EXAM RULES:



* This is a take-home exam, which means that you can use all and any study materials you need.
* Please complete this exam by yourself. It is expected that you will work by yourself and will not ask neither receive help for the exam from any other students or individuals.
* Prepare your answers on separate sheet(s) of paper. You can either write your answers down by hand or prepare them on a computer and then print and submit that. Write your *full name* and *student id* on your answers!

Good luck!

**Q1:** An algorithm performs a calculation by first splitting a dataset of *n* numbers into 2 parts of size and , and performing calculation on the first part. 1/3 of the time a condition is met which allows the algorithm to proceed by recursively calling itself on the first 1/3-part of the dataset, 1/3 of the time the algorithm needs to continue the calculation on the remaining 2/3-part of the dataset, and 1/3 of the time the algorithm makes a transformation on the entire dataset and calls itself again on the entire (transformed) dataset. Estimate the expected run-time of this algorithm in *Θ*-notation. Assume that the probabilities of the conditions becoming true are independent over iterations. A pseudo-code of the algorithm is shown below. HINT: construct the recurrence relationship for T(n) – the average runtime of the algorithm for input of size n – using the information provided in the question and then solve it.

**PROCESS-DATA**(A)

**input** A is an array of numbers

if A.length<=2

return FINAL-CALCULATE(A) // -time

else

PREPARE-PARTS(A,a,b) // split into 2 parts of size and , -time

CALCULATE (a) // -time

if condition1 // true 1/3 of the time

PROCESS-DATA(a) // recursion

else if condition2 // true 1/3 of the time

PROCESS-DATA(b) // recursion

else // true 1/3 of the time

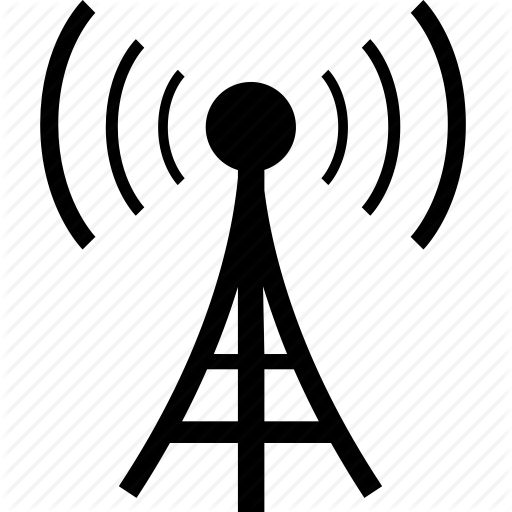
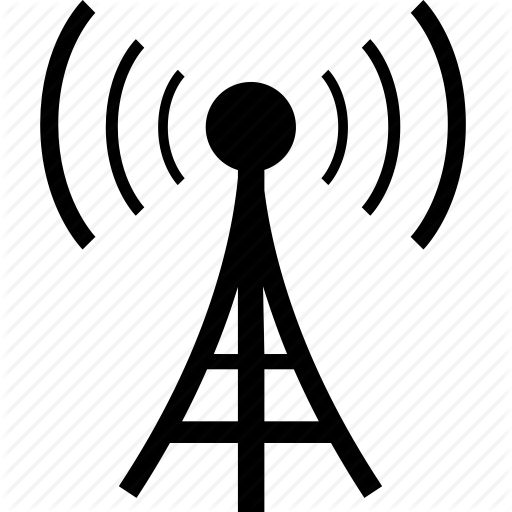
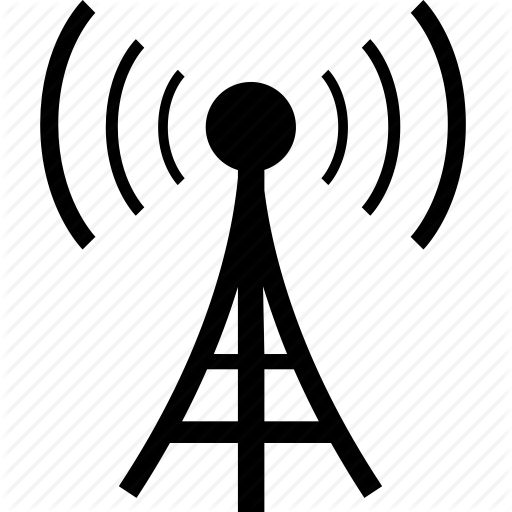
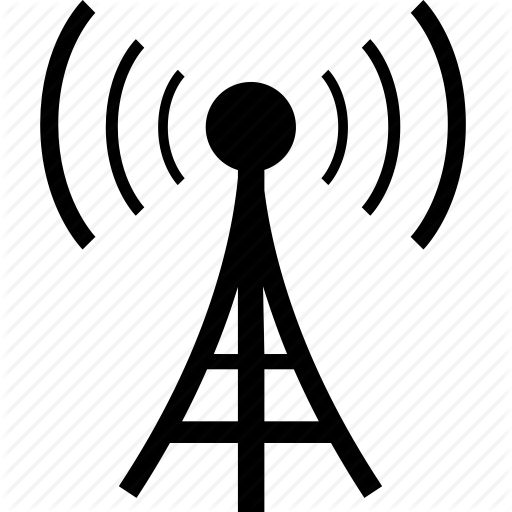
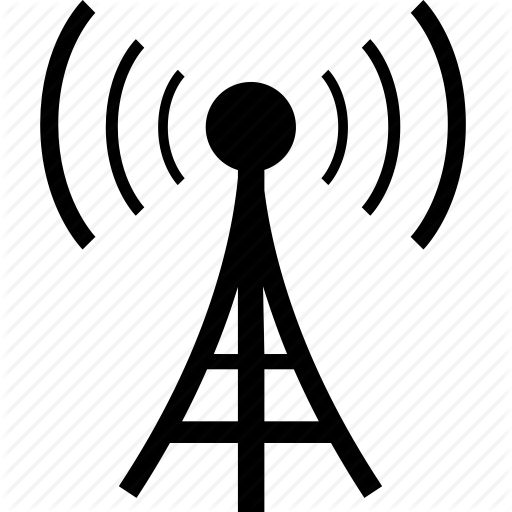
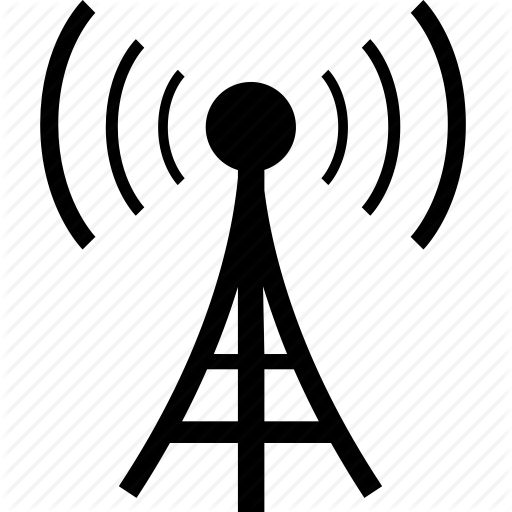
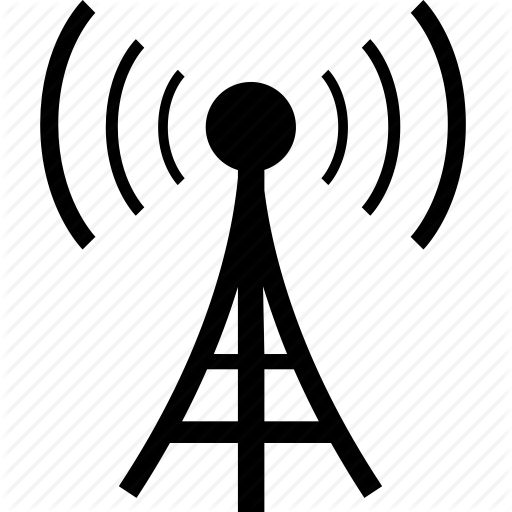
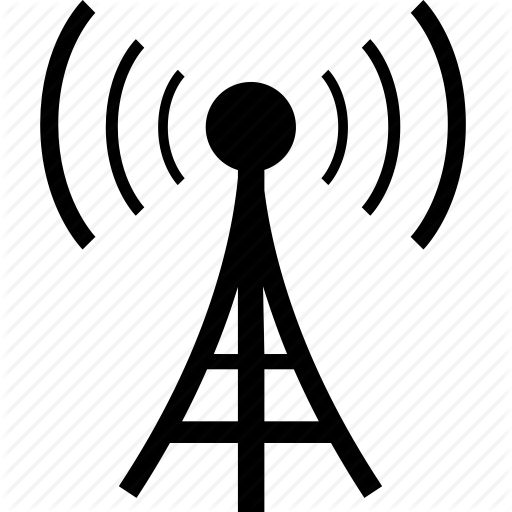
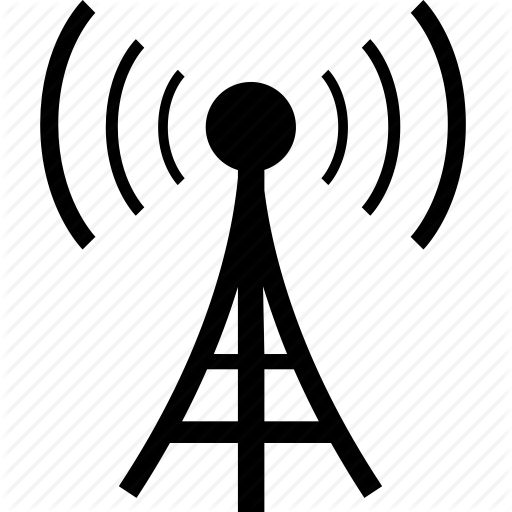
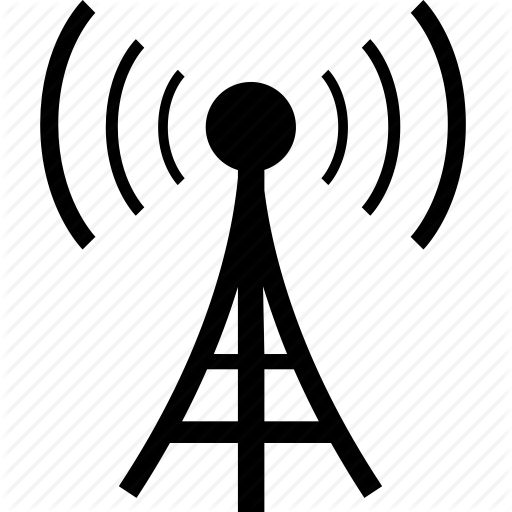
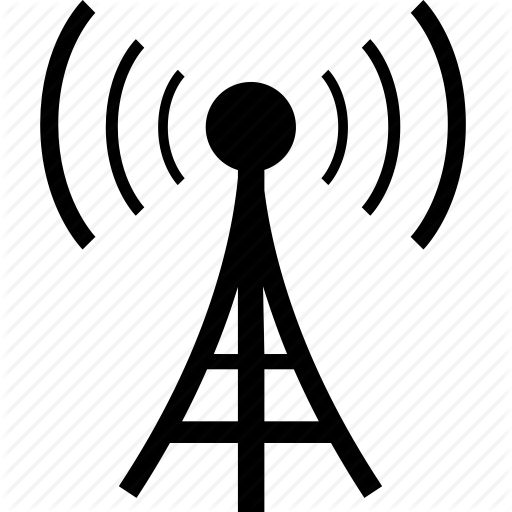
A=SHUFFLE(A) // -time

PROCESS-DATA(A) // recursion

**Q2:** Consider a sequence of coins of given values **V**=. We need to find the smallest number of coins such that they sum to a total of **S**. a) Design and write the pseudo-code for a *greedy solution algorithm* to this problem. b) Design and write the pseudo-code for a *dynamic programming solution* algorithm to this problem. c) Apply your greedy and dynamic programming algorithms to solve the problem instance **V**=(1,5,10,25,50) and **S**=78. Are the answers the same?

**Q3:** Consider an array of positive integers that are first increasing and then decreasing, for example, such as . Propose an algorithm for finding the largest element in this array. Present the pseudo-code.

**Q4:** Cellular phone operators use a number of cell-towers connected with each other by landline high-speed Internet connections to provide service, such as shown in schematic picture to the right. In preparation for transition to new 5G cellular standards, Turkcell decided to upgrade all groups of towers on its network that are all-to-all connected with each other (such as the three towers indicated on the left) with ultra high-speed fiberglass Internet connection. Turkcell gave your division the job of identifying such groups of towers. Now your project manager wants you to write a software that will find exactly all the largest groups of such all-to-all connected towers in a defined region of Turkcell’s network, and then run your algorithm to identify the towers to be upgraded on neighborhood (~10 towers), city (~1000 towers), and national (~106towers) scales. Is this possible? Whether you say *yes* or *no*, explain your answer to the manager. Then offer an algorithm that will solve this problem (in any way you can) and present the pseudo-code for it.



**Q5:** One of the strategies for building decision trees is to split each time on the attribute that offers the largest reduction in the average entropy of the tree. For the tree with terminal nodes each containing data points of class , the average entropy is defined as

where is the total number of data points in terminal node and is the total number of data points. The attribute that would provide the largest decrease in the average entropy of the tree is chosen every time to make the split. Design the algorithm constructing decision tree in this way. For simplicity, you can assume that the decision tree is binary (*S*=2), all attributes are *categorical* (that is each attribute can contain only discrete values), and the number of distinct values in each attribute is always small enough to perform a split into all possibilities. Write the pseudo code for your algorithm and use it to construct the decision tree for the data below.

Personal credit decisions history:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Age | Student | Credit-Rank | Decision |  | Age | Student | Credit-Rank | Decision |
| C1 | Young | Yes | Good | Yes | C10 | Senior | No | Fair | No |
| C2 | Middle | Yes | Fair | Yes | C11 | Young | Yes | Bad | Yes |
| C3 | Young | No | Good | No | C12 | Middle | Yes | Bad | Yes |
| C4 | Middle | No | Bad | Yes | C13 | Middle | Yes | Good | Yes |
| C5 | Senior | No | Bad | No | C14 | Young | No | Bad | No |
| C6 | Senior | No | Fair | No | C15 | Senior | No | Fair | No |
| C7 | Middle | Yes | Good | Yes | C16 | Senior | No | Good | Yes |
| C8 | Young | No | Fair | No | C17 | Young | Yes | Fair | Yes |
| C9 | Senior | No | Good | Yes | C18 | Middle | No | Bad | Yes |