Problem 1. **/ 15 P**

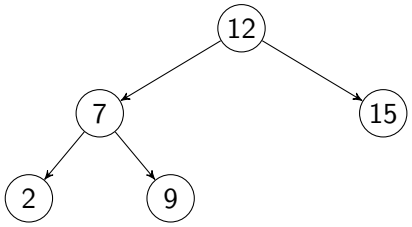
Please note:

1) In this problem, you have to provide **solutions only**. You can write them right on this sheet.
2) If you use algorithms and notation other than that of the lecture, you need to **briefly** explain them in such a way that the results can be understood and checked.
3) We assume letters to be ordered alphabetically and numbers to be ordered ascendingly, according to their values.

/ 1 P a) Insert the keys 27, 14, and 16 in this order into the hash table below. Use double hashing with the hash function $h(k) = k \bmod 11$. Resolve collisions using $h'(k) = 1 + (k \mod 9)$ for probing (to the *left*, i.e. substract $h'(k)$).

			3		5			19		
0	1	2	3	4	5	6	7	8	9	10

/ 1 P b) You want to insert *a single* integer key that results in a double rotation into the following AVL tree. All keys stored in the AVL tree should be pairwise disjoint. Indicate *all possible* integer candidate keys.

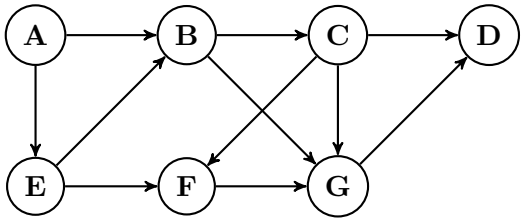


Candidates: _____.

/ 1 P c) Given the following array that results from a quicksort partition phase. Which key has been used as pivot element? Mark *all possible* candidates.

2	1	3	5	4	6	9	8	7
1	2	3	4	5	6	7	8	9

/ 1 P d) Provide a topological ordering for the graph below.



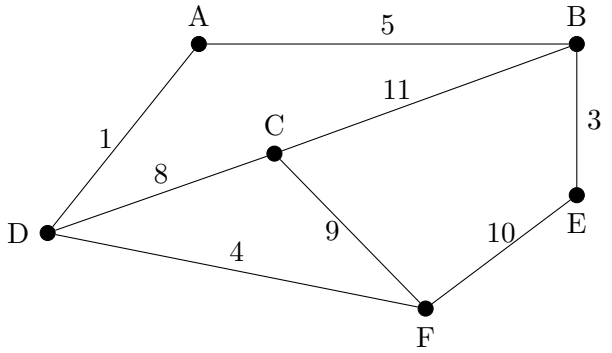
Topological ordering:

——, ——, ——, ——, ——, ——, ——.

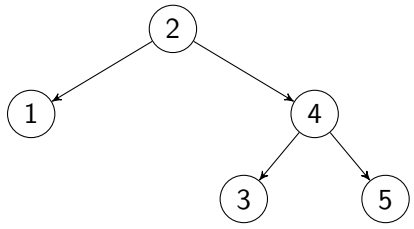
/ 1 P e) Draw all $2,3$-trees (B-trees with up to 3 children for each node) for the key set $\{1, 2, 3, 4, 5\}$.

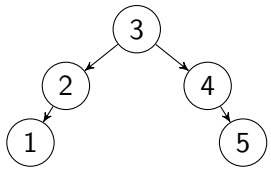
/ 1 P f) Prove or disprove: Let $G = (V, E)$ be an undirected graph, $s, t \in V$ two vertices and $T = (V, E')$ with $E' \subseteq E$ a minimum spanning tree of G. The path connecting s and t in T is a shortest path in G.

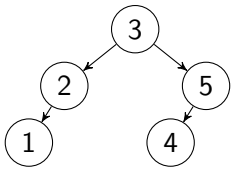
/ 1 P g) Mark the first *three* edges in the following graph that are selected for the minimum spanning tree by the algorithm of Jarník, Prim und Dijkstra *starting from vertex A*.

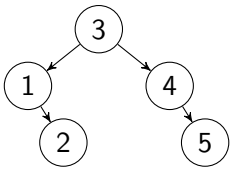


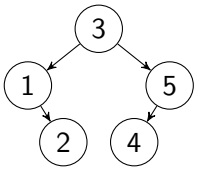
/ 1 P h) The keys 1 and 3 are accessed in this order in the following splay tree. Mark the resulting splay tree below.

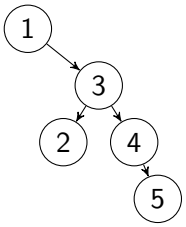


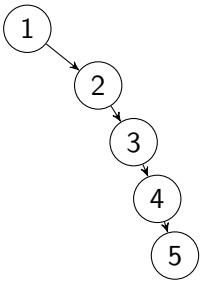
☐ 

☐ 

☐ 

☐ 

☐ 

☐ 

/ 1 P i) Given an array $A[1..n]$ and the following Java implementation of the sorting algorithm *insertion sort*. The function `swap(A, i, j)` exchanges (swaps) the elements $A[i]$ and $A[j]$. The algorithm is called with the parameters $l = 1$ and $r = n$ to sort the array $A[1..n]$ in ascending order. Complete the implementation (lines 2, 3, and 4).

```
1 public void insertionSort(int[] A, int l, int r) {
2     for (int i=_______; i<=_______; i++)
3         for (int j=i-1; j>=l && A[j]>A[_______]; j--)
4             swap(A, _______, _______);
5 }
```

/ 3 P

j) Consider the following recursive formula:

$$T(n) := \begin{cases} 3 \cdot T(n/7) + 4 & n > 1 \\ 3 & n = 1 \end{cases}$$

Specify a closed (non-recursive) form for $T(n)$ that is *as simple as possible*, and prove its correctness using mathematical induction. You may assume that n is a power of 7, i.e. use $n = 7^k$ or $k = \log_7(n)$.

Hint: For $q \neq 1$, we have $\sum_{i=0}^{k} q^i = \frac{q^{k+1}-1}{q-1}$.

Derivation (if required):

Closed and simplified form:

$T(n) = T(7^k) =$

Proof by induction:

/ 1 P

k) Specify (as concisely as possible) the asymptotic running time of the following code fragment in Θ notation depending on $n \in \mathbb{N}$. You do not need to justify your answer.

```
1 for ( int i = n; i >= 1; i = i / 2 ) {
2         int j = n;
3         while ( j >= 1 ) {
4                 j = j − 20;
5         }
6 }
```

Running time as concisely as possible in Θ notation:

/ 1 P

l) Specify (as concisely as possible) the asymptotic running time of the following code fragment in Θ notation depending on $n \in \mathbb{N}$. You do not need to justify your answer.

```
1 int f( int n ) {
2         if ( n <= 1 ) return 1;
3         for ( int i = 1; i · i <= n; i = i + 2 ) {
4                 ;
5         }
6         return f( n/4 ) + 1;
7 }
```

Running time as concisely as possible in Θ notation:

/ 1 P

m) Specify an **order** for the functions below such that the following holds: If function f is left of function g, then $f \in \mathcal{O}(g)$.

Example: The three functions n^3, n^7, n^9 are already in a correct order, since $n^3 \in \mathcal{O}(n^7)$ and $n^7 \in \mathcal{O}(n^9)$.

$$4^{n/2},\ \sqrt{n},\ \binom{n}{4},\ 10^{64},\ \log(n!),\ n^n,\ \log(n^{16})$$

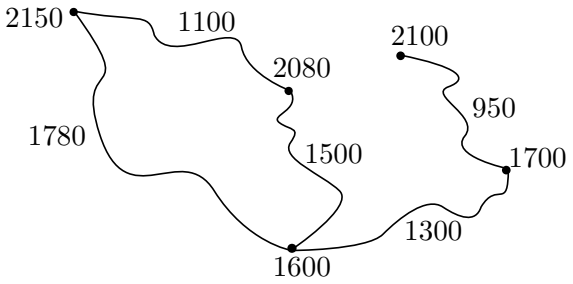
Solution: _____, _____, _____, _____, _____, _____, _____.

Problem 2. **/ 12 P**

In a ski region there are n stations that are connected by m ski slopes. The stations $i \in \{1, \ldots, n\}$ are sorted in descending order by their height above sea level $h_i \in \mathbb{N}$, i.e. from $i < j$ follows $h_i \geq h_j$. The slopes are given as pairs (i, j) for stations $i, j \in \{1, \ldots, n\}$ with $i \neq j$, where $h_i > h_j$ holds for every slope. Moreover, the length of all slopes $l_1, l_2, \ldots, l_m$ with $l_i \in \mathbb{N}$ for all $i \in \{1, \ldots, m\}$, the maximum height of a station $H = \max_{i \in \{1,\ldots,n\}} h_i$, and the total length of all slopes in the ski region $L = \sum_{i=1}^{m} l_i$ are given.

A ski-run is a sequence of slopes that can be used continuously, i.e. for all but the last slope of the ski-run it holds that the terminal station of one slope is the starting station of the following slope. The length of a ski-run is the sum of the lengths of the used slopes. Compute a longest ski-run in the ski region.

Example: There are $n = 5$ stations (dots) and $m = 5$ slopes (wavy lines). The heights of the stations and the lengths of the slopes are given in the figure. The total length of all slopes is $L = 6630$ meters and the maximal height of a station is $H = 2150$ meters. The longest ski-run has length 2600 meters.



/ 7 P a) Provide a dynamic programming algorithm that as efficiently as possible computes the length of a longest ski-run. Define the size of the DP table and the meaning of a table entry and address the following aspects in your solution.

1) What is the meaning of a table entry, and what size does the DP table have?
2) How can an entry be computed from the values of previously computed entries?
3) In which order can the entries be computed?
4) How can the length of a longest ski-run be obtained from the DP table?

First choose the dimension of the DP table:

☐ **(I) one-dimensional** → Use the scheme on page 8.

☐ **(II) two-dimensional** → Use the scheme on page 9.

☐ **(III) other number of dimensions** → Describe your dynamic program on page 10.

Answer the subtasks b) and c) on page 11.

/ 2 P b) Describe in detail how you can recognize from the DP table which slopes are selected for a ski-run of maximal length.

/ 3 P c) Provide the running time of the algorithm developed in a) and in b), and justify your answer. Is the running time polynomial? Justify your answer.

(I) For algorithms using a one-dimensional DP table:

Size of the DP table / number of entries:

☐ n ☐ 2^n ☐ $L+1$ ☐ other:

☐ m ☐ 2^m ☐ $H+1$ _____

Meaning of a table entry:

Meaning of the entry at position i:

$DP[i]$: _____

_____

Initialization:

Computation of an entry:

$DP[i] =$

Order of computation:

How can the length of a longest ski-run be obtained from the DP table?

(II) For algorithms using a two-dimensional DP table:

Size of the DP table: *Number of rows*

☐ n ☐ m ☐ other: _____

Size of the DP table: *Number of columns*

☐ n ☐ $L+1$ ☐ other: _____

☐ m ☐ $H+1$

Meaning of a table entry:

Meaning of the entry in row i and column j:

$DP[i,j]$: _____

_____

Initialization:

Computation of an entry:

$DP[i,j] =$

Order of computation:

How can the length of a longest ski-run be obtained from the DP table?

(III) For algorithms using a DP table with 3 or more dimensions:

Subtasks b) and c)

b) Describe in detail how you can recognize from the DP table which slopes are selected for a ski-run of maximal length.

c) Provide the running time of the algorithm developed in a) and in b), and justify your answer. Is the running time polynomial? Justify your answer.

/ 7 P

Problem 3.

In a hospital the blood preservations have to be allocated to the patients. The blood group compatibilities have to be respected. The table on the right summarizes the blood group compatibilities. For example, a patient with blood group A can get a blood preservation of blood group 0 or A, while a patient with blood group 0 can only get a blood preservation of blood group 0.

	blood preservation			
patient	**0**	**A**	**B**	**AB**
AB	✓	✓	✓	✓
A	✓	✓		
B	✓		✓	
0	✓			

Given the number of available blood preservations k_a, k_b, k_{ab}, and k_0 of blood group A, B, AB, and 0, and the number of patients b_a, b_b, b_{ab}, and b_0 with blood group A,B, AB, and 0 in need of one blood preservation each, develop an algorithm that computes if an allocation exists such that every patient gets a blood preservation of a compatible blood group.

Example: The number of available blood preservations and patients are given in the table on the right. The following allocation respects the blood group compatibilities: The blood preservations of blood group 0 are given to patients with blood group 0 (2 blood preservations) and with blood group B (1 blood preservation), blood preservations of blood group A are given to patients with blood group A (3 blood preservations) and blood group AB (1 blood preservation). Finally, blood preservations of every other blood groups are given to the patients with the same blood group.

Blood group	**0**	**A**	**B**	**AB**
Demand	2	3	2	4
Stock	3	4	1	3

/ 4 P a) Model the above problem as a flow problem. For this purpose, describe the construction of an appropriate network $N = (V, E, c)$ with the vertex set V and edge set E, and describe which capacities c the edges should have. How can you deduce from the value of a maximum flow if an allocation exists such that every patient gets a blood preservation of a compatible blood group?

/ 3 P b) Suppose that we already solved the flow problem described in a), i.e., we know for a maximum flow φ the value of the flow φ_e for each edge e, and suppose that an allocation exists that gives each patient a compatible blood preservation. Describe in detail an algorithm that uses the φ_e values and that computes such an allocation. What is the running time of your algorithm if every φ_e can be accessed in constant time?

Subtask a)

Definition of the network N (if possible in words and not formally):

Vertex set V: _____

_____

Edge set E: _____

_____

Capacities c: _____

_____

Schematic representation / drawing of the network:

An allocation exists if: _____

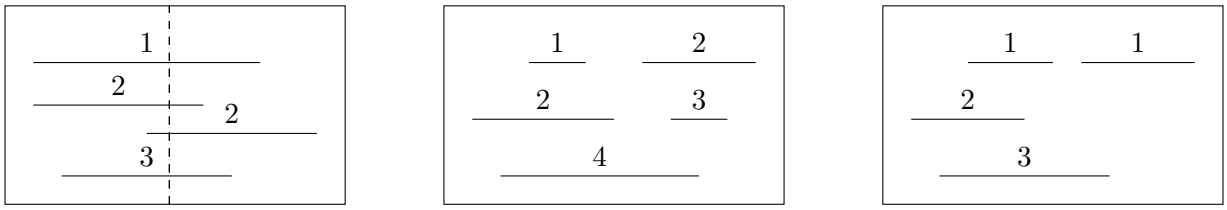
_____

Subtask b)

b) Suppose that we already solved the flow problem described in a), i.e., we know for a maximum flow φ the value of the flow φ_e for each edge e, and suppose that an allocation exists that gives each patient a compatible blood preservation. Describe in detail an algorithm that uses the φ_e values and that computes such an allocation. What is the running time of your algorithm if every φ_e can be accessed in constant time?

Problem 4. **/ 10 P**

Given a set of n horizontal, non-intersecting line segments in the plane. Each line segment is given by the coordinates (x_l, y) and (x_r, y) of both endpoints. A numbering of the line segments with numbers from $\{1, \ldots, m\}$ is *feasible* if for each vertical line it holds that the line segments crossed by this line are numbered from top to bottom in increasing order. Compute the smallest number $m \leq n$ such that there exists a feasible numbering of the line segments with numbers from $\{1, \ldots, m\}$.



Examples: Left: The numbering of the line segments is not feasible, as the indicated vertical line (dashed line) crosses line segments that are not numbered in increasing order from top to bottom. *Center:* The numbering is feasible, but there exists a feasible numbering with $m = 3$. *Right:* The numbering is feasible and there exists no feasible numbering with $m < 3$.

/ 8 P a) Design an efficient sweepline algorithm for the above problem. Address the following aspects in your solution.

1) In which direction is the sweepline moving, and what are the stopping points?
2) Which objects have to be stored in the sweepline data structure, and what is an appropriate choice for this data structure?
3) What happens if the sweepline encounters a new stopping point?
4) How can the smallest number m such that there exists a feasible numbering with numbers from $\{1, \ldots, m\}$ be computed?
5) Provide the running time of your algorithm depending on n, and justify your answer.

/ 2 P b) Describe how your algorithm can be modified such that it additionally computes a feasible numbering of the line segments with numbers from $\{1, \ldots, m\}$.

Movement of the sweepline (choose exactly one option):

- The sweepline is a *vertical line*; it moves
  ☐ from left to right / ☐ from right to left.
- The sweepline is a *horizontal line*; it moves
  ☐ from top to bottom / ☐ from bottom to top.
- The sweepline is a *half-line*; it rotates
  ☐ in clockwise direction / ☐ in anticlockwise direction around point _____.

☐ other movement: _____

_____.

Stopping points:

set of stopping points: _____

_____

ordered by: _____

_____

Sweepline data structure:

Operations when encountering a new stopping point:

Computation of the smallest number m:

Running time of the algorithm:

Subtask b)

b) Describe how your algorithm can be modified such that it additionally computes a feasible numbering of the line segments with numbers from $\{1, \ldots, m\}$.