There are many ways to approach this problem. A simple brute force approach will work.

```
int n = sc.nextInt();

// Create a list of candidates from the first string, ordered by size
String first = sc.next();
int k = first.length();
ArrayDeque<String> candidates = new ArrayDeque<String>(k*(k-1)/2);
for( int len=k; len>0; --len ) for( int start=0; start+len<=k; start++ )
{
    candidates.add( first.substring( start, start+len ) );
}

// Eliminate candidates that don't fit subsequent strings
for( int i=1; i<n; i++ )
{
    String permutation = sc.next();
    Iterator<String> it = candidates.iterator();
    while( it.hasNext() )
    {
        String candidate = it.next();
        if( !permutation.contains( candidate ) ) it.remove();
    }
}

// The first candidate in the list has to be the largest common substring,
// since we generated them by size, largest to smallest.
ps.println( candidates.size()==0 ? 0 : candidates.getFirst().length() );
```