Assignment 11

R-5.1 Fractional Knapsack Problem

Greedy Approach:

1. Sort items by value-to-weight ratio (V/W) in descending order.
2. Iterate through sorted items:
    - If the current item's weight is less than or equal to the remaining capacity, add the whole item to the knapsack.
    - Otherwise we add a fraction of the item to fill the remaining capacity.

R-5.3 Task Scheduling Problem

Greedy Approach:

1. Sort tasks by their finish times in increasing order.
2. Iterate through sorted tasks:
    - If the current task's start time is greater than or equal to the finish time of the previously scheduled task, schedule it.

R-5.11 0-1 Knapsack Problem

Dynamic Programming Approach:

```
function knapsack(W, wt, val, n)
  if n == 0 or W == 0
    return 0
  if wt[n-1] > W
    return knapsack(W, wt, val, n-1)
  else
    return max(val[n-1] + knapsack(W-wt[n-1], wt, val, n-1), knapsack(W,[1] wt, val, n-1))
```

R-5.12 Internet Auction as Knapsack Problem

- 0-1 Knapsack: If each bidder can only win one item, it's a 0-1 knapsack problem.
- Fractional Knapsack: If bidders can win fractions of items, it's a fractional knapsack problem.

A. Memoized 0-1 Knapsack

```
function memoized_knapsack(W, wt, val, n, memo)
  if memo[n][W] != -1
    return memo[n][W]
  if n == 0 or W == 0
    memo[n][W] = 0
    return 0
  if wt[n-1] > W
    memo[n][W] = memoized_knapsack(W, wt, val, n-1, memo)
    return memo[n][W]
  else
    memo[n][W] = max(val[n-1] + memoized_knapsack(W-wt[n-1], wt, val, n-1, memo), memoized_knapsack(W, wt, val, n-1, memo))
    return memo[n][W]
```

C-5.9 Tracing Back for 0-1 Knapsack

```
function traceback(W, wt, val, n, memo)
  if n == 0 or W == 0
    return
  if memo[n][W] == memo[n-1][W]
    traceback(W, wt, val, n-1, memo)
  else
    print("Item", n, "included")
    traceback(W-wt[n-1], wt, val, n-1, memo)
```

B. Subset Sum Problem

```
function subset_sum(W, wt, n)
  if n == 0 or W == 0
    return false
  if wt[n-1] > W
    return subset_sum(W, wt, n-1)
  return subset_sum(W, wt, n-1) or subset_sum(W-wt[n-1], wt, n-1)
```

- The memoization technique is used to store intermediate results and avoid redundant calculations, improving the efficiency of the recursive solution.
- The traceback function helps to determine the actual subset of items that contribute to the optimal solution.

- The subset sum problem is a variation of the 0-1 knapsack problem where we only consider the weight/size of items and not their value.