**Problem description**

If you have 50 boxes that contains 50 pieces of metal all of the same known weight.

one of these boxes contains fake metal pieces that weigh 1 kilogram less than the pieces in the rest of the boxes.

You can use a digital scale only once to find this fake box.

Design a brute force algorithm to solve this problem.

**Detailed assumptions**

1) If we assume that each real box weigh **[ a kilogram ]** , then fake box weigh **[ a-1 kilograms ]**

2) Then each real metal piece in real boxes weight

3) Then each real metal piece in real boxes weight

**Detailed solution including the pseudo-code and the steps of solution**

**First step** : we give each box a unique number from 1 to 50

**second step** : we take a number of metal pieces from each box depending on box's number

Example: 1 piece of box 1 and 2 pieces of box 2 and so on until we take 50 pieces of box 50

**Third step** : we weigh the gathered pieces and store the result

**Fourth step** : by computing 50 probabilities of expected results and storing them

**Fifth step** : comparing the result the digital scale with computed results, we find the fake box

**Pseudo code**

findFakeBox(array boxes,float digitalScaleReading ,double realMetalPieceWeigt, double fakeMetalPieceWeigt)

// boxes: array of number of pieces taken from each box

float sum = 0;

arrayOfProbablities[50]

for i <- 1 to 50

sum = 0;

for j <- 0 to 49

if j+1 == i

sum += boxes[j] \* fakeMetalPieceWeigt;

else

sum += boxes[j] \* realMetalPieceWeigt;

endfor

arrayOfProbablities[i-1] = sum;

endfor

for i <- 0 to 49

if digitalScaleReading == arrayOfProbablities[i]

return i+1;

endfor

**Complexity analysis for the algorithm**

**T(n) = Σ*1*≤*i*≤*50 (*Σ*0*≤*j*≤*49 1*) *+* Σ*0*≤*i*≤*49 = O(1)***

**Sample output of the solution**

A black background with white text

Description automatically generated

A black screen with white text

Description automatically generated

**A comparison between your algorithm and at least one other technique**

|  |  |  |
| --- | --- | --- |
|  | **Brute Force** | **Divide & Conquer** |
| **Time Complexity** | O(n2) | O(n2) |
| **Space Complexity** | O(1) | O(1) |
| **Readability** | Easy | Easy |

**Conclusion**

Brute force is an efficient solution for this problem