CS917 Foundations of Computing
– Algorithms Coursework

2023/2024

Administration

This assessed piece of coursework must be submitted by **Noon 12:00, Monday, 27th November 2023**. Submission is done electronically via Tabula, and two files are expected for submission:

1. A PDF of your solutions named Answers.pdf.
2. A file named Practical.py of your python solution for the parts of Question 9.

The solutions submitted must be your **own** work. You are expected to work independently, not in groups. Please show the full work progress where appropriate, detailing how you have arrived at your answer.

Questions

Question 1 (10 points)

For each function $f(n)$ and time t in the following table, determine the largest size n (an integer) of a problem that can be solved in time t, assuming that the algorithm to solve the problem takes $f(n)$ microseconds. Use Python as a tool to assist in working out answers. **Explain your work progress, not just write down the answers.** (We will assess your work progress based on whether you have shown a clear understanding of the problem and that you have solved this problem independently; simply writing down the answers without explanation will receive no marks).

	1 second	1 minute	1 hour	1 day
$n \log_2 n$				
n^2				
n^3				
2^n				
$n!$				

Question 2 (10 points)

Determine whether the following complexity statements are true or false. For each answer, explain your reasoning according to the formal definition of O, Ω and Θ.

(a) $10n^2 + 9$ in $O(n^2)$

(b) $5n^4 - 4n^2 + 3$ in $\Omega(n^4)$

(c) $19n^3 - 8n$ in $O(n^4)$

(d) $3n^4 + 2n^2 + n$ in $\Theta(n^2)$

(e) $2n^2 + 16n + 115$ in $\Omega(n)$

Question 3 (10 points)

Where possible, apply the master theorem for the following. If not possible, state the reason why.

(a) $T(n) = 2^n T(\frac{n}{2}) + n$

(b) $T(n) = 16T(\frac{n}{4}) + 16n^2$

(c) $T(n) = 4T(\frac{5n}{3}) + 50n^3$

(d) $T(n) = 2T(n/4) + f(n)$ and all you know about $f(n)$ is that $f(n) = O(n^2)$

(e) $T(n) = 8T(n/4) + f(n)$ and all you know about $f(n)$ is that $f(n) = \Omega(n)$

Question 4 (10 points)

For each of the following input lists, state what you believe to be the most effective sorting algorithm. Use only the algorithms that are covered in the lecture. Justify your reasoning, taking into account complexity, number of operations, memory usage and any other properties of sorted lists. If there are features or attributes of the algorithm that are implementation dependent, state these in your answer.

(a) [1, 2, 3, 4, 6, 5, 7, 10]

(b) [13, 12, 9, 6, 5, 4, 3, 2, 1]

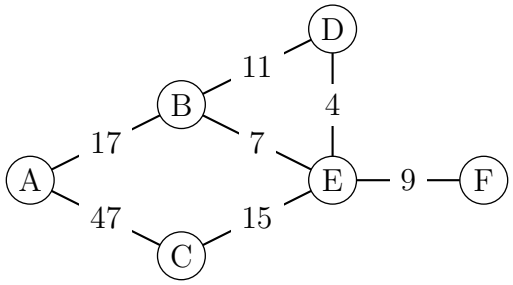
(c) [1, 10, 5, 8, 3, 9, 6, 4, 2]

(d) [6, 5, 6, 5, 9, 11, 1, 23, 7]. Two objects with the same values need to preserve their order after sorting.

(e) [(3, 9), (4, 5), (4, 4), (9, 5), (8, 7), (10, 6)] where each tuple (key, value) is sorted by the key. The ordering of the values must be preserved where the keys are equivalent.

Question 5 (5 points)

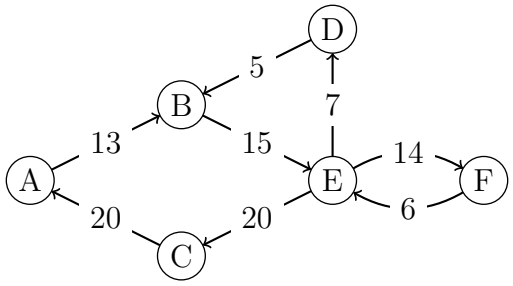
Apply Dijkstra's Algorithm to the following graph, computing the shortest path for all vertices from vertex A.



Write a paragraph to explain your work process and **present the results by drawing the updated diagram** after each vertex has been processed. In each diagram, highlight the vertices that have been processed and the labels of distance at each vertex.

Question 6 (5 points)

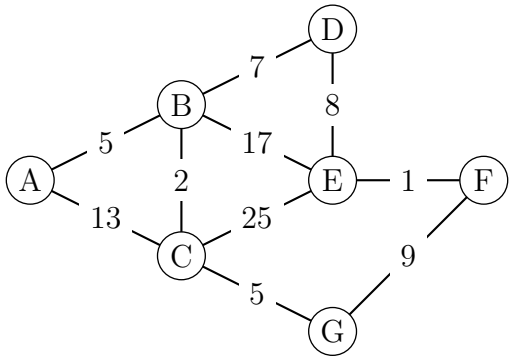
Apply Dijkstra's Algorithm to the following directed graph, and compute the shortest path for all vertices from vertex A.



As in Question 5, write a paragraph to explain your progress and present your results by drawing the updated diagram after each vertex is processed. In each diagram, highlight the vertices that have been processed and the labels of distances at each vertex.

Question 7 (10 points)

For the following graph:



(a) Apply Prim's Algorithm, beginning at vertex A. Show the results of each stage of the algorithm. In this case, we define a stage as the addition of a new vertex and a new edge to the MST S={V,E}.

Present the results in the format of the following table (note that **values provided are only examples and do not correspond to the solution**), and **write a paragraph** to explain your work process.

Stage	V	E
0	(A)	()
1	(A,B)	((A,B))
⋮	⋮	⋮
n	(A,B,C.D,E,F)	((A,B),(B,C),(C,D),(D,E),(E,F))

(b) Apply Kruskal's Algorithm. Show the results of each stage of the algorithm. In this case, we define a stage as the processing of an edge from the graph, whether or not it is added to the MST S={V,E}.

Present the results in the format of the following table (**values provided are only examples and do not correspond to the solution**), and **write a paragraph** to explain your work process.

Stage	Edges	Components	E
0	((A,B), (B,C), (C,D), (D,E), (E,F))	((A), (B), (C), (D), (E), (F))	()
1	((B,C), (C,D), (D,E), (E,F))	((A,B), (C), (D), (E), (F))	((A,B))
⋮	⋮	⋮	⋮
n		((A,B,C,D,E,F))	((A,B), (B,C), (C,D), (D,E), (E,F))

Note the division of the MST Vertices Set into Connected Components.

Question 8 (10 points)

Figure 1 shows left and right rotations in a red-black tree to resolve violations. T denotes the tree, while x and y are two nodes on the tree. The operation LEFT-ROTATE(T, x) transforms the configuration of the two nodes on the left into the configuration on the right by changing a constant number of pointers. The inverse operation RIGHT-ROTATE(T, y) transforms the configuration on the right into the configuration on the left. The letters α, β, and γ represent arbitrary sub-trees.

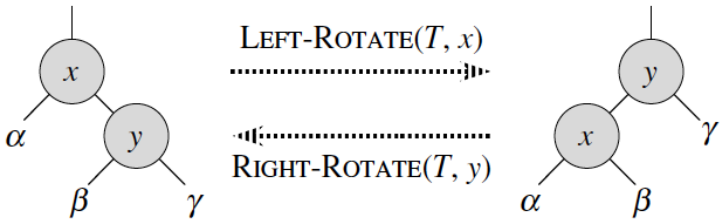


Figure 1: The rotation operations on a red-black tree

The pseudo-code for LEFT-ROTATE is shown in Algorithm 1. A node's left child, right child and parent are represented as .left, .right and .p respectively. T.nil represents an empty node. This algorithm assumes that x.right $\neq$ T.nil and that the root's parent is T.nil. Complete the pseudo-code for RIGHT-ROTATE in Algorithm 2.

Algorithm 1 LEFT-ROTATE(T, x)

```
Require: x.right ≠ T.nil, T.root.p == T.nil
 1: y = x.right                          // set y
 2: x.right = y.left                     // turn y's left substree into x's right subtree
 3: if y.left ≠ T.nil then
 4:     y.left.p = x
 5: end if
 6: y.p = x.p                            // link x's parent to y
 7: if x.p == T.nil then
 8:     T.root = y
 9: else if x == x.p.left then
10:     x.p.left = y                     // link y to be the left child of x's parent
11: else
12:     x.p.right = y
13: end if
14: y.left = x                           // put x on y's left
15: x.p = y
```

Question 9 (30 points)

The following parts are intended to test your application of algorithms and understanding of their performance. Implement your solutions within the file Practical.py and include it in your submission. You will be assessed not only on functionality but also on your approach and implementation, so include comments detailing how your code functions and apply appropriate good programming practices. You may write additional methods or classes for any extra data structures you feel you may need to answer the question, but **do not change the signatures of existing functions in the skeleton file Practical.py** (we will check your program based on testing these functions). An example of the usage of these tasks is

Algorithm 2 RIGHT-ROTATE(T, y)

Require: y.left $\neq$ T.nil, T.root.p == T.nil

```
1:                                                  // set x
2:
3: if                  then
4:
5: end if
6:                                                  // link y's parent to x
7: if               then
8:
9: else if              then
10:                                                  // link x to be the left child of y's parent
11: else
12:
13: end if
14:                                                 // put y on x's right
15:
```

provided in the skeleton file. When it is requested that you explain something specifically for a question, you may include the answer in Answers.pdf.

(a) **Morse Code**

You've taken up a summer job at a lighthouse, when you notice a light blinking in the distance - it looks like morse code! With your computer close at hand, you decide to write a method that can decode morse code as quickly as possible. Morse code is a series of dot (.) and dash (-) symbols that when combined in a certain order represent a single letter. This ordering is fixed, as found in Figure 2. By processing each letter, we can construct a word.

In the file Practical.py, write a method called `morseDecode` that will take a single input of a list of Strings. Each string in the list will represent a single letter of a word in morse code (i.e. a string of . and -). Your method should take the list of strings and return a string that contains the message translated into English. **Write a paragraph** to detail your design decisions and implementation of this method in Answers.pdf. (If you find it easier to structure your answers in several paragraphs instead of one, that will also be acceptable.)

(b) **Incomplete Morse Code**

Unfortunately, the seas are very busy this night. Passing ships have been blocking your view of the light. Luckily, we still are able to figure out how many characters there were per morse code letter, but we are missing the first character of every morse-code letter.

In Practical.py, write a method called partialMorseDecode that is passed a single word in morse code as a list of strings (each index being a single letter in morse code, the

same as part (a)). To represent a missing symbol, we will use the lowercase letter 'x'. The function should return a list of strings, consisting of valid words only. We consider a valid word to be any word that exists in a provided file, dictionary.txt. You may write any additional data structures or methods that you believe you will need. It is recommended to begin with that you use short words. As part of your answer, **Write a paragraph** to discuss your implementation approach and the complexity of this algorithm in Answers.pdf.

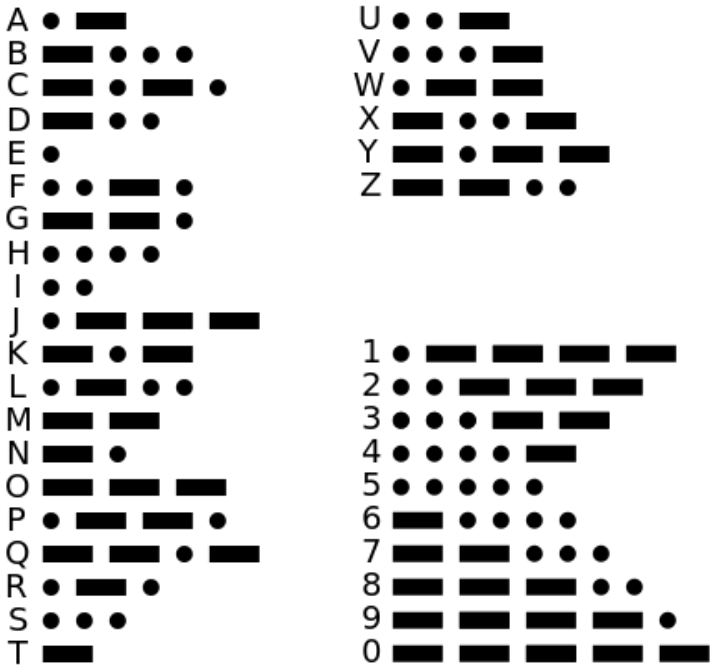


Figure 2: Morse Code

(c) **The Maze**

The morse code has led you on a search for treasure! Following the message's instructions, you find yourself at the top of a set of stairs, with a maze lying below. You can see which parts are walls and which parts are open, but the maze is winding and a path through difficult to spot. A mystery person calls out, mapping the maze from above, dividing it into a grid like system as in Figure 3:

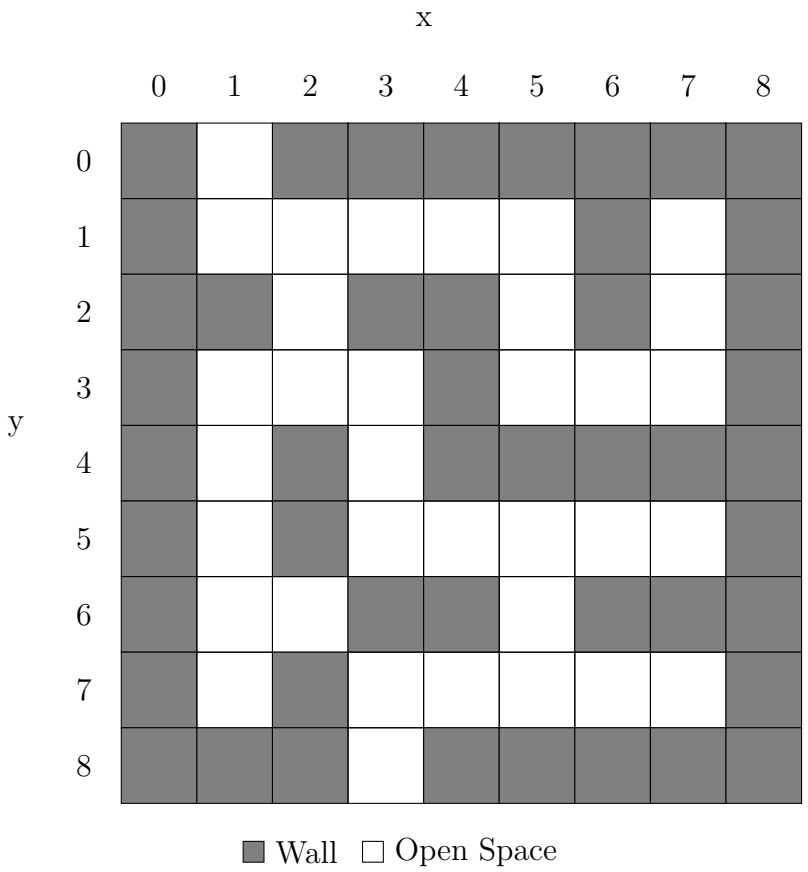


Figure 3: Maze

The above is an example maze, but the maze you are looking at is far more complex. Assuming a coordinate system that begins with (0,0) at the top left, you must store information about the state of the maze, and use that information to map a route through it. *You may assume all mazes used for testing are rectangles, i.e., $M \times N$ where M and N are determined by the maximum of the given x and y coordinates respectively.* Given a class called `Maze`, complete these three methods:

- `addCoordinate(x, y, blockType)`: The x represents the x coordinate on the grid. The y represents the y coordinate on the grid. The blockType can be either 0, to represent an open space, or 1 to represent a wall. The only coordinates you know about in a maze are those added by addCoordinate. If you have been told nothing about a coordinate, but it fits within the current height and width of the maze (i.e. the highest x and y values that have been passed), you may assume it is a wall.
- `printMaze()`: This method should print a representation of the maze, using a star (*) character for walls, and a empty space for open spaces, similar to how Figure 2 is presented (but with symbols instead of squares). You should go up to the maximum height and width of the coordinates you know of so far from addCoordinate.

- `findRoute(x1, y1, x2, y2)`: This method returns a list of (x, y) tuples that map a route to follow, starting from (x1, y1) and moving to (x2, y2). They should be ordered in the list such that you can always move from the coordinate at index i to the coordinates at index i+1, starting at (x1, y1) in index 0 and finishing at coordinates (x2,y2) in the last index position. If no route is available, return an empty list.

In Answers.pdf, **write a paragraph** to detail your implementation and why you chose that approach. Again, you may write additional methods and classes if you wish, as long as they are in the Practical.py file and the methods described above are completed.