**CS 1120 Computer Science II (with Python),** Fall 2020  
Department of Computer Science, Western Michigan University  
Instructor: Wassnaa Al-Mawee, Lab TA: Utkarsha Prashant Kotkar

Andrew Kroll

**SOFTWARE LIFE CYCLE REPORT FOR LAB ASSIGNMENT 3**

**PHASE 1: SPECIFICATION (“What do we build?)**

Write a Python application to find the longest ‘A’ path on a map of capital (uppercase) letters. The map is represented as a matrix (2-dimensional list) of capital letters. Starting from any point, you can go left, right, up and down (**but not diagonally**). A path is defined as the unbroken sequence of letters that only covers the spots with the letter A. The length of the A path is defined as the number of ‘A’s on the path. Your program needs to find the longest A path on the map. In the following example, the characters in the longest A path are highlighted and the length of the longest path is 7.

**PHASE 2: DESIGN**

**2.1 Modules (Classes) and Their Structure (Class Hierarchy)**

**main:**

Contains the main program control

**read\_data\_from\_file:**

accepts file name argument

reads width and height data from file

creates two dimensional array from dimensions

reads data from file

returns two dimensional array of data

**find\_longest\_path\_length:**

accepts two dimensional data array argument

iterates through all coordinates in array checking for longest path lengths

returns longest path length

**find\_path\_length\_recursive:**

accepts two dimensional data array, two dimensional boolean array, row number, and column number arguments

returns 0 if not the letter ‘A’, already checked, or outside bounds

checks value of position in data array, calls itself recursively and marks the space as checked

returns 1 + max value returned by neighboring cells

**Find\_max:**

accepts four numeric arguments

returns the largest of the four arguments

**2.2 Pseudocode for the Modules (Classes)**

**File LA2Main.py:**

# main():

# for each file:

# read data from file

# find longest path

# print longest path length for file

# read\_data\_from\_file(file\_name):

# open file to read

# get width and height from line 1

# create two dimensional array with width and height

# for remaining lines:

# for each character in the line:

# update appropriate cell in array to character

# return two dimensional array

# find\_longest\_path\_length(data):

# iterate through each row in data

# iterate through each column in data

# if cell isn't the letter 'A', continue iterating

# create boolean two dimensional array for tracking checked cells

# mark current cell checked

# find longest path of each neighboring cell

# return longest path length

# find\_path\_length\_recursive(data, checked, row, column):

# return 0 if row or column is outside bounds of two dimensional array

# return 0 if cell is not the letter 'A' or has been checked already

# mark cell as checked

# return 1 + maximum path length of neighboring cells

# find\_max(a, b, c, d):

# return the maximum value in a, b, c, or d

**PHASE 3: RISK ANALYSIS (“What can go wrong, and how bad can it be?”)**

An improperly formatted input file has the potential to crash the program, as well as one or more of the files missing or being improperly named.

**PHASE 4: VERIFICATION (“Are the algorithms correct?”)**

All “algorithms” are functional. No user input aside from files.

**PHASE 5: CODING**

**5a) Code Refinement #1 (class structure with pseudocode only; pseudocode is used as comments)**

**File LA2Main.py:**

# main():

# for each file:

# read data from file

# find longest path

# print longest path length for file

def main():

pass

# read\_data\_from\_file(file\_name):

# open file to read

# get width and height from line 1

# create two dimensional array with width and height

# for remaining lines:

# for each character in the line:

# update appropriate cell in array to character

# return two dimensional array

def read\_data\_from\_file(file\_name):

pass

# find\_longest\_path\_length(data):

# iterate through each row in data

# iterate through each column in data

# if cell isn't the letter 'A', continue iterating

# create boolean two dimensional array for tracking checked cells

# mark current cell checked

# find longest path of each neighboring cell

# return longest path length

def find\_longest\_path\_length(data):

pass

# find\_path\_length\_recursive(data, checked, row, column):

# return 0 if row or column is outside bounds of two dimensional array

# return 0 if cell is not the letter 'A' or has been checked already

# mark cell as checked

# return 1 + maximum path length of neighboring cells

def find\_path\_length\_recursive(data, checked, r, c):

pass

# find\_max(a, b, c, d):

# return the maximum value in a, b, c, or d

def find\_max(a, b, c, d):

pass

**5b) Code Refinement #2 (still incomplete program: class and constructor/method structure with pseudocode only; pseudocode is used as comments)**

**File LA2Main.py:**

# main():

# for each file:

# read data from file

# find longest path

# print longest path length for file

def main():

for i in range(1, 5):

file\_name = “sam\_{}.txt”.format(i)

data = read\_data\_from\_file(file\_name)

print(“{} Longest Path: {}”.format(file\_name, find\_longest\_path\_length(data)))

# read\_data\_from\_file(file\_name):

# open file to read

# get width and height from line 1

# create two dimensional array with width and height

# for remaining lines:

# for each character in the line:

# update appropriate cell in array to character

# return two dimensional array

def read\_data\_from\_file(file\_name):

with open(file\_name, ‘r’) as file:

lines = file.readlines()

lines[0] = lines[0].strip()

width = int(lines[0].split(“ “)[0])

height = int(lines[0].split(“ “)[1])

data = [[“” for c in range(width)] for r in range(height)]

for r in range(1, len(lines):

line = lines[r].strip()

for c in range(len(line)):

data[r-1][c] = line[c]

return data

# find\_longest\_path\_length(data):

# iterate through each row in data

# iterate through each column in data

# if cell isn't the letter 'A', continue iterating

# create boolean two dimensional array for tracking checked cells

# mark current cell checked

# find longest path of each neighboring cell

# return longest path length

def find\_longest\_path\_length(data):

longest = 0

for r in range(len(data)):

for c in range(len(data[r])):

pass

# find\_path\_length\_recursive(data, checked, row, column):

# return 0 if row or column is outside bounds of two dimensional array

# return 0 if cell is not the letter 'A' or has been checked already

# mark cell as checked

# return 1 + maximum path length of neighboring cells

def find\_path\_length\_recursive(data, checked, r, c):

if r < 0 or c < 0 or r >= len(data) or c >= len(data[r]):

return 0

if data[r][c] != ‘A’ or checked[r][c] is True:

return 0

checked[r][c] = True

pass

# find\_max(a, b, c, d):

# return the maximum value in a, b, c, or d

def find\_max(a, b, c, d):

return max(max(a, b), max(c, d))

**5c) Code Refinement #3 (complete program – with complete fields/properties, code for constructor/methods)**

**LA2Main.py:**

*def* main():  
 *"""  
 Handles file selection, retrieval of longest path length, and display of  
 longest path length.  
 """  
 for* i *in range*(1, 5):  
 file\_name = "sam\_{}.txt".format(i)  
 data = read\_data\_from\_file(file\_name)  
 *print*("{} Longest Path: {}".format(file\_name,  
 find\_longest\_path\_length(data)))  
  
  
*def* read\_data\_from\_file(file\_name):  
 *"""  
 Handles reading data from file into a two-dimensional array* ***:param*** *file\_name: The name of the file to read from* ***:return****: A two-dimensional array of characters  
 """  
 with open*(file\_name, 'r') *as* file:  
 lines = file.readlines()  
 lines[0] = lines[0].strip()  
 width = *int*(lines[0].split(" ")[0])  
 height = *int*(lines[0].split(" ")[1])  
 data = [["" *for* c *in range*(width)] *for* r *in range*(height)]  
 *for* r *in range*(1, *len*(lines)):  
 line = lines[r].strip()  
 *for* c *in range*(*len*(line)):  
 data[r-1][c] = line[c]  
 *return* data  
  
  
*def* find\_longest\_path\_length(data):  
 *"""  
 Iterates through all cells in the passed two-dimensional array and begins  
 recursive search for the maximum path length* ***:param*** *data: A two-dimensional array of characters* ***:return****: The maximum path length in the two-dimensional array  
 """* longest = 0  
 *for* r *in range*(*len*(data)):  
 *for* c *in range*(*len*(data[r])):  
 *if* data[r][c] != 'A':  
 *continue* checked = [[*False for* xc *in range*(*len*(data[r]))] *for* xr *in  
 range*(*len*(data))]  
 checked[r][c] = *True* longest = *max*(longest, 1 +  
 find\_max(find\_path\_length\_recursive(data, checked,  
 r-1, c),  
 find\_path\_length\_recursive(data, checked,  
 r+1, c),  
 find\_path\_length\_recursive(data, checked,  
 r, c-1),  
 find\_path\_length\_recursive(data, checked,  
 r, c+1)))  
 *return* longest  
  
  
*def* find\_path\_length\_recursive(data, checked, r, c):  
 *"""  
 Recursively search for the longest path in the two-dimensional array* ***:param*** *data: A two-dimensional character array* ***:param*** *checked: A two-dimensional boolean array* ***:param*** *r: The row number of the cell to check* ***:param*** *c: The column number of the cell to check* ***:return****: The maximum path length of neighboring cells, plus one  
 """  
 if* r < 0 *or* c < 0 *or* r >= *len*(data) *or* c >= *len*(data[r]):  
 *return* 0  
 *if* data[r][c] != 'A' *or* checked[r][c] *is True*:  
 *return* 0  
 checked[r][c] = *True  
 return* 1 + find\_max(find\_path\_length\_recursive(data, checked, r - 1, c),  
 find\_path\_length\_recursive(data, checked, r + 1, c),  
 find\_path\_length\_recursive(data, checked, r, c - 1),  
 find\_path\_length\_recursive(data, checked, r, c + 1))  
  
  
*def* find\_max(a, b, c, d):  
 *"""  
 Finds the maximum value of four values* ***:param*** *a: A decimal number* ***:param*** *b: A decimal number* ***:param*** *c: A decimal number* ***:param*** *d: A decimal number* ***:return****: The largest number passed to the function  
 """  
 return max*(*max*(a, b), *max*(c, d))  
  
  
main()

**PHASE 6: TESTING (“Did we build it correctly?”)**

Yes. See test output below…

sam\_1.txt Longest Path: 9

sam\_2.txt Longest Path: 20

sam\_3.txt Longest Path: 7

sam\_4.txt Longest Path: 18

**PHASE 7: REFINING THE PROGRAM (“Add bells and whistles to the program”)**

Nothing to add.

**PHASE 8: PRODUCTION**

Uploaded, along with this document, to the dropbox.

**PHASE 9: MAINTENANCE**

If any maintenance is required, I may do so upon receiving feedback.