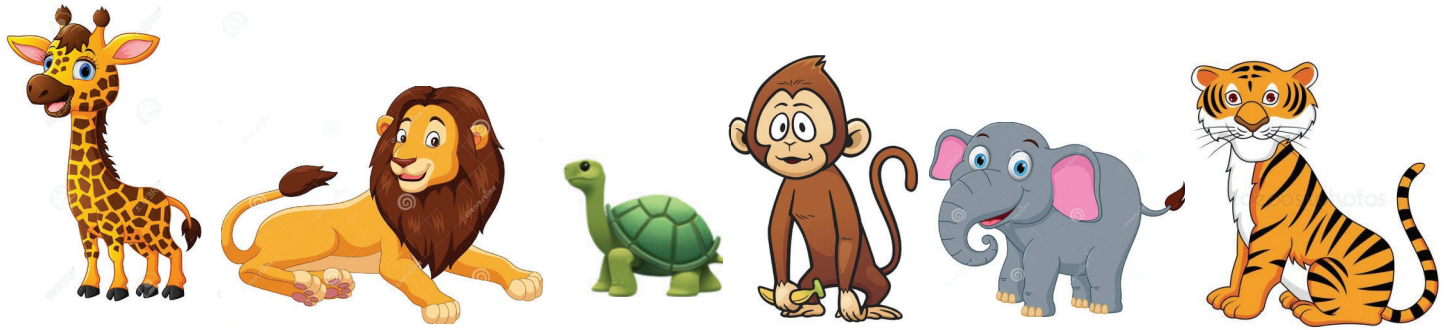
CS 580: Algorithm Design and Analysis

Finding the maximum

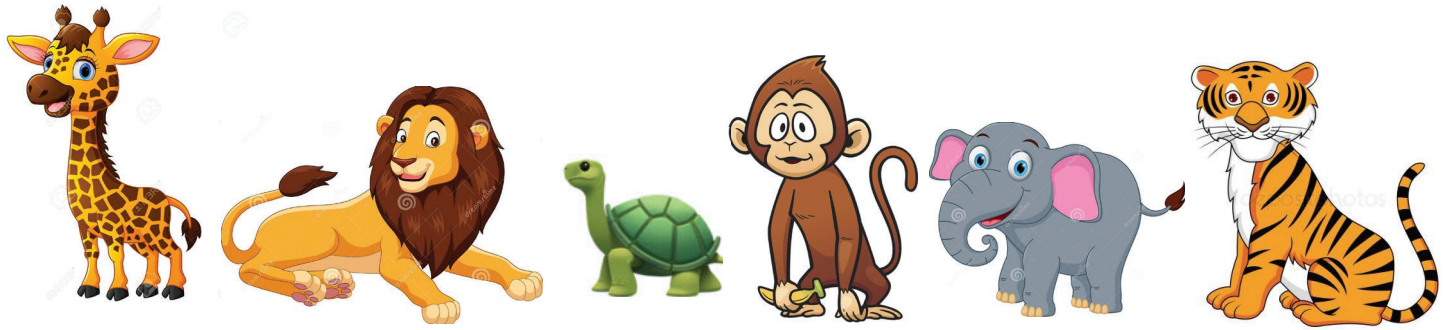
Input:



Goal: find the maximum element (or one of them if multiple ones exist) using comparison queries.

Finding the maximum

Input:

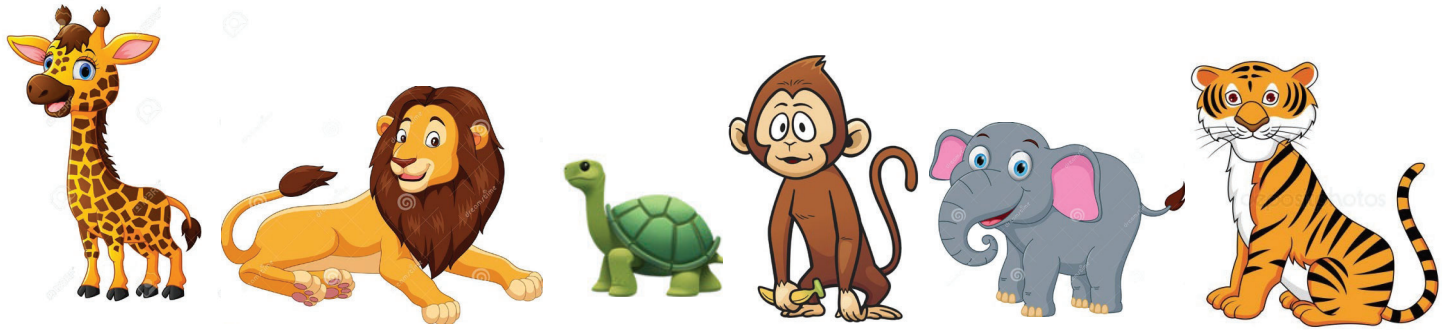


Goal: find the maximum element (or one of them if multiple ones exist) using comparison queries.

Q: How many comparisons are needed?

Finding the maximum

Input:



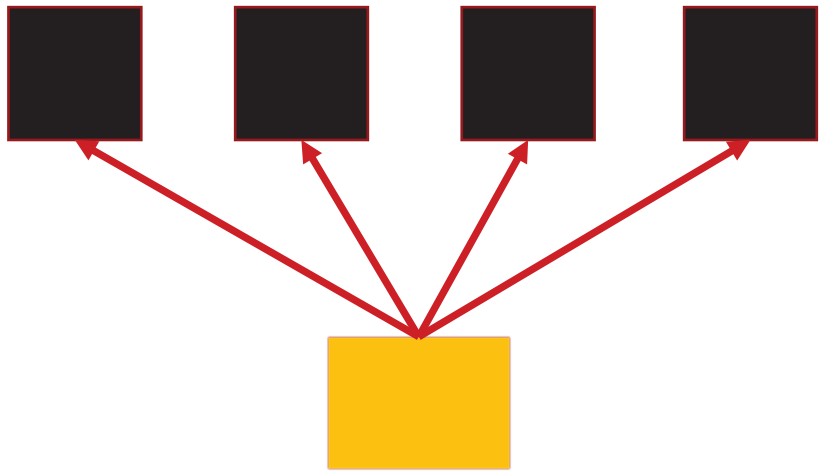
Goal: find the maximum element (or one of them if multiple ones exist) using comparison queries.

Q: How many comparisons are needed?

A: Depends on the number of **rounds** allowed.

Algorithms in rounds

Algorithm that runs in k rounds can also be seen as: central machine issues in each round j a set of queries, one to each processor, then waits for the answers before issuing the next set of parallel queries in round j+1.



Query complexity in k rounds informs how many processors are needed to achieve a parallel time of k.

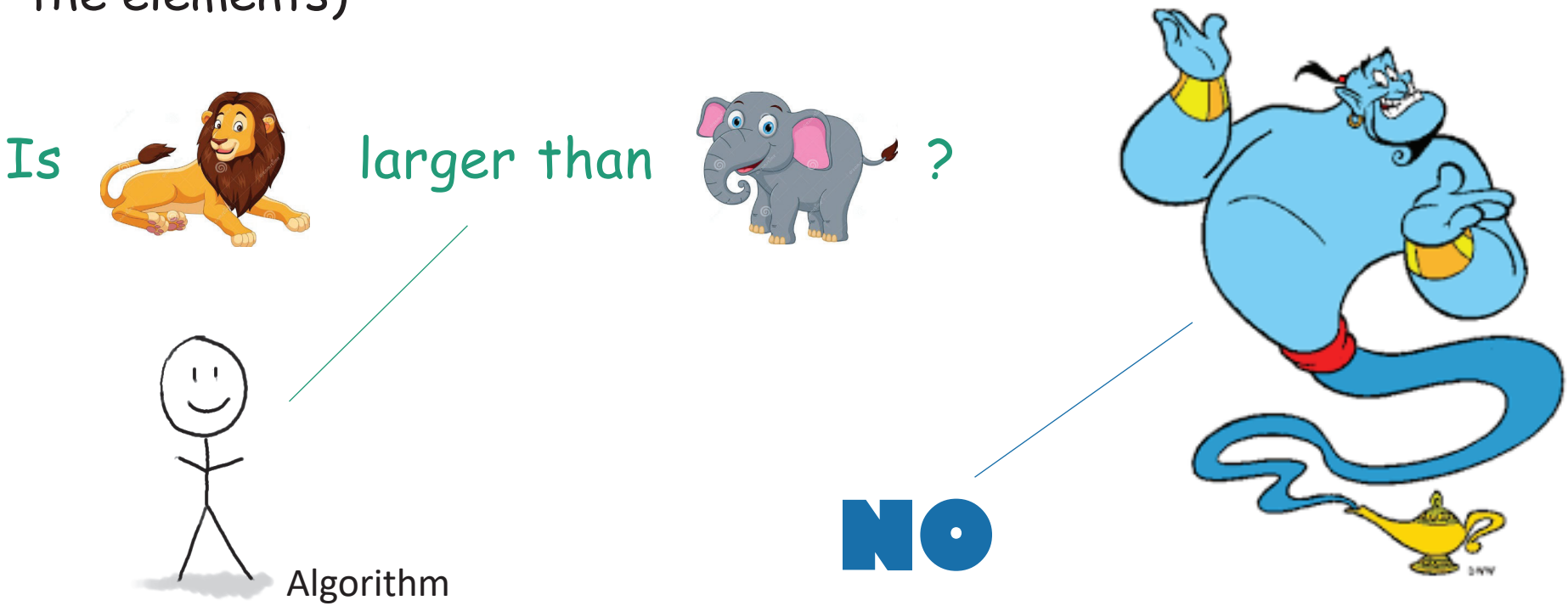
Why rounds?

- Minimizing the number of rounds is important when computation is done by many (small) computers that interact over a network – e.g. phones, laptops

- Scenarios where rounds are expensive: crowdsourcing, blockchain

Recall the comparison model

- An algorithm A in the comparison model gets input vector x
- A can do anything except open up the contents of the entries x_i
- Algorithm A has access to an oracle O that, given any query $x_i < x_j$? can return True/False (i.e., O does the work of inspecting the elements)



Algorithms with rounds

Input: vector $x = (x_1, \ldots, x_n)$

{Internal computation}

Submit set of queries S_1 to O, then get back the answers

{Internal computation}

Submit set of queries S_2 to O, then get back the answers

. . .

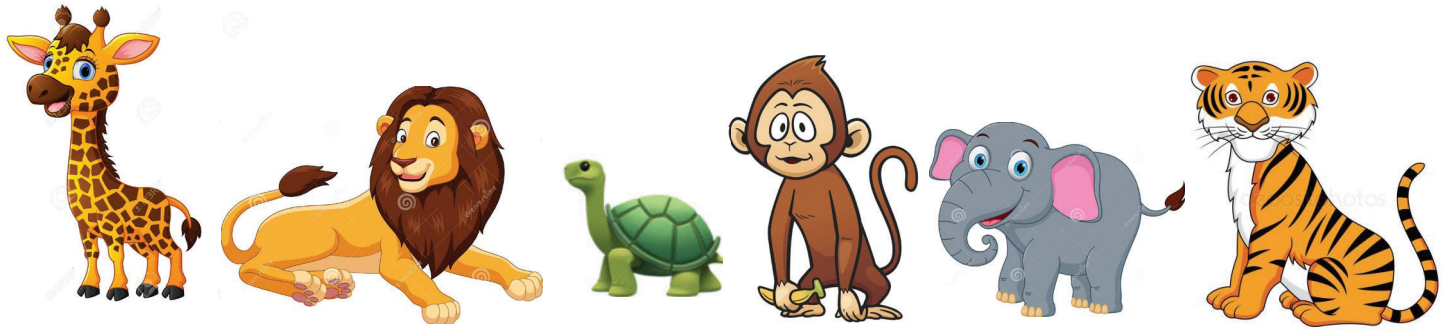
Submit set of queries S_k to O, then get back the answers

{Internal computation}

Output

Finding the maximum

Input:



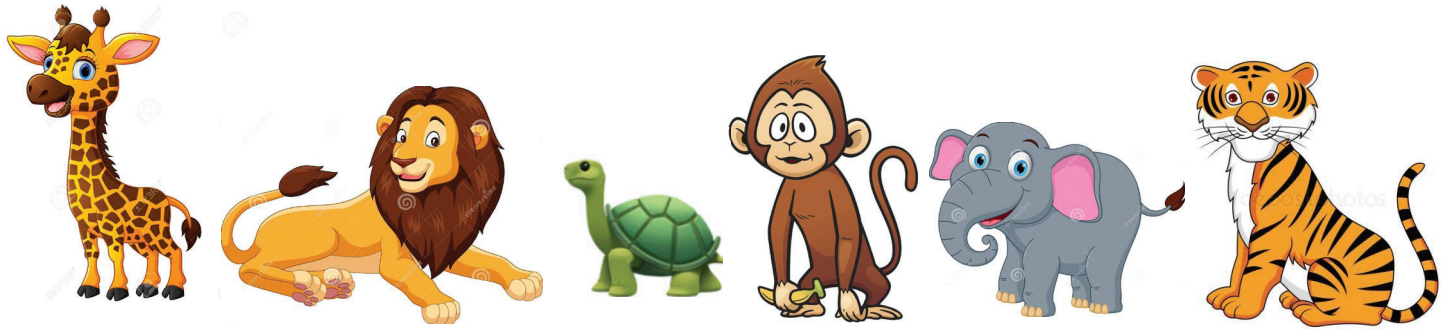
Goal: given input vector x and upper bound k on the allowed number of rounds of interaction, find the maximum element

- using as few comparisons as possible (count total)
- using at most k rounds of interaction with the oracle

Question: Given vector $x = (x_1, \ldots, x_n)$ and number $k \in \{1, \ldots, n\}$, how many comparisons do we need to find the maximum of x in k rounds?

Finding the maximum

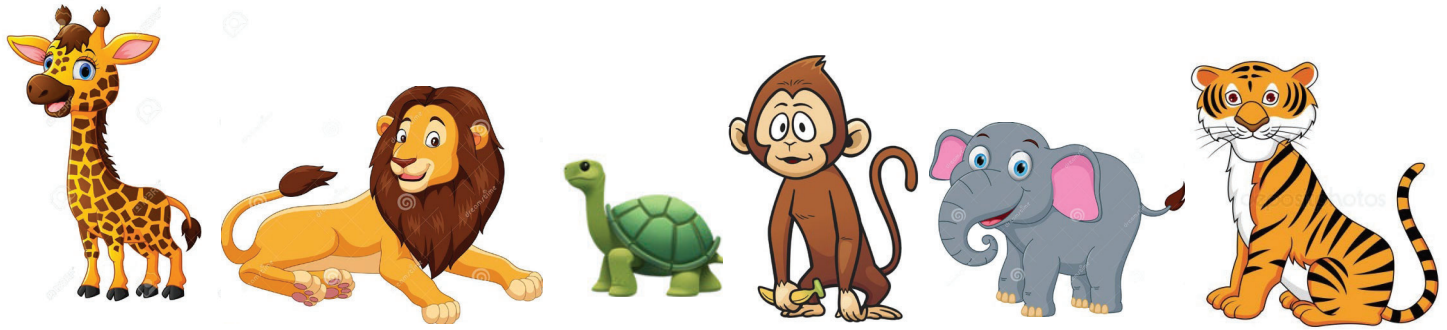
Input:



Unlimited number of rounds (fully adaptive)?

Finding the maximum

Input:



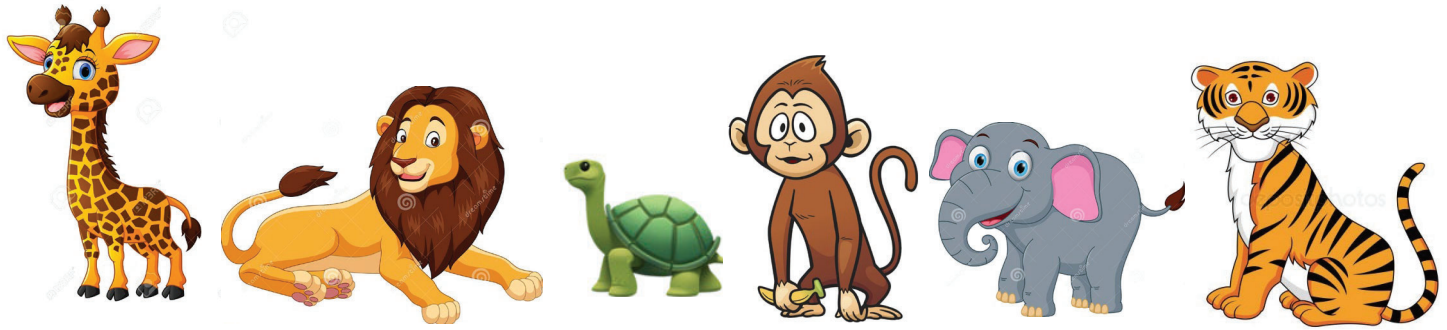
Unlimited number of rounds (fully adaptive):

- Upper bound:

- Lower bound:

Finding the maximum

Input:



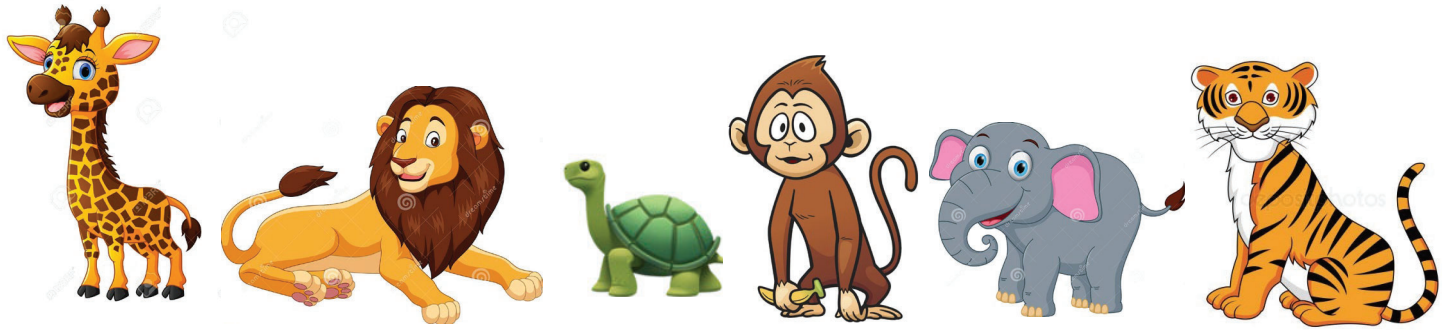
Unlimited number of rounds (fully adaptive):

- **Upper bound:** Go through each element and remember the max seen so far => n-1 comparisons.

- **Lower bound:**

Finding the maximum

Input:



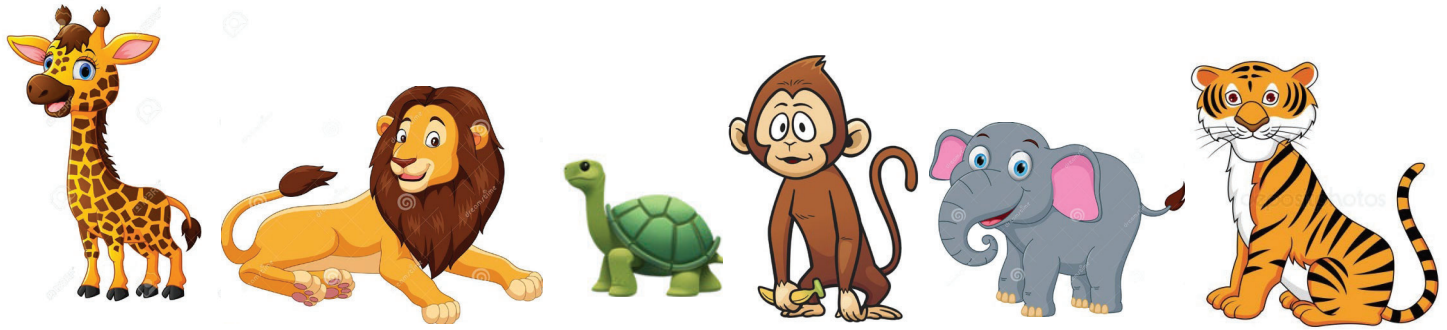
Unlimited number of rounds (fully adaptive):

- **Upper bound:** Go through each element and remember the max seen so far => n-1 comparisons.

- **Lower bound:** If an element is not compared to anything, then it could be the maximum. The comparisons give a graph on n elements that must be connected => need at least n-1 comparisons.

Finding the maximum

Input:



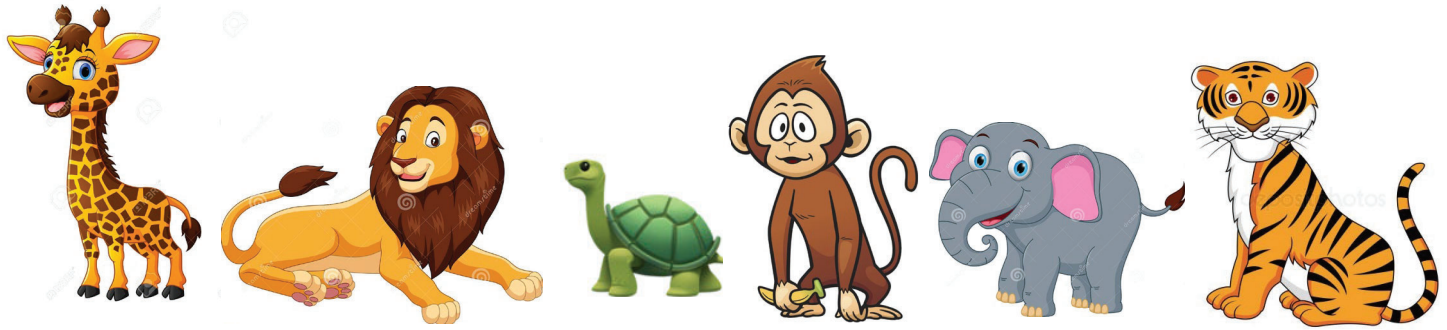
Unlimited number of rounds (fully adaptive): $\Theta(n)$

- **Upper bound:** Go through each element and remember the max seen so far => n-1 comparisons.

- **Lower bound:** If an element is not compared to anything, then it could be the maximum. The comparisons give a graph on n elements that must be connected => need at least n-1 comparisons.

Finding the maximum

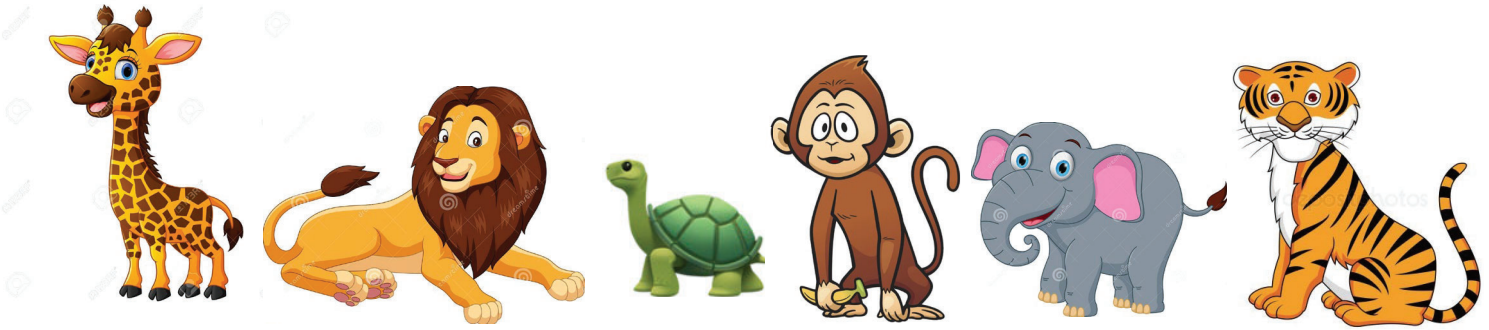
Input:



r = 1 rounds: How many comparisons?

Finding the maximum

Input:



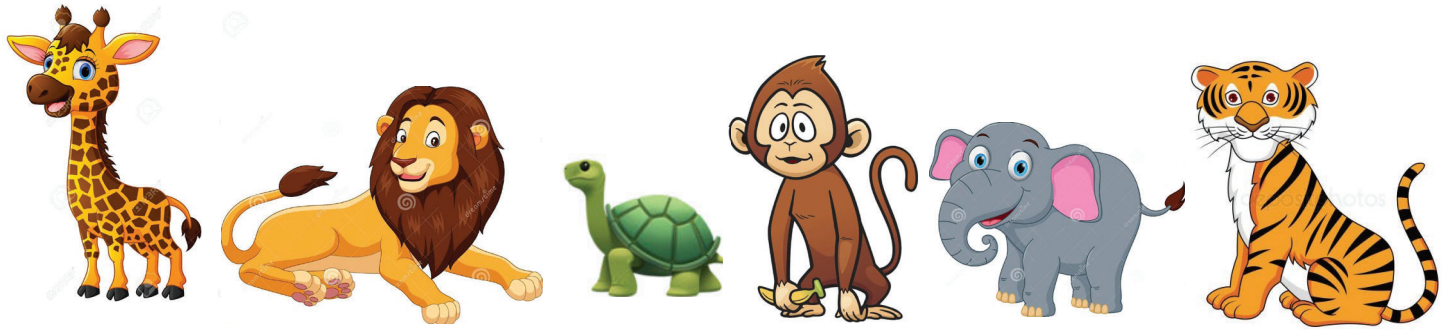
r = 1 rounds:

Upper bound:

Lower bound:

Finding the maximum

Input:



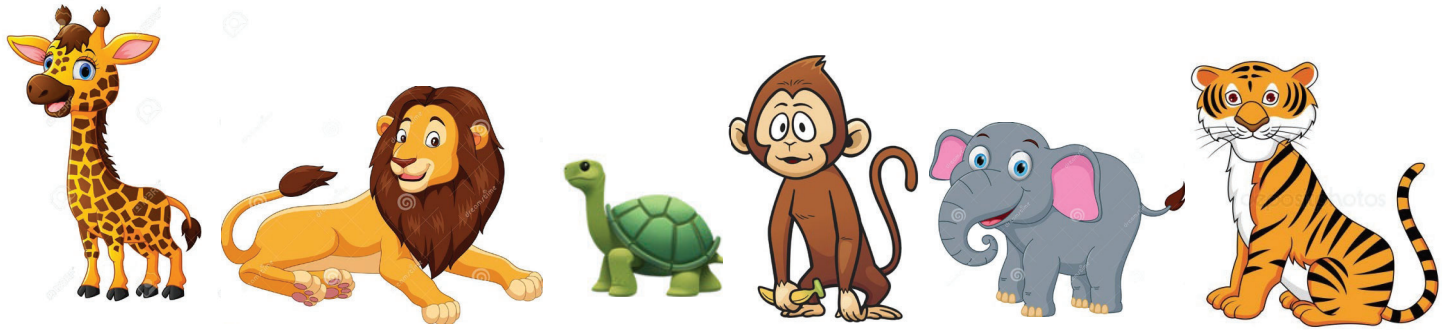
r = 1 rounds:

Upper bound: Ask for all pairwise comparisons $\binom{n}{2}$ and output the maximum after receiving the answers.

Lower bound:

Finding the maximum

Input:



r = 1 rounds:

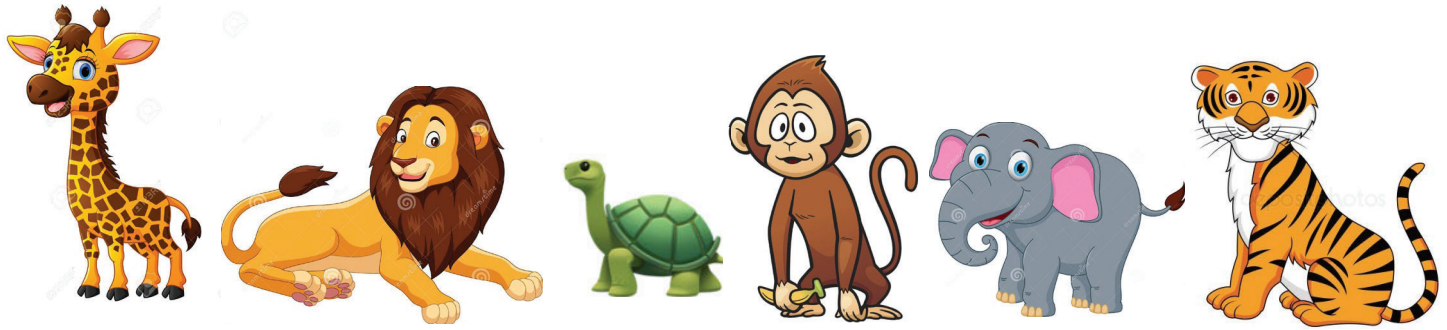
Upper bound: Ask for all pairwise comparisons $\binom{n}{2}$ and output the maximum after receiving the answers.

Lower bound: If even one comparison is missing, that can be enough to return the wrong maximum:

- Suppose there is a comparison missing between elements x_i and x_j in the input vector x.
- Then adversary can answer the queries so that x_i and x_j are greater than all the other elements -> not enough info to determine the max.

Finding the maximum

Input:



r = 1 rounds: $\Theta(n^2)$.

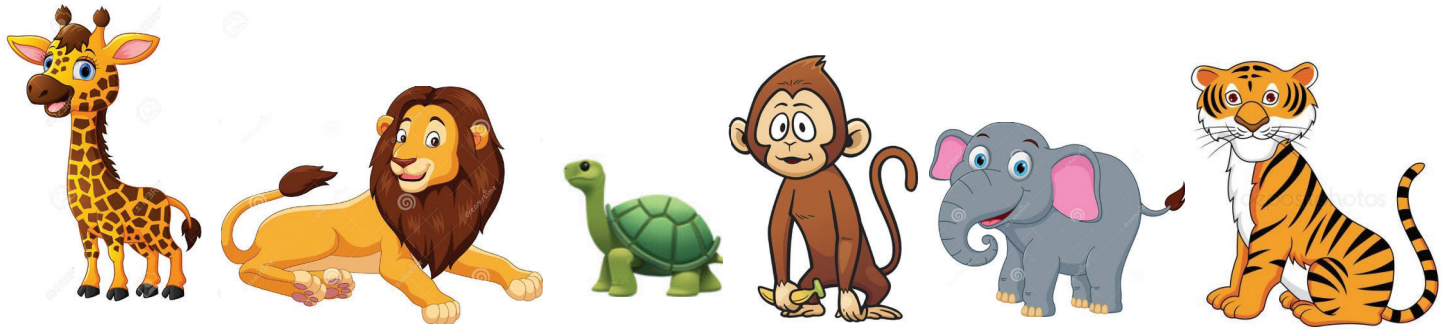
Upper bound: Ask for all pairwise comparisons $\binom{n}{2}$ and output the maximum after receiving the answers.

Lower bound: If even one comparison is missing, that can be enough to return the wrong maximum:

- Suppose there is a comparison missing between elements x_i and x_j in the input vector x.
- Then adversary can answer the queries so that x_i and x_j are greater than all the other elements -> not enough info to determine the max.

Finding the maximum

Input:



r = 2 rounds: How many comparisons?

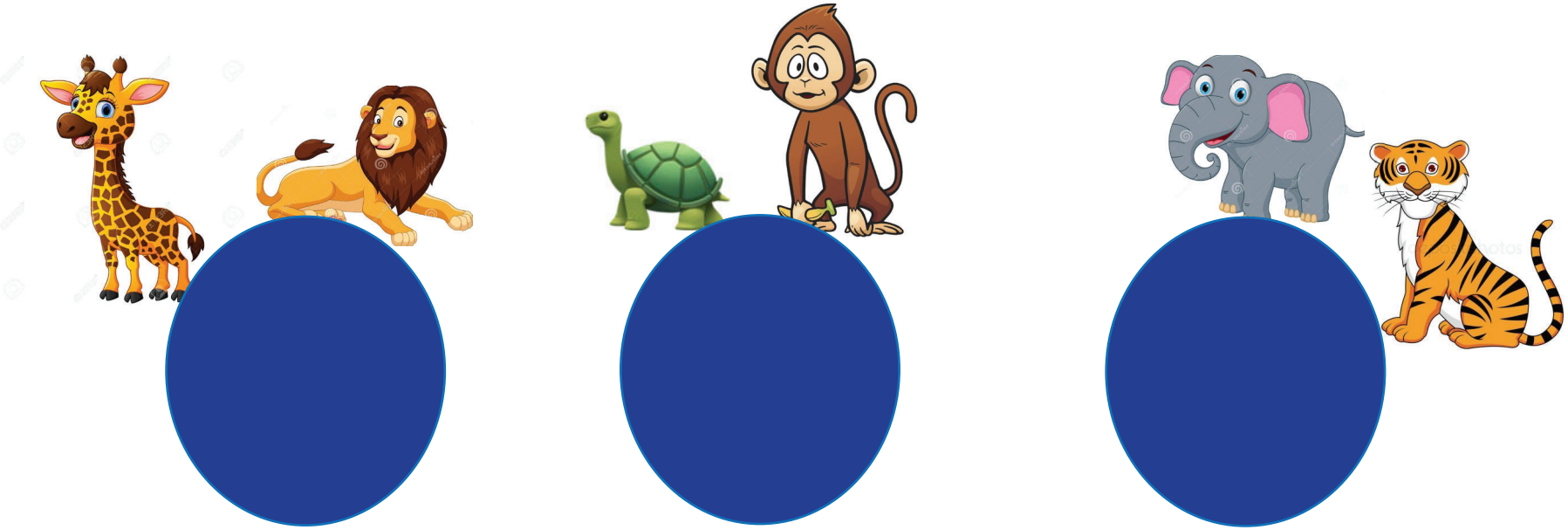
Finding the maximum

r = 2 rounds. ***Upper bound:***

Finding the maximum

r = 2 rounds. ***Upper bound:***

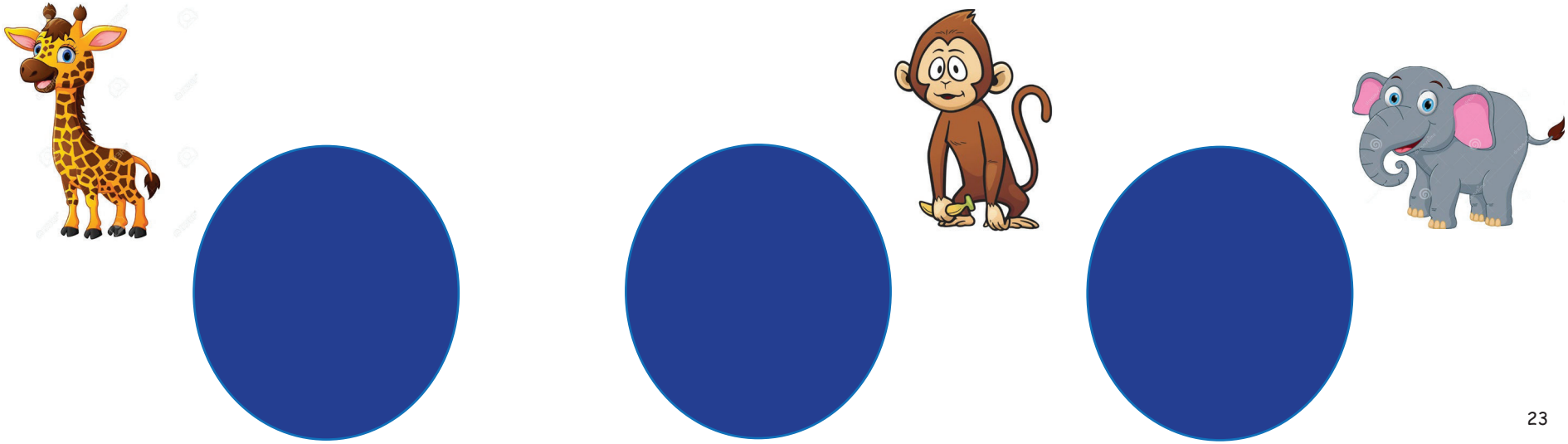
Round 1. Divide the items into k groups of size n/k and select the max from each group => $\binom{n/k}{2}$ comparisons inside each group => at most $k * \frac{n^2}{k^2}$ comparisons overall.



Finding the maximum

r = 2 rounds. ***Upper bound:***

Round 1. Divide the items into k groups of size n/k and select the max from each group => $\binom{n/k}{2}$ comparisons inside each group => at most $k * \frac{n^2}{k^2}$ comparisons overall.

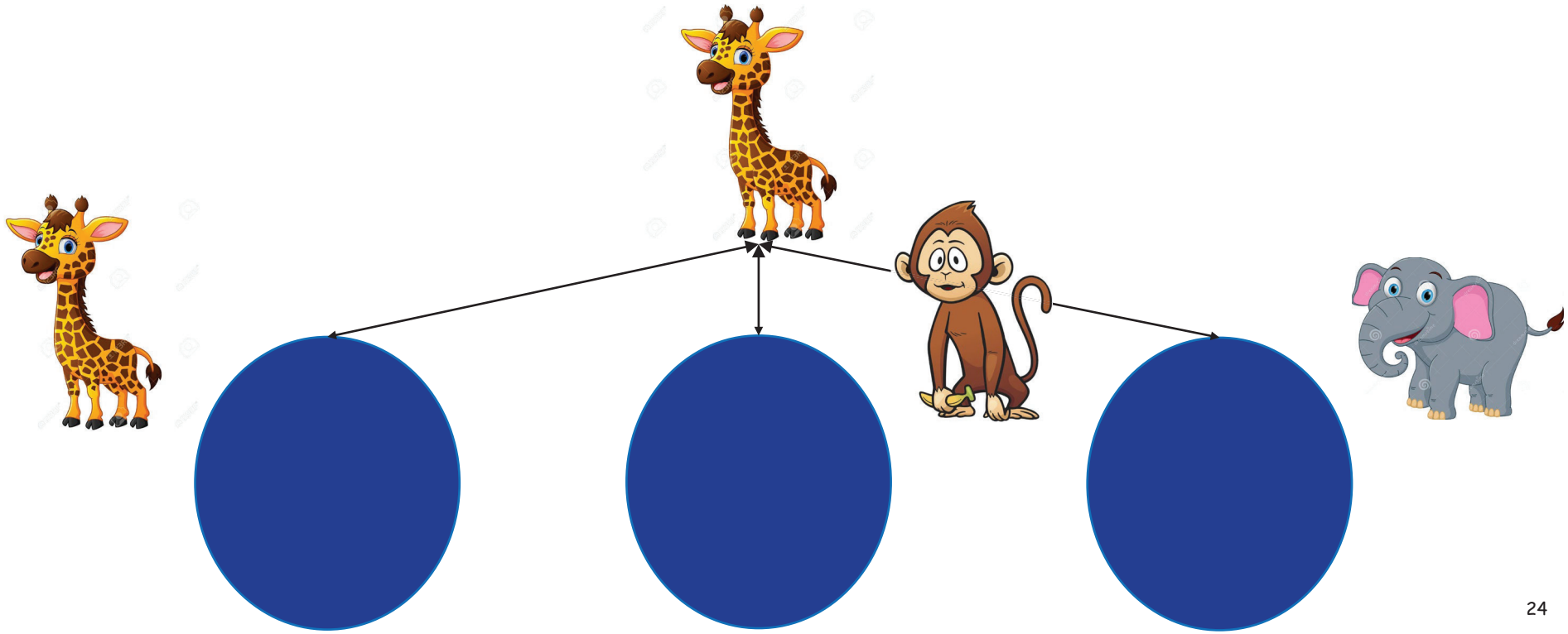


Finding the maximum

r = 2 rounds. *Upper bound:*

Round 1. Divide the items into k groups of size n/k and select the max from each group => $\binom{n/k}{2}$ comparisons inside each group => at most $k * \frac{n^2}{k^2}$ comparisons overall.

Round 2. Select the maximum among the group maxima => round 1 protocol for finding the max of k elements => k^2 comparisons



Finding the maximum

r = 2 rounds. *Upper bound:*

Round 1. Divide the items into k groups of size n/k and select the max from each group => $\binom{n/k}{2}$ comparisons inside each group => at most $k * \frac{n^2}{k^2}$ comparisons overall.

Round 2. Select the maximum among the group maxima => round 1 protocol for finding the max of k elements => k^2 comparisons.

The total number of comparisons is at most $k * \frac{n^2}{k^2} + k^2$.

Set k to equalize the work for rounds 1 and 2:

$$k * \frac{n^2}{k^2} = \mathrm{k}^2 \Rightarrow \mathrm{k} = \mathrm{n}^{2/3}.$$

Total number of comparisons is $2 \cdot k^2 = 2 \cdot n^{4/3}$.

Finding the maximum

r = 2 rounds. ***Upper bound:*** $O(n^{\frac{4}{3}})$

Round 1. Divide the items into k groups of size n/k and select the max from each group => $\binom{n/k}{2}$ comparisons inside each group => at most $k * \frac{n^2}{k^2}$ comparisons overall.

Round 2. Select the maximum among the group maxima => round 1 protocol for finding the max of k elements => k^2 comparisons.

The total number of comparisons is at most $k * \frac{n^2}{k^2} + k^2$.

Set k to equalize the work for rounds 1 and 2:

$$k * \frac{n^2}{k^2} = k^2 \Rightarrow k = n^{2/3}.$$

Total number of comparisons is $2 \cdot k^2 = 2 \cdot n^{4/3}$.

Finding the maximum

r = 2 rounds. ***Lower bound:*** $\Omega(n^{\frac{4}{3}})$