


Please read carefully and start with answering those questions that you are sure you can solve. Write clearly, always state which exercise you are answering when using extra sheets, and indicate by crossing out if you want parts of your answer to be disregarded. Any ambiguity or unclarity will result in zero points, or in the worst of several solutions being graded. If you are uncertain of a formal statement or proof, write it down informally rather than giving no solution.

Note: you can write all solutions on the exam sheet, there is enough space reserved after each exercise. If you prefer or need extra space, you can also use exam paper. The exam has a total of 100 points.

Best Wishes!

Note: The grading scheme of the actual exam can differ! This is an example ONLY.

Question 1 (12 points)

Which of the following statements are true given the assumptions in the following parts? List the implied statements and give a short reason for your decision. 

a) (4 points) Let M be a stable matching between residents and hospitals, computed by the Gale-Shapley algorithm.

1) There exists no partial matching M' of any sets of residents R and hospitals H such that every vertex in R and H prefers their partner in M' over their partner in M .

False, since G.-S. only guarantees stability, defined as 'no two vertices r and h can make a profitable switch'

2) As a participant in the matching process, it is more favorable to be on the side that accepts matching offers than on the side that has to propose.

False, we showed G.-S. computes a stable matching that is best for the proposers but worst for the other side.


b) (4 points) Let G be an undirected graph, where the A^* algorithm is used to compute a shortest path from vertex s to vertex t .

1) The algorithm considers no other edges of the graph than the ones that are part of a shortest path from s to t .

False. When using the O -function as heuristic, it even considers all of them (like Dijkstra).

2) You have moved from house a to house s , and already know the best way to go from a to anywhere else in the city. $h(x) = \text{dist}(a,x) + \text{dist}(a,t)$ is a good heuristic to use in

the algorithm, where $dist$ is the function representing your known routes from and to a .

False. This heuristic is not monotone  and not even admissible (might be shorter to not go via a).

c) (4 points) Assume that decision problem $L \in NP$.

1) Then, there exists no polynomial time reduction from SAT (the satisfiability problem) to L .

False: If L is NP-complete, then L is even guaranteed to be 'as hard as SAT , up to polyn. effort'.

2) Then, it must be unknown whether there exists a polynomial time algorithm for L .

False. Since $P \subseteq NP$, L might be in P .

Question 2 (14 points)

Name two algorithmic paradigms/modeling techniques from the lecture that should be useful to solve the following problems optimally. For each, give an algorithm or a problem representation that does so (e.g., a greedy procedure, a formula for DynProg, a Flow Network) You do **not** need to state a proof of correctness.

Hint: those techniques include Greedy, Dynamic Programming, Informed Search, Network Flows, Linear Programs, Integer Linear Programs.

a) (6 points) You bought a new ventilation system for your house, which is very strong and can deliver fresh air to each of your n rooms R . However, you need to build a system of pipes such that for every room, there exists some pipe connection to the basement $b \in R$, where the ventilation system is located. Any pipe can be used for arbitrarily many rooms at the same time, and you know the costs for installing a pipe between any $(r_1, r_2) \in R \times R$. What is the minimum cost to get the ventilation system to work?

Greedy:

- Model the problem as graph G , vertices = rooms, edge weights $w(e) = \text{costs to connect two endpoints (rooms)}$
- Start at $V' = \{\text{basement}\}$. As long as there are rooms in $V \setminus V'$:
 - Find smallest $w(u, v) \in E$ s.t. $u \in V'$, $v \notin V'$.
Add v to V' and add (u, v) to the solution set S .

ILP: Let $E(V') = \{(u,v) \in E \text{ s.t. } u,v \in V'\}$.

min $\sum_e x_e$ s.t.

$$\sum_{e \in E(V')} x_e = |V'| - 1, \forall V' \subseteq S, |V'| \geq 1. \quad (*)$$

$$x_e \in \{0,1\}, \forall e \in E.$$

unnecessary { (*) no subset of the vertices can have a cycle, which would be $|V'|$ edges.
The maximum number of edges in a cycle-free graph is always $n-1$ if the graph is connected, and this can only be reached by connecting all nodes in the same tree.

b) (8 points) The Knapsack Problem, i.e. given a number $K \geq 0$ and n items I with costs c_i and values v_i , choose a subset $S \subseteq I$ with $\sum_{i \in S} c_i \leq K$ that maximizes $\sum_{i \in S} v_i$.



Dyn Prog: For $i > 0$:

$$\text{OPT}(i, k) = \begin{cases} \max \{ \text{OPT}(i-1, k), (\text{OPT}(i-1, k - c_i) + v_i) \} & \text{if } c_i \leq k \\ \text{OPT}(i-1, k) & \text{otherwise.} \end{cases}$$

$$\text{OPT}(1, k) = \begin{cases} v_1 & \text{if } c_1 \leq k \\ 0 & \text{otherwise.} \end{cases}$$

ILP:

$$\max \sum_{i \in [n]} v_i x_i \quad \text{s.t.}$$

$$\sum_{i \in [n]} c_i x_i \leq K$$

$$x_i \in \{0,1\} \quad \forall i \in [n].$$

Question 3 (8 points)

Recall that given an undirected graph $G = (V, E)$, a set $A \subseteq V$ is called *dominating* if for all $v \in V$ we have that $v \in A$ or there is a $u \in A$ such that $\{u, v\} \in E$. The dominating set problem is defined as follows: given a graph G and a natural number k , does G have a dominating set of size k ?

The set cover problem is formulated as follows: Given a set U , a natural number n and a collection $\{S_1, S_2, \dots, S_m\}$ with each $S_i \subseteq U$, is there a set of indices $I \subseteq \{1, 2, \dots, m\}$ of size n such that $\bigcup_{i \in I} S_i = U$?



Prove that dominating set \leq_p set cover.

Goal: Polytime - Computable $f: (k \in \mathbb{N}, \text{Graph } G) \rightarrow (n \in \mathbb{N}, U, S_1, \dots, S_m)$
such that $(k, G) \in \text{Dominating Set} \Leftrightarrow (n, U, S_1, \dots, S_m) \in \text{Set Cover}$.

Define $U = V$, and $n = k$.

$\forall v_i \in V: S_i = \{v_i \cup \underbrace{N(v_i)}_{\text{neighborhood of } v_i, \text{ i.e. } \{v \in V \mid \exists e = (v, v_i) \in E\}}\}$ (therefore, $m = |V|$)

Proof: Assume $(k, G) \in \text{DomSet}$.

$\Rightarrow (\exists A, |A| = k, \forall v \in V: v \in A \text{ or } \exists e = (u, v) \in E: u \in A.)$

Then, for this set A : Define $I = \{i \in \mathbb{N}: v_i \in A\}$.

$\Rightarrow \forall v \in V: v \in \bigcup_{i \in I} S_i \Rightarrow \bigcup_{i \in I} S_i = U \Rightarrow (k, V, S_1, \dots, S_m) \in \text{SC}.$

Also:

Assume $(k, V, S_1, \dots, S_m) \in \text{Set Cover}$.

$\Rightarrow \exists \text{ set } I \text{ such that } |I| = k, V = \bigcup_{i \in I} S_i.$

Define $A = \{v_i \in V \mid i \in I\}$.

Fix any $v \in V$.

I is a set cover $\Rightarrow \exists i^* \in I: v \in S_{i^*}$

Also, $v_{i^*} \in A$. By def. of S_{i^*} , it holds $v_{i^*} \in N(v)$, or $v_{i^*} = v$.

$\Rightarrow A$ is a dominating set.

Finally, computing (k, V, S_1, \dots, S_m) only requires finding the neighbors of each vertex, which is possible in $O(n^2) \Rightarrow \text{Polytime}.$

□

Question 4 (14 points)

Recall that a propositional logic formula φ is in 2-CNF if it is of the form $(\ell_{1,1} \vee \ell_{1,2}) \wedge (\ell_{2,1} \vee \ell_{2,2}) \wedge \dots \wedge (\ell_{m,1} \vee \ell_{m,2})$, where each ℓ_{ij} is a literal. Show that deciding whether there exists a satisfying assignment for a formula in 2-CNF is in P, or prove that it is NP-hard.

Hint: Finding a graph representation for the problem could help.

Graph:

Vertices V for each variable in true / negated form

$$\begin{matrix} x_1 \\ \bar{x}_1 \end{matrix} \quad \begin{matrix} x_2 \\ \bar{x}_2 \end{matrix} \quad \dots \quad \begin{matrix} x_n \\ \bar{x}_n \end{matrix} \quad V = \{x_1, \bar{x}_1, \dots, x_n, \bar{x}_n\}$$

each clause has only 2 literals \Rightarrow if the 1st is wrong, the 2nd must be right! Add edges for clauses depicting these implications:

$$(x_i, \bar{x}_j) \leadsto \begin{matrix} \bar{x}_i & \longrightarrow & \bar{x}_j \\ x_j & \longrightarrow & x_i \end{matrix} \quad \forall C = (\ell_{ij}, \ell_{ik}), (\bar{\ell}_{ij}, \ell_{ik}) \in E \text{ and } (\bar{\ell}_{ik}, \ell_{ij}) \in E.$$

Algorithm:

(1) Pick unmarked vertex $v \in G$ s.t. \bar{v} is also unmarked.

Mark v as 'tent'.

Let $V_v = \{v' \in V \mid \exists \text{ path from } v \text{ to } v' \text{ in dir. Graph } G\}$.

For all $v' \in V_v$:

If v' unmarked, mark v' as 'tent'

If for all $u \in V$, either u or \bar{u} is unmarked:

Change all 'tent' markings in G to 'true'.

Else: change all 'tent' in G into 'unmarked'.

Set v to \bar{v} , and do (*) with this new v .

IF there exists a pair $u, \bar{u} \in V$ s.t. both u, \bar{u} marked:

output UNSATISFIABLE.

ELSE Go to (1).

// clearly polytime:
runs BFS/DFS
 $O(|V|)$ times

The algorithm outputs a legal choice of variables (vertices marked 'true'), or outputs 'UNSAT.'. Indeed, if the algo does not output 'UNSAT.', then all clauses are satisfied since every variable has exactly 1 'true' vert. x and for any clause that contains \bar{x} , the other literal y satisfies it since $\exists \text{ edge } x \rightarrow y$.

On the other hand, if 'UNSAT.' is the result, there exists some variable which the alg. tried both x and \bar{x} , but both resulted in a contradiction. ^{for} By construction, this can not be due to wrong choices the algo made earlier: Even if it encounters a 'true' node at some point after choosing $x=v$ or $\bar{x}=v$, all implications of said node were already checked. Ergo, the contrad. is indeed caused by the input instance.

Question 5 (10 points)

Design an algorithm that given a Graph $G = (V, E)$ and a weight function $w : E \rightarrow \mathbb{R}$, finds a matching of maximum weight: that is a set $A \subseteq E$ such that $w(A)$ is maximised. Analyse its runtime and argue for correctness, and in case of a non-optimal algorithm, prove an approximation bound.

Alg.: As long as $E \neq \emptyset$:

- find $(u, v) \in E$ with maximum weight, add (u, v) to matching M .
- erase u, v , and all edges incident to u or v from G

Corr.: The output is a matching since all vertices are erased after the 1st incid. edge is chosen. The algo terminates since it deletes 2 vertices in each iteration.

Runtime: $O(|E|)$ to find max edge
 $O(|E|)$ to find if edges contain u, v and erase
 $O(|V|)$ iterations of the above
 $\rightarrow O(|V||E|)$ (can be optimized of course)

Approx. Guarantee:

Let OPT denote an opt. matching and $G_i = (V_i, E_i)$ the graph after iteration i . Let M_i be the matching after iteration i .

It holds $OPT \cap E_0 = OPT$, and therefore $w(OPT_0) + \underbrace{w(M_0)}_{=0} = w(OPT)$

It also holds that for all i ,

$$w(OPT_i) - w(OPT_{i+1}) \leq 2(w(M_{i+1}) - w(M_i)). \quad (*)$$

Proof: This is true for any $i, i+1$ since G_i results from G_0 by erasing two vertices and their edges, therefore OPT_i has at most 2 edges more than OPT_{i+1} . Also, both these edges are of weight $\leq w(u, v)$ for $(u, v) \in M_{i+1} \setminus M_i$.

It follows, since in every step, M gains at least half what OPT loses, and $OPT_{\frac{n}{2}} = \emptyset$, that $2ALG \geq 2w(OPT)$. The alg. is a 2-approximation.

Question 6 (8 points)



a) (2 points) Show $n^2 \notin \Omega(3n^3)$.

Fix any constant C , $n_0 \in \mathbb{N}$. Disprove: $\forall n \geq n_0, n^2 \geq C 3n^3$
Pick n s.t. $n > \frac{1}{3C}$.
 $\Rightarrow C 3n^3 > C 3 \frac{1}{3C} n^2 = n^2 \Rightarrow C 3n^3 > n^2 \quad \nexists$

b) (2 points) Give the definition of a PTAS.

A family of algorithms ALG_ϵ for any $\epsilon > 0$ such that

- ALG_ϵ is a $(1-\epsilon)$ approximation to the opt. sol. of Maxim. Problem P ,
i.e. outputs feasible sol. of value $\geq (1-\epsilon)OPT$ (or $\geq (1+\epsilon)OPT$ for min.)
- ALG_ϵ runs in polynomial time in the input size

c) (2 points) Define the class NP .

All decision problems P for which there exists a polytime fct C (certifier)
such that $(\exists t$ with $|t|$ polynomial and $C(I, t) = \text{TRUE}$)
if and only if $I \in P$.
 \uparrow in $|I|$

d) (2 points) Is it known whether the maximum matching problem is NP -hard? Give a short reason.

No. If $P=NP$, then every problem in P is NP -hard.
Max Matching $\in P$.

Question 7 (10 points)



a) (5 points) Give an input instance (with n vertices, for any $n \in \mathbb{N}_{\geq 2}$) to the k -center problem such that the greedy algorithm from the lecture, given that it starts with picking the first point in the input set V , always yields an optimal solution.

Easy back: $v_1 = v_2 = \dots = v_n$.

Intended (Ex.):

$\forall i \in \mathbb{N}$ define $x_i = i$. Define distance function $\text{dist}(x_i, x_j) = \begin{cases} 1 & \text{if } x_i \neq x_j \\ 0 & \text{otherwise} \end{cases}$

This results in $OPT=0$ iff the metric space has more than k points.

Otherwise, picking any centers gives $ALG = OPT = 1$.

b) (5 points) Give a running time analysis for the k -center algorithm from the lecture (in general, not for the special instance constructed here).

- Finding the max-distance point: $O(n)$
- Recomputing / updating all distances: $O(n)$
- Repeating the above k times $\leadsto O(kn)$

Question 8 (13 points)

Given two numbers $2 \leq a \leq b$, and two operations: P (for Plus 1) and D (for Double), where $P(x) = x + 1$ and $D(x) = 2x$. Design a greedy algorithm to find the least amount of operations, starting from a , to get to b . Show its correctness. For example, if $a = 2$ and $b = 10$, the order would be DPD : $D(a) = 4, P(4) = 5$, and $D(5) = 10$.

Algorithm:

While $b \geq 2a$:

IF b even: Set $b = \frac{b}{2}$ and add 'D' to the beginning of the sequence
ELSE Set $b = b - 1$ and add 'P' to the beginning of the sequence

While $b > a$, Set $b = b - 1$ and add 'P' " " " .

Proof:

correctness { The algo (1st loop) computes $\frac{b}{2^{i^*}}$ with $i^* = \max i \in \mathbb{N} : \frac{b}{2^i} \geq a$.
IF $\frac{b}{2^{i^*}} > a$, it then reduces by 1 (2nd loop) until it reaches a .
Since the sequence of adding and doubling steps exactly inverts each step of the algorithm, the output is indeed a sequence of operations that yields a . Since $b < \infty$, it always terminates.

optimality { Clearly, doubling is better than adding for any numbers > 0 : fewer oper.!
So we need to minimize the number of 'P'-operations. It holds for any c that $a \leq \frac{1}{c}b \iff ca \leq b$, so the number i^* is also the number of doubling steps we get when starting the 1st loop at a instead of b .
Since starting at b , each doubled number is larger than from a , the alg is optimal.

Question 9 (14 points)

A company has r customers and q consultants. Every customer needs to be assigned to exactly one consultant. Every consultant can only be assigned to z customers. Finally, no consultant is allowed to drive more than 50 kilometers to any of their assigned customers; you may assume a table with all customer-consultant distances is given. You want to decide if there is a consultant-customer assignment that meets all of these demands.

- a) (8 points) Model this problem as a maximum flow problem. Describe the graph you construct and argue why there exists a flow of value r in this graph if and only if a consultant-customer assignment exists.

$F = (V, E, c)$, $c : E \rightarrow \mathbb{N}_{\geq 0}$, such that

$V = R \cup Q \cup \{s\} \cup \{t\}$, where R = customers, Q = consultants,

$E = \{(q, r) \in Q \times R \text{ s.t. } \text{dist}(r, q) \leq 50\}$

$\cup \{(s, q) \mid q \in Q\} \cup \{(r, t) \mid r \in R\}$, and

$$c(e) = \begin{cases} z & \text{if } e = (s, q) \text{ for } q \in Q \\ 1 & \text{otherwise} \end{cases}$$

Flow \Rightarrow Assignment

\Rightarrow r -flow \Rightarrow all (r, t) edges satur., no q -vertex has flow $> z$, no flow between cons. / customers with dist > 50 (since there is no edge)
 \Rightarrow assigning every cust. the consultant that sends him flow is feas.

Assignment \Rightarrow Flow

\Rightarrow feasible ass. \Rightarrow send flow 1 on each (r, q) edge s.t. r ass. to q .
 send flow k_q = number of cust. of cons. q on edge $(s, q) \forall q \in Q$ ($\leq z$!).
 send flow 1 on each edge (r, t) , $r \in R$. This flow is feasible.

- b) (6 points) Argue how this problem could be solved by finding a bipartite matching. Describe the graph you construct.

$G = (Q', R, E)$ where R customers, and Q' contains z copies of each consultant. $E = \{(q', r) \mid q' \in Q', r \in R, \text{dist}(q, r) \leq 50 \text{ for } q \text{ the cons. that } q' \text{ is a copy of.}\}$

Now, a max. bip. matching matches the highest-poss. number of cust. possible without exceeding the 50 km or the z -limit.
 If the size of such a matching is exactly R , the problem is solvable.