CS917 Foundations of Computing - Algorithms and Complexity

1 Administration

This assessed piece of work must be submitted by **Midday, Monday 26th November, 2018**. Submission is done electronically through 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 8.

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

If you have questions or issues, please contact me at J.Davis.4@warwick.ac.uk

2 Questions

Question 1 (5 points)

Suppose you have a machine that can perform 817 operations per second. Given the following functions (assume this is the exact number of operations as a function of n), what is the maximum size of n for which you can complete computation within an hour? Assume n must be an integer.

(a) $f(n) = 4n$

(b) $f(n) = 3n^3$

(c) $f(n) = 100000.log_2(n) + 10$

(d) $f(n) = 87.(n/2)$

(e) $f(n) = 2^n$

Question 2 (5 points)

Determine whether the following complexity statements are true or false. Explain your reasoning in the context of the formal definition of O, Ω and Θ.

(a) $8n^2 + 4$ is $O(n^2)$

(b) $16n^3 - 4n + 3$ is $O(n^4)$

(c) $3n^4 - n^2 + 7$ is $\Omega(n^2)$

(d) $3n^4 - 2n^2 + 3$ is $O(n^2)$

(e) $6n + 143$ is $\Theta(n)$

Question 3 (10 points)

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

(a) $8T(\frac{n}{2}) + 160n$

(b) $2^nT(\frac{n}{2}) + n$

(c) $T(\frac{2n}{3}) + 1$

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

(e) $9T(\frac{n}{3}) + 97n^3$

Question 4 (10 points)

For each of the following input lists, state what you believe to be the most effective sorting algorithm. 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) $1, 2, 3, 4, 6, 5$

(b) $6, 2, 3, 4, 5, 1$

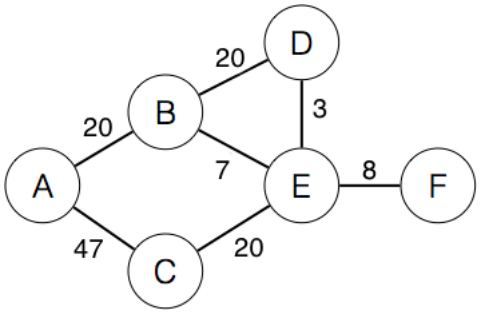
(c) $6, 5, 4, 3, 2, 1$

(d) $1, 5, 4, 6, 3, 2$

(e) ((9,3), (2,5), (2,4), (2,7), (1,4), (5,7)) where each tuple(key, value) is sorted by the key. The ordering of the values must be retained 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. Present the results after each vertex has been processed.



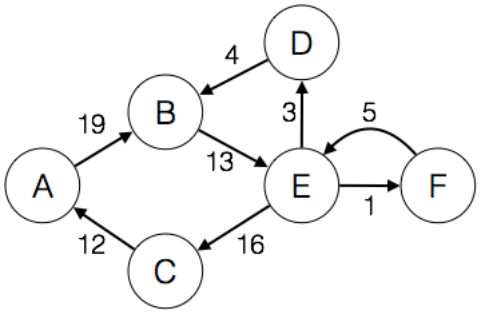
You may wish to present the results in the format of the following table:

Stage	Current Vertex	Labels and Distances											
		A		B		C		D		E		F	
0	-	A	0	-	∞	-	∞	-	∞	-	∞	-	∞
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
n	F	A	0	D	231	A	213	E	4	F	21	A	90

Each row states (a) the current stage, (b) the vertex just added to the search graph, and (c) the current updated predecessor node label and distance from A for each vertex in the graph. Any values given here in the table are merely for example, and do not correspond to the solution for this problem. The vertices should be processed in the order dictated by Dijkstra's Algorithm.

Question 6 (5 points)

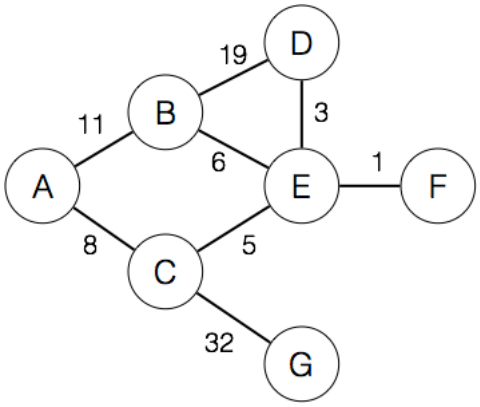
Apply Dijkstra's Algorithm as in Question 5, once again providing the results at each stage, to the following directed graph:



As before, compute the shortest path for all vertices from vertex A.

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 (values provided are only examples and do not correspond to the solution):

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):

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 (50 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, as long as you ensure they are included in the file Practical.py. An example of the usage of these tasks is 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 combined in a certain order represent a single letter. This ordering is fixed, as found in Figure 1. 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. Detail your design decisions and implementation of this method in Answers.pdf.

(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, discuss your implementation approach and the complexity of this algorithm in Answers.pdf

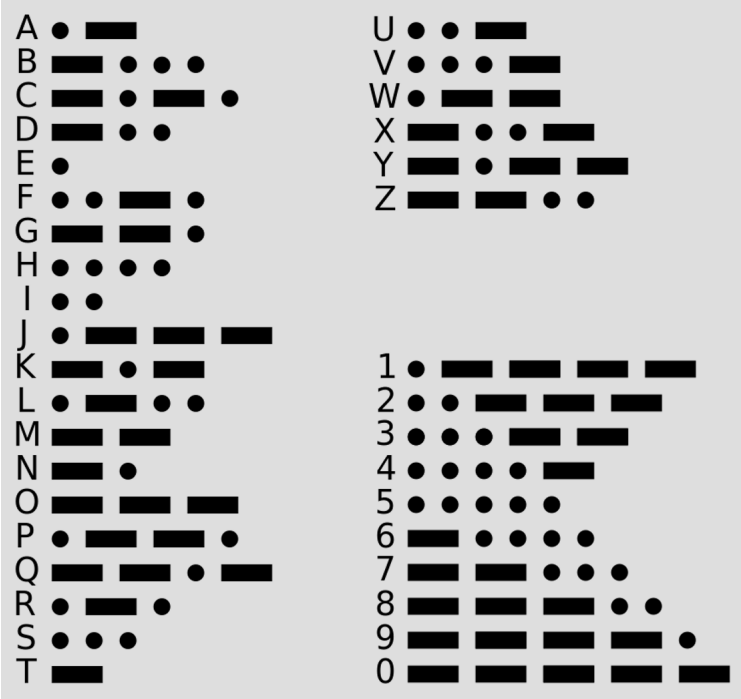


Figure 1: Morse Code

(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 2:

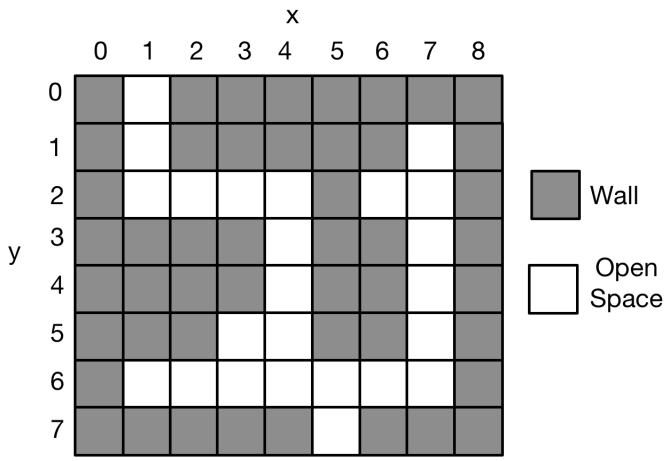


Figure 2: Maze

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. 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 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.