Divide-and-conquer algorithms

CS 146 - Spring 2017

Today

- Review: mergesort
- Anatomy of a divide-and-conquer algorithm
- Example: maximum subarray
- Example: Karatsuba's multiplication

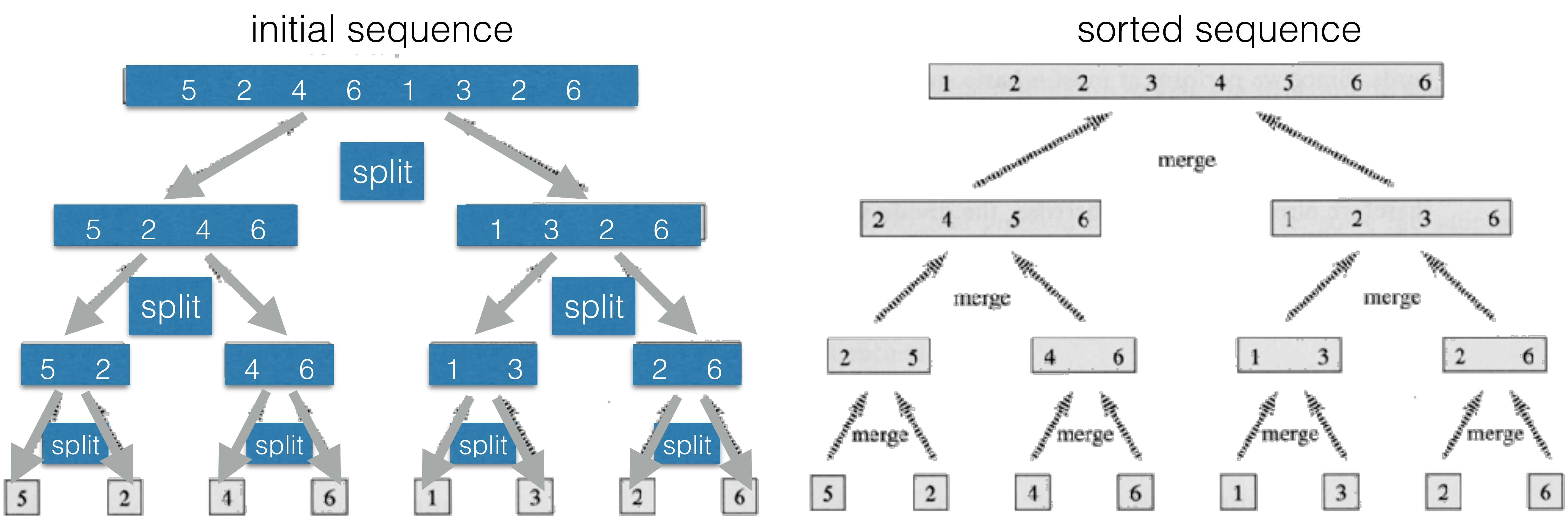
Question

what is the maximum sum of values
among all **contiguous subsequences**
of this sequence ?

13, -3, -25, 20, -3, -16, -23, 18, 20, -7, 12, -5, -22, 15, -4, 7

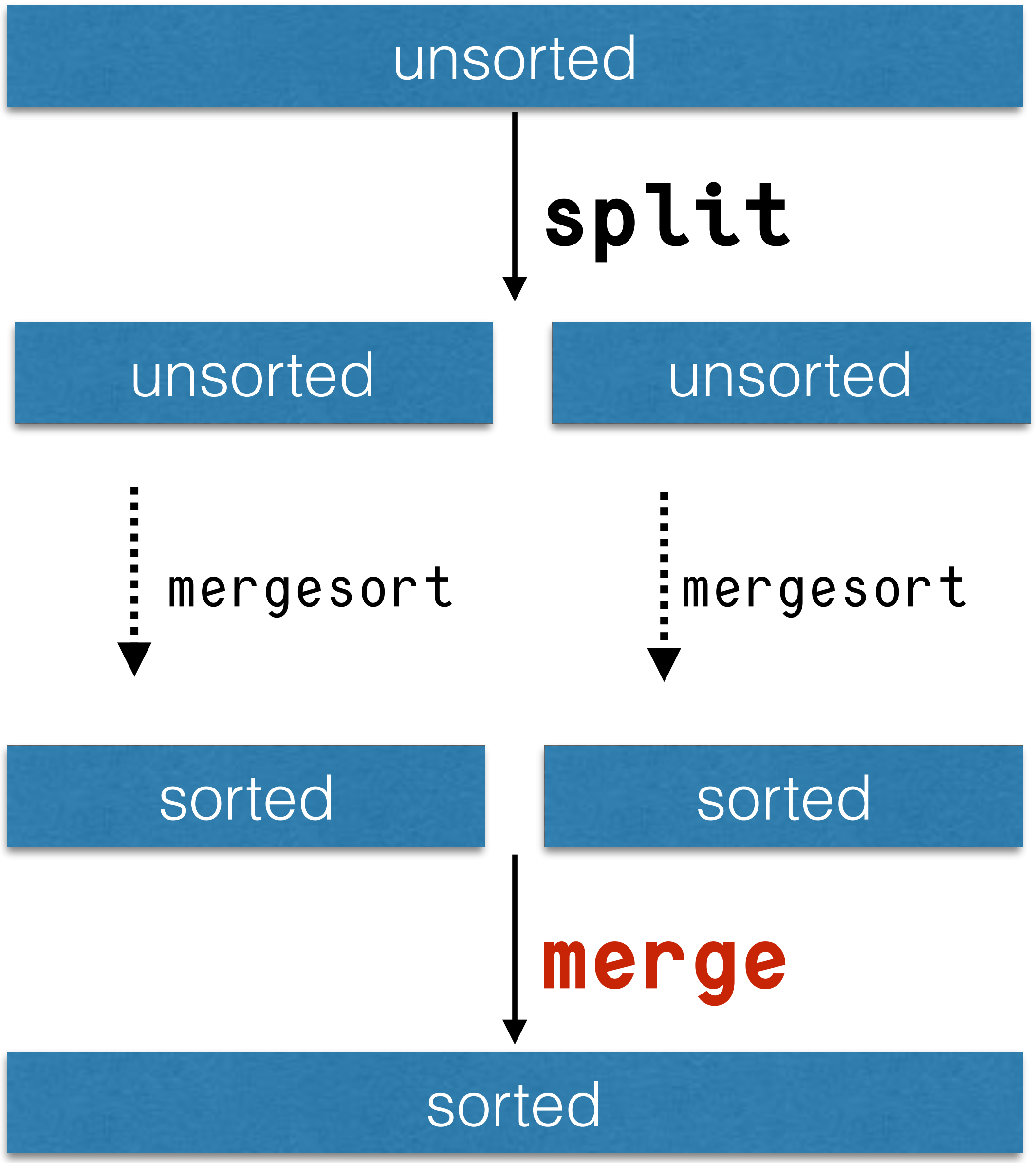
give an algorithm for
the maximum **subarray**
problem

Recall: mergesort

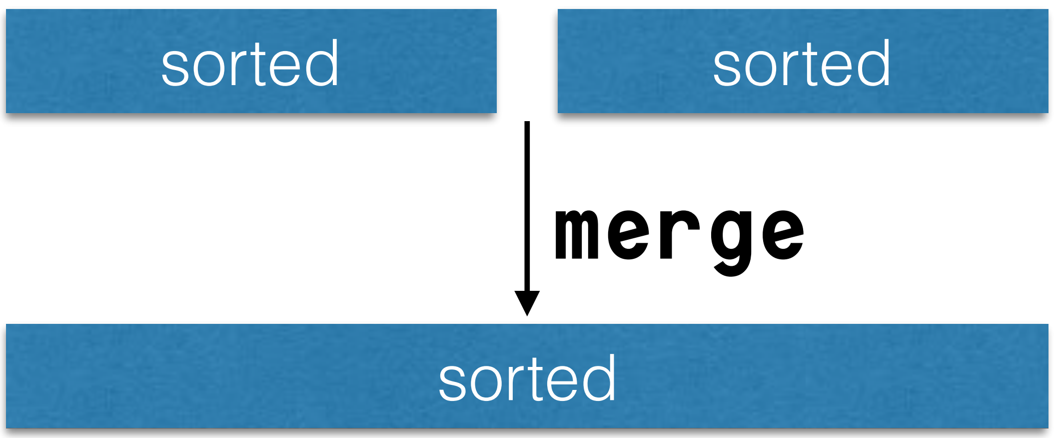


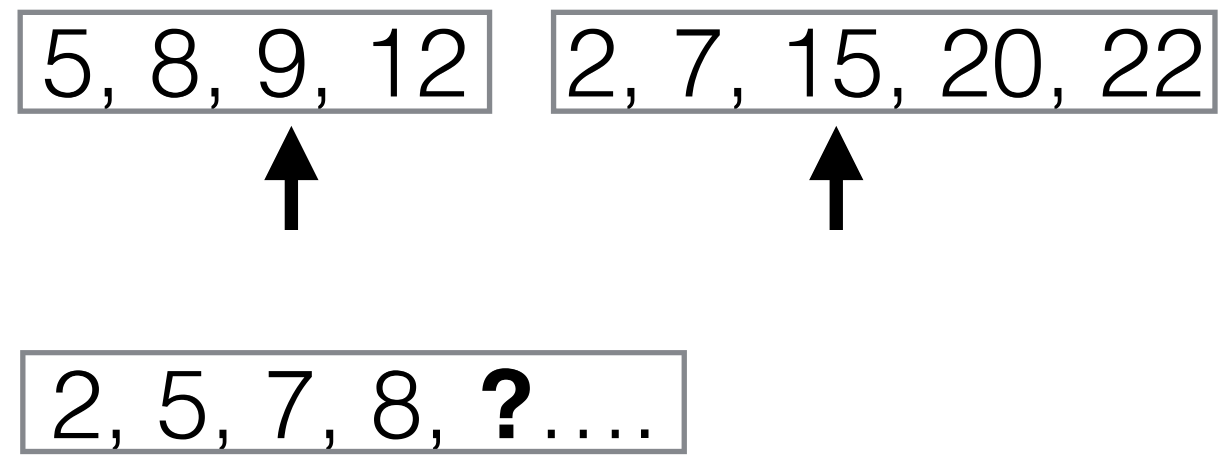
http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Sorting/mergeSort.htm

```
void mergesort(list) {

    if (length(list) <= 1) return;

    split list into left and
.                     right sublists

    mergesort(left);

    mergesort(right);

    merge left and right;

}
```



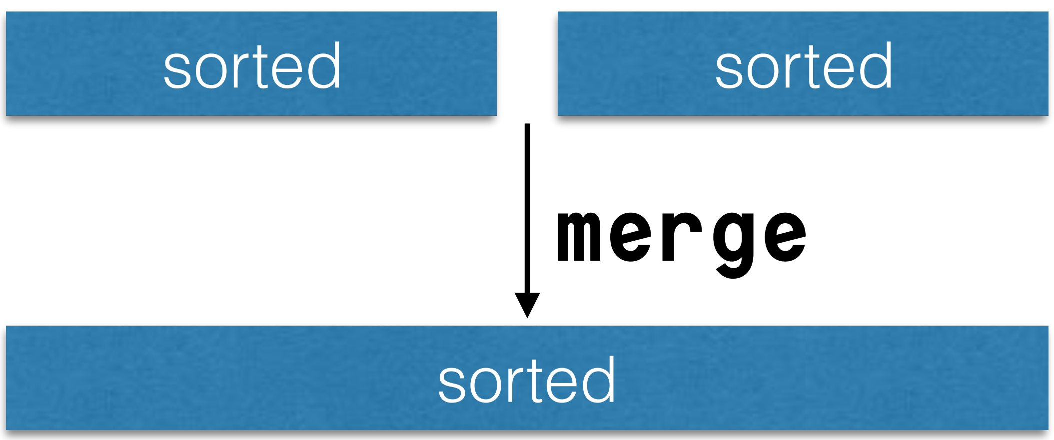
Recall: merge step

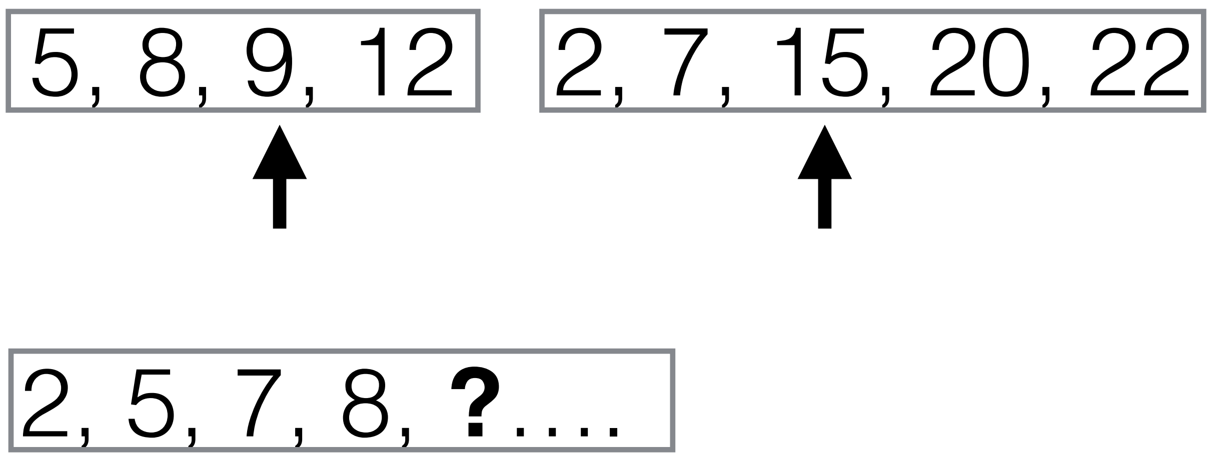





```
merge(a, b) {
    initialize empty result list,    set pointers to beginning of the 2 lists
    while there are 2 elements left to compare
        pick the smaller of the 2 being pointed at
        append it to the result list,   increment its pointer
    append rest of remaining input list to result and return that
}
```

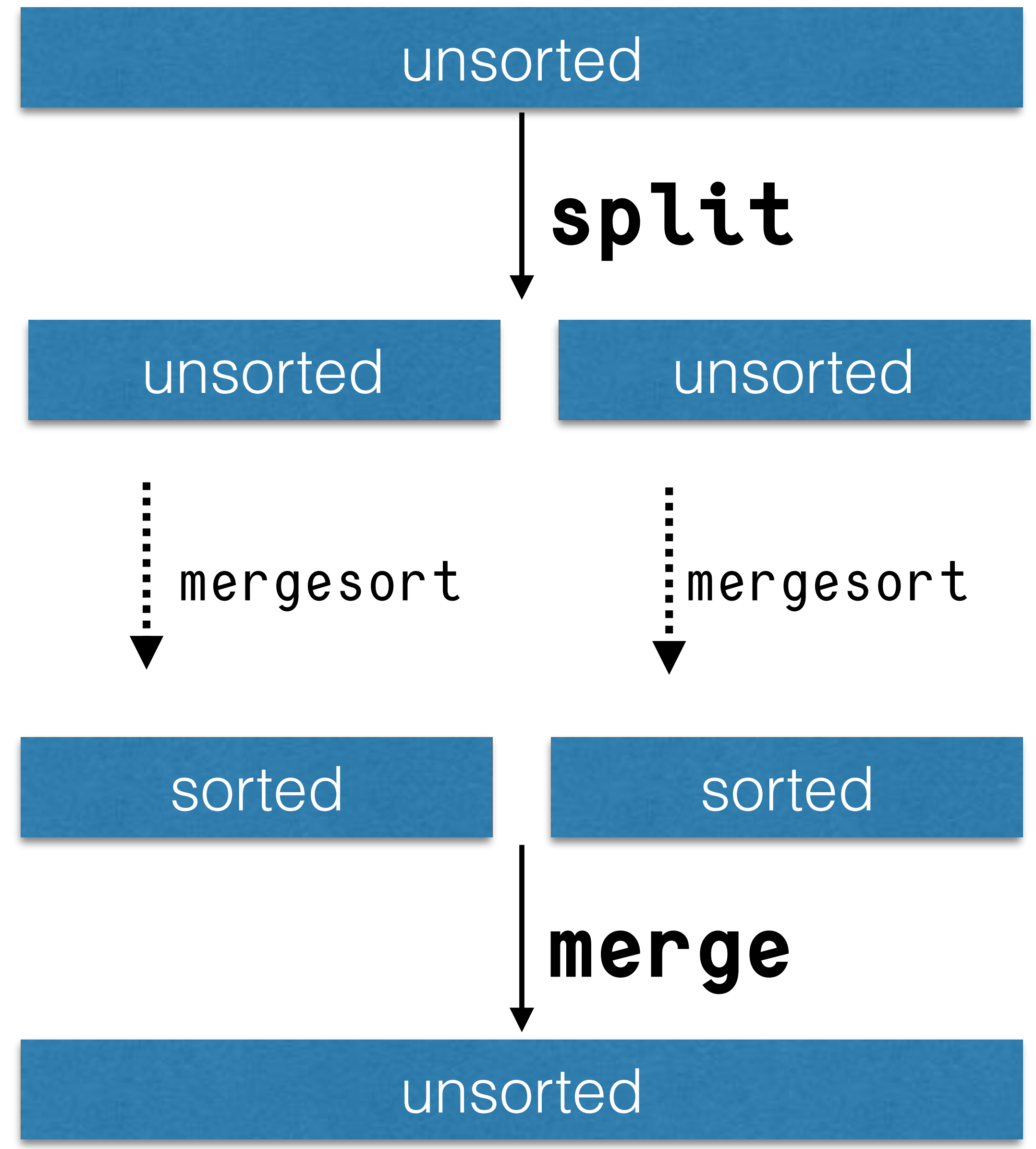
#comparisons in merge?





How do you analyze this??

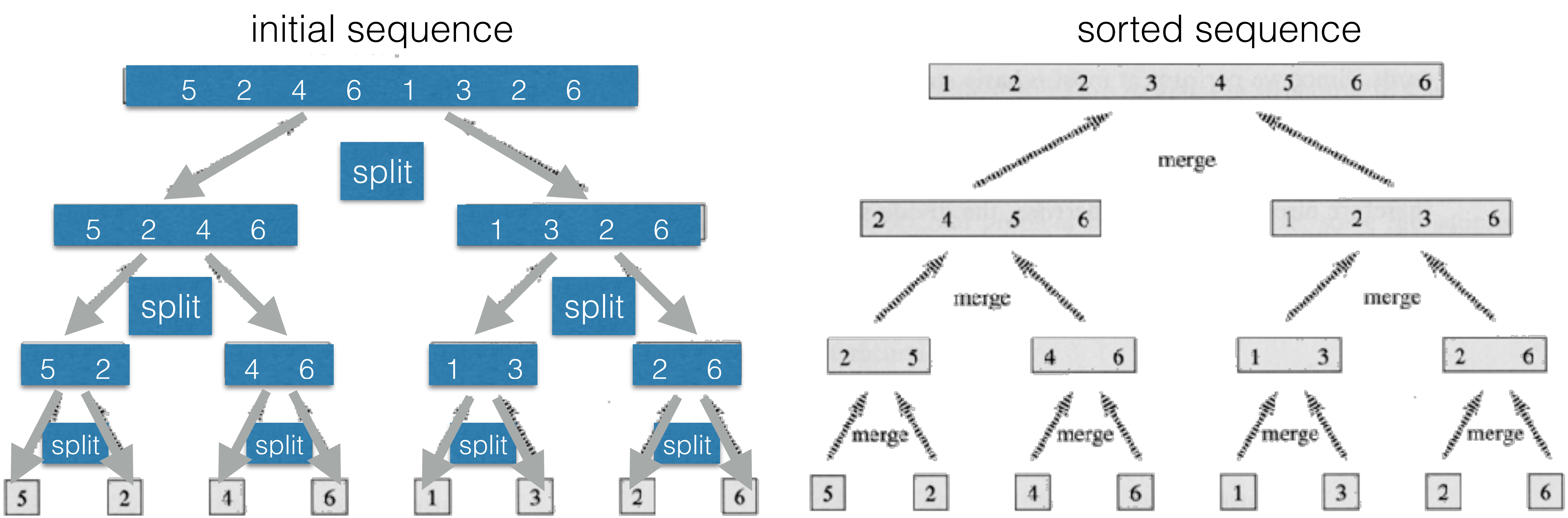
```
void mergesort(list) {

    if (length(list) <= 1) return;

    split list into left and
.                      right sublists

    mergesort(left);

    mergesort(right);

    merge left and right;

}
```



Recall the main idea: sum up work done in initial call AND all subsequent calls

Recall the main idea: sum up work done in initial call AND all subsequent calls

How do you analyze this??

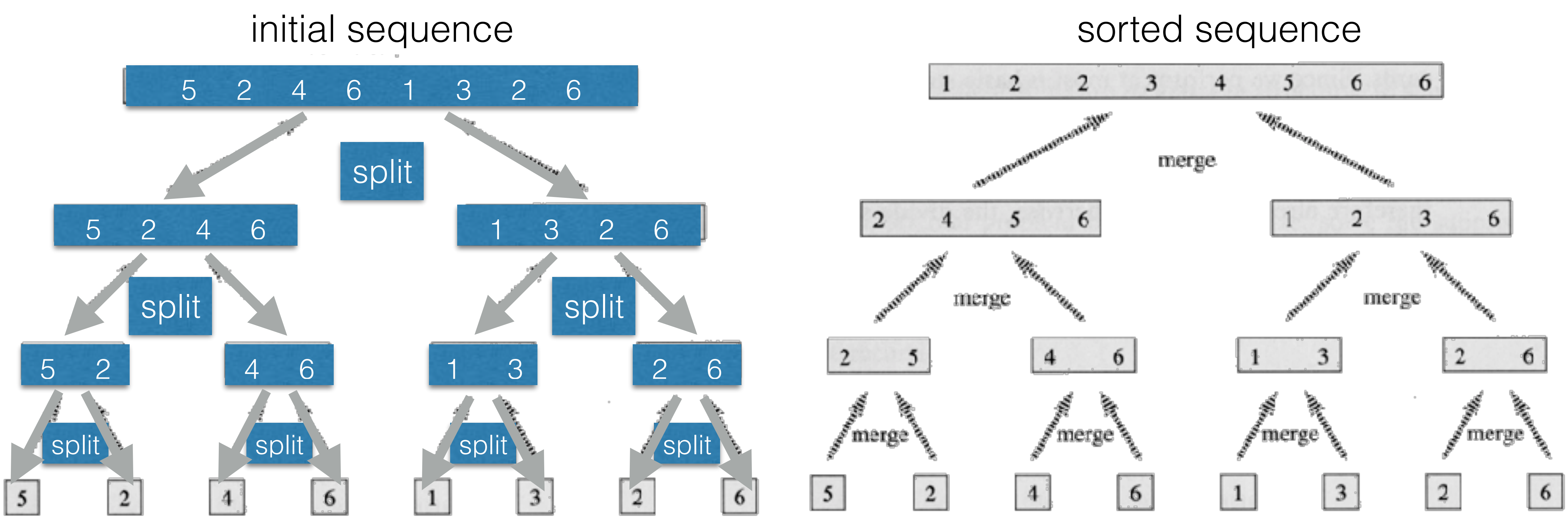


Obs 1: each call does 1 split and 1 merge and merge dominates

http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Sorting/mergeSort.htm

Recall the main idea: sum up work done in initial call AND all subsequent calls

How do you analyze this??

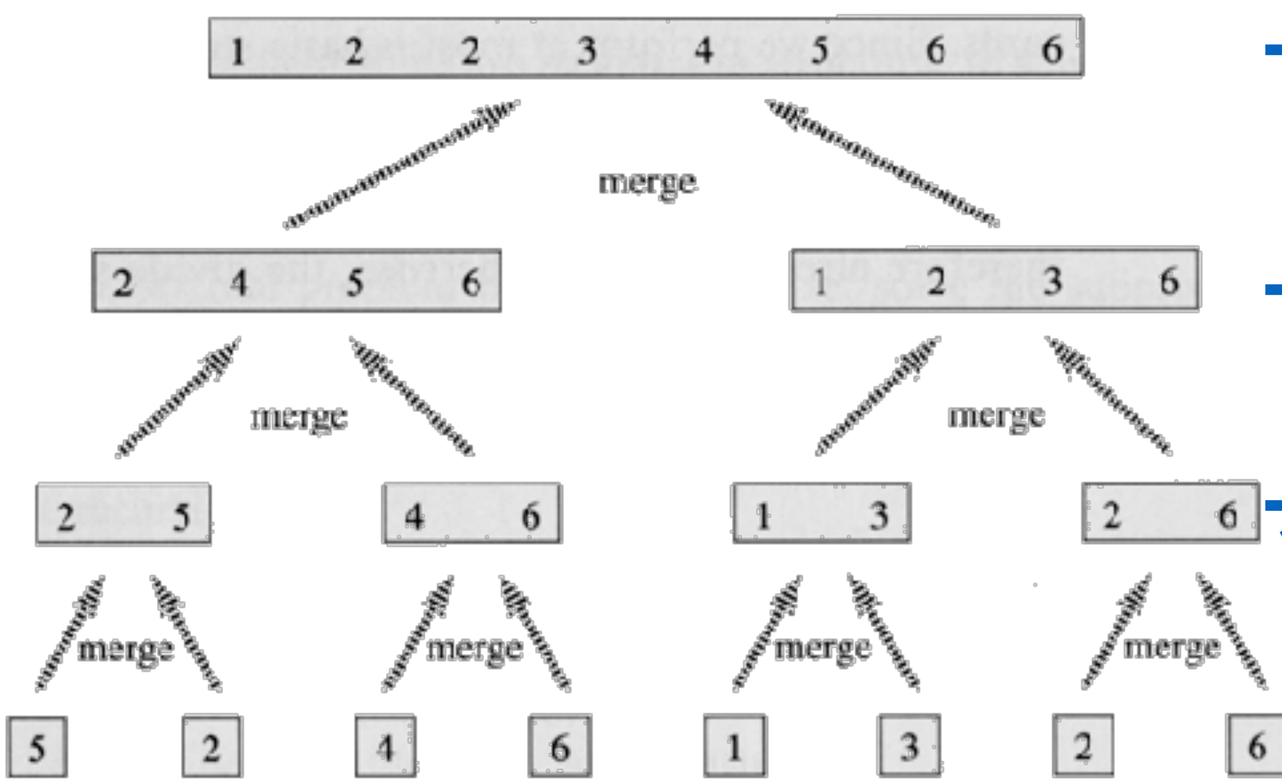


Obs 2: each call at the same level of the recursion tree does the same amount of work

http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Sorting/mergeSort.htm

Mergesort analysis

sorted sequence



#merges	time/ merge	time at this level
1	n	**n**
2	n/2	**n**
4	n/4	**n**
…	…	**…**
n/2	2	**n**

log n levels

total time: **n log n**

What makes mergesort fast?

- Insertion sort: O(n^2)
- Selection sort: O(n^2)
- Mergesort: O(n log n)

is it because it’s a
divide-and-conquer
algorithm?

what is the maximum sum of values
among all **contiguous subsequences**
of this sequence ?

13, -3, -25, 20, -3, -16, -23, 18, 20, -7, 12, -5, -22, 15, -4, 7

divide-and-conquer idea?

13, -3, -25, 20, -3, -16, -23, 18, 20, -7, 12, -5, -22, 15, -4, 7

given the max sum in each half,
can we get the best in the whole array?

13, -3, -25, 20, -3, -16, -23, 18, 20, -7, 12, -5, -22, 15, -4, 7

the best contiguous subsequence can be

- entirely contained on the left of split
- entirely contained on the right of split
- cross the split

that's a job for recursion

let's do this ourself

```
int maxSubarraySum(list) {
    leftMax = maxSubarraySum(left half of list)
    rightMax = maxSubarraySum(right half of list)
    crossMax = crossMax(list)
    return max(leftMax, rightMax, crossMax)
}
```

13, -3, -25, 20, -3, -16, -23, 18, 20, -7, 12, -5, -22, 15, -4, 7

13, -3, -25, 20, -3, -16, -23, 18, 20, -7, 12, -5, -22, 15, -4, 7

13, -3, -25, 20, -3, -16, -23, 18, 20, -7, 12, -5, -22, 15, -4, 7

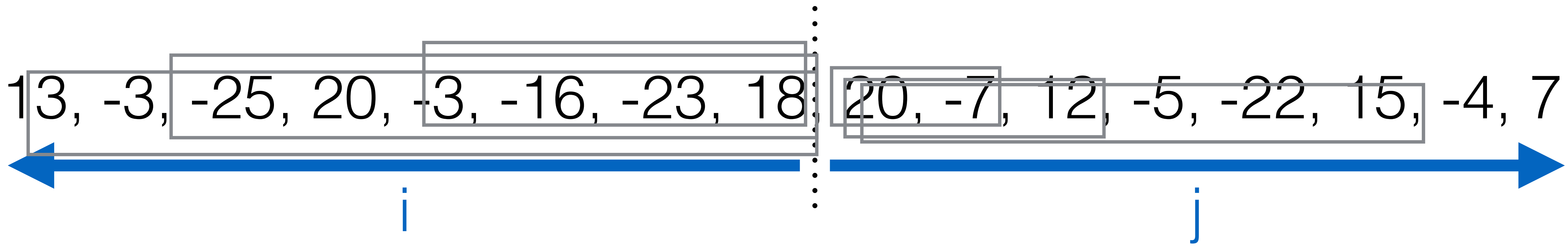
```
int maxSubarraySum(list) {
    leftMax = maxSubarraySum(left half of list)
    rightMax = maxSubarraySum(right half of list)
    crossMax = crossMax(list)
    return max(leftMax, rightMax, crossMax)
}
```

13, -3, -25, 20, -3, -16, -23, 18, 20, -7, 12, -5, -22, 15, -4, 7

i j

```
int crossMax(list) {
    int crossMax = -inf (or sum all neg vals in list)
    for (int i = 0; i < n/2-1; i++)
        for (int j = n/2+1; j < n; j++)
            sum = sum values in list from i to j
            crossMax = max(crossMax, sum)
    return crossMax
}
```

time?



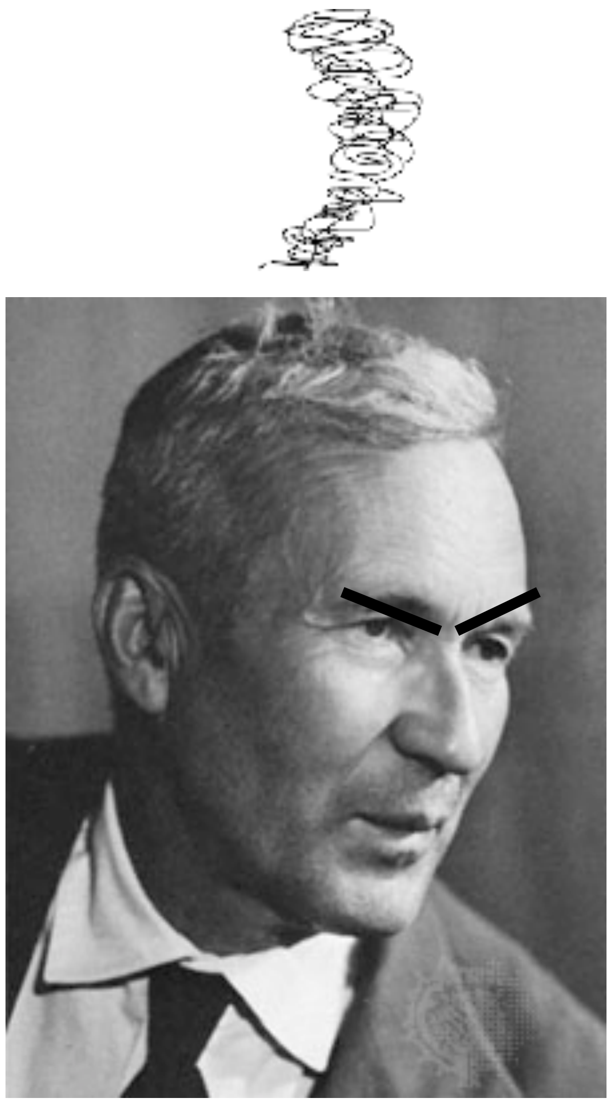

```
int crossMax2(list) {
    int rightCrossMax = -inf
    int rightRunningSum = 0
    for (i = n/2; i < n; i++) {
        rightRunningSum += list(i)
        rightCrossMax = max(rightCrossMax, rightRunningSum)
    }
    int leftCrossMax = … (same idea, but going towards left)
    return leftCrossMax + rightCrossMax;
}
```

Time?

```
int maxSubarraySum(list) {
    leftMax = maxSubarraySum(left half of list)
    rightMax = maxSubarraySum(right half of list)
    crossMax = crossMax2(list)
    return max(leftMax, rightMax, crossMax)
}
```

time?

The 1st divide-and-conquer algorithm



I conjecture that multiplication cannot be solved in faster than Theta(n^2).

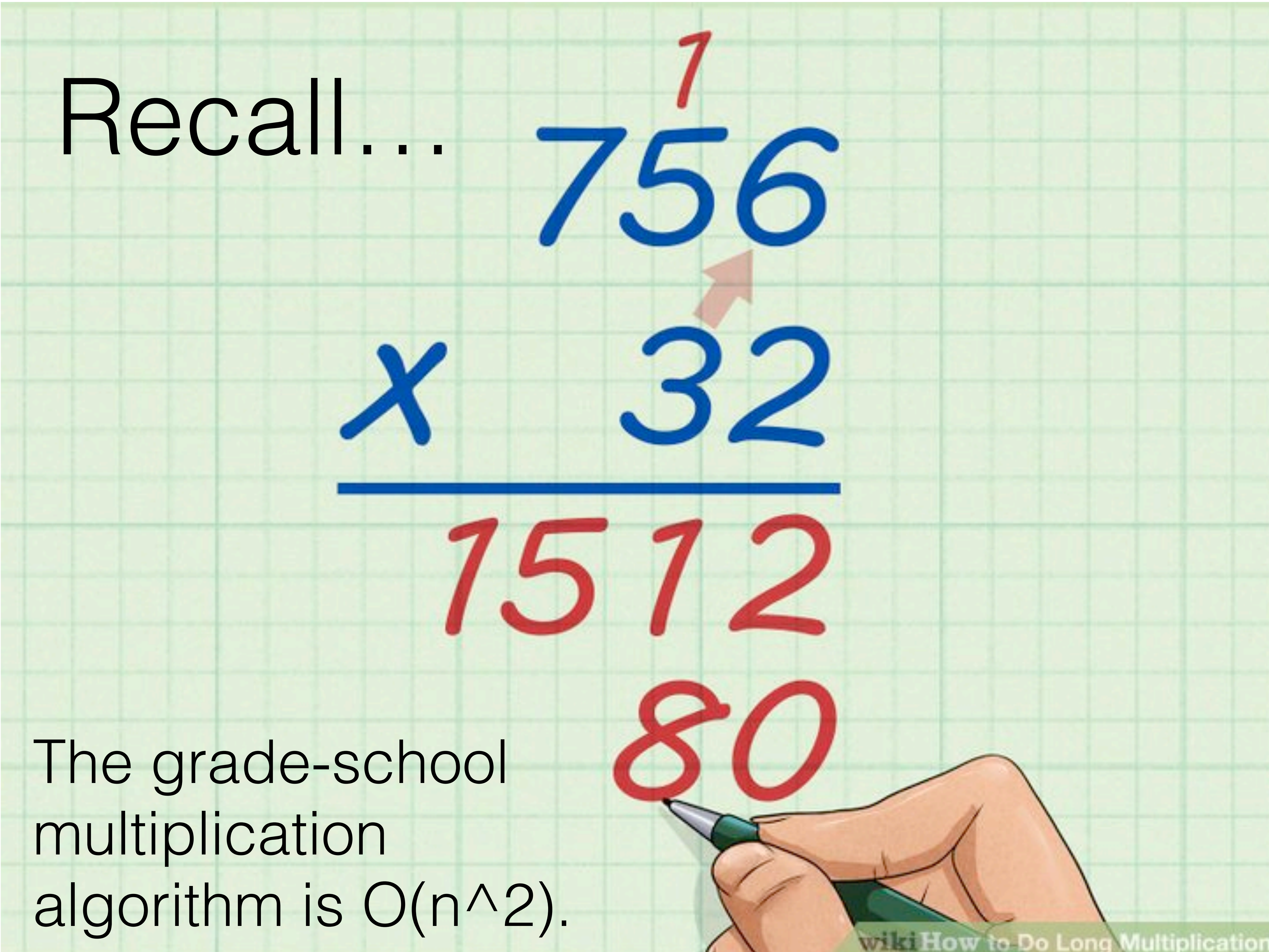
Uh, … Prof?
I think you can…
Will you check
my work?

A. N. Kolmogorov

A. A. Karatsuba

https://www.britannica.com/biography/Andrey-Nikolayevich-Kolmogorov
https://lt.wikipedia.org/wiki/Vaizdas:A.A.Karatsuba_graduate.jpg

Recall…



The grade-school multiplication algorithm is O(n^2).

http://www.wikihow.com/Do-Long-Multiplication

Karatsuba fast multiplication

explained on the board

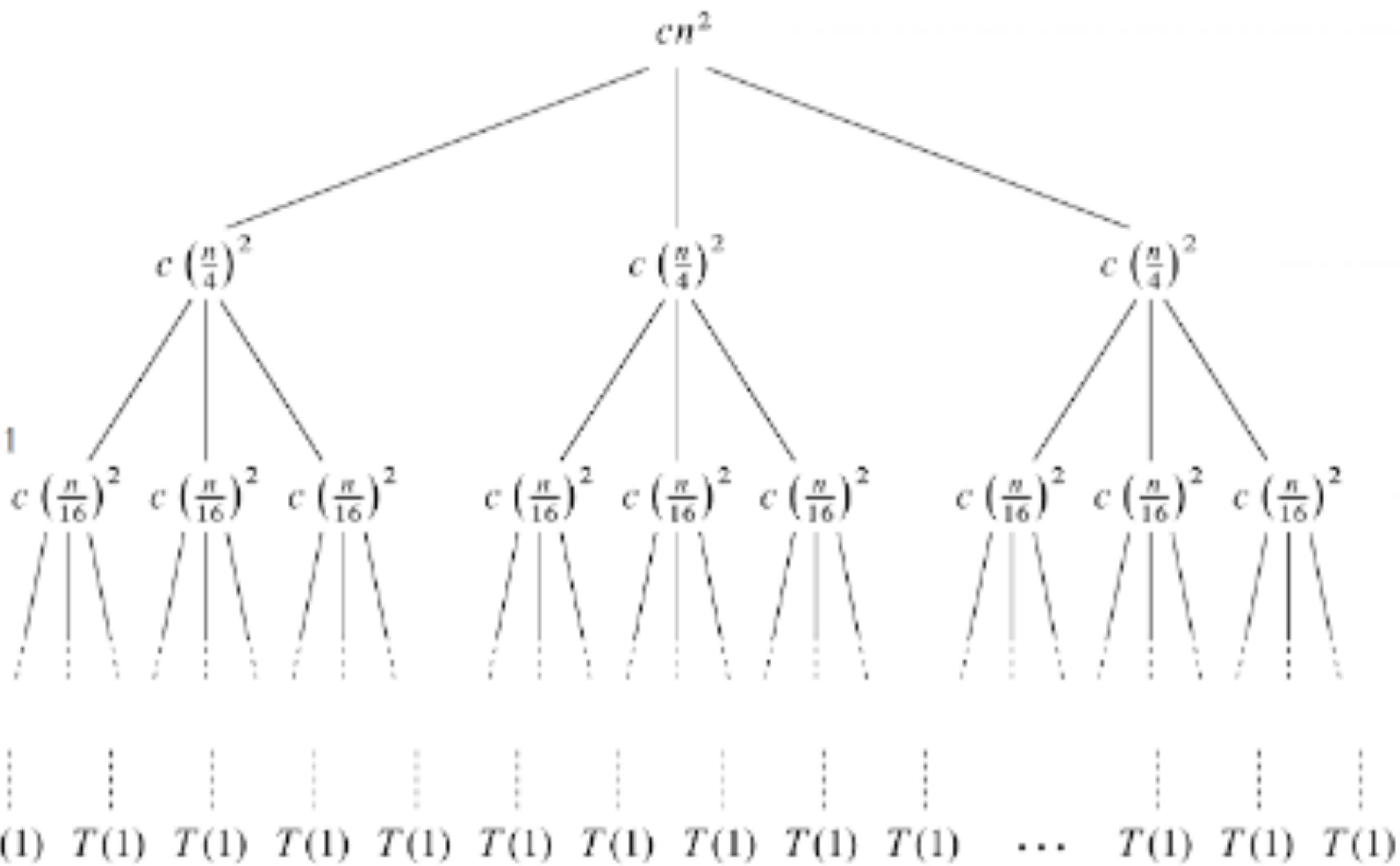
Recap: divide-and-conquer algorithms

- anatomy of a divide-and-conquer algorithm
  - divide: split problem into smaller subproblems
  - recurse: solve each subproblem recursively
  - conquer: combine the results of the subproblems
- D&C not inherently more efficient. Need
  - evenly split subproblems
  - efficient divide and conquer steps

Recap: divide-and-conquer algorithms

Problem	Non-D&C	D&C
sorting	insertion sort selection sort $O(n^2)$	mergesort quick sort O(n log n)
max-subarray-sum	brute force $O(n^3)$	O(n log n)
multiplication	grade school $O(n^2)$	Karatsuba $O(n^{\log_2 3})$

Divide-and-conquer algorithms in practice

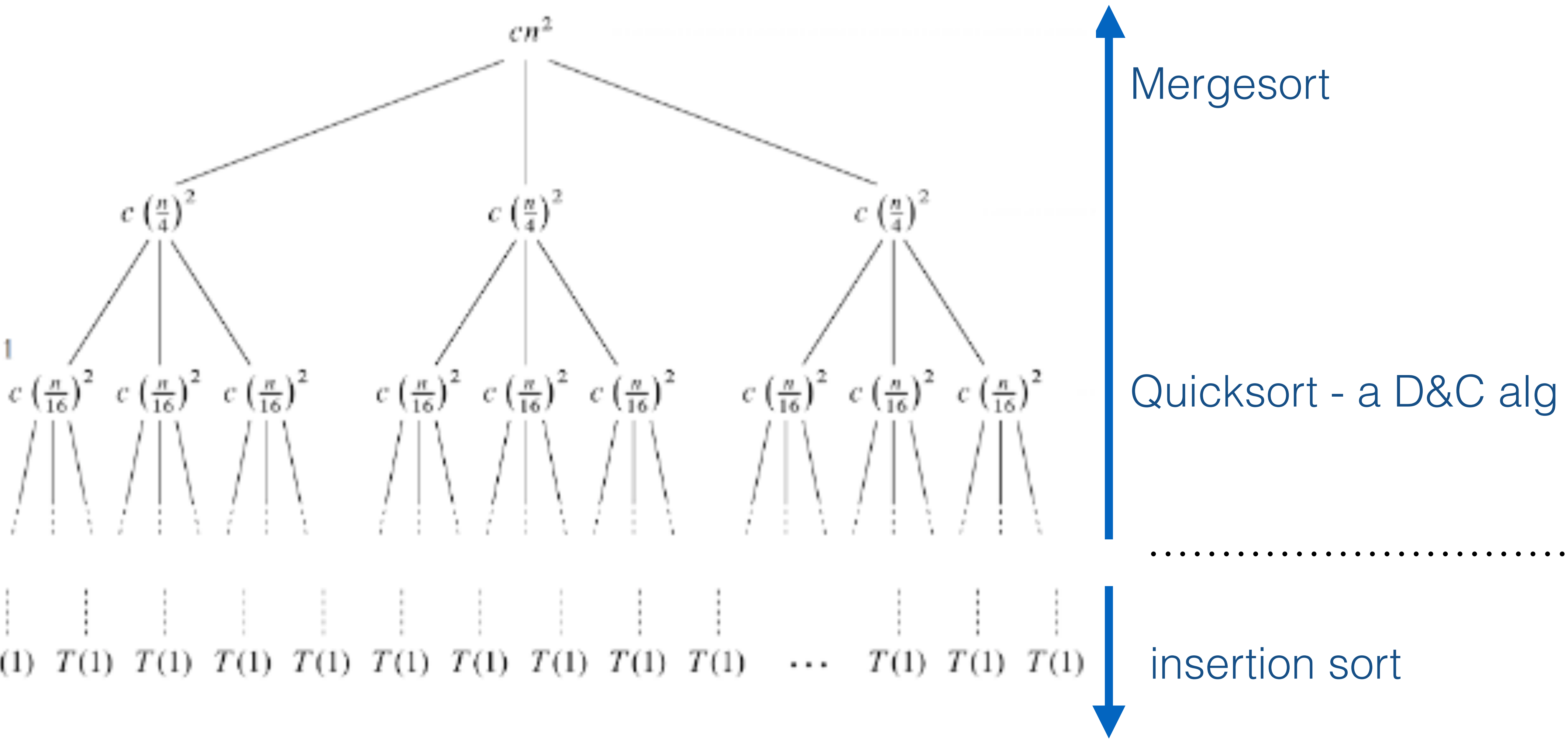


above a certain threshold size, asymptotic behavior dominate -
use a D&C alg

below threshold size, constant overhead dominates -
use simple arg

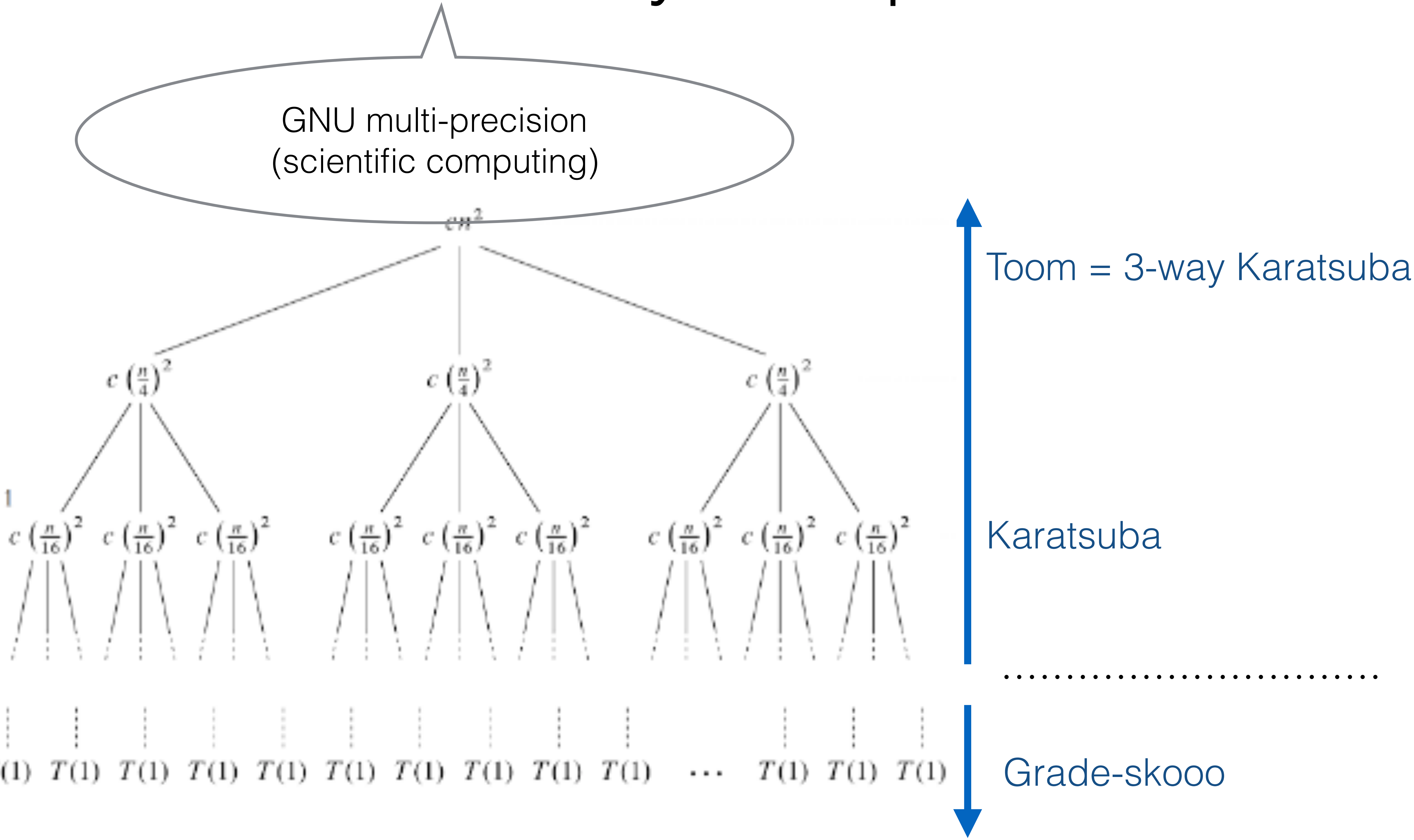
http://staff.ustc.edu.cn/~csli/graduate/algorithms/book6/chap08.htm

Java implementation of Arrays.sort



http://staff.ustc.edu.cn/~csli/graduate/algorithms/book6/chap08.htm

GMP library multiplication



http://staff.ustc.edu.cn/~csli/graduate/algorithms/book6/chap08.htm

Next time…

- Why Karatsuba's running time is $O(n^{\log_2 3})$
- How to analyze D&C algorithms
- Exam study questions, bring them…