Greedy Algorithms:

- Problems that demand either minimum or maximum results.
- Example: Problem – Journey from A to B.
  - Multiple Solutions: Walk, Cycle, Motor Cycle, Car, Train, Plane etc.
  - Do we have any constraints?
    - e.g. 12 hrs MAX
    - Available Options are called feasible options
    - Min/Max – If the options is feasible AND it satisfies the condition, then it is called "Optimal Solution" – **USUALLY only 1**.
    - Examples for MIN: Time, Cost.
    - Example for MAX: Profit.

EXAMPLE 1:
Coin/Note Distribution: Rs. 56 is to be distributed. Options of note/coins are 20, 10, 5, 1. Condition: Minimum number of notes/coins to be used.

Select 20	Balance: 36
Select 20	Balance: 16
Select 10	Balance: 6
Select 5	Balance: 1
Select 1	Balance: 0

Optimal Solution: 5 Notes/Coins

EXAMPLE 2:
Time: 6hrs.
Number of tasks: 5
Time required for the tasks: {5,3,4,2,1}

Greed: Maximum number of the tasks

Sorted List #1: {1,2,3,4,5}

Select 1 hr	Balance 5 hr	
Select 2 hr	Balance 3 hr	
Select 3 hr	Balance 0 hr	Total number of Tasks performed: 3

Sorted List #2: {5,4,3,2,1}

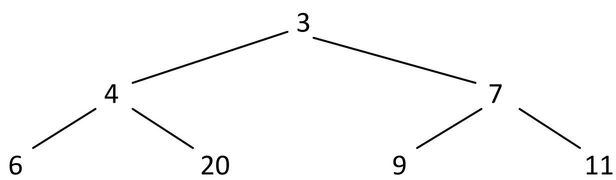
Select 5 hr	Balance 1 hr	
Select 1 hr	Balance 0 hr	Total number of Tasks performed: 2

Optimal Solution: Sorted List #1

Greedy Approach/Method/Algorithm:

- Makes optimal choice at each step as it attempts to find the overall optimal way to solve the entire problem.
    - Pure Greedy
    - Orthogonal Greedy
    - Relaxed Greedy
- Not same as Dynamic Programming BECAUSE it does not reconsider the choices.
- In many cases, it may not work and may result in "Unique Worst Possible Solution".

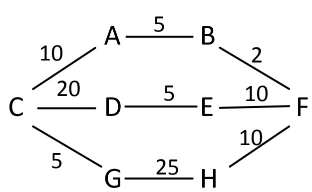
Failure EXAMPLE:



Problem: Longest root to leaf sum

Start: 3
Select 7 Total: 3+7 = 10
Select 11 Total: 10+11 = 21 **(WRONG)**

Correct: 3+4+20 = 27



Problem: Shortest path from C to F

Start: C
Select 5 Arrive at G
Select 25 Arrive at H
Select 10 Arrive at F

Solution: 5+25+10 = 40 **(WRONG)**

Optimal Solution: CABF: 10+5+2 = 17

Properties:

1. Greedy-Choice Property: A global optimum can be arrived at by selecting a local optimum.
2. Optimal Substructure: An optimal solution to the problem contains an optimal solution to the sub-problems.

Application:

- Activity Selection
- Job Sequencing
- Task Scheduling with Resource Allocation
- Fractional Knapsack Problem
- Prim's Minimal Spanning Tree
- Huffman Coding etc.

Fractional Knapsack Problem:

- Select the items that have more value
- We can break items in order to maximize value

Bag Capacity : W
Item Weights : {w1,w2,w3,…}
Item Values : {v1,v2,v3,…}

Capacity (C): 50

	Item 1 (A)	Item 2 (B)	Item 3 (C)
Weight (w)	10	20	30
Values (v)	60	100	120

Complete Item Selection : 0/1 Knapsack Problem
Fraction of Item Selection : Fractional Knapsack Problem

Items: A + B + 2/3 of C: Total weight: 10+20+(30*2/3)=50, Total Value: 60+100+(120*2/3)=240

Greedy Approach:

- Calculate the ratios (v/w) for each item.
- Sort items wrt ratios in descending order.
- Take item with highest ratio
  - Take full item if possible
  - Take fraction if full item is not possible

	Item 1 (A)	Item 2 (B)	Item 3 (C)
Weight (w)	10	20	30
Values (v)	60	100	120
Ratio (v/w)	6	5	4

Items Sorted wrt Ratio: A, B, C

C: 50	Select A	w=10	v=60	Remaining C: 50-10=40	Total Value: 60
C: 40	Select B	w=20	v=100	Remaining C: 40-20=20	Total Value: 60+100=160
C: 20	Select C	w=20/30*30=20 C/w*w			
		V=20/30*120=80 C/w*v		Remaining C: 0	Total Value: 160+80=240