

# 代码补全的测试结果记录

说明：

- 主要对比三个模型的性能：Github Copilot、InCoder、CodeGen-2B；
- 其中Github Copilot作为baseline；

语言： Python

---

测试案例1： 合并两个有序数组

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
```

输入：

```
def merge_sorted_arrays():
```

**Github Copilot** 输出:

```
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**输出:

```
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**输出:

```
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：计算两个队伍的elo评分

原代码：

```

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_corr = 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

```

输入:

```

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

```

Github Compilot输出:

```

# 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输出:

```
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**输出:

```
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：游戏的阵容生成器

原代码:

```
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1:

```
class ArmyGenerator:
```

Github Copilot输出:

```

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输出:

```

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_spd, army_res_res_res, 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_res_

```

InCoder输出:

```

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.army_config.army_size

```

输入2:

```

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]:

```

Github Copilot输出:

```

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]:
        with open(filepath, "rb") as f:
            return marshal.load(f)

```

CodeGen-2B输出:

```

def _read_file(filepath: Path) -> dict[str, Any]:
    with open(filepath) as f:
        return json.load(f)

```

InCoder输出:

```

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]:
    """读取文件"""
    file = open(filepath, "r", encoding="utf8")
    rawdata = file.read()
    file.close()
    return json.loads(rawdata)

def _generate_item(self, rawdata: dict[str, Any], is_army: bool) -> dict[str, Any]:
    """生成单个装备"""
    if is_army:
        army = self._generate_army(rawdata)
        return army
    else:
        equipment = self._generate_equipment(rawdata)
        return equipment

def _generate_army(self, rawdata: dict[str, Any]) -> Army:
    """生成阵容"""
    army: Army = Army(raw

```

## 测试案例4：神经网络训练设置

输入1:

```
def train(net, train_iter, test_iter, num_epochs, lr):
```

Github Copilot输出:

```

def train(net, train_iter, test_iter, num_epochs, lr):
    print("training...")
    net.train()
    optimizer = torch.optim.Adam(net.parameters(), lr=lr)
    loss_func = nn.CrossEntropyLoss()
    for epoch in range(num_epochs):
        for step, (batch_x, batch_y) in enumerate(train_iter):
            out = net(batch_x)
            loss = loss_func(out, batch_y)
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()
            if step % 100 == 0:
                print(f"epoch {epoch} step {step} loss {loss.item():.4f}")
    print("training finished.")
    return net

```

CodeGen-2B输出:

```
def train(net, train_iter, test_iter, num_epochs, lr):
    loss = gluon.loss.SoftmaxCrossEntropyLoss()
    trainer = gluon.Trainer(net.collect_params(), 'sgd', {'learning_rate': lr})
    for epoch in range(num_epochs):
        train_l_sum, train_acc_sum, n, start = 0.0, 0.0, 0, time.time()
        for X, y in train_iter:
            with autograd.record():
                y_hat = net(X)
                l = loss(y_hat, y).sum()
            l.backward()
            trainer.step(batch_size=1)
            y = y.astype('float32')
            train_l_sum += l.asscalar()
            train_acc_sum += (y_hat.argmax(axis=1) == y).sum().asscalar()
            n += y.size
        test_acc = evaluate_accuracy(test_iter
```

InCoder输出:

```
def train(net, train_iter, test_iter, num_epochs, lr):
    """
    Train a given network on the MNIST dataset.

    Args:
        net: The network to train.
        train_iter: The training data iterator.
        test_iter: The test data iterator.
        num_epochs: The number of epochs to train for.
        lr: The learning rate to train with
    """

    # Make sure we update the parameters of the net
    net.reset_parameters()

    # Put the network in training mode
    net.train()

    # Initialize the optimizer
    optimizer = torch.optim.Adam(net.parameters(), lr=lr)

    # Store the loss and accuracy history
    loss_history = []
    accuracy_history = []

    # Loop over epochs
```

输入2:

```
def train(net, train_iter, test_iter, num_epochs, lr, device):
    def init_weights(m):
        if type(m) == torch.nn.Linear or type(m) == torch.nn.Conv2d:
            torch.nn.init.xavier_uniform_(m.weight)
```

```

net.apply(init_weights)
print('training on', device)
net.to(device)
optimizer = torch.optim.SGD(net.parameters(), lr=lr)
loss = torch.nn.CrossEntropyLoss()

history = {
    'train_l': [],
    'train_acc': [],
    'test_acc': [],
}
fig, ax = plt.subplots(figsize=(5, 5))
for epoch in range(num_epochs):
    metrics = []
    times = []
    times_data = []

    net.train()
    start = time.time()

    for i, (X, y) in enumerate(train_iter):

```

Compilot输出:

```

def train(net, train_iter, test_iter, num_epochs, lr, device):
    def init_weights(m):
        if type(m) == torch.nn.Linear or type(m) == torch.nn.Conv2d:
            torch.nn.init.xavier_uniform_(m.weight)
    net.apply(init_weights)
    print('training on', device)
    net.to(device)
    optimizer = torch.optim.SGD(net.parameters(), lr=lr)
    loss = torch.nn.CrossEntropyLoss()

    history = {
        'train_l': [],
        'train_acc': [],
        'test_acc': [],
    }
    fig, ax = plt.subplots(figsize=(5, 5))
    for epoch in range(num_epochs):
        metrics = []
        times = []
        times_data = []

        net.train()
        start = time.time()

        for i, (X, y) in enumerate(train_iter):
            X, y = X.to(device), y.to(device)
            y_hat = net(X)
            loss_ = loss(y_hat, y)
            optimizer.zero_grad()
            loss_.backward()

```

```

optimizer.step()
if i % 100 == 0:
    print(f"epoch {epoch} step {i} loss {loss_.item():.4f}")
    metrics.append(loss_.item())
    times.append(time.time() - start)
    times_data.append(time.time() - start)

```

CodeGen-2B输出:

```

for i, (X, y) in enumerate(train_iter):
    X = X.to(device)
    y = y.to(device)
    output = model(X)
    loss = criterion(output, y)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    if i % 100 == 0:
        print('epoch: {}, loss: {}'.format(epoch, loss.item()))
torch.save(model.state_dict(), './model.pth')

def test(model, test_iter):
    model.eval()
    correct = 0
    total = 0
    with torch.no_grad():
        for X, y in test_iter:
            X = X.to(device)
            y = y.to(device)
            output = model(X)
            _, predicted = torch.max(output.data, 1)
            total += y.size(0)
            correct += (predicted == y).sum().item()
    print('Accuracy of the model on the 10000

```

InCoder输出:

```

def train(net, train_iter, test_iter, num_epochs, lr, device):
    def init_weights(m):
        if type(m) == torch.nn.Linear or type(m) == torch.nn.Conv2d:
            torch.nn.init.xavier_uniform_(m.weight)
    net.apply(init_weights)
    print('training on', device)
    net.to(device)
    optimizer = torch.optim.SGD(net.parameters(), lr=lr)
    loss = torch.nn.CrossEntropyLoss()

    history = {
        'train_l': [],
        'train_acc': [],
        'test_acc': [],
    }

```

```

fig, ax = plt.subplots(figsize=(5, 5))
for epoch in range(num_epochs):
    metrics = []
    times = []
    times_data = []

    net.train()
    start = time.time()

    for i, (X, y) in enumerate(train_iter):
        X = X.to(device)
        y = y.to(device)

        optimizer.zero_grad()
        preds = net(X)
        loss = loss(preds, y)
        loss.backward()
        optimizer.step()

        metrics.append(accuracy(preds, y))
        times.append(time.time() - start)
        times_data.append(times[-1])

    history['train_l'].append(np.mean(metrics))
    history['train_acc'].append(np.mean(metrics))
    history['test_acc'].append(test_acc(net, test_iter))

    print('epoch', epoch, 'train_l:', history['train_l'][epoch], 'train_acc:',
          history['train_acc'][epoch],
          'test_acc:', history['test_acc

```

## 测试案例5：画图

源代码：

```

def linepoints(x1, y1, x2, y2):
    # generate points with two float points
    points = []

    dx = x2 - x1
    dy = y2 - y1
    steep = abs(dx) < abs(dy)
    if steep:
        x1, y1 = y1, x1
        x2, y2 = y2, x2
        dx, dy = dy, dx
    sign = 1 if x2 > x1 else -1

    sx = int(x1 + 0.5 * sign)
    ex = int(x2 - 0.5 * sign)

```

```

de = dy / dx
y = y1 + de * (sx + 0.5 - x1)

points.append((int(y1), int(x1)) if steep else (int(x1), int(y1)))
for x in range(sx, ex + sign, sign):
    points.append((int(y), x) if steep else (x, int(y)))
    y += de * sign
points.append((int(y2), int(x2)) if steep else (int(x2), int(y2)))

return points

```

输入1:

```

def linepoints(x1, y1, x2, y2):
    # generate points with two float points
    points = []
    dx = x2 - x1
    dy = y2 - y1

```

Copilot输出:

```

def linepoints(x1, y1, x2, y2):
    # generate points with two float points
    points = []

    dx = x2 - x1
    dy = y2 - y1
    steep = abs(dx) < abs(dy)

```

CodeGen-2B输出:

```

dx = x2 - x1
dy = y2 - y1

if dx > dy:
    for i in range(x1, x2):
        print(i, end=' ')
else:
    for i in range(y1, y2):
        print(i, end=' ')
<|endoftext|>#

```

InCoder输出:

```

def linepoints(x1, y1, x2, y2):
    # generate points with two float points
    points = []
    dx = x2 - x1
    dy = y2 - y1
    x, y = x1, y1
    while x <= x2:

```

```

        points.append((x, y))
        x += dx
        y += dy
    return points

def calcLinePtsInd(x, y, pts):
    # calculate points in the line defined by two points
    inds = []
    for i in range(len(pts) - 1):
        x1 = pts[i][0]
        y1 = pts[i][1]
        x2 = pts[i + 1][0]
        y2 = pts[i + 1][1]
        if (x2 - x1) != 0:
            inds.append(math.floor((x - x1) / (x2 - x1)))
        else:
            inds.append(0)
    return inds

def calcLinePtsIndR(x, y, pts):

```

## 测试案例6：画图

源代码：

```

def VoronoiDiagram():
    width, height = 512, 512

    img = Image.new("RGB", (width, height), (50, 50, 50))
    draw = ImageDraw.Draw(img)

    points = []
    for i in range(100):
        x = random.uniform(0, width)
        y = random.uniform(0, height)
        points.append((x, y))

    for (x, y) in points:
        draw.point((x, y), fill='red')

    vp = voronoi.Voronoi(points)
    vp.process()
    print(vp.event)

    lines = vp.get_output()
    print(lines)

    for line in lines:
        draw.line(line, fill='yellow', width=2)

```

输入1:

```
def VoronoiDiagram():
    width, height = 512, 512

    img = Image.new("RGB", (width, height), (50, 50, 50))
    draw = ImageDraw.Draw(img)

    points = []
    for i in range(100):
```

copilot输出:

```
def VoronoiDiagram():
    width, height = 512, 512

    img = Image.new("RGB", (width, height), (50, 50, 50))
    draw = ImageDraw.Draw(img)

    points = []
    for i in range(100):
        x = random.randint(0, width)
        y = random.randint(0, height)
        points.append((x, y))
        draw.ellipse((x - 2, y - 2, x + 2, y + 2), fill=(255, 255, 255))
```

CodeGen-2B输出:

```
def VoronoiDiagram():
    width, height = 512, 512
    img = Image.new(, (width, height), (50, 50, 50))
    draw = ImageDraw.Draw(img)
    draw.rectangle(((0, 0), (width, height)), fill=(255, 255, 255))
    draw.rectangle(((0, 0), (width, height)), fill=(0, 0, 0))
    draw.rectangle(((0, 0), (width, height)), fill=(255, 255, 255))
    draw.rectangle(((0, 0), (width, height)), fill=(0, 0, 0))
    draw.rectangle(((0, 0), (width, height)), fill=(255, 255, 255))
    draw.rectangle(((0, 0), (width, height)), fill=(0, 0, 0))
    draw.rectangle(((0, 0), (width, height)), fill=(255, 255, 255))
    draw.rectangle(((0, 0), (width, height)), fill=(0, 0, 0))
    draw.rectangle(((
```

InCoder输出:

```
def VoronoiDiagram():
    width, height = 512, 512

    img = Image.new("RGB", (width, height), (50, 50, 50))
    draw = ImageDraw.Draw(img)

    points = []
    for i in range(100):
        points.append(Point((random.random() * width, random.random() * height)))
```



```

points.append(Point((random.random() * width, random.random() * height)))

vor = Voronoi(points)

for vertex in vor.vertices:
    draw.line(vertex, fill="black", width=3)

img.show()

def VoronoiDiagram2():
    width, height = 512, 512

    img = Image.new("RGB", (width, height), (50, 50, 50))
    draw = ImageDraw.Draw(img)

    points = []
    for i in range(100):
        points.append(Point((random.random() * width, random.random() * height)))
    points.append(Point((random.random() * width, random.random() *

```

## 测试案例7：划线

源代码：

```

def draw_line():
    x = np.linspace(0, 2, 100)

    fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
    ax.plot(x, x, label='linear')
    ax.plot(x, x**2, label='quadratic')
    ax.plot(x, x**3, label='cubic')
    ax.set_xlabel('x label')
    ax.set_ylabel('y label')
    ax.set_title('Simple Plot')
    ax.legend()

```

输入1:

```

def draw_line():
    x = np.linspace(0, 2, 100)
    fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')

```

copilot输出:

```

def draw_line():
    x = np.linspace(0, 2, 100)
    fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
    ax.plot(x, np.sin(x), 'b-')

```

CodeGen-2B输出:

```
def draw_line():
    x=np.linspace(0,2,100)
    fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
    ax.plot(x, x, 'r', linewidth=2)
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_title('Line')
    plt.show()
```

InCoder输出:

```
def draw_line():
    x = np.linspace(0, 2, 100)
    fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
    ax.plot(x, x, label='$x$')
    ax.legend(loc='upper left', bbox_to_anchor=(1, 1))

def draw_triangle():
    x = np.linspace(0, 1, 100)
    y = np.sin(x)
    fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
    ax.plot(x, y, 'o', label='$sin(x)$')
    ax.legend(loc='upper left', bbox_to_anchor=(1, 1))

def draw_circle():
    x = np.linspace(0, np.pi, 100)
    y = np.sin(x)
    fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
    ax.plot(x, y, 'o', label='$sin(x)$')
    ax.legend(loc='upper left', bbox_to_anchor=(1,
```

输入2:

```
def draw_line():
    x = np.linspace(0, 2, 100)
    fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
    ax.plot(x, x, label='linear')
    ax.plot(x, x**2, label='quadratic')
```

copilot输出:

```
def draw_line():
    x = np.linspace(0, 2, 100)
    fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
    ax.plot(x, x, label='linear')
    ax.plot(x, x**2, label='quadratic')
    ax.plot(x, x**3, label='cubic')
```

CodeGen-2B输出:

```
def draw_line():
    x = np.linspace(0, 2, 100)
    fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
    ax.plot(x, x, label='linear')
    ax.plot(x, x**2, label='quadratic')
    ax.plot(x, x**3, label='cubic')
    ax.plot(x, x**4, label='quartic')
    ax.plot(x, x**5, label='quintic')
    ax.set_xlabel('x label')
    ax.set_ylabel('y label')
    ax.set_title('Simple Plot')
    ax.legend()
    plt.show()
```

InCoder输出:

```
def draw_line():
    x = np.linspace(0, 2, 100)
    fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
    ax.plot(x, x, label='linear')
    ax.plot(x, x**2, label='quadratic')
    ax.plot(x, x**3, label='cubic')
    ax.legend(loc='upper left')
    ax.set_title('Linear vs. quadratic vs. cubic')
    ax.set_xlim((0, 4))
    ax.set_ylim((0, 4))
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.grid(True)
    plt.show()
    fig.savefig('images/linear_quadratic_cubic.png', dpi=200)

def draw_image():
    np.random.seed(0)
    fig = plt.figure(figsize=(5.5, 2.5))
    fig.subplots_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05)
    for i in range(1, 5):
        ax = fig.add_subplot(1, 5, i)
        ax.axis('off')
        ax.set_xticklabels([])
        ax.set_yticklabels([])
```

## 测试案例8:

源代码:

```
def CreateHexFogSprite(self, rc, isInSideEdge):
    import commonres
    rootNode = self.m_FogPipLine.GetScene()

    if isInSideEdge:
```

```

        idx = misc.RandInt(1, 2)
    else:
        idx = misc.RandInt(1, 5)

    if isInSideEdge:
        randScale = (misc.RandInt(1, 10) - 3) / 100.0
    else:
        randScale = (misc.RandInt(1, 30) - 10) / 100.0
    fScale = 0.5 + randScale

    oSprite = commonres.MazeFogId2Sprite(idx)
    oSprite = SetAnchorPoint(0.5, 0.5)
    oSprite.SetScale(fScale)
    oSprite.SetPos(*HEX_SHAPE.HexCenterRC2XY(*rc))
    oSprite.SetParent(rootNode)
    oSprite.IsInSideEdge = isInSideEdge
    self.m_HexFogGrids[rc] = oSprite

    disx = misc.RnadInt(1, 30) - 15
    disy = misc.RandInt(1, 30) - 15
    action = C_object.CMoveBy(6.0, disx, disy)
    action2 = C_object.CMoveBy(6.0 -disx, -disy)
    oSprite.RunAction(C_object.CRepeatForever(C_object.CSequence([action, action2])))

```

输入1:

```

def CreateHexFogSprite(self, rc, isInSideEdge):
    import commonres
    rootNode = self.m_FogPipLine.GetScene()

    if isInSideEdge:
        idx = misc.RandInt(1, 2)
    else:
        idx = misc.RandInt(1, 5)

    if isInSideEdge:
        randScale = (misc.RandInt(1, 10) - 3) / 100.0
    else:

```

copilot输出:

```

def CreateHexFogSprite(self, rc, isInSideEdge):
    import commonres
    rootNode = self.m_FogPipLine.GetScene()

    if isInSideEdge:
        idx = misc.RandInt(1, 2)
    else:
        idx = misc.RandInt(1, 5)

    if isInSideEdge:
        randScale = (misc.RandInt(1, 10) - 3) / 100.0
    else:
        randScale = (misc.RandInt(1, 10) - 5) / 100.0

```

CodeGen-2B输出:

```

def CreateHexFogSprite(self, rc, isInSideEdge):
    import commonres
    rootNode = self.m_FogPipLine.GetScene()
    if isInSideEdge:
        idx = misc.RandInt(1, 2)
    else:
        idx = misc.RandInt(1, 5)
    if isInSideEdge:
        randScale = (misc.RandInt(1, 10) - 3) / 100.0
    else:
        randScale = (misc.RandInt(1, 10) - 5) / 100.0
    randScale = max(0.1, randScale)
    randScale = min(1.0, randScale)
    randScale = randScale * 0.5
    randScale = randScale * 0.5
    randScale = randScale * 0.5
    randScale = randScale * 0.5
    randScale = randScale * 0.5
    randScale = randScale * 0.5
    randScale = randScale * 0.5
    randScale = randScale * 0.5
    rand

```

InCoder输出:

```

def CreateHexFogSprite(self, rc, isInSideEdge):
    import commonres
    rootNode = self.m_FogPipLine.GetScene()

    if isInSideEdge:
        idx = misc.RandInt(1, 2)
    else:
        idx = misc.RandInt(1, 5)

```

```

if isInSideEdge:
    randScale = (misc.RandInt(1, 10) - 3) / 100.0
else:
    randScale = (misc.RandInt(1, 10) - 3) / 100.0

if idx <= 1:
    fogNode = rootNode.CreateChild("FogFogNode")
    fogNode.SetPosition([0, 0, 0])
    fogNode.SetScale([randScale, randScale, randScale])
    fogNode.SetFocalPoint([0.0, 0.0, 0.0])
    fogNode.SetColorRGB(commonres.kFogColorRed, commonres.kFogColorGreen,
commonres.kFogColorBlue)
    fogNode.SetColorAlpha(commonres.kFogColorAlpha)
    fo

```

## 测试案例9：最短路径

源代码：

```

def minDistance(word1, word2):
    m, n = len(word1), len(word2)
    dp = [[-1] * (n + 1) for _ in range(m + 1)]
    dp[0][0] = 0
    for i in range(m + 1):
        dp[i][0] = i
    for j in range(n + 1):
        dp[0][j] = j
    for i in range(1, m + 1):
        for j in range(1, n + 1):
            if word1[i - 1] == word2[j - 1]:
                dp[i][j] = dp[i - 1][j - 1]
            else:
                dp[i][j] = min(dp[i - 1][j - 1], dp[i - 1][j], dp[i][j - 1]) + 1
    return dp[m][n]

```

输入1:

```

def minDistance(word1, word2):
    m, n = len(word1), len(word2)
    dp = [[-1] * (n + 1) for _ in range(m + 1)]
    dp[0][0] = 0
    for i in range(m + 1):
        dp[i][0] = i
    for j in range(n + 1):
        dp[0][j] = j
    for i in range(1, m + 1):

```

copilot输出:

```

def minDistance(word1, word2):

```

```

m, n = len(word1), len(word2)
dp = [[-1] * (n + 1) for _ in range(m + 1)]
dp[0][0] = 0
for i in range(m + 1):
    dp[i][0] = i
for j in range(n + 1):
    dp[0][j] = j
for i in range(1, m + 1):
    for j in range(1, n + 1):
        if word1[i - 1] == word2[j - 1]:
            dp[i][j] = dp[i - 1][j - 1]
        else:
            dp[i][j] = min(dp[i - 1][j - 1], dp[i - 1][j], dp[i][j - 1]) + 1

```

CodeGen-2B输出:

```

def minDistance(word1, word2):
    m, n = len(word1), len(word2)
    dp = [[-1] * (n + 1) for _ in range(m + 1)]
    dp[0][0] = 0
    for i in range(m + 1):
        dp[i][0] = i
    for j in range(n + 1):
        dp[0][j] = j
    for i in range(1, m + 1):
        for j in range(1, n + 1):
            if word1[i - 1] == word2[j - 1]:
                dp[i][j] = dp[i - 1][j - 1]
            else:
                dp[i][j] = min(dp[i - 1][j], dp[i][j - 1], dp[i - 1][j - 1]) + 1
    return dp[m][n]

```

InCoder输出:

```

def minDistance(word1, word2):
    m, n = len(word1), len(word2)
    dp = [[-1] * (n + 1) for _ in range(m + 1)]
    dp[0][0] = 0
    for i in range(m + 1):
        dp[i][0] = i
    for j in range(n + 1):
        dp[0][j] = j
    for i in range(1, m + 1):
        for j in range(1, n + 1):
            if word1[i - 1] == word2[j - 1]:
                dp[i][j] = dp[i - 1][j - 1]
            else:
                dp[i][j] = 1 + min(dp[i - 1][j], dp[i][j - 1], dp[i - 1][j - 1])
    return dp[m][n]

```

```

word1 = "computer"
word2 = "computers"
print(minDistance(word1, word2))
word1 = "computer"
word2 = "computer"
print(minDistance(word1, word2))
word1 = "computer"
word2 = "computers"
print(minDistance(word1, word2))
word1 = "computer"
word2 = "computer"

```

## 测试案例10：递增路径

源代码：

```

def IncreasingPath(matrix):
    m, n = len(matrix), len(matrix[0])
    flag = [[-1] * n for _ in range(m)] # 存储从(i, j)出发的最长递归路径

    # 寻找节点的最长路径
    def dfs(i, j):
        d = 1
        # 最长路径已知（记忆化搜索，避免重复的计算）
        if flag[i][j] != -1:
            d = flag[i][j]
        else:
            for (x, y) in [[-1, 0], [1, 0], [0, 1], [0, -1]]:
                x, y = i + x, j + y
                if 0 <= x < m and 0 <= y < n and matrix[x][y] > matrix[i][j]:
                    d = max(d, dfs(x, y) + 1) # 取四个邻接点的最长
            flag[i][j] = d # 在该节点上记录其所需最大路径长度
        return d

    res = 0
    for i in range(m): # 遍历矩阵计算最长路径
        for j in range(n):
            if flag[i][j] == -1:
                res = max(res, dfs(i, j))
    return res

```

输入1：

```

def IncreasingPath(matrix):
    m, n = len(matrix), len(matrix[0])

```



copilot输出:

```
def IncreasingPath(matrix):
    m, n = len(matrix), len(matrix[0])
    dp = [[0] * n for _ in range(m)]
    for i in range(m):
        for j in range(n):
            if i == 0 and j == 0:
                dp[i][j] = matrix[i][j]
            elif i == 0:
                dp[i][j] = dp[i][j - 1] + matrix[i][j]
            elif j == 0:
                dp[i][j] = dp[i - 1][j] + matrix[i][j]
            else:
                dp[i][j] = min(dp[i - 1][j], dp[i][j - 1]) + matrix[i][j]
    return dp[m - 1][n - 1]
```

CodeGen-2B输出:

```
def IncreasingPath(matrix):
    m, n = len(matrix), len(matrix[0])
    res = []
    for i in range(m):
        for j in range(n):
            if i == 0 or j == 0:
                if matrix[i][j] > matrix[i][j-1] and matrix[i][j] > matrix[i-1][j]:
                    res.append([i, j])
            elif i == 0 or j == n-1:
                if matrix[i][j] > matrix[i][j-1] and matrix[i][j] > matrix[i-1][j]:
                    res.append([i, j])
            elif matrix[i][j] > matrix[i-1][j] and matrix[i][j] > matrix[i][j-1]:
                res.append([i, j])
    return res
```

InCoder输出:

```
def IncreasingPath(matrix):
    m, n = len(matrix), len(matrix[0])
    if m == 0 or n == 0:
        return matrix

    x, y = 0, 1
    matrix[0][0] = 1

    for _ in range(m):
        for i in range(n):
            if matrix[i][y] > matrix[i][x]:
                matrix[i][y] = matrix[i][x]
                matrix[i][x] = 1
```

```

        else:
            matrix[i][x] = matrix[i][y]
            matrix[i][y] = 1
            x, y = y, x+1

    return matrix

def IncreasingPath2(matrix):
    m, n = len(matrix), len(matrix[0])
    if m == 0 or n == 0:
        return matrix

    x, y = 0, 1
    matrix[0][0] = 1

    for

```

## 测试案例11：宽度为K的窗口

源代码：

```

def window(List, K):
    window = []
    for i in range(len(List) - K + 1):
        res = []
        for j in range(K):
            res.append(List[i + j])
        window.append(res)
    return window

```

输入1:

```

def window(List, K):
    window = []
    for i in range(len(List) - K + 1):

```

copilot输出:

```

def window(List, K):
    window = []
    for i in range(len(List) - K + 1):
        window.append(List[i:i + K])

```

CodeGen-2B输出:

```
def window(List, K):
    window = []
    for i in range(len(List) - K + 1):
        window.append(List[i:i + K])
    return window
```

InCoder输出:

```
def window(List, K):
    window = []
    for i in range(len(List) - K + 1):
        window.append(List[i:i + K])
    return window

def mergeSortRecursive(List, K):
    if len(List) <= 1:
        return List
    else:
        mid = len(List) // 2
        left = mergeSortRecursive(List[:mid], K)
        right = mergeSortRecursive(List[mid:], K)
        return merge(left, right, K)

def mergeSort(List, K):
    return mergeSortRecursive(List, K)

def merge(left, right, K):
    List1 = []
    List2 = []
    List3 = []
    if len(left) >= K:
        for i in range(len(left) - K):
            List1.append(left[i])
        if len(left) -
```

## 测试案例12：最大值

源代码:

```
def max_num(List):
    MAX = []
    if len(List) == 0:
        return print("The List is empty!")
    else:
        for i in range(len(List)):
            max_Num = List[i][0]
            for j in range(len(List[i])):
                if max_Num < List[i][j]:
                    max_Num = List[i][j]
            MAX.append(max_Num)
    return MAX
```

输入1:

```
def max_num(List):  
    MAX = []
```

copilot输出:

```
def max_num(List):  
    MAX = []  
    for i in range(len(List)):  
        MAX.append(List[i])  
        for j in range(i + 1, len(List)):  
            if List[i] < List[j]:  
                MAX.append(List[j])  
            else:  
                break
```

CodeGen-2B输出:

```
def max_num(List):  
    MAX = []  
    for i in range(len(List)):  
        MAX.append(max(List[i]))  
    return MAX
```

InCoder输出:

```
def max_num(List):  
    MAX = []  
    for x in List:  
        if x > MAX:  
            MAX = x  
    return MAX  
def min_num(List):  
    MIN = []  
    for x in List:  
        if x < MIN:  
            MIN = x  
    return MIN  
def mean_num(List):  
    Sum = sum(List)  
    Mean = Sum / len(List)  
    return Mean  
def median_num(List):  
    Median = []  
    List.sort()  
    for x in List:  
        if x % 2 == 0:  
            Median.append(x)  
        elif x % 2 == 1:  
            Median.append(x + 1)
```

```
if len(Median) > 0:
    Median = Median[len(Median) // 2]
```

### 测试案例13：求最大面积

源代码：

```
def maximalRectangle(rows, cols, List):
    res = [0]
    for i in range(rows):
        for j in range(cols):
            H = List[i][j]
            D = 1
            L = j - 1
            R = j + 1
            while (L >= 0) and (H <= List[i][L]):
                L -= 1
                D += 1
            while (R < cols) and (H <= List[i][R]):
                R += 1
                D += 1
            res.append(D * H)
    return max(res)
```

输入：

```
def maximalRectangle(rows, cols, List):
    res = [0]
    for i in range(rows):
```

copilot输出：

```
def maximalRectangle(rows, cols, List):
    res = [0]
    for i in range(rows):
        for j in range(cols):
            if List[i][j] == 0:
                res.append(0)
            else:
                res.append(res[-1] + 1)
    return res
```

CodeGen-2B输出：

```
def maximalRectangle(rows, cols, List):
    res = [0]
    for i in range(rows):
        for j in range(cols):
            if List[i][j] == '1':
                res.append(max(res[-1], dp(i, j, List)))
    return max(res)
```

InCoder输出:

```
def maximalRectangle(rows, cols, List):
    res = [0]
    for i in range(rows):
        res.append(0)
        for j in range(cols):
            res[i] = max(res[i], List[j+(i*cols)])
    return res

def triangleArea(rows, cols, List):
    res = [0]
    for i in range(rows):
        res.append(0)
        for j in range(cols):
            res[i] = max(res[i], List[j+(i*cols)])
    return res

def checkTriangle(rows, cols, List):
    res = [0]
    for i in range(rows):
        res.append(0)
        for j in range(cols):
            res[i] = max(res[i], List[j+(i*cols)])
    return res

def checkRectangle(rows, cols, List):
    res = [0]
    for i in range(rows):
        res.append(0)
        for j in range(col
```

## 测试案例14：求矩形面积

源代码:

```
def ectangleArea(matrix):
    res = []
```

```

if matrix == []:
    rows = 0
    cols = 0
else:
    rows = len(matrix)
    cols = len(matrix[0])
    for i in range(rows):
        height = [0] * cols
        stop = [False] * cols
        I = i + 1
        while I:
            for j in range(cols):
                if matrix[I - 1][j] == '1' and (not stop[j]):
                    height[j] += 1
                else:
                    stop[j] = True
            I -= 1
        res.append(height)
    return rows, cols, res

```

输入1:

```

def ectangleArea(matrix):
    res = []
    if matrix == []:
        rows = 0
        cols = 0
    else:

```

copilot输出:

```

def ectangleArea(matrix):
    res = []
    if matrix == []:
        rows = 0
        cols = 0
    else:
        rows = len(matrix)
        cols = len(matrix[0])
        for i in range(rows):
            for j in range(cols):
                if matrix[i][j] == 0:
                    res.append(0)
                else:
                    res.append(res[-1] + 1)

```

CodeGen-2B输出:

```
def ectangleArea(matrix):
    res = []
    if matrix == []:
        rows = 0
        cols = 0
    else:
        rows = len(matrix)
        cols = len(matrix[0])
    for i in range(rows):
        for j in range(cols):
            if matrix[i][j] == 1:
                res.append(getArea(i, j, rows, cols))
    return res
```

InCoder输出:

```
def ectangleArea(matrix):
    res = []
    if matrix == []:
        rows = 0
        cols = 0
    else:
        rows = len(matrix)
        cols = len(matrix[0])
    for row in range(rows):
        res.append([])
        for col in range(cols):
            res[row].append(matrix[row][col])
    sum = 0
    for row in res:
        sum += row[0]*row[-2]*row[-1]
    return sum/3

def ectanglePerimeter(matrix):
    res = []
    if matrix == []:
        rows = 0
        cols = 0
    else:
        rows = len(matrix)
        cols = len(matrix[0])
    for row in range(rows):
        res.append([])
        for col in range(cols):
            res[row].append(matrix[row][col])
    sum = 0
    for row in res:
```

**测试案例15：**拆分棋子。返回棋子编号及索引

源代码:



```
def splitting(str, num):
    B, R = [], []
    str = list(str)
    for i in range(len(str)):
        if str[i] == 'B':
            B.append([num[i], i])
        else:
            R.append([num[i], i])
    return B, R
```

输入1:

```
def split(str, num):
    B, R = [], []
    str = list(str)
```

copilot输出:

```
def split(str, num):
    B, R = [], []
    str = list(str)
    for i in range(len(str)):
        if i % num == 0:
            B.append(str[i])
        else:
            R.append(str[i])
```

CodeGen-2B输出:

```
def split(str, num):
    B, R = [], []
    str = list(str)
    for i in range(len(str)):
        if i % num == 0:
            R.append(str[i])
        else:
            B.append(str[i])
    return ''.join(B), ''.join(R)
```

InCoder输出:

```
def split(str, num):
    B, R = [], []
    str = list(str)
    while num:
        B = str[: num // 2]
        R = str[num // 2 :]
        num, B = num, num // 2
        R += B
    return R

def split_into_half(sentence):
```

```

half_word_count = round(len(sentence)/2)
return split(sentence, half_word_count)

def split_into_pairs(sentence):
    pair_count = round(len(sentence)/2)
    return split(sentence, pair_count)

def split_into_three(sentence):
    three_count = round(len(sentence)/3)
    return split(sentence, three_count)

def split_into_four(sentence):

```

## 测试案例16：元祖的异或值

源代码：

```

def XORed(tuple):
    x0 = tuple[0]
    for i in range(len(tuple)):
        if i >= 1:
            x0 ^= tuple[i]
    return x0

```

输入1:

```

def XORed(tuple):
    x0 = tuple[0]
    for

```

copilot输出:

```

def XORed(tuple):
    x0 = tuple[0]
    for i in range(1, len(tuple)):
        x0 ^= tuple[i]

```

CodeGen-2B输出:

```
def XORed(tuple):
    x0 = tuple[0]
    for i in range(1, len(tuple)):
        x0 = XOR(x0, tuple[i])
    return x0
```

InCoder输出:

```
def XORed(tuple):
    x0 = tuple[0]
    for i in range(1, len(tuple)):
        x1 = tuple[i]
        temp = x0
        x0 = x1
        x1 = temp
    return x0

def XOR(tuple):
    x0 = tuple[0]
    for i in range(1, len(tuple)):
        x1 = tuple[i]
        x0 = x0 ^ x1
    return x0

def ANDed(tuple):
    x0 = tuple[0]
    for i in range(1, len(tuple)):
        x1 = tuple[i]
        temp = x0
        x0 = x1
        x1 = temp
    return x0

def AND(tuple):
    x0 = tuple[0]
    for i in range(1, len(tuple)):
        x1 = tuple[i]
```

## 测试案例17： 判断一个数是否为质数

源代码:

```
def prime_number(value):
    for i in range(2, value):
        if value % i == 0:
            return False
    return True
```

输入:

```
def prime_number(value):
```

copilot输出:

```
def prime_number(value):
    if value == 1:
        return False
    for i in range(2, value):
        if value % i == 0:
            return False
    return True
```

CodeGen-2B输出:

```
def prime_number(value):
    if value < 2:
        return False
    for i in range(2, value):
        if value % i == 0:
            return False
    return True
```

InCoder输出:

```
def prime_number(value):
    count = 0
    for i in range(2, int(math.sqrt(value)) + 1):
        if value % i == 0:
            count += 1
            if count == 2:
                return i
    return value

def prime_number_list(lst):
    prime_list = []
    for number in lst:
        prime_list.append(prime_number(number))
    return prime_list

def prime_number_list_in_order(lst):
    prime_list = []
    prime_number_list = prime_number_list(lst)
```

```

for i in range(0, len(prime_number_list)):
    for j in range(0, i):
        if prime_number_list[i] > prime_number_list[j]:
            temp = prime_number_list[i]
            prime_number_list[i] = prime_number_

```

## 测试案例18：快速读取

原代码：

```

class FastIO(IOBase):
    # 快读函数
    newlines = 0

    def __init__(self, file):
        self._fd = file.fileno()
        self.buffer = BytesIO()
        self.writable = "x" in file.mode or "r" not in file.mode
        self.write = self.buffer.write if self.writable else None

    def read(self):
        while True:
            b = os.read(self._fd, max(os.fstat(self._fd).st_size, BUFSIZE))
            if not b:
                break
            ptr = self.buffer.tell()
            self.buffer.seek(0, 2), self.buffer.write(b), self.buffer.seek(ptr)
            self.newlines = 0
            return self.buffer.read()

    def readline(self):
        while self.newlines == 0:
            b = os.read(self._fd, max(os.fstat(self._fd).st_size, BUFSIZE))
            self.newlines = b.count(b"\n") + (not b)
            ptr = self.buffer.tell()
            self.buffer.seek(0, 2), self.buffer.write(b), self.buffer.seek(ptr)
            self.newlines -= 1
            return self.buffer.readline()

    def flush(self):
        if self.writable:
            os.write(self._fd, self.buffer.getvalue())
            self.buffer.truncate(0), self.buffer.seek(0)

```

输入1:

```

class FastIO(IOBase):
    newlines = 0
    def __init__(self, file):

```

copilot输出:

```

class FastIO(IOBase):
    newlines = 0
    def __init__(self, file):
        self._fd = file.fileno()
        self.buffer = BytesIO()
        self.writable = "x" in file.mode or "r" not in file.mode
        self.readable = "r" in file.mode
        self.reset_newlines()

```

CodeGen-2B输出:

```

class FastIO(IOBase):
    newlines = 0
    def __init__(self, file):
        self._fd = file.fileno()
        self.buffer = BytesIO()
        self.writable = "x" in file.mode or "r" not in file.mode
        self.write = self.buffer.write if self.writable else None
    def read(self):
        while True:
            b = os.read(self._fd, max(os.fstat(self._fd).st_size, BUFSIZE))
            if not b:
                break
            ptr = self.buffer.tell()
            self.buffer.seek(0, 2), self.buffer.write(b), self.buffer.seek(ptr)
            self.newlines = 0
            return self.buffer.read()
    def readline(self):
        while self.newlines == 0:
            b = os.read(self._fd, max(os.fstat(self._fd).st_size, BUFSIZE))
            self.newlines = b.count(b"\n")

```

InCoder输出:

```

class FastIO(IOBase):
    newlines = 0
    def __init__(self, file):
        self._fd = file.fileno()
        self.buffer = BytesIO()
        self.writable = "x" in file.mode or "r" not in file.mode
        self.write = self.buffer.write if self.writable else None

    def readable(self):
        return True

    def writable(self):
        return self.writable

    def seekable(self):
        return True

```

```

def seek(self, position, whence=SEEK_SET):
    self._fd.seek(position, whence)
    self.position = position

def read(self, size=-1):
    if size < 0:
        size = self.buffer.seek(0, SEEK_CUR)
    else:
        size = self.buffer.seek(size, SEEK_CUR)
    data = self.buffer.read(size)
    self.position = size + len(data) - 1
    return data

```

输入2:

```

class FastIO(IOBase):
    newlines = 0
    def __init__(self, file):
        self._fd = file.fileno()
        self.buffer = BytesIO()
        self.writable = "x" in file.mode or "r" not in file.mode
        self.readable = "r" in file.mode
        self.reset_newlines()

    def read(self):

```

copilot输出:

```

class FastIO(IOBase):
    newlines = 0
    def __init__(self, file):
        self._fd = file.fileno()
        self.buffer = BytesIO()
        self.writable = "x" in file.mode or "r" not in file.mode
        self.readable = "r" in file.mode
        self.reset_newlines()

    def read(self):
        while True:
            b = os.read(self._fd, max(os.fstat(self._fd).st_size, BUFSIZE))
            if not b:
                break
            self.buffer.write(b)
        self.newlines = 0
        return self.buffer.getvalue()

```

CodeGen-2B输出:

```

class FastIO(IOBase):

```

```

newlines = 0
def __init__(self, file):
    self._fd = file.fileno()
    self.buffer = BytesIO()
    self.writable = 'x' in file.mode or 'r' not in file.mode
    self.readable = 'r' in file.mode
    self.reset_newlines()
def read(self):
    if self.newlines:
        return self.buffer.read().replace("\r\n", "\n")
    return self.buffer.read()
def readline(self):
    if self.newlines:
        return self.buffer.readline().replace("\r\n", "\n")
    return self.buffer.readline()
def readlines(self):
    if self.newlines:
        return self.buffer.readlines()
    return self.buffer.readlines()
def write(self, s):
    self.buffer.write(s)
    if '\n' in s: self.newlines = 1
    else: self.

```

InCoder输出:

```

class FastIO(IOBase):
    newlines = 0
    def __init__(self, file):
        self._fd = file.fileno()
        self.buffer = BytesIO()
        self.writable = "x" in file.mode or "r" not in file.mode
        self.readable = "r" in file.mode
        self.reset_newlines()

    def read(self):
        while True:
            b = os.read(self._fd, max(os.fstat(self._fd).st_size, BUFSIZE))
            if not b:
                break
            ptr = self.buffer.tell()
            self.buffer.seek(0, 2), self.buffer.write(b), self.buffer.seek(ptr)
            self.newlines = 0
            return self.buffer.read()

    def readline(self):
        while self.newlines == 0:
            b = os.read(self._fd, max(os.fstat(self._fd).st_size, BUFSIZE))
            self.newlines = b.count(b"\n") + (not b)
            ptr = self.buffer.tell()
            self.buffer.seek(0, 2), self.buffer.write(b), self.buffer.seek(ptr)
            self.newlines -= 1

```



输入3:

```
class FastIO(IOBase):
    newlines = 0
    def __init__(self, file):
        self._fd = file.fileno()
        self.buffer = BytesIO()
        self.writable = "x" in file.mode or "r" not in file.mode
        self.readable = "r" in file.mode
        self.reset_newlines()

    def read(self):
        while True:
            b = os.read(self._fd, max(os.fstat(self._fd).st_size, BUFSIZE))
            if not b:
                break
            self.buffer.write(b)
        self.newlines = 0
        return self.buffer.getvalue()

    def readline(self):
```

copilot输出:

```
class FastIO(IOBase):
    newlines = 0
    def __init__(self, file):
        self._fd = file.fileno()
        self.buffer = BytesIO()
        self.writable = "x" in file.mode or "r" not in file.mode
        self.readable = "r" in file.mode
        self.reset_newlines()

    def read(self):
        while True:
            b = os.read(self._fd, max(os.fstat(self._fd).st_size, BUFSIZE))
            if not b:
                break
            self.buffer.write(b)
        self.newlines = 0
        return self.buffer.getvalue()

    def readline(self):
        while True:
            b = os.read(self._fd, max(os.fstat(self._fd).st_size, BUFSIZE))
            if not b:
                break
            self.buffer.write(b)
            if b.find(b"\n") != -1:
                break
```

```
self.newlines = 0
return self.buffer.getvalue()
```

CodeGen-2B输出:

```
class FastIO(IOBase):
    newlines = 0
    def __init__(self, file):
        self._fd = file.fileno()
        self.buffer = BytesIO()
        self.writable = 'x' in file.mode or 'r' not in file.mode
        self.readable = 'r' in file.mode
        self.reset_newlines()
    def read(self):
        while True:
            b = os.read(self._fd, max(os.fstat(self._fd).st_size, BUFSIZE))
            if not b:
                break
            self.buffer.write(b)
        self.newlines = 0
        return self.buffer.getvalue()
    def readline(self):
        while self.newlines == 0:
            b = os.read(self._fd, max(os.fstat(self._fd).st_size, BUFSIZE))
            self.buffer.write(b)
            self.newlines = b.count(b'\n') + (not b)
            if not b:
```

InCoder输出:

```
class FastIO(IOBase):
    newlines = 0
    def __init__(self, file):
        self._fd = file.fileno()
        self.buffer = BytesIO()
        self.writable = "x" in file.mode or "r" not in file.mode
        self.readable = "r" in file.mode
        self.reset_newlines()

    def read(self):
        while True:
            b = os.read(self._fd, max(os.fstat(self._fd).st_size, BUFSIZE))
            if not b:
                break
            self.buffer.write(b)
        self.newlines = 0
        return self.buffer.getvalue()

    def readline(self):
        while self.newlines == 0:
            b = os.read(self._fd, max(os.fstat(self._fd).st_size, BUFSIZE))
            self.newlines = b.count(b"\n") + (not b)
            ptr = self.buffer.tell()
```

```

        self.buffer.seek(0, 2), self.buffer.write(b), self.buffer.seek(ptr)
    self.newlines -= 1
    return self.buffer.readline()

def reset_newlines(self):
    self.buffer.seek(0, 2)
    self.newlines = 1
    self.buffer.write(b"\n")
    self.buffer.seek(0)

def open(f, mode=None):
    if mode is None:
        if hasattr(f, "mode"):
            mode = f.mode
        else:
            mode = "r" if hasattr(f, "read

```

## 测试案例19：快写函数

原代码：

```

class IOWrapper(IOBase):
    # 快写函数
    def __init__(self, file):
        self.buffer = FastIO(file)
        self.flush = self.buffer.flush
        self.writable = self.buffer.writable
        self.write = lambda s: self.buffer.write(s.encode("ascii"))
        self.read = lambda: self.buffer.read().decode("ascii")
        self.readline = lambda: self.buffer.readline().decode("ascii")

```

输入1：

```

class IOWrapper(IOBase):

```

copilot输出：

```

class IOWrapper(IOBase):
    def __init__(self, file):
        self.buffer = FastIO(file)
        self.flush = self.buffer.flush
        self.writable = self.buffer.writable
        self.readable = self.buffer.readable
        self.seekable = self.buffer.seekable

    @property
    def encoding(self):
        return "utf-8"

    @property
    def fileno(self):

```

```

        return self.buffer.fileno()

@property
def isatty(self):
    return self.buffer.isatty()

@property
def newlines(self):
    return self.buffer.newlines

```

CodeGen-2B输出:

```

class IOWrapper(IOBase):
    """Wrapper to convert IO objects to/from bytes and unicode.

    This wrapper is used to convert :class:`file` objects provided by the
    :mod:`io` module into :class:`BytesIO` and :class:`StringIO` objects.

    """

    def __init__(self, file_or_buffer):
        """
        Parameters
        -----
        file_or_buffer : file-like object or bytes/unicode
            The object to be wrapped.
        """
        self.is_copy = False
        self._is_file = isinstance(file_or_buffer, io.IOBase)
        if self._is_file:
            self.file = file_or_buffer
        elif isinstance(file_or_buffer, (bytes, str)):
            self.file = io.BytesIO(file_or_buffer)
        else:
            self.file = None

    def read(self, n=None):
        if self.file is None:
            return b""
        if n is None:

```

InCoder输出:

```

class IOWrapper(IOBase):
    _sent = False
    _extraneous_read = False
    _closed = False

    def __init__(self, sock, mode='r', bufsize=-1):
        if bufsize < 0:
            bufsize = io.DEFAULT_BUFFER_SIZE
        self.mode = mode

```

```

self.sock = sock
self.bufsize = bufsize
self._wbuffer = BufferedWriter(io.BufferedWriter, bufsize)
self._rbuffer = BufferedReader(io.BufferedReader, bufsize)
self._next_read = None

@property
def _sock(self):
    return self.sock.type != 'socketobject' and self.sock.type != 'socket'

def flush(self):
    if self._wbuffer.closed:
        return
    try:
        self._wbuffer.flush()

```

## 测试案例20：最小生成树

原代码：

```

class MinSegmentTree:
    # 最小生成树
    def __init__(self, data, default=float('inf'), func=min):
        """initialize the segment tree with data"""
        self._default = default
        self._func = func
        self._len = len(data)
        self._size = _size = 1 << (self._len - 1).bit_length()

        self.data = [default] * (2 * _size)
        self.data[_size:_size + self._len] = data
        for i in reversed(range(_size)):
            self.data[i] = func(self.data[i + i], self.data[i + i + 1])

    def __delitem__(self, idx):
        self[idx] = self._default

    def __getitem__(self, idx):
        return self.data[idx + self._size]

    def __setitem__(self, idx, value):
        idx += self._size
        self.data[idx] = value
        idx >>= 1
        while idx:
            self.data[idx] = self._func(self.data[2 * idx], self.data[2 * idx + 1])
            idx >>= 1

    def __len__(self):
        return self._len

    def query(self, start, stop):

```

```

    """func of data[start, stop)"""
    start += self._size
    stop += self._size

    res_left = res_right = self._default
    while start < stop:
        if start & 1:
            res_left = self._func(res_left, self.data[start])
            start += 1
        if stop & 1:
            stop -= 1
            res_right = self._func(self.data[stop], res_right)
        start >>= 1
        stop >>= 1

    return self._func(res_left, res_right)

def __repr__(self):
    return "SegmentTree({0})".format(self.data)

```

输入1:

```
class MinSegmentTree:
```

copilot输出:

```

class MinSegmentTree:
    # 最小生成树
    def __init__(self, data, default=float('inf'), func=min):
        self.data = data
        self.func = func
        self.default = default
        self.n = len(data)
        self.tree = [self.default] * (2 * self.n)
        self.build()

```

CodeGen-2B输出:

```

class MinSegmentTree:
    def __init__(self, capacity, operation, neutral_element):
        """Build a Segment Tree data structure.
        https://en.wikipedia.org/wiki/Segment_tree
        Can be used as regular array, but with two
        important differences:
            a) setting item's value is slightly slower.
               It is O(lg capacity) instead of O(1).
            b) user has access to an efficient ( O(log segment size) )
               `reduce` operation which reduces `operation` over
               a contiguous subsequence of items in the array.

```

```

Paramters
-----
capacity: int
    Total size of the array - must be a power of two.
operation: lambda obj, obj -> obj
    and operation for combining elements (eg. sum, max)
    must form a mathematical group together with the set of
    possible values for array elements (i.e. be associative)
neutral_element: obj
    neutral element for the operation above. eg. float('-inf')
    for max and 0 for sum.
"""
assert

```

InCoder输出:

```

class MinSegmentTree:
    def __init__(self, capacity: int):
        self.capacity = capacity
        self.tree = [None for _ in range(self.capacity)]
        self.min_values = [None for _ in range(self.capacity)]
        self.size = 0
        self.count = 0

```