Example: Merge Sort

Idea:

• To sort a list, divide it in half, sort the two halves independently, and then merge the two sorted lists into a single sorted list.
• For this presentation, we use linked lists.

Sort a list:

procedure MSORT(list):
    if list is empty or has only one element then return list
    else begin
        Divide list into two equal size lists, list1 and list2.
        return MERGE(MSORT(list1), MSORT(list2))
    end
end
(merge sort continued)

Merge two sorted lists in linear time and space:

```plaintext
function MERGE(list1, list2)
    list3 := the empty list
    while both list1 and list2 are not empty do begin
        Compare the first elements of list1 and list2 and whichever is smaller (ties can broken arbitrarily), delete it and append it to list3.
    end
    append list1 and list2 to the end of list3
    return list3
end
```
(merge sort continued)

**Time to sort a list of n elements:**

- If $T(n)$ denotes the time to sort $n$ elements, then if $n \leq 1$, $T(n) \leq a$ for some constant $a$.
- For $n>1$, two recursive calls take time $2T([n/2])$, and for some constant $b$, $bn$ time is used for the dividing and merging.
- If $N$ is smallest power of 2 $\geq n$, and $c$ the maximum of $a$ and $b$, then:

$$T(n) \leq 2T\left(\frac{N}{2}\right) + cN$$

$$= 4T\left(\frac{N}{4}\right) + 2c \frac{N}{2} + cN$$

$$= 8T\left(\frac{N}{8}\right) + 4c \frac{N}{4} + 2c \frac{N}{2} + cN$$

$$= cN + cN + \cdots + cN \quad - \quad \log_2(N) + 1 \text{ terms}$$

$$= cN(\log_2(N) + 1)$$

$$= O(n \log(n))$$