## The Josephus Game

Given $n$ player sitting in a circle, and a number $m$.

A hot potato starts at player $1$, and is passed around $m$ times. The player holding the potato then is eliminated, the next player gets the potato, and the game continues until only one player is left.

$n = 6,\ m = 2$

Ⓐ



## Can we do it in less than quadratic time?

Our linked-list Josephus program needs $(n-1)m$ link transversals.
Can we do it more efficiently?

First observation: If we are currently at position $p$, then after $m$ passes we will be at position $(p+m) \bmod n$ (positions numbered from $0$ to $n-1$).

Difficulty: How can we maintain the names of the people remaining in the game?

We need a data structure that stores a sequence of $n$ elements, and supports one main operation: Remove the $k$th element.

No standard Python data structure supports this operation efficiently. We need to implement it ourselves...



## The Rank Tree

Supports the following operations:

- Construct from an array with $n$ elements;
- `find(k)` returns the item at rank (index) $k$;
- `remove(k)` removes the element at rank $k$;
- `size()` returns the current size.

Idea: Store the elements in a binary tree in in-order sequence. Store in each node $t$ the size of the subtree whose root is $t$.

To find the node with rank $k$, we just have to follow a path from the root.



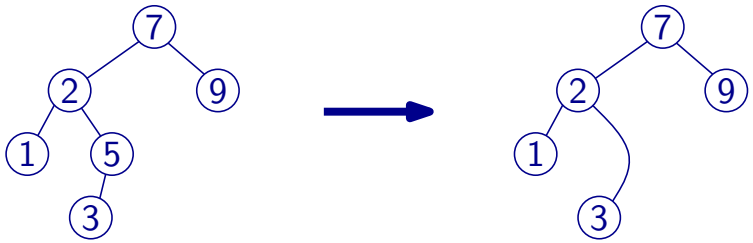
## The remove operation

How to remove the node $t$ with rank $k$?

First find the node $t$ of rank $k$.
Then there are three cases:

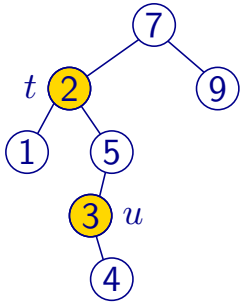
1. Easy case: $t$ is a leaf node.

2. Slightly harder case: $t$ has one child



## The remove operation

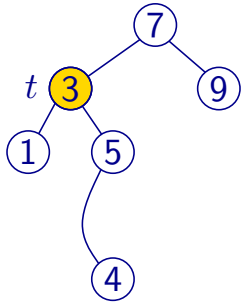
3. If $t$ has two children, then find the leftmost node $u$ in the right subtree of $t$. Replace the element stored at $t$ with the element from $u$. Finally, remove the node $u$.




KAIST CS206

## The remove operation

3. If $t$ has two children, then find the leftmost node $u$ in the right subtree of $t$. Replace the element stored at $t$ with the element from $u$. Finally, remove the node $u$.




KAIST CS206

## Rank tree analysis

`find` and `remove` take time $O(h)$, where $h$ is the height of the tree.

When we construct the tree, we can make a perfectly balanced tree.
Its height is $\lceil \log(n+1) \rceil - 1$.

Therefore `find` and `remove` take time $O(\log n)$, and the total running time for the Josephus problem is $O(n \log n)$.