Question 1

Phase	Distance	Number of Rounds	Active Processors	Number of Messages Sent
1	k	$2k$	n	$< 2nk$
2	k^2	$2k^2$	$< \frac{n}{(k+1)}$	$< \frac{2nk^2}{(k+1)}$
3	k^3	$2k^3$	$< \frac{n}{(k^2+1)}$	$< \frac{2nk^3}{(k^2+1)}$
4	k^4	$2k^4$	$< \frac{n}{(k^3+1)}$	$< \frac{2nk^4}{(k^3+1)}$
5	k^5	$2k^5$	$< \frac{n}{k^4+1}$	$< \frac{2nk^5}{k^4+1}$
⋮	⋮	⋮	⋮	⋮
i	$k^i = n$	$2k^i = 2\text{n}$	$< \frac{n}{k^{i-1}+1}$	$< \frac{2nk^i}{k^{i-1}+1}$

$\because k^i = n \quad \therefore i = \log_k n$

Time Complexity:

The time complexity is the number of rounds needed by an algorithm to terminate. In each phase, it takes $2k^i$rounds to complete ('i' is the phase number) and it also takes n-1 rounds for sending <END> message. Therefore, the time complexity is:

$$f(n) = 2k + 2k^2 + \cdots + 2k^i + n - 1 = n - 1 + 2\sum_{i=1}^{\log_k n} k^i$$

$$= n - 1 + 2\left(k^{\log_k n+1} - 1\right) < n + 2\left(k^{\log_k n+1} - 1\right)$$

$$= n + 2k^{\log_k n+1} - 2 = \text{n} + 2k^{\log_k n}\text{k} - 2 = n + 2nk - 2$$

$$< 3nk \qquad (\because k \geq 1 \ \therefore nk > n)$$

Therefore, the Time Complexity is $\mathbf{O(nk)}$

Communication Complexity:

The communication complexity is the number of messages sent by an algorithm. In each phase, it takes $\frac{2nk^i}{k^{i-1}+1}$ messages ('i' is the phase number) plus n-1 <END> message at the end. Therefore, the communication complexity is:

$$2nk + \frac{2nk^2}{(k+1)} + \frac{2nk^3}{(k^2+1)} + \cdots + \frac{2nk^i}{k^{i-1}+1} + (n-1)$$

$$< 2nk + \frac{2nk^2}{k} + \frac{2nk^3}{k^2} + \cdots + \frac{2nk^i}{k^{i-1}} + (n-1)$$

$$= 2nk + 2nk + 2nk + \cdots + 2nk + (n-1)$$

$$= 2nk \log_k n + (n-1)$$

$$< 2nk \log_k n + n$$

Therefore, the Communication Complexity is $\boldsymbol{O(nk \log_k n)}$

Question 2

Algorithm Description:

Input: Processor id
Output: the larger of the ids of the two end processors

Each process transmits the leftmost processor's id to its right neighbor in the line. When the rightmost process receives this incoming id, it compares it with its own. If the leftmost process's id is smaller than its own, it replaces it with its own id, and then sends a response message back. Each process sends a response containing the larger end processor id from right to left. Consequently, all processors know what the larger end processor is.

Java Implementation:

Algorithm is implemented in *LargerEndProcessor.java*

Time Complexity

When considering communication from left to right, there are n-1 rounds, and when communicating from right to left, there are also n-1 rounds. Additionally, the leftmost node's sending of <end> takes 1 round. Therefore, the overall time complexity is **O(n)**, where n is the number of processors in the network line.

$$f(n) = (n-1) + (n-1) + 1 = 2n - 1 < 2n$$

Communication Complexity

When communicating from the leftmost to the rightmost processor, there are n-1 messages, and when communicating from the rightmost back to the leftmost processor, there are another n-1 messages. Additionally, there is one <end> message. Hence, the communication complexity is **O(n)**, where n is the number of processors in the network line

$$f(n) = (n-1) + (n-1) + 1 = 2n - 1 < 2n$$

Question 3

Algorithm Description:

Input: Process id
Output: The second largest process id

In the algorithm, each message contains a destination, along with two data items. The first data item stores the current largest id, and the second data item holds the second-largest id. During the first round, each process sends its own id and a placeholder to its right neighbor. Upon receiving the message, the process performs a comparison, updating the current largest and second largest ids in the received data. This message is then passed to the next round until the final largest and second largest values are determined. When a process receives a message with its own id, it knows that it has visited all processors in the ring, prompting it to return the second largest id. Subsequently, each process sends a response message containing this second largest id throughout the ring. Consequently, all processors know what the second largest id is.

Java Implementation:

Algorithm is implemented in *SecondLargest.java*

Termination

Once the processor with the second largest id receives a message with its own id, it returns the value and creates an <END> message.

The <END> message is never discarded, but it is always forwarded to the right neighbor; hence all processors will receive the <END> message.

After forwarding the <END> message the algorithm terminates, so all the processors terminate the execution of the algorithm

Proof of Correctness (correct output)

The message containing the largest processor id and the second largest processor id will survive, while other messages will be discarded during the message comparison. Let P_{2ndmax} represent the processor with the second largest id. The algorithm ensures that the message containing P_{2ndmax}'s id is never discarded.

As a result, all processors will receive and forward the message containing P_{2ndmax}'s id. When P_{2ndmax} receives a message with its own id, it indicates that all other processors have also received it. Therefore, P_{2ndmax} can determine that it has the second largest id and proceeds to return this value.

Subsequently, each process sends a response message containing this second largest id throughout the ring. This ensures that all processors become aware of the second largest id in the network.

Time Complexity

The time complexity of the algorithm is **O(n)**, where n is the number of processors in the network ring.

This is because the number of rounds is the count of message exchanges within a ring. In this algorithm, messages circulate through the entire ring, visiting all n processors. Therefore, the time complexity is O(n).

Communication Complexity

The communication complexity of this algorithm is **$O(n^2)$,** where n is the number of processors in the network ring.

This is because all processors are in a ring, and each node sends one of its own messages and forwards (n-1) messages from other processors in the ring, totaling (n-1) + 1 = n messages for each processor. With n processors in the ring, the communication complexity is $O(n^2)$.

$$f(n) = \big(1 + (n-1)\big) \times n = n^2$$