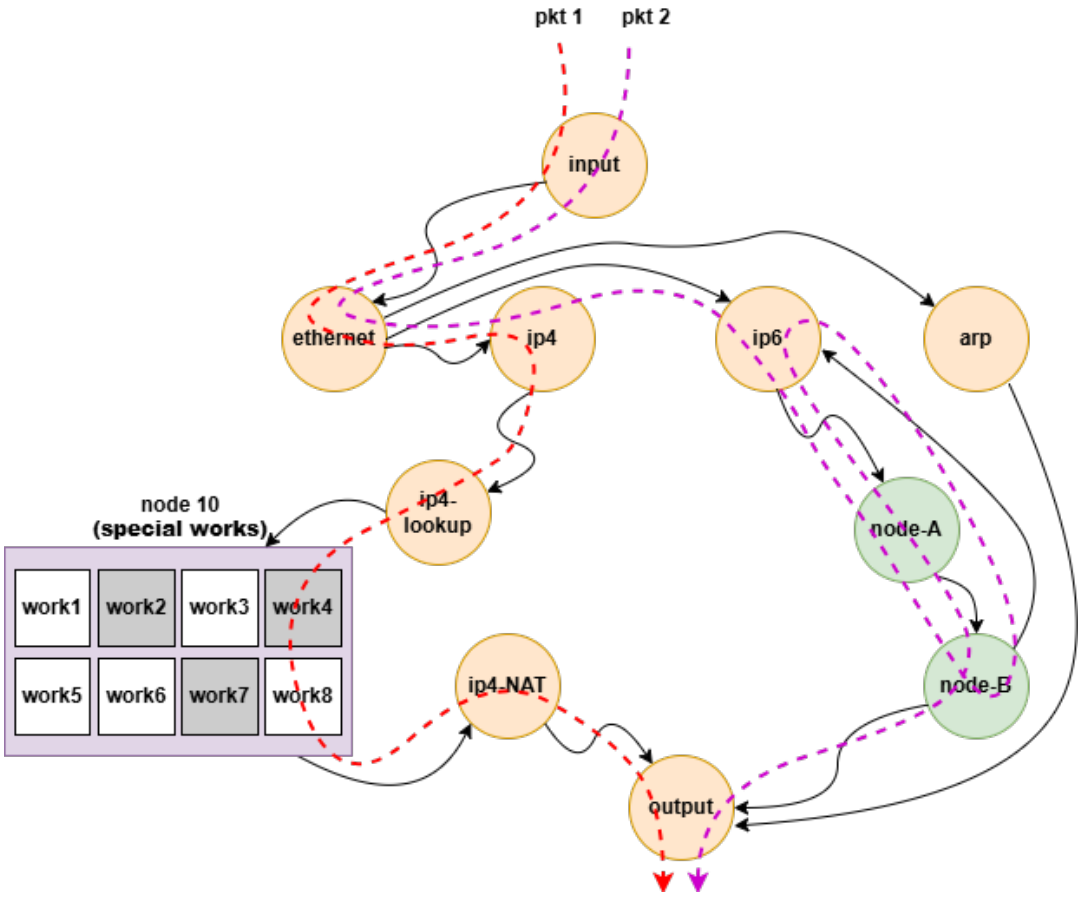
Problem A. HPF Multi-Core Graph Scheduling

Type: `Interactive`

HPF (High-Performance Forwarding) is a core component for CPU-based forwarding devices. With the evolution towards many-core architectures, scheduling algorithm optimization for 32+ cores has become a critical bottleneck in achieving high-performance software forwarding. By optimizing the scheduling algorithm, costs can be reduced while ensuring forwarding performance on lightweight hardware.

In this problem, you are asked to design a scheduling algorithm aimed at optimizing both throughput and latency for multi-core packet forwarding. The HPF forwarding process can be abstracted as a computation graph (potentially cyclic) similar to the following picture:



Incoming nodes will first arrive at the input node, been processed by a series of nodes, and finally be sent out through the output node. There are 7 types of packets, different packets may traverse different paths. The computation graph and the computation paths are fixed throughout this problem.

Each type of packet has a latency constraint. The latency of a packet is the duration between it arrives and being sent out. The score will be penalized each time the latency constraint of a packet is violated, so it is critical to schedule the task orders to avoid latency violations if possible.

One key feature of HPF is **batch processing**. In a task of HPF, we select a batch of packets on the same computation node, and perform their computation on a specific core assigned by your scheduling algorithm. The computation time depends on the computation node and the batch size (in practice, enlarging batches is likely to improve the average processing efficiency up to a certain range).

Note that node 8 is a special node. This node models 8 special work that possibly need to be completed. When packet i reaches node 8, a value $work_i$ will be revealed. The j-th bit $work_i[j]$ of $work_i$ is set to 1 if the j-th special work on node 8 needs to be done for packet i. The cost for processing a batch of packets on node 8 will additionally include the sum of the special work that need to be completed for all packets

in the batch.

The cost model of the HPF system will be specified in later sections.

Another key feature of HPF is **multi-core scheduling**. As previously mentioned, a scheduling decision involves selecting a batch of packets at a node and assigning them to a core for batch processing. There are *numcores* cores available; a core can handle only one task at a time. Each time a packet is reassigned to another core, a computational penalty is incurred.

The core challenge to your algorithm is to wisely make these scheduling decisions, such that high throughput is achieved without violating a large percentage of latency constraints.

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

Interaction Protocol

The Computation Graph: The first 28 lines of the input describe the computation graph and its corresponding costs. These values are fixed throughout all testcases. Their meanings are explained as follows:

First there are 7 lines, indicating the computation path for each type of packet. Each line i starts with an integer l_i, denoting the length of the computation path for packet type i. Next l_i integers follow, describing the node indices on the computation path in order. The indices may duplicate due to cycles.

Next there are 20 lines, each line i starts with an integer b_i, denoting the maximum batch size for node i. Next b_i integers follow, the j-th integer $c_{i,j}$ denotes the computation cost for processing a batch of j packets on node i.

The next line contains 8 integers, where the i-th integer $special\_cost_i$ denotes the computation cost for the i-th special work on node 8.

Subtasks: The next line contains 3 integers, $numcores$, c_{switch} and c_r, denoting the number of cores, the cost for reassigning a packet to another core, and the cost for trying to receive incoming packets.

Next there are 5 subtasks, they are sampled from the same packet type distribution, but with different throughput. You need to solve each subtask independently. Each subtask is described as follows: The first line of a subtask contains an integer n, which specifies the number of packets in this subtask. After reading the integer n, you should start the interaction to receive further data.

You can perform four types of actions:

1. **R** t: Try to receive incoming packets. In order to make this action you should print a character R followed by an integer t denoting the start time of this call. This action is executed on an additional core with $coreId = 0$.

After that, you should first read an integer p — the number of packets that you received, and then p lines of data describing the packets. Each line contains 4 integers i, $arrive_i$, $type_i$ and $timeout_i$ — the index of packet i, the arrive time of packet i, the type of packet i, and the timeout threshold of packet i. You will receive all packets that arrive after the previous call of action 1 and no later than time t. The incoming packets are ordered according to their arrival time.

If you read a value $p = -1$, this means that the time t that you printed is invalid. In this case, you have to stop the execution of your program immediately, and you will receive a Wrong Answer verdict for the current test case.

This action takes time c_r: the start time of the action is t and the end time is $t + c_r$. You can start the next action 1 on core 0 at the time $t + c_r$ or later.

2. **E** t $coreId$ $nodeId$ s id_1 id_2 $\cdots$ id_s: Execute a task. In order to make this action you should print a character E, followed by an integer t denoting the start time of this task, an integer $coreId$ denoting the core index for executing this task, an integer $nodeId$ denoting the index of the computation node, an integer s denoting the number of packets in the batch, and s integers id_k indicating the IDs of the packets in this batch. The value t should be strictly greater than the end time of the previous action on

core $coreId$ (or greater than 0 for the first action on this core). The computation node $nodeId$ must be on the computation path of all packets in this batch. Each packet id_k must be ready for execution on the node $nodeId$ at time t.

If $nodeId = 8$, this action takes time $t_a = c_{nodeId,s} + numSwitches \cdot c_{switch} + \Sigma_{i=1}^{s} special\_cost[id_i]$ on core $coreId$, where $numSwitches$ denote the number of packets that are reassigned to another core; $special\_cost[j] = \Sigma_{k=1}^{8} work_j[k] \cdot special\_cost_k$ denote the overall cost for handling all special work of packet j on node 8. Otherwise this action takes time $t_a = c_{nodeId,s} + numSwitches \cdot c_{switch}$. The start time of the action is t and the end time is $t + t_a$. You can start the next action 2 on core $coreId$ at the time $t + t_a$ or later.

If nodeId is the last node on the computation path of packet i in the batch, then packet i is considered to finish processing at time $departure_i = t + t_a$.

3. **Q** t i: This action queries the value of $work_i$, which can be interactively revealed after packet i reaches node 8. This action is valid only if node 8 is on the execution path of packet i, and when at time t, packet i has finished execution on the previous node of node 8. This action can be executed on packet i at most once.

After this action, you should read an integer $work_i$ — the indicator bitmap of the special works corresponding to packet i. If $work_i = -1$ then the action was invalid.

4. **F**: Finish execution of this subtask. This action needs to be performed when all n packets have completed processing.

When all subtasks have finished, the interaction is over. Your solution must stop its execution. You will receive an Accepted verdict for this test case and your score will be calculated.

Input constraints

$2 \le n \le 10^4$,

$1 \le m \le 7$,

$1 \le numcores \le 32$,

$1 \le l_i \le 30$,

$1 \le b_i \le 128$,

$1 \le c_{i,j} \le 10^4$,

$1 \le special\_cost_i \le 100$,

$1 \le c_{switch} \le 20$,

$c_r = 20$,

$1 \le arrive_i \le 5 \cdot 10^6$,

$1 \le type_i \le 7$,

$1 \le timeout_i \le 10^5$,

$0 \le work_i \le 2^8 - 1$.

Output constraints

$1 \le t \le 10^7$,

$1 \le coreId \le numcores$,

$1 \le nodeId \le 20$,

$1 \le s \le b_{nodeId}$,

$1 \le id_i \le n$.

Please note, that the time t of each action should be no less than the end time of the previous action (or greater than 0 for the first action).

Scoring

HPF's primary goal is throughput optimization, but additional penalties apply if a packet exceeds its latency constraint. Thus the algorithm must balance between these two metrics.

The design of subtasks models how we test the throughput of the HPF system. For a fixed packet distribution, we gradually increase the incoming rate of the packets, until the system become overloaded.

If on any time step your solution performs an invalid action, or violates one of the output constraints, it is considered to be incorrect and for such a test case you receive 0 points. Otherwise, we first grade the subtasks separately. Your score for the i-th subtask is calculated as follows:

$$subscore_i = \max(0, \lfloor (throughput_i - 10^4 \cdot timeoutRate_i) \cdot 10^2 \rfloor),$$

where $throughput_i = \frac{10^6 \cdot (n-1)}{numcores \cdot \left(\max_{1 \le j \le n} departure_j - \min_{1 \le j \le n} arrive_j\right)}$ is defined as the average number of packets been processed per second (a second is 10^6 time units), and $timeoutRate_i = timeoutPkts_i / n$ is defined as the percentage of packets whose latency constraint is violated. The latency constraint of packet j is violated if $processed_j - arrive_j > timeout_j$.

Your overall score for a testcase is the maximum score among all subtasks in this testcase:

$$score = \max_{1 \le i \le 5} subscore_i.$$

Your overall score is the sum of your scores over all test cases.

Note

There are 100 provisional test cases. Your submissions will be evaluated on the provisional set during the submission phase.

You can submit your code once per 10 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.