Quiz 12 (Trees)

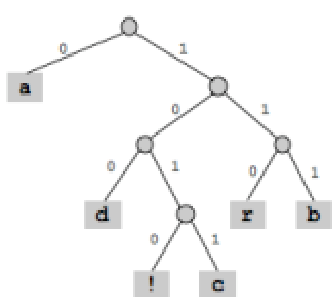
Question 1. The wheel graph W_4 has 5 vertices: 4 vertices form a cycle graph C_4 – a square; one more vertex sits in the middle and is connected with the remaining 4 vertices. (W_4 is isomorphic to an ordinary Egyptian pyramid.) Assume that all vertices in the wheel graph are named with letters and are distinguishable. Find the number of unrooted spanning trees in the W_4.
Write a positive integer in your answer.

Question 2. It is known that a full m-ary tree T has 25 leaves, but the parameter m is not known – it can take any fixed value: $m = 2, 3, 4, \ldots$. How many inner nodes can T have? Find all possible answers.
Write an increasing comma-separated list.

Question 3.
There is a rooted tree with 111 vertices and each vertex can have up to 3 children. Find the minimum and the maximum height of this tree.
Write two comma-separated integers.

Question 4.
Assume that there is a rooted tree (with anonymous/unnamed vertices and unordered children) and its vertices have the following degrees $3, 3, 2, 2, 1, 1, 1, 1$. It is also known that its root vertex has 3 children. Find the number of such rooted unordered trees.
Write a positive integer.

Question 5.



char	encoding
a	0
b	111
c	1011
d	100
r	110
!	1010

Figure 1: Encoding with a Tree

The given tree shows an efficient method send messages (like `abracadabra!`) using 6 characters. Assume that the characters appear with the following probabilities:

Symbol	a	b	d	r	c	!
Probability	1/2	1/8	1/8	1/8	1/16	1/16

Find the expected number of bits used per character (i.e. $E(X)$ – the expected value of the random variable X, which describes the number of bits used per one character from this random distribution.)

Write a real number – the number of bits rounded to the nearest thousandth.

Question 6. The string

$$\to\ \to\ \to\ \to\ \to\ a\ b\ \triangle\ \neg\ c\ \neg\ d\ c\ e\ \to\ \neg\ a\ \to\ d\ e.$$

is the prefix notation for a Boolean logic expression with one symbol replaced by a $\triangle$. What can this $\triangle$ represent?
(A) It is a propositional variable (a, b, c or similar).
(B) It is a unary Boolean operator ($\neg$ or similar).
(C) It is a binary Boolean operator ($\to$ or similar).
(D) It cannot be any of these; the expression is invalid in all these cases.
Write the answer letter (A, B, C, or D).

Question 7. Imagine that you search for all ways how to place 4 queens on a 4×4 chessboard so that they do not attack each other. (A chess queen attacks all squares on its horizontal, on its vertical and also on both diagonals.)
Imagine that you build a tree for this:
(Level 0 to Level 1) The root of the tree is an empty 4×4 chess-board; you add to it four children (all 4 ways how you can place a queen on the 1st row of the chess-board).
(Level 1 to Level 2) For any of the vertices you added in the previous step, add children on level 2 by placing another queen on the 2nd row so that it does not attack the first one.
In general, the vertex on level $L = i$ has queens on rows $1, \ldots, i$ that do not attack each other. Any vertex on level $L = 4$ will be a solution to this "4 queens problem" with all four rows containing a queen.
Write the total number of vertices in this tree (that the backtracking algorithm will visit).

Question 8. An undirected graph $G = (V, E)$ has the set of vertices V – the set of all positive divisors of the number 900 (including 1 and 900 itself). A pair of divisors (d_1, d_2) is an edge in G iff their ratio d_1/d_2 (or d_2/d_1) is a prime number.
In the root vertex $v_1 = 1$ we start the BFS (Breadth-first-search) traversal of the graph G. For every vertex we visit all its adjacent vertices in increasing order (for example edges $(1, 2)$, $(1, 3)$ and $(1, 5)$ are visited in this order. When all the children of vertex 1 are visited, we start visiting all the adjacent vertices of $v_2 = 2$, and so on.). We get the BFS traversal order like this:

$$v_1 = 1,\ v_2 = 2,\ v_3 = 3,\ v_4 = 5,\ v_5 = 4,\ \ldots$$

Find the vertices v_{13}, v_{14}, v_{15} in this BFS order.
Write three comma-separated numbers.

Question 9.

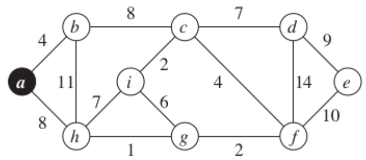


Figure 2: Weighted Graph

Find the weight of a minumum spanning tree (MST) in the given tree. In order to construct this tree, you can use Prim's algorithm (Rosen2019, p.836). Start in vertex a - this is your first tree T_1. In every step pick the minimum weight edge that is adjacent to T_i (and does not create any loop) – add it to the tree T_i, and obtain the next tree T_{i+1}. Continue adding the minimum weight edges until all vertices are connected. (In the second step you will have a choice to add (b, c) or (a, h) as both edges have the same weight 8. There may be several MSTs in a graph, but all of them will have the same total weight.)
Write the total MST weight as a number.

Question 10.

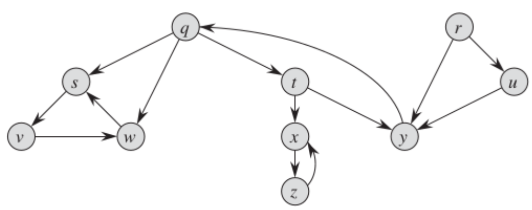


Figure 3: Graph for DFS traversal

The vertices in the above directed graph are visited in the DFS order. You start with the alphabetically smallest vertex (q), order all the vertices connected to it alphabetically, then build the DFS traversal.
Write the sequence with parentheses and vertices for the graph on Figure 3 – similar to the sequence (1) – see Appendix. Each of its 10 vertices should be mentioned in your traversal order twice (the first time with an opening parenthesis, the second time with a closing parenthesis).

Appendix: DFS

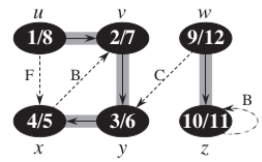


Figure 4: Sample DFS traversal

Figure 4 shows a DFS (Depth-First-Search) traversal in a directed graph. Vertexes are visited in alphabetical order (so u is visited first; followed by its alphabetically first child v, followed by y, followed by x. After that we visit another tree (unreachable from the first one) – w followed by z.)
Tree edges that belong to the DFS traversal tree are shaded;
Back edges that point back from a vertex to its ancestor in the tree (or a loop to itself) are labeled by B;
Forward edges that jump from a vertex to its descendant in the DFS tree (other than a child) are labeled by F;
Cross edges that jump between two vertices that are not descendants/ancestors of each other are labeled by C.

The following sequence

`(u (v (y (x x) y) v) u) (w (z z) w)` (1)

denotes the DFS traversal order in the oriented graph. Every time when we enter some vertex (and its subtree), we open a parenthesis and write the vertex name; when we leave, we write the vertex name again and close the parenthesis.
This order is also written inside each vertex (for example `1/8` for vertex u means that we entered it in Step 1, and left it in Step 8). For vertex z this pair is `10/11` (we entered this leaf of the DFS tree and immediately left it).