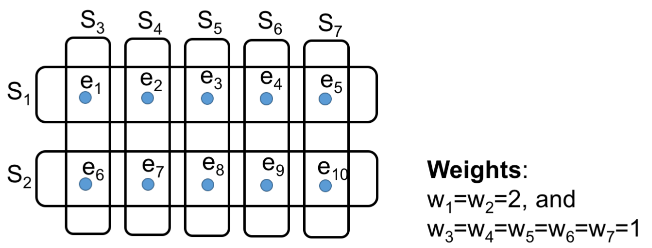
Exam Advanced Algorithms, 27 October 2106, 12.00-14.45

- It is not allowed to use any books, notes, or calculator. Just pen and paper.
- Solutions will be on BB after the exam.
- The table shows the maximum number of points per (sub-)question.

Question	1a	1b	1c	2a	2b	3a	3b	4	5a	5b	5c	6	Σ
Points	2	2	4	4	6	4	6	6	2	6	2	6	50

1. The figure below is an instance of the weighted set cover problem. In fact, it is an instance of the weighted vertex cover problem since each element appears in exactly two sets.



(a) Draw the graph which corresponds to this vertex cover instance.

Chapter 1 gives a primal-dual algorithm for the weighted set cover problem. Remember that this algorithm constructs the primal- and dual solution step by step without actually solving the dual to optimality.

(b) Formulate this dual for this instance.

(c) Now apply the primal-dual algorithm to this instance. Describe how the values of the primal and dual variables change in each iteration. Also, give the value of the solution found. (There is more than one possible outcome. You only need to give one.)

2. Chapter 2 describes the List Scheduling algorithm for minimizing the makespan (length) on identical parallel machines.

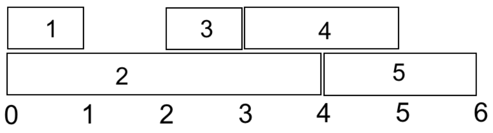
(a) Prove that List Scheduling is a 2-approximation algorithm.

Now assume that each job j has a release time r_j, that means, job j cannot start before time r_j. We adjust the list scheduling algorithm as follows:

List scheduling with release times
Step 1: Label the jobs in order of release time: $r_1 \leqslant r_2 \leqslant \cdots \leqslant r_n$.
Step 2: Apply list scheduling in this order.

Let T_j be the completion time of the machine that completes earliest in the list schedule for the first $j-1$ jobs. Then, step 2 can be described as follows. If $T_j \leqslant r_j$ then start job j at time $s_j = r_j$ on some machine and start job j at time $s_j = T_j$ otherwise.

Example:



	1	2	3	4	5
p_j	1	4	1	2	2
r_j	0	0	2	2	3
T_j	0	0	1	3	4
s_j	0	0	2	3	4

(b) Prove that list scheduling is a 2-approximation algorithm for minimizing makespan under release time constraints.
Hint: Let k be the job that completes last. Show that no machine is idle between r_k and s_k.

3. The following problem is closely related to the Knapsack Problem:

SUBSET SUM:

Instance:	A number B and n items $1,2,\dots,n$ where item i has size $s_i \leqslant B$. All numbers are positive integers.
Solution::	$S \subseteq \{1,\dots,n\}$ of items such that $\sum_{i\in S} s_i \leqslant B$
Value:	The total size of the solution, i.e., $\sum_{i\in S} s_i$
Goal:	Maximize the value.

(a) Give a Dynamic Program (DP) that solves the subset sum problem in $O(nB)$ time.

(b) Give a polynomial time approximation scheme (PTAS) for the problem. *Hint: Partition the items into* large *and* small *items and use (a).*

4. Chapter 4 shows a 4-approximation algorithm for the uncapacitated facility location problem (UFL). There, it is assumed that the distances satisfy the triangle inequality. The approximation guarantee becomes much worse if the triangle inequality is not assumed.

It is known that the set cover problem cannot be approximated better than $O(\log n)$, where n is the number of elements. In other words, there exists some (small) constant c such that there is no $c \log n$-approximation algorithm for set cover, assuming $P \neq NP$.

Use this fact about set cover to show that the UFL without triangle inequality cannot be approximated better than $O(\log |D|)$, assuming $P \neq NP$, where, $|D|$ is the number of clients in the UFL instance.

Hint: Make a reduction from set cover to UFL with $|D| = n$ clients. Describe the facility openings cost f_i and the connection costs. Explain your reduction by drawing a small example.

5. Consider the following generalization of the maximum weighted cut problem:

MAXIMUM WEIGHTED k-CUT:

Instance:	A graph $G = (V, E)$ and a weight w_e for all $e \in E$. Number k.
Solution::	A partition of V into k sets: $S_1, S_2, \ldots, S_k$.
Value:	The total weight of edges between different sets of the partition, i.e., $\sum_{e \in C} w_e$ where $C = \{(i, j) \in E \mid i \text{ and } j \text{ are in different sets}\}$
Goal:	Maximize value.

(a) Give a randomized $(1 - \frac{1}{k})$-approximation algorithm for this problem and give a proof for this ratio.

Now suppose that we restrict this maximum k-cut problem to graphs which are 3-colorable. That means, an instance is given by an edge-weighted 3-colorable graph and a number k. (NB. The 3-coloring itself is not given since otherwise the problem is trivial for $k \geqslant 3$.)

(b) Show how to use the vector programming approach of Chapter 6 to get a randomized $8/9$ approximation when $k = 4$.

(c) Next, generalize (b) to any k. To simplify the analysis you may assume that $k = 2^q$ for some integer q. Show that this approach gives a $(1 - k^{-\log_2 3}) \approx (1 - 1/k^{1.58})$-approximation.

6. Consider the following variant of the Satisfiability problem (SAT). We are given m clauses, each containing exactly 3 positive variables and no negative variables. Example:

$$(x_1 \vee x_2 \vee x_3),\ (x_1 \vee x_2 \vee x_5),\ (x_3 \vee x_4 \vee x_5),\ (x_2 \vee x_5 \vee x_6),\ (x_5 \vee x_6 \vee x_7).$$

Clearly, one can satisfy all clauses by giving all variables the value `True`. The problem, however, is to do this for a minimum number of variables. In the example, the optimal value is 2: Set $x_1 = x_5 =$ `True` and set the others to `False`.

VARIANT OF SAT:

Instance:	n variables and m clauses of exactly 3 positive variables each.
Solution::	A subset $S \subseteq \{1, \ldots, n\}$ such the assignment which sets $x_i =$ `True` for $i \in S$ and $x_i =$ `False` for $i \notin S$ satisfies all clauses.
Value:	$\lvert S \rvert$
Goal:	Minimize the value.

Give a 3-approximation algorithm for this problem. (NB. If you make use of some known approximation algorithm then also include the proof for the ratio of that algorithm).