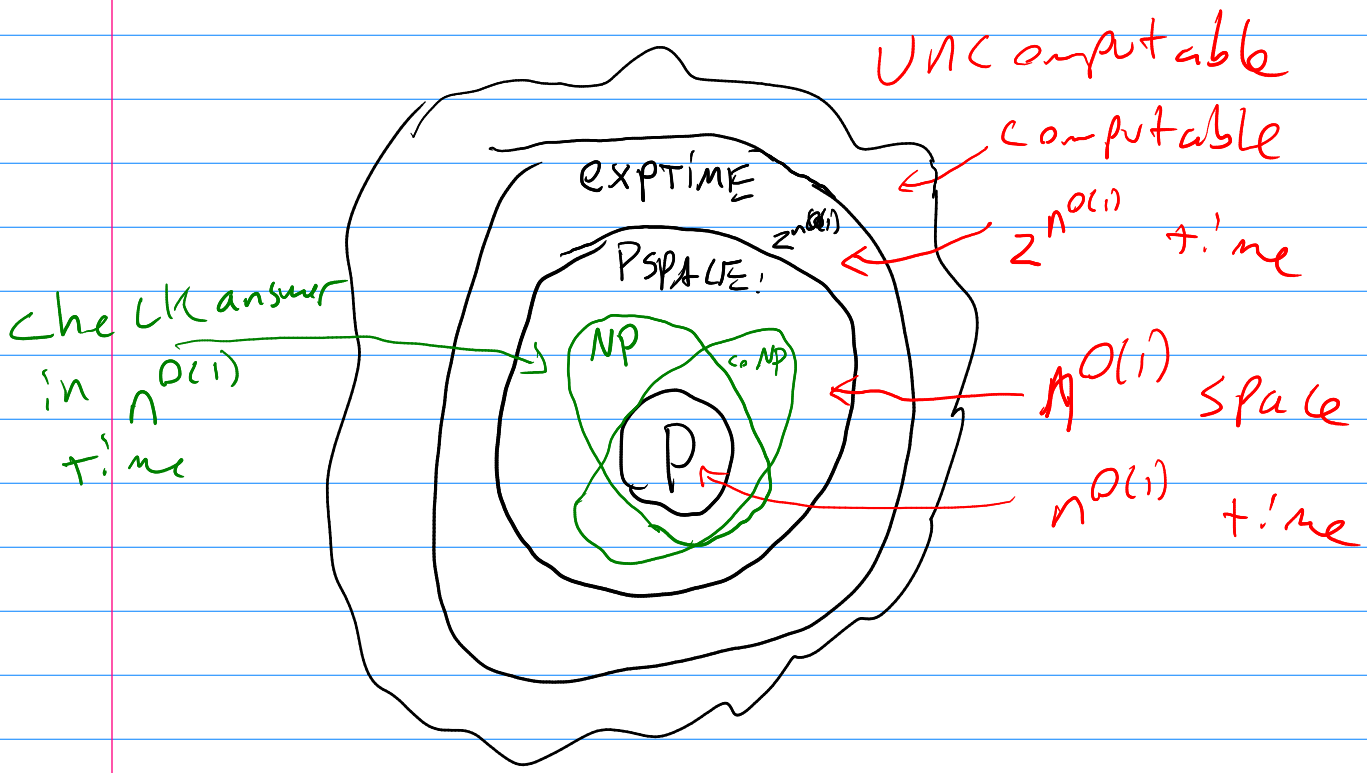
Algorithms & Complexity

How quickly we can/can't solve problems

Complexity: Broad classification of problems



Algorithms: focuses within P.

Insertion Sort vs Merge Sort
$O(n^2)$ $O(n \log n)$

Key topics:
- Dynamic Programming
- Graph Algorithms
- Linear Programming

Example Algorithm

Multiplication

```
   331
 x 388
-----
  2648
 2648
 993
-----
128428
```

n intermediate products of length n

How many steps for n digit multiplication?

First: n digit addition

single pass, right to left, constant time each

$\Rightarrow O(n)$ time.

multiplication:

multiply all pairs

$\Rightarrow O(n^2)$ time

Can we do better?

Karatsuba multiplication

$5124\,7980 \times 6172\,5863\,37$

A_1 A_2 B_1 B_2

$$A = 10^{n/2} \cdot A_1 + A_2$$
$$B = 10^{n/2} B_1 + B_2$$

$$AB = 10^n \cdot A_1 B_1 + 10^{n/2}(A_1 B_2 + B_1 A_2) + A_2 B_2$$

n digit Product
= 4 products of $\frac{n}{2}$ digits + 3 length-$O(n)$ additions

$$f(n) = 4f(\tfrac{n}{2}) + O(n)$$
$$= O(n^2)$$

$4^{\log_2 n} = 2^{2\log_2 n} = n^2$

$$(A_1 + A_2)(B_1 + B_2) = A_1 B_1 + A_2 B_2 + A_1 B_2 + B_1 A_2$$

So suffices to compute 3 products of $\frac{n}{2}$ digits:

$A_1 B_1$, $A_2 B_2$, and $(A_1 + A_2)(B_1 + B_2)$.

$$f(n) = 3f(\tfrac{n}{2}) + O(n)$$
$$= O(n^{\log_2 3}) \approx O(n^{1.585})$$

$3^{\log_2 n} = 2^{\log_2 3 \cdot \log_2 n} = n^{\log_2 3}$

Fancier Results

Toom-Cook:

split 3 ways, do 5 multiplications

$$O(n^{\log_3 5}) \approx n^{1.465}$$

4 ways, do 7 multiplications

$$n^{\log_4 7} \approx n^{1.403}$$

k ways, do $2k-1$ multiplications

$$c(k) \cdot n^{\log_k (2k-1)} \text{ time}$$

↑ exponential in k.

Schönhage-Strassen: $O(n \log n \log\log n)$ via an FFT

Fürer, Harvey-van der Hoeven: $O(n \log n)$

Simpler algorithms: better constants

Fancier algorithms: better asymptotics.