Recurrence solutions:

* T(n) = T(n-1) + n Θ(n2)
  + Recursive algorithm that loops through the input to eliminate one item
* T(n) = T(n/2) + c Θ(lgn)
  + Recursive algorithm that halves the input in one step
* T(n) = T(n/2) + n Θ(n)
  + Recursive algorithm that halves the input but must examine every item in the input
* T(n) = 2T(n/2) + 1 Θ(n)
  + Recursive algorithm that splits the input into 2 halves and does a constant amount of other work

Methods for solving recurrence solutions:

* Iteration method
  + Convert the recurrence into a summation and try to bound it using a known series
  + Iterate the recurrence until the initial condition is reached.
  + Use back-substitution to express the recurrence in terms of *n* and the initial (boundary) condition.
* Substitution method
  + Guess a solution
  + **T(n) = O(g(n))**
  + Induction goal: apply the definition of the asymptotic notation
  + **T(n) ≤ c g(n), for some c > 0 and n ≥ n0**
  + Induction hypothesis: T(k) ≤ c g(k) for all k < n
  + Prove the induction goal
  + Use the **induction hypothesis** to find some values of the constants d and n0 for which the **induction goal** holds
* Recursion tree method
* A recursion tree models the costs (time) of a recursive execution of an algorithm.
* Convert the recurrence into a tree:
  + - Each node represents the cost incurred at various levels of recursion
    - Sum up the costs of all levels
* The recursion-tree method can be unreliable, just like any method that uses ellipses (…).

Usually involves geometric series

* Master method
  + The master method applies to recurrences of the form
  + *T*(*n*) = *a T*(*n*/*b*) + *f* (*n*) ,
  + where *a* ³ 1, *b* > 1, and *f* is asymptotically positive.
  + “Formula” for solving recurrences of the form:
  + where, a ≥ 1, b > 1, and f(n) > 0
  + **case 1:** if f(n) = O(nlogba -ε) for some ε > 0, then: T(n) = Θ(nlogba)
  + **case 2:** if f(n) = Θ(nlogba), then: T(n) = Θ(nlogba lgn)
  + **case 3:** if f(n) = Ω(nlogba +ε) for some ε > 0, and if
  + af(n/b) ≤ cf(n) for some c < 1 and all sufficiently large n, then:
  + T(n) = Θ(f(n))
* Muster method
  + **Muster Method** for **“***decrease and conquer”* recurrences of the form
  + ***T(n) = a T(n-b) + f(n)***
  + for some integer constants a, b > 0, d ≥ 0 .
  + If  *f(n)* is O*(nd)* then