Please use this Google doc to code during your interview. To free your hands for coding, we recommend that you use a headset or a phone with speaker option.

Given a string which contains only lowercase letters, remove duplicate letters so that every letter appear once and only once. You must make sure your result is the smallest in **lexicographical** order among all possible results.

Example 1:

Input: "**bc**a**bc**"

Output: "abc"

Example 2:

Input: "**cb**acd**c**b**c**"

Output: "acdb"

abcbc => abc

bacb => bac or **acb** ?

**cbcb**a**c**b**b**cd**a** => abcd

step1

is dup?

a:times

time == 1

step2

former > next && it dup

def is\_dup(s):

   dic\_a = {}

   lst = list(s)

   for i in range(len(lst)):

       if !(lst[i] in dic\_a):

           dic\_a[lst[i]] = 1

       else:

           dic\_a[lst[i]] += 1

   for j in range(len(lst)):

       if dic\_a[lst[j]] > 1:

           if lst[j]

Please use this Google doc to code during your interview. To free your hands for coding, we recommend that you use a headset or a phone with speaker option.

Tech Question:

1 Build a key/value cache which supports expiration times.

e.g.

cache.put(key,value,expiration\_time)

such that the value will be deleted after the expiration\_time

step 1

build k/v st

step 2

{key ,[value, ex time]}

def class cache:

   def \_\_init\_\_(self):

       self.cache = dict()

   def \_\_get\_expirtion\_time\_\_(self, key):

       return self.cache[key][1]

   def \_\_check\_expiration\_time\_\_():

       if (cache.\_\_get\_expirtion\_time\_\_(self, key) - time.now()) <= 0:

           return False

           cache.\_\_clear\_expiration\_time\_\_(key)

   def \_\_clear\_expiration\_time\_\_(key):

       self.cache.remove(key)

   def get\_cache(self,key):

       return self.cache[key]

   def put\_cache(self, key, value, expiration\_time):

       self.cache[key][0] = value