Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Spring 2019

Homework 1 – Word Hunt Puzzle

Due: 27/02/2019, Wednesday, 21:00

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You can NOT collaborate with your friends and discuss solutions. You have to write down the code on your own. Plagiarism and homework trading will not be tolerated!

Introduction

Word hunts are one of the rather common word puzzles out there. The aim is to find words from a seemingly random letters spread out on a 2D matrix, with each cell on the matrix containing one letter. In this version of the word hunt, players will be able to start from any letter and follow any path by going one cell a time in vertical and horizontal directions. Diagonal movement is not allowed, and the words do not have to be in a straight line. An example matrix and a word marked on it is given in Figure 1.

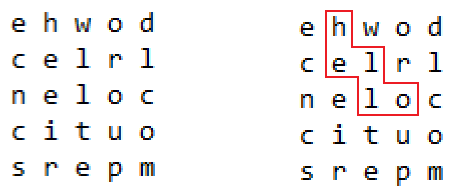

Figure 1: A sample puzzle and a word marked in the puzzle

Inputs, Outputs and Program Flow

First, player will be prompted for the name of the puzzle file. If the file with the given name cannot be opened, player will be prompted again until a file can successfully be opened. You **must** store this puzzle using a **2D matrix** (i.e. a `vector` of `vector` of `char`). Once the puzzle is read from the file, the player will be prompted for the name of a dictionary file. Same as with puzzle file, player will be prompted for the name of the dictionary file until the file is successfully opened. You may store the contents of the dictionary file in a vector if you wish so. This dictionary file contains words in English in alphabetically sorted manner. There is one word in each line of this file; you do not need to check this. All letters are lowercase in this file. We provide this file (words.dat) within the homework package.

Once both files have been opened successfully, current state of the puzzle will be displayed, the game starts. The player's aim is to hunt valid words by concatenating neighboring letters in the matrix. To do so, the player first enters the coordinates of starting location (as *x* and *y* coordinates, both integers) and a path to follow from there. The path will be entered as a string that contains only `'r'`, `'l'`, `'u'` and `'d'` characters, representing "right", "left", "up" and "down". The word starts at the starting location, including the letter in that cell as the first letter of the word, and appending letters in each cell visited according to letters in the path string. Here note that the coordinates start with 0 both vertically and horizontally; i.e. the upper-left corner of the matrix has coordinate of (0,0). An example case is given in Figure 2.

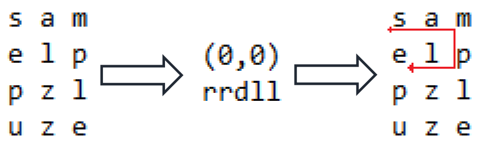

Figure 2: The path followed by inputs `0 0` and `rrdll` on a sample matrix

The minimum length of a word must be 3, which means minimum length of the path string must be 2. If a meaningful word exists along the given path, which will be checked by looking for the word in the dictionary, (1) the player will be awarded points equal to square of number of letters in the word, and (2) the letters in the cells used to create the word will be removed from the matrix and be replaced by underscore ('_') character. An example is given in Figure 3. In this example, first word found by player, in this case "sample", has a length of 6 character, therefore the player will receive 36 points, calculated by taking square of the length of word. Then the player manages to find the word "puzzle" to earn another 36 points, increasing the total score to 72 from 36. <u>This is just an example to help you understand and visualize the flow of the program, how the matrix progressively changes and how the score is calculated.</u>

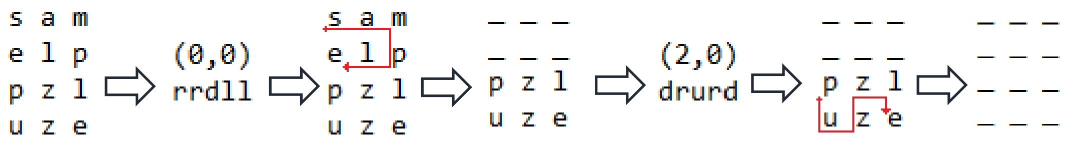

Figure 3: Removal of found words from puzzle

The game will continue to hunt more words. The game will end either (i) when all letters in the matrix are used <u>OR</u> (ii) the player decides to quit by entering -1 as one of the coordinates of the starting position. If the player has managed to use all the letters, an additional message will be displayed to mention that the puzzle has been solved completely. Afterwards, the player will be shown his/her final score (no matter he/she finished puzzle or not) and the program will end.

Input Checks

There are a couple things you need to pay attention to when dealing with inputs. First, the puzzle should be a proper 2D matrix, meaning each line should contain same number of cells (letters). For example, if there are 5 cells in first line, every other line should also contain 5 cells. On top of it, each cell should only contain one lowercase character, meaning both non-letter characters and uppercase letters would be invalid entries.

Rows may contain arbitrary number of white characters (space, tab), you may not make any assumptions. You may assume there will not be any empty lines including before, in between and after the matrix.

Aside from puzzle, the inputs for hunting a word needs to be checked. The starting location should consist of two integer values, and this coordinate should be within the limits of the matrix. If there are n rows and m columns in the matrix, the first coordinate entry must be between 0 and $n-1$, and the second entry must be between 0 and $m-1$. Moreover, all of the cells visited along the path via *r*, *l*, *u*, *d* sequence must also stay within the limits of the matrix. Such an invalid move is called impossible move and if such a move is entered by the user, you have to display an error message and continue with the game.

The second type of impossible move is *reuse* of a cell. This can happen in two ways; (i) first being attempting to use a cell whose content has been replaced with an underscore, and (ii) the second being attempting a reuse of a cell in the within the current path. In such cases, you have to give an appropriate error message (see "sample runs" for examples).

Another error case is with path input characters. Of the four available directions, right is represented with ‘r’, left with ‘l’, up with ‘u’ and down with ‘d’. Any other character is considered an invalid input. You may choose to accept uppercase versions of these character, but it is not required. Please also note that since the minimum word length is 3, minimum number of characters in the path string must be 2.

Sample Runs

Some sample runs are given below, but these are not comprehensive, therefore you must consider **all possible cases** to get full mark.

Dictionary file (words.dat) and the puzzle sample files are provided within the homework package.

Sample 1:

```
Please enter the name of the puzzle file: puzzle.txt

File could not be opened.

Please enter the name of the puzzle file: puzzle1.txt

Please enter the name of the dictionary file: words.txt

File could not be opened.

Please enter the name of the dictionary file: words.dat


e h w o d

c e l r l

n e l o c

c i t u o

s r e p m
```

```
Enter the starting location for word (x, y): 0 6
Invalid coordinate - out of range

e h w o d
c e l r l
n e l o c
c i t u o
s r e p m

Enter the starting location for word (x, y): 0 1
Enter the path: drruu
Invalid path - out of range

e h w o d
c e l r l
n e l o c
c i t u o
s r e p m

Enter the starting location for word (x, y): 0 1
Enter the path: ldd1rru
Invalid path - non-RLUD input

e h w o d
c e l r l
n e l o c
c i t u o
s r e p m
```

```
Enter the starting location for word (x, y):  0 1
Enter the path: ldddrurull
Invalid path - cell visited twice

e h w o d
c e l r l
n e l o c
c i t u o
s r e p m

Enter the starting location for word (x, y): 0 1
Enter the path: ddd
Invalid word - non-existent word "heei"

e h w o d
c e l r l
n e l o c
c i t u o
s r e p m

Enter the starting location for word (x, y): 0 1
Enter the path: drdr
Word found: hello
Word length: 5
Points gained: 25
Current score: 25

e _ w o d
c _ _ r l
n e _ _ c
```

```
c i t u o
s r e p m

Enter the starting location for word (x, y):  **3 0**
Enter the path: **rrrruull**
Invalid path - cell visited previously

e _ w o d
c _ _ r l
n e _ _ c
c i t u o
s r e p m

Enter the starting location for word (x, y): **0 2**
Enter the path: **rdru**
Word found: world
Word length: 5
Points gained: 25
Current score: 50

e _ _ _ _
c _ _ _ _
n e _ _ c
c i t u o
s r e p m

Enter the starting location for word (x, y): **2 4**
Enter the path: **ddluldl**
Word found: computer
Word length: 8
```

```
Points gained: 64

Current score: 114


e _ _ _ _

c _ _ _ _

n e _ _ _

c i _ _ _

s _ _ _ _


Enter the starting location for word (x, y): 4 0

Enter the path: uruluu

Word found: science

Word length: 7

Points gained: 49

Current score: 163


Congratulations, you have solved the puzzle perfectly!

Your final score is 163.

Press any key to continue . . .
```

Sample 2:

```
Please enter the name of the puzzle file: puzzle2.txt

Please enter the name of the dictionary file: words.dat


g n c n

t i o u

u p m f


Enter the starting location for word (x, y): 0 0

Enter the path: l
```

```
Invalid path - path too short

g n c n
t i o u
u p m f

Enter the starting location for word (x, y): 0 2
Enter the path: dlu
Word found: coin
Word length: 4
Points gained: 16
Current score: 16

g _ _ n
t _ _ u
u p m f

Enter the starting location for word (x, y): -1 0

You didn't completely solve the puzzle.
Your final score is 16.
Press any key to continue . . .
```

Sample 3:

```
Please enter the name of the puzzle file: puzzle3.txt
Invalid matrix - inconsistent dimensions
Press any key to continue . . .
```

Sample 4:

```
Please enter the name of the puzzle file: puzzle4.txt
```

```
Invalid matrix - char not lowercase letter
Press any key to continue . . .
```

Sample 5:

```
Please enter the name of the puzzle file: puzzle5.txt
Invalid matrix - cell entry is not a char
Press any key to continue . . .
```

Sample 6:

```
Please enter the name of the puzzle file: puzzle2.txt
Please enter the name of the dictionary file: words.dat

g n c n
t i o u
u p m f

Enter the starting location for word (x, y): 0 -1

You didn't completely solve the puzzle.
Your final score is 0.
Press any key to continue . . .
```

Some Important Rules

In order to get a full credit, your programs must be efficient and well presented, presence of any redundant computation or bad indentation, or missing, irrelevant comments are going to decrease your grades. You also have to use understandable identifier names, informative introduction and prompts. Modularity is also important; you have to use functions wherever needed and appropriate.

When we grade your homework, we pay attention to these issues. Moreover, in order to observe the real performance of your codes, we may run your programs in *Release* mode and **we may test your programs with very large test cases**. Of course, your program should work in *Debug* mode as well.

What and where to submit (PLEASE READ, IMPORTANT)

You should prepare (or at least test) your program using MS Visual Studio 2012 C++. We will use the standard C++ compiler and libraries of the abovementioned platform while testing your homework. It'd be a good idea to write your name and last name in the program (as a comment line of course).

Submissions guidelines are below. Some parts of the grading process might be automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.

Name your solution, project, cpp file that contains your main program using the following convention (the necessary file extensions such as .sln, .cpp, etc, are to be added to it):

"SUCourseUserName_YourLastname_YourName_HWnumber"

Your SUCourse user name is your SUNet user name which is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the file name must be:

Cago_Ozbugsizkodyazaroglu_Caglayan_hw1

In some homework assignments, you may need to have more than one .cpp or .h files to submit. In this case add informative phrases after the hw number. However, do not add any other character or phrase to the file names.

Now let us explain which files will be included in the submitted package. Visual Studio 2012 will create two *debug* folders, one for the solution and the other one for the project. You should delete these two *debug* folders. Moreover, if you have run your program in release mode, Visual Studio may create *release* folders; you should delete these as well. Apart from these, Visual Studio 2012 creates a file extension of *.sdf*; you will also delete this file. The remaining content of your solution folder is to be submitted after compression. Compress your solution and project folders using WINZIP or WINRAR programs. Please use "zip" compression. "rar" or another compression mechanism is NOT allowed. Our homework processing system works only with zip files. Therefore, make sure that the resulting compressed file has a zip extension. Check that your compressed file opens up correctly and it contains all of the solution, project and source code files that belong to the latest version of your homework. Especially double-check that the zip file contains your cpp and (if any) header files that you wrote for the homework.

Moreover, we strongly recommend you to check whether your zip file will open up and run correctly. To do so, unzip your zip file to another location. Then, open your solution by clicking the file that has a file extension of .sln. Clean, build and run the solution; if there is no problem, you could submit your zip file. Please note that the deleted files/folders may be regenerated after you build and run your program; this is normal, but do not include them in the submitted zip file.

You will receive no credits if your compressed zip file does not expand or it does not contain the correct files. The naming convention of the zip file is the same. The name of the zip file should be as follows:

SUCourseUserName_YourLastname_YourName_HWnumber.zip

For example, zubzipler_Zipleroglu_Zubeyir_hw1.zip is a valid name, but

Hw1_hoz_HasanOz.zip, HasanOzHoz.zip

are **NOT** valid names.

Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!
Albert Levi, Bora Makar