Summary of the Tasks

Develop algorithms to complete tasks involving the relative order of \( n \) distinct integers.

- The integers are random and unknown to you. All you know is that they are named \( a_1, a_2, \ldots, a_n \).
- For each task, your approach must work no matter what the order of the integers is.
- A helpful machine \( M \) is available. The machine knows the relative order of these integers. To use it, you enter the index of two integers into the machine and it will tell you which of the two corresponding integers is larger.
- For each task, there is a limit on the number of times you can use the machine. This limit applies no matter what the relative order of the \( n \) integers happens to be.
- Your memory is perfect and you can remember (or record) the result every time you use the machine.

Details of the Three Tasks

<table>
<thead>
<tr>
<th>( n )</th>
<th>Task</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Determine the largest integer.</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>Determine both the largest integer and the smallest integer.</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>Determine the second largest integer.</td>
<td>9</td>
</tr>
</tbody>
</table>

Solution for Task 1

We will keep track of an index named \( c \). The “current maximum” will be at index \( c \). Begin by setting \( c = 1 \). Then, repeatedly set \( c = M(c, i) \) for \( i = 2, 3, \ldots, 7 \). (Setting \( c = M(c, i) \) means we compute the value of \( M(c, i) \) and then update \( c \) to be this value.) This uses the machine exactly 6 times. We can then declare \( a_c \) to be the largest integer because the machine has told us either directly or indirectly that it is larger than each of the other integers.

Can you see why this algorithm works? If the index of the largest integer is 1, then we will see \( M(1, i) = 1 \) at each stage and the value of \( c \) will never change. If the index of the largest integer is not 1, then at some point the value of \( c \) will change. In particular, the value of \( c \) will change from 1 to \( i \) the first time we see a comparison of the form \( M(1, i) = i \) for some \( i \geq 2 \). The value of \( c \) will continue to change each time we find an integer that is larger than our “current maximum”.

An example using the provided tool is on the next page.
The value of \( c \) just before each use of the machine and right after the last use of the machine will be

\[ 1, 2, 2, 4, 5, 5, 7. \]

**Solution for Task 2**

Begin by computing \( M(1, 2), M(3, 4), M(5, 6), \) and \( M(7, 8) \) and recording the answers as indices \( a, b, c, d \). Name the other indices \( e, f, g, h \). At this point we have used the machine 4 times and know the largest integer must be at index \( a, b, c \) or \( d \). This is because the machine has told us that the integers at indices \( e, f, g \) and \( h \) are each smaller than at least one of the integers at indices \( a, b, c \) or \( d \). Similarly, the smallest integer must be at index \( e, f, g \) or \( h \).

Now notice that our solution for Task 1 generalizes to give a way to find the largest of \( n \) integers using the machine \( n - 1 \) times. Moreover, if we adjust things by using \( c \) to keep track of the index of a “current minimum”, we can also use it to find the smallest of \( n \) integers using the machine \( n - 1 \) times.

Using this generalization, we can find the largest integer among those at indices \( a, b, c \) and \( d \) using the machine 3 more times. We can also use it to find the smallest integer among those at indices \( e, f, g \) and \( h \) using the machine 3 more times. These two values are the largest and smallest integers overall and we used the machine \( 4 + 3 + 3 = 10 \) times.
Solution for Task 3

Begin by using the machine 4 times as in the solution for Task 2, and defining $a$, $b$, $c$, and $d$ in the same way. Then compute $M(a, b)$ and $M(c, d)$, recording the answers as the indices $x$ and $y$. Next, compute $M(x, y)$ and record the answer as the index $z$. You can think of this approach as a standard “tournament” or “bracket” in a competition where only the winners move on at each stage.

The integer $a_z$ must be the largest integer because it is the only integer that has not been declared smaller than some other integer by the machine. Moreover, the second largest integer can only have been declared smaller than $a_z$. So to find the second largest integer, we need only find the largest integer among those at indices entered into the machine with $z$. Note that $z$ was entered into the machine 3 times so we can use our generalized solution to Task 1 by using the machine 2 more times to determine the second largest integer. In total, we used the machine $4 + 2 + 1 + 2 = 9$ times.

Note

We have only shown that we can complete these tasks within the limits of 6, 10 and 9. Do we need the limits to be this high or can they be lowered? It turns out that they cannot be lowered; they are optimal. That is, for each task, there do not exist correct algorithms that always use the machine strictly fewer times than the given limit. Proving this is very challenging.