Part I

Question 1 (25 points)

This exercise is about finding a bounded search tree algorithm[1] for the problem of *minimum hyperplane cover of points in space*. This problem is defined as follows: Given a set of n points in a d-dimensional space and an integer k, is there a set of k or less hyper-planes that cover (go through) all points?

(**Hint**: in your answers you may assume that checking whether n points lie on a hyperplane in dimension d can be done in $O(nd)$.)

(a) (2 points) Consider the following instance of this problem. Given the following set of five points $\{(0,0),(2,0),(1,1)(0,2),(2,2)\}$ in the two-dimensional plane. Does a set of at most two lines exist that covers all points? If so, give such a set of lines. If not, explain why this is impossible. Note that a line can be defined by giving any two of the points it covers. Therefore, define each line by giving two points that are on this line.

(b) (3 points) Describe an (exponential) brute-force algorithm (or give pseudocode) to solve the problem for $d = 2$ *and* give an upper bound on its run-time complexity.

(c) (3 points) Describe a simple $O(d \cdot k)$ solution (or give pseudocode) for the case that $d \cdot k \geq n$.

(d) (4 points) Now consider the case that $d \cdot k < n$. A hyper-plane of dimension d is uniquely defined by at most d points on this plane. Describe an (exponential) brute-force algorithm for the problem of minimum hyperplane cover of points in space (or give pseudocode) *and* give an upper bound on its run-time complexity.

(e) (9 points) Describe a bounded search tree algorithm for this problem (or give pseudocode).

(f) (4 points) Analyze the run-time complexity of the bounded search tree solution. Give

1. the run-time for one node of the search tree,
2. the fan-out (number of branches) of each node,
3. the depth of the search tree, and
4. the run-time for the complete tree.

Give a brief (at most 2 lines) explanation for each of these.

[1] Bounded search tree algorithms are introduced on pages 38–42 and 62–65 of the reader.

Question 2 (25 points)

Some major problems in applications of graphs concern the finding of maximal subsets of edges that satisfy a specific property. This question is about finding maximal subsets of edges that are acyclic.

Consider an undirected weighted graph $G = (V, E, w)$ where the weight function $w : E \rightarrow \mathbb{Z}^+$ associates a positive weight $w(e) > 0$ to every edge $e \in E$. A subset E' of E is called *acyclic* if the graph $G' = (V, E')$ generated by E' is acyclic.

We would like to find a *maximal* acyclic subset E^* of E, that is an acyclic subset E^* of E such that its total weight $w(E^*) = \Sigma_{e \in E^*} w(e)$ is maximal among all acyclic subsets E' of E.

(a) (6 points) Show that $(E, \mathcal{I})$ is a matroid, where $I \in \mathcal{I}$ iff I is an acyclic subset of E;
(**Hint**: a maximal acyclic subset of edges of an undirected graph is a forest of spanning trees. Remember that a spanning tree with n nodes consists of $n - 1$ edges.)

(b) (3 points) Give a polynomial algorithm to compute an acyclic subset of maximal weight.

(c) (7 points) Explain why you can't use matroids to compute a maximal acyclic subset of *directed* graphs.

(d) (9 points) Give a polynomial 2-approximation algorithm for computing a cardinality maximal acyclic subgraph of a *directed* graph $G = (V, A)$ where each edge has the same weight.
(**Hint**: consider an arbitrary ordering of the set V and consider the subset A_1 of edges $(u, v) \in A$ such that u precedes v in the ordering and the subset A_2 of edges $(u, v) \in A$ such that v precedes u in the ordering.)

Part II

Question 3 (25 points)

Given the 2-CNF consisting of the clauses:

$$(p \vee q) \wedge (p \vee \neg q) \wedge (\neg p \vee q) \wedge (\neg p \vee \neg q) \wedge (r \vee s) \wedge (r \vee \neg s) \wedge (\neg r \vee s) \wedge (\neg r \vee \neg s)$$

(a) (5 points) Evaluate first the homogeneous quadratic function F in variables p and q, to be optimized in the Goemans-Williamson relaxation of the Max-2-SAT problem for $(p \vee q) \wedge (\neg p \vee \neg q)$.

(b) (5 points) Evaluate the homogeneous quadratic function G in variables p, q, r and s, to be optimized in the Goemans-Williamson relaxation of the Max-2-SAT problem for the whole instance, and simplify this function as much as possible.

(c) (5 points) In the general case, does the SDP relaxation provide an upper bound or a lower bound? Please, motivate.

(d) (10 points) How is the situation in this specific example?

Question 4 (25 points)

In the SUBSET-SUM problem (see e.g. Kleinberg & Tardos, section 6.4), we are given a set $S = \{1, 2, \ldots, n\}$ of n items with nonnegative integer weights w_i (for $i = 1, \ldots, n$), and an integer W. We are asked to select a subset $T \subseteq S$ that maximizes the sum of the weights in the subset, without exceeding W. We represent such a solution T as a bitstring $t = t_1 t_2 \cdots t_n$, where, for $i = 1, 2, \ldots, n$, $t_i = 1$ if $i \in T$ and $t_i = 0$ if $i \notin T$.

(a) (12 points) Describe an algorithm that uses Tabu Search as presented during Lecture 13 to solve given instances of the SUBSET-SUM problem (or present it using pseudocode).

(b) (6 points) Consider the SUBSET-SUM instance defined by the weights $w = (2, 5, 8, 11, 12, 18)$ and the bound $W = 22$. Run deterministic iterative improvement using steepest descent as a neighbor selection rule with a suitably defined cost function. Let the algorithm start with the bitstring $s = 111111$, i.e., the set containing all items.

(c) (7 points) Give three iterations of your Tabu Search algorithm when run on this instance, starting from the bitstring 101100. Specify for each iteration the current solution, the neighborhood according to your choices, the next solution according to your algorithm, and any further relevant information (depending on your choices).

End of test