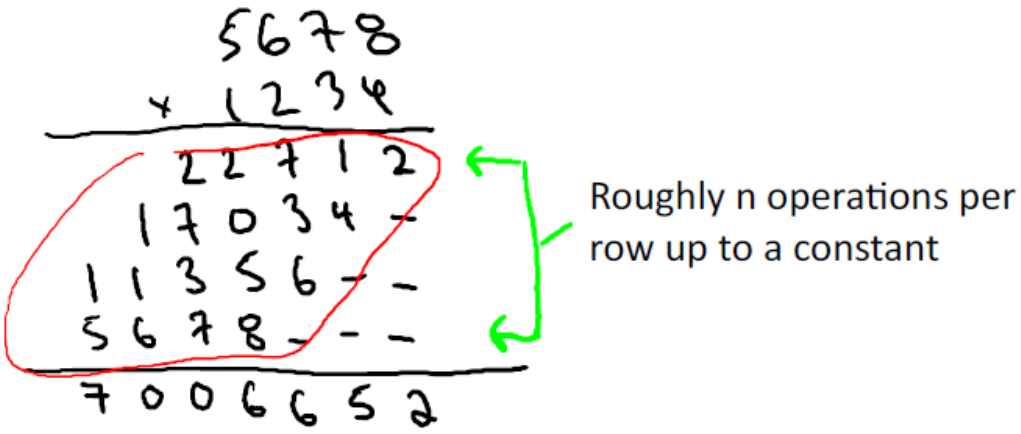
Part1-Introduction

Completed on	@2020/03/14
Key videos	https://www.coursera.org/learn/algorithms-divide-conquer/lecture/8vama/about-the-course https://www.coursera.org/learn/algorithms-divide-conquer/lecture/wKEYL/karatsuba-multiplication https://www.coursera.org/learn/algorithms-divide-conquer/lecture/wW9On/merge-sort-analysis https://www.coursera.org/learn/algorithms-divide-conquer/lecture/q5fV4/guiding-principles-for-analysis-of-algorithms
Note	Introduction to algorithms by two examples, i.e., Karatsuba Multiplication and Merge Sort.

Note

▼ Integer Multiplication

- 最原始的计算：输入：两个n位数字；输出：两个数字的乘积；最原始的运算包括加(add)与乘(multiply)
- 一般整数乘法：考虑乘法+加法，每一行地复杂度≤O(2n)；共有n行 ⇒ 总体地算法复杂度为O(n^2)



- 能否有更好的大数乘法?

▼ Karatsuba Multiplication

$$x \cdot y = 10^n ac + 10^{n/2}(ad + bc) + bd$$

一种快速相乘算法，将两个大数x,y分解成各自一办得到a,b,c,d，然后就可以根据下列公式，得到优化后的成绩，主要是中间部分可以分解为以下方法，节省一次运算，此外可以进行递归，recursive operation进一步分解：

$$ad + bc = (a + b)(c + d) - ac - bd$$

相比之下，只需三次基本操作，而不需要原有的四次操作分别计算ac，ad，bc，bd

▼ About the Course

主要关注主题

- Vocabulary for design and analysis of algorithm

"Big-Oh" notation，应该值得是算法复杂度

"sweet spot”

- Divide and conquer algorithm design paradigm
- Randomization in algorithm design

QuickSort, primality testing, hashing

- Primitives for reasoning about graphs

最短路径，连通性问题

- Use and implementation of data structure

Heaps, balanced **binary search trees**, **hashing tables**（重点讲） and some variants

主要讲解问题

1. Greedy algorithm design paradigm 贪婪算法
2. Dynamic programming algorithm design paradigm 动态规划
3. NP-complete problems and what to do about them NP完备问题
4. Fast heuristics with provable guarantees
5. Fast exact algorithms for special cases

6. Exact algorithms that beat brute-force search

▼ Merge Sort: Motivation and Example

- Motivation
  - 是对于分而治之方法的范例，比Selection, Insertion, Bubble sorts这些排序算法更为高效O(nlogn)
  - 可以帮助掌握最坏情况分析以及渐进分析的原理 guiding principles for algorithm analysis (worst-case and asymptotic analysis)
- 代码思路：显示把一串数列分为一分为二，然后递归排序之后，再进行合并（Merge）。

▼ Merge Sort: Pseudocode

- 伪代码：就是同时从开头遍历两个分数组，若其中一个元素较小，则复制到已声明的空间中并+1，以此类推

Pseudocode for Merge:

```
C = output [length = n]
A = 1st sorted array [n/2]
B = 2nd sorted array [n/2]
i = 1
j = 1
```

```
for k = 1 to n
        if A(i) < B(j)
                C(k) = A(i)
                i++
        else [B(j) < A(i)]
                C(k) = B(j)
                j++
end
```

(ignores end cases)

- 复杂度分析，主要考虑merge这个步骤，合并数组长度为N

  初始复制2次操作，后续循环加判断4N个操作 ⇒ 总体复杂度 ≤ 4N+2 ≤ 6N
- 运行时间分析（Running time），精准的上界 ⇒ 最多以下操作次数：

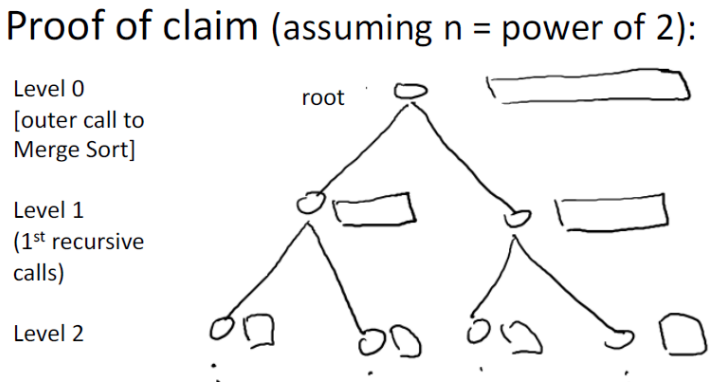
$$T(n) \leq 6n \log_2 n + 6n = \mathcal{O}(n \log n)$$

  比一般排序的O(n^2)复杂度更低

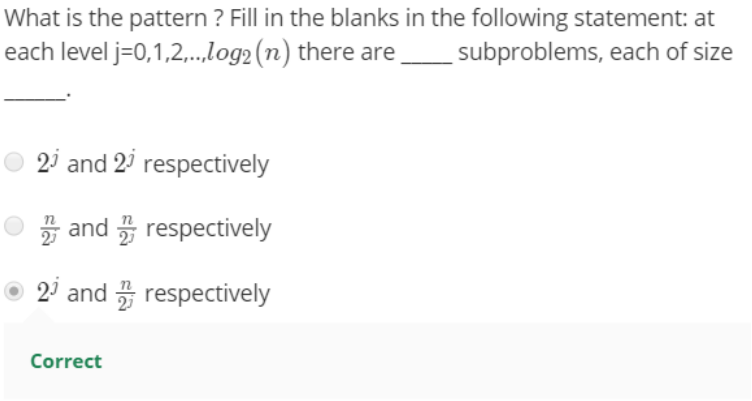
▼ Merge Sort: Analysis

- recursion tree 递归跟踪分析方法

  一共有 log2N个层次（this recursion tree has log2N levels）



  对于第j层的子问题数量和大小分析：



- 最终复杂度分析

  1）首先对于长度为n的数组，每一层都会会花6n的时间进行合并 ⇒ 一共有log2N个子层+1最终层

Proof of claim (assuming n = power of 2) :

At each level j=0,1,2,.., $\log_2 n$,
Total # of operations at level j = 0,1,2,...,$\log_2 n$

$$\leq 2^j * 6(\frac{n}{2^j}) = 6n$$

# of level-j subproblems

Size of level-j subproblem

Work per level – j subproblem

Total

$$6n(\log_2 n + 1)$$

Work per level

# of levels

▼ Guiding Principles for Analysis of Algorithms

结合归并排序算法进行divide-and-conquer范例算法得学习

- worst-case 最恶劣情况分析：计算整个算法的上界，分析相对较为简单；

  对应的"平均情况"和基准分析需要相关领域的知识

As Opposed to
--"average-case" analysis
--benchmarks

REQUIRES DOMAIN KNOWLEDGE

- 分析时会忽略常数项以及低阶项 ⇒ 真正影响性能的是最高次项
- asymptotic analysis 渐进分析：比较注重分析较为大型连续的数据，而不是简单的单个案例分析！

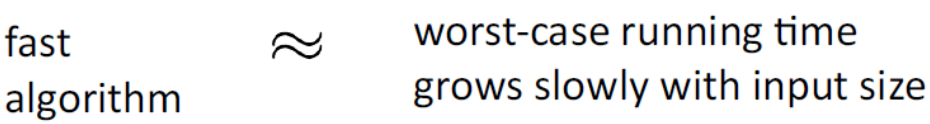
Asymptotic Analysis : focus on running time for large input sizes n

Eg : $6n\log_2 + 6n$ "better than" $\frac{1}{2}n^2$

MERGE SORT

INSERTION SORT

- 什么是快的算法：随着输入大小增长慢，接近常数级复杂度



Usually : want as close to linear (O(n)) as possible

Reference

- 大数乘法Karatsuba Multiplication

  Karatsuba算法
  https://zh.wikipedia.org/wiki/Karatsuba%E7%AE%97%E6%B3%95

- Merge Sort

  归并排序
  https://zh.wikipedia.org/wiki/%E5%BD%92%E5%B9%B6%E6%8E%92%E5%BA%8F

  插入排序
  https://zh.wikipedia.org/wiki/%E6%8F%92%E5%85%A5%E6%8E%92%E5%BA%8F

- Sorting Algorithms

  Asymptotic Analysis--渐近分析
  https://blog.csdn.net/dingchenxixi/article/details/52449424