AZA - Practical implementation and analysis of an algorithm (30%) – deadline 8/12/2023

Tasks:

1. Design and implement an algorithm to solve a problem for scheduling with deadlines (Algorithm 4.4).

   Scheduling with Deadlines

   Problem: Determine the schedule with maximum total profit given that each job has a profit that will be obtained only if the job is scheduled by its deadline.

   Inputs: n, the number of jobs, and array of integers *deadline*, indexed from 1 to n, where *deadline*[i] is the deadline for the ith job. The array has been sorted in nonincreasing order according to the profits associated with the jobs.

   Outputs: an optimal sequence J for the jobs.

```
void schedule (int n,
               const int deadline[],
               sequence_of_integer& J)
{
  index i;
  sequence_of_integer K;

  J = [1];
  for (i = 2; i <= n; i++){
     K = J with i added according to nondecreasing values of deadline[i];
     if (K is feasible)
        J = K;
  }
}
```

   Table 1.1. Consider the following jobs, deadlines, and profits, and use the scheduling with deadlines algorithm to maximize the total profit. Show the optimal sequence of jobs with max profit on the screen.

Job	Deadline	Profit
1	2	40
2	4	15
3	3	60
4	2	20
5	3	10
6	1	45
7	1	55

2. Consider the procedure schedule in the Scheduling with Deadlines algorithm (Algorithm 4.4). Let d be the maximum of the deadlines for n jobs. Modify the procedure so that it adds a job as late as possible to the schedule being built, but no later than its deadline. Do this by initializing d+1 disjoint sets, containing the integers 0, 1, ..., d. Let small(S) be the smallest member of set S. When a job is scheduled, find the set S containing the minimum of its deadline and n. If small(S) = 0, reject the job. Otherwise, schedule it at time small(S), and

merge S with the set containing small(S)−1. Assuming we use Disjoint Set Data Structure III in Appendix C, show that this version is θ(nlgm), where m is the minimum of d and n.

3. Suppose we assign **n** persons to **n** jobs. Let C_{ij} be the cost of assigning the i^{th} person to the j^{th} job.
    a. Use a greedy approach to write an algorithm that finds an assignment that minimizes the total cost of assigning all n persons to all n jobs. Analyze your algorithm and show the results using order notation.
    b. Use the dynamic programming approach to write an algorithm for the same problem. Analyze your algorithm and show the results using order notation.

For illustration, consider the following matrix of costs:

	J1	J2	J3
P1	10	5	5
P2	2	4	10
P3	5	1	7

Show the optimal assignments on the screen.

What you should hand in

Single ZIP file via AIS system. Name convention for the ZIP file:

FirstName_ LastName _ AZA_practical.ZIP

ZIP file should include four CPP files and a single PDF document with your complexity analysis. No other format, such as Word or Open Office, should be included.

Note: late submission, missing files or ZIP file which cannot be expanded, will cost you zero marks for the assignment. It is individual work. Two or more identical submissions will be investigated as plagiarism.

Submission Deadline: 8th December 2023 at 5 PM

Marking table

1. Scheduling with deadline algorithm implementation	3
Correct output on a screen based on input from the table 1.1	1
2. Algorithm Modification – code description	2
Use of Disjoint Set Data Structure III	3
Implementation of the modification	6
Correct output on a screen based on input from the table 1.1	1
Modified algorithm analysis	4
3. Use a Greedy approach to design and implement the algorithm	3
Use Dynamic programming approach to design and implement the algorithm	3
Complexity analysis and comparison of both approaches	4
	30