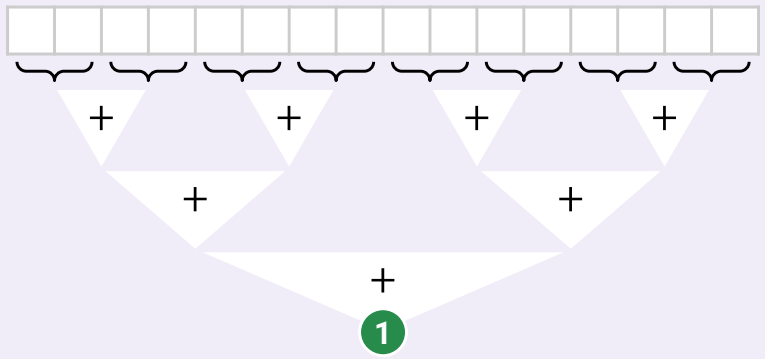
Q SumTask Call Tracing



Q1: Trace the execution of SumTask if we call: left.fork(); right.compute(); left.join();

Q2: Trace the execution of SumTask if we call: left.fork(); left.join(); right.compute();

Q What's missing from this parallel MaxTask class?

```
class MaxTask {
    int lo; int hi; int[] a;
    MaxTask(int[] arr, int l, int h) { lo = l; hi = h; a = arr; }
    public Integer compute() {
        int mid = (hi + lo) / 2;
        MaxTask left  = new MaxTask(a, lo, mid);
        MaxTask right = new MaxTask(a, mid, hi);
        return Math.max(left.compute(), right.compute());
    }
}
```


Q1: What's missing from this parallel MaxTask class?

Q RecursiveTask<T> vs. RecursiveAction

Sometimes, it's not necessary to return a result in compute().

The fork/join framework **RecursiveAction** has void return type.

In what scenario(s) would we want to use RecursiveAction?

How do we keep track of information if it's not passed via return type?

Q1: In what scenario(s) would we want to use RecursiveAction?

Q2: How do we keep track of information if it's not passed via return type?

Q Parallel Map

Parallelize vector addition. Describe your algorithm in English first before writing code.

```
def add(a1, a2, result):
    assert len(a1) == len(a2) == len(result)
    for i in range(len(a1)):
        result[i] = a1[i] + a2[i]
```


Q1: Parallelize vector addition. Describe your algorithm in English first before writing code.