Coding Competitions Farewell Rounds - Round A

Analysis: Colliding Encoding

Test set 1

For the small test set, we compare each word encoding with the other words encodings. If any of the two encodings are equal then the answer is `YES`. Otherwise, the answer is `NO`. The time complexity for encoding all the words is $O(L)$ where L is the sum of the lengths of the given $\mathbf{N}$ words. Since the length of a word is at most 10, $O(L)$ can be written as $O(\mathbf{N})$. Therefore overall time complexity for this solution is $O(\mathbf{N}^2)$ which is sufficient for the small test set.

Test set 2

The above solution is not suitable for the large test set. Instead of comparing each pair of encoded words, we can calculate encoding for each word and then use hashing or sorting algorithms to check if any two encodings are equal.

Method 1

Hashing: We use a hash table to store the encodings. Before inserting an encoding of a word, we check if it already exists there. If it already exists then the answer is `YES`. If we had to insert all the words encodings then all are unique and the answer is `NO`. The time complexity for encoding all the words is $O(\mathbf{N})$. The time complexity for checking all the words in the hash table is also $O(\mathbf{N})$. The overall time complexity of this solution is $O(\mathbf{N})$ which is sufficient for the large test set.

Sample Code(C++)

```
string anyCollisions(vector<string> words, vector<int> encoding) {
   unordered_set<string> encoded_words;

   for(int i = 0; i < words.size(); i++) {
     string encoded_word;
     // Calculate the encoding for the word and store in encoded_word.
     if (encoded_words.contains(encoded_word)) {
       return "YES";
     }
     encoded_words.insert(encoded_word);
   }

   // If we reach here then there are no collisions.
   return "NO";
}
```

Method 2

Sorting: We store the encoding for each word in an array and sort them. If any two adjacent encodings are equal then the answer is `YES`. Otherwise, the answer is `NO`. The time complexity for encoding all the words is $O(\mathbf{N})$ and for sorting it is $O(\mathbf{N} \log \mathbf{N})$. The overall time complexity of this solution is $O(\mathbf{N} \log \mathbf{N})$ which is sufficient for the large test set.

Sample Code(C++)

```
string anyCollisions(vector<string> words, vector<int> encoding) {
   vector<string> encoded_words;

   for(int i = 0; i < words.size(); i++) {
     string encoded_word;
     // Calculate the encoding for the word and store in encoded_word.
     encoded_words.add(encoded_word);
   }

   encoded_words.sort();
   for(int i = 0; i < encoded_words.size() - 1; i++) {
     if (encoded_words[i] == encoded_words[i+1]) {
       return "YES";
     }
   }

   // If we reach here then there are no collisions.
   return "NO";
}
```