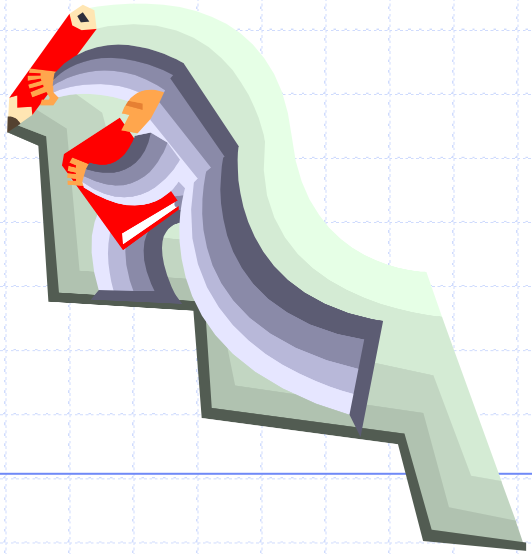
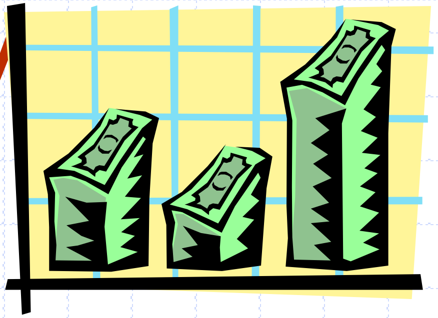
The Greedy Method



Introduction: The Greedy Method Technique



- **The greedy method** is a general algorithm design paradigm, built on the following elements:
  - **configurations**: different choices, collections, or values to find
  - **objective function**: a score assigned to configurations, which we want to either maximize or minimize
- It works best when applied to problems with the **greedy-choice** property:
  - a globally-optimal solution can always be found by a series of local improvements from a starting configuration.

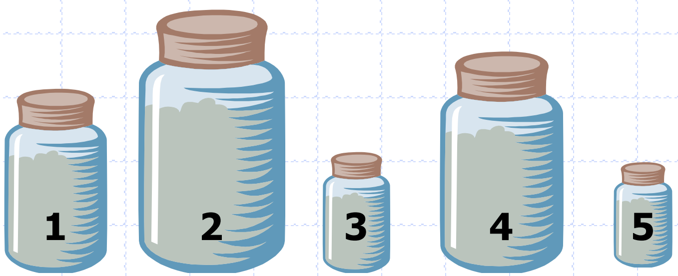
The Fractional Knapsack Problem

- Given: A set S of *n* items, with each item i having
  - b_i - a positive benefit
  - w_i - a positive weight
- Goal: Choose items with maximum total benefit but with weight at most *W*.
- If we are allowed to take fractional amounts, then this is the **fractional knapsack problem**.
  - In this case, we let x_i denote the amount we take of item i
  - Objective: maximize $$\sum_{i \in S} b_i (x_i / w_i)$$
  - Constraint: $$\sum_{i \in S} x_i \leq W$$

Example

- Given: A set S of n items, with each item i having
  - b_i - a positive benefit
  - w_i - a positive weight
- Goal: Choose items with maximum total benefit but with weight at most W.

Items:



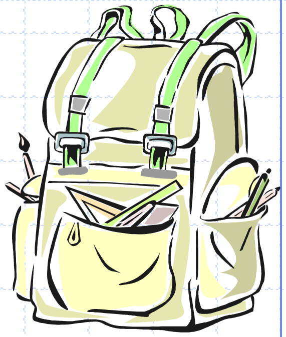
Weight:	4 ml	8 ml	2 ml	6 ml	1 ml
Benefit:	\$12	\$32	\$40	\$30	\$50
Value: ($ per ml)	3	4	20	5	50

"knapsack"

10 ml

Solution:
- 1 ml of 5
- 2 ml of 3
- 6 ml of 4
- 1 ml of 2

The Fractional Knapsack Algorithm



- Greedy choice: Keep taking item with highest **value** (benefit to weight ratio)
  - Use a heap-based priority queue to store the items, then the time complexity is O(n log n).
- Correctness: Suppose there is a better solution
  - there is an item *i* with higher value than a chosen item *j* (i.e., $v_j < v_i$) , if we replace some j with i, we get a better solution
  - Thus, there is no better solution than the greedy one

Algorithm *fractionalKnapsack*(S, W)
  Input: set S of items w/ benefit b_i and weight w_i; max. weight W
  Output: amount x_i of each item i to maximize benefit with weight at most W

for *each item i in S*
  $x_i \leftarrow 0$
  $v_i \leftarrow b_i / w_i$ {value}
$w \leftarrow 0$ {current total weight}
while $w < W$
  remove item i with highest v_i
  $x_i \leftarrow \min\{w_i, W - w\}$
  $w \leftarrow w + \min\{w_i, W - w\}$

Huffman codes

- In telecommunication, how do we represent a set of messages, each with an access frequency, by a sequence of 0's and 1's?
- To minimize the transmission and decoding costs, we may use short strings to represent more frequently used messages.
- This problem can by solved by using an extended binary tree which is used in the 2-way merging problem.

An example of Huffman algorithm

- Symbols: A, B, C, D, E, F, G

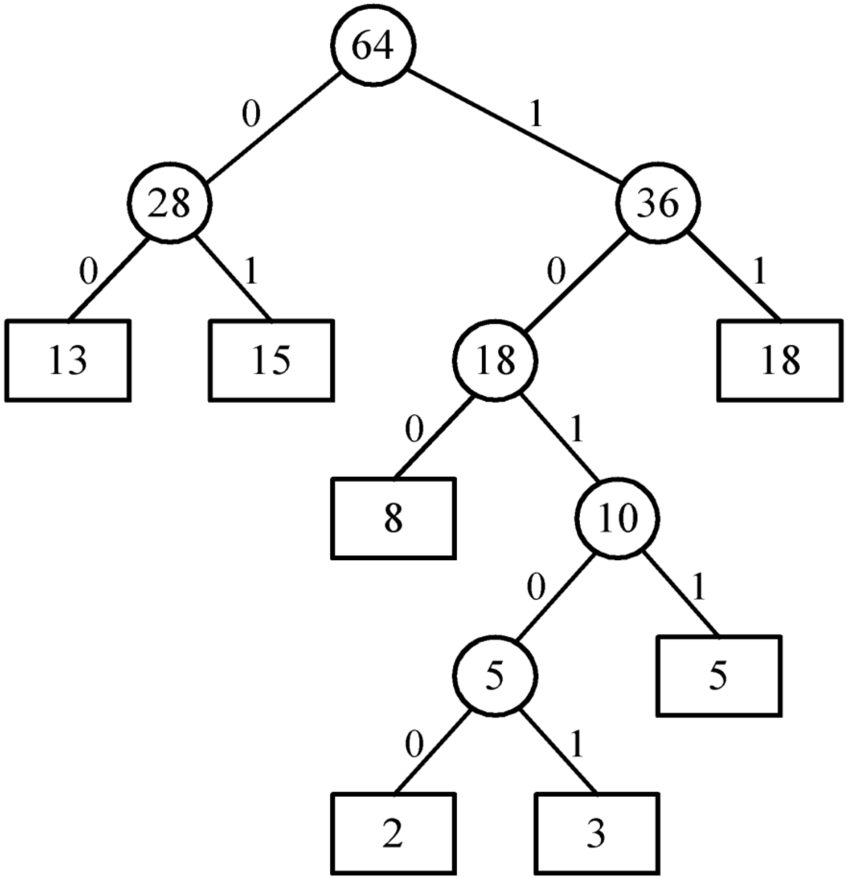
  freq. : 2, 3, 5, 8, 13, 15, 18

- Huffman codes:

  A: 10100 B: 10101 C: 1011

  D: 100 E: 00 F: 01

  G: 11



A Huffman code Tree

Application of Greedy method

- Network routing
- Huffman Tree
- Optimal storage on tape

Scope of Research of Greedy Method

- To find guaranteed optimal solution in Decision learning tree

Assignment

- Q.1) What is Greedy method?
- Q.2) Explain general method of Greedy method.
- Q.3)Explain fractional Knapsack problem with example.