Coin Change Problem Using Dynamic Programming

Making Change Problem – What is it ?

- Making Change problem is to find change for a given amount using a minimum number of coins from a set of denominations.

- Explanation: If we are given a set of denominations D = {d_0, d_1, d_2, …, d_n} and if we want to change for some amount N, many combinations are possible. Suppose {d_1, d_2, d_5, d_8}, {d_0, d_2, d_4}, {d_0, d_5, d_7} all feasible solutions.

- The aim of making a change is to find a solution with a minimum number of coins / denominations. Clearly, this is an optimization problem.

- This problem can also be solved by using a greedy algorithm. However, greedy does not ensure the minimum number of denominations.

How to Solve Making Change using Dynamic Programming?

- Conventional / greedy approach selects largest denomination first which is not greater than remaining amount. On selecting denomination of size d_i in every step, problem size keeps reducing by amount d_i.

- If current denomination is not possible to select then select second largest denomination and so on. Continue this process until solution is found.

Mathematical Formulation

- Sort all the denomination and start scanning from largest to smallest denomination. In every iteration i, if current denomination d_i is acceptable, then 1 coin is added in solution and total amount is reduced by amount d_i. Hence,
$$C[i, j] = 1 + (c\,[i, j - d_i])$$

- If current denomination is larger than current problem size, then we have to skip the denomination and stick with previously calculated solution. Hence,
$$C[i, j] = C[i - 1, j]$$

- Our objective is to find minimum number of coins, so at each step, we have to stick with choice which returns minimum number of coin. Mathematically, we formulate the problem as,
$$C[i, j] = \min \{C[i - 1, j]\ ,\ 1 + C[i, j - d_i]\}$$

Where,

d_i = i^{th} denomination ,$1 \leq i \leq n$
j = Size of sub problem

C[1…..n, 0…..N] = Size of the problem

C[n, N] = Solution of the problem

n = Number of denomination

N = Amount for which change is required By combining all three cases, we have

$$C[i, j] = \begin{cases} 1 + C[1, j - d_1], & \text{, if } i = j \\ C[i - 1, j], & \text{if } j < d_i \\ \min(C[i - 1, j], 1 + C[i, j - d_i]), & \text{otherwise} \end{cases}$$

Note: If no combination of coins sums to amount j, then C[i, j] = ∞

Example

Example: Consider an instance of a problem with coins 1, 4 and 6 units. Illustrate its solutions using dynamic programming approach involving a payment of 8 units or less.
Solution:
Optimal substructure for make a change problem,

$$C[i, j] = \begin{cases} 1 + C[1, j - d_1], & \text{, if } i = j \\ C[i - 1, j], & \text{if } j < d_i \\ \min(C[i - 1, j], 1 + C[i, j - d_i]), & \text{otherwise} \end{cases}$$

Let us find the optimal solution for D = {d_1, d_2, d_3} = {1, 4, 6} and N = 8. Initialize the table C [i, 0] = 0 and C [0, j] = 0. These indicate we don't have any denomination and size of problem is zero respectively. So solution is trivial.

		j								
		0	1	2	3	4	5	6	7	8
i=0		0	0	0	0	0	0	0	0	0
i=1	d_1=1	0								
i=2	d_2=4	0								
i=3	d_3=6	0								

Filling first column, j = 1 :
C [1, 1] ⇒ i = 1, j = 1, $d_i = d_1 = 1$
As, i = 1,

C [i, j] = C [1, 1] = 1 + C [i, j – d_i]

$= 1 + C[1, 1 - 1] = 1 + C[1, 0] = 1$

C [2, 1] $\Rightarrow$ $i = 2$, $j = 1$, $d_i = d_2 = 4$
As, $j < d_2$,
$C[i, j] = C[2, 1] = C[i - 1, j] = C[1, 1] = 1$

C [3, 1] $\Rightarrow$ $i = 3$, $j = 1$, $d_i = d_3 = 6$
As, $j < d_2$,
$C[3, 1] = C[i - 1, j] = C[2, 1] = 1$

Filling second column, j = 2 :
C [1, 2] $\Rightarrow$ $i = 1$, $j = 2$, $d_i = d_1 = 1$
As, $i = 1$,

$C[i, j] = C[1, 2] = 1 + C[1, j - d_i]$
$= 1 + C[1, 2 - 1] = 1 + C[1, 1] = 1 + 1 = 2$

C [2, 2] $\Rightarrow$ $i = 2$, $j = 2$, $d_i = d_2 = 4$
As, $j < d_2$,
$C[i, j] = C[2, 2] = C[i - 1, j] = C[1, 2] = 2$

C [3, 2] $\Rightarrow$ $i = 3$, $j = 2$, $d_i = d_3 = 6$
As, $j < d_2$,
$C[i, j] = C[3, 2] = C[i - 1, j] = C[2, 2] = 2$

Filling third column, j = 3 :
C [1, 3] $\Rightarrow$ $i = 1$, $j = 3$, $d_i = d_1 = 1$
As, $i = 1$,

$C[i, j] = 1 + C[i, j - d_1]$
$= 1 + C[1, 3 - 1] = 1 + C[1, 2] = 1 + 2 = 3$

C [2, 3] $\Rightarrow$ $i = 2$, $j = 3$, $d_i = d_2 = 4$
As, $j < d_i$,
$C[i, j] = C[2, 3] = C[i - 1, j] = C[1, 3] = 3$

C [3, 3] $\Rightarrow$ $i = 3$, $j = 3$, $d_i = d_3 = 6$
As, $j < d_i$,
$C[i, j] = C[i - 1, j] = C[3, 3] = C[2, 3] = 3$

Filling fourth column, j = 4 :
C [1, 4] $\Rightarrow$ $i = 1$, $j = 4$, $d_i = d_1 = 1$
As, $i = 1$,

$C[i, j] = 1 + C[i, j - d_i]$
$C[1, 4] = 1 + C[1, 4 - 1] = 1 + C[1, 3] = 1 + 3 = 4$

C [2, 4] $\Rightarrow$ $i = 2$, $j = 4$, $d_i = d_2 = 4$
As $i \neq 1$ and $j \geq d_i$
$C[i, j] = \min \{ C[i - 1, j], 1 + C[i, j - d_i] \}$
$C[2, 4] = \min \{ C[1, 4], 1 + C[2, 4 - 4] \}$

= min { C [1, 4], 1 + C [2, 0] } = min {4, 1 + 0} = 1

C [3, 4] $\Rightarrow$ i = 3, j = 4, $d_i = d_3 = 6$
As $j < d_i$,
C [i, j] = C [i – 1, j]

C [3, 4] = C [2, 4] = 1

Filling fifth column, j = 5 :
C [1, 5] $\Rightarrow$ i = 1, j = 5, $d_i = d_1 = 1$
As, i = 1,

C [i, j] = 1 + C [i, j – d_i]
= 1 + C [1, 5 – 1] = 1 + C [1, 4] = 1 + 4 = 5

C [2, 5] $\Rightarrow$ i = 2, j = 5, $d_i = d_2 = 4$
As $i \neq 1$ and $j \geq d_i$, So,
C [i, j] = min { C [i – 1, j], 1 + C [i, j – d_i] }
= min { C [1, 5], 1 + C [2, 1] } = min {5, 1 + 1} = 2

C [3, 5] $\Rightarrow$ i = 3, j = 5, $d_i = d_3 = 6$
As $j < d_i$,
C [i, j] = C [i – 1, j]

C [3, 5] = C [2, 5] = 2
Filling sixth column, j = 6 :
C [1, 6] $\Rightarrow$ i = 1, j = 6, $d_i = d_1 = 1$
As, i = 1,

C [i, j] = 1 + C [i, j – d_i]
= 1 + C [1, 5] = 1 + 5 = 6

C [2, 6] $\Rightarrow$ i = 2, j = 6, $d_i = d_2 = 4$
As $i \neq 1$ and $j \geq d_i$
C [i, j] = min { C [i – 1, j], 1 + C [i, j – d_i] }
= min { C [1, 6], 1 + C [2, 2] } = min {6, 1 + 2} = 3

C [3, 6] $\Rightarrow$ i = 3, j = 6, $d_i = d_3 = 6$
As $i \neq 1$ and $j \geq d_i$
C [i, j] = min { C [i – 1, j], 1 + C [i, j – d_i] }
= min { C [2, 6], 1 + C [3, 0] } = min {6, 1 + 0} = 1

Filling seventh column, j = 7 :
C [1, 7] $\Rightarrow$ i = 1, j = 7, $d_i = d_1 = 1$
As, i = 1,

C [i, j] = 1 + C [i, j – d_i]
= 1 + C [1, 6] = 1 + 6 = 7

C [2, 7] $\Rightarrow$ i = 2, j = 7, $d_i = d_2 = 4$
As $i \neq 1$ and $j \geq d_i$
C [i, j] = min { C [i – 1, j], 1 + C [i, j – d_i] }

= min { C [1, 7], 1 + C [2, 3] } = min {7, 1 + 3} = 4

C [3, 7] ⇒ i = 3, j = 7, $d_i = d_3 = 6$
As $i \neq 1$ and $j \geq d_i$
C [i, j] = min { C [i – 1, j], 1 + C [i, j – d_i] }
= min { C [2, 7], 1 + C [3, 17] } = min {4, 2} = 2

Filling eighth column, j = 8 :
C [1, 8] ⇒ i = 1, j = 8, $d_i = d_1 = 1$
As, i = 1,

C [i, j] = 1 + C [i, j – d_i]
= 1 + C [1, 7] = 1 + 7 = 8

C [2, 8] ⇒ i = 2, j = 8, $d_i = d_2 = 4$
As $i \neq 1$ and $j \geq d_i$
C [i, j] = min { C [i – 1, j], 1 + C [i, j – d_i] }
= min { C [1, 8], 1 + C [2, 4] } = min {8, 1 + 1} = 2

C [3, 8] ⇒ i = 3, j = 8, $d_i = d_3 = 6$
As $i \neq 1$ and $j \geq d_i$
C [i, j] = min { C [i – 1, j], 1 + C [i, j – d_i] }
= min { C [2, 8], 1 + C [3, 2] } = min {2, 1 + 2} = 2

- Finally, table would look like,

		j								
		0	1	2	3	4	5	6	7	8
i=0		0	0	0	0	0	0	0	0	0
i=1	d_1=1	0	1	2	3	4	5	6	7	8
i=2	d_2=4	0	1	2	3	1	2	3	4	2
i=3	d_3=6	0	1	2	3	1	2	1	2	**2**

Minimum number of coins required for change of amount 8 is 2. Let us trace the solution and find the denominations required for that.

TRACING SOLUTION:

For last cell in table, i and j would be 3 and 8 respectively.

C[n, N] = C[i, j] = C[3, 8] = 2

Step 1 : (i = 3, j = 8) : C[3, 8] = 2 and C[2, 8] = 2
Here, C[i, j] = C[i – 1, j], implies nothing was added in previous solution. Algorithm has propagated the solution of previous iteration in current iteration. So go to previous step by reducing i. So, i = i – 1 = 3 – 1 = 2

Step 2 : (i = 2, j = 8) : C[2, 8] = 2 and C[1, 8] = 8
Here, C[i, j] ≠ C[i – 1, j], implies denomination $d_i = d_2 = 4$ was added in previous solution. So add d_i in solution set and reduce problem size j by d_i.

Solution set S = {4}, j = j – d_i = 8 – 4 = 4.
Now we have, i = 2 and j = 4.

Step 3 : (i = 2, j = 4) : C[2, 4] = 1 and C[1, 4] = 4
Here, C[i, j] ^{1} C[i – 1, j], implies denomination d_i = d_2 = 4 was added in previous solution. So add d_i in solution set and reduce problem size j by d_i.
Solution set S = {4, 4}, j = j – d_i = 4 – 4 = 0.
Problem size j is zero. So stop. We need two coins of amount 4 for optimal change of amount 8.

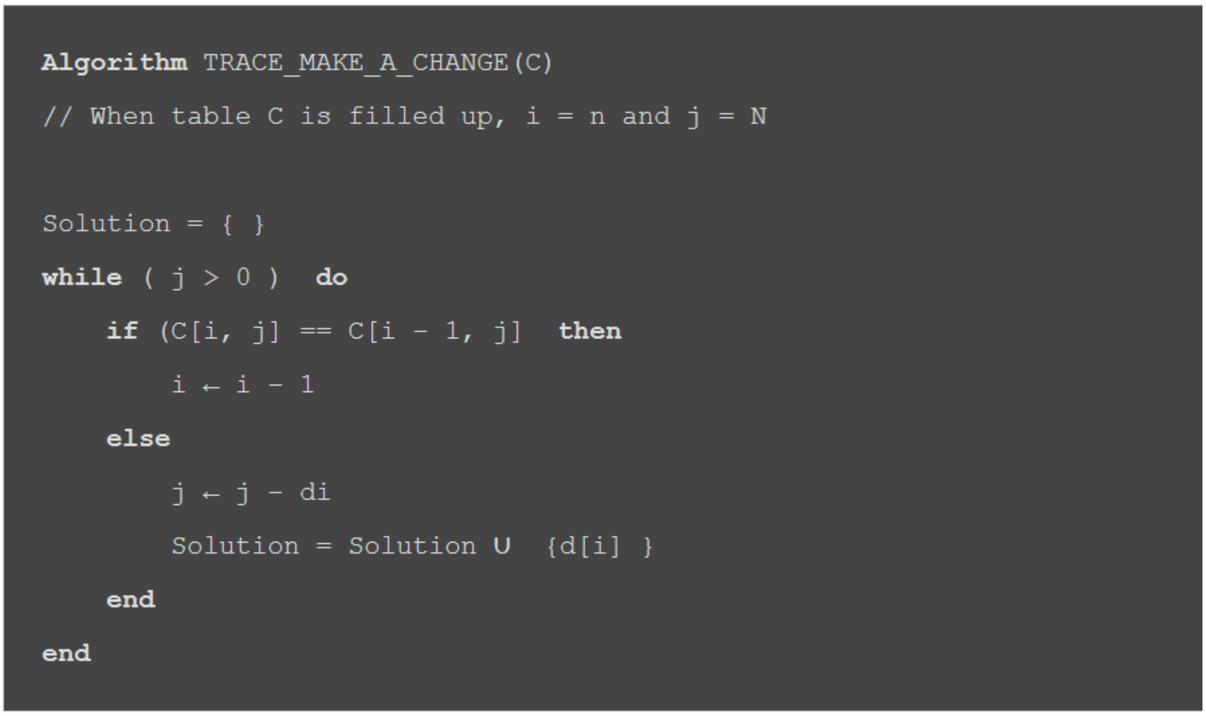
Algorithm for Making Change

```
Algorithm MAKE_A_CHANGE(d,N)
// d[1…n] = [d1,d2,…,dn] is array of n denominations
// C[1…n, 0…N]  is n x N array to hold the solution of sub problems
// N is the problem size, i.e. amount for which change is required

for i ← 1 to n do
   C[i, 0] ← 0
end

for i ← 1 to n do
   for j ← 1 to N do
      if i = = 1 ans j < d [i]  then
         C[i, j]  ←  ∞
      else if i == 1 then
         C[i, j]  ← 1 + C[1, j – d[1])
      else if j < d [i] then
         C[i, j]  ←  C[I – 1, j]
      else
         C[i, j]  ← min (C[i – 1, j] ,1 + C[i, j – d[i])
      end
   end
end

return C[n, N]
```

- Cell C[n, N] will only tell us about how many minimum number of coins are required to find change for N. It does not speak anything about which coins should be selected.

- We can find the required denomination by tracing table in backward direction. Algorithm to trace the solution is described below :

- We shall compare C[i, j] and C[i – 1, j]. If both are same it means we propagated the solution of previous iteration (i – 1) in current iteration i, which implies nothing was added in iteration i. Otherwise we should select denomination d_i and problem size j should be reduce by amount d_i and continue the process till problem size j is greater than zero.



```
Algorithm TRACE_MAKE_A_CHANGE(C)
// When table C is filled up, i = n and j = N

Solution = { }
while ( j > 0 )  do
    if (C[i, j] == C[i - 1, j]  then
        i ← i - 1
    else
        j ← j - di
        Solution = Solution U  {d[i] }
    end
end
```

Complexity Analysis

Dynamic programming finds optimal solution by constructing table of size n x N, where n is number of denominations and N is the amount for which we want change. This table can be filled up in O(n.N) time, same is the space complexity.