

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

```

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        for (i = p; p - i < wordSize; i--)
49        {
50            puzzleHash[i][q] = (wordNumber % 7) + 1;
51        }
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

```
ZUBVV TBLTFYK
VLIPFEWKGYFA
PERXLCWTHQFI
FPKWSTMJPDHR
BSCDPFBGKBN O
ZQLZECXQAKCC
RFS DHJJQSVGR
NDYUOKMNNXOF
UJNKXQIGMIVY
LACKEKUZX SVE
ZHPQJMOBWQNR
NSOQLZBBJUCC
KDI FHPERCQVQ
PIZEQDOWTYVD

PFEWKG
BKGBFPD
OCR FYER
IDSHAJ
ZLRWPCJN
VQJBUQO
MUGNVC
PGXJOK

Input your file name: tc1_small.txt

ZUBVV TBLTFYK
VLIPFEWKGYFA
PERXLCWTHQFI
FPKWSTMJPDHR
BSCDPFBGKBN O
ZQLZECXQAKCC
RFS DHJJQSVGR
NDYUOKMNNXOF
UJNKXQIGMIVY
LACKEKUZX SVE
ZHPQJMOBWQNR
NSOQLZBBJUCC
KDI FHPERCQVQ
PIZEQDOWTYVD

Execution Time: 0 seconds
```

2. tc2_small.txt

```
PFD CADZPS OYMA PRB
ZDIOCNXOPI MANNCZ
ERMOYENIJFVAIYMZ
VZOUNFOMKRUMVTTH
LWIWKYZZFDHOVLQN
WQVQSISXKREJGAKF
QJLAJBJR FINEOBIJ
WKQLQJRJZYZSCLJW
MJIOWEP CFCGIEPV
FXXYCCI QYIMEG OBS
FMUZXITDDJOUNCFS
XXRGRREETCTNPBMP
XSKWTGCPBAXBTCBW
AYQTKXPAYSIGGQJK
CNSBJWPBBVWBGD SO
JQPPJBQAWTPPQFZP
JKEEHZSKBIXWZLBL
KBJHPNOBQWVNGKVE

JRFINEO
GBWVBBP
SGEUNBGB
XJKJOWZ
RWKWQKQ
LVAMO
QBYAT
PWJEJ

Input your file name: tc2_small.txt

PFD CADZPS OYMA PRB
ZDIOCNXOPI MANNCZ
ERMOYENIJFVAIYMZ
VZOUNFOMKRUMVTTH
LWIWKYZZFDHOVLQN
WQVQSISXKREJGAKF
QJLAJBJR FINEOBIJ
WKQLQJRJZYZSCLJW
MJIOWEP CFCGIEPV
FXXYCCI QYIMEG OBS
FMUZXITDDJOUNCFS
XXRGRREETCTNPBMP
XSKWTGCPBAXBTCBW
AYQTKXPAYSIGGQJK
CNSBJWPBBVWBGD SO
JQPPJBQAWTPPQFZP
JKEEHZSKBIXWZLBL
KBJHPNOBQWVNGKVE

Execution Time: 0.000986 seconds
```

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: 0.000998 seconds

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

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: 0.000979 seconds
```

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: 0.001 seconds

7. tc7_large.txt

I L D R T P H G N W S E A I L L P J T H Q J S U B J E C T I V E I K	ACCEPTABLE
Y W N V D V J S P L U F E Y E J W G O U O E L I H Z B Y P X A N L O	BOORISHNESS
F U B Z J E V P J A K K V T V J R R N L X D K R V P J S E I R O T S	BRITTLE
B K T L I D C I X D K S F G E R O N E L W E R M K U M R N O E E W Q	CAUTIOUS
G S O K N I W E V R U L W A L Z N U R J Y T A W C B H G G A G I J Y	CREATURE
X Z O K S M M R I N D N Z S E J G H B R Z A C W J L J N Z I Z X W X	DIVIDING
P G J L U K E U D V L B I U D O D C I V P I W B U I C I O N M Y V Y	EARTHQUAKE
W F W D O P A T T I E P T Y U P O J P Q H C L D D C R R V C X I F W	EYEFUL
B A R A I A O A H X V R T M N L I X Y X A O E F E I H R W E T C L P	FIESTA
X J U T C M N E O Y S I K M T V N M W R I S W P H S P I H P G E C Q	FIREWOOD
A B Z R A E E R B Y I S D P P U G L I W S S N U H T I T H T T L A M	GABLE
T J A Y D M W C Q W E I A I F U B M A J S A S R M C L S I I I H S T	HINDRANCE
S G E R U L K L T H X O Y I N H W D B G E W P J P G O C P O Y G I D	IMAGO
E E O U A F Z M E C S F Z E T G I N W O M Y R R O C T E P N H N A Z	KEYNOTE
I S G S U O I T U A C Q T L U F E C A R G S I D E R S T X M O S O K	LAVENDER
F E W N M F Z R U O Q B R W S W D D S A L P J S L V M H B B U Y I D	MESSIAH
E R T W I M E E O Y Q Z E S D V C C R S H W S E V V A F W R T Q M O	NEWSBOY
A O A P A R K Q L L K V S O N V F A R C H E R Y V W P L P Y H Z P O	ORPHAN
P P E Q C E I L O Y S P T R A H U F M G N Z X M J T A R E A B D Z W	PILOTS
K M R E S Q A T B B L N R Y E A Y G H H P E E L S Z I N C N X E B E	RESTRICTIVE
E K Z G X I Q P E Y I K I T A D W L S D H I E W M S C C O C C E L R	SOAP
M K K N T G H X C R O H C G C E N I W N B B U I I R E X O P S E F I	SPATIALLY
H K E A F P W E V K N D T L Q E R E T N R D N N S P T N Y I L R Q F	STORIES
O Q P K R F I O G I G M I P B O W W V I G L G X T Y S R A O I U B M	TONER
P S X I P O Y N V M S D V K O G Y Y T A G K E A K O V A V G L S J I	UNLEARN
Z Z A Z G Y A T H G P A E B F X F T S Y L T B K R W U E Y R E L E C	VOLE
G Z R A K B G A A L V G I B D B L Z N O D L B T T P Q L I J D F X X	WRONGDOING
D Q M E K A U Q H T R A E N E E U S T B E W E P V O C N P T C Q J P	
T I K Q V S T N E M H C A T T A A B O S M D N B U Z A U Q P U B A I	
P Q J T Z U Z V M R E T S I S P E T S W H I N D R A N C E O D N P Y	
Q H N L P W M E T M S E I Z U R E U T E V W Q K G B Q A V Y T Y I W	
N A H P R O G A B L E B G E S K O R K N A N Z Q N E T O N Y E K V W	

Input your file name: tc7_large.txt

```

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

```

Execution Time: 0.006021 seconds

8. tc8_large.txt

AVPVNB JGHLUI LQCHUB CVZPNKIB JUVOLRKFV GOTEYZ XEJOXYKECP MELBAVOMMIGGEZZYCOO YQNLVCHMNLPHPRRTJFVYUWIFYUHBROJSRRRQ HXYFSGCFHIZFZDETAROPAVERC PHZDQYFAEE ROBYQC P YFWLPBUZVTRERGFYGA VNGEVVPGVM MSZPQHJG P YGCA MRZTBKWC GNITEEHSNLNSER FTQQHXKWDWLBNEVKBPUPHEEQDRSKHCUSD RF HSAFUSSYJNJSDIMPLB X IEXKOA BZTYTMDRJ C TZAPJRBPZTLNUBMQZMKISG CIBAMSEDIWELW KWTXVRJZMYEFYODP XJWLOIAGUTFP MCMKVEK YFZKVNPNENCUMBERINGEB POLSHIOM OYXYADI BEAOTM NQKMZQZUNCNWHMHBR LMXNSKPLYDM GCEGNCA DAPTIVEOV IUHZOQIHS DILEXCBDIR SWHHQMEGBTTGLGEUNDE SCWJJPCJMLWYBTFA OBDWOYWG EAZGKGOKCFUBELACTI GLOMSOCPX NCBHFJYLMKAUAPYDDSP LDBWLYKSLHKDDKOB TDWKNOTAAGZADORERRRRHSHFDCCOTWLP HGZK WNEEIHRHEISBTSLYNNEJUJUCRSKUCMFIUGC AFDLHDKNFSGEZPJUSSCAMOH DUOEKBYDJUDB FAFWLASDI TYFAATEOJ LLLFUNESLCTNQIPEI LSZOREACL CDGKRXIIFUCISJGLPDCIQBWHAA GUJSPZPOYEAXILIAVWDSAITSEGNOLLHAI VF TTBGZUIMZKDT SJWNFDIDWQJSSFMQNDLKL LS NAQTDNXYIUI NESMEGONRIAKETGRBSFTELWZ RTRKCYULS OLOT DGO PRGVCRPGLDYKKRJPRIC SSPLSGBOU TWFGSSTG PATRIARCHY CORSDPTQ PUONXODS XZRDAECNEREFNOCUHWA FMLNEJYM ITMQMJJC NFLLFFYLGCCHXVZYSA BFRVZWABZN HSNQGQVWYBHIQWIDWCZWJLGGRWE OXKCOGSTS VGURVHSJWF PDUVFUQWDSHTDOMZWEGIOCWDX LIZDDPGNNYGIKILZGQXAVYRFCCOCSI DHCIFT KTKCBKQQVSMANHOLEUMBCHVXQMJBXORJMK T QXOGTQYN OITISOPNPMEATLESSWVR L GDS CXM CAOEA LTRXP NPHZYINLANFSUPREMACYKUCEK WNSJDNOYE BUDGETINGZZB XBEACROZKWIJEV	ADAPTIVE BUDGETING COSMOLOGICAL DECOMPOSE EFFORT FORNICATED ICONIC LUDICROUSLY MANHOLE NUTRITIOUS OWNS PATRIARCHY REDNECK SCHOLAR STATUS SUPREMACY THROWBACK UPHILL VERBATIM WHOLESOME
--	---

Input your file name: tc8_large.txt

AVPVNB JGHLUI LQCHUB CVZPNKIB JUVOLRKFV GOTEYZ XEJOXYKECP MELBAVOMMIGGEZZYCOO HSAFUSSYJNJSDIMPLB X IEXKOA BZTYTMDRJ C TZAPJRBPZTLNUBMQZMKISG CIBAMSEDIWELW KWTXVRJZMYEFYODP XJWLOIAGUTFP MCMKVEK YFZKVNPNENCUMBERINGEB POLSHIOM OYXYADI BEAOTM NQKMZQZUNCNWHMHBR LMXNSKPLYDM GCEGNCA DAPTIVEOV IUHZOQIHS DILEXCBDIR SWHHQMEGBTTGLGEUNDE SCWJJPCJMLWYBTFA OBDWOYWG EAZGKGOKCFUBELACTI GLOMSOCPX NCBHFJYLMKAUAPYDDSP LDBWLYKSLHKDDKOB TDWKNOTAAGZADORERRRRHSHFDCCOTWLP HGZK WNEEIHRHEISBTSLYNNEJUJUCRSKUCMFIUGC AFDLHDKNFSGEZPJUSSCAMOH DUOEKBYDJUDB FAFWLASDI TYFAATEOJ LLLFUNESLCTNQIPEI LSZOREACL CDGKRXIIFUCISJGLPDCIQBWHAA GUJSPZPOYEAXILIAVWDSAITSEGNOLLHAI VF TTBGZUIMZKDT SJWNFDIDWQJSSFMQNDLKL LS NAQTDNXYIUI NESMEGONRIAKETGRBSFTELWZ RTRKCYULS OLOT DGO PRGVCRPGLDYKKRJPRIC SSPLSGBOU TWFGSSTG PATRIARCHY CORSDPTQ PUONXODS XZRDAECNEREFNOCUHWA FMLNEJYM ITMQMJJC NFLLFFYLGCCHXVZYSA BFRVZWABZN HSNQGQVWYBHIQWIDWCZWJLGGRWE OXKCOGSTS VGURVHSJWF PDUVFUQWDSHTDOMZWEGIOCWDX LIZDDPGNNYGIKILZGQXAVYRFCCOCSI DHCIFT KTKCBKQQVSMANHOLEUMBCHVXQMJBXORJMK T QXOGTQYN OITISOPNPMEATLESSWVR L GDS CXM CAOEA LTRXP NPHZYINLANFSUPREMACYKUCEK WNSJDNOYE BUDGETINGZZB XBEACROZKWIJEV

Execution Time: 0.00499 seconds

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	AMBER
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	AYAKA
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	BARBARA
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	BEIDOU
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	DIONA
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	EULA
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	FISCHL
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	GANYU
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	HUTAO
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	JEAN
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	KEQING
B V J L H C X R L X U M E X Z M M O B S T Q R D B O B O G G I J K N F U D O O T	KLEE
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	KOKOMI
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	LISA
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	NINGGUANG
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	NOELLE
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	QIQI
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	BAAL
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	SAYU
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	ROSARIA
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	SHENHE
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 D X D I G D B O N W P J	SUCROSE
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	XIANGLING
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	YANFEI
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	YOIMIYA
L X T U T U F G D U I M N Y M Y A Y U A U M X I D T A D N E H G J F T N V V B S	YUNJIN
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 M 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 B G N Z N Q N G E K X L B H U A W W 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 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 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 M 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 M 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 D 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 M 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 M 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 B G N Z N Q N G E K X L B H U A W W 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 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 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 M 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: 0.007999 seconds

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.	√	