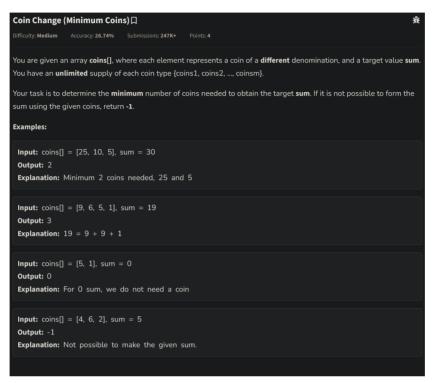
Coin Changei-

similar to 0-1 knapsack Problem (inclusion exclusion principle)



As we know in 0-1 Knapsack problem, only single unit was given. But here, we have unlimited supply of items.

Britistor a parameters defined:

(i, currsum)

Transition 1- am = INT_MAX if (coins(i) <= currsum) & if (fm(i-1, currsum - coin(i)) != INT_MAX)) ans=1+x; } ans=min(am, fn(i-1, curr sum)
yetum ans;

Base Case :-

if (cur Sum == 0) return 0; if(i(o) return INT-MAT;

```
class solution {
  public:
    int mincoins(vector<int> &coins, int sum) {
        // code here
        int n-coins.size();
        int ans-help(coins, n-1, sum);
        return ans==INI_MMX?-1:ans;
    }
  int help(vector<int> &coins, int i, int sum){
        if(sum=-0)return 0;
        if(sum-e)return 1NI_MAX;
        int ans=INI_MAX;
        int ans=INI_MAX;
        if (sum>-coins[i]){
            if x-help(coins, i, sum-coins[i]);
            if(x|=INI_MAX)ans-1:x;
        }
        ans-min(ans, help(coins, i-1, sum));
        return ans;
    }
};
```

T. C. O (2") S. C. O (n) + recordion stack

Optimize & Solution :- Using Recursion + Memoization:

T.C. O(size of coins(1) (sum)

S.C. O(size of coins(1) (sum) — dp array