Theory of Algorithms, First Midterm
Grading Guide.

The purpose of the scoring guide is to provide a uniform evaluation of the tests by the graders. Therefore, the guide gives you the main ideas (and at least one possible solution) for each problem and the associated scores. This guide is not intended to be a detailed description of a complete solution to each problem; the steps described are sketches of a solution that achieves maximum score.
The partial scores given in the guide are only awarded to the solver if the related idea is included in the test as a step in a clearly described and reasoned solution. Thus, for example, the mere description of statements or definitions in the material without its application (even if any of the facts described do indeed play a role in the solution) is not worth the point.
A partial score is given for any idea or part solution from which the correct solution of the problem could be obtained with the appropriate addition of the reasoning described in the test. If a solver starts several solutions that are significantly different, then you can give score for at most one. If each described solution or part of solution is correct or can be extended to be correct, then the solution with the most points will be evaluated. However, if there are correct and incorrect solutions between several solution attempts and it is not clear from the test which one the solver considered to be correct, then the solution with fewer points will be evaluated (even if this score is 0). The scores in this guide may be further subdivided as necessary. Of course, a good solution other than that described in this guide is also worth the maximum score.
In the event of an arithmetic error, one point is deducted from each exercise. An exception to this is when computing the numbers significantly simplifies or modifies the structure of the task. In these cases, no points are awarded to the steps which follow the computation and are missing from the solution of the student.

---

1. Determine constants c and n_0 with which you can show that $f(n) = O(g(n))$, where the functions are as given below:

$$f(n) = 2023n^2 \cdot \log n - 27 \cdot \sqrt{n} \qquad\qquad g(n) = \frac{1}{10^{10}} \cdot n^3 + n \cdot \log n$$

Scoring guide:
(2 points) $f(n) = O(g(n))$ definition with c, n_0
(5 points for the inequalities that follow)
$f(n) = 2023n^2 \cdot \log n - 27 \cdot \sqrt{n} \le 2023n^2 \cdot \log n$
$\le 2023n^3$
$= 2023 \cdot 10^{10} \frac{1}{10^{10}} \cdot n^3$
$\le 2023 \cdot 10^{10} (\frac{1}{10^{10}} \cdot n^3 + n \cdot \log n$

(1 point) All inequalities were true for $n \ge 1$
(2 points) and with $c = 2023 \cdot 10^{10}$

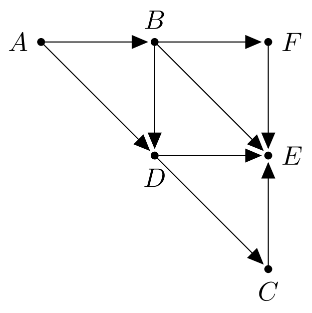
2. An advertising company has n different ways to advertise its product. It is known that it can spend at most M dollars on the whole campaign. For each advertising method, we know the amount the company has to pay/spend for it (denoted by p_i for method i), and also the number of people who are influenced by it (denoted by e_i for method i). Further assume that the sets of people influenced by different methods are pairwise disjoint, i.e., no person can be influenced by two different methods. Given the (strictly) positive integers $M, p_1, p_2, \dots, p_n$ and $e_1, e_2, \dots, e_n$, determine in running time $O(Mn)$, the maximum number of people that can be influenced using at most M dollars.

Scoring guide:
(2 points) Subproblems
(1 point) Order
(1 point) Start
(3 points) Continue:

(1 point) End:
(1 points) Correctness:
(1 point) Running Time:

Full points are also awarded if it is shown to be Knapsack with proper correspondence of the people to value, money to weight etc, and with details about Knapsack's running time.

3. The DFS (depth first search) algorithm when run on the graph given here, starts with visiting the vertices A, B, F in this order.
(a) In what order are the other vertices discovered?
(b) What are the finishing indices of the vertices?
Please give reasons for your answers.



Scoring guide:
(5 points) Table for DFS
(5 points) Reasoning for the entries in the table (a brief list of steps when entries were filled in the table - this is required because we are not given the adjacency list, so you have to support your discovery/finishing indices with why alternate indices were not possible)

4. We are given a table of size $n \times n$, each field of which contains a positive integer. A number may occur more than once in the table. In this table, a legal step consists of going from a square with number x to an adjacent square (square sharing a side) with the number $(x+1)$ on it. We say a path in this table is legal if it consists of only legal steps. Give an algorithm with running time $O(n^2)$ that gives the length of the longest legal path in the table, where length of the path is the number of steps in it.
Scoring guide
(1 point) create a graph, vertices are squares, two squares are connected for a legal step
(1 point) we want the longest path in this graph
(1 point) we need another new vertex connected to all field because we don't know the source or starting field for the longest path
(1 point) we give the graph as Adjacency list (note matrix doesn't work)
(2 point) its a DAG and why
(1 point) run the DAG-LONGEST-PATH
(1 point) correctness: every path on the table is there in this graph, and then DAG-LONGEST-PATH gives the longest such path.
(2 points) Running time: 1 point for RT of creating the graph 1 point for DAG-LONGEST-PATH

A maximum of 3 points can be awarded if tried to solve the problem with DP if subproblems, continue, end are well thought of.

5. Dijkstra's algorithm is run on a **directed** graph where all edges have a positive weight. The table below shows the status of the array Q (of processed vertices) during the run.
(a) Can there be edge from vertex E to vertex B? If yes, what can its weight be?
(b) Can there be edge from vertex B to vertex E? If yes, what can its weight be?

A	B	C	D	E	F
$*$	∞	9	∞	2	5
$*$	∞	5	7	$*$	3
$*$	8	4	5	$*$	$*$
$*$	7	$*$	5	$*$	$*$
$*$	6	$*$	$*$	$*$	$*$

Scoring guide:
(5 points) constructing the mandatory edges in the graph based on updates in the Q table.
(2 points) No for (a) because then in the second row we will not have ∞ for B.
(2 points) Yes for (b) because E has been added to P processed or finalized vertices when B is being processed. So the edge can be there, the algorithm will not look at it because E has attained its minimum path length and is processed.
(1 point) For the same reason as above, the weight of the edge can be any positive number.

6. A country's road network is given as an undirected, weighted graph: the vertices of the graph are the cities (there are n cities), the edges represent the direct roads between cities, and the weights of the edges represent the time taken to travel them. It is known that only some of these roads are free while others are paid. Assume that you are given the adjacency matrix of this graph and whether a road is free or paid is given as an edge attribute.

   We want to get from city A to city B and we want to know if it takes longer, and if so, by how much, if we only travel by roads that are free. Determine which algorithm can solve this problem in $O(n^2)$ running time. Please specify the details of how it will be used.

   Scoring guide:
   (1 point) since positive edge weights are given and we are not sure about cycles, and because we need shortest paths, we should use Dijkstra's.
   (1 point) run Dijkstra's with A as source and record the time to B
   (1 point) we modify the graph by deleting the paid routes.
   (1 point) this is done by changing 1's to 0's in the adjacency matrix
   (1 point) run Dijkstra's again and record time to B.
   (1 point) we give the difference.
   (2 points) Correctness: if there is a free cost path, it is there in our modified graph, and Dijkstra's will find the shortest such path.
   (1 point) Changing the matrix is $O(n^2)$ running time.
   (1 point) Dijkstra's is $O(n^2)$ and so the total running time is also $O(n^2)$.

   If some other shortest path algorithm is used, but it is there that it is run twice and we look at the difference then 5 points are awarded.

7. We are given a sequence of positive integers $a_1, a_2, \ldots, a_n$. For a strictly increasing subsequence $a_{i_1} < a_{i_2} < \ldots < a_{i_k}$ with $i_1 < i_2 < \ldots i_k$, we define its value as $a_{i_1} \cdot a_{i_2} \cdot \ldots \cdot a_{i_k} + a_{i_k}$, i.e., we multiply the numbers in the subsequence and add the last value. Give a dynamic programming algorithm with running time $O(n^2)$ to find the largest value we can obtain as the value of a strictly increasing subsequence. (Note: you don't have to find the subsequence which attains this largest value).
   (For example, in the sequence $2, 5, 1, 3, 10, 7, 9$, the largest value is 639. It is attained by the subsequence $2, 5, 7, 9$ where $2 \cdot 5 \cdot 7 \cdot 9 + 9 = 639$.)

   Scoring guide

(1 point) Subproblems
(1 point) Start
(4 points) Continue
(1 point) End
(1 point) Correctness
(2 points) Running time