**Problem 1.**

A array to store number of coins

N is the sum

coin is array of coin value

Algorithm:

function mincoinnum(value n):

if n = 0:

return 0

x = INTMAX

for i = 1 to 3: //get coin value

if coin[i] <= n:

subval = mincoinnum(n - coin[i])

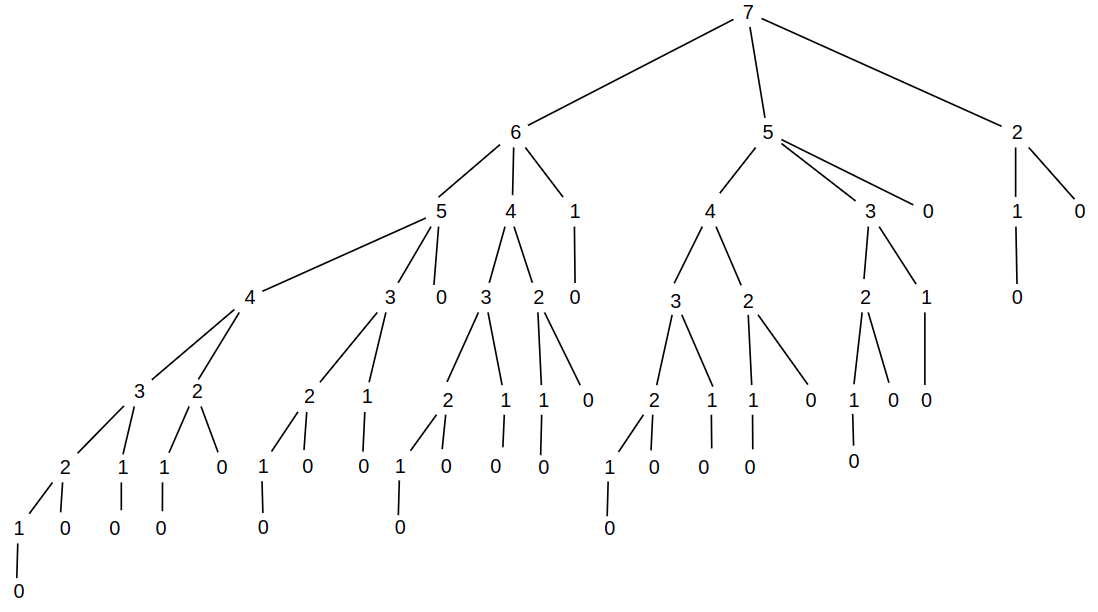
if subval < INTMAX:

x = min(x, 1+subval)

return x

call: mincoinnum(N)

**Problem 2.**



The lower bound is Omega(N)

The upper bound is Theta(N^2)

**Problem 3.**

A array to store number of coins

N is the sum

coin is array of coin value

Algorithm:

function mincoinnum(value n):

if n = 0:

A[0]=0

return 0

if A[n] != INTMAX

return A[n]

x = INTMAX

for i = 1 to 3: //get coin value

if coin[i] <= n:

subval = mincoinnum(n - coin[i])

if subval < INTMAX:

x = min(x, 1+subval)

//memoizing value  
A[n] = x

return x

The recursion loop is N with coin value 1, so

Time complextity: O(N)

**Problem 4.**

As we can see:

2 = 1+1 is the optimal solution with coin array: 1

5 = 2+2+1 is the optimal solution with coin array: 1,2

Consider N is the current sum number.

In order to find optimal solution for N, we need to find optimal solution for N-i with i=1,2,5

Easy to see for N>=5 with N-5 is the smallest number and it can be constructed by N-2-2-1 or N-2-1-1-1 or N-1-1-1-1-1 also. So N-5 is the optimal solution. In this case N-5 is also the next value of greedy algorithm.

For 2<=N<5, greedy algorithm will choose N-2 for the next value, it’s the optimal solution as well.

For N=1, The optimal solution is N-1

So, the greedy algorithm will always find the optimal solution with coins 1,2,5

**Problem 5.**

Consider N=19 is the current sum number.

Follow the greedy algorithm, the next value is: 12, 5, 4, 3, 2, 1, 0

So 19 = 7+7+1+1+1+1+1

But the optimal solution is 19=7+6+6

So the greedy algorithm not always give an optimal solution with coins 1,6,7