COIN CHANGE PROBLEM

(An application of dynamic programming)

DP is doing brute force smartly.

Problem statement:

- You are given coins of different denominations and a total amount of money *amount*. Compute the fewest number of coins that you need to make up that amount. If that amount of money cannot be made up by any combination of the coins, return -1.
- *NOTE: you have infinite no of coins of each denominations mentioned in the problem.*


I have 1, 2, 5, 10 rupees coins
Let's pay using minimum no of coins
??


Give me 20 Rs for the juice


Here is ur
20 rs


Thanks

Different ways to solve this problem

- Brute force method
- Greedy approach
- Divide and conquer (coupled with dynamic programming)

Brute force method

- Given four denominations Rs 1, Rs 2, Rs 5, Rs 10
  assume we take a, b, c, d number of coins of 1, 2, 5, 10 denominations
  then n = a*1 + 2*b + 5*c + 10*d
  such that (a + b + c + d) is minimum.

- - at most there can be n coins each of Rs 1.
  - Try all combinations where a <= n, b <= n, c <= n and d <= n.
  - Choose all valid combinations that give n = a + 2*b + 5*c + 10*d
  - take the minimum valid (a+b+c+d) combination as the answer.

- Time complexity: ??
- Space complexity: ??

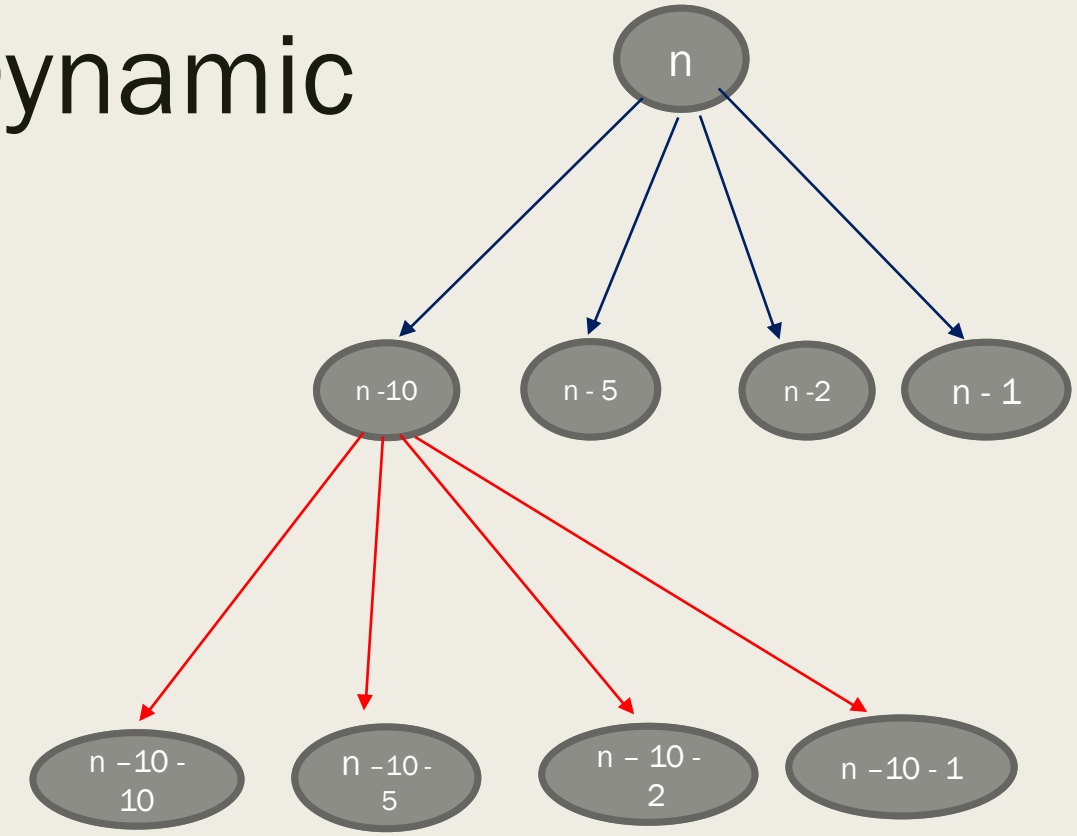
Greedy method

- If(n >= 10) take one 10 rupee coin and solve for (n – 10)
  else if(n >= 5) take one 5 rupee coin and solve for (n – 5)
  else if(n >= 2) tale one 2 rupee coin and solve for (n – 2)
  else we take n coins as 1 rupee coins

- For some combination of denominations this approach might always give optimal result. Consider a combination of 1, 5, 10 and 25 rupee coins. (Prove this)
- For some combination of denominations this approach may or may not give optimal answer. Consider n = 15, and denominations are (1, 7, 10).

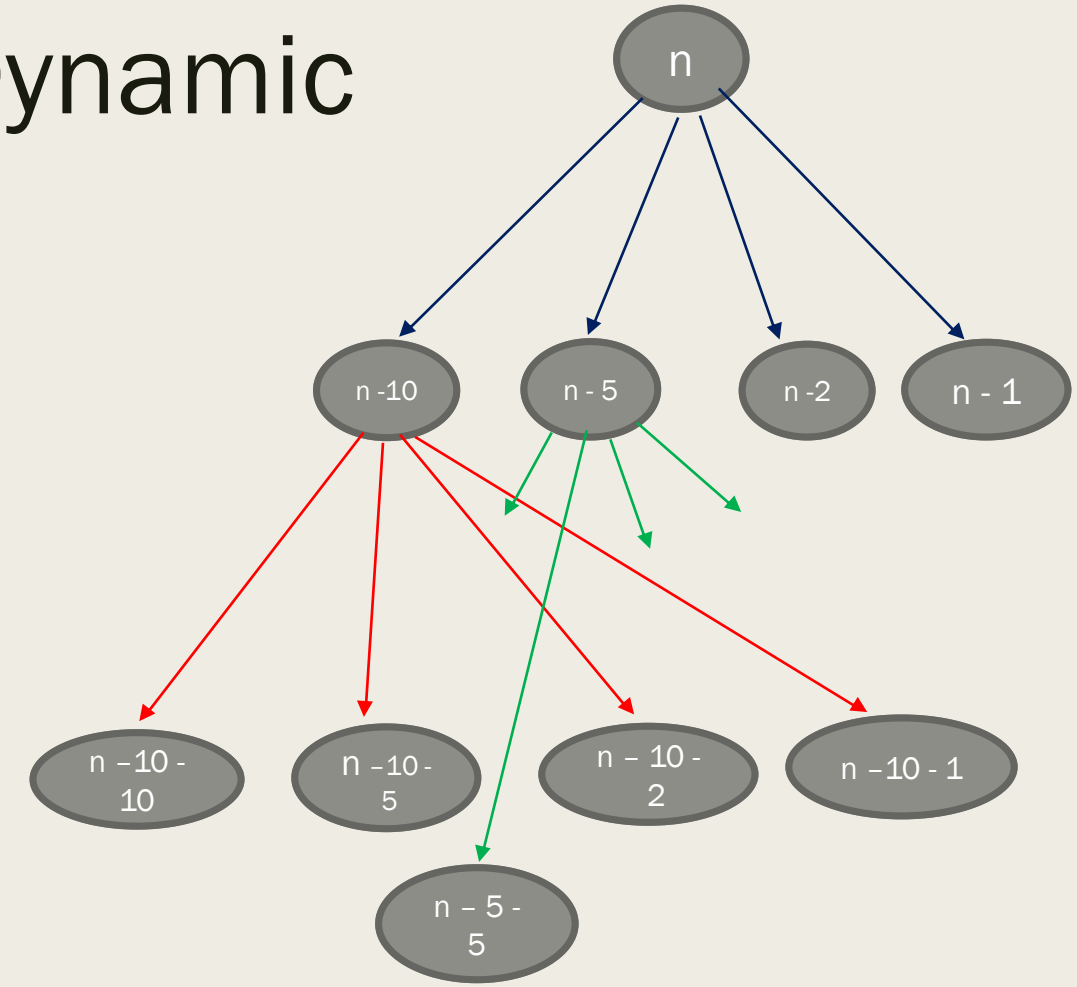
Divide and conquer (Dynamic Programming)

```
Int solve( int n ) {

    If( n == 0 ) {
        return 0;
    }
    ans1 = ans2 = ans3 = ans4 = INT_MAX;

    if ( n >= 10 )
        ans1 = 1 +  solve( n – 10);

    if (n >= 5)
        ans2 = 1 + solve( n – 5);

    if (n >= 2)
        ans3 = 1 + solve( n – 2);

    ans4 = 1 + solve( n - 1);

    return min ( ans1, ans2, ans3, ans4);
}
```



Divide and conquer (Dynamic Programming)

```
Int solve( int n ) {

    If( n == 0 ) {
        return 0;
    }

    if (dp[n] != -1 ) return dp[n];

    ans1 = ans2 = ans3 = ans4 = INT_MAX;

    if ( n >= 10 )
        ans1 = 1 + solve( n – 10);

    if (n >= 5)
        ans2 = 1 + solve( n – 5);

    if (n >= 2)
        ans3 = 1 + solve( n – 2);

    ans4 = 1 + solve( n - 1);

    return dp[n] = min ( ans1, ans2, ans3, ans4);
}
```



THANK YOU

someone@example.com