**Recursion trees:** Use recursion trees to find the closed form for the following. Ensure that the input is always an integer by assuming $n$ is a power of 3.

$T(1) = 47$

$T(n) = 3T(n/3) + 13n$ for $n \geq 2$

**Mystery Code:**

The procedure **crunch** takes an array of integers $a_1, a_2, \ldots, a_n$ (where $n \geq 1$) and returns an integer. Assume that dividing the array in two requires only constant time.

```
crunch(a[1], ..., a[n] : array of integers)
  if (n = 1 and a[1] ≤ 0) return 1
  else if (n = 1) return 0
  else
      m = ⌊n/2⌋
      output = crunch(a[1], ..., a[m]) + crunch(a[m + 1], ..., a[n])
      return output
```

- Give a succinct English description of what **crunch** computes.
- Suppose that $T(n)$ is the running time of **crunch** on an input array of length $n$. Write down a recurrence for $T(n)$, including a base case. For simplicity, you may assume that $n$ is a power of 2.
- What is the big-$\Theta$ running time of **crunch**? Justify your answer either by unrolling the recurrence or by drawing/describing a recursion tree.

**Mystery Code II:**

The procedure **maxthree** is calling itself on a version of the input list with the $k$th element ($a_k$) removed. Assume it takes constant time to temporarily remove $a_k$ from the list. (Doing this in constant time actually requires some extra details that we are hiding for clarity.)

```
maxthree(a1, ..., a_n) : list of n positive integers, n ≥ 3)
  if (n = 3) return a_1 + a_2 + a_3
  else
      bestval = 0
      for k = 1 to n
          newval = maxthree(a_1, a_2, ..., a_{k-1}, a_{k+1}, ..., a_n)
          if (newval ≥ bestval) bestval = newval
      return bestval
```

- Describe (in English) what **maxthree** computes.
- Suppose that $T(n)$ is the running time of **maxthree** on an input array of length $n$. Give a recursive definition of $T(n)$.
- Does **maxthree** run in $O(2^n)$ time? Briefly explain why or why not.