

Exact Bounds for Coin Weighing by Pairwise Comparisons

Te-Sheng Tan*

Dai-Yang Wu*

Wing-Kai Hon

1 Introduction

There is a classic puzzle about counterfeit coin problem: a man has 12 coins among which there has a counterfeit coin, which can only be told apart by its weight. How can one tell in not more than three weighings and determine which one is a counterfeit coin [1].

The problem was so popular that have many other variants[2], for example, the weight of counterfeit coin is heavy or light are known; given an extra coin known to be real; or even answer the question by using a spring balance i.e. a weighing device that will return the exact weight. Halbeisen[3] generalize this problem when we are allowed to use more than 1 balances and consider more than one counterfeit coin.

Here we extend this question by a new direction: Now we have a real coins and b counterfeit coins ($a, b > 0$). The real coins are all the same weight and so are the counterfeit ones, but weight of a fake coin is less than weight of a real coin. Each time we compare only two coins have the same weight or not. In this case, assume a, b are known. We want to study the smallest number of comparisons that we can guarantee to find

1. a fake coin
2. a real coin
3. a fake coin and a real coin(need to know which one is real)
4. a fake coin and a real coin(don't need to know which one is real)

Note that the solution to Problem 3 could solve Problem 4, (solution for Problem 4)+(comparison of this pair) could solve Problem 3. That is, $ans(Problem4) \leq ans(Problem3) \leq ans(Problem4) + 1$

We shall discuss Problem 4 first, then solve Problem 1,2,3 similarly.

Problem 4 can transfer to a new circumstances: there is an engine broken because of burned wire, and we need to find a pair of new electric wires to fix it, but all the wire are mass up, we only know that we have a positive wires and b negative wires in the beginning, each time we can choose 2 of them to connect to the engine and check if they are the same Electrical polarity (not work) or not (work). The object is minimize the worst case of testing times that we can fix the engine.

2 An Equivalent Graph Problem

We can represent a testing scheme \mathcal{S} for the coin-weighing problem by a graph $G_{\mathcal{S}}$ as follows:

*Tan and Wu are co-first authors and main contributors of this work.

1. It contains $a + b$ vertices, where each vertex represents a distinct coin.
2. Two vertices are joined by an edge if and only if \mathcal{S} compares the two corresponding coins.

Note that we cannot gain extra information by comparing the same pair of coins twice; WLOG, we assume that each pair of coins are compared at most once.

Definition 1. For a coin-weighing problem with a real coins and b fake coins, we say a testing scheme \mathcal{S} is foolproof if no matter how a vertices in $G_{\mathcal{S}}$ are assigned as real and the remaining vertices are assigned as fake, $G_{\mathcal{S}}$ contains at least an unbalanced edge whose endpoints are with different labels; else, \mathcal{S} is non-foolproof.

Suppose that $a = 3$ and $b = 5$. Figure 1 shows two different foolproof schemes, while Figure 2 shows a non-foolproof scheme.



Figure 1: Two foolproof schemes for $a = 3$ and $b = 5$. Isolated vertices are omitted for brevity.

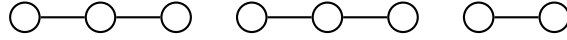


Figure 2: A non-foolproof scheme for $a = 3$ and $b = 5$: If the leftmost a vertices are assigned as real, there is no unbalanced edge.

Let $\mathcal{G}(a, b)$ be a graph formed by the union of two cliques K_a and K_b of sizes a and b , respectively. We have the following theorem.

Theorem 1. A testing scheme \mathcal{S} is foolproof if and only if $G_{\mathcal{S}}$ is not isomorphic to any subgraph of $\mathcal{G}(a, b)$.

Proof. Suppose that $G_{\mathcal{S}}$ is isomorphic to some subgraph of $\mathcal{G}(a, b) = K_a \cup K_b$. Then, we may select an arbitrary isomorphism, and label as real for all those a vertices in $G_{\mathcal{S}}$ that are mapped to the vertices in K_a , while label as fake for the remaining b vertices. Under such a labeling, there will be no unbalanced edge, so that \mathcal{S} is non-foolproof.

Conversely, if \mathcal{S} is non-foolproof, there exists a way to label a vertices in $G_{\mathcal{S}}$ as real and the remaining b vertices as fake so that there is no unbalanced edge. In other words, $G_{\mathcal{S}}$ can be partitioned into two subgraphs with a and b vertices, respectively; the former one is isomorphic to a subgraph of K_a and the latter one is isomorphic to a subgraph of K_b , so that $G_{\mathcal{S}}$ is isomorphic to a subgraph of $\mathcal{G}(a, b)$. \square

Definition 2. For a coin-weighing problem with a real coins and b fake coins, we say a foolproof testing scheme \mathcal{S} is optimal if it requires the minimal number of comparisons in the worst case; the minimal number of comparisons is denoted by $\tau(a, b)$.

Remark. By definition, $\tau(a, b) = \tau(b, a)$.

The target of this paper is to design optimal foolproof testing scheme so that by comparing coins according to the scheme, we are guaranteed to compare a real coin with a fake coin using the fewest comparisons in the worst case. By Theorem 1, it is equivalent to finding a graph G , with the fewest number of edges, that is not isomorphic to any subgraph of $\mathcal{G}(a, b)$.

2.1 Lower Bound

Theorem 2. *Any graph with $n \leq a + b$ vertices and $m \leq \lfloor (a + b)/2 \rfloor - 1$ edges is always isomorphic to some subgraph of $\mathcal{G}(a, b)$.*

Proof. Without loss of generality, we assume that $a \leq b$. We shall prove this theorem by induction on the sum $a + b$.

(Basis Case:) If $a = b = 1$, then $\lfloor (a + b)/2 \rfloor - 1 = 0$. Any graph with $n \leq 2$ vertices and $m \leq 0$ edges (i.e., no edges) is always isomorphic to some subgraph of $\mathcal{G}(1, 1)$.

(Inductive Case:) Suppose that the theorem holds for all $a + b \leq k$. Our target is to show that the theorem also holds for the case $a + b = k + 1$ with $a \leq b$. Consider a graph G with $n \leq k + 1$ vertices and with $m \leq \lfloor (k + 1)/2 \rfloor - 1$ edges.

1. If G is connected, then $n \leq m + 1 \leq (k + 1)/2 \leq b$, which is isomorphic to some subgraph of K_b , and thus isomorphic to some subgraph of $\mathcal{G}(a, b)$.
2. Otherwise, G is not connected. If G has no edges, then G is obviously isomorphic to some subgraph of $\mathcal{G}(a, b)$ since G has at most $a + b$ vertices. Else, let C be the connected component of G with the largest number n' of vertices (so that $n' \geq 2$). Then, the number of edges in C is at least $n' - 1$. To complete the proof, it is sufficient to show that $G - C$ is isomorphic to some subgraph of $\mathcal{G}(a, b - n')$, as we can map the vertices of C to an arbitrary set of n' vertices in the K_b component of $\mathcal{G}(a, b)$.

The number of vertices in $G - C$ is $k + 1 - n' = a + (b - n')$, and the number of edges of $G - C$ is at most

$$\begin{aligned} m - (n' - 1) &\leq \lfloor (k + 1)/2 \rfloor - n' = \lfloor (k + 1)/2 - n' \rfloor \\ &= \lfloor (k + 1 - n')/2 - n'/2 \rfloor \\ &\leq \lfloor (k + 1 - n')/2 - 1 \rfloor \\ &= \lfloor (k + 1 - n')/2 \rfloor - 1 = \lfloor (a + (b - n'))/2 \rfloor - 1. \end{aligned}$$

By induction hypothesis, $G - C$ is isomorphic to a subgraph of $\mathcal{G}(a, b - n')$, and consequently G is isomorphic to some subgraph of $\mathcal{G}(a, b)$.

In both cases, G is isomorphic to a subgraph of $\mathcal{G}(a, b)$. This completes the proof of the induction case, so that by the principle of mathematical induction, the theorem follows. \square

Immediately, we have the following corollaries.

Corollary 1. *For any a and b , $\tau(a, b) \geq \lfloor (a + b)/2 \rfloor$.*

Proof. The corollary is a direct consequence of Theorems 1 and 2. \square

Corollary 2. *When a and b are both odd, $\tau(a, b) = (a + b)/2$.*

Proof. Consider a testing scheme \mathcal{S} that partitions the coins into $(a + b)/2$ pairs, and then compares each pair of coins. The corresponding graph $G_{\mathcal{S}}$ consists of $(a + b)/2$ disjoint edges (as in Figure 1(b)), and is not isomorphic to any subgraph of $\mathcal{G}(a, b)$. Thus, $G_{\mathcal{S}}$ is foolproof, and $\tau(a, b) \leq (a + b)/2$. On the other hand, by Corollary 1, we have $\tau(a, b) \geq (a + b)/2$ for odd a and odd b . The corollary thus follows. \square

Corollary 3. *Consider a testing scheme \mathcal{S} whose corresponding graph $G_{\mathcal{S}}$ forms a path with $\max(\mathbf{a}, \mathbf{b}) + 1$ nodes (as in Figure 1(a)). Then, \mathcal{S} is foolproof, and it uses at most twice the number of comparisons of any optimal foolproof scheme.*

Proof. The graph $G_{\mathcal{S}}$ is not isomorphic to any subgraph of $\mathcal{G}(\mathbf{a}, \mathbf{b})$, so that it is foolproof. The number of comparisons by \mathcal{S} is

$$\begin{aligned} \max(\mathbf{a}, \mathbf{b}) &\leq \mathbf{a} + \mathbf{b} - 1 &= 2 \times ((\mathbf{a} + \mathbf{b} - 1)/2) \\ &\leq 2 \times \lfloor (\mathbf{a} + \mathbf{b})/2 \rfloor &\leq 2 \times \tau(\mathbf{a}, \mathbf{b}), \end{aligned}$$

where the last inequality comes from Corollary 1. The corollary thus follows. \square

3 From Graph to Integer Partition

This section shows an interesting connection between designing an optimal foolproof scheme and searching for an integer partitioning of some special form. We begin with the following simple lemma, which is crucial in establishing the connection.

Lemma 1. *If \mathcal{S} is an optimal foolproof scheme, then its corresponding graph $G_{\mathcal{S}}$ does not contain any cycle.*

Proof. Suppose on the contrary that $G_{\mathcal{S}}$ does. Then, we can obtain a graph G' by removing an edge from some cycle in $G_{\mathcal{S}}$. If G' is not isomorphic to any subgraph of $\mathcal{G}(\mathbf{a}, \mathbf{b})$, by Theorem 1, this would imply G' corresponds to a foolproof scheme; furthermore, such a scheme performs fewer comparisons than \mathcal{S} , so that \mathcal{S} is not optimal. Otherwise, if G' is isomorphic to some subgraph of $\mathcal{G}(\mathbf{a}, \mathbf{b})$, then $G_{\mathcal{S}}$ would also be isomorphic to some subgraph of $\mathcal{G}(\mathbf{a}, \mathbf{b})$ (using the same mapping between the vertices), so that \mathcal{S} is not foolproof by Theorem 1. Contradiction occurs in both cases, and the lemma thus follows. \square

Based on the above lemma, the graph $G_{\mathcal{S}}$ corresponding to an optimal foolproof scheme \mathcal{S} must be a *forest*. Indeed, we may observe that the *shape* of each connected tree is not important: Precisely, for each tree T in $G_{\mathcal{S}}$ that connects some set U of vertices, we can replace T by any other tree that connects the vertices in U , and the resulting scheme remains foolproof.¹ Also, after replacing T , the new scheme remains optimal as it uses the same number of comparisons. This naturally implies that an optimal foolproof scheme is related to some kind of *partitioning* of the integer $\mathbf{a} + \mathbf{b}$. In the following, we shall unveil the property of such a partitioning. We first define a related concept.

Definition 3. *Let P be a multiset of positive integers. We say P avoids a positive integer x if for any subset $P' \subseteq P$, the sum of all integers in P' is not equal to x .*

Let $P = \{p_1, p_2, \dots, p_{|P|}\}$ be a multiset of positive integers whose sum is $\mathbf{a} + \mathbf{b}$. In other words, P forms an integer partition of $\mathbf{a} + \mathbf{b}$. We say a testing scheme \mathcal{S} *corresponds to* P if $G_{\mathcal{S}}$ is a forest whose trees have sizes $p_1, p_2, \dots, p_{|P|}$, respectively. Then, we have the following theorem.

Theorem 3. *Let \mathcal{S} be a testing scheme whose corresponding graph $G_{\mathcal{S}}$ is a forest. Then \mathcal{S} is foolproof if and only if \mathcal{S} corresponds to a partition P of the integer $\mathbf{a} + \mathbf{b}$ that avoids \mathbf{a} .*

¹If not, the latter graph is isomorphic to some subgraph of $\mathcal{G}(\mathbf{a}, \mathbf{b})$, but then $G_{\mathcal{S}}$ would also be isomorphic to some subgraph of $\mathcal{G}(\mathbf{a}, \mathbf{b})$ under the same vertex mapping.

Proof. We prove the necessary and sufficient conditions separately, each by contradiction.

(\Rightarrow) Suppose that \mathcal{S} is foolproof. Assume on the contrary that P does not avoid \mathbf{a} , then we can partition P into two subsets $P_{\mathbf{a}}$ and $P_{\mathbf{b}}$ such that the sum of integers in $P_{\mathbf{a}}$ is equal to \mathbf{a} (and thus, the sum of integers in $P_{\mathbf{b}}$ is \mathbf{b}). Then, consider those trees in $G_{\mathcal{S}}$ with sizes $p \in P_{\mathbf{a}}$, they together would be isomorphic to some subgraph of $K_{\mathbf{a}}$; similarly, the remaining trees in $G_{\mathcal{S}}$ would be isomorphic to some subgraph of $K_{\mathbf{b}}$. This implies that $G_{\mathcal{S}}$ is isomorphic to some subgraph of $\mathcal{G}(\mathbf{a}, \mathbf{b})$, so that \mathcal{S} is not foolproof by Theorem 1. A contradiction occurs.

(\Leftarrow) Suppose that P avoids \mathbf{a} . Assume on the contrary that \mathcal{S} is not foolproof. By Theorem 1, $G_{\mathcal{S}}$ is isomorphic to some subgraph of $\mathcal{G}(\mathbf{a}, \mathbf{b})$. Consider a particular isomorphism f . Note that the number of vertices of both $G_{\mathcal{S}}$ and $\mathcal{S}(\mathbf{a}, \mathbf{b})$ are the same, so that f is a bijection between the two vertex sets. Based on f , we can partition the vertices in $G_{\mathcal{S}}$ into two groups, where the first group contains those who are mapped to vertices in the $K_{\mathbf{a}}$ component of $\mathcal{G}(\mathbf{a}, \mathbf{b})$, and the second one contains those remaining vertices. Also, all vertices from the same tree in $G_{\mathcal{S}}$ must be in the same group. By focussing on those trees whose vertices are mapped to the first group, their sizes adds up to \mathbf{a} , so that P does not avoid \mathbf{a} . A contradiction occurs. \square

For instance, although the scheme corresponding to Figure 2 is not foolproof when $\mathbf{a} = 3$ and $\mathbf{b} = 5$, it is foolproof when $\mathbf{a} = 1$ and $\mathbf{b} = 7$, or when $\mathbf{a} = 4$ and $\mathbf{b} = 4$. Also, we have the following corollary.

Corollary 4. *Let \mathcal{S} be an optimal testing scheme. Then, \mathcal{S} corresponds to a partition P_{\max} of $\mathbf{a} + \mathbf{b}$, with the maximal number of parts, that avoids \mathbf{a} ; furthermore, $\tau(\mathbf{a}, \mathbf{b}) = \mathbf{a} + \mathbf{b} - |P_{\max}|$, where the notation $|P|$ denotes the number of parts in a partition P .*

Proof. By Lemma 1, $G_{\mathcal{S}}$ is a forest. Then, by Theorem 3, \mathcal{S} corresponds to a partition P of $\mathbf{a} + \mathbf{b}$ that avoids \mathbf{a} . Furthermore, the number of comparisons performed by \mathcal{S} is exactly $\mathbf{a} + \mathbf{b} - |P|$. As \mathcal{S} is optimal, this implies \mathcal{S} minimizes the number of comparisons, so that the corresponding partition P maximizes the number of parts. The corollary thus follows. \square

Corollary 5. *For any $\mathbf{b} \geq 1$,*

$$\tau(2, \mathbf{b}) = \mathbf{b} + 2 - \left\lfloor \frac{\mathbf{b} + 2}{3} \right\rfloor.$$

Proof. For any partition P of $2 + \mathbf{b}$ that avoids 2, P contains at most one 1 and no 2s, so that

$$|P| \leq \max \left\{ 1 + \frac{\mathbf{b} + 1}{3}, \frac{\mathbf{b} + 2}{3} \right\} \leq 1 + \frac{\mathbf{b} + 1}{3} = \frac{\mathbf{b} + 4}{3}.$$

Furthermore, since $|P|$ is an integer, the above inequality implies that

$$|P| \leq \left\lceil \frac{\mathbf{b} + 4}{3} \right\rceil = \left\lfloor \frac{\mathbf{b} + 2}{3} \right\rfloor.^2$$

Thus by Corollary 4, we have $\tau(2, \mathbf{b}) \geq \mathbf{b} + 2 - \lfloor (\mathbf{b} + 2)/3 \rfloor$.

However, by setting P to be

- $P = \{3, 3, \dots\}$ (where \dots represents trailing 3s) for $\mathbf{b} + 2 \equiv 0 \pmod{3}$, or
- $P = \{4, 3, \dots\}$ (where \dots represents trailing 3s) for $\mathbf{b} + 2 \equiv 1 \pmod{3}$, or
- $P = \{1, 4, 3, \dots\}$ (where \dots represents trailing 3s) for $\mathbf{b} + 2 \equiv 2 \pmod{3}$,

each partition P is a partition of $2 + \mathbf{b}$ that avoids 2, and with $|P| = \lfloor (\mathbf{b} + 2)/3 \rfloor$. By Corollary 4, this implies that $\tau(2, \mathbf{b}) \leq \mathbf{b} + 2 - \lfloor (\mathbf{b} + 2)/3 \rfloor$. As $\tau(2, \mathbf{b})$ is now upper-bounded and lower-bounded by the same desired quantity, the corollary thus follows. \square

²The equality follows from the fact: for any integer n and any positive integer m , $\lceil n/m \rceil = \lfloor (n + m - 1)/m \rfloor$.

4 Computing $\tau(\mathbf{a}, \mathbf{b})$

$R = \{\mathbf{a} + \mathbf{b}\}$

for each partition P of the number $\mathbf{a} + \mathbf{b}$ which avoids \mathbf{a} **do**

if $|P| > |R|$ **then**

$R = P$

end if

end for

output R

check P's avoidance by Dynamic-Programming as solving discrete knapsack problem, keep avoidance vector when enumerating.

Denote the number of partitions of n as $p(n)$

Denote the number of partitions avoids a of n as $p_{\text{avoids } a}(n)$

Time Complexity:

$$O(a \times p_{\text{avoids } \mathbf{a}}(\mathbf{a} + \mathbf{b})) \leq O(\mathbf{a} \times p(\mathbf{a} + \mathbf{b})) \leq O(\mathbf{a} \frac{\exp(\sqrt{\frac{2(\mathbf{a} + \mathbf{b})}{3}})}{\mathbf{a} + \mathbf{b}})$$

4.1 Results

$\rho(a, b)$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	1	2	2	3	3	4	4	5	5	6	6	7	7	8	8	9	9	10	10
2	1	2	2	2	3	3	3	4	4	4	5	5	5	6	6	6	7	7	7	8
3	2	2	3	3	4	3	5	4	6	5	7	6	8	7	9	8	10	9	11	10
4	2	2	3	4	4	4	4	4	5	5	5	5	6	6	6	7	7	7	8	8
5	3	3	4	4	5	5	6	5	7	5	8	6	9	7	10	8	11	9	12	10
6	3	3	3	4	5	6	6	6	6	6	6	6	7	7	7	7	7	7	7	8
7	4	3	5	4	6	6	7	7	8	7	9	7	10	7	11	8	12	9	13	10
8	4	4	4	4	5	6	7	8	8	8	8	8	8	8	8	8	9	9	9	10
9	5	4	6	5	7	6	8	8	9	9	10	9	11	9	12	9	13	9	14	10
10	5	4	5	5	5	6	7	8	9	10	10	10	10	10	10	10	10	10	10	10
11	6	5	7	5	8	6	9	8	10	10	11	11	12	11	13	11	14	11	15	11
12	6	5	6	5	6	6	7	8	9	10	11	12	12	12	12	12	12	12	12	12
13	7	5	8	6	9	7	10	8	11	10	12	12	13	13	14	13	15	13	16	13
14	7	6	7	6	7	7	7	8	9	10	11	12	13	14	14	14	14	14	14	14
15	8	6	9	6	10	7	11	8	12	10	13	12	14	14	15	15	16	15	17	15
16	8	6	8	7	8	7	8	8	9	10	11	12	13	14	15	16	16	16	16	16
17	9	7	10	7	11	7	12	9	13	10	14	12	15	14	16	16	17	17	18	17
18	9	7	9	7	9	7	9	9	9	10	11	12	13	14	15	16	17	18	18	18
19	10	7	11	8	12	7	13	9	14	10	15	12	16	14	17	16	18	18	19	19
20	10	8	10	8	10	8	10	10	10	10	11	12	13	14	15	16	17	18	19	20
$\tau(a, b)$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	2	2	3	3	4	4	5	5	6	6	7	7	8	8	9	9	10	10	11
2	2	2	3	4	4	5	6	6	7	8	8	9	10	10	11	12	12	13	14	14
3	2	3	3	4	4	6	5	7	6	8	7	9	8	10	9	11	10	12	11	13
4	3	4	4	4	5	6	7	8	8	9	10	11	11	12	13	13	14	15	15	16
5	3	4	4	5	5	6	6	8	7	10	8	11	9	12	10	13	11	14	12	15
6	4	5	6	6	6	6	7	8	9	10	11	12	12	13	14	15	16	17	18	18
7	4	6	5	7	6	7	7	8	8	10	9	12	10	14	11	15	12	16	13	17
8	5	6	7	8	8	8	8	8	9	10	11	12	13	14	15	16	16	17	18	18
9	5	7	6	8	7	9	8	9	9	10	10	12	11	14	12	16	13	18	14	19
10	6	8	8	9	10	10	10	10	10	10	11	12	13	14	15	16	17	18	19	20
11	6	8	7	10	8	11	9	11	10	11	11	12	12	14	13	16	14	18	15	20
12	7	9	9	11	11	12	12	12	12	12	12	12	13	14	15	16	17	18	19	20
13	7	10	8	11	9	12	10	13	11	13	12	13	13	14	14	16	15	18	16	20
14	8	10	10	12	12	13	14	14	14	14	14	14	14	14	15	16	17	18	19	20
15	8	11	9	13	10	14	11	15	12	15	13	15	14	15	15	16	16	18	17	20
16	9	12	11	13	13	15	15	16	16	16	16	16	16	16	16	16	17	18	19	20
17	9	12	10	14	11	16	12	16	13	17	14	17	15	17	16	17	17	18	18	20
18	10	13	12	15	14	17	16	17	18	18	18	18	18	18	18	18	18	18	19	20
19	10	14	11	15	12	18	13	18	14	19	15	19	16	19	17	19	18	19	19	20
20	11	14	13	16	15	18	17	18	19	20	20	20	20	20	20	20	20	20	20	20

Previously, to find OTS, we enumerate all partitions of $a + b$. It cost much time. Discovering from tables in Results 3.3. We found the cycle property. Take $a = 5$ as example.

$\tau(a, b)$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
5	3	4	4	5	5	6	6	8	7	10	8	11	9	12	10	13	11	14	12	15

From $b \geq 9$, $\tau(a, b)$ forms cycle of length 2. In general, when b is large enough $\tau(a, b)$ would forms into cycles. Using this property to improve our algorithm for the case b is much larger than a . The improved algorithm runs in time complexity with no b .

Definition 4. d is the smallest non-factor of a

Lemma 2. $\rho(a, b)$ has lower-bound $\lceil \frac{b}{d} \rceil$

Proof. To proof a lower-bound, we show a case whose number of partitions= $\lceil \frac{b}{d} \rceil$

The case is trivial, divide $a + b$ into $\lceil \frac{b}{d} \rceil - 1$ instances of d and a $a + b - d\lceil \frac{b}{d} \rceil - 1$ (denoted by s)

Prove the case avoids a by contradiction

If the subset contains s since $s > a \implies$ contradiction

If the subset doesn't contains s , it contains only several d , but since $d \nmid a \implies$ contradiction \square

Lemma 3. If a multiset S avoids a and composed only by factors of a , $|S| < a$

Proof. We shall prove this by contradiction

We denote the number of i in S as X_i , that is, we have X_i 'i'(s) in S

, the first k elements of S as S_k . S_0 is the empty set.

We define set $U(s)$ for a multiset s as subset-sums $\leq a$ of s . For example, $a = 5$, $U(1, 3, 3) = \text{ignore}_{>5}(\{0, 1, 3, 4, 6, 7\}) = \{0, 1, 3, 4\}$

Note that for any s , $U(s)$ always include zero.

We divide the proof into two sections.

1. $\forall k, |U(P_{k+1})| > |U(P_k)|$

Denote $U(P_{k+1})$ as u' and $U(P_k)$ as u .

Prove this by contradiction, if $|u'| = |u|$ for some k , let w is the $k+1$ th element of S , then

Since $u \subseteq u'$, $u = u'$

$\forall i \geq w, i \in u \implies i + w \in u' \implies i + w \in u$

Since U always include 0, $0 \in u \implies tw \in u \forall t$

w is a factor of a , thus $w \in u$, contradicts S avoids a

2. We discover $U(S_0)$ to $U(S_{|S|})$, since $U(S_0) = \{0\}$, each time U at least a new element. If $|S| \geq a$, then $U(S)$ must be $\{0, 1, \dots, a\}$, contradicts S avoids a \square

Theorem 4. $\forall b > d(d+1)(a-1) + (a - (a \bmod d))$, an optimal partition contains at least $\lfloor \frac{a}{d} \rfloor$ instances of d

Proof. Let S is a partition of $a + b$ avoids a , X_i is the number of times i appears in S and $X = \sum_i X_i$

Assume $X_d < \lfloor \frac{a}{d} \rfloor$,

Then

$$X = \sum_{i < d} X_i + X_d + \sum_{i > d} X_i$$

Since $\sum_i X_i \times i = \mathbf{a} + \mathbf{b}$, to maximize X , we use as much small number as possible. Hence

$$\begin{aligned} X &\leq \mathbf{a} - 1 + \lfloor \frac{\mathbf{a}}{d} \rfloor + \lfloor \frac{\mathbf{b} - (\mathbf{a} - (\mathbf{a} \bmod d))}{d+1} \rfloor \\ &\leq \mathbf{a} - 1 + \frac{\mathbf{a}}{d} + \frac{\mathbf{b} - (\mathbf{a} - (\mathbf{a} \bmod d))}{d+1} \end{aligned}$$

$$\forall \mathbf{b} > d(d+1)(\mathbf{a}-1) + (\mathbf{a} - (\mathbf{a} \bmod d)),$$

$$X < \mathbf{a} + \lfloor \frac{\mathbf{a}}{d} \rfloor + \mathbf{a}d - d \leq \lceil \frac{\mathbf{b}}{d} \rceil. \text{ contradicts lemma 4.1}$$

Hence S is not optimal □

Theorem 5. $\forall \mathbf{b} > d(d+1)(\mathbf{a}-1) + (\mathbf{a} - (\mathbf{a} \bmod d)), \rho(\mathbf{a}, \mathbf{b} + d) = \rho(\mathbf{a}, \mathbf{b}) + 1$

Proof. S is a optimal partition of $\mathbf{a}+\mathbf{b}$ avoids a ($\mathbf{b} > d(d+1)(\mathbf{a}-1) + (\mathbf{a} - (\mathbf{a} \bmod d))$)

$S' = S + 'd'$ is a partition of $\mathbf{a}+\mathbf{b}+d$

We divide the proof into avoidance part and optimal part

(Avoidance) Prove this by contradiction

If S' has subset-sum \mathbf{a} , S must has subset-sum \mathbf{a} or $\mathbf{a} - d$. The former is not possible from definition.

If the later happens, S has subset-sum \mathbf{a} since it has at least $\lfloor \frac{\mathbf{a}}{d} \rfloor$ instances of ' d ', and $\mathbf{a} - d$ couldn't has all of them. Also contradict definition.

S' avoids \mathbf{a} .

(Optimal) prove by contradiction

If S' is not optimal there exist an optimal partition P' that $|P'| > |S'|$ and has no subset-sum= a

P' has at least one d , let $P = P' - d$ ($\text{sum}(P) = \text{sum}(S)$) then P has no subset-sum= a and $|P| = |P'| - 1 > |S'| - 1 = |S|$, which contradicts S is optimal.

S' is optimal.

Thus the theorem holds. □

Corollary 6. for a constant \mathbf{a} , $\rho(\mathbf{a}, \mathbf{b})$ forms cycle of length d if \mathbf{b} is sufficiently large

4.2 Algorithm using cycle property

let $K = \mathbf{a} + \mathbf{b} - kd > d(d+1)a$ (choose maximal k , e.g. minimal K)

$R = \{\mathbf{a} + \mathbf{b}\}$

for each partition P of the number K which avoids \mathbf{a} **do**

if $|P| > |R|$ **then**

$R = P$

end if

end for

output $R+k$ instances of ' d '

check P 's avoidance by Dynamic-Programming as solving discrete knapsack problem, keep avoidance vector when enumerating.

Time Complexity:

Denote the number of partitions of n as $p(n)$

Denote the number of partitions avoids a of n as $p_{\text{avoid } a}(n)$

$$O(k + a \times p_{\text{avoid } a}(K)) = O(a \times p(K)) = O(a \frac{\exp(\sqrt{\frac{2(1+d(d+1)(\mathbf{a}-1)+\mathbf{a})}{3}})}{1 + d(d+1)(\mathbf{a}-1) + \mathbf{a}}) = O(\frac{\exp(\sqrt{\frac{2(1+d(d+1)(\mathbf{a}-1)+\mathbf{a})}{3}})}{d^2})$$

5 extended problem

Let's consider problem 1, 2 mentioned in the introduction. We shall solve Problem 1 first, then similarly solve Problem 2.

Optimal testing solution for finding a fake coin

Similar to Section 2, we could present a solution as a graph.

If after testing a graph, we could solve problem 1. This graph is a testing solution for fake (TS^-) , feasible. Otherwise it's infeasible.

Definition 5. *Optimal testing solution for fake (OTS^-) is TS^- with minimum number of comparison.*

denote the number of edges of OTS^- as $\tau^-(a, b)$.

Note that $OTS(a, b)$ could find a real and a fake coin. Thus $\tau(a, b)$ is an upper bound of $\tau^+(a, b)$ and $\tau^-(a, b)$.

A $TS^-(a, b)$ better than $OTS(a, b)$ must use another strategy, which is, though all comparisons result in balance we could still find a '1'.

Before solving TS^- , we observe how a solution works.

When claiming a solution, after these comparisons, there are two cases

Case 1. there is an unbalanced

Case 2. they are all balanced

If **Case 1.** happens then the solution works. If **Case 2.** happens, we must claim a fake coin, and this coin couldn't be a real one in any condition.

We can still present a solution as a integer partition.

Example for **Case 2.** let $(a, b) = (2, 6)$ the optimal solution is $[3, 3, '1', 1]$ the '1' is the fake one

Denote the part of the fake coin as $i (i \leq a)$. Beyond the remaining $a + b - i$ parts,

(1) they must avoid a , since if there's a subset sum a , assign the subset '+', this claim is wrong

(2) they must contain $a - i$ or equivalently contain b , since there must exist a condition such that the coin is fake.

Definition 6. *For convenience, we define the number of edges of a partition of n as number of edges of the graph transformed in Section 3. $Q(n, a, b)$ as the number of edges of optimal partition avoids a , but contains b*

Note that $Q(n, a, b) \geq \tau(a, n - a)$

There are only **Case 1.** and **Case 2.**, and for **Case 2.**, $i=1$ a for TS^- , thus we get the following lemma.

Lemma 4. $\tau^-(a, b) = \min\{\tau(a, b), \{i - 1 + Q(a + b - i, a, b) | 1 \leq i \leq a\}\}$

We are ready to solve OTS^- now.

Theorem 6. $OTS^-(a, b) = \text{optimal}\{OTS(a, b), \{a \text{ tree of } i-1 \text{ nodes and } OTS(a, b - i) | 1 \leq i \leq a\}\}$

Proof. We divide the proof into feasible part and optimal part.

(Feasible Part:)

$OTS(a, b)$ could find a fake coin by definition.

For $\{a \text{ tree of } i \text{ nodes and } OTS(a, b - i) | 1 \leq i \leq a\}$, the i part is real only if

1. remaining parts contain $b - i$ or
2. there is at least one unbalanced pair.

The first condition contradicts definition of $\tau(a, b - i)$.

If the second condition happens then we get a fake coins from the unbalanced pair, the graph is feasible.

(Optimal Part:) from lemma 5.2

$$\begin{aligned} \tau^-(a, b) &= \min\{\tau(a, b), \{i - 1 + Q(a + b - i, a, b) | 1 \leq i \leq a\}\} \\ &\geq \min\{\tau(a, b), \{i - 1 + \tau(a, b - i) | 1 \leq i \leq a\}\} \end{aligned}$$

is a optimal lower bound.

And $optimal\{OTS(a, b), \{a \text{ tree of } i \text{ nodes and } OTS(a, b - i) | 1 \leq i \leq a\}\}$ reaches this lower bound. □

Optimal testing solution for finding a real coin

OTS^+ is trivial. That is, comparing a coins in a group.

Theorem 7. $OTS^+(a, b) = a \text{ tree of } a + 1 \text{ nodes}$

Proof. We divide the proof into feasible part and optimal part.

(Feasible Part:)

If it results in an unbalanced pair. The heavier coin of the pair is the real one.

If it results in all balanced. This means this group is real.

(Optimal Part:)

$$\begin{aligned} \tau^+(a, b) &= \min\{\tau(a, b), \{i - 1 + Q(a + b - i, b, a) | 1 \leq i \leq b\}\} \\ &\geq \min\{\tau(a, b), \{i - 1 + \tau(a - i, b) | 1 \leq i \leq b\}\} && \text{by Theorem 2.2.} \\ &\geq \min\{\lfloor \frac{a+b}{2} \rfloor, i - 1 + \frac{a - i + b}{2}\} \\ &\geq \min\{a, a\} = a \end{aligned}$$

is a optimal lower bound.

a tree of $a + 1$ nodes reaches optimal lower bound. □

6 Open questions

6.1 Cycle happens earlier

As in theorem 4.3 and 4.4, we proved $\rho(a, b)$ would be in cycle for $b > d(d + 1)(a - 1) + (a - (a \bmod d))$. But observing ρ table in Results 3.2, ρ seems to get in cycle earlier. After computer confirmation, we got results bellow.

a	b: $\rho(a, b)$ start to cycle by theorem	b: $\rho(a, b)$ start to cycle by computer confirmation
1	1	1
2	13	1
3	15	5
4	40	10
5	29	9
6	105	29
7	43	13
8	91	22
9	57	17
10	118	28

Since the algorithm cost exponential time, we could only confirm this in small case.

But if it get in cycle earlier in general case, we can use a better(smaller) K algorithm 4.1 and run faster.

6.2 Avoidance integer partition

As in algorithm 3.1,4.1, to find an optimal solution require enumerating partitions of $a + b$ avoiding a . In the algorithms, we enumerating all partitions of $a + b$ and check if it avoids a . So the time complexity is analyzed as the (number of partitions) times checking avoidance. If exist a way to enumerate avoidance and bound it tighter, the time complexity would be better.

References

- [1] Howard D. Grossman. The Twelve-Coin Problem, *Scripta Mathematica* XI, pages 360–361, 1945.
- [2] Richard K. Guy and Richard J. Nowakowski. Coin-Weighing Problems, *The American Mathematical Monthly*, 102(2):164–167, 1995.
- [3] Lorenz Halbeisen and Norbert Hungerbühler. The General Counterfeit Coin Problem, *Discrete Mathematics*, 147(1–3):139–150, 1995.