

General Information (that was already listed at the homepage)

It is not allowed to share answers!

- You have 2 hours to solve the exam and you get additional 20 minutes to upload your solutions.
- All single pages of your solutions must be clearly marked with your StudentID and name (no anonymity code).
- Upload your solutions **not later than March 19 - 3:20pm** at the bottom of the DA3004 homepage": <https://kurser.math.su.se/course/view.php?id=1019> or using the link <https://kurser.math.su.se/mod/assign/view.php?id=61716>.
ONLY IN CASE something does not work with the upload, you can also send your solutions via email to marc.hellmuth@math.su.se.
- In case you have questions during the exam, use the zoom-line <https://stockholmuniversity.zoom.us/j/65349503510>
- In total, you can get 100 points and you need to get at least 55 points to pass the exam.

Problem	1	2	3	4	5	6	7	8	TOTAL
Points									

Problem 1 (Runtime)

7+5+5=17p

- (a) In big-O analysis we sometimes use $n^2 \leq 2^n$ for all integers $n \geq 4$. Prove this statement.
 (b) Show that $T(n) = (n+2)^3 \in \Theta(n^3)$ (in "big-Theta").
 (c) Consider the following recursive algorithm for computing the sum $S(n) = \sum_{i=1}^n i^3$.

$S(\text{positive integer } n)$
 1: **if** $n = 1$ **then return** 1
 2: **else return** $S(n-1) + n * n * n$.

Determine the runtime in big-O notation (best possible bound) assuming that basic operations as return, addition and multiplication can be done in constant time. Explain your results.

(a) Induction: $n=4: 4^2 = 16 \leq 2^4 = 16 \quad \checkmark$
 true for $n=k: k^2 \leq 2^k$
 $n=k+1:$

$$2^{k+1} = 2 \cdot 2^k \underset{\substack{\text{ind} \\ \text{hyp.}}}{\geq} 2k^2 = k^2 + k^2 \underset{n \geq 4}{\geq} k^2 + 4k = k^2 + 2k + 2k$$

$$\underset{k \geq 4}{\geq} k^2 + 2k + 8 = k^2 + 2k + 1 = (k+1)^2 \quad \square$$

(b) to show: $(n+2)^3 \in \Omega(n^3)$
 $\in O(n^3)$

$$(n+2)^3 = n^3 + 6n^2 + 12n + 8 = T(n)$$

$$T(n) \leq n^3 + 6n^3 + 12n^3 + 8n^3 = 27n^3 \quad \forall n \geq 1 \Rightarrow T(n) \in O(n^3)$$

$$T(n) \geq n^3 \quad \forall n \geq 1 \Rightarrow T(n) \in \Omega(n^3)$$

$$\Rightarrow T(n) \in \Theta(n^3)$$

c) runtime: $T(n) = T(n-1) + c \quad // \text{1st step}$
 $= (T(n-2) + c) + c \quad // \text{2nd step}$
 \vdots
 $= T(n-i) + i \cdot c \quad // i\text{-th step}$
 $= T(1) + (n-1) \cdot c \quad \text{last step}$
 $\in O(n)$.

Problem 2 (Complexity)

5+5+5=15p

Suppose we are given a decision problem A that has as one of the inputs an arbitrary graph. Let \mathcal{U} denote the set of all undirected graphs and \mathcal{D} denote the set of all directed graphs. Note, we can consider undirected graphs as a special case of directed graph by replacing all edges $\{u, v\}$ by two arcs $(u, v), (v, u)$ and thus, we can assume that $\mathcal{U} \subseteq \mathcal{D}$.

- (a) Assume we have shown NP-hardness of A by reduction from 3-SAT by constructing a special directed graph $G \in \mathcal{D} \setminus \mathcal{U}$. Explain shortly, if this implies that A is NP-hard for the class of graphs in \mathcal{U} ?
- (b) Assume we have shown NP-hardness of A by reduction from 3-SAT by constructing a special undirected graph $G \in \mathcal{U}$. Explain shortly, if this implies that A is NP-hard for the class of graphs in \mathcal{D} ?
- (c) Briefly discuss the following statement: There is no polynomial-time algorithm for NP-hard problems.

a) We don't know whether the problem remains NP-hard for graphs in \mathcal{U} , since NP-hardness was shown only for graphs that are not contained in \mathcal{U} .

b) Yes, this implies NP-hardness for graphs in \mathcal{D} since we have used a reduction to some $G \in \mathcal{U} \Rightarrow G \in \mathcal{D}$.

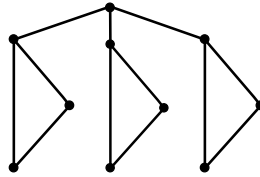
c) We don't know whether this statement is true or not, if we assume $P = NP$ statement is true, otherwise $P \neq NP$ statement is false.

Problem 3 (*Trees*)

5+10=15p

A tree is a connected undirected acyclic graph.

- (a) How many spanning trees has the following graph? Shortly explain your results.



- (b) Recap Kruskal's algorithm.

```
KRUSKAL( $G = (V, E), w: E \rightarrow \mathbb{R}$ ) //  $m = |E|$ 
1: sort edges such that  $w(e_1) \leq w(e_2) \dots \leq w(e_m)$ 
2:  $F = \emptyset, T = (V, F)$ 
3: for  $i = 1, \dots, m$  do
4:   if  $(V, E \cup \{e_i\})$  is acyclic then
5:      $T = (V, E \cup \{e_i\})$ 
6: return  $T$ 
```

Explain, in words, how to use Kruskal's algorithm to compute the number of connected components in an undirected graph.

a) each Δ has 3 spanning Trees.

each spanning tree must contain all edges in

& we are free for choosing sp.T. in the 3 Δ

$\Rightarrow 3 \cdot 3 \cdot 3 = 27$ sp.T.



b) in general: Kruskal returns an acyclic graph

this graph is connected if
input graph and else not
 \Rightarrow in all cases we get

forest

\Rightarrow count nr m^* of added edges

& assume we have $C_1 \dots C_k$ conn. comp. (k unknown)

for each C_i we have $|V_i| - 1$ edges

$$\Rightarrow m^* = \sum_{i=1}^k |V_i| - 1 = \sum_{i=1}^k |V_i| - k = |V| - k$$

$$\Rightarrow k = |V| - m^*$$

Problem 4 (Shortest Paths)

10p

Recap the following algorithm to compute the length of a shortest paths between a source vertex s and the other vertices.

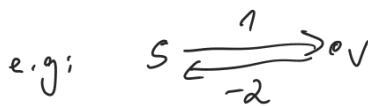
BELLMANN-FORD(digraph G , weight function w , source s)

- 1: **for** each vertex v of G **do**
- 2: $v.d = \infty$
- 3: $s.d = 0$
- 4: **for** $i = 1, \dots, |V| - 1$ **do**
- 5: **for** every edge $(u, v) \in E$ **do**
- 6: **if** $v.d > u.d + w(u, v)$ **then**
- 7: $v.d = u.d + w(uv)$

Explain shortly for which type of weighting functions BELLMANN-FORD computes a correct solution and provide a *minimal* example (a digraph without loops and with the fewest number of vertices) that shows that the algorithm BELLMANN-FORD may not work if we have as input a directed graph with arbitrary weighting function. To this end, apply BELLMANN-FORD on your example, provide the results for all intermediate steps and show where the algorithm fails.

Bellmann needs conserv. weight. functions, i.e.,
the input graph as no negative cycles,

=> min. example must contain negative cycles



i.u.t.
 $v.d = \infty$
 $s.d = 0$

$i=1$: (sv) $v.d > s.d + w(sv)$
 $\infty > 0 + 1$
 $\rightarrow v.d = 1$

(vs) $s.d = 0 > 1 + w(vs)$
 $\rightarrow s.d = -2$

$i=2$: (sv) $v.d = 1 > s.d + w(sv)$
 $\Rightarrow v.d = -1$

(vs) $s.d = -2 > v.d + w(vs)$
 $s.d = -3$

=> although $d(s,s) = "-\infty"$
or "only" opt $d(s,s) = -2$

Problem 5 (Greedy and Matroids)

5+7=12p

Recall that in the Knapsack problem, we are given two things. First a list of n items i_1, \dots, i_n , where item i_j has positive value v_j and positive weight w_j . Second a positive capacity C which is the maximum weight that can be carried in the knapsack. The items are indivisible so that we must either place the entire item in the knapsack or not place it in the knapsack. Our goal is to maximize the sum of the values of all the items that are placed in the knapsack.

- (a) Define the independence system (E, \mathbb{F}) that describes this problem and also prove that (E, \mathbb{F}) is an independence system.
- (b) Prove that a greedy algorithm will in general not optimally solve this problem by showing that (E, \mathbb{F}) is not a matroid.

$$E = \{i_1, \dots, i_n\}$$

$$\mathbb{F} = \{F \subseteq E : w(F) := \sum_{i \in F} w_i \leq C\}$$

$$1) \emptyset \in \mathbb{F} \text{ since } w(\emptyset) := 0 \leq C$$

$$2) \begin{array}{l} E \in \mathbb{F} \\ E' \subseteq E \end{array} \Rightarrow w(E') \leq w(E) \leq C \\ \Rightarrow E' \in \mathbb{F}$$

b) $C=10$

	i_1	i_2	i_3
w	6	5	5

 \rightarrow first $\mathbb{F} = \{\{i_1\}, \{i_2\}, \{i_3\}, \{i_2, i_3\}\}$

but for $\{i_1\}$ & $\{i_2, i_3\}$
exchange axiom not satisfied,

Problem 6 (Dynamic Programming)

5+10=15p

Given an ordered list of $n \geq 1$ strictly positive integers k_1, \dots, k_n the product-sum is the *largest* sum that can be formed by multiplying adjacent elements in the list. Each element can be multiplied with at most one of its neighbors. For example, given the list 1, 2, 3, 1 the product sum is $8 = 1 + (2 \cdot 3) + 1$, and given the list 2, 2, 1, 3, 2, 1, 2, 2, 1, 2 the product sum is $19 = (2 \cdot 2) + 1 + (3 \cdot 2) + 1 + (2 \cdot 2) + 1 + 2$.

- (a) Compute the product-sum of 1, 4, 3, 2, 3, 4, 2 (4)
(b) Give a recursive formula for computing the product-sum of the first j elements.

c) $1 + (4 \cdot 3) + 2 + (3 \cdot 4) + 2$

b) $OPT(j)$ \rightarrow two cases: either k_j is just deleted
as multiplied with k_{j-1}
element.
 $j = \# \text{ of pos. int.}$
 $k_1 \dots k_j$

$$\Rightarrow OPT(j) = \max \left\{ \begin{array}{l} OPT(j-1) + k_j \\ OPT(j-2) + k_{j-1} \cdot k_j \end{array} \right\}$$

if $j \geq 2$


$$OPT(1) = k_1$$

$$OPT(0) = 0$$

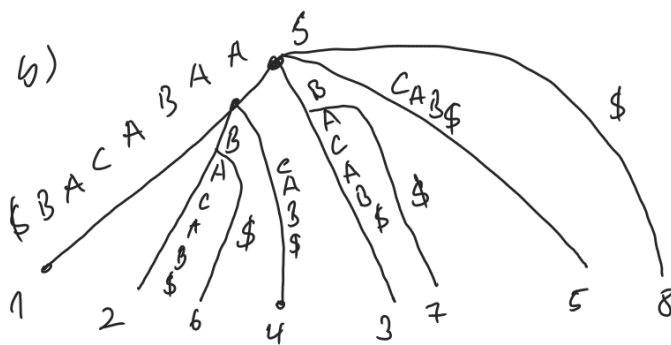
Problem 7 (Suffix Trees)

5+5=10p

- (a) Argue why for a given string $S = s_1 \dots s_n$, where $s_n = \$ \neq s_i$ for all $i \in \{1, \dots, n-1\}$ the root of a suffix tree has at least two children.
- (b) Build the suffix tree for the string $S = AABACAB\$$.



 upto $s_1 \dots s_{n-1}$ \rightarrow no add $s_n = \$$ since it does not occur
 k no edge of tree so far starts with $\$$
 \Rightarrow $\$$ must added as extra edge.



Problem 8 (*Balanced binary search trees*)

6p

What is the best possible asymptotic bound for searching an element in a balanced binary search tree with $n \cdot 2^n$ vertices in a worst case: A1, A2, A3 or A4? Prove your answer.

(A1) $O(n \cdot \log_2(n))$

(A2) $O(n \cdot 2^n)$

(A3) $O(n)$

(A4) $O(\log_2(n))$

search in balanced tree works in $O(\log_2(|V|))$ time

$$\& \quad |V| = n \cdot 2^n$$

$$\Rightarrow \log_2(n \cdot 2^n) = \log_2(n) + \log_2(2^n) = \log_2(n) + n$$

$$\text{since } \log_2(n) \leq n \Rightarrow O(n).$$

$$\text{in 1 a): } 2^n \geq n \quad \forall n \geq 4$$

$$\Rightarrow \log_2(2^n) \geq \log_2(n) \quad \forall n \geq 4$$

"n"