Theory of Algorithms, Repeat of the 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. Provide constants c and n_0 to show that $37n^2 \log n + 17\sqrt{n} + 7 \in O(n^2)$ or prove that such constants don't exist.

   Scoring guide:
   (1 point) Such constants don't exist (but this point is awarded only if an attempt is made to explain why)
   (1 point) Prove by contradiction,
   (2 points) Let c, n_0 be such that $37n^2 \log n + 17\sqrt{n} + 7 \leq cn^2$ for $n \geq n_0$
   (2 points) this implies that $37n^2 \log n \leq cn^2$ for $n \geq n_0$
   (1 point) or $37 \log n \leq c$ for $n \geq n_0$
   (1+2 points) this is a contradiction because $\log n$ cannot be bounded by any constant for $n \geq n_0$.

   If it is only given that the above function is $O(n^2 \log n)$ and so cannot be $O(n^2)$ then 2 points are to be awarded.

2. We are given a cycle with n vertices, such that each vertex has a positive integer smaller than n assigned to it. If a vertex has the number k assigned to it, then in a single jump, we can move from this vertex to a vertex exactly k vertices away to the left or right (for example: if the cycle is $abcdefg$ and the number assigned to e is 3, then we can jump from e to b to the left or to a to the right). Given a vertex A and a vertex B in this cycle, we want to determine whether it is possible to get from A to B and if so, what is the least number of jumps required. Give an algorithm with running time $O(n)$ to solve this problem.

   Scoring guide:
   (2 points) Create a directed graph with an edge from u to v if a single jump from u takes us to v.
   (1 point) We need the shortest path in this graph from A to B
   (2 points) For this we can use the BFS algorithm
   (1 point) started from A and without restarting it from unvisited vertices. The distance of B is what we need.

(1+1 points) Correctness: a sequence of jumps is a path in our graph, and length of the path is equal to the number of jumps.
(1+1 points) Running Time: We must create an adjacency list (a matrix won't work) of the graph and since each vertex has outdegree 2, the graph has $2n$ edges, and creating the list takes $O(n)$ time. Running BFS on it takes $O(n + 2n) = O(n)$ time.

A maximum of 4 points can be awarded if Dijkstra's or DFS is used instead of BFS.

3. We run the DFS (depth-first search) on a simple undirected graph G with eight vertices, and select the following edges in the DFS tree in this order: $ab, be, ac, cd, dg, dh, cf$. What is the minimum possible value of the degree of vertex c in G and what is its maximum possible value?

   Scoring guide:
   (5 points) Table for DFS with discovery and finishing times, along with the DFS tree. Points are also awarded if the discovery and finishing times are written on the tree and not in a table.
   (1 point) Minimum degree is 3 as c has neigbors a, d, f in the DFS tree.
   (1 point) Graph is simple, so cannot have more edges to a, d, f.
   (1 point) There cannot be edges between c and b, e because they were finished before c is discovered.
   (1 point) There can be edges between c and g, h, since they have been discovered before we finished c, so these edges will not be a part of the tree.
   (1 point) So the maximum degree is 5.

4. We are given the adjacency matrix of a simple, directed, weighted graph on n vertices with no directed cycles. Some of the vertices of the graph contain a giant chocolate bar while the others are empty. Give an algorithm with a running time of $O(n^2)$, that determines for each vertex the distance of the nearest chocolate (the distance of the nearest chocolate is equal to the length of the shortest path to a vertex containing a chocolate).

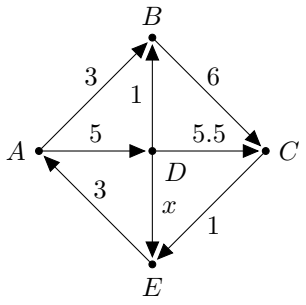
   Scoring guide
   (2 points) The direction of all the edges must be reversed.
   (1 point) we must add a new starting vertex v_0 which has a 0-weight edge to all vertices with a chocolate.
   (2 points) we need to run the DAG-SHORTEST-PATHS algorithm here with source as the new vertex v_0.
   (1+1+1 points) Correctness: the paths in the original graph from a vertex u to a vertex v with a cholocate will translate to a path from v_0 to v with the 0 weight edge, and then the path to u. And since the weight of edge from v_0 is 0, length of these paths is the same. And since the original graph is DAG, this is also, since the new vertex has no incoming edges and can't make a cycle.
   (1+1 points) Running Time: constructing the new graph is $O(n^2)$ and the DAG-Shortest path is $O((n + 1)^2) = O(n^2)$ (because its an adjacency matrix). So the total running time is $O(n^2)$.

   If Dijkstra's is run instead of DAG-ShortestPath, then given every other step is correct, only one point is deducted.

   If the DAG is run from every vertex with chocolate, instead of introducing a new source, then a maximum of 4 points are awarded.

   If only DAG is run but edges are not reversed etc, then a maximum of 2 points are awarded.

5. Run the Dijsktra's algorithm to determine the length of the shortest paths from vertex A to all other vertices in the graph given below as a function of the positive real parameter x. It is not necessary to justify, but show the Q table updates, and the tables for the final distances and parents.



Scoring guide:
(1 point) Q table proper initialization
(1 point) Q table proper updates after B is processed.
(2 points) Q table proper updates after D is processed.
(1 point) We need to consider cases based on x
(2 points) Correct analysis for $x < 4$
(3 points) Correct analysis for $x \geq 4$

6. We are given a country's road network as the adjacency matrix of an undirected weighted graph: vertices are the cities (n cities), edges are the direct roads between cities, and the weight of the edges is the expected time taken to travel the road. Given that some cities in the country have an ambulance station, we want to decide whether it is true that any city in the country can be reached from at least one ambulance station in at most M minutes.
Which algorithm can be used to solve this problem in $O(n^2)$ running time, and how?

Scoring guide:
(2 points) We need a new vertex v_0 with a 0 weight edge to all cities with an ambulance station.
(2 points) We run Dijkstra's with v_0 as the source.
(1 point) and then we check if the minimum distance to every vertex is at most M.
(1+1 point) Correctness: the weights are positive, so we can use Dijktra's algorithm. correctness of Dijkstra's gives us the shortest paths, and since v_0 is only connected to cities with ambulance, we only find paths from such a vertex.
(1+1+1 point) Modifying the adjacency matrix is $O(n^2)$, Dijkstra's is $O((n+1)^2) = O(n^2)$. Checking if distance is M to every city is $O(n)$.

Maximum of 5 points are awarded if some other algorithm is used and not Dijkstra's.

7. In a table of size $n \times n$, we move from the top left corner cell to the bottom right corner cell of the table by moving to an adjacent cell either below or to the right in each step. Some of the cells in the table are dangerous (these are known to us) and we can only step on at most one dangerous cell on the way. Furthermore, each cell of the table contains an integer and the value of a path is the sum of the numbers on the cells in the path.

Give a dynamic programming algorithm with running time $O(n^2)$ that determines the value of the path with the largest value that obeys the above rules.

Scoring guide

(1 point) Subproblems
(1 point) Start
(4 points) Continue
(1 point) End
(1 point) Correctness

(2 points) Running time