代码补全的测试案例解析及模型对比

- GitHub Copilot
- InCoder
- CodeGen-2B

测试案例：快速排序

输入

```
def quicksort(filename):
    """Fast sorting of lists by value size"""
```

GitHub Copilot

```
def quicksort(filename):
    """Fast sorting of lists by value size"""
    with open(filename, 'r') as f:
        lines = f.readlines()
    lines = [line.strip() for line in lines]
    lines = [line.split(',') for line in lines]
    lines = [[int(x) for x in line] for line in lines]
    lines = np.array(lines)
    np.random.shuffle(lines)
    return lines
```

InCoder

```
def quicksort(filename):
    """Fast sorting of lists by value size"""
    with open(filename) as infile:
        data = infile.readlines()
    data = sorted(data, key=lambda x: (len(x), x))
    left, right = [], []
    pivot = data[0]
    left.append(pivot)
    for entry in data[1:]:
        if len(entry) > len(pivot):
            right.append(entry)
        else:
            left.append(entry)
```

CodeGen-2B

```
def quicksort(filename):  # Fast sorting of lists by value size
    with open(filename, 'r') as f:
        data = f.readlines()
    data = [int(x.strip()) for x in data]
    data.sort()
    return data
```

测试案例1：合并两个有序列表

输入

```
def merge_sorted_arrays():
```

原代码

```
def merge_sorted_arrays(arr1: list[int], arr2: list[int]) -> list[int]:
    """合并两个有序数组

    Parameters
    ----------
    arr1
        有序数组1
    arr2
        有序数组2

    Returns
    -------
        合并后的数组
    """
    i = 0
    j = 0
    res = []
    while i < len(arr1) and j < len(arr2):
        if arr1[i] < arr2[j]:
            res.append(arr1[i])
            i += 1
        else:
            res.append(arr2[j])
            j += 1
    if i < len(arr1):
        res += arr1[i:]
    if j < len(arr2):
        res += arr2[j:]

    return res
```

GitHub Copilot

```
def merge_sorted_arrays():
    """
    Merge two sorted arrays into one sorted array.
    """
    a = [1, 3, 5, 7, 9]
    b = [2, 4, 6, 8, 10]
    c = []
    i = 0
    j = 0
    while i < len(a) and j < len(b):
        if a[i] < b[j]:
            c.append(a[i])
            i += 1
        else:
            c.append(b[j])
            j += 1
    if i < len(a):
        c.extend(a[i:])
    if j < len(b):
        c.extend(b[j:])
    return c
```

CodeGen-2B

```python
def merge_sorted_arrays(arr1: list[int], arr2: list[int]) -> list[int]:
    """
    Merge two sorted arrays into one sorted array.

    Time complexity:
        O(n)
    Space complexity:
        O(1)
    """
    merged_arr = []
    i = j = 0
    while i < len(arr1) and j < len(arr2):
        if arr1[i] < arr2[j]:
            merged_arr.append(arr1[i])
            i += 1
        else:
            merged_arr.append(arr2[j])
            j += 1
    merged_arr += arr1[i:]
    merged_arr += arr2[j:]
    return merged_arr
```

InCoder

```python
def merge_sorted_arrays():
    # Merging two ordered arrays
    arr1 = [1, 5, 6, 8]
    arr2 = [10, 20, 30, 40, 50]

    merged = []
    i = 0
    j = 0

    while i < len(arr1) and j < len(arr2):
        if arr1[i] < arr2[j]:
            merged.append(arr1[i])
            i += 1
        else:
            merged.append(arr2[j])
            j += 1

    while i < len(arr1):
        merged.append(arr1[i])
        i += 1

    while j < len(arr2):
        merged.append(arr2[j])
        j += 1

    return merged
```

测试案例2：计算两个团队的elo评分

输入

```python
def elo_rating(rating1: float, rating2: float, ep: float, rounds: int = 0) -> tuple[float, float]:
```

原代码

```python
def elo_rating(rating1: float, rating2: float, ep: float, rounds: int = 0) -> tuple[float, float]:
    """计算两个队伍的elo评分

    Parameters
    ----------
    rating1
        队伍1的elo评分
    rating2
        队伍2的elo评分
    ep
        真实胜率
    rounds, optional
        同一时间对战轮数, by default 0

    Returns
    -------
        两队的赛后评分
    """
    k = 32
    k_corrs = math.pow(0.9, rounds)

    delta = rating1 - rating2 if ep > 0.5 else rating2 - rating1
    # 分差修正公式
    corr_m = 2.2 / (delta * 0.001 + 2.2)

    prob1 = 1 / (1 + math.pow(10, (rating2 - rating1) / 400))
    prob2 = 1 / (1 + math.pow(10, (rating1 - rating2) / 400))

    rating1 += (ep - prob1) * k * corr_m * k_corrs
    rating2 += (1 - ep - prob2) * k * corr_m * k_corrs

    return rating1, rating2
```

GitHub Copilot

```python
# 输出1
def elo_rating(rating1: float, rating2: float, ep: float, rounds: int = 0) -> tuple[float, float]:
    """
    Calculate the new ratings of two players after a game.
    """
    if rating1 == rating2:
        return rating1, rating2
    elif rating1 > rating2:
        rating1_new = rating1 + ep
        rating2_new = rating2 + ep
    else:
        rating1_new = rating1 + ep
        rating2_new = rating2 + ep
    return rating1_new, rating2_new

# 输出2
def elo_rating(rating1: float, rating2: float, ep: float, rounds: int = 0) -> tuple[float, float]:
    """
    Calculate the new ratings of two teams after a match.
    """
    if rounds == 0:
        rounds = 1
    expected1 = 1 / (1 + math.pow(10, (rating2 - rating1) / 400))
    expected2 = 1 / (1 + math.pow(10, (rating1 - rating2) / 400))
    rating1 = rating1 + ep * (rounds * (1 - expected1))
    rating2 = rating2 + ep * (rounds * (0 - expected2))
    return rating1, rating2
```

CodeGen-2B

```python
def elo_rating(rating1: float, rating2: float, ep: float, rounds: int = 0) -> tuple[float, float]:
    """
    Calculates the elo rating of a player based on the ratings of the two players.
    :param rating1: The rating of the first player.
    :param rating2: The rating of the second player.
    :param ep: The elo rating adjustment factor.
    :param rounds: The number of rounds played.
    :return: A tuple containing the new rating of the first player and the new rating of the second player.
    """
    rating1 = rating1 + ep * (rating2 - rating1)
    rating2 = rating2 + ep * (rating1 - rating2)
    return rating1, rating2
```

InCoder

```python
def elo_rating(rating1: float, rating2: float, ep: float, rounds: int = 0) -> tuple[float, float]:
    """
    Calculates a simple elo rating

    :param rating1: rating 1
    :param rating2: rating 2
    :param ep: elo rating
    :param rounds: number of rounds
    :return: tuple of (rating1, rating2, ep)
    """
    if rounds == 0:
        rounds = 10

    ep = elo_rating_calculator(rating1, rating2, ep, rounds)
    return rating1, rating2, ep


def elo_rating_calculator(rating1: float, rating2: float, ep: float, rounds: int) -> float:
    """
    Calculates the elo rating

    :param rating1: rating 1
    :param rating2: rating 2
    :param
```

测试案例3：游戏阵容生成器

原代码

```python
class ArmyGenerator:
    """阵容生成器"""

    def __init__(self, folder: str) -> None:
        try:
            self._p = Path(folder)
            self._filelst = list(self._p.iterdir())

        except ValueError as e:
            print(f"Error: {e}|无法连接到文件夹{folder}")

    @staticmethod
    def _read_file(filepath: Path) -> dict[str, Any]:
        data = None
        with open(filepath, "rb") as fi:
            data = marshal.load(fi)
        return data

    @staticmethod
    def _extract_army(army: dict[str, Any]) -> tuple[str, list]:
        troop_type = army.get("troop_type", 0)
        if general_data := army.get("general_data", None):
            general_lst = []
            for k in sorted(general_data.keys()):
                general_id = general_data[k]["general_id"]
                if skills := general_data[k].get("skills", None):
                    re_skills = [skill[1] for skill in skills]
                    re_skills[1:] = sorted(re_skills[1:])
                general_lst.append((general_id, re_skills))
        return troop_type, general_lst

    @property
    def filelist(self) -> list[Path]:
        return self._filelst

    def random_army(self) -> tuple[str, list]:
        """随机生成阵容"""
        army = self._read_file(random.choice(self._filelst))
        return self._extract_army(army)

    def pick_army(self, query_txt: str = None) -> tuple[str, list]:
        """根据查询文本选择阵容"""
        if not query_txt:
            return self.random_army()
        name_lst = [f.name for f in self._filelst]
        match_lst = get_close_matches(query_txt, name_lst, cutoff=0.1)
        if len(match_lst) <= 0:
            return self.random_army()
        army = self._read_file(self._p / match_lst[0])
        return self._extract_army(army)
```

输入

```python
class ArmyGenerator:
```

GitHub Copilot

```python
class ArmyGenerator:
    """
    Generate armies for a game.
    """
    def __init__(self, army_size: int = 10, army_count: int = 10, seed: int = 0):
        self.army_size = army_size
        self.army_count = army_count
        self.seed = seed
        self.random = Random(seed)

    def generate(self) -> list:
        """
        Generate armies.
        """
        armies = []
        for i in range(self.army_count):
            armies.append(self.random.sample(range(self.army_size), self.army_size))
        return armies
```

CodeGen-2B

```
class ArmyGenerator:
    def __init__(self, army_size, army_type, army_level, army_exp, army_hp, army_atk, army_def, army_spd, army_res,
                 army_res_atk, army_res_def, army_res_spd, army_res_res, army_res_res_atk, army_res_res_def,
                 army_res_res_res_spd, army_res_res_res_atk, army_res_res_res_def,
army_res_res_res_spd,
                 army_res_res_res_res, army_res_res_res_res_atk, army_res_res_res_res_def, army_res_res_res_res_spd,

                 army_res_res_res_res_res, army_res_res_res_res_res_atk, army_res_res_res_res_res_def,
                 army_res_res_res_
```

InCoder

```
class ArmyGenerator:
    """
    ArmyGenerator generates armies from an initial army configuration
    """
    def __init__(self, army_config: ArmyConfig) -> None:
        self.army = Army()
        self.army.set_configs(army_config)
        self.army.init_army()
        self.army.army_length = self.army.army_config.length
        self.army.max_army_length = self.army.army_config.length
        self.army.army_position = self.army.army_config.position
        self.army.army_config.army_type = self.army.army_config.army_type
        self.army.army_config.army_size = self.army.
```

输入	原代码	GitHub Copilot	InCoder	CodeGen-2B

输入

原代码

GitHub Copilot

InCoder

CodeGen-2B