = j;
36                break;
37            }
38        }
39        if (found)
40        {
41            p = i;
42            break;
43        }
44    }
45
46    if (found)
47    {
48
49        for (i = p; i - p < wordSize; i++)
50        {
51            puzzleHash[i][q] = (wordNumber % 7) + 1;
52        }
53    }
54 }
```

```

1 void searchDiagonalRightUp(vector<vector<char>> puzzle, int row, int col,
2 string word, vector<vector<int>> &puzzleHash, int wordNumber)
3 {
4     bool found = false;
5     int wordSize = word.size();
6     int wordCount = 0;
7     int i, j, k, l;
8     int p, q;
9
10    for (i = 0; i < row; i++)
11    {
12        for (j = 0; j < col; j++)
13        {
14            wordCount = 0;
15            k = i;
16            l = j;
17            while (i - k < wordSize && l - j < wordSize)
18            {
19                if (k < 0 || l ≥ col)
20                {
21                    break;
22                }
23
24                if (puzzle[k][l] == word[i - k])
25                {
26                    wordCount++;
27                }
28                else
29                {
30                    break;
31                }
32                k--;
33                l++;
34            }
35
36            if (wordCount == wordSize)
37            {
38                found = true;
39                q = j;
40                break;
41            }
42        }
43        if (found)
44        {
45            p = i;
46            break;
47        }
48    }
49
50    if (found)
51    {
52
53        i = p;
54        j = q;
55        while (p - i < wordSize && j - q < wordSize)
56        {
57            puzzleHash[i][j] = (wordNumber % 7) + 1;
58            i--;
59            j++;
60        }
61    }
62 }

```

```

1 void searchDiagonalRightDown(vector<vector<char>> puzzle, int row, int col,
2 string word, vector<vector<int>> &puzzleHash, int wordNumber)
3 {
4     bool found = false;
5     int wordSize = word.size();
6     int wordCount = 0;
7     int i, j, k, l;
8     int p, q;
9
10    for (i = 0; i < row; i++)
11    {
12        for (j = 0; j < col; j++)
13        {
14            wordCount = 0;
15            k = i;
16            l = j;
17            while (k - i < wordSize && l - j < wordSize)
18            {
19                if (k >= row || l >= col)
20                {
21                    break;
22                }
23
24                if (puzzle[k][l] == word[k - i])
25                {
26                    wordCount++;
27                }
28                else
29                {
30                    break;
31                }
32                k++;
33                l++;
34            }
35
36            if (wordCount == wordSize)
37            {
38                found = true;
39                q = j;
40                break;
41            }
42        }
43        if (found)
44        {
45            p = i;
46            break;
47        }
48    }
49
50    if (found)
51    {
52
53        i = p;
54        j = q;
55        while (i - p < wordSize && j - q < wordSize)
56        {
57            puzzleHash[i][j] = (wordNumber % 7) + 1;
58            i++;
59            j++;
60        }
61    }
62 }

```



```
1 void searchDiagonalLeftUp(vector<vector<char>> puzzle, int row, int col,
2 string word, vector<vector<int>> &puzzleHash, int wordNumber)
3 {
4     bool found = false;
5     int wordSize = word.size();
6     int wordCount = 0;
7     int i, j, k, l;
8     int p, q;
9
10    for (i = 0; i < row; i++)
11    {
12        for (j = 0; j < col; j++)
13        {
14            wordCount = 0;
15            k = i;
16            l = j;
17            while (i - k < wordSize && j - l < wordSize)
18            {
19                if (k < 0 || l < 0)
20                {
21                    break;
22                }
23
24                if (puzzle[k][l] == word[i - k])
25                {
26                    wordCount++;
27                }
28                else
29                {
30                    break;
31                }
32                k--;
33                l--;
34            }
35
36            if (wordCount == wordSize)
37            {
38                found = true;
39                q = j;
40                break;
41            }
42        }
43        if (found)
44        {
45            p = i;
46            break;
47        }
48    }
49
50    if (found)
51    {
52
53        i = p;
54        j = q;
55        while (p - i < wordSize && q - j < wordSize)
56        {
57            puzzleHash[i][j] = (wordNumber % 7) + 1;
58            i--;
59            j--;
60        }
61    }
62 }
```

```

1 void searchDiagonalLeftDown(vector<vector<char>> puzzle, int row, int col,
2 string word, vector<vector<int>> &puzzleHash, int wordNumber)
3 {
4     bool found = false;
5     int wordSize = word.size();
6     int wordCount = 0;
7     int i, j, k, l;
8     int p, q;
9
10    for (i = 0; i < row; i++)
11    {
12        for (j = 0; j < col; j++)
13        {
14            wordCount = 0;
15            k = i;
16            l = j;
17            while (k - i < wordSize && j - l < wordSize)
18            {
19                if (k >= row || l < 0)
20                {
21                    break;
22                }
23
24                if (puzzle[k][l] == word[j - l])
25                {
26                    wordCount++;
27                }
28                else
29                {
30                    break;
31                }
32                k++;
33                l--;
34            }
35
36            if (wordCount == wordSize)
37            {
38                found = true;
39                q = j;
40                break;
41            }
42        }
43        if (found)
44        {
45            p = i;
46            break;
47        }
48    }
49
50    if (found)
51    {
52
53        i = p;
54        j = q;
55        while (i - p < wordSize && q - j < wordSize)
56        {
57            puzzleHash[i][j] = (wordNumber % 7) + 1;
58            i++;
59            j--;
60        }
61    }
62 }

```

D.Input dan Output

1. tc1_small.txt

```
ZUBVVVTBLTFYK
VLIPFEWKGYFA
PERXLCWTHQFI
FPKWSTMJPDHR
BSCDPFBGKBNO
ZQLZECXQAKCC
RFSDDHJJQSVGR
NDYUOKMNNXOF
UJNKXQIGMIVY
LACKEKUZXSVE
ZHPQJMOBWQNR
NSOQLZBBJUCC
KDIHFHPERCQVQ
PIZEQDOWTYVD

PFEWKG
BKGBFPD
OCRFYER
IDSHAJ
ZLRWPCJN
VQJBUQO
MUGNVC
PGXJOK

Execution Time      : 0.000998 seconds
Total Comparision  : 10442
```

2. tc2_small.txt

```
PFDCAADZPSOYMAPRB
ZDIOCNXOPIMANNCZ
ERMOYENIJFVAIYMZ
VZOUNFOMKRUMVTTH
LWIWKYZZFDHOVLQN
WOVQSISXKREJGAKF
QJLAJBJRFINEOBIJ
WKQLQJRJZYZSCLJW
MJIOWEPCFCCGIEPV
FXXYCCIQYIMEGGBS
FMUZXITDDJOUNCFS
XXRGRREETCTNPBMP
XSKWTGCPBAXBTCBW
AYQTKXPAYSIGGQJK
CNSBJWPBBVWBGDSO
JQPPJBQAWTPPQFZP
JKEEHZSKBIXWZLBL
KBJHPNOBQWVNGKVE

JRFINEO
GBWVBBP
SGEUNGBB
XJKJOWZ
RWKWQKQ
LVAMO
QBYAT
PWJEJ

Input your file name: tc2_small.txt

PFDCAADZPSOYMAPRB
ZDIOCNXOPIMANNCZ
ERMOYENIJFVAIYMZ
VZOUNFOMKRUMVTTH
LWIWKYZZFDHOVLQN
WOVQSISXKREJGAKF
QJLAJBJRFINEOBIJ
WKQLQJRJZYZSCLJW
MJIOWEPCFCCGIEPV
FXXYCCIQYIMEGGBS
FMUZXITDDJOUNCFS
XXRGRREETCTNPBMP
XSKWTGCPBAXBTCBW
AYQTKXPAYSIGGQJK
CNSBJWPBBVWBGDSO
JQPPJBQAWTPPQFZP
JKEEHZSKBIXWZLBL
KBJHPNOBQWVNGKVE

Execution Time      : 0.000999 seconds
Total Comparision  : 18159
```

3. tc3_small.txt

```
S P I H M H P U P P Y L O V E L F
C H E R I S H B S I H P M E M V W
Y E L E N O R E G L Y T U R T O B
A M T E V L E V E U L B S I I A L
D L H I T Q A N G M Y N A F B P P
T Y E D K R P Y T A I D P Y B N G
L I T G N Z M H D V O X L S A Q C
O M W X N M R R E W O O H I R S R
H Z I W Y A E U N A V O R R E O Y
Y V S U L T N I N E T E R J T U I
P A T V S N O E F A L W X G I L N
A V T E S W D O E L W Q A G H M G
N X Y S N S S Y A T R A M V W A W
A I M A R A C V Y M Y D Y A E N W
```

```
BABYLOVE
BLUEVELVET
CARAMIA
CHERISH
CRYING
DOWNTOWN
ELENORE
FIRE
GROOVIN
HEATWAVE
HOLIDAY
MEMPHIS
MYGUY
PUPPYLOVE
RUNAWAY
SOULMAN
STAY
TEENANGEL
THETWIST
VALLERI
WHITERABBIT
WINDY
YESTERDAY
```

Input your file name: tc3_small.txt

```
S P I H M H P U P P Y L O V E L F
C H E R I S H B S I H P M E M V W
Y E L E N O R E G L Y T U R T O B
A M T E V L E V E U L B S I I A L
D L H I T Q A N G M Y N A F B P P
T Y E D K R P Y T A I D P Y B N G
L I T G N Z M H D V O X L S A Q C
O M W X N M R R E W O O H I R S R
H Z I W Y A E U N A V O R R E O Y
Y V S U L T N I N E T E R J T U I
P A T V S N O E F A L W X G I L N
A V T E S W D O E L W Q A G H M G
N X Y S N S S Y A T R A M V W A W
A I M A R A C V Y M Y D Y A E N W
```

Execution Time : 0.001995 seconds
Total Comparision : 43178

4. tc4_med.txt

```
D Z N F B F F F F Y P T H R U X Y B R I N Y
V G C B X D X I R R E G U L A R J Y J I R R
X J L R G M N U U T M V O H J B X T J H F E
I B W G K D W P U S O L I D F L T L Q U E C
T I N E R T N E S S Z E S F W B T D A D C O
O O N F C H B L K D G C V N G Z X E D H D V
S M H I N V N A B V O N C N Y S M B L X I E
O J W R D N C N H O U E J W T M J I L P A R
J Q O M V E A I C B F R G K S L D R O W R U
F Y E N U J I F D P A E Y Y N K A C G X B P
P R E S C R I B E D D H A R A N X S Q F K Z
C K N I I Q G H N X M D I U L R L N P K X D
R U R U G U A Y D G C A C J U R V I M L Q B
K Q C V J W N Z K S Q J V R T W B I J G V I
U H O E D I K H E O G I F E A R S R Q B I B
Y I H E R E W O H S J H Y P P G G Q T P H L
S I S A S Q D E S U M A S J S N W P M A R I
O J O T T O T A L L Y A A O T H U V S H O C
K C K C R E R R I X T O G G N I K C A S C A
T S E N O H S I D G T Z M F U P J M Z J Y L

ADHERENCE
AMUSE
BIBLICAL
BRAID
BRINY
DISHONEST
FINAL
INERTNESS
INSCRIBED
IRREGULAR
JUNE
PERJURY
PRESCRIBED
RECOVER
SACKING
SHOWER
SOLID
SPATULA
```

Input your file name: tc4_med.txt

```
D Z N F B F F F F Y P T H R U X Y B R I N Y
V G C B X D X I R R E G U L A R J Y J I R R
X J L R G M N U U T M V O H J B X T J H F E
I B W G K D W P U S O L I D F L T L Q U E C
T I N E R T N E S S Z E S F W B T D A D C O
O O N F C H B L K D G C V N G Z X E D H D V
S M H I N V N A B V O N C N Y S M B L X I E
O J W R D N C N H O U E J W T M J I L P A R
J Q O M V E A I C B F R G K S L D R O W R U
F Y E N U J I F D P A E Y Y N K A C G X B P
P R E S C R I B E D D H A R A N X S Q F K Z
C K N I I Q G H N X M D I U L R L N P K X D
R U R U G U A Y D G C A C J U R V I M L Q B
K Q C V J W N Z K S Q J V R T W B I J G V I
U H O E D I K H E O G I F E A R S R Q B I B
Y I H E R E W O H S J H Y P P G G Q T P H L
S I S A S Q D E S U M A S J S N W P M A R I
O J O T T O T A L L Y A A O T H U V S H O C
K C K C R E R R I X T O G G N I K C A S C A
T S E N O H S I D G T Z M F U P J M Z J Y L
```

Execution Time : 0.001998 seconds
Total Comparision : 62420

5. tc5_med.txt

```
V H S T R I P E S O R Q J Q Z S X Z R A P V
I T N L A N O O M X E I H L A M M A M S J I
O Z X F G H A W D X T G W O G Y L T A E L S
I R E E E L E I X Z N V E L G N U J S E P Y
C K N W S C H G R R U I C Z N Q O E S O H W
A Z I B U W E M E E H Q L R Q V N W Q W Z Y
R A L B M W W R T O B C A O G I N A I S A Q
N Z E G A R V Z K M O I W S H M A L A Y A N
I U F V T O O W A N D D S C C C Z J T N D E
V S W N R O G T S U J G O U R D T H E O J D
O S D A A Z V E A U E D G Q D O I N X A V T
R P N I N C R S U D N I S T J Z D Q A F M D
E G I N V V N E R I E J B C B A P O E R U N
E R E T A E T A E M C R B S N F W Z X I A C
T F B T L A G N E B X V P G T M I I Q C W H
Q N I O I I K N D L I W E E E J D K B A V G
D O L I A T A T U R Z R E T A E N A M N V V
N C Y U J B A S P W E T Z I X V A Q A P A C
J V B W F B I L G D H X B K H Q T X V N M P
K F F T S G N I R A O R J E Z I P J F M D H
```

AFRICAN
ASIAN
BENGAL
CLAWS
FELINE
HUNTER
JUNGLE
ORANGE
PREDATOR
ROARING
SIBERIAN
STRIPES
TAIL
TEETH
WILD
|

Input your file name: tc5_med.txt

```
V H S T R I P E S O R Q J Q Z S X Z R A P V
I T N L A N O O M X E I H L A M M A M S J I
O Z X F G H A W D X T G W O G Y L T A E L S
I R E E E L E I X Z N V E L G N U J S E P Y
C K N W S C H G R R U I C Z N Q O E S O H W
A Z I B U W E M E E H Q L R Q V N W Q W Z Y
R A L B M W W R T O B C A O G I N A I S A Q
N Z E G A R V Z K M O I W S H M A L A Y A N
I U F V T O O W A N D D S C C C Z J T N D E
V S W N R O G T S U J G O U R D T H E O J D
O S D A A Z V E A U E D G Q D O I N X A V T
R P N I N C R S U D N I S T J Z D Q A F M D
E G I N V V N E R I E J B C B A P O E R U N
E R E T A E T A E M C R B S N F W Z X I A C
T F B T L A G N E B X V P G T M I I Q C W H
Q N I O I I K N D L I W E E E J D K B A V G
D O L I A T A T U R Z R E T A E N A M N V V
N C Y U J B A S P W E T Z I X V A Q A P A C
J V B W F B I L G D H X B K H Q T X V N M P
K F F T S G N I R A O R J E Z I P J F M D H
```

Execution Time : 0.001003 seconds
Total Comparision : 51699

6. tc6_med.txt

```
I M B E O Q R G V Y W Z C S Z I J W N Z
J W M Q E G A X X J Q Z X K M I I X Q A
H A K M Z K K W W A J K K B P R Q G R Z
I Y E B G I F M R F U D R E F O R L N L
D A P C W S T P L D R T O F L D J G R D
Q J B G J S E D R F T P C S A T V G O P
Y V J X Q Q E S L O M J Z K O P D S S U
W D Q Y T B Z Z Q L B E Z B H T V V Y C
V Z C B S F E E I F H F J S Q Y I Z A M
V A O T W L D W M Z J F K O S C U I Q K
Z A N F G X F Z V S Y K W Q P G B O U H
X G X Y U V J U F A N Y N K I R O R U B
M V D A H Z K C Q S N L P M M Q M N F Y
F P T V O F J C V B O D O J W T E I I K
K E T X R H V U O H M P V X F R U G R B
N D W B T H H H U C P M Y R T B Q I O I
X Q A Q U H U I N X U D X K M Y F A H Y
K Z K M C D U X H R C N Z J N A G M J S
C X T Y E M Q G M O W K O Y G J Q R H P
C R D T H Y C X J Z J X S N R R T Z R R
B I M R I R I C F T R R I X F D M K P L
S T H K M T I N W O N V U A Z D E M N D

SEDRFTPC
J0D0BVCJF
TSWGUHO
UIVDVJR
BWSEZIZ
TX0L
RHMUIU
XYDC0J
```

Input your file name: tc6_med.txt

```
I M B E O Q R G V Y W Z C S Z I J W N Z
J W M Q E G A X X J Q Z X K M I I X Q A
H A K M Z K K W W A J K K B P R Q G R Z
I Y E B G I F M R F U D R E F O R L N L
D A P C W S T P L D R T O F L D J G R D
Q J B G J S E D R F T P C S A T V G O P
Y V J X Q Q E S L O M J Z K O P D S S U
W D Q Y T B Z Z Q L B E Z B H T V V Y C
V Z C B S F E E I F H F J S Q Y I Z A M
V A O T W L D W M Z J F K O S C U I Q K
Z A N F G X F Z V S Y K W Q P G B O U H
X G X Y U V J U F A N Y N K I R O R U B
M V D A H Z K C Q S N L P M M Q M N F Y
F P T V O F J C V B O D O J W T E I I K
K E T X R H V U O H M P V X F R U G R B
N D W B T H H H U C P M Y R T B Q I O I
X Q A Q U H U I N X U D X K M Y F A H Y
K Z K M C D U X H R C N Z J N A G M J S
C X T Y E M Q G M O W K O Y G J Q R H P
C R D T H Y C X J Z J X S N R R T Z R R
B I M R I R I C F T R R I X F D M K P L
S T H K M T I N W O N V U A Z D E M N D
```

Execution Time : 0.000985 seconds
Total Comparision : 27543

7. tc7_large.txt

ILDRTPHGNWSEAILLPJTTHQJSSUBJECTIVEIK	ACCEPTABLE
YWNVDVJSPPLUFEYEJWGOUOELIHZBYPXANLO	BOORISHNESS
FUBZJJEVPJAKKVTVJRRLNXDKRVPJSEIROTS	BRITTLE
BKTLIDCIXDKSFGERONELWERMKUMRNOEEWQ	CAUTIOUS
GSOKNIWEVRULWALZNURJYTAWCBHGGAGIJY	CREATURE
XZOKSMRINDNZSEJGHBZRZACWJLJNZIZXWX	DIVIDING
PGJLUKEUDVLBIUDODCIVPIWBUICTIONMYVY	EARTHQUAKE
WFWODOPATTIEPTYUPOJPQHCLDDCRRVCXIFW	EYEFUL
BARAIAOAHXVRTMNLIXYXAOEFEIHRWETCLP	FIESTA
XJUTCNMNEOYSIKMTVNMWRISWPHSPIHPGECQ	FIREWOOD
ABZRAEERBYISDPUGLIWSSNUHTITHTTLAM	GABLE
TJAYDMWCQWEIAIFUBMAJSASRMCLSIHST	HINDRANCE
SGERULKLTTHXOYINHWDBGEWPPJPGOCPOYGID	IMAGO
EEOUAFZMECSFZETGINWOMYRROCTEPNHNAZ	KEYNOTE
ISGSUOITUACQTLUFECARGSIDERSTXMOOSOK	LAVENDER
FEWNMFZRUDQBRWSWDDSALEPJSLSVMHBBUYID	MESSIAH
ERTWIMEEOYQZESDVCRCRSHWSEVVAFWRTQMO	NEWSBOY
AOPARKQQLLKVSQNVFARCHERYVWPLPYHZPO	ORPHAN
PPEQCEILOYSPTRAHUFMGNZXMTAREABDZW	PILOTS
KMRESQATBBLNRYEAYGHHPEELSZINCXEBE	RESTRICTIVE
EKZGXIQPEYIKITADWLSDHIEWMSCCOCCELRL	SOAP
MKKNTGHCXROHCGCENIWNBBUIIREXOPSEFI	SPATIALLY
HKEAFPWEVKNDTLQERETNRDNNSPTNYILRQF	STORIES
OQPKRFIOGIGMIPBOWWVIGLGXTYSRAOIUBM	TONER
PSXIPOYNVMSDVKOGYYTAGKEAKOVAVGLSJJI	UNLEARN
ZZAZGYATHGPAEBFXFTSYLTBKRWUEYRELEC	VOLE
GZRAKBGAALVGI BDBLZNODLBTTPLIJDFFXX	WRONGDOING
DQMEKAUQHTRAENEEUSTBEWEPVOCNPTCQJJP	
TIKQVSTNEMHCATTAAABOSMDNBUZAUQPUBAI	
PQJ TZUZVMRETSISPETSWHINDRANCEODNPY	
QHNLPWMETMSEIZUREUTEVWQKGBQAVYTYIW	
NAHPRGABLEBGESKORKNANZQNETONYEKVW	

ILDRTPHGNWSEAILLPJTTHQJSSUBJECTIVEIK
YWNVDVJSPPLUFEYEJWGOUOELIHZBYPXANLO
FUBZJJEVPJAKKVTVJRRLNXDKRVPJSEIROTS
BKTLIDCIXDKSFGERONELWERMKUMRNOEEWQ
GSOKNIWEVRULWALZNURJYTAWCBHGGAGIJY
XZOKSMRINDNZSEJGHBZRZACWJLJNZIZXWX
BARAIAOAHXVRTMNLIXYXAOEFEIHRWETCLP
XJUTCNMNEOYSIKMTVNMWRISWPHSPIHPGECQ
ABZRAEERBYISDPUGLIWSSNUHTITHTTLAM
TJAYDMWCQWEIAIFUBMAJSASRMCLSIHST
SGERULKLTTHXOYINHWDBGEWPPJPGOCPOYGID
EEOUAFZMECSFZETGINWOMYRROCTEPNHNAZ
ISGSUOITUACQTLUFECARGSIDERSTXMOOSOK
FEWNMFZRUDQBRWSWDDSALEPJSLSVMHBBUYID
ERTWIMEEOYQZESDVCRCRSHWSEVVAFWRTQMO
AOPARKQQLLKVSQNVFARCHERYVWPLPYHZPO
PPEQCEILOYSPTRAHUFMGNZXMTAREABDZW
KMRESQATBBLNRYEAYGHHPEELSZINCXEBE
EKZGXIQPEYIKITADWLSDHIEWMSCCOCCELRL
MKKNTGHCXROHCGCENIWNBBUIIREXOPSEFI
HKEAFPWEVKNDTLQERETNRDNNSPTNYILRQF
OQPKRFIOGIGMIPBOWWVIGLGXTYSRAOIUBM
PSXIPOYNVMSDVKOGYYTAGKEAKOVAVGLSJJI
ZZAZGYATHGPAEBFXFTSYLTBKRWUEYRELEC
GZRAKBGAALVGI BDBLZNODLBTTPLIJDFFXX
DQMEKAUQHTRAENEEUSTBEWEPVOCNPTCQJJP
TIKQVSTNEMHCATTAAABOSMDNBUZAUQPUBAI
PQJ TZUZVMRETSISPETSWHINDRANCEODNPY
QHNLPWMETMSEIZUREUTEVWQKGBQAVYTYIW
NAHPRGABLEBGESKORKNANZQNETONYEKVW

Execution Time : 0.005992 seconds
Total Comparision : 232477

8. tc8_large.txt

```

AVPVNBGJGHLUILQCHUBCVZPNKIBJUVOLRKFV
GOTEYZXEJJOXYKECPMELBAVOMHIGGEZZYCOO
YQNLVCHMNLPHPRRTJFVYUWIFYUHBROJSRRRQ
HXYFSGCFHIZFZDETAROPAVERCPHZDQYFAEE
ROBYQCPYFWLPBUZVTRERGFYGA VNGEVVPGVM
MSZPQHJGPGYGCAMRZTBKWCNGNITEEHSNLSNER
FTQQHXKWDWLBNEVKBPUPHEEQDRSKHCUSDRF
HSAFUSSYJNJSDIMPLBIXIEXKOA BZTYTMDRJ
TZAPJRBPZTLNUBMQZMKISGCI BAMS EDIWELW
KWTXVRJZMYEFYODPXJWL OIAGUT FPMCKKVEK
YFZKVN PEN CUMBER INGE B POLSH IO MOYXYADI
BEAOTMNQKMZQZUNCCNWHMHBRLMXNSKPLYDM
GCEGNCA DAPTIVE OVIUH Z OQIHSD ILEXCBDIR
SWHHQMEGBTTGLGEUNDE SCWJJPCJMLWYBTFA
OBDWOYWGEAZGKGOKCFUBELACI GOLOMSOCPX
NCBH FJYLMKAUAPYDDSP LDBWL YKSLHKDDKOB
TDWKNO TAAGZADORERRRRHSHFDCCOTW LPHGZK
WNEEIHRHEISBTSLYNNEJUJUCRSKUCMFIUGC
AFDLHDKNFSGEZPJUSSCAMOHDOUEKBYDJUDB
FAFWLASDITYFAATEOJLLLFUNESLCTNQIPEI
LSZOREACL CDGKRXIIFUCISJGLPDCIQBWHAA
GUJSPZPOYEAXILIAVWDSAITSEGNOLLHAIVF
TTBGZUIMZKDT SJWNFDIDWQJSSFMQNDLKL
NAQTDNXYIUINESMEGONRIAKETGRBSFTELWZ
RTRKCYULS OLOT DGO PRGVC R PGLDYKKRJPRIC
SSPLSGBOU TWFGSSTGPATRIARCHYCORSDPTQ
PUONXODS XZRDAECNEREFNOCUHWAFMLNEJYM
ITMQMJJC NFFFLG CCHXVZYSABFRVZWABZN
HSNGQVWYBHIQWIDWCZWJLGGRWEOXKCOGSTS
VGURVHSJWFPDUFVUQWDSHTDOMZWEGIOCWDX
LIZDDPGNNYGIKILZGQXAVYRFCOCSIDHCIFT
KTKCBKQQVS MANHOLEUMBCHVXQMJBXORJMK
TXOGTQYNOITISOPNPMEATLESSWVR LGDSCXM
CAOEALTRXP NPHZYINLANFSUPREMACYKUCEK
WNSJDNOYE BUDGETING ZZBXBEAC ROZKWIJEV
ADAPTIVE
BUDGETING
COSMOLOGICAL
DECOMPOSE
EFFORT
FORNICATED
ICONIC
LUDICROUSLY
MANHOLE
NUTRITIOUS
OWNS
PATRIARCHY
REDNECK
SCHOLAR
STATUS
SUPREMACY
THROWBACK
UPHILL
VERBATIM
WHOLESOME
HXYFSGCFHIZFZDETAROPAVERCPHZDQYFAEE
ROBYQCPYFWLPBUZVTRERGFYGA VNGEVVPGVM
MSZPQHJGPGYGCAM RZTBKWCNGNITEEHSNLSNER
FTQQHXKWDWLBNEVKBPUPHEEQDRSKHCUSDRF
HSAFUSSYJNJSDIMPLBIXIEXKOA BZTYTMDRJ
TZAPJRBPZTLNUBMQZMKISGCI BAMS EDIWELW
KWTXVRJZMYEFYODPXJWL OIAGUT FPMCKKVEK
YFZKVN PEN CUMBER INGE B POLSH IO MOYXYADI
BEAOTMNQKMZQZUNCCNWHMHBRLMXNSKPLYDM
GCEGNCA DAPTIVE OVIUH Z OQIHSD ILEXCBDIR
SWHHQMEGBTTGLGEUNDE SCWJJPCJMLWYBTFA
OBDWOYWGEAZGKGOKCFUBELACI GOLOMSOCPX
NCBH FJYLMKAUAPYDDSP LDBWL YKSLHKDDKOB
TDWKNO TAAGZADORERRRRHSHFDCCOTW LPHGZK
WNEEIHRHEISBTSLYNNEJUJUCRSKUCMFIUGC
AFDLHDKNFSGEZPJUSSCAMOHDOUEKBYDJUDB
FAFWLASDITYFAATEOJLLLFUNESLCTNQIPEI
LSZOREACL CDGKRXIIFUCISJGLPDCIQBWHAA
GUJSPZPOYEAXILIAVWDSAITSEGNOLLHAIVF
TTBGZUIMZKDT SJWNFDIDWQJSSFMQNDLKL
NAQTDNXYIUINESMEGONRIAKETGRBSFTELWZ
RTRKCYULS OLOT DGO PRGVC R PGLDYKKRJPRIC
SSPLSGBOU TWFGSSTGPATRIARCHYCORSDPTQ
PUONXODS XZRDAECNEREFNOCUHWAFMLNEJYM
ITMQMJJC NFFFLG CCHXVZYSABFRVZWABZN
HSNGQVWYBHIQWIDWCZWJLGGRWEOXKCOGSTS
VGURVHSJWFPDUFVUQWDSHTDOMZWEGIOCWDX
LIZDDPGNNYGIKILZGQXAVYRFCOCSIDHCIFT
KTKCBKQQVS MANHOLEUMBCHVXQMJBXORJMK
TXOGTQYNOITISOPNPMEATLESSWVR LGDSCXM
CAOEALTRXP NPHZYINLANFSUPREMACYKUCEK
WNSJDNOYE BUDGETING ZZBXBEAC ROZKWIJEV
Execution Time : 0.004998 seconds
Total Comparision : 194036

```

9. tc9_large.txt

```

X Y G H V U O S G T I Q Y C J Z S J P A A E V R L U C J V G K Z T S D T S X D A
B Q I S U U J A I O M U Z W F G Y E U Y U G H H I C T G Z I O D A Z I N U P V Q
J D Y M X C J K S V T C C X Z X D A H A M S C N D H U E R M P Y C J W M Q L K U
K Y A M O S F D I D Q B W M W W W N Q K C S S R E W P T Q E U B F G H H W I D C
M A F J Q K B U H T E I N V X E Q N F A I O E O V H A H W X M H I Z B M Y I Q A
A Y G G B K O S P R V S G V E N K W R F S E N W M G S F I B X E L T S M X L M I
F M P Q U A O K S K S J D T D M D D L W A U S J A N X L Y R M F A X V E R A W V
U D B G F N C I D D C Q E V U U T L A A I R A S O R P B T Z C U I E Q U N L B C
R W I E S X R A H H B R R J D H C D U F Z W B X D I O R M T G L Z C T V K P B N
E V R V R T V P X F I B K X F S V I T X G R Y K L E K E D C Q H B T N A E X K A
H A U A W X T Z D E S I A N I X A R E A M T Q J Y K T S T U Y F E N T I E E L K
B V J L H C X R L X U M E X Z M H O B S T Q R D B O B O G G I J K N F U D O O T
E H F W S R S G G W D V S D D Z K L N A J P N G Y U R E E Q Y K Q R J Y X A I M
O A A Y H K O O J N P L O N I Z B N W U Y H M V Z Q H O I G L P S X M Q U L I J
V N R Y K Y A N F E I Z R Y H V Y O W A B Z C Z Q P V J L J S Q K N W A T R C C
U G U V T L T Z Y J F P C T I N Q S T G U X F N I J C L Q R F B X D T B A N N X
I B B U J Z W V Y O W A U R A S C X U X E W U K Y G F U S R H N M Y L W O L F A
C A N P J O P Z T H I Y S E T N P Y P I T E C V K G D R A A P H O X C Z G Z K J
E A H S L C V Y M I B M H J N Z O O J Z X O I X G Q S Z T T N V W O T N F R F C
W L G T C G X R D D Y W I R T K M U H J L Z E N M W N X J C Q O A U A B S Z K D
C S W T P M J D N D Z L V Y C J X F P U J X P I Z B N Z Z R F G I U A U K X C D
Z R R E B S C X M O B X U Z A Y S J B J W Q T A G A Q I O X D I G D B O N W P J
G P A F B W K B E O Z R J K R H T B R L Y K F G I B H M C R O G Z F E I O B D K
M V A F Z J Q E C V K G Z W S U O O G O K S M D J B Q V Z Y N V W H H Z W N Z S
O Y H T L E I N E P Q C Y L A P I Q G D K M S K H O R U S I K Z X Y B Y G N R P
L X T U T U F G D U I M N Y M Y A Y U A U N X I D T A D N E H G J F T N V V B S
O P H B T M R A T D S B Z P N A M V P J P B K F T G L C S S L E G Z I L T E T J
P K E P W A U P M J N F O A F W I L V L F L H P K Y U M P I O X A L S Q J L U B
K R Q J O S O O A B O I G Y V O H F J C Z W S G M X E G A N Y U G Y S S L J H J
A W F B C F N O X B P D M U N Z X W T P N K T D K Q C I P B A N K Z W V B K I Z
H P U O D I E B A K P V A N J J Z O O R E H A A F B R Y S J A X D F F J N V S I
S T G D A Z M O X U Z U M J Q D L C O Q J J L X Z S Y O C I H E X M D T C D J Q
I X O V S K D O W Q D O V I K Z K T I B Z A K T D Q A C X G T J P B U M B H E Q
H I Z I L P H C C O K G G N Z N Q N G E K X L B H U A W N U Y I A F Y G T R I C
N O E L L E F R B I Q D U G G M G B X Q I O F V X F S B Y X F R V N V S H T A Y
G W B A S Q I M W I I K M U F Z U Y U B V L W X B I S I L B K K Q K P E U I M I
Y Y F E B W P R S K U F R S I U E Z A Q M P S F S R L L S A I Q A T Z J N I W K
V P V W K I X X J Z R Z D N Z N Q E Y X Y L G M E W L C R L V N X T Q F C D C L
S L U N B U K U L R V Y U O W N X R T V P R T Y F Z N A W L F T Y B A N G Y H V

```

Input your file name: tc9_large.txt

```

X Y G H V U O S G T I Q Y C J Z S J P A A E V R L U C J V G K Z T S D T S X D A
B Q I S U U J A I O M U Z W F G Y E U Y U G H H I C T G Z I O D A Z I N U P V Q
J D Y M X C J K S V T C C X Z X D A H A M S C N D H U E R M P Y C J W M Q L K U
K Y A M O S F D I D Q B W M W W W N Q K C S S R E W P T Q E U B F G H H W I D C
M A F J Q K B U H T E I N V X E Q N F A I O E O V H A H W X M H I Z B M Y I Q A
A Y G G B K O S P R V S G V E N K W R F S E N W M G S F I B X E L T S M X L M I
F M P Q U A O K S K S J D T D M D D L W A U S J A N X L Y R M F A X V E R A W V
U D B G F N C I D D C Q E V U U T L A A I R A S O R P B T Z C U I E Q U N L B C
R W I E S X R A H H B R R J D H C D U F Z W B X D I O R M T G L Z C T V K P B N
E V R V R T V P X F I B K X F S V I T X G R Y K L E K E D C Q H B T N A E X K A
H A U A W X T Z D E S I A N I X A R E A M T Q J Y K T S T U Y F E N T I E E L K
B V J L H C X R L X U M E X Z M H O B S T Q R D B O B O G G I J K N F U D O O T
E H F W S R S G G W D V S D D Z K L N A J P N G Y U R E E Q Y K Q R J Y X A I M
O A A Y H K O O J N P L O N I Z B N W U Y H M V Z Q H O I G L P S X M Q U L I J
V N R Y K Y A N F E I Z R Y H V Y O W A B Z C Z Q P V J L J S Q K N W A T R C C
U G U V T L T Z Y J F P C T I N Q S T G U X F N I J C L Q R F B X D T B A N N X
I B B U J Z W V Y O W A U R A S C X U X E W U K Y G F U S R H N M Y L W O L F A
C A N P J O P Z T H I Y S E T N P Y P I T E C V K G D R A A P H O X C Z G Z K J
E A H S L C V Y M I B M H J N Z O O J Z X O I X G Q S Z T T N V W O T N F R F C
W L G T C G X R D D Y W I R T K M U H J L Z E N M W N X J C Q O A U A B S Z K D
C S W T P M J D N D Z L V Y C J X F P U J X P I Z B N Z Z R F G I U A U K X C D
Z R R E B S C X M O B X U Z A Y S J B J W Q T A G A Q I O X D I G D B O N W P J
G P A F B W K B E O Z R J K R H T B R L Y K F G I B H M C R O G Z F E I O B D K
M V A F Z J Q E C V K G Z W S U O O G O K S M D J B Q V Z Y N V W H H Z W N Z S
O Y H T L E I N E P Q C Y L A P I Q G D K M S K H O R U S I K Z X Y B Y G N R P
L X T U T U F G D U I M N Y M Y A Y U A U N X I D T A D N E H G J F T N V V B S
O P H B T M R A T D S B Z P N A M V P J P B K F T G L C S S L E G Z I L T E T J
P K E P W A U P M J N F O A F W I L V L F L H P K Y U M P I O X A L S Q J L U B
K R Q J O S O O A B O I G Y V O H F J C Z W S G M X E G A N Y U G Y S S L J H J
A W F B C F N O X B P D M U N Z X W T P N K T D K Q C I P B A N K Z W V B K I Z
H P U O D I E B A K P V A N J J Z O O R E H A A F B R Y S J A X D F F J N V S I
S T G D A Z M O X U Z U M J Q D L C O Q J J L X Z S Y O C I H E X M D T C D J Q
I X O V S K D O W Q D O V I K Z K T I B Z A K T D Q A C X G T J P B U M B H E Q
H I Z I L P H C C O K G G N Z N Q N G E K X L B H U A W N U Y I A F Y G T R I C
N O E L L E F R B I Q D U G G M G B X Q I O F V X F S B Y X F R V N V S H T A Y
G W B A S Q I M W I I K M U F Z U Y U B V L W X B I S I L B K K Q K P E U I M I
Y Y F E B W P R S K U F R S I U E Z A Q M P S F S R L L S A I Q A T Z J N I W K
V P V W K I X X J Z R Z D N Z N Q E Y X Y L G M E W L C R L V N X T Q F C D C L
S L U N B U K U L R V Y U O W N X R T V P R T Y F Z N A W L F T Y B A N G Y H V
Q F H T Z O F M P A X C Q Q U A Y B V V H S C F E L R B A J W P I X H T G F X A

```

Execution Time : 0.008 seconds
Total Comparision : 322162

E. Alamat Drive

Kode bisa dilihat pada repository Github di [tautan berikut](#) setelah 26 Januari 2022 pukul 12.30 WIB.

F. Tabel *Checklist*

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan (<i>no syntax error</i>)	√	
2. Program berhasil <i>running</i>	√	
3. Program dapat membaca file masukan dan menuliskan luaran	√	
4. Program berhasil menemukan semua kata di puzzle.	√	