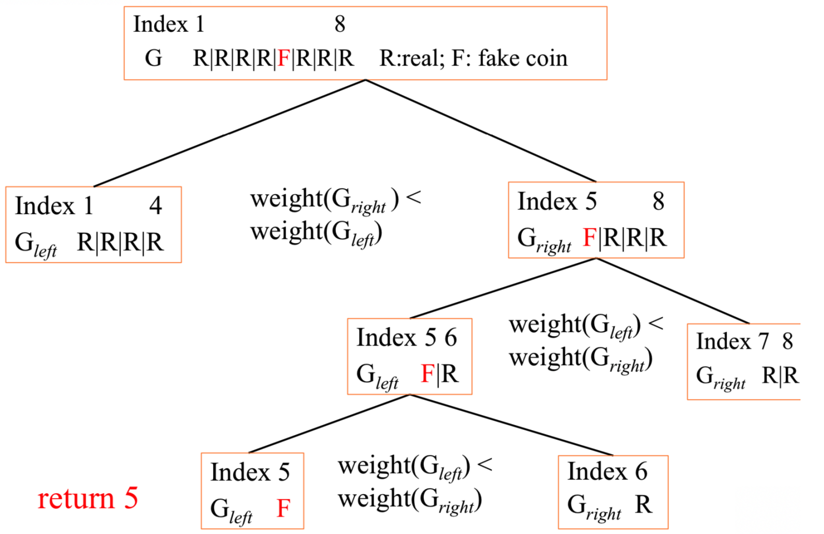
Solution 2: Divide and Conquer (Continue) and Greedy Algorithm

1 Answer 1



To find a fake coin within n coins, we call *FindFakeCoin*(G, 1, n)

FindFakeCoin(G, n_L, n_R)

Input: an array of coins $G[n_L..n_R]$; **Output**: the index of the fake coin

```
n_G = n_R – n_L +1
if (n_G == 1)                                   Base case
    return n_L
if (n_G is even)
    let mid = (n_L + n_R – 1) / 2 ;   last = n_R      Divide
else
    let mid = (n_L + n_R – 2) / 2 ;   last = n_R – 1
let A = G[n_L..mid] ;   B = G[mid+1..last]
Compare the weights of A and B
if ( A is lighter than B )
    return FindFakeCoin (G, n_L, mid )               Conquer
else if ( B is lighter than A )
    return FindFakeCoin (G , mid+1, last)
else                                                 Combine
    return n_R
```

- Another algorithm
  - Divide the coins into **three** groups (*A*, *B*, *C*) such that the first two groups (*A*, *B*) have the same number of coins
  - Compare the weights of A and B
  - 3 cases to consider
    - If *A* is lighter than *B*, then *A* …
    - If *B* is lighter than *A*, then *B* …
    - If *A* and *B* have the same weight, then *C* …
  - This algorithm has the same time complexity with *FindFakeCoin* (G, n_L, n_R) :

$$T(n) = O(\log n)$$

$$log_2 n \ vs \ log_3 n$$

2 Answer 2

a) The main difference is: the question in tutorial contains one assumption which is $v_{i+1} = 2v_i$. However, this question does not contain this assumption.

b) No. We provide one counter example. Given k=3, $v_1 = 1$, $v_2 = 6$, $v_3 = 10$ and n=12, using the greedy choice in the tutorial, we choose v_3 one time and v_1 two times. **Totally, it requires three times (10+1+1=12)**. However, **the optimal solution is to choose v_2 two times (6+6=12)**.

3 Answer 3

a) [0,2] and [0,3] can both cover the integer 0, [0,3] is the best.

b) Each time our greedy choice is the interval ls_j which can cover the current position x. and it contains the largest R_j.

The algorithm is specified as follows:

- Minimum_Coverage(S, M)

```
create an empty set C
for x ← 0 to M do
    for each line segment ls_j ∈ S do
        if L_j ≤ x ≤ R_j  and R_j is largest
            ls_g ← ls_j
    S ← S − {ls_g}
    C ← C ∪ {ls_g}
    x ← R_g + 1
return C
```

What if x cannot be covered during searching?
When to return NIL?

c) Let the greedy choice be ls_g which covers x

Let the optimal set be C* and it covers x by interval l_o
Two possible cases are shown as follows.

<u>Case 1: $ls_g = l_o$</u>

C* contains the greedy choice <u>ls_g</u>.

<u>Case 2: $ls_g \neq l_o$</u>

By the greedy choice property, $l_o.R_o < ls_g.R_g$

The optimal set C* must fill in the gap between $l_o.R_o+1$ and $ls_g.R_g$ in order to fulfill the coverage requirement.

If this new interval cannot cover $ls_g.R_g+1$, we can eliminate l_o and this new interval in C* and replace it by ls_g. Thus, the size of the set is reduced. Therefore, it leads to contradiction. (Not optimal)

(R_o + new_interval <= R_g)

If this new interval can cover $ls_g.R_g+1$, the proof goes back to Case 1 and Case 2 directly. (l_o and l_g are both feasible and optimal solutions)

Thus, it shows that the optimal solution set C* **either contains the greedy choice or leads to contradiction**. Therefore, this greedy strategy guarantees the optimal solution.