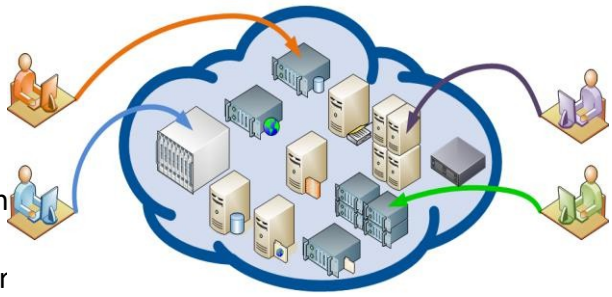
ALGORITHMS 2021-2022

B PROGRAMMING EXERCISE (1 point)

Optimal assignment of tasks to cloud machines using dynamic scheduling



Cloud computing companies such as Amazon have a set of virtual machines (VMs) which they provide to users for a certain fee for the total period of use.
There are many types of machines with different characteristics each, e.g., a virtual machine of the same cost B in r outperforms A in processes that access the hard disk. There may also be a more expensive machine C that outperforms both A and B in performance (but costs more). The problem of choosing the optimal virtual machine each time is often difficult in practice. This is the problem that this exercise deals with.

We have a complex **process** consisting of **N steps** (or **processes**) in the form of a chain: **D1 → D2 → D3 → ..... →DN**. The arrows indicate that the results of **D1** are passed as input to **D2**, those of **D2** to **D3**, a n d so on.
We also have **M virtual machine types**. As input, we also have **two tables**. One is **N X M** and denotes the **total cost to run a process on a virtual machine type**. The second table is **M X M** and denotes the **cost to send data from one machine to another.** An example input is the following:

```
N = 4 processes (□ 1 →□ 2 →□ 3 →□ 4 )
M = 3 types of machines (VM1,VM2,VM3)

    VM1  VM2  VM3
Δ1  5     6    3
Δ2  7     8    5
Δ3  7     8    3
Δ4  2     7    6

    VM1  VM2  VM3
VM1 07    2
VM2 7     0    2
VM3 2     2    0
```

In this example, the complex process has 4 steps and 3 VM types are available. The first type performs the 4 processes at a cost of 5, 7, 7

and 2 respectively. It also communicates with the other two types of machines (second and third) at a cost of 7 and 2 respectively. The second type performs the 4 processes at a cost of 6, 8, 8 and 7 respectively. It also communicates with the other two types of machines (first and third) at a cost of 7 and 2 respectively. Finally, the third type performs the 4 processes at a cost of 3, 5, 3 and 6 respectively. It also communicates with the other two types of machines (first and second) at a cost of 2 and 2 respectively. The machine communication table is generally always square but not necessarily symmetric as it is in the above example.

What needs to be implemented

In this task you have to implement a dynamic programming algorithm to find the minimum cost of the complex process at each step and for each machine.

More precisely, this algorithm will populate a **Costs N X M** table, where each cell **Cost(i,j)** will show the **lowest total cost of the complex process up to step i when step i is executed on machine j**. To execute step i on machine j, **either the results of step i-1** must **already be on machine j or they must be transferred to it from other machines taking into account the communication cost**. In the above example this table will have the following form:

	VM1	VM2	VM3
Δ2	**12**	**13**	**8**
Δ1	**5**	**6**	**3**
Δ3	**17**	**18**	**11**
Δ4	**15**	**20**	**17**

The minimum costs for the first step (D1) are obviously 5, 6, 3 depending on whether it is executed on VM1, VM2, VM3 respectively (we have no previous step).

For the minimum cost up to the second step (D2) we have:

If D2 is executed on VM1 it will be 12 because we have the following cases:

→if D1 is running on VM1 the total cost will be:
Execution Cost (D1,VM1) + Communication Cost (VM1,VM1) + Execution Cost (Δ2,VM1) = 5 + 0 + 7 = 12

→if D1 is running on VM2 the total cost will be:
Execution Cost (D1,VM2) + Communication Cost (VM2,VM1) + Execution Cost (Δ2,VM1) = 6 + 7 + 7 = 20

→if D1 is running on VM3 the total cost will be:
Execution Cost (D1,VM3) + Communication Cost (VM3,VM1) + Execution Cost (Δ2,VM1) = 3 + 2 + 7 = 12

If D2 is executed on VM2 it will be 13 because we have the following cases:

→if D1 is running on VM1 the total cost will be:
Execution Cost (D1,VM1) + Communication Cost (VM1,VM2) + Execution Cost (Δ2,VM2) = 5 + 7 + 8 = 20

→if D1 is running on VM2 the total cost will be:

Execution Cost (D1,VM2) + Communication Cost (VM2,VM2) + Execution Cost (Δ2,VM2) = 6 + 0 + 8 = 14

→if D1 is running on VM3 the total cost will be:
Execution Cost (D1,VM3) + Communication Cost (VM3,VM2) + Execution Cost (Δ2,VM2) = 3 + 2 + 8 = 13
and so on.
Finally, the least expensive execution of the composite process costs 15 and is achieved when the last process is executed on the first virtual machine.

As **input**, you will be given a data file in the following format:

```
4
3

5 6 3
7 8 5
7 8 3
2 7 6

0 7 2
7 0 2
2 2 0
```

That is, the first line will be **N**, the second line will be **M**, followed by a blank line. The next lines will give the **table of execution costs (N X M)** with the costs separated by a blank. Then a blank line will follow and then **the communication cost table (M X M) will be** given with the costs separated by a blank. All costs will always be **integers and non-negative numbers.**

The program should calculate and print on the screen the final table **N X M** with the minimum costs strictly in the following output format:

```
5 6 3
12 13 8
17 18 11
15 20 17
```

(i.e. with the costs separated by a space).

Method of implementation:

1) Use only the **Java** language, following the instructions for submitting java exercises to the Eagle system (**Instructions-Eagle-2022.pdf**).
2) The **only input argument** will be the input file. This argument should be **args[0**] from the command line arguments, which is read by **main.**
3) For the solution use an efficient dynamic programming algorithm, as N and M may be larger and **there will be a time constraint on the execution of the test cases**.

4) The source code should have **concise inline comments** and **comments** above each function, explaining the rationale for your implementation.

Deliverable:

- **The** whole program (**source code**) should be implemented in a **single java file named DPnet.java** (DPnet class).
- This file should be uploaded and **submitted to** the system **Eagle** (eagle.csd.auth.gr) with your account.
- The Eagle system will **automatically execute** your code in various test cases (both overt and covert) and **score it automatically** (maximum score of **100**).
- You can submit to the system up to 20 times. Therefore you should have first implemented it in another environment of your choice (e.g. Netbeans, IntelliJ, etc.) and tested it to **run correctly** and **produce correct results** before submitting.
- **Start of submissions: Monday 30/5/2022 10:00am.**
- **End of submissions: Thursday 9/6/2022 11:59pm.**

Clarifications:

- It is allowed to use ready-made code freely available (ready-made implementations with appropriate customization by you) provided that the source is clearly indicated in the comments. However, the programs will be checked by an automatic copy detection system. If copying between students is detected, these students will be zeroed out.
- At the top of the java file with the source code, be sure to include in comments your full name, AEM and academic email.
- *Students should be prepared to give oral explanations of their implementation if asked.*
- ***The grades of the exercises will also be valid for the September or degree examinations.***

Another test case:

Entrance:

```
5
4

5 1 3 2
4 2 1 3
1 5 2 1
2 3 4 2
1 1 3 1

0 1 2 4
1 0 2 3
2 2 0 1
4 3 1 0
```

Exit:

```
5 1 3 2
6 3 4 5
5 8 6 6
7 9 10 8
8 9 12 9
```