Snap Dragons III: Binary Snap

Input File: *snapin.txt*
Output File: *snapout.txt*

Have you ever heard of Melodramia, my friend? It is a land of forbidden forests and boundless swamps, of sprinting heroes and dashing heroines. And it is home to two dragons, Rose and Scarlet, who, despite their competitive streak, are the best of friends.

Rose and Scarlet love playing *Binary Snap*, a game for two players. The game is played with a deck of cards, each with a numeric label from 1 to N. There are two cards with each possible label, making $2N$ cards in total. The game goes as follows:

- Rose shuffles the cards and places them face down in front of Scarlet.
- Scarlet then chooses either the top card, or the second-from-top card from the deck and reveals it.
- Scarlet continues to do this until the deck is empty. If at any point the card she reveals has the same label as the previous card she revealed, the cards are a *Dragon Pair*, and whichever dragon shouts 'Snap!' first gains a point.

After many millenia of playing, the dragons noticed that having more possible Dragon Pairs would often lead to a more exciting game. It is for this reason they have summonned you, the village computermancer, to write a program that reads in the order of cards in the shuffled deck and outputs the maximum number of Dragon Pairs that the dragons can find.

Input

The first line of input will contain a single integer, N. The following $2N$ lines will each contain an integer, v_i (where $1 \leq v_i \leq N$), representing the label of the ith card from the top of the shuffled deck.

Output

Output should consist of a single integer: the maximum number of Dragon Pairs Scarlet can deal in a game of Binary Snap with the deck in the given order.

Sample Input

```
4
1
4
1
2
3
2
3
4
```

Sample Output

```
3
```

Explanation

We can make Dragon Pairs from all of the cards except the 4s with the following sequence of moves:

- `1 4 1 2 3 2 3 4`: The deck of cards we are given, first we deal the top '`1`' from the deck.
- `4 1 2 3 2 3 4`: Next we deal the '`1`' from second top position in the deck. This gives us a Dragon Pair.
- `4 2 3 2 3 4`: Next we deal the '`4`' from the top.
- `2 3 2 3 4`: Next we deal the '`2`' from the top.
- `3 2 3 4`: Next we deal the '`2`' second from the top. This gives us another Dragon Pair.
- `3 3 4`: Next we deal the '`3`' from the top.
- `3 4`: Next we deal the '`3`' from the top. This gives us another Dragon Pair.
- `4`: Finally we deal the '`4`' from the top.

This gives us 3 Dragon Pairs in total, so we must output 3.

Subtasks & Constraints

For all subtasks, $1 \leq N \leq 100\,000$.

- For Subtask 1 (30 marks), $N \leq 10$.
- For Subtask 2 (20 marks), $N \leq 100$ and cards with the same labels are *at least* 3 spaces apart (i.e. if $v_i = v_j$, then $|i - j| \geq 3$).
- For Subtask 3 (20 marks), $N \leq 100$.
- For Subtask 4 (30 marks), there are no further constraints.

Scoring

The score for each input scenario will be 100% if the correct answer is written to the output file, and 0% otherwise.