3 Scallion Pancake Party

We describe two different solutions to this task.

3.1 Solution 1

For convenience, we will call the guest in room i "guest i".

The core idea is to partition the bags into N distinct sets, with each set assigned to a specific guest according to their room number. This assignment allows each guest to relay information to the guest outside through their respective sets. However, implementing this approach effectively requires addressing several technical challenges, which we explore in the following subsections.

3.1.1 Subtask 2

Starting with Subtask 2, let's explore how the guest outside can correctly recover the permutation P.

Since every guest knows their room number, we can naturally partition the bags into N sets, assigning one set to each guest based on their room number. This allows each guest to communicate information to the outside guest by selectively eating bags only from their assigned set.

One partitioning strategy is to assign the $(i+1)$-th occurrence of every flavor to guest i. By counting the occurrences of each flavor, all guests can easily reach a consensus on which bags belong to them.

Furthermore, as each guest now has exactly one bag of each flavor in their assigned set, guest i can simply eat all bags except the one with flavor $P[i]$.

Consequently, the guest outside can identify which set each remaining bag belongs to and determine P based on the remaining flavor.

3.1.2 Subtask 3

In this subtask, we no longer have access to the room numbers. If we wish to apply the "partition and eat all but one" strategy from subtask 2, each guest must first determine their own room number by observing what previous guests have eaten.

In subtask 3, the bags follow a convenient structure: their flavors are a concatenation of N permutations. Thus, we can partition the bags by their indices such that the i-th bag is assigned to the $\left\lfloor \frac{i}{N} \right\rfloor$-th set.

But how can a guest identify their room number? Recall the strategy: the guest in room i eats all bags in their set except for the one with flavor $P[i]$. In an optimistic scenario, guest i sees that bags with indices in the range $[(i-1) \cdot N, i \cdot N)$ have already been eaten, allowing them to infer that their room number is **at least** i.

Specifically, if the flavor of bag $i \cdot N$ is not $P[i]$, guest i can eat it. Subsequent guests will then observe this and realize their room number must be greater than i. A problem arises only when the flavor of bag $i \cdot N$ is exactly $P[i]$.

Let's simulate this case. Assume this first occurs for room i. Guest $i+1$ will find that bag $i \cdot N$ remains uneaten and mistakenly conclude that their room number is i. Consequently, bag $i \cdot N$ will be eaten unexpectedly by guest $i+1$. However, once guest i eats bag $i \cdot N + 1$, guest $i+1$ will notice this and realize their mistake. Furthermore, since bag $i \cdot N$ was already eaten by guest $i+1$, all

subsequent guests can correctly recognize that their room number is at least $i+1$. By induction, it can be proven that this process succeeds, with one exception: all bags belonging to the i-th set are eaten **if and only if** the flavor of bag $i \cdot N$ is exactly $P[i]$.

Finally, we recover the permutation P as the outside guest. If any bag remains uneaten in the i-th set, $P[i]$ is determined by its flavor. For sets where all bags were eaten, we use the "if and only if" condition: the flavor of bag $i \cdot N$ must have been $P[i]$. Thus, all values of P can be successfully recovered.

3.1.3 Subtask 4

We continue applying the "partition and eat all but one" strategy from subtask 2. With $K = N$, we have almost sufficient capacity to notify all guests of their room numbers using only the first bag. Specifically, each guest eats one slice from the first bag; thus, guest i will observe that exactly $i-1$ slices have already been eaten, identifying their room number as i.

A problem arises if a guest's assigned flavor $P[i]$ matches the flavor of this first bag. Since each flavor appears exactly once per permutation, this collision occurs exactly once. Suppose this happens for the guest j. All guests after them will observe one fewer eaten slice than expected, leading them to underestimate their room number by one (mistaking it for $j, j+1, ...$ instead of $j+1, j+2, ...$), while guests 1 through j will identify their numbers correctly.

For convenience, we ignore any subsequent bags that share the same flavor as the first bag. These bags are reserved solely for transmitting room number information; no further action is required when a guest encounters them (though they are welcome to eat from them if they are particularly hungry).

We then partition the remaining bags into N sets based on flavor occurrences, as in subtask 2. For guests in rooms $i < j$, the strategy works perfectly. However, the guest in room j must notify subsequent guests of the offset error. Since $K \geq 2$, the guest in room j can eat exactly one slice from the first bag of their assigned set. Subsequent guests will detect this "missing slice" signal and, upon receiving it, increment their inferred room number by one to correct the mistake.

Finally, the outside guest can recover all values of P except for $P[j]$. Since P is a permutation, $P[j]$ can then be uniquely determined by the process of elimination.

3.1.4 Subtask 6

We skip subtask 5 as it is intentionally left for some suboptimal solutions.

We now generalize the strategy from subtask 3. Once again, we partition the bags into N sets based on their occurrences, where set i consists of the $(i+1)$-th occurrence of every flavor. Each guest continues to follow the established strategy: eat all bags belonging to their assigned set, except for one.

We can generalize the observation made in subtask 3 as follows:

Observation 3.1. If a guest observes that the i-th occurrence of any flavor has been completely eaten, they can infer that their room number is at least i.

However, relying solely on this observation leads to problems, e.g., when the very first bag of set i happens to have the flavor $P[i]$.

The primary issue is that guest $i+1$ will see this uneaten bag and mistake their room number for i, leading them to unexpectedly eat the bag in set i with flavor $P[i]$. While guest i can detect this mistake the moment they encounter the bag, unlike in subtask 3, they cannot always notify guest $i+1$ simply by eating the entirety of the next bag. Nevertheless, we claim that a robust strategy for the guest outside can resolve this ambiguity by examining the next incoming uneaten bag according to these three cases:

1. If the bag belongs to set i (i.e., it is another flavor's $(i+1)$-th occurrence): Guest i can directly eat the entire bag. Guest $i+1$ will see this and immediately realize their mistake.
2. If the bag is the next occurrence of the same flavor: This bag will be eaten by guest $i+2$. This cascading effect is generally acceptable, though it may result in the final occurrence of this flavor remaining uneaten. We will explain how the outside guest resolves this ambiguity later.
3. If the bag belongs to a previous set (i.e., it is an occurrence $< i+1$): This bag was intentionally left uneaten to notify the outside guest of a previous P value, just simply ignore it.
   - If you do not realize that simply ignoring Case 3 is sufficient for the guest outside, you might be tempted to implement a more complex signaling system. For instance, one could utilize the $K \geq 3$ condition to notify guest $i+1$ of their mistake by eating a single slice from the bag, and guest $i+1$ will need to eat an additional slice to cancel the signal.

Under this strategy, we can establish the following:

Observation 3.2. Let B_i be the bag of flavor F belonging to set i, where F is a flavor not equal to any of $P[0], P[1], ..., P[i-1]$. Guest i will always successfully deduce their correct room number by the time they encounter bag B_i.

Proof. We prove this by contradiction and induction. Assume the statement holds for guests 0 through $i-1$. Suppose guest i still does not know their room number when they reach bag B_i. This implies that the previous occurrence of flavor F was not eaten by guest $i-1$; otherwise, by Observation 3.1, guest i would have known their room number.

However, this is a contradiction to the induction hypothesis, as flavor F is not equal to any of $P[0], P[1], ..., P[i-2]$, and guest $i-1$ should eat it according to our strategy. □

Finally, we describe the strategy for the outside guest to recover the permutation P. Assume we have successfully recovered $P[0]$ through $P[i-1]$, and we now aim to deduce $P[i]$:

- If some bags in set i remain uneaten: By Observation 3.2, guest i eventually knew their room number and ate the remaining bags in set i. Thus, the flavor of the last uneaten bag in set i must be exactly $P[i]$.
- If all bags in set i are eaten: The flavor of the first bag in set i that does not belong to $\{P[0], ..., P[i-1]\}$ must be $P[i]$.
  - If this first available flavor were not $P[i]$, guest i would have recognized their room number upon seeing it (since it avoids all previous P values) and would have eaten it, leaving the actual $P[i]$ bag uneaten. Furthermore, all subsequent guests would correctly deduce that their room numbers are strictly greater than i, ensuring none of them would mistakenly eat this remaining $P[i]$ bag in set i. Thus, the $P[i]$ bag would definitely remain uneaten, resulting in a contradiction.

3.2 Solution 2

Let $A_{f,i}$ denote the index of the $(i+1)$-th bag that contains flavor f.

The core idea is to force a guest with forbidden flavor P_i to only eat pancakes from bags with flavor $(P_i+1) \bmod N$.

Unless specified, eating a bag means eating all remaining slices from the bag.

3.2.1 Subtask 1

Guest i always eats bag $A_{(P_i+1) \bmod N,0}$. They only eat bag $A_{(P_i+1) \bmod N,1}$ if $A_{P_i,0} < A_{(P_i+1) \bmod N,0}$ and bag $A_{P_i,0}$ has not been eaten.

For each guest i, the condition $A_{P_i,0} < A_{(P_i+1) \bmod N,0}$ can be easily checked the moment they receive bag $A_{(P_i+1) \bmod N,0}$. They simply check whether a bag with flavor P_i has appeared before. Since $A_{(P_i+1) \bmod N,0} < A_{(P_i+1) \bmod N,1}$, they will know whether they should eat $A_{(P_i+1) \bmod N,1}$ by the time they receive it.

For example, suppose $F = [1, 0, 0, 1]$.

- If $P = [0, 1]$, then guest 0 only eats bag 0 and guest 1 only eats bag 1.
- If $P = [1, 0]$, then guest 0 eats bags $1, 2$ and guest 1 only eats bag 0.

For the guests outside, they can clearly tell which of $A_{0,0}, A_{1,0}$ comes first. Suppose $A_{f,0} < A_{1-f,0}$ for some $f = 0$ or 1.

Observe that bag $A_{f,1}$ will never be eaten. The guests will guess P based on whether bag $A_{1-f,1}$ is empty. If bag $A_{1-f,1}$ is empty, then $P = [1-f, f]$. Otherwise, $P = [f, 1-f]$.

3.2.2 Subtask 2

In this subtask, each guest i knows the value of i.

Guest i simply eats bag $A_{(P_i+1) \bmod N,i}$.

For the guests outside, they will see exactly N empty bags. Each empty bag $A_{f,i}$ tells them the value of $P[i]$, which is just $(f + N - 1) \bmod N$.

3.2.3 Subtask 3

In this subtask, $A_{0,0}, A_{1,0}, ..., A(N-1, 0)$ are the first N bags.

For the first N bags, guest i only eats $A_{(i+1) \bmod N,0}$.

After the first N bags, guest i will see exactly $i+1$ empty bags. So each guest can learn their room index once they reach this point.

So the strategy for the remaining $N(N-1)$ bags are:

- For each $0 \le i \le N-2$, guest i eats bag $A_{(P_i+1) \bmod N,i+1}$.
- Guest $N-1$ does nothing.

For the guests outside, they can determine P based on empty bags from the last $N(N-1)$ bags.

Each empty bag $A_{f,i}, i \ge 1$ tells them the value of $P[i-1]$, which is just $(f + N - 1) \bmod N$.

There will be exactly one flavor f without empty bags. Which gives $P[N-1] = (f + N - 1) \bmod N$. Alternatively, since there will be exactly one value, $P[N-1]$, missing in P, it can also be determined using the filled values.

3.2.4 Subtask 6

We skip subtasks 4 and 5 and jump straight into the full solution.

Without extra info or specific bag sequence patterns, it seems quite hard for guests to first learn their room index and encode that information with later bags. To solve this issue, perhaps we can extend the idea for subtask 1.

First of all, each guest i will always eat bag $A_{(P_i+1) \bmod N,0}$.

For each guest i, Let e_{P_i} be the number of flavors f such that $A_{f,0} < A_{(P_i+1) \bmod N,0}$ and bag $A_{f,0}$ is empty. They can determine e_{P_i} once they receive bag $A_{(P_i+1) \bmod N,0}$.

Then, each guest i encodes e_{P_i} using $A_{(P_i+1) \bmod N,1}, A_{(P_i+1) \bmod N,2}, ..., A_{(P_i+1) \bmod N,N-1}$. More specifically, they eat bag $A_{(P_i+1) \bmod N,j}$ if and only if the j-th bit of e_{P_i} is 1. Since $e_{P_i} \leq N-1$, the number of bags is enough to encode the bit representation of e_{P_i}.

For the guests outside, they can easily obtain e_i by the binary representation of $A_{(i+1) \bmod N,1}, A_{(i+1) \bmod N,2}, ..., A_{(i+1) \bmod N,N-1}$. We describe how to reconstruct P using $e_0, e_1, ..., e_{N-1}$.

Let $f_0, f_1, ..., f_{N-1}$ be the order of flavors such that $A_{f_0,0} < A_{f_1,0} < ... < A_{f_{N-1},0}$. In other words, f_i is the $(i+1)$-th flavor that appeared for the first time in the sequence of bags.

Observe that for each $1 \leq i \leq N-1$, e_{f_i} is precisely the number of flavors f among $f_0, f_1, ..., f_{i-1}$ such that f appears earlier than f_i in P. This means we can do something like insertion sort to determine P.

More specifically, we start with $P = [f_0]$. Then, for each i from 1 to $N-1$, we insert f_i to P such that it becomes the $\left(e_{f_i} + 1\right)$-th element in P. In the end, we will have the correct P.

3.3 Remarks

- The score of the last subtask originally had partial scoring: Higher score is awarded if the total number of slices remaining is lower. We decided to remove the partial scoring since it was not an important part of the solution, and it barely changes the difficulty of the problem.
- There was a "harder" version of the task: Reduce the number of bags of each flavor to 6 instead of N. This cuts out solution 1 while solution 2 still passes since it requires just $\lceil \log_2 N \rceil + 1 \leq 6$ bits of information for each flavor. We ultimately decided to keep the current version so that the entire problem set wouldn't be too hard. (The problem set is already hard enough.)