1. (Special note: Below there are five questions for you to answer. Naturally there are many mathematical formulas and equations in the problem statements. However, due to some technical issues on Coursera, from time to time for some learners, some formulas and equations cannot be displayed correctly (while some others can). As there is no way for the instructing team to solve this issue, we post all the problem statements in a PDF file here. If unfortunately some formulas and equations are not readable for you, please use the PDF file to prepare your answers and then come back to Coursera to choose/fill in your answers.) 1 / 1 point

Fifteen jobs, each with its processing time, should be scheduled on three machines. If two jobs cannot be scheduled on the same machine, they are called conflicting jobs. Table 1 lists the job IDs, processing times, and sets of conflicting jobs. For example, we cannot schedule any pair of jobs out of jobs 2, 5, 8 on the same machine. Note that a job may have no conflicting jobs.

Job	Processing time	Conflicting jobs
1	7	None
2	4	5, 8
3	6	None
4	9	None
5	12	2, 8
6	8	9
7	10	10
8	11	2, 5
9	8	6
10	7	7
11	6	15
12	8	None
13	15	None
14	14	None
15	3	11

Table 1: Data for Problem 1

We want to schedule the jobs to minimize makespan. For example, we may schedule jobs 1, 4, 7, 8, and 13 to machine 1, jobs 2, 6, 10, 11, and 14 to machine 2, and jobs 3, 5, 9, 12, and 15 to machine 3. The total processing times on the three machines are 52, 39, and 37, respectively. The makespan is thus 52. While this is a feasible schedule, this may or may not be an optimal schedule. When we try to improve the schedule, be careful about conflicting jobs. For example, we cannot exchange jobs 8 and 11 (even though this reduces the makespan) because that will result in machine 2 processing conflicting jobs 2 and 8, which is infeasible.

Formulate a linear integer program that generates a feasible schedule to minimize makespan. Then write a computer program (e.g., using Python to invoke Gurobi Optimizer) to solve this instance and obtain an optimal schedule. Write down the minimized makespan (i.e. the objective value of an optimal solution). Do not have any symbol other than numeric values in your answer.

A special note posted on 2022/3/5: There are just too many learners asking me about how to solve this problem. I feel sorry that I cannot do better in making it clearer, and I just do not have time to help everyone finding errors in your formulation or computer program. The best I may do to help you learn is, I guess, to post the answer/solution so that you have a reference. The correct answer for this question is "43". If you want to see a Python program invoking Gurobi to solve this problem, please go to the "Resources" section. Hope that helps.

43

Correct

2. A city is divided into n districts. The time (in minutes) it takes an ambulance to travel from District i to District j is denoted as d_{ij}. The population of District i (in thousands) is p_i. An example is shown in Table 2 and Table 3. The distances between districts are shown in Table 2, and the population information is shown in Table 3. In this instance, we have $n = 8$ districts. We may see that, e.g., it takes 5 minutes to travel from District 2 to District 3, and there are 40,000 citizens. 1 / 1 point

District	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	0	3	4	6	8	9	8	10
2	3	0	5	4	8	6	12	9
3	4	5	0	2	2	3	5	7
4	6	4	2	0	3	2	5	4
5	8	8	2	3	0	2	2	4
6	9	6	3	2	2	0	3	2
7	8	12	5	5	2	3	0	2
8	10	9	7	4	4	2	2	0

Table 2: Data for Problem 2 (distances)

District	**Population**
1	40
2	30
3	35
4	20
5	15
6	50
7	45
8	60

Table 3: Data for Problem 2 (population)

The city has m ambulances and wants to locate them to m of the districts. For each district, the *population-weighted firefighting time* is defined as the product of the district population times the amount of time it takes for the closest ambulance to travel to it. The decision maker aims to locate the m ambulances to minimize the *maximum* population-weighted firefighting time among all districts.

As an example, suppose that $m = 2$, $n = 8$, d_{ij} and p_i are provided in Table 2, and the two ambulances are located in District 1 and 8. We then know that for Districts 1, 2, and 3 the closest ambulance is in District 1 and for the remaining five districts the closet ambulance is in District 8. The firefighting time for the eight districts are thus 0, 3, 4, 4, 4, 2, 2, and 0 minutes, respectively. The population-weighted firefighting times may then be calculated as 0, 90, 140, 80, 60, 100, 90, and 0. The maximum among the eight districts is therefore 140.

For this problem, formulate an integer program that can minimize the maximum population-weighted firefighting time among all districts. Then write a program to invoke a solver (e.g., write a Python program to invoke Gurobi Optimizer) to solve the above instance and find an optimal solution for each problem. Write down the minimized maximum population-weighted firefighting times among all districts of the two districts that ambulances should be located in (i.e., the objective value of an optimal solution). Do not have any symbol other than numeric values in your answer.

Hint. To formulate a linear integer program for this problem, you may try to define the following set of decision variables: x_j is 1 if an ambulance is located in District j and 0 otherwise, y_{ij} is 1 if for District i the closest ambulance is located in District j and 0 otherwise, and w_i as the distance between District i and its closest ambulance. You may then want to consider the following IP

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{n} w_i \\
\text{s.t.} \quad & \sum_{j=1}^{n} x_j = m \\
& y_{ij} \le x_j \quad \forall i = 1, \ldots, n, j = 1, \ldots, n \\
& \sum_{j=1}^{n} y_{ij} = 1 \quad \forall i = 1, \ldots, n \\
& w_i \ge \sum_{j=1}^{n} d_{ij} y_{ij} \quad \forall i = 1, \ldots, n. \\
& x_i, y_{ij} \in \{0, 1\} \quad \forall i = 1, \ldots, n, j = 1, \ldots, n \\
& w_i \ge 0 \quad \forall i = 1, \ldots, n
\end{aligned}
$$

about what it will do. You may then try to modify this IP to form your IP for this problem.

A special note posted on 2022/3/5: There are just too many learners asking me about how to solve this problem. I feel sorry that I cannot do better in making it clearer, and I just do not have time to help everyone finding errors in your formulation or computer program. The best I may do to help you learn is, I guess, to post the answer/solution so that you have a reference. The correct answer for this question is "135". If you want to see a Python program invoking Gurobi to solve this problem, please go to the "Resources" section. Hope that helps.

135

Correct

3. Continue from the previous question. For any value of m, consider the following heuristic algorithm which runs m iterations. In each iteration, we locate an ambulance in a district that (1) currently does not have an ambulance, and (2) may minimize the maximum population-weighted firefighting times among all districts. If there are multiple districts satisfying these two conditions, pick the one with the smallest district ID among them. We then proceed to the next iteration to look for the next district to locate an ambulance. 1 / 1 point

Consider a tiny example with $n = 4$, $m = 2$, and d_{ij} and p_j are provided in Table 4 and Table 5. To locate the first ambulance, we examine the maximum population-weighted firefighting times of locating an ambulance in Districts 1, 2, 3, and 4 as 140, 175, 160, and 240, respectively. We will choose District 1. To locate the second ambulance, we take the ambulance in District 1 as given and examine the maximum population-weighted firefighting times of locating an ambulance in Districts 2, 3, and 4 as 140, 90, and 90, respectively. We will choose District 3. The final objective value of locating two ambulances in Districts 1 and 3 is 90 (which is 30×3 for District 2).

District	1	2	3	4
1	0	3	4	1
2	3	0	5	8
3	4	5	0	1
4	1	8	1	0

Table 4: Data for Problem 3 (distances)

District	Population
1	40
2	30
3	35
4	5

Table 5: Data for Problem 3 (population)

Coming back to the instance provided in Table 2 and Table 3 with $n = 8$. Now let $m = 3$. Use the heuristic algorithm introduced above to generate a feasible solution. Then use the program you wrote for Question 2 to generate an optimal solution. Write down the absolute optimality gap between the two solutions (i.e., the difference between the two objective values). Do not have any symbol other than numeric values in your answer.
A special note posted on 2022/3/5: There are just too many learners asking me about how to solve this problem. I feel sorry that I cannot do better in making it clearer, and I just do not have time to help everyone finding errors in your formulation or computer program. The best I may do to help you learn is, I guess, to post the answer/solution so that you have a reference. The correct answer for this question is "0". If you want to see a Python program invoking Gurobi to solve this problem, please go to the "Resources" section. Hope that helps.

0

Correct