Kevin Ramirez

CAAP-CS Assignment 1

1. A greedy algorithm is based on the idea of breaking one's current problem into many nested subproblems starting with the outer most problem and choosing the optimal solution at each stage. This assumes that the optimal solution for the outer problem will lead to optimal solutions for the subproblems. It's akin to making the best choice at each stage and then solving any of the smaller problems it leaves for the next step, not considering what its future choices may be for its decision making process in the present.

2. The problem could be solved in reverse knowing the user's inputted change, and starting from zero adding the largest denomination first until the difference between the input and the sum is negative. The program would then move on to smaller and smaller denominations each cycle until our sum is equal to that of the inputted change. This however would still be considered a greedy algorithm since it still finds the optimal solution at each step. A differing algorithm all together would be one which calculates two layers at once so beginning at the larger denomination, but also the next subproblem solutions and compares the branches to see which branch yields the most optimal solution then repeating on the next level down. This resulting algorithm would have less short sighted issues that greedy algorithms. Example: If given coins of values {1, 15, 25} and having to give change for 30, greedy would choose one 25 then five 1s for a total of six coins versus the branching one which would recognize that the next denomination down can be used twice to make 30.

3. Another problem in which greedy algorithms can be used for is the knapsack problem. Given a few items with differing values and sizes how many items can be fit into a knapsack of a certain size as to accumulate the most point values? Although a greedy algorithm can be used in this instance to find local optimal solution, there are other algorithms that are better suited for this type of problem as the global optimum may be hidden by being a combination of nonoptimal solutions for every subproblem. Greedy algorithms can also be used to find the shortest path between two nodes but once again are not the best suited as a short distance between two intermediate nodes in once instance may place the program in an unoptimal position for the next subproblem, as a longer direct path may have been the best solution.