LeetCode 49 [Group Anagrams]

Given an array of strings **strs**, group the anagrams together. You can return the answer in any order.

Bat < Tab, Abt, Bta

s: [eat | tea | tan | ate | nat | bat]

Ans: [[eat, tea, ate], [tan, nat], [bat]]

Time: $O(n^3)$

Sorting

Order alphabetically each list item

[aet | aet | ant | aet | ant | abt]

Time: $O(n \log n \times K)$

Now we do this

abc
cba
bac

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

key	List
111 0000...	abc, cba, bac
0101000...	tea, eat
1000101...	tan, nat

Time: $O(n \times K)$

Space: $O(n \times K)$

Java solution

```
public List<List<String>> groupAnagrams(String [] strs){
    if (strs.length ==0) {
        return new ArrayList<>();
    }

    Map<String, List<String>> ansMap = new HashMap<>();

    int[] count = new int[26];

    for ( String s: strs) {
        Arrays.Fill (count, 0);
        for ( char c : s.toCharArray()){
            count [c-'a']++;
        }

        StringBuilder sb = new StringBuilder ();

        for(int i=0; i<26; i++){
            sb.append ("#");
            sb.append (count[i]);
        }

        String key = sb.toString();
        if (!ansMap.containsKey (key)) {
            ansMap.put (key, new ArrayList<>();
        }
        ansMap.get (key).add (s);
    }
    return new ArrayList<> (ansMap.values());
}
```

Kotlin solution

```
Fun groupAnagrams(strs: Array<String>): List<List<String>>{
    if(strs.isEmpty()) return emptyList()

    val ansMap = mutableMapOf<String, MutableList<String>>()
    val count = IntArray(26)

    for(s in strs){
        count.Fill(0)

        for(c in s){
            count[c-'a']++
        }

        val sb = StringBuilder()
        for(i in 0 until 26){
            sb.append('#')
            sb.append(count[i])
        }

        val key = sb.toString()
        ansMap.getOrPut(key){ mutableListOF() }.add(s)
    }

    return ansMap.values.toList()
}
```