WELCOME TO THE PRESENTATION

TOPIC : COIN CHANGING (DP & GREEDY)

THINGS TO BE EXPLAINED:

- DP & Greedy
- Definition Of Coin Changing
- Example with explanation
- Time complexity
- Difference between DP & Greedy in Coin Change Problem

DP : DYNAMIC PROGRAMMING

- ❖ Dynamic programming is a method for solving a complex problem by breaking it down into a collection of simpler sub problems just once and storing their solutions.

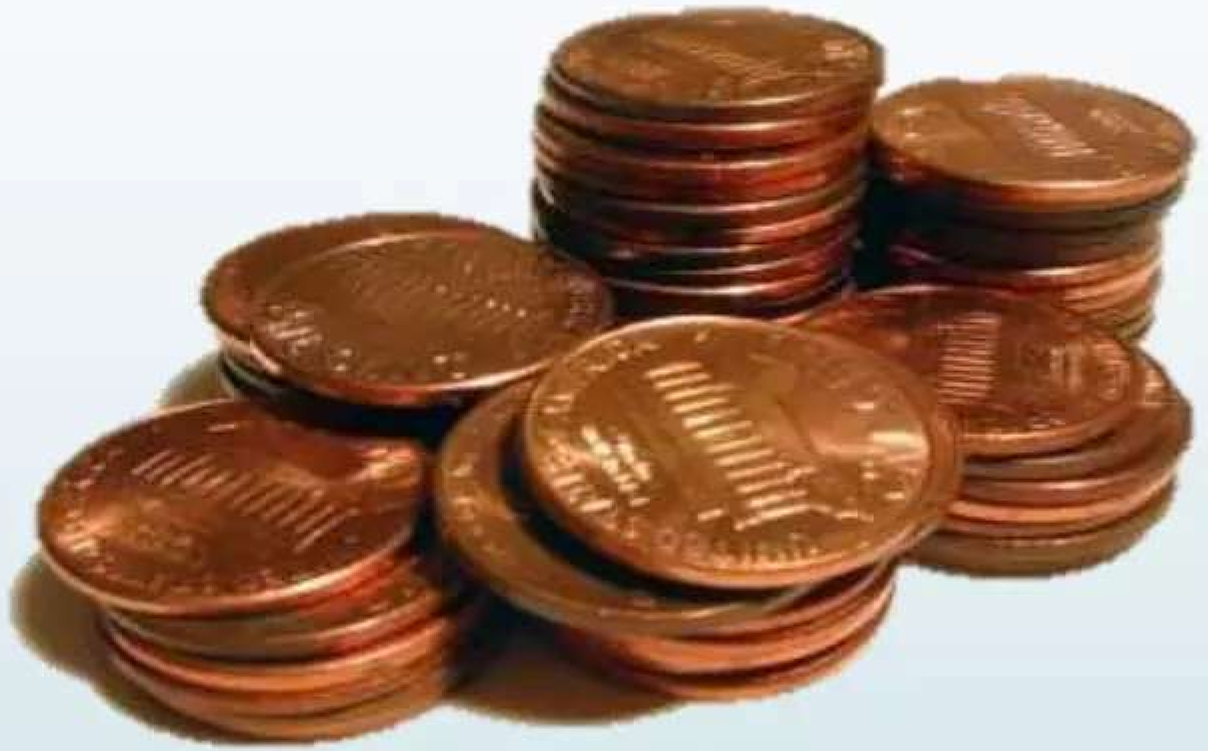
- ❖ DP is used to solve problems with the following characteristics :

  - ➢ Simple Sub problems
  - ➢ Optimal Structure Of The Problems
  - ➢ Overlapping Sub problems

GREEDY ALGORITHM

- A greedy algorithm always makes the choice that looks best at the moment. It makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution.
- An algorithm is greedy if it builds a solution in small steps, choosing a decision at each step locally , not considering what happen ahead to optimize some underlying criterion

COIN CHANGE PROBLEM



Coin change is the problem of finding the minimum number of ways of making changes for a particular amount of taka, using a given unlimited amounts of coins. It is a general case of Integer Partition.

EXAMPLE:

Set Of Coins, D = { 1,5,6,9 }
Make Change Of Taka, n = 11

MINIMUM NUMBER OF COINS

P	0	1	2	3	4	5	6	7	8	9	10	11
C[P] =	0	1	2	3	4	1	1	2	3	1	2	2
S[P] =	0	1	1	1	1	5	6	1	1	9	5	5

For P=6;

D[1] = C[P – D[1]] + 1; here D[1] = 1;
= C[6 – 1] +1;
= C[5] +1 ;
= 5 +1;
= 6

D[2] = C[P – D[1]] + 1; here D[2] = 5;
= C[6 – 5] +1;
= C[1] +1 ;
= 1+1;
= 2

D[3] = C[P – D[1]] + 1; here D[3] = 6;
= C[6 – 6] +1;
= C[0] +1 ;
= 0 +1;
= 1

$$C[P] = \begin{cases} 0, \text{ if } P = 0 \\ \min \left(C[P-D[i]] + 1 \right), \text{ if } P > 0 \\ D[1] <= D[i] <= P \end{cases}$$

EXPLANATION OF COIN CHANGE (DP) :

- If D is unsorted , then sort D in Ascending Order
- Take two array C[P] and S[P] where,

  C[P] = Minimum number of coins used for P Tk

  S[P] = Last coin used to make P Tk.
- Find C[P] using the following equation,

$$C[P] = \begin{cases} 0, \text{ if } P = 0 \\ \min(\, C[P-D[i]] + 1 \,), \text{ if } P > 0 \\ D[1] <= D[i] <= P \end{cases}$$

- Fill the C[P] and S[P] tables

TIME COMPLEXITY:

Time Complexity = O(m*n), where m = Total taka & n = Number of coin

EXAMPLE:

Set Of Coins, D = {9,6,5,1}
Make Change Of Taka, n = 11

1. 11 / 9 = 1
2. 11 % 9 = 2
3. 2 / 6 = 0
4. 2 % 6 = 2
5. 2 / 5 = 0
6. 2 % 5 = 2
7. 2 / 1 = 2
8. 2 % 1 = 0

TAKEN COIN :
9(1) & 1(2)

EXPLANATION OF COIN CHANGE (GREEDY):

- If D is unsorted, then sort D in descending order.
- Find the taken coin and their minimum number using the following algorithm

```
for(D[1]  to D[n] )
{
     result = n/D[i];
     n = n (mod) D[i];
     if(result =  0)
          print result and D[i]
}
```

- Repeat for each coin

TIME COMPLEXITY:

Time complexity = O(n), where n = Number of coin

DP

- Set Of Coins, D = {1,5,6,9}
- Make Change Of Taka, n = 11

- <u>Result :</u>
  Minimum Number of coin : 2.
  The coins : 5, 6.

- Set Of Coins, D = {1,5,8,10}
- Make Change Of Taka, n = 15

- Result :
  Minimum Number of coin : 2.
  The coins : 10, 5.

GREEDY

- Set Of Coins, D = {9,6,5,1}
- Make Change Of Taka, n = 11

- <u>Result :</u>
  Minimum Number of coin : 3.
  The coins : 9(1), 1(2).

- Set Of Coins, D = {10,8,5,1}
- Make Change Of Taka, n = 15

- Result :
  Minimum Number of coin : 2.
  The coins : 10(1), 5(1).

THANK YOU

ANY QUESTIONS ??