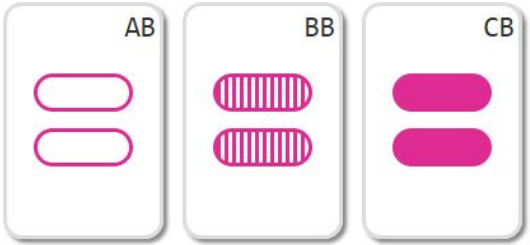
S263 - Set

Daniel Hsieh {QwertyPi}

2026-02-14

Background



Problem Idea by `QwertyPi`

Preparation by `QwertyPi`, `firewater`, `WongChun1234`

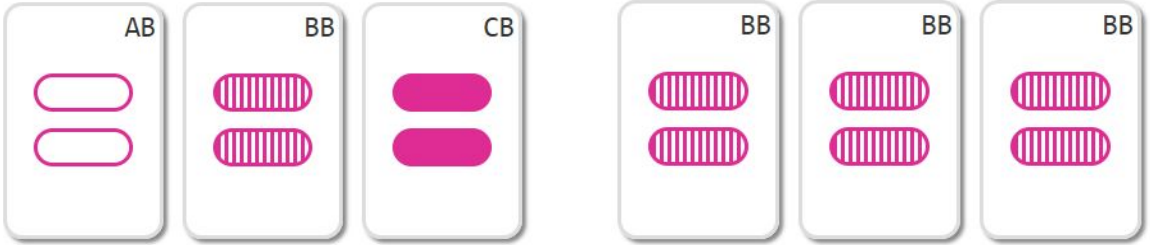
Inspired by the senior heat question that inspired by the game SET

6. Complete the program `f(a, b, c)` so that, for integers $1 \leq a, b, c \leq 3$, it returns `True` (C++: `true`) if and only if `a`, `b`, `c` are either all the same or all different.
完成程式 `f(a, b, c)`，對於滿足 $1 \leq a, b, c \leq 3$ 的整數，當且僅當 `a`、`b`、`c` 全部相同或全部不同時回傳 `True` (C++: `true`)。

Solution: `(a + b + c) % 3 == 0`

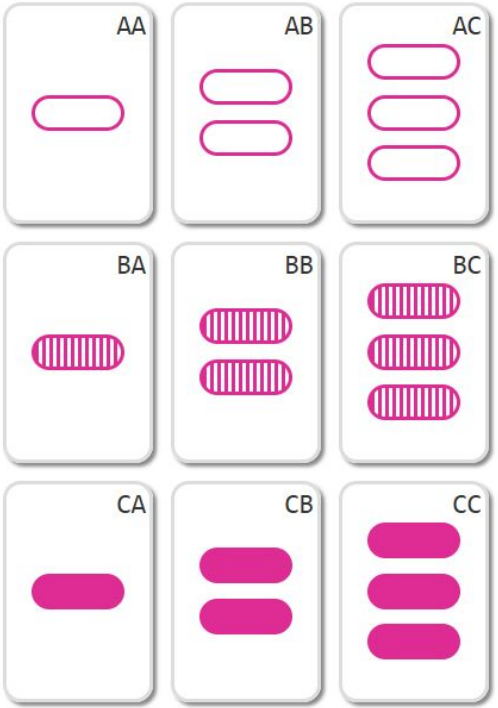
Notations

- **Card**: **L features, each 3 possibilities**
  - encoded by a length-L ABC string
- **Deck**: consists of each of the 3^K unique cards
- **Set**: a collection of 3 cards, each feature either **all same or all different**



Valid Sets

Complete Deck of L = 2



Problem Restatement

Given **N (≤ 60000) cards**, each of **L (≤ 9) features** on a table.
Partition the cards into **valid sets** by adding **complete decks** onto the table.

It is guaranteed that the game can be won <u>in all the test cases</u>.

- A simple counterexample is `A A B` - it is not solvable no matter how many decks you added
- In all the testcases ≠ Over all the possibilities

You cannot add too many complete decks.

- Total number of cards = **$N + M \times 3^L$ must not exceed 6×10^5**, M the number of decks added
- Mostly affect the last subtask

Subtasks

For all cases: $3 \le N \le 60000$, $1 \le L \le 9$

Subtask	Score	L	Additional Constraints
1	7	≤ 9	Given cards form a complete deck
2	12	$= 1$	/
3	16	≤ 2	Exactly one card starts with `B` / `C`
4	21		/
5	12	≤ 4	/
6	32	≤ 9	/

Statistics

```
100 points  0 +  0 +  0 +  1 =  1
 68 points  0 +  0 +  0 +  1 =  1
 56 points  0 +  0 +  1 +  2 =  3
 35 points  0 +  1 +  2 +  2 =  5
 28 points  0 +  0 +  1 +  1 =  2
 19 points  7 + 11 + 11 +  3 = 32
 12 points  0 +  3 +  3 +  0 =  6
  7 points  0 +  1 +  1 +  0 =  2
  0 points 10 +  8 +  0 +  0 = 18
```

First and Only Solve by **dbsculver0412** at **2h 35m 15s**.

Subtask 1 (7%, Given cards form a deck)

Note that the cards `XA`, `XB`, `XC` forms a set where `X` is a string of length L - 1.

Since every card appears exactly once, we can form all sets by looping through all possibilities of `X`.

How can we do that?
Method 1: Use recursion to enumerate
Method 2: Simply input and sort the cards

Number of extra decks: M = 0

Subtask 2 (12%, L = 1)

From now on, we denote `cnt[X]` be the number of appearance of the card `X`.

Observation 1. When L = 1, the game can be won if and only if

`cnt[A] % 3 = cnt[B] % 3 = cnt[C] % 3`

What if say `cnt[A] % 3` ≠ `cnt[B] % 3` initially? Observe that if we

- form a set of `X X X (X = A, B, C)`: none of `cnt[X] % 3` is changed
- form a set of `A B C`: all `cnt[X] % 3` change to `(cnt[X] - 1) % 3`
- request a deck of `A B C`: all `cnt[X] % 3` change to `(cnt[X] + 1) % 3`

Therefore, `cnt[A] % 3` ≠ `cnt[B] % 3` still holds no matter what

Which means `cnt[A]` and `cnt[B]` cannot both be 0!

Subtask 2 (12%, L = 1)

From now on, we denote `cnt[X]` be the number of appearance of the card `X`.

Observation 1. When L = 1, the game can be won if and only if

`cnt[A] % 3 = cnt[B] % 3 = cnt[C] % 3`

It is guaranteed that the game can be won, so we can assume this condition.

The game can be won by forming

- `(cnt[X] // 3)` sets of `X X X` where `X = A, B, C`
- `(cnt[A] % 3)` sets of `A B C`

Number of extra decks: M = 0

Subtask 3 (16%, L = 2, Exactly one card starts with B / C)

Observation 2. Given any two cards X and Y, we can find the **unique third card** Z such that X Y Z forms a set.

Card X	A	B	A	B	C
Card Y	A	B	C	C	B
Card Z	?	?	?	?	?

Subtask 3 (16%, L = 2, Exactly one card starts with B / C)

Observation 2. Given any two cards `X` and `Y`, we can find the **unique third card** `Z` such that `X Y Z` forms a set.

Card X	A	B	A	B	C
Card Y	A	B	C	C	B
Card Z	A	B	B	A	A

You can compute that position by position (directly from definition):

- Two equal chars then keep it
- Two different chars then pick the third one

Subtask 3 (16%, L = 2, Exactly one card starts with B / C)

There is only one card starts with `B / C`, say `BY` and `CZ`.

Therefore, we can find the third card `AX` where `AX BY CZ` forms a set.

Case I If `cnt[AX] > 0`, then after playing the set `AX BY CZ`, we are left with all cards start with `A` - that means we can simply reuse the algorithm for L = 1!

Case II What if `cnt[AX] = 0`? We can simply **request a deck** and play the sets `BA BB BC` and `CA CB CC` - then we can proceed exactly as Case I!

Number of extra decks: M = 1

General Case (21% / 12% / 32%, L = 2 / L ≤ 4 / L ≤ 9)

From now on, our target would be the general case.

As you can see from subtask 3, one of the tricky points is: how to use all the cards exactly and avoid the card count going to the negative?

Observation 3. We can actually **ignore whether the card count goes to the negative**, and just make all the **`cnt[X]` divisible by 3** for all card `X`. We then fix the negative counts by adding decks and form sets of identical cards.

General Case (21% / 12% / 32%, L = 2 / L ≤ 4 / L ≤ 9)

Observation 3. We can actually **ignore whether the card count goes to the negative**, and just make all the **`cnt[X]` divisible by 3** for all card `X`. We then fix the negative counts by adding decks and form sets of identical cards.

```
# Step 1: omitted, magical way to make cnt[X] % 3 == 0 for all X (allow negative)
M := max(0, -min_X(cnt[X])) # Step 2: request decks to make cnt[X] ≥ 0 for all X
For all choices of X:
    S_X = (cnt[X] + M) // 3 # Step 3: form (cnt[X] + M) // 3 sets of X, noting
    Play the set X X X for S_X times # both cnt[X] and M are divisible by 3
```

We will **assume step 2 & 3 are done** and focus on step 1 from now on.

General Case (21% / 12% / 32%, L = 2 / L ≤ 4 / L ≤ 9)

```
# Step 1: omitted, magical way to make cnt[X] % 3 == 0 for all X (allow negative)
M := max(0, -min_X(cnt[X])) # Step 2: request decks to make cnt[X] ≥ 0 for all X
For all choices of X:
    S_X = (cnt[X] + M) // 3 # Step 3: form (cnt[X] + M) // 3 sets of X, noting
    Play the set X X X for S_X times # both cnt[X] and M are divisible by 3
```

Let's say you successfully form sets with **`cnt'[X]` card `X`**

- Note that **`M ≤ max(cnt'[X])`**, since the initial count always nonnegative
- Intuitively - as long as each card is not played too many times, it is ok

General Case (21% / 12% / 32%, L = 2 / L ≤ 4 / L ≤ 9)

```
# Step 1: omitted, magical way to make cnt[X] % 3 == 0 for all X (allow negative)
```

How can we do step 1, perhaps not take the same card too many times?
There are many solutions to do this and lead to different number of decks requested - we will introduce some of them in the following slides

The **upper limit of extra decks** in each subtask:
Subtask 4 (L = 2): M ≤ 60000
Subtask 5 (L ≤ 4): M ≤ 6666
Subtask 6 (L ≤ 9): **M ≤ 27**

Constraints

- **N ≤ 60000**
- **N + M × 3^L must not exceed 6×10^5**

Solution 1 (Random Count) M = O($3^{3^{\wedge}L}$)

```
while cnt is not all divisible by 3:
    X, Y := any random card
    Z := the unique card that forms a set with X, Y
    play the set X Y Z
    subtract 1 from each of cnt[X], cnt[Y] and cnt[Z]
```

Actually - you can simply form sets randomly!

Since each `cnt[X] % 3` are either 0, 1 or 2, there are only $3^{3^{\wedge}L}$ states.

You can make all `cnt[X] % 3 == 0` by forming on average $3^{3^{\wedge}L}$ sets.

Solution 2 (Card Pair) $M = O(3^L)$

```
while cnt is not all divisible by 3:
    X, Y := any random card where X ≠ Y, cnt[X] % 3 ≠ 0 and cnt[Y] % 3 ≠ 0
    Z := the unique card that forms a set with X, Y
    play the set X Y Z
    subtract 1 from each of cnt[X], cnt[Y] and cnt[Z]
```

Surprisingly, if you pair up existing cards only, the number of decks on average requested would becomes $O(3^L)$ instead.

Note that if `cnt[Z] % 3 ≠ 0`, then total sum of `cnt % 3` would decrease by 3. Otherwise the total sum of `cnt % 3` would be unchanged. Heuristically, therefore, it would work in $M = O(3^L)$ decks!

Solution 3 (Repeat Elimination) $M = O(2^L)$

One of the intuitive ideas maybe - can we reduce the “space” of possible cards repeatedly? What if all the cards starts with `A`? Or with `A` and `B` only?

`cnt[AA]`	`cnt[AB]`	`cnt[AC]`
`cnt[BA]`	`cnt[BB]`	`cnt[BC]`
`cnt[CA]`	`cnt[CB]`	`cnt[CC]`

Solution 3 (Repeat Elimination) $M = O(2^L)$

	A_	B_	C_
_A	?	?	?
_B	?	?	?
_C	?	?	?

Solution 3 (Repeat Elimination) $M = O(2^L)$

	A_	B_	C_
_A	?	?	?
_B	?	?	?
_C	0	?	?

Solution 3 (Repeat Elimination) $M = O(2^L)$

	A_	B_	C_
_A	?	?	?
_B	?	?	?
_C	0	0	?

Solution 3 (Repeat Elimination) $M = O(2^L)$

	A_	B_	C_
_A	?	?	?
_B	?	?	?
_C	0	0	0

Solution 3 (Repeat Elimination) $M = O(2^L)$

	A_	B_	C_
_A	?	?	?
_B	?	?	0
_C	0	0	0

Solution 3 (Repeat Elimination) $M = O(2^L)$

	A_	B_	C_
_A	?	?	0
_B	?	?	0
_C	0	0	0

Solution 3 (Repeat Elimination) $M = O(2^L)$

	A_	B_	C_
_A	?	?	0
_B	0	?	0
_C	0	0	?

Solution 3 (Repeat Elimination) $M = O(2^L)$

	A_	B_	C_
_A	?	?	0
_B	0	0	0
_C	0	0	?

Solution 3 (Repeat Elimination) $M = O(2^L)$

	A_	B_	C_
_A	?	0	?
_B	0	0	0
_C	0	0	?

Solution 3 (Repeat Elimination) $M = O(2^L)$

	A_	B_	C_
_A	?	0	?
_B	0	0	0
_C	0	0	?

Solution 3 (Repeat Elimination) $M = O(2^L)$

If `cnt[_B] % 3 ≠ cnt[_C] % 3`, then impossible to win!

	A_	B_	C_
_A	?	0	?
_B	0	0	0
_C	0	0	0

Solution 3 (Repeat Elimination) $M = O(2^L)$

	A_	B_	C_
_A	?	0	?
_B	0	0	0
_C	0	0	0

Solution 3 (Repeat Elimination) $M = O(2^L)$

If `cnt[A_] % 3 ≠ cnt[B_] % 3` or `cnt[C_] % 3 ≠ cnt[B_] % 3`, then impossible to win!

	A_	B_	C_
_A	0	0	0
_B	0	0	0
_C	0	0	0

Solution 3 (Repeat Elimination) $M = O(2^L)$

```
Step 1. remove all cards _C with _A _B _C
Step 2. remove all cards C_ with A_ B_ C_
Step 3. remove all cards _B with _A _B CC
Step 4. remove all cards BA with AA BA CA
```

We can, in fact, generalise this to arbitrary L! But we will need exponential no. of `C...C`, and so this only works for $L \leq 4$.

	A_	B_	C_
_A	0	0	0
_B	0	0	0
_C	0	0	0

Winning Condition

Define $\texttt{cnt}_K\texttt{[X]}$ be the number of cards with the K^{th} position being `X`

The game can be won if and only if

$$\texttt{cnt}_K\texttt{[A] \% 3 == cnt}_K\texttt{[B] \% 3 == cnt}_K\texttt{[C] \% 3}$$

for **every position K**.

Winning Condition

The game can be won if and only if

$$\text{cnt}_K[A] \% 3 == \text{cnt}_K[B] \% 3 == \text{cnt}_K[C] \% 3$$

for **every position K**.

If any position violates this condition, we can consider that position only and we cannot even win with L = 1, and so the game cannot be won.
Otherwise we can apply solution 3 (repeat elimination) to win the game.

Solution 4 (Divide and Conquer) M = O(L)

Observation 4. Suppose we treat `A` as 0, `B` as 1 and `C` as 2. Then the game can be won iff the sum of each position are all divisible by 3 and the total number of cards is divisible by 3. That is,

$$(\text{cnt}_K[A] \times 0 + \text{cnt}_K[B] \times 1 + \text{cnt}_K[C] \times 2) \% 3 == 0$$

For each K.

In fact, this is equivalent to

$$\text{cnt}_K[A] \% 3 == \text{cnt}_K[B] \% 3 == \text{cnt}_K[C] \% 3$$

Since $\text{cnt}_K[A] + \text{cnt}_K[B] + \text{cnt}_K[C] = N$ is divisible by 3!

Solution 4 (Divide and Conquer) M = O(L)

Another interpretation: If we treat each card as a ternary number and defines **Ternary XOR**, then the game can be won iff the ternary XOR (SUM % 3) of every cards is 0, and the number of cards is divisible by 3!

Card X	A	B	A	B	C
Card Y	A	B	C	C	B
Card Z	A	B	B	A	A

X	0	1	0	1	2
Y	0	1	2	2	1
Z	0	1	1	0	0
SUM % 3	0	0	0	0	0

All 0 → solvable!

Solution 4 (Divide and Conquer) M = O(L)

```
def solve(L, prefix, cnt): # cnt is the frequency array of cards
    if L == 1:
        # Base Case: solve directly and returns
    # TODO: make each of cnt[3^(L-1)*i..3^(L-1)*(i+1)-1] solvable
    for i from 0 to 2: # Divide-and-Conquer
        solve(L - 1, prefix + ('A' + i), cnt[3^(L-1)*i..3^(L-1)*(i+1)-1])
```

Intuitively, we can split the cards into three piles according to their first letter. If we somehow make each of the piles solvable, then we can solve recursively.

Solution 4 (Divide and Conquer) M = O(L)

Consider the following case. After splitting the piles, the piles are not solvable.

Card 1	A	B	C
Card 2	A	B	C
Card 3	B	B	C
Card 4	B	A	C
Card 5	C	C	C
Card 6	C	B	C
Sum % 3	0	0	0

All 0 → solvable!

Card 1	0	1	2
Card 2	0	1	2
SUM % 3	0	2	1

Card 3	1	1	2
Card 4	1	0	2
SUM % 3	2	1	1

Not all 0 → not solvable :(

Card 5	2	2	2
Card 6	2	1	2
SUM % 3	1	0	1

Solution 4 (Divide and Conquer) M = O(L)

We can “remove” one card into each pile, then each pile becomes solvable!

Notice that the three cards removed must form a valid set.

Don’t forget that card count can actually go negative.

All 0 → solvable!

Card 1	0	1	2
Card 2	0	1	2
Removed Card X	-0	-2	-1
SUM % 3	0	0	0

Card 3	1	1	2
Card 4	1	0	2
Removed Card Y	-2	-1	-1
SUM % 3	0	0	0

Removed Card X	0	2	1
Removed Card Y	2	1	1
Removed Card Z	1	0	1
SUM % 3	0	0	0

Card 5	2	2	2
Card 6	2	1	2
Removed Card Z	-1	-0	-1
SUM % 3	0	0	0

Solution 4 (Divide and Conquer) M = O(L)

To fix the issue where the number of cards not a multiple of 3, you can simply take away cards of form XAA...AA, since it is 0 if seen in ternary.

For each layer of the divide-and-conquer, we take away the same card at most 3 times. Therefore, each card is taken away at most 3(L - 1) times in total, sufficient for full.

Fact: The bound of 3(L - 1) is very loose, and you can optimise this easily with some tricks. Can you think of any?

Solution 5 (Wormhole Folding) M = O(L)

Notice that if we remove all cards starts with `B` or `C`, that is just the case of L - 1.

Therefore, by requesting several decks and make `cnt[B_] = cnt[C_]`, we can pair up cards of form `B_` and of form `C_` randomly. Only cards of form `A_` remains, and so we can simply repeat the process!

Heuristically, this passes M ≤ 3L quite easily.

Remarks

- Constructive in general: If you have no idea how to proceed, think what you can do first, then try to reduce the task to what you can do.

- Think before you implement: Always find generic ways to avoid handling cases, and reuse written utility functions. Some implementation method might be much easier than the other (e.g. cards vs frequency array).

- Unsolved Question: Is there a deterministic solution that involves constant number of extra decks for arbitrary L? Feel free to talk to me if you solve this :)