**Gena Playing Hanoi**

* [**Problem**](https://www.hackerrank.com/challenges/gena/problem)
* [Submissions](https://www.hackerrank.com/challenges/gena/submissions)
* [Leaderboard](https://www.hackerrank.com/challenges/gena/leaderboard)
* [Discussions](https://www.hackerrank.com/challenges/gena/forum)
* [Editorial](https://www.hackerrank.com/challenges/gena/editorial)

The [Tower of Hanoi](https://en.wikipedia.org/wiki/Tower_of_Hanoi) is a famous game consisting of  rods and a number of discs of incrementally different diameters. The puzzle starts with the discs neatly stacked on one rod, ordered by ascending size with the smallest disc at the top. The game's objective is to move the entire stack to another rod, obeying the following rules:

1. Only one disc can be moved at a time.
2. Each move consists of taking the topmost disc from a stack and moving it to the top of another stack.
3. No disc may be placed on top of a *smaller* disc.

Gena has a modified version of the [Tower of Hanoi](https://en.wikipedia.org/wiki/Tower_of_Hanoi). His *Hanoi* has  rods and  discs ordered by ascending size. He made a few moves (following the rules above), but stopped and lost his place. He wants to restore the tower to its original state by making valid moves. Given the state of Gena's *Hanoi*, help him calculate the minimum number of moves needed to restore the tower to its original state.

**Note:** Gena's rods are numbered from  to . All discs are initially located on rod .

**Input Format**

The first line contains a single integer, , denoting the number of discs.  
The second line contains  space-separated integers, where the  integer is the index of the rod where the disk with diameter  is located.

**Constraints**

**Output Format**

Print the minimum number of moves Gena must make to restore the tower to its initial, ordered state on the first rod.

**Sample Input**

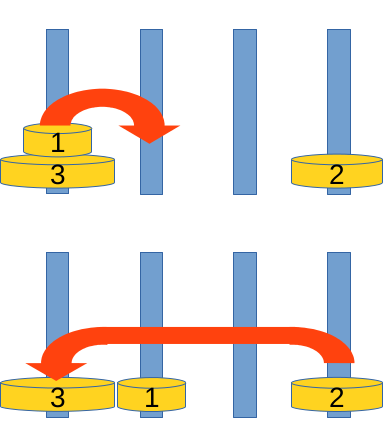
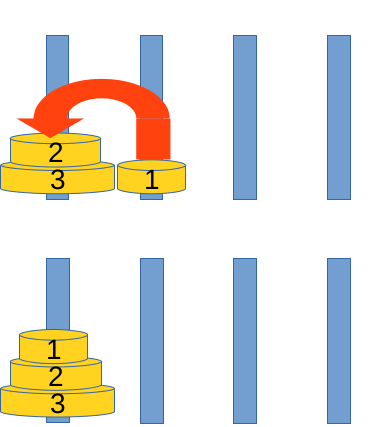
3

1 4 1

**Sample Output**

3

**Explanation**

 moves are enough to build the tower. Here is one possible solution:  
  


**#include<**bits/stdc++.h**>**

**#define** **int** **long** **long** **int**

**#define** pb push\_back

**#define** pp pop\_back

**#define** pii pair<**int**,**int**>

**#define** vec vector<**int**>

**#define** mp make\_pair

**#define** inf 1e10

**using** **namespace** std;

int32\_t main()

{

  ios\_base::sync\_with\_stdio(**false**);

  cin.tie(NULL);

  cout.tie(NULL);

**int** tt=1;

*//cin>>tt;*

**while**(tt--)

  {

**int** n,i;

    cin>>n;

**int** d[(1<<20)];

**int** mask=0;

**for**(i=0;i<n;i++)

    {

**int** k;

      cin>>k;

      mask|=((k-1)<<(2\*i));

    }

**for**(i=0;i<(1<<20);i++)

        d[i]=inf;

    queue<**int**> q;

    q.push(mask);

    d[mask]=0;

**while**(!q.empty())

    {

      mask=q.front();

      q.pop();

**if**(mask==0)

**break**;

      vector<**int**> b[4];

**for**(i=0;i<n;i++)

            b[((mask>>(2\*i))&3)].pb(i);

**for**(i=0;i<4;i++)

        b[i].pb(inf);

**for**(i=0;i<4;i++)

      {

**for**(**int** j=0;j<4;j++)

        {

**if**(b[i].front()>=b[j].front())

**continue**;

**int** id=b[i].front();

**int** to=mask^((i^j)<<(2\*id));

**if**(d[to]<inf)

**continue**;

          d[to]=d[mask]+1;

          q.push(to);

        }

      }

    }

    cout<<d[0];

  }

}