**Next Happy Number :-**

Hard Accuracy: 53.97% Submissions: 22K+ Points: 8

For a given non-negative integer **N**, find the next smallest Happy Number. A number is called **Happy** if it leads to 1 after a sequence of steps. Wherein at each step the number is replaced by the sum of squares of its digits that is, if we start with Happy Number and keep replacing it with sum of squares of its digits, we reach 1 at some point.   
**Example 1:**

**Input:**

N = 8

**Output:**

10

**Explanation:**

Next happy number after 8 is 10 since

1\*1 + 0\*0 = 1

**Example 2:**

**Input:**

N = 10

**Output**

13

**Explanation:**

After 10, 13 is the smallest happy number because

1\*1 + 3\*3 = 10, so we replace 13 by 10 and 1\*1 + 0\*0 = 1.

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function **nextHappy()** which takes an integer **N** as input parameters and returns an integer, next Happy number after N.

**Expected Time Complexity:** O(Nlog10N)  
**Expected Space Complexity:** O(1)  
   
**Constraints:**  
1<=N<=105

**Code :-**

//{ Driver Code Starts

#include<bits/stdc++.h>

using namespace std;

// } Driver Code Ends

unordered\_map<int,int> happy;

class Solution{

public:

bool func(int N){

//N is already happy

if(happy[N]==1){

happy[N] = 1;

return true;

}

//N is already calculated to be repeating & unhappy

if(happy[N]==-1)

return false;

//calculations

int sum=0, curn = N;

while(N){

int rem = N % 10;

sum += (rem \* rem);

N = N / 10;

}

if(sum==1){

happy[curn] = 1;

return true;

}

happy[curn] = -1;

N = sum;

//backtracking + recursion

return (func(N)==true) ? (happy[curn]=1 && true) : false ;

}

int nextHappy(int N){

while(++N){

if(func(N)) return N;

}

}

};

//{ Driver Code Starts.

int main()

{

int t;

cin>>t;

while(t--)

{

int N;

cin>>N;

Solution ob;

cout << ob.nextHappy(N) << endl;

}

return 0;

}

// } Driver Code Ends

**T.C :- O(N\*log10N)**

**S.C :- O(N)**