MINIMUM COIN CHANGE

COIN CHANGE PROBLEM

- A problem of finding a number of ways of making changes for a target amount, n, using a given set of denominations $C = \{ c_1, c_2, \ldots, c_d \}$

- For instance, the US coin system is {1, 5, 10, 25, 50, 100}

EXAMPLE

- n = 4
- C = { 1, 2, 3 }
- Possible Solutions?
  - {1, 1, 1, 1}
  - {1, 1, 2}
  - {2, 2}
  - {1, 3}

MINIMUM COIN CHANGE

- Given
  - a particular amount of change n
  - a set of denominations $C = \{ c_1, c_2, \ldots, c_d \}$
- Goal:
  - Minimize the number of coins returned for a particular (given) quantity of change.

EXAMPLE 1

- n = 30
- C = { 1, 5, 10, 25, 50 }

- The minimum number of coins return is 2 coins.
- How do we obtain this number?
  - 25 + 5

EXAMPLE 2

- n = 67
- C = { 1, 5, 10, 25 }

- What is the minimum number of coins returned?

- Answer: 6 coins (25 + 25 + 10 + 5 + 1 + 1)

EXAMPLE 3

- n = 17
- C = { 1, 2, 3, 4 }

- What is the minimum number of coins returned?

- Answer: 5 coins
  - (4 + 4 + 4 + 4 + 1)
  - (4 + 4 + 3 + 3 + 3)

GREEDY ALGORITHM?

- Greedy solution does not always give an optimal solution.

- Consider the following example.
  - n = 7
  - C = { 1, 3, 4, 5 }

- What is the minimum number of coins returned?
- Answer: 2 coins (4 + 3)

Greedy Solution = 3 coins
(5 + 1 + 1)

DEMO EXAMPLE

- n = 30
- C = { 1, 5, 10, 25, 50 }

DIVIDE AND CONQUER?

- Choose the smallest number of coins out of the following:
  - 1 + MinChange(29)
  - 1 + MinChange(25)
  - 1 + MinChange(20)
  - 1 + MinChange(5)

Note that
n = 30
C = { 1, 5, 10, 25, 50 }

- What seems to be the problem here?
  - What are the values of MinChange(29), MinChange(25), MinChange(20), MinChange(5) ?

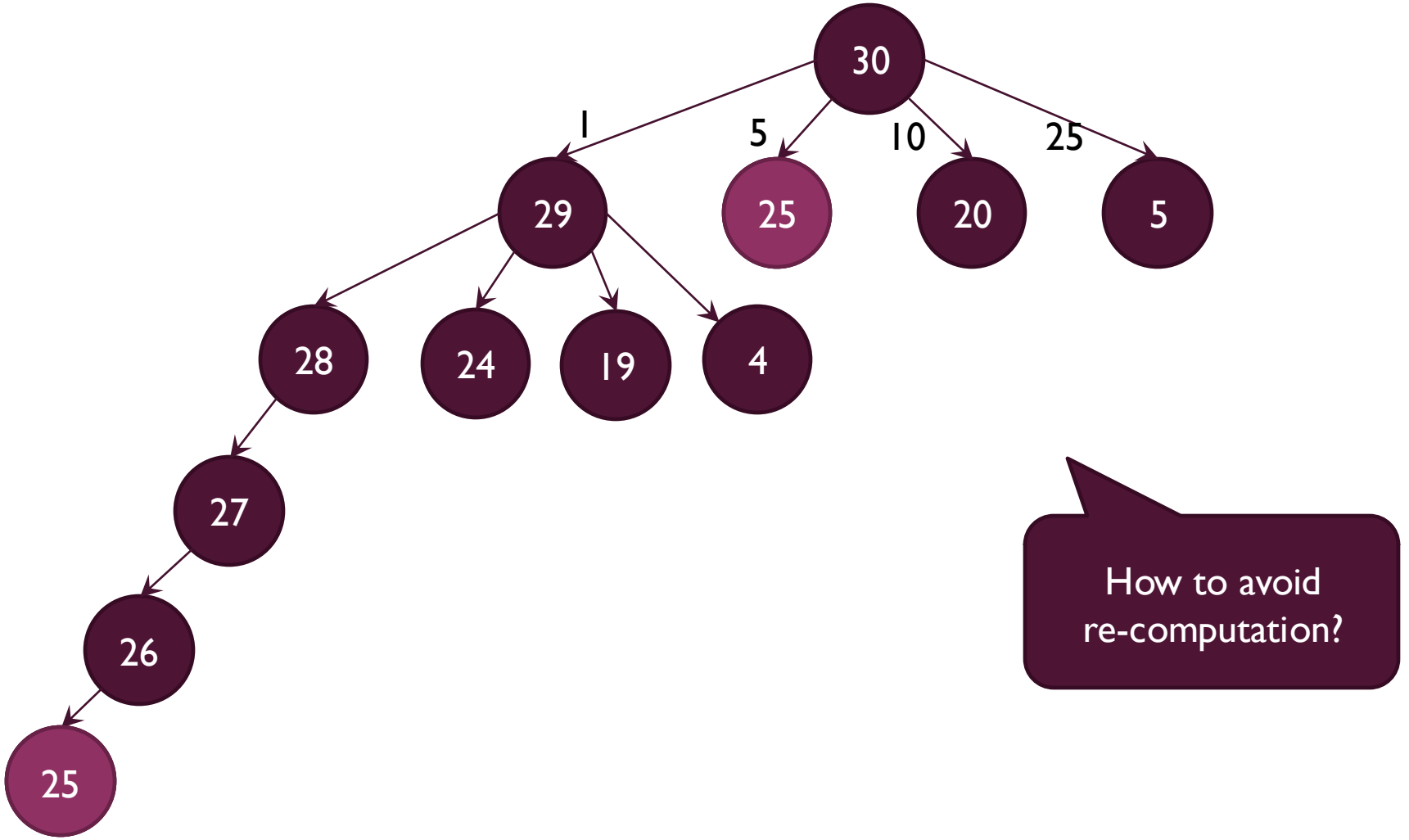
RECURSIVE ALGORITHM?

```
MinChange(n, C)
    if n = 0
        return 0
    v = ∞
    for each c in C ≤ n
        v = min { MinChange(n-c, C) + 1, v }
    return v
```

What do you think about this algorithm?

It recalculates the optimal coin combination for a given amount repeatedly!!

WHY IS IT INEFFICIENT?



How to avoid re-computation?

SAVE THE INTERMEDIATE RESULTS

MinChange(n, C)

```
if minCoin[n] not empty
    return minCoin[n]
if n = 0
    return 0
for each c in C ≤ n
    v = min {MinChange(n-c) + 1, v}
minCoin[n] = v
return v
```

EXERCISE 1: MINIMUM COIN CHANGE

Sample Input	Sample Output
1 5 10 25 50 30	2
1 5 10 25 67	6
1 2 3 4 17	5
1 3 4 5 7	2

EXERCISE 2: MINIMUM COIN CHANGE

Input	Sample Output
1 5 10 25 50 30	2 5 25
1 5 10 25 67	6 1 1 5 10 25 25
1 2 3 4 17	5 1 4 4 4 4
1 3 4 5 7	2 3 4