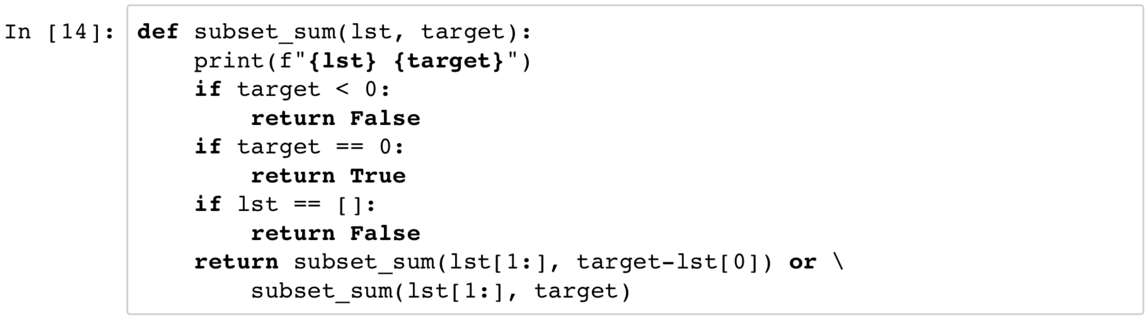
Today

- Complexity analysis
- Text decomposition
- DP



```
In [14]: def subset_sum(lst, target):
             print(f"{lst} {target}")
             if target < 0:
                 return False
             if target == 0:
                 return True
             if lst == []:
                 return False
             return subset_sum(lst[1:], target-lst[0]) or \
                 subset_sum(lst[1:], target)
```

Asymptotic Worst-case Runtime Complexity

Runtime of Alg is $\Theta(f(\underline{n}))$
↑
size of the input

Complexity of multiplication

lattice → $\Theta(n^2)$ $\Theta(n^{\log_2 3})$ ← Karatsuba $\Theta(n\log n)$ Someone [2019

1000 x 2000

$n(\log 1000)$

$T(n)$ → worst-case runtime of alg X on input of size n

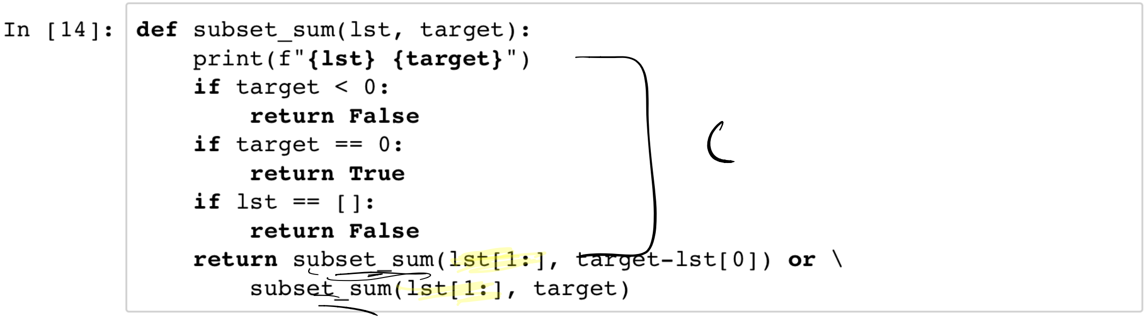
$T(n) = \Theta(f(n))$

$T(n)$ is $O(f(n))$

$f(n)$ is $O(T(n))$

Merge Sort is $O(n^n)$

is <u>not</u> $\Theta(n^n)$



```
In [14]: def subset_sum(lst, target):
             print(f"{lst} {target}")
             if target < 0:
                 return False
             if target == 0:
                 return True
             if lst == []:
                 return False
             return subset_sum(lst[1:], target-lst[0]) or \
                 subset_sum(lst[1:], target)
```

$SS(n)$ - worst-case subset-sum on list of length n

$$SS(n) \leq SS(n-1) + SS(n-1) + O(1)$$

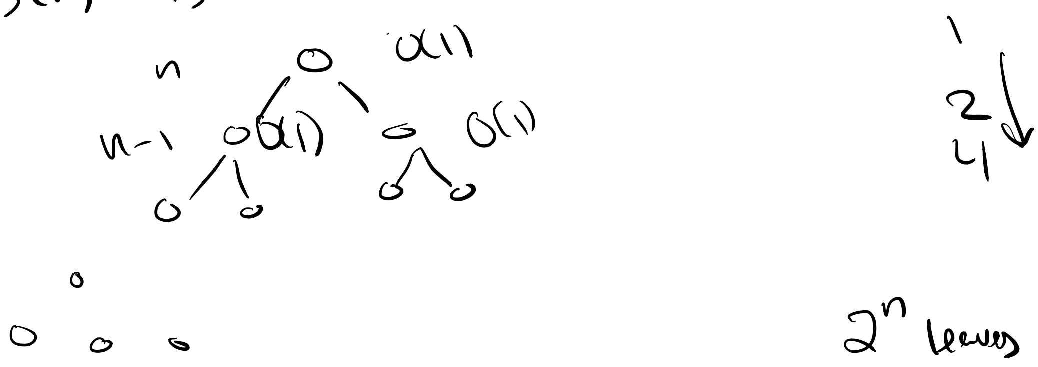
$$SS(n-1) \leq 2\,SS(n-2) + O(1)$$

$$SS(0) = O(1)$$

$$SS'(n) = 2SS'(n-1) + O(1) \rightarrow \Theta(2^n)$$

$$SS'(0) = O(1)$$

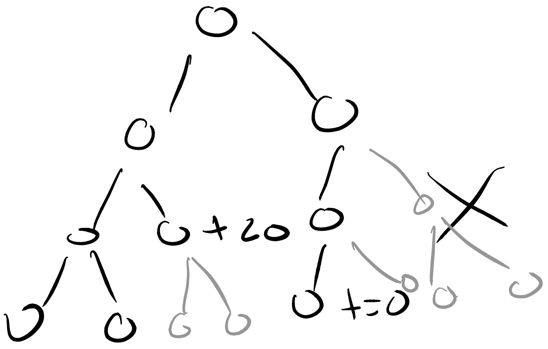
$$SS(n) \text{ is } O(SS'(n))$$



in a binary tree

# of leaves is Θ(# of nodes)

SS(n) is $O(2^n)$



$T($ subset_sum$($ S, sum(S)+1$)) = \Theta(2^n)$

worst-case

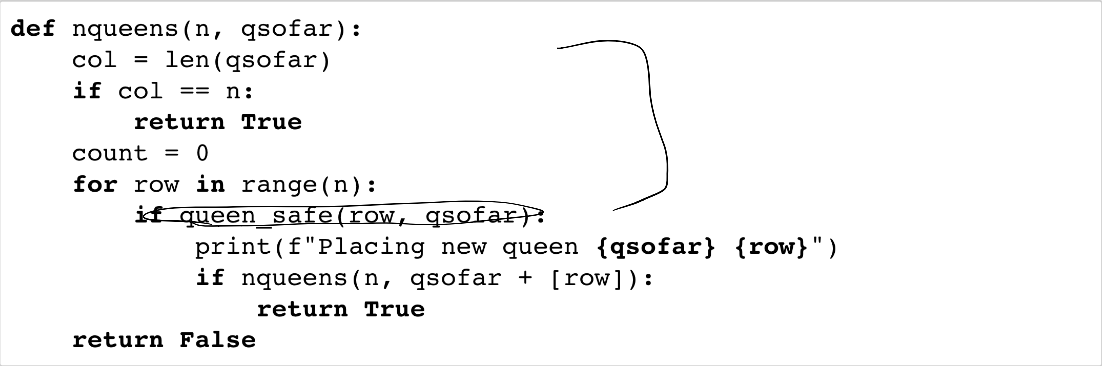
$\downarrow$

SS(n) $\geq$ subset_sum(S, sum(S)+1)

$\uparrow$ n

$= \Theta(2^n)$

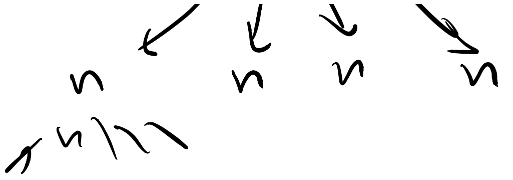
SS(n) is $\Omega(2^n)$ [2^n is O(SS(n))]



```
def nqueens(n, qsofar):
    col = len(qsofar)
    if col == n:
        return True
    count = 0
    for row in range(n):
        ~~if queen_safe(row, qsofar):~~
            print(f"Placing new queen {qsofar} {row}")
            if nqueens(n, qsofar + [row]):
                return True
    return False
```

NQ(n) → runtime of nqueens(n, [])

$\frac{\Sigma}{n}$



n^2

n^3

n^n

$O(n^n)$

Text segmentation

"This is a hard course" (no spaces)

this|is|a|hard|course|

this| isahardcourse|

i| sahardcourse| ✗

is| ahardcourse|

a| hardcourse|

|ha|rdcourse|

rd|course) ✗

hard| course|

```
segment ( string, n)
  for i = 1 to len(string) - n:
    if isword (string (n .. n+i))
      if segment ( string, n+i)
        return True
  return False
```