1. “Shopping-cart problem” - a listener wins a radio station contest where they can fill a single shopping cart with items from a grocery store. If the listener wants to fill the cart so as to maximize the total value within the cart, suggest "greedy" criteria the listener should use to select items to put in the cart.

2. A similar problem is the 0-1 *Knapsack problem*:

> A thief breaks into a jewelry store carrying a knapsack that will break if its weight limit (*W*) is exceeded. The thief wants to maximize the total value in the knapsack without exceeding its weight limit *W*.

a) If the jewelry store has *n* items and we try a brute force consideration of all possible subsets of items to steal, then how many subsets of n items exist?

3. Consider solving the 0-1 Knapsack problem using a greedy algorithm.

a) What greedy criteria should we use to decide the next item to consider putting in the knapsack?

b) When would the item in (b) be rejected as infeasible?

c) Write your greedy algorithm using high-level psuedo-code (English steps).

d) What is the *O*( ) of your greedy algorithm?

e) Does your greedy algorithm produce the best (*globally optimal*) solution?

3. Consider the following 0-1 Knapsack problem with four items and a knapsack weight limit of W=10 oz.

Item, i	**Weight, w_i**	**Value , v_i**	**Value/Weight**
1	4 oz.	$40	$10/oz.
2	7 oz.	$63	$9/oz.
3	5 oz.	$25	$5/oz.
4	3 oz.	$12	$4/oz.

a) What items would your greedy algorithm have the thief steal?

b) What items would maximize the total value in the knapsack without exceeding its weight limit W ?

4. To speedup finding the globally optimal solution to the 0-1 Knapsack problem, lets consider applying dynamic programming. We need to view the problem recursively (i.e., solve optimal larger problem using optimal smaller problems)

a) What does a problem description include for 0-1 Knapsack?

b) What do we mean by the problem size of a 0-1 Knapsack instance?