13march2017

March 14, 2017

Rearrange characters in a string such that no two adjacent are same

Idea is to use a loop invariant such that we start with an empty result substring. We increase the size of the substring such that no adjacent characters in this substring are same. We keep on increasing the length of this substring till the substring gets to the length of the original string. If not, then we return that it's impossible.

```
In [38]: def adj_chars_not_same(inpt):
             # convert input string to list, so that we can mutate it.
             st = list(inpt)
             # loop invariant: st[:p] must not have two adjacent characters that are equal
             q = 0
             p = q - 1
             while q < len(st) and p < len(st):
                 p += 1
                 q += 1
                 while q < len(st) and st[p] == st[q]:
                     q += 1
                 # here we satisfy the loop invariant while increasing the size of the substring
                 if q < len(st):
                     p += 1
                     st[p], st[q] = st[q], st[p]
                     q = p + 1
                 else:
                     break
             # termination condition: p has to be one less than q else we know that p is less th
             if p == q - 1:
                 return "".join(st)
             else:
                 return "Not Possible"

In [39]: adj_chars_not_same("aaabc")

Out[39]: 'abaca'

In [36]: adj_chars_not_same("aa")

Out[36]: 'Not Possible'

In [37]: adj_chars_not_same("aaabb")
```

```
Out[37]: 'ababa'

In [33]: adj_chars_not_same("aaaabc")

Out[33]: 'Not Possible'
```