### Algorithm: Quickselect

Princess Elara needed to find the strongest sea monster in the Kraken's army. To do so, she used the Quickselect algorithm to find the k-th strongest monster quickly.

#### Initialize Data Structures:

* Princess Elara used a trident (list) to hold the strengths of the sea monsters.

#### Partition and Select:

* She divided the sea monsters into those weaker and stronger than a pivot.
* She recursively focused on the part of the list that contained the k-th strongest monster.

#### Implementation:

| **def** quickselect(monsters: List[int], k: int) -> int:  **def** partition(low, high):  pivot = monsters[high]  i = low  **for** j **in** range(low, high):  **if** monsters[j] <= pivot:  monsters[i], monsters[j] = monsters[j], monsters[i]  i += 1  monsters[i], monsters[high] = monsters[high], monsters[i]  **return** i  low, high = 0, len(monsters) - 1  **while** low <= high:  pivot\_index = partition(low, high)  **if** pivot\_index == k:  **return** monsters[pivot\_index]  **elif** pivot\_index < k:  low = pivot\_index + 1  **else**:  high = pivot\_index - 1  *# Example usage:*  monsters = [5, 3, 8, 4, 2, 7, 1, 9, 6]  k = 4  print(quickselect(monsters, k)) *# Output: 5 (5th strongest monster)* |
| --- |

#### Explanation:

Initialize:

* monsters: A list of sea monster strengths.

Partition and Select:

* Princess Elara divided the monsters into those weaker and stronger than a pivot.

### She recursively focused on the part of the list containing the k-th strongest monster.