Date : / /

Minimum Coins

f(ind, T) ⇒ Minimum no. of coins needed to make T using coins from 0 to ind.

2 choices

take → Use this coin

notTake → skip this coin, move to previous index

Since coins are infinite, we stay at same index

```
f(ind, T) {
  if (ind == 0) {
    if (T % a[i] == 0) return T/a[i];
    else return 1e9;
  }
  int notTake = 0 + f(ind-1, T);
  int take = INT_MAX;
  if (coins[ind] <= T) take = 1 + f(ind, T - coins[ind])
  return min(take, notTake);
}
```

```
if (T % a[i] == 0) return T/a[i];
```

If we only have coin coins[0].

If T is divisible by coin → we can make T using only this coin

If not divisible → impossible ∴ return 1e9 ⇒ as infinity

Date : / /

~~Memoization~~ Memoization

```
f(ind, T, dp) {
  if(ind == 0){
  if(T % nums[0] == 0) return T/nums[0];
  return 1e9;
  }
  if(dp[ind][T] != -1) return dp[ind][T];
  int notTake = 0 + f(ind-1, T, nums, dp);
  int take = INT_MAX;
  if(nums[ind] <= T) {
    take = 1 + f(ind, T-nums[ind], dp);
  }
  return dp[ind][T] = min(take, notTake);
```

Date : / /

Tabulation

```
int[][] dp = new int[n][target+1];
for(int T=0; T<=target; T++){
  if(T%nums[0]==0)
      dp[0][T] = T/nums[0];
else
  dp[0][T] = INF;
}
for(int ind=1; ind<n; ind++){
  for(int T=0; T<=target; T++){
    int notTake = dp[ind-1][T];
    int take = Integer.MAX_VALUE;
    if(nums[ind]<=T){
        take = 1+dp[ind][T-nums[ind]];
    }
    dp[ind][T] = Math.min(take, notTake);
  }
}
int ans = dp[n-1][target];
if(ans >= INF) return -1;
return ans;
}
```

Date : / /

Space Optimize

~~int [] prev = new int[];~~
int [] prev = new int[target+1];
int [] curr = new int[target+1];
for(int T=0; T<=target; T++){
if(T % nums[0] == 0)
prev[T] = T/nums[0];
else
prev[T] = INF;
}
for(int ind=1; ind<n; ind++){
for(int T=0; T<=target; T++){
int notTake = prev[T];
int take = INF;
if(nums[ind] <= T){
take = 1+cur[T-nums[ind]];
}
cur[T] = Math.min(take, notTake);
}
prev = cur.clone(); //or System.arraycopy(cur, 0, prev, 0, target+1);
Arrays.fill(curr, 0); }
int ans = prev[target];
if(ans >= INF) return -1;
return ans; }