`fn top`

Michael Shoemate

April 29, 2025

1 Hoare Triple

Precondition

Compiler-verified

Types consistent with pseudocode.

Caller-verified

- `iter` must be finite
- `greater_than` must form a total order

Pseudocode

```
def top(
    iter: Iterator[T],
    k: usize,
    greater_than: Callable[[T, T], bool],
) -> list[T]:
    heap = []  #

    if k == 0: #
        return heap

    for value in iter:  #
        if len(heap) == k: #
            if greater_than(heap[-1], value): #
                continue
            heap.pop() #

        # insert value into heap #
        index = partition_point_mut(heap, lambda x: greater_than(x, value))
        heap.insert(index, value)
    return heap
```

Postcondition

Theorem 1.1. Returns the top k elements from the iterator in sorted order.

Proof. Line 6 initializes empty storage for the top k elements.

First, consider the case where k is zero. By line 8, the function returns the heap, satisfying the postcondition.

Now consider the case where k is greater than zero. The heap has two invariants:

- It has at most k elements

- Elements are stored in descending order according to `greater_than`

First, the heap will never have more than k elements. If there are k elements in the heap, then either no element is added on line 13, or the smallest element is removed on line 15. Therefore at line 17, the heap will always have less than k elements. Since only one element can be added to the heap at a time, the heap will never have more than k elements, satisfying the first invariant.

Next, the heap will always be sorted in descending order. By the precondition that `greater_than` forms a total order, then `heap` is partitioned by the predicate function. Also, by the precondition on mutability of elements in `iter`, the precondition for `partition_point_mut` is satisfied. Since the preconditions for `partition_point_mut` are met, the postcondition provides the index of the first element in the heap that is less than `value`. Therefore, upon insertion on line 17, the heap will remain sorted in descending order.

Since elements smaller than the smallest element in the heap are not added, and the heap is sorted in descending order, the heap will always contain the top k elements in sorted order. □