Notebook

August 2, 2024

[ ]:
```python
#### https://github.com/doocs/leetcode/blob/main/solution/0200-0299/0247.
↪Strobogrammatic%20Number%20II/README_EN.md


class Solution:
    def isStrobogrammatic(self, num: str) -> bool:
        d = [0, 1, -1, -1, -1, -1, 9, -1, 8, 6]
        i, j = 0, len(num) - 1
        while i <= j:
            a, b = int(num[i]), int(num[j])
            if d[a] != b:
                return False
            i, j = i + 1, j - 1
        return True


"""
Input: n = 2
Output: ["11","69","88","96"]
Input: n = 1
Output: ["0","1","8"]
"""

class Solution:
    def findStrobogrammatic2(self, m):
        def dfs(n):
            if n == 0:
                return [""]
            if n == 1:
                return ["0", "1", "8"]
            ans = []
            for num in dfs(n-2):
                ans.append("1" + num + "1")
                ans.append("6" + num + "9")
                ans.append("8" + num + "8")
                ans.append("9" + num + "6")
                if n != m:
```

```
                    ans.append("0" + num + "0")
            return ans
        return dfs(m)

s = Solution()
print(s.findStrobogrammatic2(5))


#### find all strobogrammatic numbers between given range

class Solution:
    def strobogrammaticInRange(self, low: str, high: str) -> int:
        def dfs(u):
            if u == 0:
                return ['']
            if u == 1:
                return ['0', '1', '8']
            ans = []
            for v in dfs(u - 2):
                for l, r in ('11', '88', '69', '96'):
                    ans.append(l + v + r)
                if u != n:
                    ans.append('0' + v + '0')
            return ans

        a, b = len(low), len(high)
        low, high = int(low), int(high)
        ans = 0
        for n in range(a, b + 1):
            for s in dfs(n):
                if low <= int(s) <= high:
                    ans += 1
        return ans
```

[3]:
```
##### https://github.com/doocs/leetcode/blob/main/solution/0600-0699/0681.
↪Next%20Closest%20Time/README_EN.md

"""
Input: time = "19:34"
Output: "19:39"
Input: time = "23:59"
Output: "22:22"
"""

class Solution:
    def nextClosestTime(self, s):
        ans, diff = None, float('inf')
```

```
        time = int(s[:2]) * 60 + int(s[3:])
        st = {c for c in s if c != ':'}

        def check(t):
            h, m = int(t[:2]), int(t[2:])
            return 0 <= h < 24 and 0 <= m < 60

        def dfs(curr):
            if (len(curr) == 4):
                if not check(curr):
                    return
                nonlocal ans, diff
                curr_time = int(curr[:2]) * 60 + int(curr[2:])
                if time < curr_time < time + diff:
                    diff = curr_time - time
                    ans = curr[:2] + ':' + curr[2:]
                    return
            for c in st:
                    dfs(curr + c)

        dfs('')
        if ans is None:
            mi = min(int(c) for c in st)
            ans = f'{mi}{mi}:{mi}{mi}'
        return ans

s = Solution()
print(s.nextClosestTime("19:34"))
```

```
---------------------------------------------------------------------------
RecursionError                            Traceback (most recent call last)
Cell In[3], line 40
     37         return ans
     39 s = Solution()
---> 40 print(s.nextClosestTime("19:34"))

Cell In[3], line 33, in Solution.nextClosestTime(self, s)
     30     for c in st:
     31             dfs(curr + c)
---> 33 dfs('')
     34 if ans is None:
     35     mi = min(int(c) for c in st)

Cell In[3], line 31, in Solution.nextClosestTime.<locals>.dfs(curr)
     29         return
     30 for c in st:
---> 31         dfs(curr + c)
```

```
Cell In[3], line 31, in Solution.nextClosestTime.<locals>.dfs(curr)
     29         return
     30 for c in st:
---> 31         dfs(curr + c)

    [… skipping similar frames: Solution.nextClosestTime.<locals>.dfs at line 3
 ↪(2970 times)]

Cell In[3], line 31, in Solution.nextClosestTime.<locals>.dfs(curr)
     29         return
     30 for c in st:
---> 31         dfs(curr + c)

RecursionError: maximum recursion depth exceeded
```

[ ]:
```python
from math import inf


class Solution:
    def nextClosestTime(self, time: str) -> str:
        def check(t):
            h, m = int(t[:2]), int(t[2:])
            return 0 <= h < 24 and 0 <= m < 60

        def dfs(curr):
            if len(curr) == 4:
                if not check(curr):
                    return
                nonlocal ans, d
                p = int(curr[:2]) * 60 + int(curr[2:])
                if t < p < t + d:
                    d = p - t
                    ans = curr[:2] + ':' + curr[2:]
                return
            for c in s:
                dfs(curr + c)

        s = {c for c in time if c != ':'}
        t = int(time[:2]) * 60 + int(time[3:])
        d = inf
        ans = None
        dfs('')
        if ans is None:
            mi = min(int(c) for c in s)
            ans = f'{mi}{mi}:{mi}{mi}'
```

```
        return ans

s = Solution()
print(s.nextClosestTime("19:34"))
```

[ ]: