Algorithmic Complexity

Introduction to Computer Programming

Dr. Paul Vrbik

November 21, 2018

Definition (Complexity)

We measure the complexity of an algorithm by counting the number of operations performed as a function of the length of the input (usually denoted n).

Linear

```
def linear(L: List[int]):
    n = len(L)                  # 1
    index = 0                   # 1
    while index < len(L):       # n
        index += 1              # n
                                # 2n + 1 = O(n)
```

Definition (Big-Oh Notation)

We say that "$f(n)$ is big-oh of $g(n)$" and write $f(n) = \mathcal{O}(g(n))$ when

$$\exists\, c,\, k \in \mathbb{R}^{>0} : \forall n \geq k;\, 0 \leq f(n) \leq cg(n).$$

Example

1. $2x^2 + x + 1 = \mathcal{O}(n^2)$ because for $x > 2$
$$2x^2 + x + 1 < 3x^2.$$

2. $x^3 - \sin(x) = \mathcal{O}(n^3)$ because for $x > 1$
$$x^3 - \sin(x) \leq x^3 + 1 < 2x^3.$$

Question

What is the complexity of `foo`?

```
def foo(L: List[int]):
    n = len(L)
    index = 0
    while index < len(L):
        index2 = 0
        while index2 < len(L):
            index2 += 1
        index += 1
```

Quadratic

```
def quadratic(L: List[int]):
    n = len(L)
    index = 0                          # 1
    while index < len(L):              # n
        index2 = 0                     # n
        while index2 < len(L):         # n * n
            index2 += 1                # n * n
        index += 1                     # n
                                       # 2n**2 + 3n + 1 = O(n**2)
```

Question

What is the complexity of `foo`?

```
def foo(L: List[int]):
    n = len(L)
    index = 0
    while i < 10**10:
        index += 1
```

Constant

```
def foo(L: List[int]):
    n = len(L)              # 1
    index = 0               # 1
    while i < 10**10:       # 10**10
        index += 1          # 10**10
                            # 2*10**10 + 2 = O(1)
```

Question

What is the complexity of `foo`?

```
def foo(L: List[int]):
    n = len(L)
    index = 0
    while 2 ** index < len(L):
        index += 1
```

Logarithmic

```
def log(L: List[int]):
    n = len(L)                          # 1
    index = 0                           # 1
    while 2 ** index < len(L):          # log[2](n)
        index += 1                      # log[2](n)
                                        # 2log[2](n) + 2 = O(ln(n))
```

Question

What is the algorithmic of the following algorithms as measured by the length of the input list?

1. Selection,
2. Insertion, and
3. Bubble.

Answer

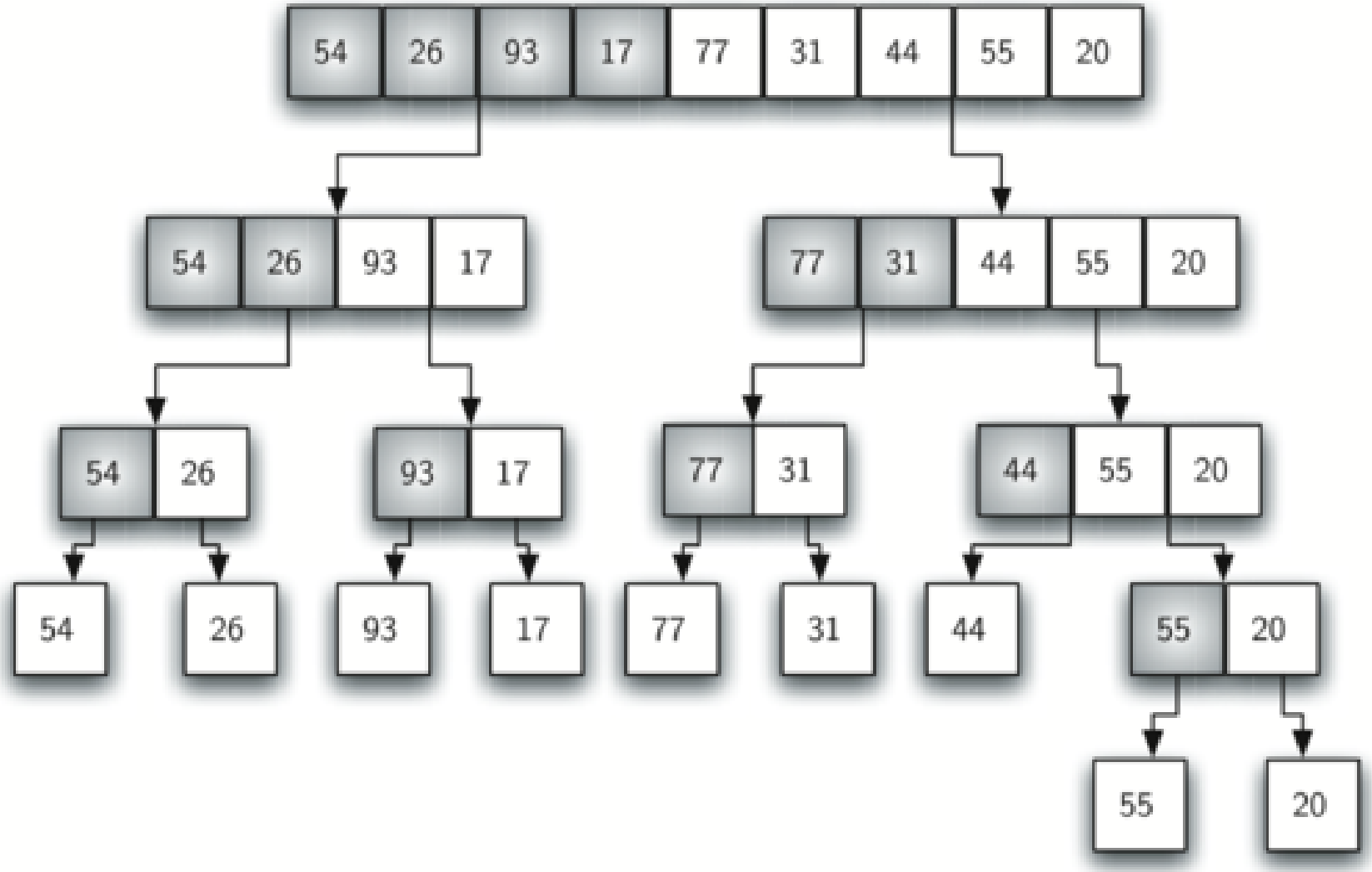
$\mathcal{O}(n^2)$.

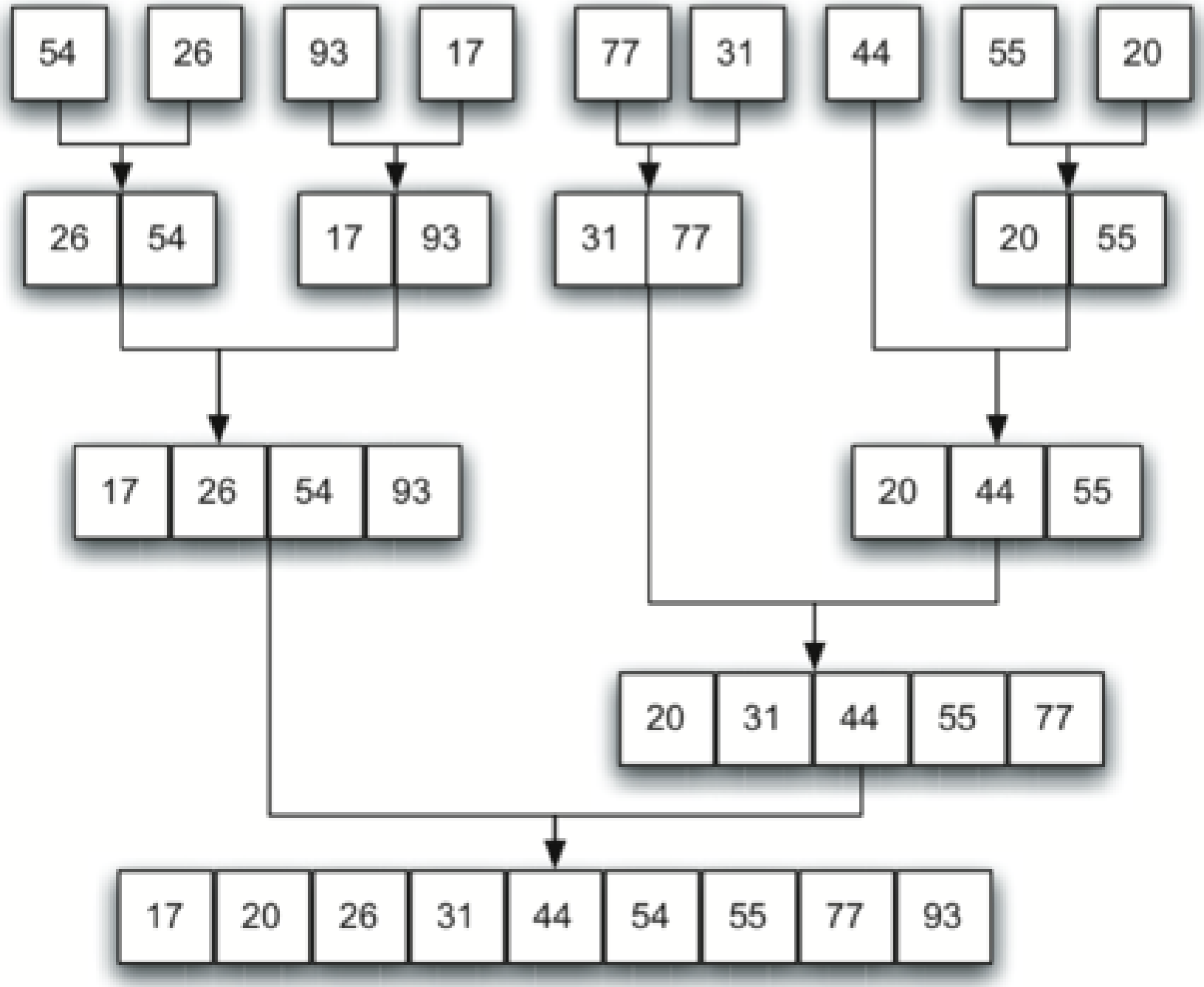
Question

Suppose the only operation we could perform on lists was merging two sorted lists into one sorted list. Could use this to sort a list?

Merge Sort

Break the list in half and sort the halves (using merge sort) then combine these pieces into a single sorted list being the answer.


54 26 93 17 77 31 44 55 20
54 26 93 17
77 31 44 55 20
54 26
93 17
77 31
44 55 20
54
26
93
17
77
31
44
55 20
55
20


54
26
93
17
77
31
44
55
20
26
54
17
93
31
77
20
55
17
26
54
93
20
44
55
20
31
44
55
77
17
20
26
31
44
54
55
77
93