Code vs Code

我的 Code 实在太长需要裁减，尤其这题尤其实出。

```
import copy
class TopVotedCandidate(object):
    def __init__(self, persons, times):
        """
        :type persons: List[int]
        :type times: List[int]
        """
        self.persons = persons
        self.times = times

        diffs = []
        cur_diff = {}  # person_id => [cnt, last_idx]
        for idx, time in enumerate(self.times):
            if self.persons[idx] in cur_diff:
                cur_diff[self.persons[idx]] = [cur_diff[self.persons[idx]][0] + 1, idx]
            else:
                cur_diff[self.persons[idx]] = [1, idx]
            max_cnt = -1
            max_idx = -1
            res_id = -1
            for person_id, v in cur_diff.items():
                cnt = v[0]
                last_idx = v[1]
                if cnt > max_cnt:
                    res_id = person_id
                    max_cnt = cnt
                    max_idx = last_idx
                elif cnt == max_cnt:
                    if last_idx > max_idx:
                        res_id = person_id
                        max_cnt = cnt
                        max_idx = last_idx
            diffs.append(res_id)
        self.diffs = diffs

    def q(self, t):
        """
        :type t: int
        :rtype: int
        """
        # get the leading vote
        # find the nearest time idx
        idx = self.getNearTimeIdx(0, len(self.times) - 1, t)
        return self.diffs[idx]

    def getNearTimeIdx(self, low, high, time):
        if low >= high:
```

可以使用 Collections.defaultdict(list) 个.

使用 if else 合成一行

合成一行

三个赋值合成一行

合成一行

合成一行

赋值语句合成一行.

cur_diff[person_id] → (num, least index)

多余不需第二层 for 循环，只要通过一个 max Count 即可.

② 忘记使用 cache 优化点.

这两行可

① 两方法可使用 bisect 类不用自己写

② 两方法写得太复杂

```
bisect_right(a, x):
    low, high = 0, len(a)
```

✓

以不要的.

Great ✓

```
while low < high:
    mid = (low + high) // 2
    if x < a[mid]: high = mid
    else: low = mid + 1
return low
```

```
        return low

    if high - low == 1:
        if self.times[high] == time:
            return high
        elif self.times[high] > time:
            return low
        else:
            return high

    mid = int((low + high) / 2)
    if self.times[mid] == time:
        return mid
    elif self.times[mid] > time:
        return self.getNearTimeIdx(low, mid, time)  # maybe include mid
    else:
        return self.getNearTimeIdx(mid, high, time)

if __name__ == "__main__":
    obj = TopVotedCandidate([0, 1, 0, 1, 1], [24, 29, 31, 76, 81])
    for t in [28, 24, 29, 77, 30, 25, 76, 75, 81, 80]:
        print obj.q(t)

class TopVotedCandidate(object):

    def __init__(self, persons, times):
        maxCount = 0
        d = collections.defaultdict(int)
        self.Q = {}
        self.cache = {}
        for i in range(len(persons)):
            d[persons[i]] += 1
            if d[persons[i]] >= maxCount:
                maxCount = d[persons[i]]
                self.Q[times[i]] = persons[i]
        self.res = sorted(self.Q.keys())

    def q(self, t):
        if t not in self.cache:
            index = bisect.bisect_right(self.res, t)
            self.cache[t] = self.Q[self.res[index-1]]
        return self.cache[t]
```

我写3.16行而只用6行即可而且还不需要用到递归

别人代码简直比我好太多.

just sorted by key.