* Cost Analysis: pick a measure of interest (e.g. number of times a certain operation is called)
  + **Recursive:** construct a recurrence relation for the measure of interest, use master theorem
  + **Iterative:** do loop analysis to show the cost of each loop and thus the total cost
* Correctness / Termination
  + **Recursive (strong or weak induction):** show that the algorithm returns the correct result for the base case, assume it returns the correct result for the recursive calls, show that it returns the correct result after the recursive calls
  + **Iterative (loop invariant)**: construct some loop invariant that shows that the algorithm has built a “correct” partial solution after iterations. Show by induction that it holds for all .
  + **Termination**: show that the recursive calls are happening on instances of decreasing size or that the while-loop’s continuation condition will eventually be false.
* Master Theorem (for , consider )
* Divide and Conquer Generic Strategy (where size of input)
  + If we are at the base case (typically ), return the trivial right answer
  + Otherwise:
    1. Break the input into parts of size
    2. Recursively call the algorithm on each of these parts to get results
    3. Take the results and merge them together to get the final answer and return it
* Generic Greedy Strategy:
  1. Sort the input according to a certain attribute (either in ascending or descending order)
  2. Initialize some answer to be returned after step 3
  3. Loop over the sorted input
     + If the current selection can be added to the answer, add it to the answer
     + Otherwise skip over the current selection
* Generic Greedy Correctness Proof:
  + Show feasibility of solution – prove that the solution doesn’t violate the constraints of the problem
  + Greedy “stays ahead”
    - Let be some optimal solution and be the solution made by the greedy algorithm
    - Come up with some function some numeric aspect of arbitrary solution , use induction to show that for all
    - Use the above relationship to show that (usually contradiction)
  + Exchange argument: let be some optimal solution and be the solution made by the greedy algorithm
    - Type 1: Swapping within the solution
      * Assume there is a pair of elements in whose order disobeys the greedy heuristic
      * Show that by switching this pair of elements to obey the greedy heuristic, the cost of this modified optimal solution is no worse than the “cost” of the original optimal solution
    - Type 2: Swapping out mismatches
      * Assume that and
      * Show that by swapping with , the “cost” of does not increase beyond the “cost” of
    - Use induction to show that if we keep doing these switches to modify the optimal solution into the greedy solution, we can transform at no further “cost”
* Dynamic Programming Solution
  + Lookup Table (the array where all solutions are stored)
    - For an input problem, let be the -dimensional array that stores optimal sub-solutions
  + Semantic Array (English description of the lookup table)
    - = value of the optimal solution for problem with .
    - The answer will be located at , where the initial input has .
  + Computational Array (Mathematical description of the lookup table)
    - operation that uses sub-solutions to obtain the solution to the initial input
  + Demonstrate that equivalence between the sematic and computational arrays to prove correctness