

ANALYSIS

Subtask 1

The solution to this subtask is simply to compare each pair of coins with numbers that differ by exactly 1. Since by the problem statement there is at least one heavy and at least one light coin, there is no way that the values returned by the `weigh` function can only be 0. When we get an answer other than 0, we directly find one light and one heavy coin and, according to the results of the remaining comparisons, determine the type of each coin. The total number of function calls is $O(2^K)$.

Subtask 2

In this subtask, we know that among the set of coins there are $N - 1$ coins of equal weight and one of different weight. Our goal will be to find the different one, no matter whether it is heavy or light. If $K = 1$, this is trivial, so let's focus on the case when $K \geq 2$. Let us initially divide the set into two equal parts, which we will denote L and R . Obviously, the result of comparing L with R will be ± 1 , but it does not give us complete information because there are two possible situations. In case $L > R$ (the set L is heavier than R) these situations are:

- L consists of $\frac{N}{2}$ heavy coins, while R consists of $\frac{N}{2} - 1$ heavy and one light coin;
- L consists of $\frac{N}{2} - 1$ light and one heavy coin, while R consists of $\frac{N}{2}$ light coins.

We notice that if we choose one of the two subsets – for example L , and divide it into two parts, which we will denote L_1 and L_2 , then the result of the weighting L_1 against L_2 determines which of the two cases we are in. If the result is 0, then L consists only of coins of equal weight, and vice versa – if the result is not 0, then there is at least one light and at least one heavy coin in L .

From here, we can apply a standard binary search by dividing the remaining set into two equal parts at each iteration, comparing them, and continuing the search in the part with at least one light coin and at least one heavy coin until exactly two coins remain. The total number of calls to the `weigh` function is $O(K)$.

Subtask 3

The solution of this subtask is also based on binary search, but not in its most standard form. From the constraint that among all N coins there are at most K that are of weight different from the prevailing one, it follows that the number of "blocks" of consecutive coins of the same weight is at most $2K + 1$. We will use a binary search to find the length of each block starting from its beginning. Let a block of consecutive coins of equal weight start from coin number p . If $p = N - 1$, then obviously this block has length 1. Otherwise, we compare coins with numbers p and $p + 1$. If the result of the comparison is ± 1 , then the block length is 1 again, but if we get 0, then the length is at least 2. Continuing in the same way, if we know that from coin p starts a block of length at least 2^k , we compare the sets of coins with numbers in the intervals $[p, p + 2^k - 1]$ and $[p + 2^k, p + 2^{k+1} - 1]$. If we get an equality, we find out that coin p starts a block of length at least 2^{k+1} . If the comparison is impossible (because $p + 2^{k+1} > N$) or we get an inequality, then the block length is in the interval $[2^k, 2^{k+1} - 1]$.

By another binary search we can find the number q of the last coin of the block. We initially know that q is in the interval $[l, r]$ (obtained from the first binary search) with a middle $m = \frac{l+r}{2}$. We can weigh the coins with the numbers in the intervals $[l, m]$ and $[2l - m - 1, l - 1]$, knowing for sure that those in the latter interval have equal weight. If the result of the comparison is equality, then q will be in the interval $[m, r]$, otherwise in the interval $[l, m - 1]$. The total number of comparisons used by this solution is $O(K \log \frac{N}{K})$.

Subtask 5

Author's approach

Let's create a procedure that divides a given set of $N = 2^K$ coins into two subsets of $\frac{N}{2}$ coins each, such that the number of heavy coins in the two subsets differ at most by 1.

We apply a random partition to obtain two initial subsets L and R .

In each of these we will have fixed and unfixed coins, with only the unfixed being able to be moved to the other subset. At the beginning, all coins are unfixed.

We make sure that $L > R$ – if not, we swap the two subsets.

1. We take half of the unfixed coins from L and half of the unfixed coins from R and exchange them.
2. If $L > R$ again, we fix the coins we just exchanged.
3. If $L < R$, we return the exchanged coins to their previous subsets, fix those of the unfixed that did not participate in the exchange.
4. If $L = R$, the procedure terminates.
5. As long as there is more than one unfixed coin in each of the two subsets, we return to step 1.

Using this procedure, counting is quite easy. If the two subsets are equally heavy, then the number of heavy coins in the whole set is equal to twice the number of heavy coins in one of the two subsets. Otherwise, the number of heavy coins in the whole set is one less than twice the number of heavy coins in the heavier subset. Of course, for it to be heavier than the other, this set must contain at least one heavy coin. We need this information if in the process of recursive calculation we get a subset consisting of only one type of coins.

The solution uses $O(K^2)$ comparisons and has time complexity of $O(N \log N)$.

Proof of the correctness of the described procedure:

Let us have two disjoint subsets L and R such that $L > R$.

Let d denote the difference of the number of fixed heavy coins in L and R (initially $d = 0$).

Let x denote the difference of the number of unfixed heavy coins in L and R .

We will prove the following invariant: $|d| < x$

Basis property: the invariant holds initially unless $x = 0$, but then the procedure terminates and returns a correct result

Inductive property:

Let $x = y + z$, where y and z denote the differences among the unselected and the selected for the exchange from the unfixed coins in the two subsets

- I) $y + z - d > 0$ (by the invariant)
- II) $d + y + z > 0$ (by $L > R$)

1 case. III) $d + y - z > 0$ (by the comparison)

$$d_{new} = d - z$$

$$x_{new} = y$$

$$\text{Since III) } \Rightarrow d - z > -y \text{ and since I) } \Rightarrow |d| - z < y \Rightarrow |d - z| < y \Rightarrow |d_{new}| < x_{new}$$

2 case. III) $d + y - z < 0$ (by the comparison)

$$d_{new} = d + y$$

$$x_{new} = z$$

$$\text{Since III) } \Rightarrow d + y < z \text{ and since II) } \Rightarrow d + y > -z \Rightarrow |d + y| < z \Rightarrow |d_{new}| < x_{new}$$

Eventual falsity of the guard: The number of unfixed coins is halved at each iteration, and the algorithm terminates when only one such remains in each of the subsets.

Correctness of the postcondition: $|d| < x \leq 1$, therefore the postcondition is satisfied.

Alternative approach

There is an alternative solution using the same strategy, but with a different procedure for dividing the coins into two subsets, based on an observation including a continuous function.

Let's divide the coins into subsets again, including in one subset all coins with numbers in the interval $[l, r]$, where $r - l + 1 = \frac{N}{2}$, and in the other subset we include all the other coins. We observe that if we decide to change the current partition so that the first subset includes the coins with numbers in the interval $[l + 1, r + 1]$, the number of heavy coins in it will change by 1 at most. This is because in the change described, at most one heavy coin joins the first subset and at most one heavy coin leaves it.

Let's define the function $f(x)$ as the number of heavy coins with the numbers in the interval $[x, x + \frac{N}{2} - 1]$ and let $p = f(0)$ and $q = f(\frac{N}{2})$. We can express the preceding observation as: $|f(x) - f(x + 1)| \leq 1$ for $\forall x \in [0, \frac{N}{2} - 1]$. Moreover, it is clear that either $p \leq \lfloor \frac{H}{2} \rfloor \leq q$ or $q \leq \lfloor \frac{H}{2} \rfloor \leq p$, where H is the number of heavy coins. In other words, the number of heavy coins in one half is less than or equal to half of them, and in the other half it is greater than or equal to half of them. Therefore, with absolute certainty $f(x)$ will obtain the value $\lfloor \frac{H}{2} \rfloor$ for some $x \in [0, \frac{N}{2}]$. It is the value of this x that we need to find an appropriate partition of the coins into two subsets.

Whether the value of $f(x)$ is greater than, less than, or equal to $\frac{H}{2}$ we can easily determine with a single call to the `weigh` function, passing as parameters the subsets of coins with numbers in the intervals $[x, x + \frac{N}{2} - 1]$ and $[0, x - 1] \cup [x + \frac{N}{2}, N - 1]$.

Now we will use a binary search on the interval $[l, r]$, with initial values of $l = 0$ and $r = \frac{N}{2}$. As we have already shown, if $m = \min(f(l), f(r))$ and $M = \max(f(l), f(r))$ and $m \leq \lfloor \frac{H}{2} \rfloor \leq M$, then certainly there exists $x \in [l, r]$ such that $f(x) = \lfloor \frac{H}{2} \rfloor$. We check the value of $f(x)$ in the middle of the interval and choose that half of it in which the condition $m \leq \lfloor \frac{H}{2} \rfloor \leq M$ still holds. Thus, we will eventually find a suitable value for x , giving us a partition in which the number of heavy coins in the two subsets differs by at most 1. The complexity in terms of number of queries and time is comparable to that of the author's solution.

Author: Dobrin Bashev