Noel Ball
HW7

15.1-2.) Selecting the last activity to start that is compatible with previously selected activities is an approach to a greedy algorithm because each step is making the most optimal choice in order to solve the global problem of fitting as many compatible activities into the schedule. This approach is optimal because the last activity to start that is compatible with the activity after it will result in the smallest activity in the possible slot, reducing to the subproblem of all the next possible activities to be compared, just like the example relating the next start time after the finish time before it but in reverse.

15.2-5.) First sort the set of points in ascending order, initialize a list to store the intervals calculated, then iterate through the sorted points. For each point x_i, if it is already included in the most recent set, move to the next point, otherwise create a new unit length interval starting from the point $[x_i , x_i+1]$. The final output of the interval list will include the smallest set of intervals possible because it makes the greedy choice of creating a new interval every time a point is not included in the hopes of including as many points with that interval to solve the global problem, and any points not included in that interval are the rest of the subproblem to solve.

15.2-7.) In order to maximize the payoff, an algorithm would have to reorder sets *A* and *B* to be in descending order, so that the largest elements appear first. This ensures that the largest coefficients are paired with the largest exponents in order to maximize each term thus maximizing the product, resulting in the maximum payoff given the sets *A* and *B*.

15.3-3.) To construct the huffman tree:
Combine a and b: (a+b) = 1+1 = 2
Combine (a+b) and c: (a+b+c) = 2+2 = 4
Combine d and e: (d+e) = 3+5 = 8
Combine (d+e) and (a+b+c): (a+b+c+d+e) = 4+8 = 12
Combine f and g: (f+g) = 8+13 = 21
Combine h and (f+g): (f+g+h) = 21+21 = 42
Combine (a+b+c+d+e) and (f+g+h): (a+b+c+d+e+f+g+h) = 12+42 = 54
Giving huffman codes:
a: 0, b: 1, c: 10, d: 110, e: 111, f: 1100, g: 1101, h: 1110
Generalization:
Iterate through the frequencies, the first n fibonacci numbers, and combine nodes to build the huffman tree. Then assign binary codes according to the traversal of the tree.

15.3-8.) For every possible lossless compression scheme, there must be at least one file for which the compressed version is longer or the same length as the original file due to the number of unique files being greater than or equal to the number of unique compressed representations.