**Week 5: Greedy Algorithms**

1. **fractional\_knapsack**:

* **Guide**:
  + For the fractional knapsack problem, instead of taking or leaving an item as a whole, you can take a fraction of it. Focus on the value-to-weight ratio of each item. Sorting items based on this ratio can offer insight into which items to prioritize.
* **Pseudocode**:
* FUNCTION fractional\_knapsack(values, weights, capacity):  
   CALCULATE value-to-weight ratio for each item  
   SORT items based on this ratio in descending order  
   INITIALIZE total\_value to 0  
   FOR each item in sorted list:  
   IF item can be taken fully:  
   INCREASE total\_value and DECREASE capacity  
   ELSE:  
   TAKE fraction of item to fill capacity and INCREASE total\_value  
   RETURN total\_value

1. **coin\_change\_greedy**:

* **Guide**:
  + In the greedy approach, always try to accommodate the largest denomination of coin possible and reduce the remaining amount. Note that the greedy method may not always yield an optimal solution for certain denominations.
* **Pseudocode**:
* FUNCTION coin\_change\_greedy(coins, amount):  
   SORT coins in descending order  
   INITIALIZE count to 0  
   FOR each coin in coins:  
   USE as many of this coin as possible without exceeding amount  
   INCREASE count and REDUCE amount  
   IF amount is 0:  
   RETURN count  
   ELSE:  
   RETURN -1

1. **activity\_selection**:

* **Guide**:
  + To schedule the most number of activities without overlap, consider sorting activities by end times. Then, pick activities one by one ensuring the start time of the current activity is after or equal to the end time of the last selected activity.
* **Pseudocode**:

FUNCTION activity\_selection(activities):  
 SORT activities by their end time  
 INITIALIZE a list with the first activity  
 FOR each activity starting from the second:  
 IF start time of this activity is after or equal to end time of last added:  
 ADD this activity to the list