COMPUTER Algorithm Term project “Fake coin problem”

지옥에서 벗어나기

[김현규, 이근희, 이완해, 정진용, 정효찬]

Overview

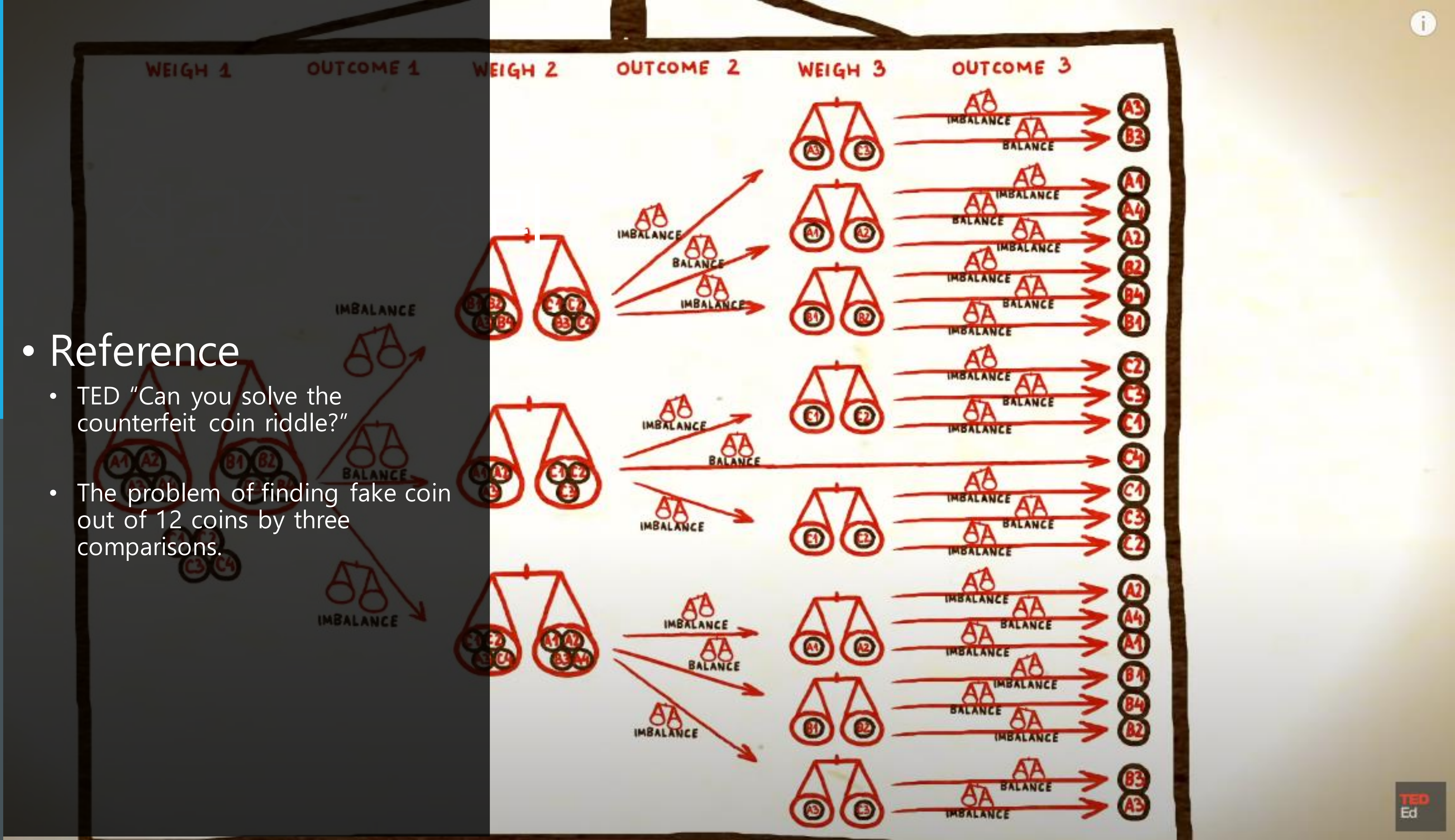
- ◆Role Allocation
- ◆Reference
- ◆Algorithm details
- ◆Performance

ROLE ALLOCATION

• 김현규	◆ Compare Algorithm Development
• 이근희	◆ Compare Algorithm Development
• 이완해	◆ Static Finding Algorithm Development
• 정진용	◆ Documenting & Statistical Algorithm Analysis
• 정효찬	◆ Documenting & Visualizing Algorithm

Reference

- TED
  - “Can you solve the counterfeit coin riddle?”
- On two problems of information theory
  - Erdgs, Paul and Alfrjd Rgnyi. (2001).
- Counting Counterfeit Coins: A New Coin Weighing Problem
  - Diaco, Nicholas. (2016).

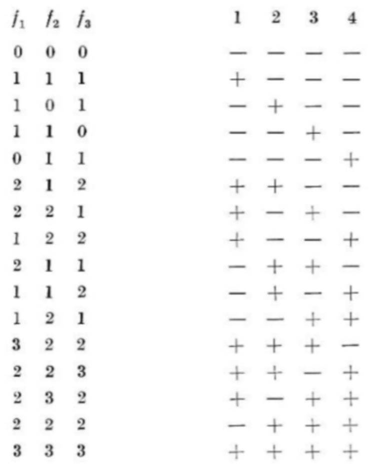


- Reference
  - TED "Can you solve the counterfeit coin riddle?"
  - The problem of finding fake coin out of 12 coins by three comparisons.

- Reference
  - On two problems of information theory by Paul Erdos and Alfred Renyi
  - Seeking counterfeit coins among N number of coins.
  - Scale show total weight of coins (not compare).

❖ N is number of coin
❖ A(n) is number of weighing

Unknown sequence of N digits divided into test sequences to identify counterfeit coins.



f_1	f_2	f_3	1	2	3	4
0	0	0	−	−	−	−
1	1	1	+	−	−	−
1	0	1	−	+	−	−
1	1	0	−	−	+	−
0	1	1	−	−	−	+
2	1	2	+	+	−	−
2	2	1	+	−	+	−
1	2	2	+	−	−	+
2	1	1	−	+	+	−
1	1	2	−	+	−	+
1	2	1	−	−	+	+
3	2	2	+	+	+	−
2	2	3	+	+	−	+
2	3	2	+	−	+	+
2	2	2	−	+	+	+
3	3	3	+	+	+	+

Ex) If there are coin [1,2,3,4]..
- F1 =1 , 2 , 3
- F2 =1 , 3 , 4
- F3 =1 , 2 , 4 coins

Upper bounds : $(1+\delta)\dfrac{n\log_2 9}{\log_2 n}$

Lower bounds : $\lim\limits_{n\to+\infty}\dfrac{A(n)\log_2 n}{n} \geq 2$

We do **not** apply this paper to our problem
because **it assumes that the specific weight of the coins is known.**

- Reference
  - Counting Counterfeit Coins : A New Coin Weighing Problem by Nicholas Diaco
  - This paper has compiled the papers on the Coin Weighing Problem

Assuming that it is related to ***log3(X)***,
this paper looked up the correlation in another paper.

But there were only similar alternatives,
no formal relation to *log3(X)* has yet been discerned.

“
With n specifically defined,
we created a decision tree associated with log3(X) !!
(See Static Finding)
”

We planned this in the last presentation.

With P	Worst case improvement Stable result
Regardless of P	Optimization
Hybrid	Adjusting the optimal method according to P

Improved !

Improved worst-case,
powerful average performance

Identifies 11 coins in 7 balance,
regardless of the probability!

Guess the probability
and select the function that matches it.

Algorithm details

- ◆Main
- ◆Compare with 3 coins
- ◆Compare with 5 coins
- ◆Static Finding

ALGORITHM DETAILS

- Main
- Compare with 3 coins
- Compare with 5 coins
- Static finding

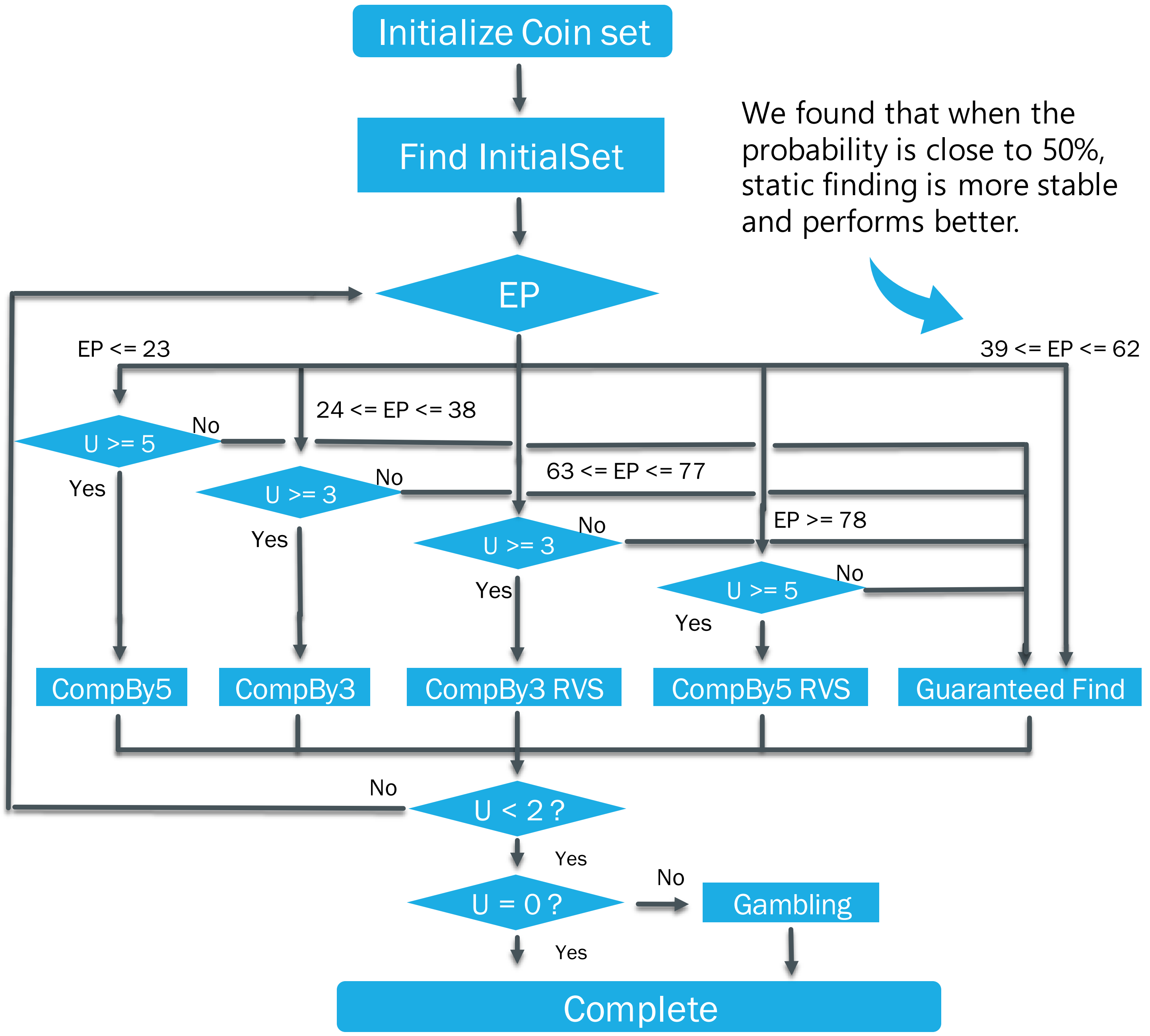
❖ ***U** is the number of Unknown coins*

❖ *EP is Estimate Probability*

(The number of fake coins found /

The number of all coins found.)

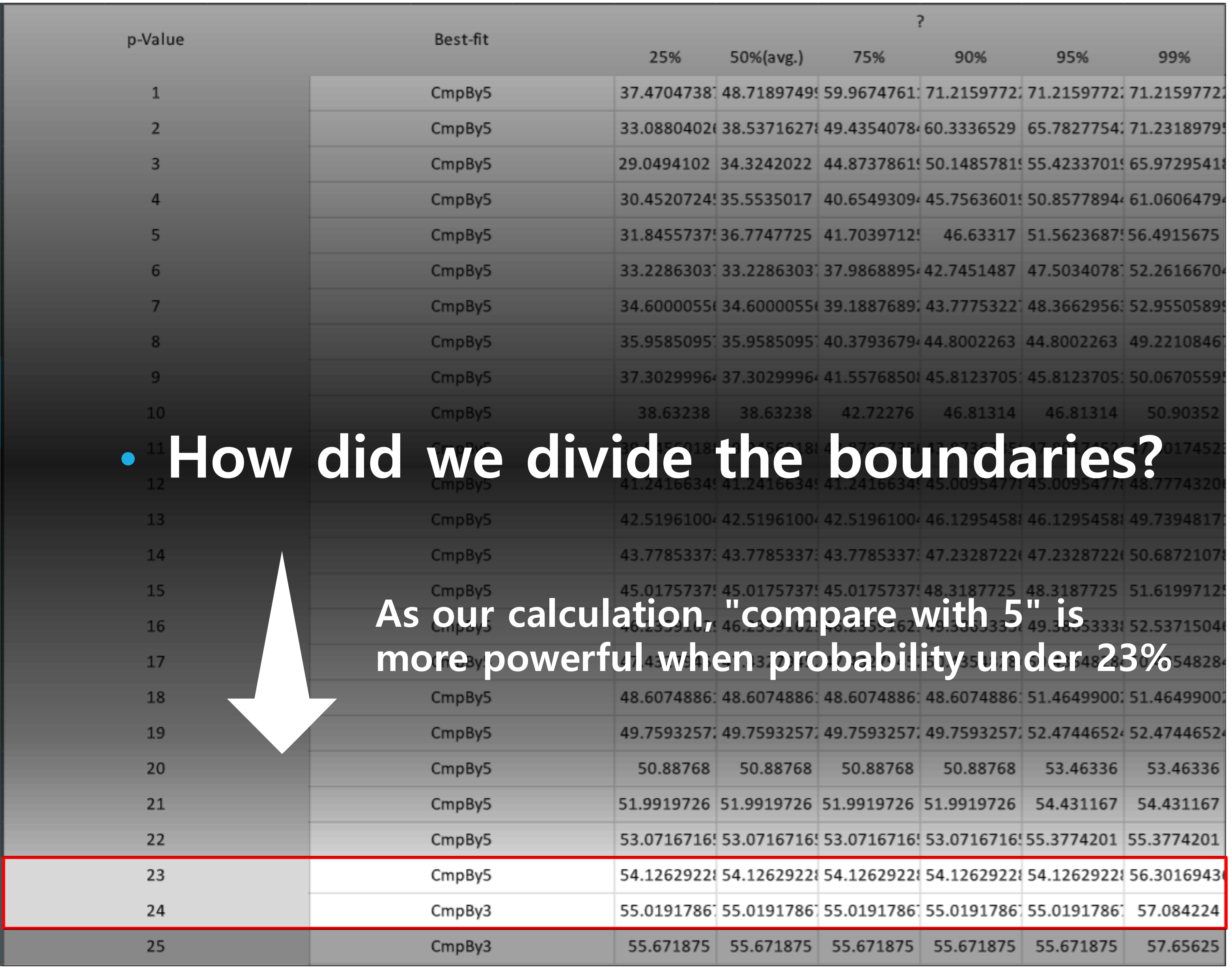
❖ The RVS function is a reversal of the standard coin from a real coin to a fake coin.



We found that when the probability is close to 50%, static finding is more stable and performs better.

ALGORITHM DETAILS

- Main
- Compare with 3 coins
- Compare with 5 coins
- Static finding



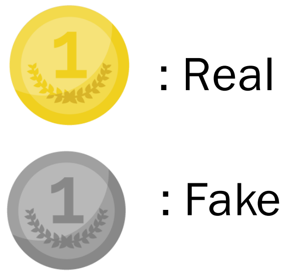
- **How did we divide the boundaries?**

As our calculation, "compare with 5" is more powerful when probability under 23%

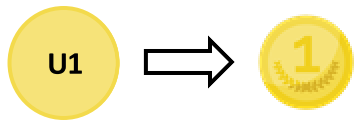
p-Value	Best-fit	?					
		25%	50%(avg.)	75%	90%	95%	99%
1	CmpBy5	37.4704738	48.7189749	59.9674761	71.2159772	71.2159772	71.2159772
2	CmpBy5	33.0880402	38.5371627	49.4354078	60.3336529	65.7827754	71.2318979
3	CmpBy5	29.0494102	34.3242022	44.8737861	50.1485781	55.4233701	65.9729541
4	CmpBy5	30.4520724	35.5535017	40.6549309	45.7563601	50.8577894	61.0606479
5	CmpBy5	31.8455737	36.7747725	41.7039712	46.63317	51.5623687	56.4915675
6	CmpBy5	33.2286303	33.2286303	37.9868895	42.7451487	47.5034078	52.2616670
7	CmpBy5	34.6000055	34.6000055	39.1887689	43.7775322	48.3662956	52.9550589
8	CmpBy5	35.9585095	35.9585095	40.3793679	44.8002263	44.8002263	49.2210846
9	CmpBy5	37.3029996	37.3029996	41.5576850	45.8123705	45.8123705	50.0670559
10	CmpBy5	38.63238	38.63238	42.72276	46.81314	46.81314	50.90352
11	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
12	CmpBy5	41.2416634	41.2416634	41.2416634	45.0095477	45.0095477	48.7774320
13	CmpBy5	42.5196100	42.5196100	42.5196100	46.1295458	46.1295458	49.7394817
14	CmpBy5	43.7785337	43.7785337	43.7785337	47.2328722	47.2328722	50.6872107
15	CmpBy5	45.0175737	45.0175737	45.0175737	48.3187725	48.3187725	51.6199712
16	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	49.3805333	52.5371504
17	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
18	CmpBy5	48.6074886	48.6074886	48.6074886	48.6074886	51.4649900	51.4649900
19	CmpBy5	49.7593257	49.7593257	49.7593257	49.7593257	52.4744652	52.4744652
20	CmpBy5	50.88768	50.88768	50.88768	50.88768	53.46336	53.46336
21	CmpBy5	51.9919726	51.9919726	51.9919726	51.9919726	54.431167	54.431167
22	CmpBy5	53.0716716	53.0716716	53.0716716	53.0716716	55.3774201	55.3774201
23	CmpBy5	54.1262922	54.1262922	54.1262922	54.1262922	54.1262922	56.3016943
24	CmpBy3	55.0191786	55.0191786	55.0191786	55.0191786	55.0191786	57.084224
25	CmpBy3	55.671875	55.671875	55.671875	55.671875	55.671875	57.65625

ALGORITHM DETAILS

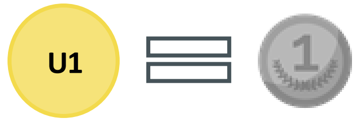
Gambling case 1



- Let's assume..



- If it's not correct answer, then..



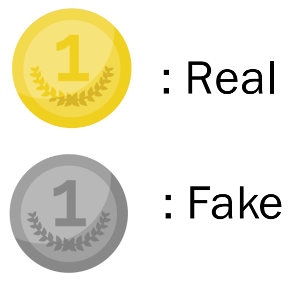
- If our hypothesis is true -> 1 time calling balance function
- If our hypothesis is false -> 2 times calling balance function

- Can reduce total call, same for worst case

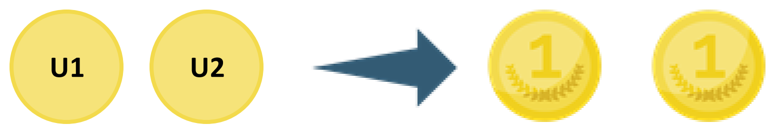
ALGORITHM DETAILS

- Main
- Compare with 3 coins
- Compare with 5 coins
- Static finding

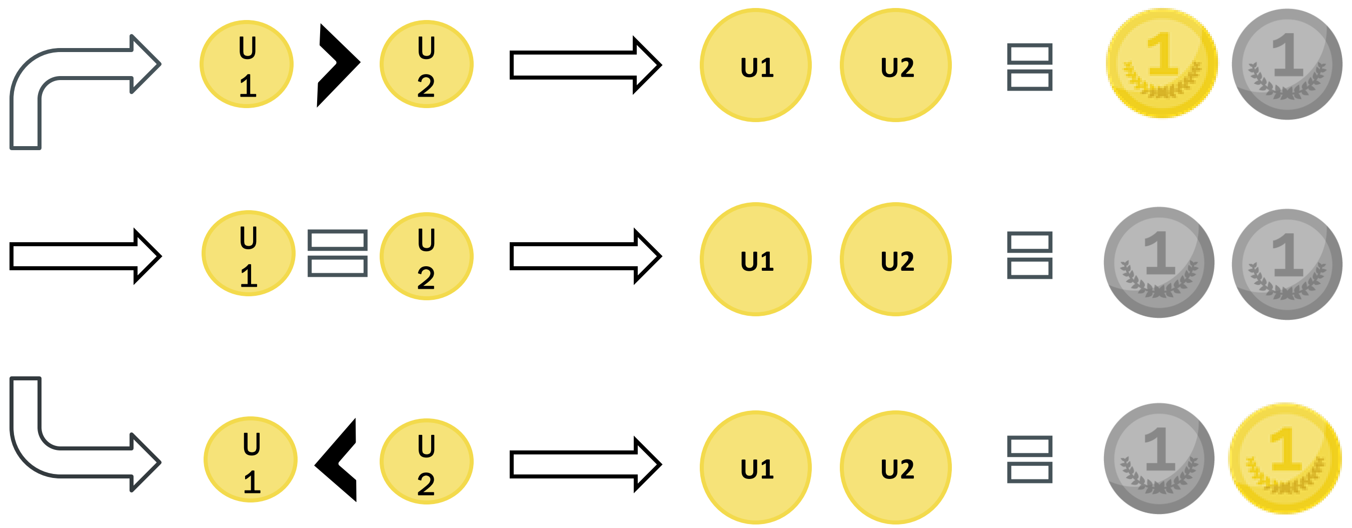
Gambling case 2



- Let's assume



- If it's correct, We can reduce 2 function call! – Yeah!! (Gamble success)
- If it's not correct answer, then..

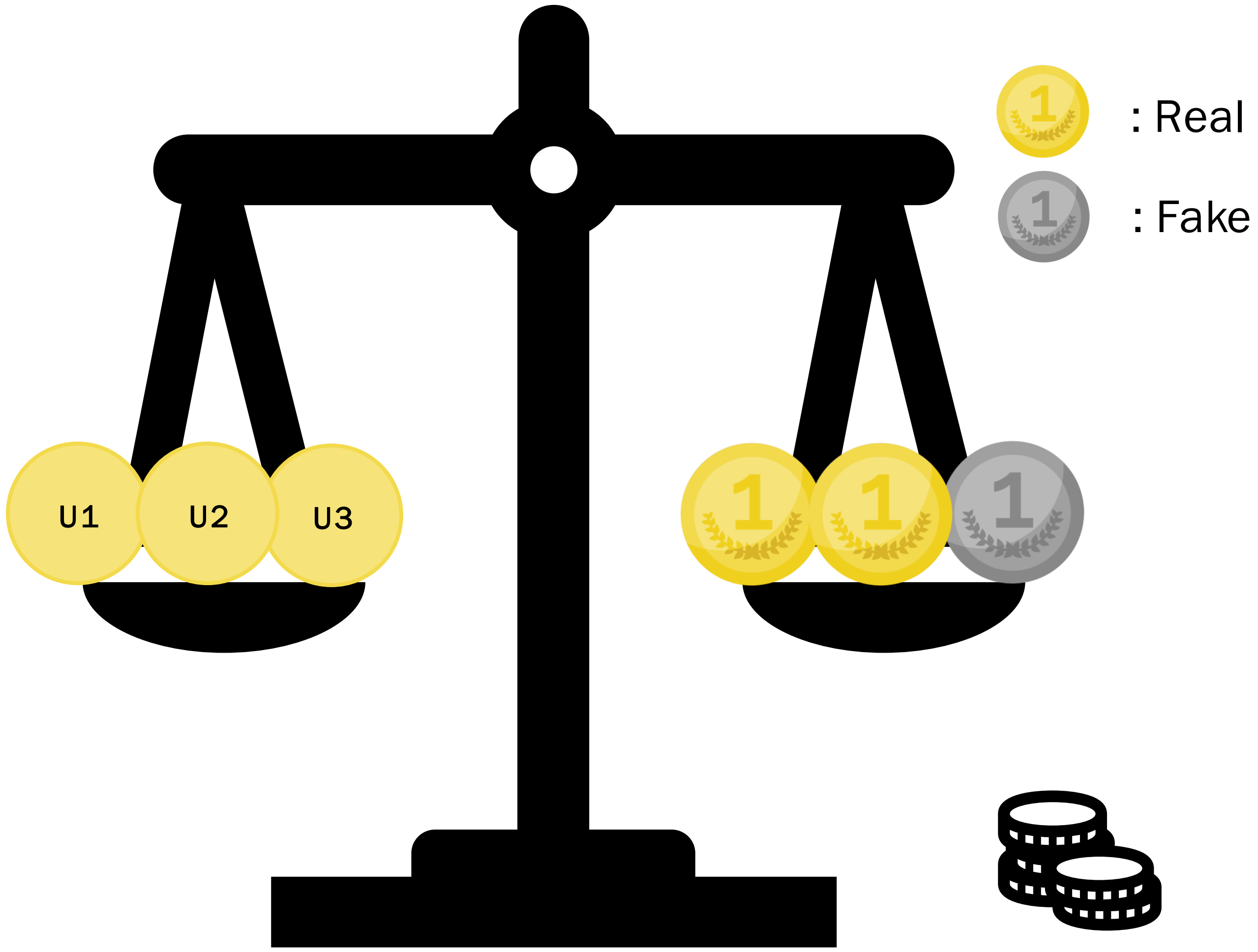


- IF U1 $\neq$ U2, Even the number of function calls increases. (Gamble Fail..)
- IF U1 $=$ U2, Same as not gambling (Well..? at least we didn't loss)

ALGORITHM DETAILS

- Main
- Compare with 3 coins
- Compare with 5 coins
- Static finding

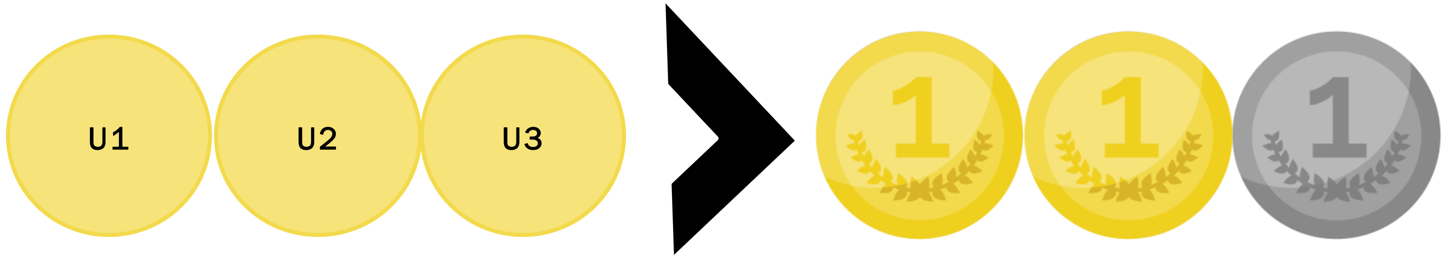
Compare with 2 real coins and 1 fake coin

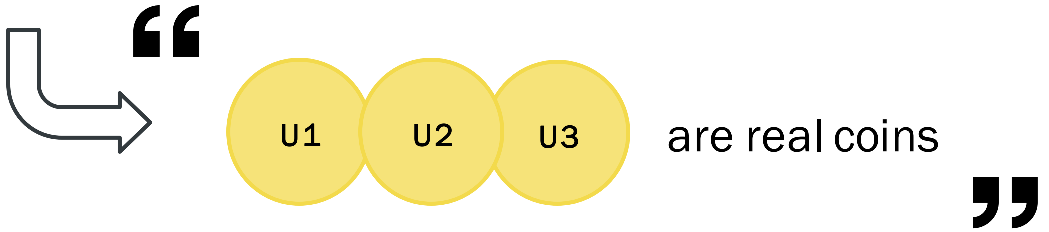


ALGORITHM DETAILS

- Main
- Compare with 3 coins
- Compare with 5 coins
- Static finding

In this case, there is less than 1 fake coin => No fake coin



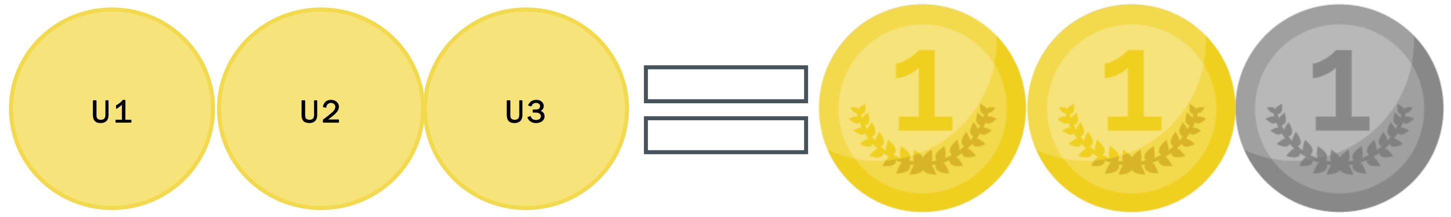


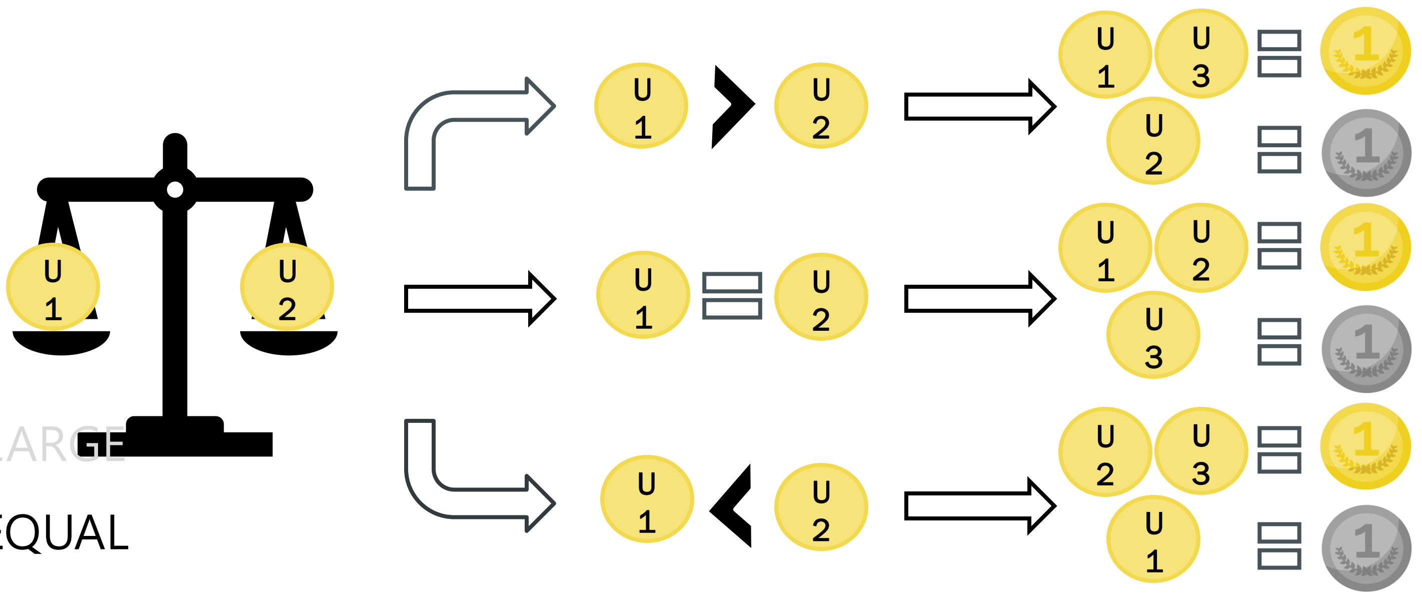
- LARGE
- EQUAL
- SMALL

ALGORITHM DETAILS

- Main
- Compare with 3 coins
- Compare with 5 coins
- Static finding

In this case, there is just 1 fake coin





•LARGE

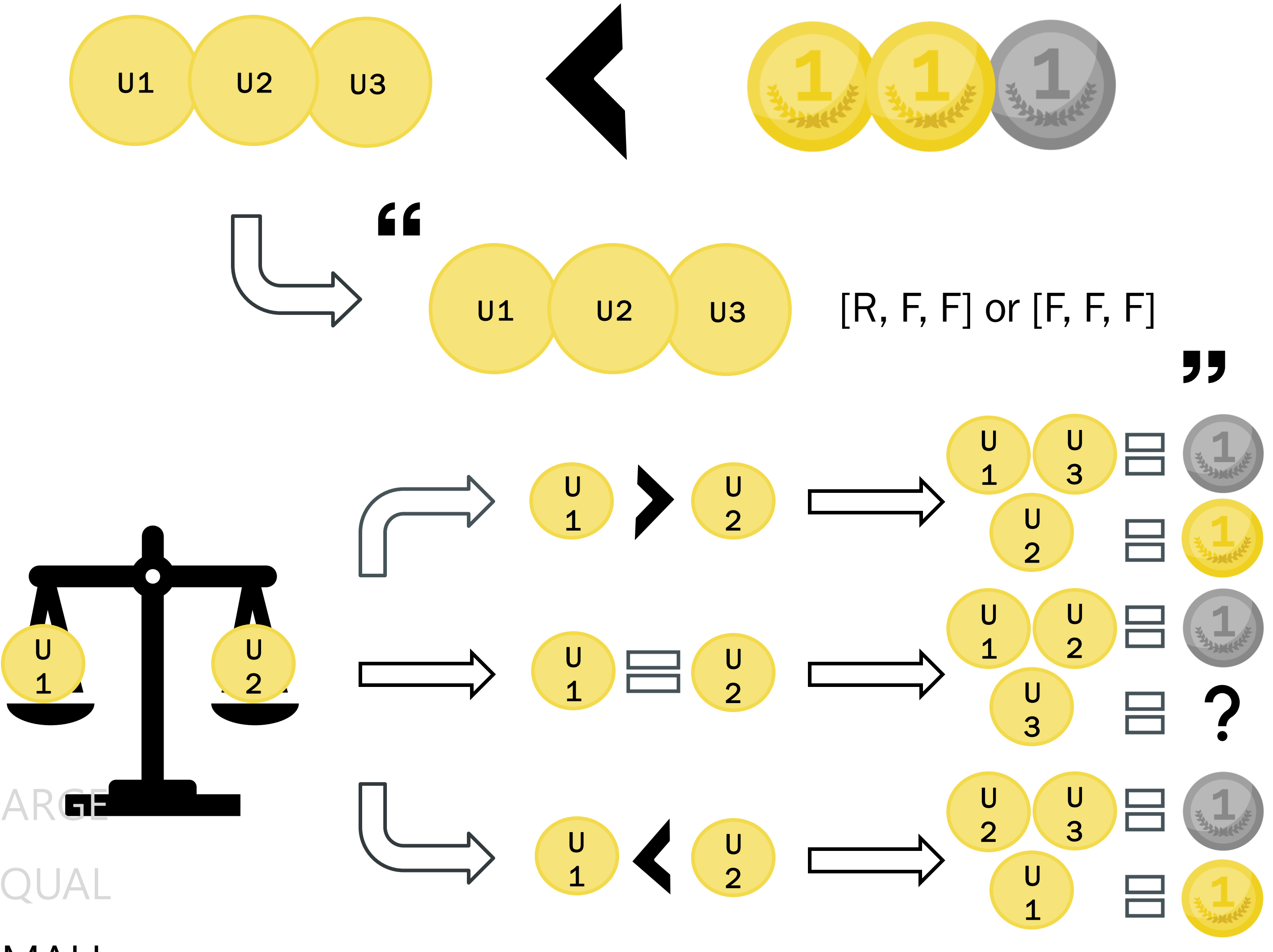
•EQUAL

•SMALL

ALGORITHM DETAILS

- Main
- Compare with 3 coins
- Compare with 5 coins
- Static finding

In this case, there are 2 or more fake coins => 1 or no real coin

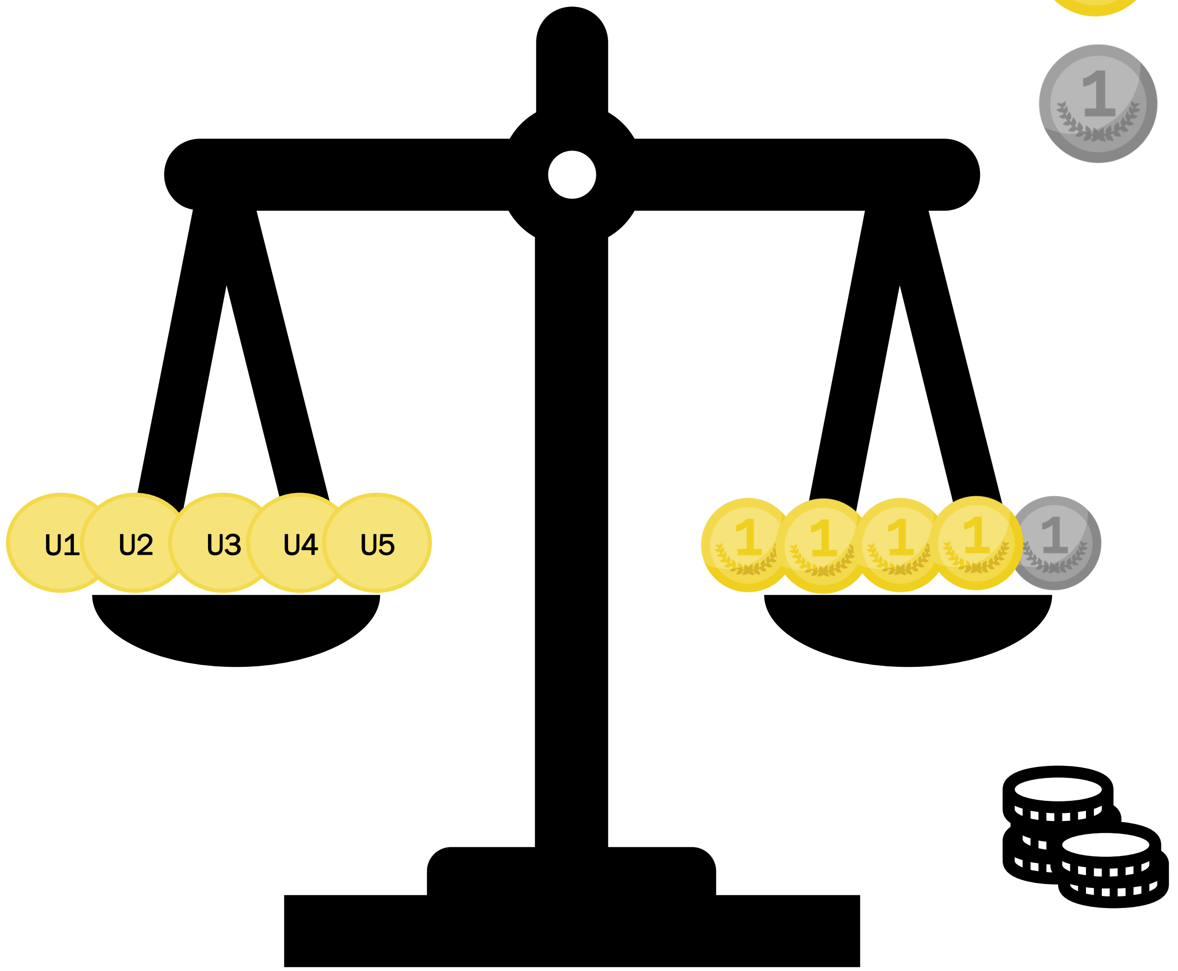


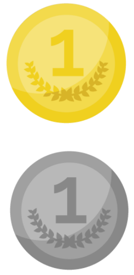
- LARGE
- EQUAL
- SMALL

ALGORITHM DETAILS

- Main
- Compare with 3 coins
- Compare with 5 coins
- Static finding

- Compare with 4 real coins and 1 fake coin

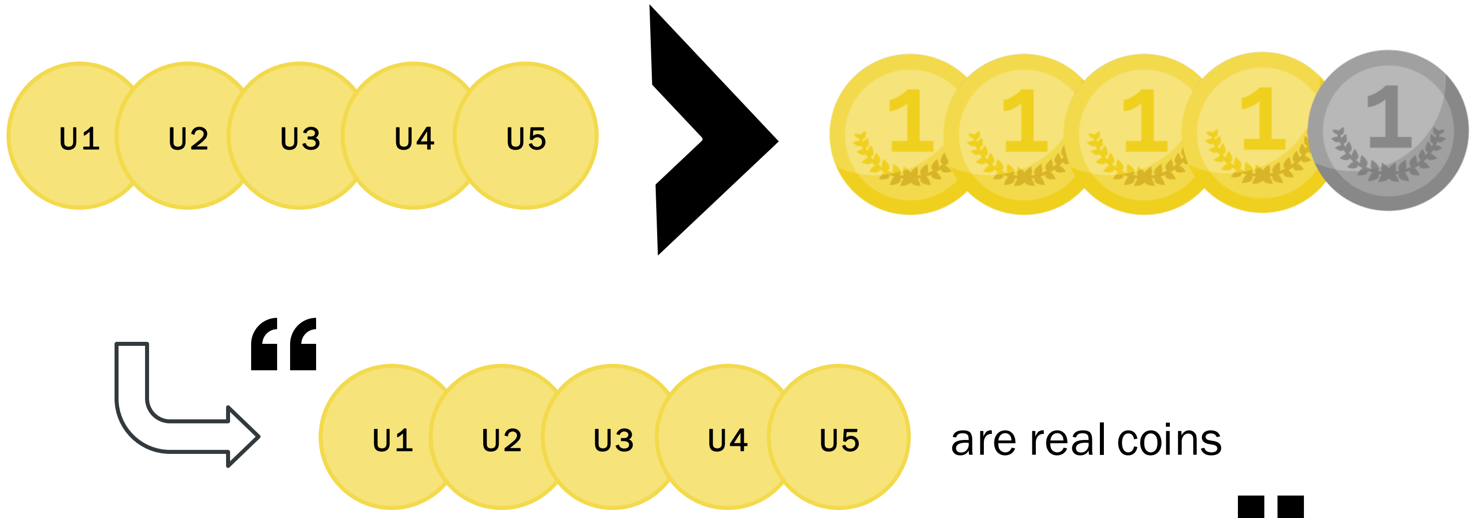




ALGORITHM DETAILS

- Main
- Compare with 3 coins
- Compare with 5 coins
- Static finding

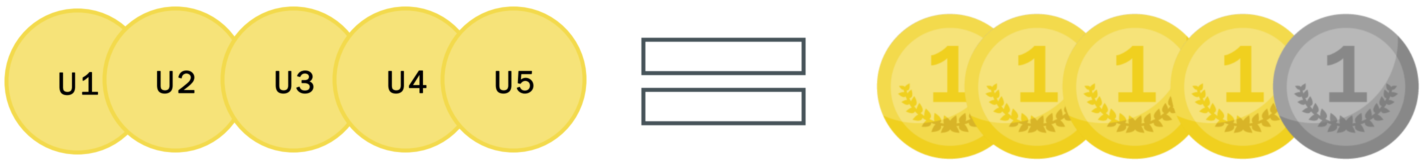
- If Case is Large, all coins on the left become real coins.

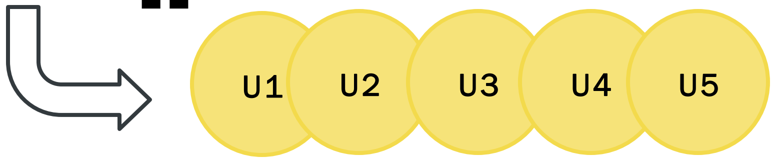


- LARGE
- EQUAL
- SMALL

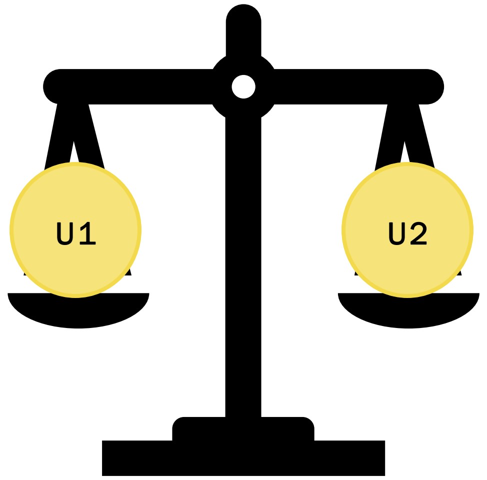
ALGORITHM DETAILS

- If case is Equal, There is only one fake coin, so compare one by one.





“Four are real coins
One is fake coin”



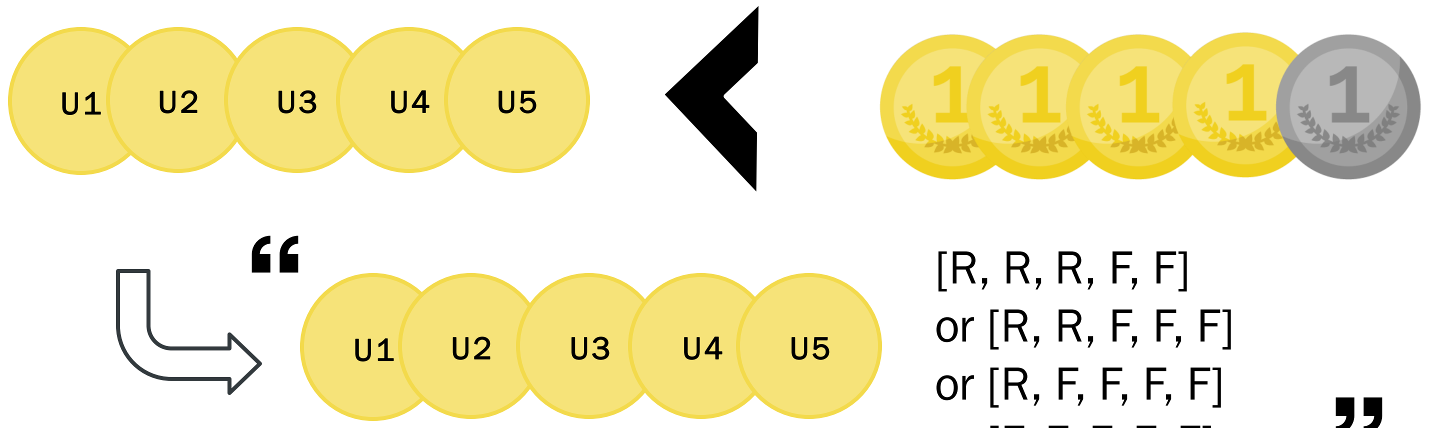
- Compare 1 and 2, if case is Small or Large, No further comparison is necessary.
- If case is Equal, compare 3 and 4.

•LARGE
•EQUAL
•SMALL

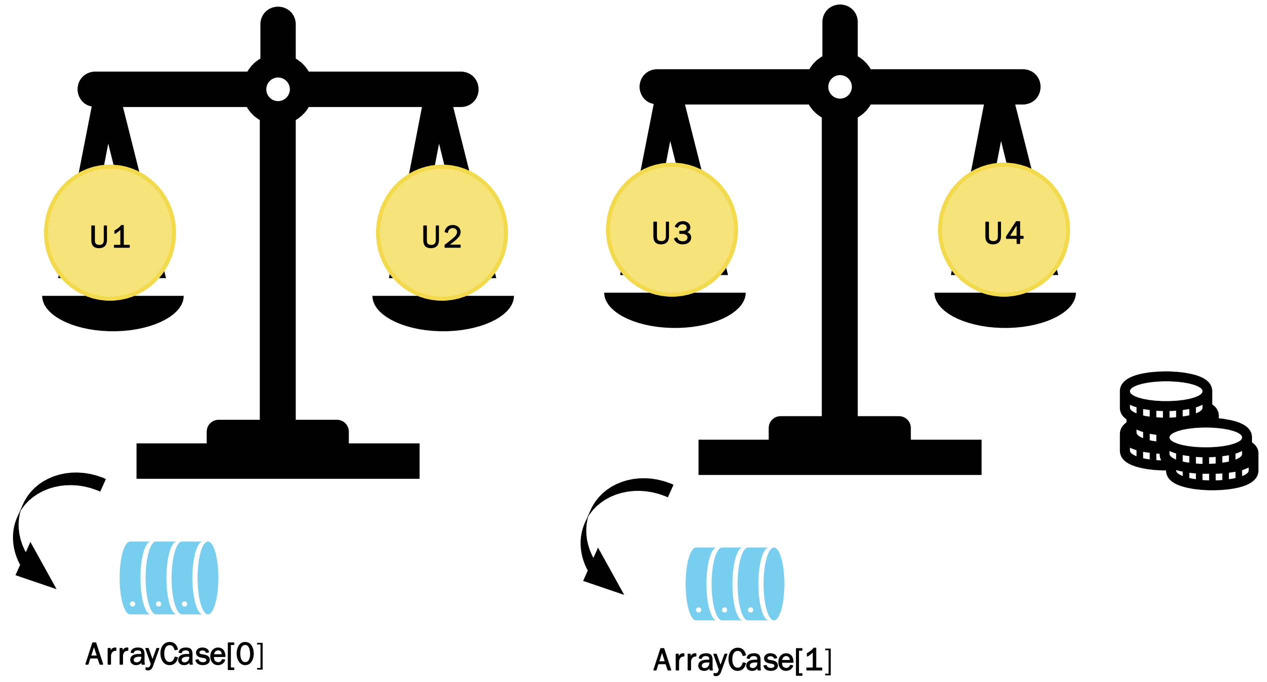
ALGORITHM DETAILS

- Main
- Compare with 3 coins
- Compare with 5 coins
- Static finding

- If case is small, compare 1, 2 and compare 3,4



"[R, R, R, F, F]
or [R, R, F, F, F]
or [R, F, F, F, F]
or [F, F, F, F, F]"



•LARGE

•EQUAL

•SMALL

ALGORITHM DETAILS

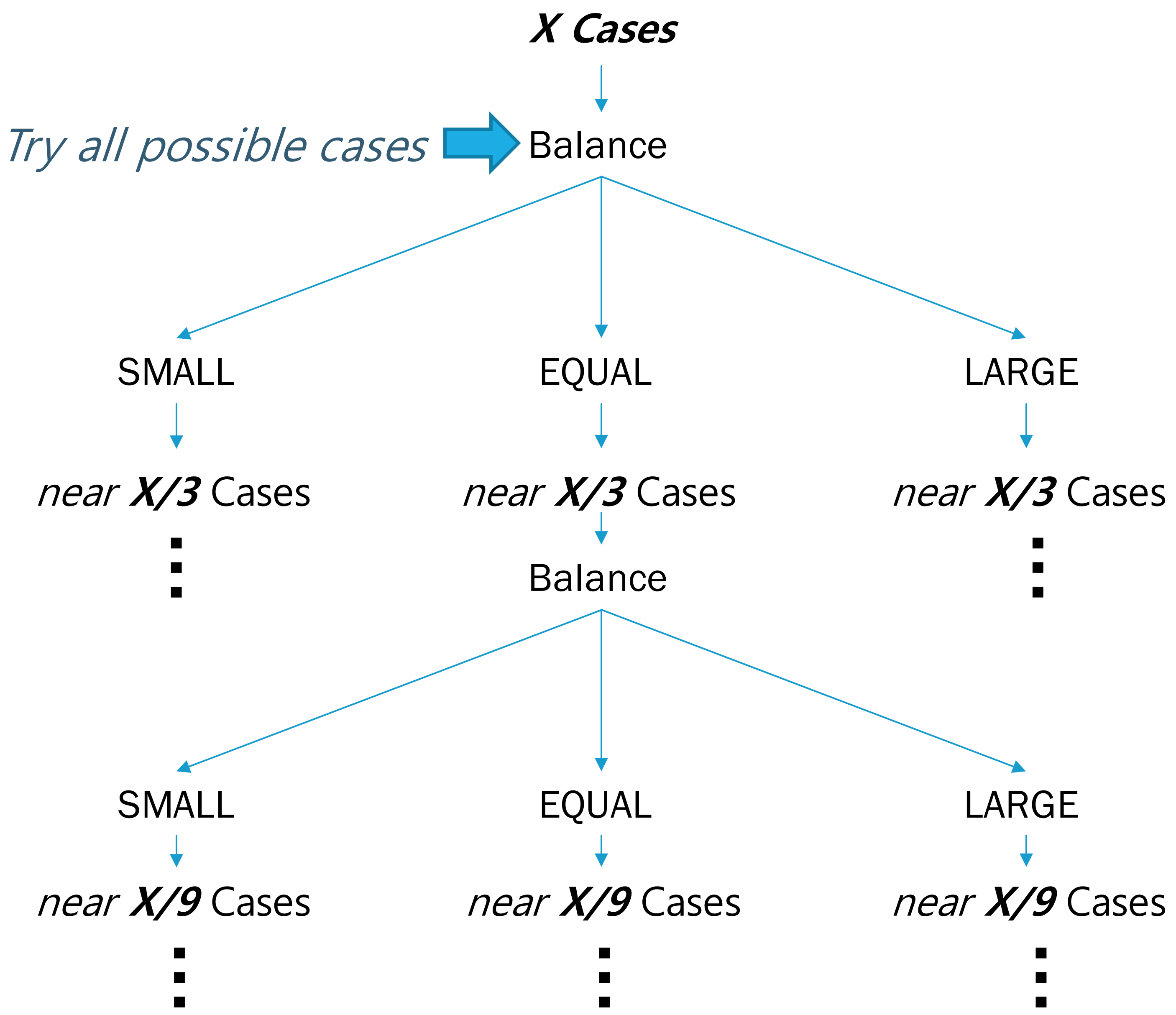
- Main
- Compare with 3 coins
- Compare with 5 coins
- Static finding

Possible status in case of comparing 1 and 2, or 3 and 4.

Compare 1,2 / Compare 3,4	Large	Equal	Small
Large	real 1,3 fake 2,4	Compare 1,5	real 2,3 fake 1,4
Equal	Compare 3,5	Compare 1,3	Compare 3,5
Small	real 1,4 fake 2,3	Compare 1,5	real 2,4 fake 1,3

ALGORITHM DETAILS

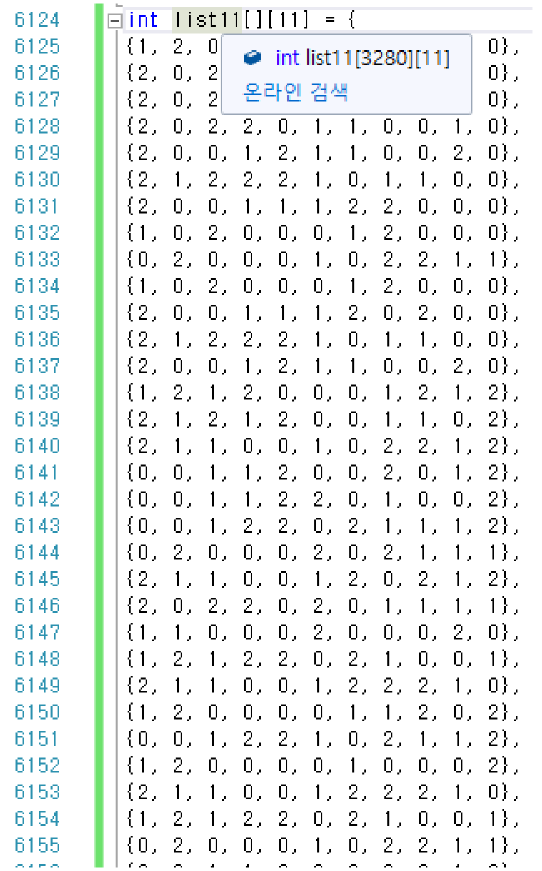
- Main
- Compare with 3 coins
- Compare with 5 coins
- Static finding



ALGORITHM DETAILS

- Main
- Compare with 3 coins
- Compare with 5 coins
- Static finding

- *Make 10 decision trees, and hard coded.*



```
int list11[][11] = {
{1, 2, 0                                0},
{2, 0, 2                                0},
{2, 0, 2                                0},
{2, 0, 2, 2, 0, 1, 1, 0, 0, 1, 0},
{2, 0, 0, 1, 2, 1, 1, 0, 0, 2, 0},
{2, 1, 2, 2, 2, 1, 0, 1, 1, 0, 0},
{2, 0, 0, 1, 1, 1, 2, 2, 0, 0, 0},
{1, 0, 2, 0, 0, 0, 1, 2, 0, 0, 0},
{0, 2, 0, 0, 0, 1, 0, 2, 2, 1, 1},
{1, 0, 2, 0, 0, 0, 1, 2, 0, 0, 0},
{2, 0, 0, 1, 1, 1, 2, 0, 2, 0, 0},
{2, 1, 2, 2, 2, 1, 0, 1, 1, 0, 0},
{2, 0, 0, 1, 2, 1, 1, 0, 0, 2, 0},
{1, 2, 1, 2, 0, 0, 0, 1, 2, 1, 2},
{2, 1, 2, 1, 2, 0, 0, 1, 1, 0, 2},
{2, 1, 1, 0, 0, 1, 0, 2, 2, 1, 2},
{0, 0, 1, 1, 2, 0, 0, 2, 0, 1, 2},
{0, 0, 1, 1, 2, 2, 0, 1, 0, 0, 2},
{0, 0, 1, 2, 2, 0, 2, 1, 1, 1, 2},
{0, 2, 0, 0, 0, 2, 0, 2, 1, 1, 1},
{2, 1, 1, 0, 0, 1, 2, 0, 2, 1, 2},
{2, 0, 2, 2, 0, 2, 0, 1, 1, 1, 1},
{1, 1, 0, 0, 0, 2, 0, 0, 0, 2, 0},
{1, 2, 1, 2, 2, 0, 2, 1, 0, 0, 1},
{2, 1, 1, 0, 0, 1, 2, 2, 2, 1, 0},
{1, 2, 0, 0, 0, 0, 1, 1, 2, 0, 2},
{0, 0, 1, 2, 2, 1, 0, 2, 1, 1, 2},
{1, 2, 0, 0, 0, 0, 1, 0, 0, 0, 2},
{2, 1, 1, 0, 0, 1, 2, 2, 2, 1, 0},
{1, 2, 1, 2, 2, 0, 2, 1, 0, 0, 1},
{0, 2, 0, 0, 0, 1, 0, 2, 2, 1, 1},
```

- *Total 9,405 lines*

ALGORITHM DETAILS

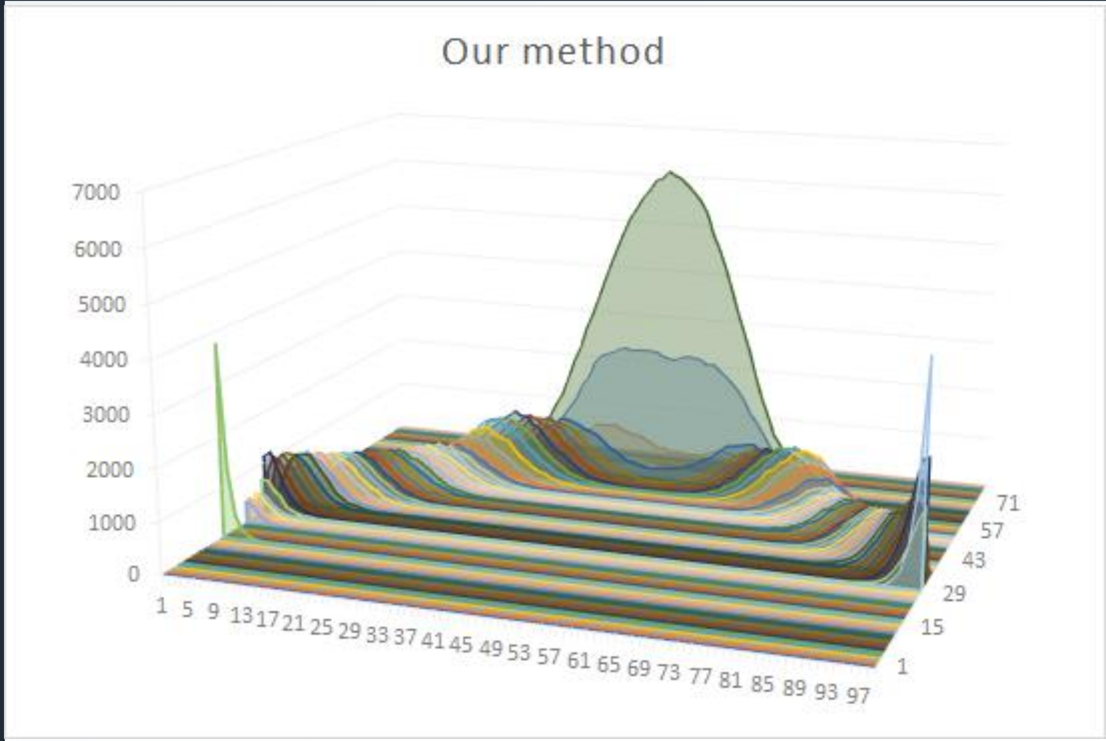
- Main
- Compare with 3 coins
- Compare with 5 coins
- Static finding

- *It can find whatever the number of fake coins is among* ***X*** *coins just in* ***Y*** *times!*

X	Y
2	1
3	2
4	3
5	3~4
6	4
7	5
8	5~6
9	6
10	6~7
11	7

Performance

- ◆Compare with 3 coins
- ◆Compare with 5 coins
- ◆Static Finding
- ◆Our method

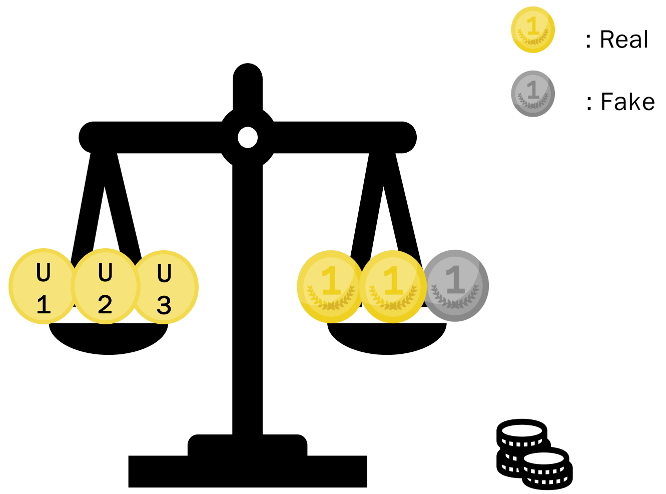


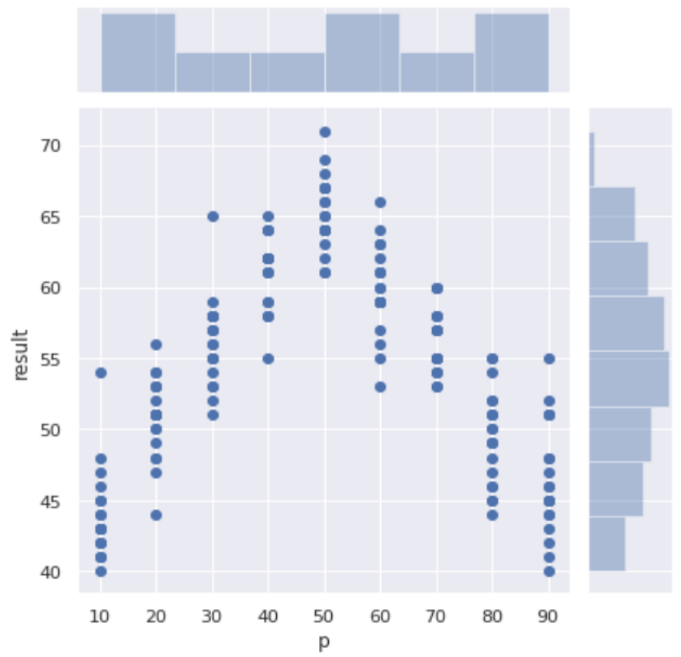
PERFORMANCE

- **Compare with 3 coins**
- Compare with 5 coins
- Static Finding
- Our method

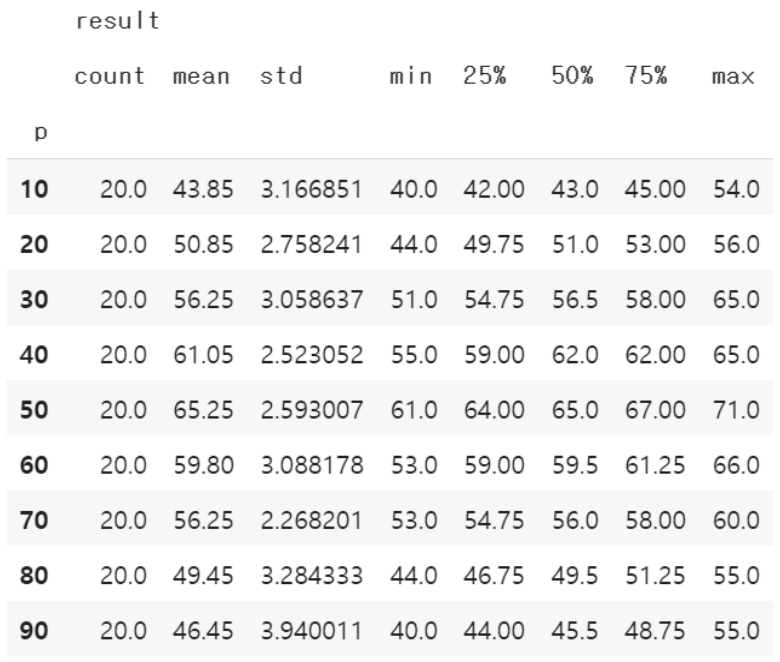
❖ ***P** is probability*

❖ ***Result** is sum of balance counts*





- Max 71, min 40
- It has the best performance where there are **more than 23 and less than 78 fake coins,**

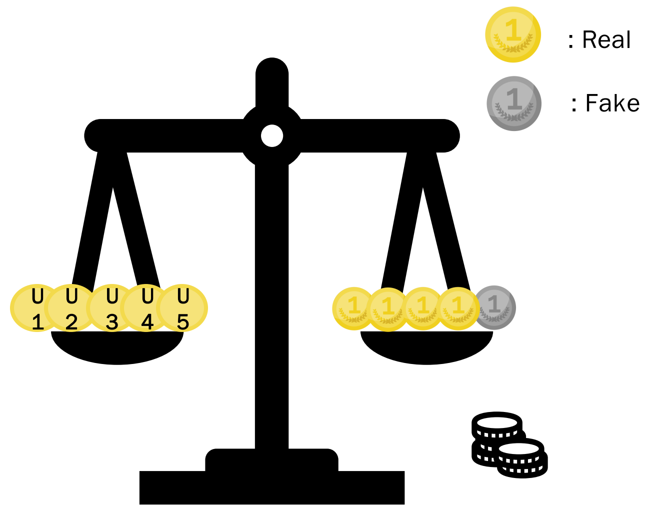


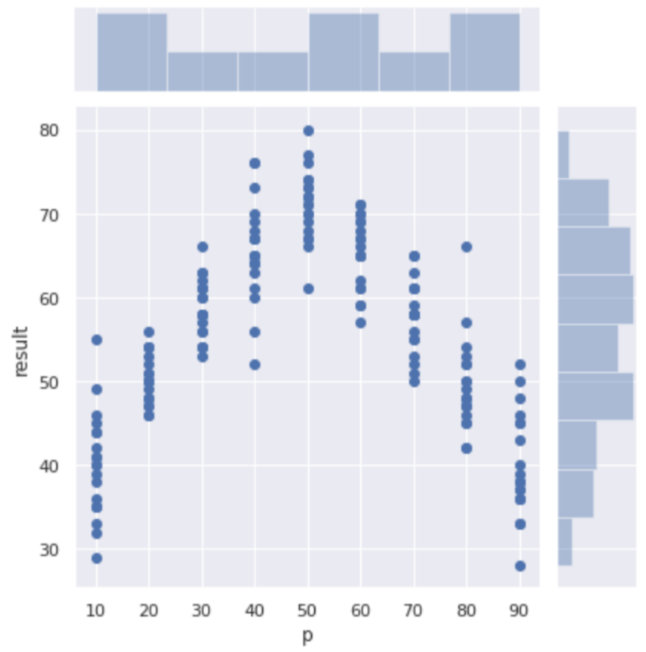
	result							
	count	mean	std	min	25%	50%	75%	max
p								
10	20.0	43.85	3.166851	40.0	42.00	43.0	45.00	54.0
20	20.0	50.85	2.758241	44.0	49.75	51.0	53.00	56.0
30	20.0	56.25	3.058637	51.0	54.75	56.5	58.00	65.0
40	20.0	61.05	2.523052	55.0	59.00	62.0	62.00	65.0
50	20.0	65.25	2.593007	61.0	64.00	65.0	67.00	71.0
60	20.0	59.80	3.088178	53.0	59.00	59.5	61.25	66.0
70	20.0	56.25	2.268201	53.0	54.75	56.0	58.00	60.0
80	20.0	49.45	3.284333	44.0	46.75	49.5	51.25	55.0
90	20.0	46.45	3.940011	40.0	44.00	45.5	48.75	55.0

PERFORMANCE

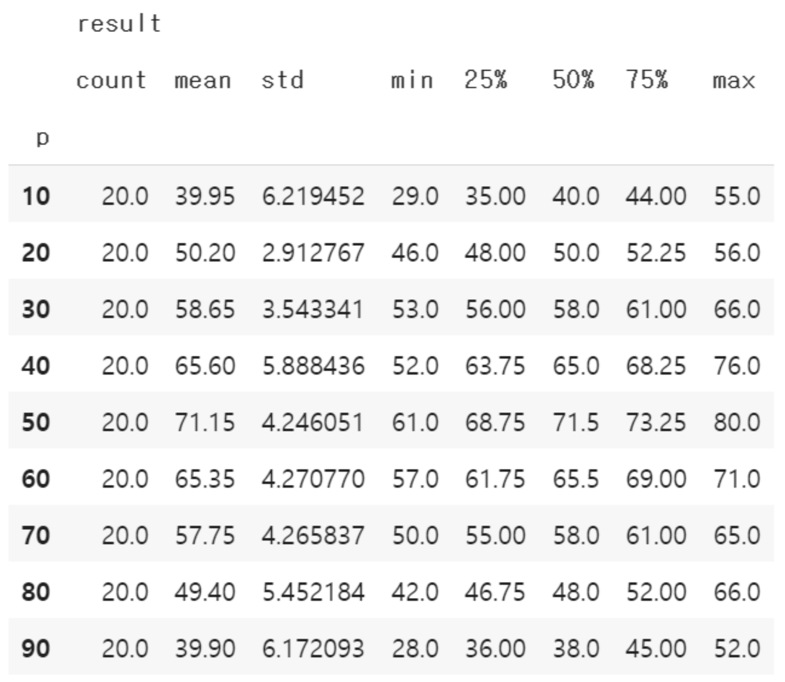
- Compare with 3 coins
- Compare with 5 coins
- Static Finding
- Our method

- ***P*** *is probability*
- ***Result*** *is sum of balance counts*





- Max 80, min 28
- It has the best performance where there are **less than 23 or more than 78 fake coins,**

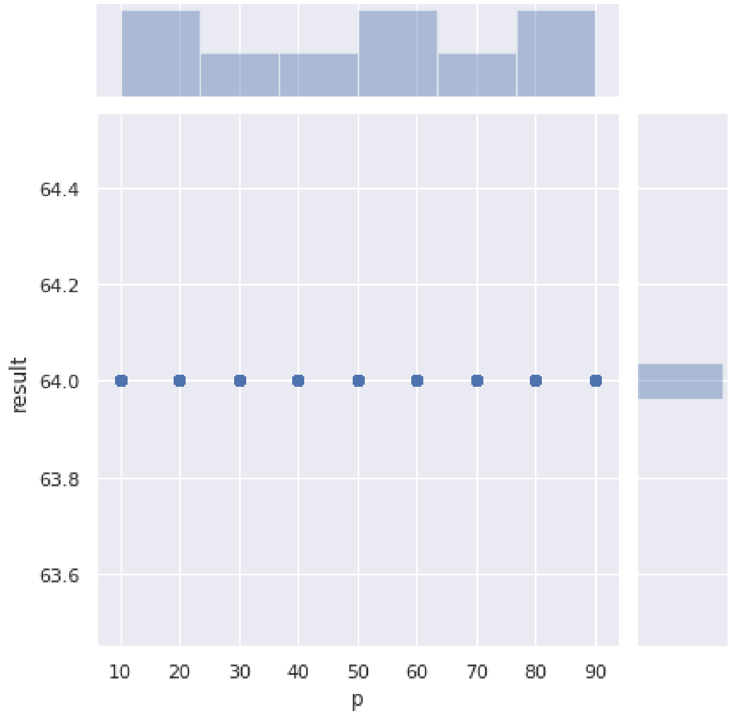


	result							
	count	mean	std	min	25%	50%	75%	max
p								
10	20.0	39.95	6.219452	29.0	35.00	40.0	44.00	55.0
20	20.0	50.20	2.912767	46.0	48.00	50.0	52.25	56.0
30	20.0	58.65	3.543341	53.0	56.00	58.0	61.00	66.0
40	20.0	65.60	5.888436	52.0	63.75	65.0	68.25	76.0
50	20.0	71.15	4.246051	61.0	68.75	71.5	73.25	80.0
60	20.0	65.35	4.270770	57.0	61.75	65.5	69.00	71.0
70	20.0	57.75	4.265837	50.0	55.00	58.0	61.00	65.0
80	20.0	49.40	5.452184	42.0	46.75	48.0	52.00	66.0
90	20.0	39.90	6.172093	28.0	36.00	38.0	45.00	52.0

PERFORMANCE

- Compare with 3 coins
- Compare with 5 coins
- Static Finding
- Our method

- ***P** is probability*
- ***Result** is sum of balance counts*



- **Almost always 64 times**
- It has best worst performance.

	result							
	count	mean	std	min	25%	50%	75%	max
p								
10	20.0	64.0	0.0	64.0	64.0	64.0	64.0	64.0
20	20.0	64.0	0.0	64.0	64.0	64.0	64.0	64.0
30	20.0	64.0	0.0	64.0	64.0	64.0	64.0	64.0
40	20.0	64.0	0.0	64.0	64.0	64.0	64.0	64.0
50	20.0	64.0	0.0	64.0	64.0	64.0	64.0	64.0
60	20.0	64.0	0.0	64.0	64.0	64.0	64.0	64.0
70	20.0	64.0	0.0	64.0	64.0	64.0	64.0	64.0
80	20.0	64.0	0.0	64.0	64.0	64.0	64.0	64.0
90	20.0	64.0	0.0	64.0	64.0	64.0	64.0	64.0

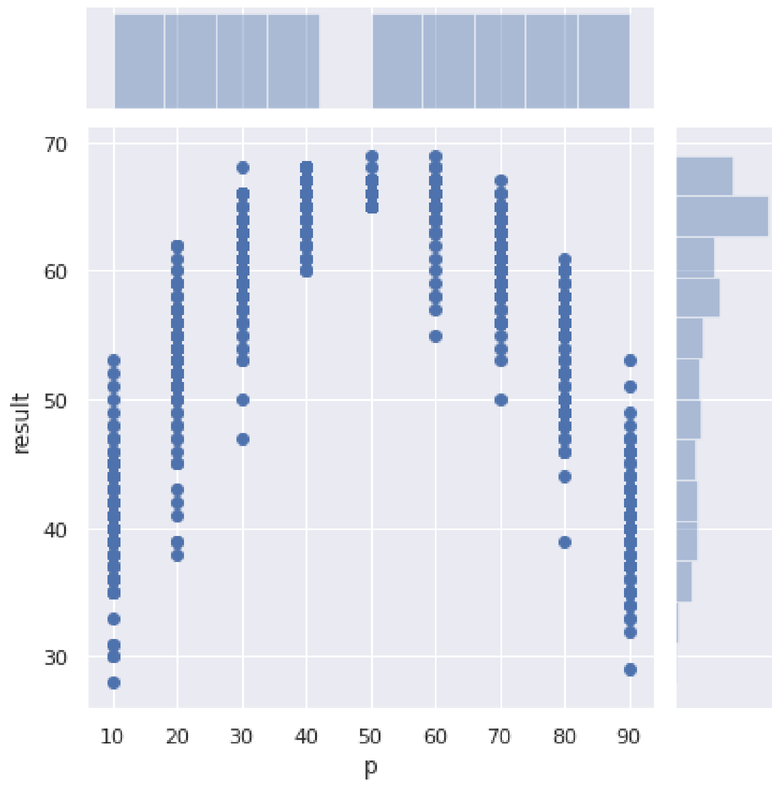
PERFORMANCE

- Compare with 3 coins
- Compare with 5 coins
- Static Finding
- Our method

❖ ***P*** *is probability*

❖ ***Result*** *is sum of balance counts*

Max 69, min 28



	result							
	count	mean	std	min	25%	50%	75%	max
p								
10	100.0	41.08	4.749652	28.0	38.75	41.0	44.25	53.0
20	100.0	52.85	4.953063	38.0	51.00	53.0	56.00	62.0
30	100.0	60.34	3.593541	47.0	58.00	61.0	63.00	68.0
40	100.0	65.13	1.967694	60.0	64.75	65.0	66.00	68.0
50	100.0	65.55	0.783349	65.0	65.00	65.0	66.00	69.0
60	100.0	64.87	2.158633	55.0	65.00	65.0	66.00	69.0
70	100.0	60.58	3.188299	50.0	58.75	60.5	63.00	67.0
80	100.0	53.40	4.465355	39.0	50.00	54.5	57.00	61.0
90	100.0	40.49	4.421013	29.0	37.00	40.0	43.25	53.0

“ ***Three advantages are combined!*** ”

감사합니다