目录

1 背景目标

手机号共 11 位，第一位是固定的：1；后 8 位是自由组合的数字；前 3 个数字代表了运营商的号段，为了保证正则的严谨和与时俱进，根据数据库中的数据来生成正则是个不错的选择。

当然生成正则的这种场景，比较适合于只能使用正则的情景；在可以不使用正则的情况下，直接截取前 3 位，判断是否在前缀的数组里就可以了。

2 思路

1. 首先我们先从数据库获取到所有前 3 位的号段并去重
2. 然后将前 3 位的数字解析成一棵树
3. 合并相同的第 3 位的区间，比如 (13\d|18\d) 可以写成 1[38]\d
4. 合并连续区间, 比如 15[012356789] 可写成 15[0-35-9]
5. 使用元字符，比如第三步中的 [0-9] 可以写成 \d

3 代码

找个手机归属地数据库：https://github.com/ls0f/phone

首先我们先将 dat 的数据文件导出成 txt 格式

```
def export(self):
    for i in range(0, int(self.phone_record_count)):
        current_offset = int(self.first_phone_record_offset + i *
↪   self.phone_fmt_length)
        buffer = self.buf[current_offset: current_offset + self.phone_fmt_length]
        cur_phone, record_offset, phone_type = struct.unpack(self.phone_fmt, buffer)
        record_content = get_record_content(self.buf, record_offset)
        yield Phone._format_phone_content(cur_phone, record_content, phone_type)
```

根据步骤 1，列出所有的前三位

```
130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 145, 147, 149, 150, 151, 152, 153, 155,
```

根据步骤 2，构建一棵树

```
threePrefix =[130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 145, 147, 149, 150,
↪   151, 152, 153, 155, 156, 157, 158, 159, 170, 171, 172, 173, 175, 176, 177, 178,
↪   180, 181, 182, 183, 184, 185, 186, 187, 188, 189]
tree = {}
for prefix in threePrefix:
    # first = prefix // 100 # always 1
```

```
    second = prefix // 10 % 10
    thirds = tree.setdefault(second, [])
    third = prefix % 10
    if third not in thirds:
        thirds.append(third)
        thirds.sort()
```

```
{3: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
 4: [5, 7, 9],
 5: [0, 1, 2, 3, 5, 6, 7, 8, 9],
 7: [0, 1, 2, 3, 5, 6, 7, 8],
 8: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]}
```

根据步骤 3 合并区间

```
def calcRegexRanges(nums):
    start = nums[0]
    end = nums[0]
    xs = []
    for s in nums:
        if s > end + 1:
            # print(v.get('seconds'), start, end)
            if start == end:
                xs.append(str(start))
            elif end - start == 9:
                xs.append('\\d')
            else:
                xs.append(str(start) + '-' + str(end))
            start = s
            end = start
        else:
            end = s
    # print(v.get('seconds'), start, end)
    if start == end:
        xs.append(str(start))
    elif end - start == 9:
        xs.append('\\d')
    else:
        xs.append(str(start) + '-' + str(end))
    # print('xs: ', xs)
    if len(xs) == 1 and '-' not in xs[0]:
        rs = xs[0]
    else:
        rs = '[' + ''.join(xs) + ']'
    return rs
```

```
groupped = {}
for k,v in sorted(tree.items()):
    i = groupped.setdefault(str(v), {})
    i.setdefault('thirds', v)
    i.setdefault('seconds', []).append(k)

segs = []
for k,v in sorted(groupped.items()):
    ssp = calcRegexRanges(v.get('seconds'))
    tsp = calcRegexRanges(v.get('thirds'))
    segs.append(ssp + tsp)

regex = '1(' + '|'.join(segs) + ')\\d{8}'
print(regex)
```

结果

```
1([38]\d|5[0-35-9]|7[0-35-8]|4[579])\d{8}
```