Container Service Preheating

Limits: 10 sec., 1024 MiB

In cloud computing environments, virtual machines (VMs) host containerized microservices. When users request to launch a container, the scheduler initially allocates containers to existing VMs, provisioning additional VMs if necessary. However, creating new VMs takes time and it can introduce additional latency for clients. This can result in a poor quality of experience (QoE). We can mitigate this by purchasing a set of VMs before the actual container requests. This strategy is known as "preheating".

The demand for new containers is unpredictable. Various VM types are available for purchase. Each type of VM has corresponding resource specifications (CPU and memory) and price per unit of time. For each container request, you must decide which VM to run the container on. Once a container is associated with a VM, the container runs on it, occupying a certain amount of resources until the container is deleted. After a container is deleted, the resources are released and can be used by other containers.

You need to determine the optimal number of VMs, the optimal time to pre-provision them and how to place containers into those VMs in order to minimize the total cost, which has two components:

- Resource Reservation Cost (RC): This is the cost of running the VMs.
- Container Launch Delay Cost (DC): This is the cost incurred when a user request cannot be immediately fulfilled due to a lack of available VMs.

You can find more information about scoring in the Scoring section of the problem statement.

Please note that this is an interactive problem. Your solution will communicate with the interactor through the stdin/stdout streams.

Interaction protocol

The first line of the input contains two integers m and d — the number of available VM types and VM creation delay. The following m lines contain the description of the types. Each line contains three space-separated integers: cpu_i, mem_i, and $price_i$ which represent the number of CPU units, the memory, and the price of the VM type. The VM types are enumerated from 1 to m.

The next line contains a single integer t — the number of time steps of the interaction. The first line of each time step contains an integer e_j — the number of events during this time step. In case if $e_j = 0$, it is followed by an integer cnt_j. This means that there are no events on the current time step, as well as $cnt_j - 1$ following time steps (cnt_j empty time steps in total). There will be no input for any of the next $cnt_j - 1$ time steps and you have to process each of them in a single batch. For each of the cnt_j time steps you have to print a list of actions as described below.

In case if $e_j > 0$, the following e_j lines contain events for the current time step. There are two types of events:

- 1 id_{jk}^{cont} cpu_{jk}^{cont} mem_{jk}^{cont}: a new container creation request has arrived. The container is identified by id_{jk}^{cont} and requires cpu_{jk}^{cont} CPU units and mem_{jk}^{cont} memory. This request can be fulfilled immediately within the current time step.
- 2 id_{jk}^{cont}: a container with an identifier id_{jk}^{cont} must be shut down. The resources occupied by this container might be used during the next time step.

After your solution has read the information about the events during the current time step, you have to print a line with an integer a_j — the number of actions that you want to perform during this time step. Then print a_j lines, one for each action. There are three possible actions you can perform:

- 1 id_{jk}^{vm} $type_{jk}^{vm}$ — start a new VM with an identifier id_{jk}^{vm} and a type $type_{jk}^{vm}$. This action takes d time steps, meaning the VM becomes available at time step $x + d$ if provisioned at time step x.

- 2 id_{jk}^{vm} — shut down a VM with an identifier id_{jk}^{vm}. To perform this action, the VM with an identifier id_{jk}^{vm} should have no containers allocated before the beginning of this time step. If the last container on a VM is terminated at step x, the VM can be shut down starting from step $x + 1$.
- 3 id_{jk}^{cont} id_{jk}^{vm} — allocate the container with an identifier id_{jk}^{cont} on a VM with an identifier id_{jk}^{vm}. To perform this action, the container with an identifier id_{jk}^{cont} has to not be yet allocated and the VM with an identifier id_{jk}^{vm} has to have at least $cpu_{id_{jk}^{cont}}^{cont}$ CPUs and $mem_{id_{jk}^{cont}}^{cont}$ memory available.

After printing a_j lines with your actions, ensure the output is flushed, so that the interactor can read your actions and proceed with the next time step.

Please note, that if $e_j = 0$, you have to print cnt_j blocks of actions, one for each time step with no events. You should flush the output only after you printed actions for all cnt_j time steps. This is needed to improve the performance of the i/o of your solution.

If the interactor prints the value $e_j = -1$ — this means that one of your actions during the previous step was invalid. Your program has to exit immediately. In this case, you will get the `Wrong Answer` verdict.

After t time steps, the interactor prints a single integer e_{t+1}, which equals 0 if your last action is valid or -1 if the last action is invalid. Afterwards, the interaction is over and your solution must exit.

Input constraints

$1 \le m \le 47$,
$1 \le d \le 40$,
$1 \le cpu_i, cpu_{jk}^{cont} \le 10^5$,
$1 \le mem_i, mem_{jk}^{cont} \le 2 \cdot 10^6$,
$1 \le price_i \le 2 \cdot 10^5$,
$1 \le id_{jk}^{cont} \le 10^9$,
each container has a unique identifier,
$1 \le t \le 2 \cdot 10^7$,
$-1 \le e_j \le 2.4 \cdot 10^4$,
$1 \le \sum e_j \le 2.4 \cdot 10^4$.
$1 \le cnt_j \le 20$,

Output constraints

$1 \le a_j \le 3 \cdot 10^7$,
$\sum a_j \le 3 \cdot 10^7$,
$1 \le id_{jk}^{vm} \le 10^9$,
each VM should have a unique identifier,
$1 \le type_{jk}^{vm} \le m$.

Interaction example

Interactor output	Solution Output	Description
2 4		We have two VM types which warm up for 4 time steps.
8 1024 1		8 CPUs, 1024 memory, cost 1 per time unit.
32 4096 2		32 CPUs, 4096 memory, costs 2 per time unit.
14		The interaction will last for 14 time units.
0 2		No events for the first 2 time steps.
	2	Two actions on the first time step.
	1 1 2	Start VM of type 2.
	1 2 2	Start VM of type 2.
	0	Do nothing on time step 2.
0 1		No events on time step 3.
	0	No actions on time step 3.
1		One event on time step 4
1 1 2 32		Create a container with 2 CPUs and 32 units of memory.
	0	The VMs are not ready yet, do nothing for now.
0 1		No events on time step 5.
	1	One action.
	3 1 2	Allocate container 1 to the VM 2.
2		Two events on time step 6.
1 2 24 256		Container with 24 CPUs and 256 memory;
1 3 16 512		Container with 16 CPU, 512 memory.
	3	Three actions.
	3 3 1	Allocate container 3 to VM 1.
	3 2 2	Allocate container 2 to VM 2.
	1 3 1	Start a new VM of type 1.
0 6		No events on time steps 7, 8, 9, 10, 11, 12.
	0	No actions on time step 7.
	0	No actions on time step 8.
	2	Two actions on time step 9.
	1 4 1	Start new VM with type 1.
	1 5 1	Start new VM with type 1.
	0	No actions on time step 10.
	0	No actions on time step 11.
	1	One action on time step 12.
	1 6 2	Start new VM with type 2.
6		Six events on time step 13.
1 5 8 1024		New container.
2 2		Shut down container 2.
2 1		Shut down container 1.
2 3		Shut down container 3.
1 6 8 1024		New container.
1 7 8 1024		New container.
	3	Three actions on time step 13.
	3 5 4	Allocate container 5 on VM 4.
	3 6 5	Allocate container 6 on VM 5.
	3 7 3	Allocate container 7 on VM 3.
0 1		No events on time steps 14.
	2	Two actions on time step 14.
	2 1	Shut down VM 1.
	2 2	Shut down VM 2.

Scoring

If on any time step your solution performs an invalid action, violates one of the output constraints, or fails to allocate one of the containers, it is considered to be incorrect and for such a test case you receive 0 points. Otherwise, your score is calculated as follows:

$$score = \left\lfloor \frac{TC \cdot BP}{RC + DC \cdot 10} \cdot 10^7 \right\rfloor$$

Where:

- TC is the total number of CPU time units in all container requests:

  $$TC = \sum (shutdown_{jk} - alloc_{jk}) \cdot cpu_{jk}^{cont}$$

  Where $shutdown_{jk}$ and $alloc_{jk}$ are the time units of container shutdown and allocation to vm respectively. If the container didn't shut down then $shutdown_{jk} = t + 1$ for such container.

- BP is the best possible price per CPU between all VM types:

  $$BP = min \frac{price_i \cdot 10^{-4}}{cpu_i}$$

- RC is the Resource Reservation Cost, which is the total cost of running all the virtual machines. Suppose that in total you started v VMs, s-th of them was launched at time step VM_s^{start}, stopped at time step VM_s^{end} (if you didn't stop a VM then $VM_s^{end} = t + 1$) and has type VM_s^{type}. Then the Resource Reservation Cost equals:

  $$RC = \sum_{s=1}^{v} (VM_s^{end} - VM_s^{start}) \cdot price_{VM_s^{type}} \cdot 10^{-4}$$

- DC is the Container Launch Delay Cost. Suppose that in total there were c container requests, f-th of them appeared at time step C_f^{start}, and was allocated to a VM at the time step C_f^{alloc}. Then the Container Launch Delay Cost equals:

  $$DC = \sum_{f=1}^{c} (1.1^{(C_f^{alloc} - C_f^{start})} - 1)$$

Submissions

- The execution time limit is 10 seconds per test case, and the memory limit is 1024 mebibytes.
- The code size limit is 64 kibibytes.
- The compilation time limit is 1 minute.
- There are 20 provisional test cases. Your submissions will be evaluated on the provisional set during the submission phase.
- You can submit your code once every 30 minutes, and you will get feedback with your score for each of the provisional tests.
- There will be 200 test cases in the final testing after the submission phase is over. Please note that provisional tests are **not included** in the final testing. The final results will be announced in one week.