/ 20 P

Programming Task P1.

Enrollment Key: `AlgoDataExam2018`

Submission: see Section 3 of the Technical Guide

Structure

In this exercise, we implement a max-heap data structure. Your task is to complete the implementation of the following methods:

- **`buildHeap()`**: builds a heap from a given array of keys in arbitrary order.
- **`insert(`x`)`**: adds a new key x to the heap.
- **`deleteMax()`**: deletes the maximum key from the heap.

Most of the implementation of the max-heap data structure is already provided by the template (including the code to read the input, allocate space for the heap structure, write the state of the heap, etc).

The heap holds N keys which are integer values in the range $[-2^{31}, 2^{31} - 1]$. For simplicity, we assume that the heap will not exceed more than 100'000 keys, i.e, $N \leq 100000$.

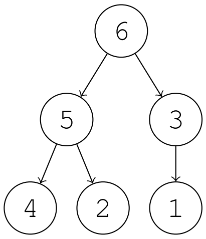
To validate the correctness of your implementation of these methods, the state of the heap is written to the output. Assuming a heap of N keys, the state of the heap is described by N partially-ordered integers that satisfy the heap property. For simplicity and convenience, the template already includes routines to output the state of the heap.

Example

- The **`buildHeap()`** method builds a heap structure by restoring the heap condition for a given array of keys. For example, given an array of $N = 5$ keys:

1	2	3	4	5	6

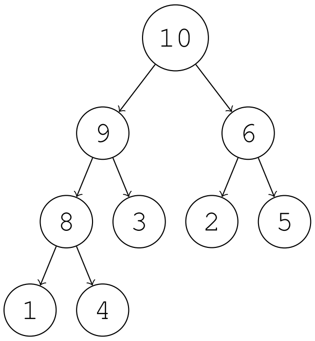
  the state of the heap, as well as the partial order on the numbers is given below:



6	5	3	4	2	1

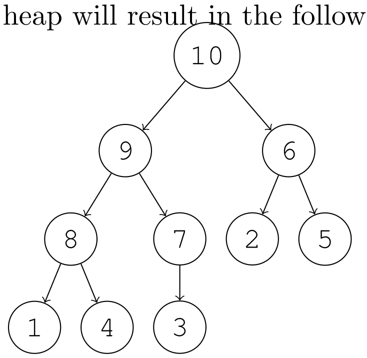
  Once the heap is built, the state of the heap is written to the output.

- For the **`insert(`*x*`)`** method, assume you are given the following heap of $N = 9$ keys:



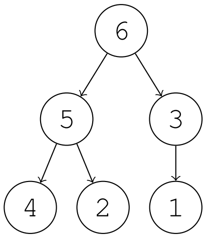
10	9	6	8	3	2	5	1	4

Inserting the key $x = 7$ into the heap will result in the following heap of $N = 10$ keys:



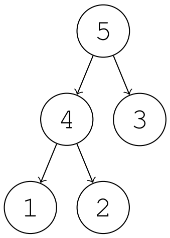
10	9	6	8	7	2	5	1	4	3

- For the **`deleteMax()`** method, assume you are given the following heap of $N = 6$ keys:



6	5	3	4	2	1

Once the maximum key is removed, the resulting heap of $N = 5$ keys looks as follows:



5	4	3	1	2

Grading

Overall, you can obtain a maximum of 20 judge points for this programming task. To get full points your program should require $O(n)$ time for the **buildHeap()** method, and $O(\log(n))$ for both **insert(**x**)** and **deleteMax()** methods (with reasonable hidden constants). Incomplete solutions can obtain partial points, namely:

- You can obtain up to 8 points for a correct implementation of **buildHeap()**.
- You can obtain up to 8 points for a correct implementation of **insert(x)**.
- You can obtain up to 4 points for a correct implementation of **deleteMax()**.

Instructions

For this exercise, we provide a program template as an Eclipse project in your workspace that helps you reading the input and writing the output. Importing any additional Java class is **not allowed** (with the exception of the already imported ones `java.io.{InputStream, OutputStream}` and `java.util.Scanner` class).

The project also contains data for your local testing and a `JUnit` program that runs your `Main.java` on all the local tests – just open and run `StructureTest.launch` in the project. The local test data are different and generally smaller than the data that are used in the online judge.

Submit only your `Main.java`.

The input and output are handled by the template – you should not need the rest of this text.

Input The input of this problem consists of a number of test-cases. The first line contains T, the number of test-cases. Each of the T cases is independent of the others, contains one line, and starts with a command number that can be either 1, 2 or 3.

1. If the command number is set to 1, then we read an array from the input, build a heap, and print the state of the heap on the output.

   The command number is followed by a number N, that describes the number of keys in the array, followed by N integer values in the range $[-2^{31}, 2^{31}-1]$. All numbers are separated by a blank space.

2. If the command number is set to 2, we read in the heap, insert elements into the heap, and output the state of the heap.

   The command number is followed by a number N that describes the number of keys in the heap, followed by N integer values in the range $[-2^{31}, 2^{31}-1]$, that satisfy the heap property. The N values are then followed by a number M that describes the number of keys that must be inserted in the heap, followed by M integers in the range $[-2^{31}, 2^{31}-1]$. All numbers are separated by a blank space.

3. If the command number is set to 3, then we read in the heap, delete the maximum element from the heap once or several times, and output the state of the heap.

The command number is followed by a number N that describes the number of keys in the heap, followed by N integer values in the range $[-2^{31}, 2^{31}-1]$, that satisfy the heap property. The N values are then followed by a number M that describes the number of times we need to remove the maximum key from the heap. All numbers are separated by a blank space.

Output For every case, the output is the state of the heap.

The output contains one line for each test-case. More precisely, the i-th line of the output contains an array of integer values that correspond to the heap state, separated by a blank space. The output is terminated with an end-line character.

Example input:

```
3
1 6 1 2 3 4 5 6
2 9 10 9 6 8 3 2 5 1 4 1 7
3 6 6 5 3 4 2 1 1
```

Example output:

```
6 5 3 4 2 1
10 9 6 8 7 2 5 1 4 3
5 4 3 1 2
```

Space for your notes. These will not be graded. Only what was submitted to the judge counts for this exercise.

Space for your notes. These will not be graded. Only what was submitted to the judge counts for this exercise.

/ 20 P

Programming Task P2.

Enrollment Key: `AlgoDataExam2018`

Submission: see Section 3 of the Technical Guide

Square

You are given a 2-dimensional binary matrix B having M rows and N columns ($0 \leq M, N \leq 1024$) filled with only 0's and 1's. Your task is now to find the area of a largest square submatrix that contains only 1's.

Example

$$\begin{array}{ccccc} 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{array}$$

$$\downarrow$$

$$\begin{array}{cc|cc|c} 1 & 0 & \multicolumn{1}{c}{1} & \multicolumn{1}{c}{0} & 0 \\ \cline{3-4} 1 & 0 & \mathbf{1} & \mathbf{1} & 1 \\ 1 & 1 & \mathbf{1} & \mathbf{1} & 1 \\ \cline{3-4} 1 & 0 & \multicolumn{1}{c}{0} & \multicolumn{1}{c}{1} & 0 \end{array}$$

In the $(M = 4) \times (N = 5)$ binary matrix illustrated above, a largest square submatrix containing only 1's has an area of 4.

Grading

Overall, you can obtain a maximum of 20 judge points for this programming task. To get full points your program should run in time $O(N \cdot M)$, with reasonable hidden constants. Slower solutions might get partial points:

- You can obtain up to 10 points for an $O(N^2 \cdot M^2)$-time solution.
- You can obtain up to 5 additional points (i.e., 15 points in total) for an $O(N \cdot M \cdot \min(M, N))$-time solution.
- You can obtain up to 5 additional points (i.e., 20 points in total) for an $O(N \cdot M)$-time solution.

Instructions

For this exercise, we provide a program template as an Eclipse project in your workspace that helps you reading the input and writing the output. Importing any additional Java class is **not allowed** (with the exception of the already imported ones `java.io.{InputStream, OutputStream}` and `java.util.Scanner` class).

The project also contains data for your local testing and a `JUnit` program that runs your `Main.java` on all the local tests – just open and run `SquareTest.launch` in the project. The local test data are different and generally smaller than the data that are used in the online judge.

Submit only your `Main.java`.

The input and output are handled by the template – you should not need the rest of this text.

Input The input of this problem consists of a number of test-cases. The first line of the input contains the number T that describes the number of test cases.

Each case is independent of the others and consists of seveal lines. The first line of each of the T test cases, contains the integers M and N separated by a space. The next M lines contain N integers either 1 or 0, separated by spaces.

Output For every case, the output should contain one integer number on a separate line – the area of a largest square submatrix that contains only 1's.

More precisely, the i-th line of the output contains a single integer corresponding to the area of the largest square found in the binary matrix of the i-th test-case.

Example input:

```
1
4 5
1 0 1 0 0
1 0 1 1 1
1 1 1 1 1
1 0 0 1 0
```

Example output:

```
4
```

Space for your notes. These will not be graded. Only what was submitted to the judge counts for this exercise.

Space for your notes. These will not be graded. Only what was submitted to the judge counts for this exercise.

/ 14 P

Theory Task T1.

Notes:

1) In this problem, you have to provide **solutions only**. You should write them directly on this sheet.
2) We assume letters to be ordered alphabetically and numbers to be ordered ascendingly, according to their values.

/ 2 P

a) Below you see four sequences of snapshots, each obtained during the execution of one of the following algorithms: `InsertionSort`, `SelectionSort`, `QuickSort`, `MergeSort`, and `BubbleSort`. For each sequence, write down the corresponding algorithm.

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

______________`Sort`

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

______________`Sort`

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

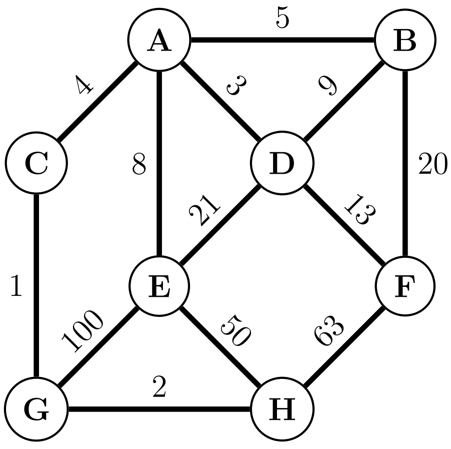
______________`Sort`

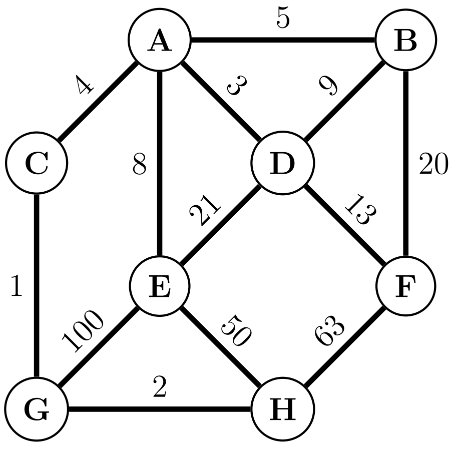
3	8	5	4	1	2	7	6
3	6	5	4	1	2	7	8
3	2	5	4	1	6	7	8

______________`Sort`

/ 2 P

b) Below you find two copies of the same graph. In the left graph, highlight the first 5 edges that Prim's algorithm chooses, when starting in vertex *D*. In the right graph, highlight the first 5 edges that Kruskal's algorithm discards.



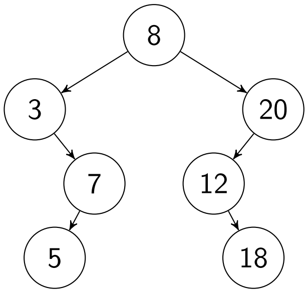


/ 1 P

c) *Binary search trees*: Draw the binary search tree that is obtained when inserting in an empty tree the keys 3, 8, 12, 5, 20, 7, 18 in this order.

/ 1 P

d) *Binary search trees*: Draw the resulting binary search tree obtained by deleting the key 8 from the following binary search tree.

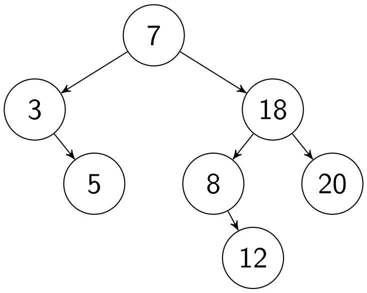


/ 1 P

e) *AVL-trees*: Draw the AVL-tree that is obtained when inserting in an empty tree the keys 3, 8, 12, 5, 20, 7, 18 in this order.

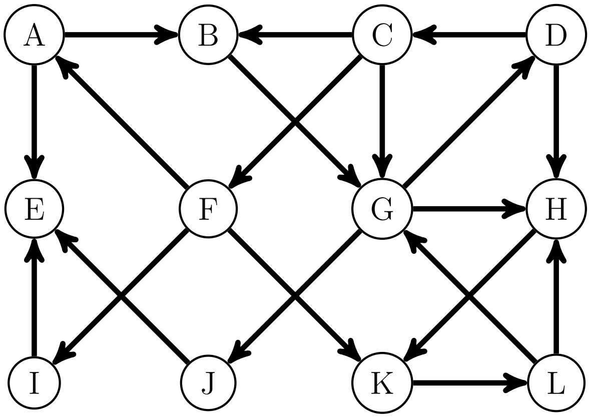
/ 1 P

f) *AVL-trees*: Draw the resulting AVL-tree obtained by deleting the key 3 from the following AVL-tree.



/ 2 P

g) In the following graph, execute a depth first search and a breadth first search. Start in vertex G and assume that each traversal visits the neighbouring vertices in alphabetical order. Write down the orders (DFS-order and BFS-order) in which the vertices are visited.



DFS-order:

_____, _____, _____, _____, _____, _____, _____, _____, _____, _____, _____, _____.

BFS-order:

_____, _____, _____, _____, _____, _____, _____, _____, _____, _____, _____, _____.

/ 2 P

h) For each of the following claims, state whether it is true or false. You get 0.5P for a correct answer, -0.5P for a wrong answer, 0P for a missing answer. You get at least 0 points in total.

claim	**true**	**false**
$\frac{n}{\log n} \leq O(\sqrt{n})$	□	□
$\log(n!) \geq \Omega(n^2)$	□	□
$n^k \geq \Omega(k^n)$, if $1 < k \leq O(1)$	□	□
$\log_3 n^4 = \Theta(\log_7 n^8)$	□	□

/ 2 P

i) Given the following recursion:

$$T(n) := \begin{cases} 9 \cdot T(\frac{n}{3}) + 32n - 16 & n > 1 \text{ and } n \text{ a power of 3} \\ 6 & n = 1 \end{cases}$$

You can assume that n is a power of 3. Prove by mathematical induction that

$$T(n) = 20n^2 - 16n + 2$$

is the corresponding closed formula.

Proof by Induction:

Extra space. Please indicate clearly to which task your notes belong. Please cross out all notes that you do not want to be graded.

Extra space. Please indicate clearly to which task your notes belong. Please cross out all notes that you do not want to be graded.

Theory Task T2.

/ 14 P

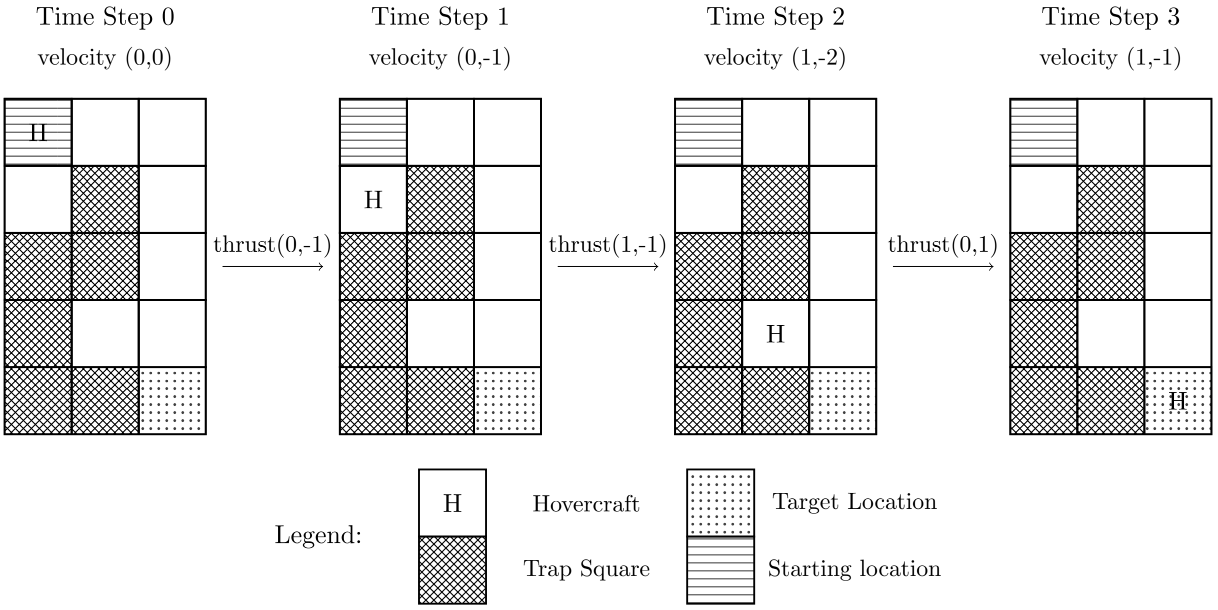
You are driving a remote-controlled hovercraft on a field. The field is a grid of $m \times n$ squares, and you want to reach a *target* location from a *starting* location. Some squares are traps and have spikes on them that can emerge from the ground and wreck your hovercraft. The locations of the trap squares are known. Also, if the hovercraft leaves the $m \times n$ field, it is destroyed. There is very little friction between your hovercraft and the ground so it is difficult to change its velocity. In order to change the hovercraft's velocity, you can use the hovercraft's thrusters.

The hovercraft has a position (x, y), $x \in \{1, \ldots, m\}$ and $y \in \{1, \ldots, n\}$, and a velocity (v_x, v_y), $v_x, v_y \in \{-2, -1, 0, 1, 2\}$. The movement of the hovercraft proceeds according to time steps. At each time step, the following actions occur (in this order):

1. The hovercraft may fire a thruster along the x axis. Then $v_x := v_x + a_x$ where $a_x \in \{-1, 1\}$. (This is only permissable if the new velocity is still valid.)
2. The hovercraft may fire a thruster along the y axis. Then $v_y := v_y + a_y$ where $a_y \in \{-1, 1\}$ (This is only permissable if the new velocity is still valid.)
3. The hovercraft's position is updated: $x := x + v_x$ and $y := y + v_y$
4. If the hovercraft is located on a trap square, the spikes emerge and the hovercraft is destroyed. The spikes then retract.

At the beginning (starting location), the hovercraft is stationary, and it can have any velocity when it reaches its destination (target location). Moreover, you can assume that it is always possible to reach the target location.

Example



/ 6 P

a) Your goal is to minimize the number of time steps it takes to get from the starting location to the target location. Model the state space as a graph $G = (V, E)$. Describe what the vertices and edges represent. For each vertex $v \in V$, precisely state to which other vertices there is an edge. Name an as efficient as possible algorithm for solving this problem, and state the running time of the algorithm in terms of m and n.

Definition of the graph (if possible, in words and not formal):

Number of vertices and edges (as concisely as possible in Θ notation in terms of n and m):

Algorithm, as efficient as possible:

Running time (as concisely as possible in Θ notation in terms of n and m). Justify your answer:

/ 4 P

b) We now modify the problem such that the hovercraft's thrusters are powered by a cartrige of compressed carbon dioxide. The hovercraft's thrusters can only be used T times, where T is a constant. Your goal is still to minimize the number of time steps it takes to get from the starting location to the target location. For this modified problem, model the state space as a graph $G = (V, E)$, describe what the vertices and edges represent, and for each vertex $v \in V$, precisely state to which other vertices there is an edge. (Thrusting in both the x and y directions costs the same as a single thrust.) Then, name an as efficient as possible algorithm for solving this problem, and state the running time of the algorithm in terms of m and n.

Definition of the graph (if possible, in words and not formal):

Number of vertices and edges (as concisely as possible in Θ notation in terms of n and m):

Algorithm, as efficient as possible:

Running time (as concisely as possible in Θ notation in terms of n and m). Justify your answer:

/ 4 P

c) We modify the original problem again. Instead of minimizing the number of time steps to reach the destination, some squares now give points when the hovercraft ends a time step on them, but most squares take points away. The goal is to score the maximum number of points, while travelling from the starting location to the end location.

Assume the following:

- You are only allowed to stop on the starting location and the target location (i.e. have velocity $(0, 0)$).
- The starting location gives a negative amount of points.
- An optimum solution exists and has a finite length.

For this modified problem, model the state space as a graph $G = (V, E)$, describe what the vertices and edges represent, and for each vertex $v \in V$, precisely state to which other vertices there is an edge. Name an efficient algorithm for solving this modified problem, and state the running time of the algorithm in terms of m and n.

Definition of the graph (if possible, in words and not formal):

Number of vertices and edges (as concisely as possible in Θ notation in terms of n and m):

Scheme continues on the next page.

Algorithm, as efficient as possible:

Running time (as concisely as possible in Θ notation in terms of n and m). Justify your answer:

Extra space. Please indicate clearly to which task your notes belong. Please cross out all notes that you do not want to be graded.

Theory Task T3.

/ 12 P

The university curriculum consists of n courses. For simplicity assume that courses are represented by their numbers from 1 to n. Each course C has a list of prerequisite courses (this list may be empty). Before attending the course C, students have to pass at least one of its prerequisite courses. Notice that being a prerequisite is not transitive: if A is a prerequisite for B and B a prerequisite for C, then this does not imply that A is a prerequisite for C. Furthermore, you can assume that the curriculum has no circular prerequisites: there is no sequence of courses $(C_0, C_1, \dots, C_{k-1}, C_k)$ such that for every $i < k$, C_i is a prerequisite for C_{i+1}, and also C_k is a prerequisite for C_0.

At this university, courses are very hard, so you can only pass one course each semester. Moreover, to better understand the material, you want to pass courses successively. That is, if in the previous semester you passed some course C, in the current semester you can only pass courses for which C is a prerequisite.

We call a course T *reachable* from S if there exists a sequence of courses

$$(S = C_0, C_1, \dots, C_{k-1}, C_k = T),$$

such that for every $i < k$, C_i is a prerequisite for C_{i+1}.

In this exercise, we are interested in the following questions:

a) How can we model the courses and prerequisites as a directed graph?

b) (For a special case of the graph) is some course T reachable from some other course S?

c) (For the general case of the graph) how many different sequences are there to attend some course T after passing some course S?

In points b) and c) you can assume that the directed graph is represented by a data structure that allows you to traverse the direct successors and direct predecessors of a vertex u in time $\mathcal{O}(\deg_+(u))$ and $\mathcal{O}(\deg_-(u))$ respectively, where $\deg_-(u)$ is the in-degree of vertex u and $\deg_+(u)$ is the out-degree of vertex u.

Example

Consider the curriculum with the five courses: Linear Algebra, Multivariable Calculus, Linear Programming, Convex Optimization, Combinatorial Optimization. Linear Algebra does not have any prerequisite courses and is a prerequisite for Multivariable Calculus and Linear Programming. Multivariable Calculus and Linear Programming are prerequisites for Convex Optimization. Linear Programming is a prerequisite for Combinatorial Optimization (but Linear Algebra is not a prerequisite for Convex Optimization and Combinatorial Optimization). For an overview of prerequisites and reachability see the following table:

Course	**Has Prerequisites**	**Is reachable from**
Linear Algebra	—	Linear Algebra
Multivariable Calculus	Linear Algebra	Multivariable Calculus Linear Algebra
Linear Programming	Linear Algebra	Linear Programming Linear Algebra
Convex Optimization	Multivariable Calculus Linear Programming	Convex Optimization Multivariable Calculus Linear Algebra Linear Programming
Combinatorial Optimization	Linear Programming	Combinatorial Optimization Linear Programming Linear Algebra

Note that the number of different sequences from Linear Algebra to Convex Optimization is two: One sequence is *(Linear Algebra, Multivariable Calculus, Convex Optimization)*, the other sequence is *(Linear Algebra, Linear Programming, Convex Optimization)*.

/ 1 P

a) Model the n courses and its prerequisites as a directed graph: give a precise description of the vertices and edges of this graph $G = (V, E)$ involved (if possible, in words and not formal).

/ 4 P

b) Suppose that there exists an introductory course I which has no prerequisites and each course is reachable from I. Moreover, assume that there exists a unique possibility to pass every course after passing I if you pass courses successively, that is, for every course J there exists a unique sequence of courses $(I = C_0, C_1, \dots, C_{k-1}, C_k = J)$, such that for every $i < k$, C_i is a prerequisite for C_{i+1}.

Using the graph as defined in a), provide an as efficient as possible algorithm that takes as input the graph G and prepocesses it so that queries "is D reachable from C?" can be answered in $\mathcal{O}(1)$ time (that is, this time does not depend on the size of the graph). State the running time of your preprocessing algorithm.

Hint: *Think about the following questions: What is the number of vertices and edges in the graph? How does the graph look like?*

Preprocessing algorithm:

$\mathcal{O}(1)$-time query algorithm:

Running time of preprocessing algorithm (as concisely as possible in Θ notation in terms of n). Justify your answer:

/ 7 P

c) Suppose that you want to find the number $N(S, T)$ of possibilities to attend a course T after passing a course S if you pass courses successively. That is, $N(S, T)$ is the number of different sequences of courses $(S = C_0, C_1, \ldots, C_{k-1}, C_k = T)$, such that for every $i < k$, C_i is a prerequisite for C_{i+1}.

Provide an as efficient as possible *dynamic programming* algorithm that takes as input the graph G from task a) and the courses S and T, and outputs $N(S, T)$. Address the following aspects in your solution and state the runnung time of your algorithm:

1) *Definition of the DP table: What are the dimensions of the table* $DP[\ldots]$? What is the meaning of each entry?
2) *Computation of an entry*: How can an entry be computed from the values of other entries? Which entries do not depend on others?
3) *Calculation order*: In which order can entries be computed so that values needed for each entry have been determined in previous steps?
4) *Extracting the solution*: How can the final solution be extracted once the table has been filled?

Hint: *Think about the following question: Which graph property is implied by the sentence "The curriculum has no circular prerequisites."?*

Size of the DP table / Number of entries: ____________________

Meaning of a table entry:

$DP[\ldots\ldots\ldots\ldots]$: ____

____

Computation of an entry (initialization and recursion):

Scheme continues on the next page.

Order of computation:

Computing $N(S, T)$:

Running time in concise Θ-notation in terms of n and m, where m is the number of edges in G. Justify your answer:

Extra space. Please indicate clearly to which task your notes belong. Please cross out all notes that you do not want to be graded.

Extra space. Please indicate clearly to which task your notes belong. Please cross out all notes that you do not want to be graded.

Extra space. Please indicate clearly to which task your notes belong. Please cross out all notes that you do not want to be graded.

Extra space. Please indicate clearly to which task your notes belong. Please cross out all notes that you do not want to be graded.

Extra space. Please indicate clearly to which task your notes belong. Please cross out all notes that you do not want to be graded.