سند بررسی کد سوال اول تمرین ۳

امیرمهدی دارائی - ۹۹۱۰۵۴۳۱

تحلیل زمان‌بندی وظایف

این سند به تحلیل زمان‌بندی وظایف با استفاده از پارامترهای مختلف می‌پردازد. کد به بخش‌های مختلف تقسیم شده است تا خوانایی و درک آن ساده‌تر باشد. در زیر توضیح مختصری از هر کلاس و تابع ارائه شده است.

کلاس Task

کلاس Task نمایانگر یک وظیفه منفرد با مشخصات زیر است:

- **task_id:** شناسه‌ی یکتا برای وظیفه.
- **exec_time:** زمان اجرای وظیفه.
- **deadline:** مهلت انجام وظیفه.
- **period:** دوره‌ی وظیفه.
- **utilization:** بهره‌وری وظیفه که از تقسیم زمان اجرا بر دوره به دست می‌آید.

```python
class Task:
    """Represents a single task with its attributes and methods."""

    def __init__(self, task_id, exec_time, deadline, period):
        """
        Initializes a Task instance.

        Args:
            task_id (int): Unique identifier for the task.
            exec_time (int): Execution time of the task.
            deadline (int): Deadline of the task.
            period (int): Period of the task.
        """
        self.task_id = task_id
        self.exec_time = exec_time
        self.deadline = deadline
        self.period = period
        self.utilization = exec_time / period
```

این کلاس شامل توابع زیر است:

- **get_utilization:** این تابع بهره‌وری وظیفه را بازمی‌گرداند. این بهره‌وری نشان‌دهنده‌ی میزان استفاده از منابع برای اجرای وظیفه در دوره مشخص شده است.
- **get_deadlines:** این تابع لیستی از مهلت‌های وظیفه تا حداکثر مهلت ممکن را تولید می‌کند. مهلت‌ها به صورت تکراری با توجه به دوره وظیفه محاسبه می‌شوند.

```python
def get_utilization(self):
    """
    Returns the utilization of the task.

    Returns:
        float: Utilization of the task.
    """
    return self.utilization

def get_deadlines(self, max_possible_deadline):
    """
    Generates a list of deadlines up to the maximum possible deadline.

    Args:
        max_possible_deadline (int): The maximum possible deadline.

    Returns:
        list: List of deadlines.
    """
    deadlines = []
    deadline = self.deadline
    while deadline < max_possible_deadline:
        deadlines.append(deadline)
        deadline += self.period
    return deadlines
```

کلاس TaskScheduler

کلاس TaskScheduler مسئول مدیریت مجموعه‌ای از وظایف و انجام تحلیل‌های زمان‌بندی است.

```python
class TaskScheduler:
    """Handles a collection of tasks and performs scheduling analysis."""

    def __init__(self, tasks):
        """
        Initializes the TaskScheduler with a list of tasks.

        Args:
            tasks (list): List of task dictionaries.
        """
        self.tasks = [Task(**task) for task in tasks]
```

این کلاس شامل وظایف زیر است:

- **total_utilization**: محاسبه‌ی کل بهره‌وری تمامی وظایف. این تابع جمع کل بهره‌وری وظایف را محاسبه می‌کند که نشان‌دهنده‌ی میزان استفاده از منابع توسط همه وظایف است.

```python
  def total_utilization(self):
      """
      Calculates the total utilization of all tasks.

      Returns:
          float: Total utilization of all tasks.
      """
      return sum(task.get_utilization() for task in self.tasks)
```

- **compute_l_star**: محاسبه‌ی زمان بحرانی (L*) برای وظایف. این زمان حد بحرانی است که وظایف باید در آن انجام شوند تا زمان‌بندی درست انجام شود.

```
def compute_l_star(self):
    """
    Computes the critical time L* for the tasks.

    Returns:
        int: The critical time L*.
    """
    l_star = sum((task.period - task.deadline) * task.utilization for task in self.tasks)
    return math.ceil(l_star / (1 - self.total_utilization()))
```

- **compute_hyper_period**: محاسبه‌ی دوره‌ی بزرگ (H) وظایف. این دوره کمترین دوره‌ای است که تمامی وظایف در آن تکرار می‌شوند.

```
def compute_hyper_period(self):
    """
    Computes the hyper period (H) of the tasks.

    Returns:
        int: The hyper period.
    """
    periods = [task.period for task in self.tasks]
    return math.lcm(*periods)
```

- **compute_deadlines**: محاسبه‌ی لیستی از مهلت‌های یکتا برای وظایف. این تابع تمامی مهلت‌های تکراری وظایف را محاسبه کرده و یک لیست یکتا از آن‌ها تولید می‌کند.

```
def compute_deadlines(self):
    """
    Computes a list of unique deadlines for the tasks.

    Returns:
        list: List of unique deadlines.
    """
    max_deadline = max(task.deadline for task in self.tasks)
    hyper_period = self.compute_hyper_period()
    l_star = self.compute_l_star()
    max_possible_deadline = min(hyper_period, max(max_deadline, l_star))

    deadlines = []
    for task in self.tasks:
        deadlines.extend(task.get_deadlines(max_possible_deadline))
    return list(set(deadlines))
```

- **compute_g**: محاسبه‌ی تابع بار تقاضا (g) در زمان L. این تابع میزان بار تقاضا را در زمان مشخص L محاسبه می‌کند که برای تحلیل زمان‌بندی استفاده می‌شود.

```
def compute_g(self, l):
    """
    Computes the demand bound function g at time L.

    Args:
        l (int): The time L.

    Returns:
        int: The demand bound function g at time L.
    """
    return sum(math.floor((l + task.period - task.deadline) / task.period) * task.exec_time for task in self.tasks)
```

- **run**: اجرای تست کاهش فواصل و تولید جدول نتایج. این تابع فواصل زمانی را بررسی کرده و نتایج تست زمان‌بندی را در قالب یک جدول ارائه می‌دهد.

```
def run(self):
    """
    Runs the reducing test intervals and generates a result table.

    Returns:
        list: Result table of test intervals with L, g(0, L), and result.
    """
    result_table = []
    deadlines = sorted(self.compute_deadlines())
    for l in deadlines:
        g_l = self.compute_g(l)
        result = 'OK' if g_l <= l else 'NOT OK'
        result_table.append([l, g_l, result])
        if result == 'NOT OK':
            break
    return result_table
```

تابع main

تابع main وظایف زیر را انجام می‌دهد:

- **خواندن داده‌های وظیفه:** خواندن داده‌های وظیفه از فایل JSON مشخص شده.
- **ایجاد نمونه‌ی TaskScheduler:** ایجاد نمونه‌ای از کلاس TaskScheduler با وظایف خوانده شده.
- **محاسبه‌ی بهره‌وری:** محاسبه‌ی کل بهره‌وری وظایف.
- **محاسبه‌ی زمان بحرانی:** محاسبه‌ی زمان بحرانی (L*).
- **محاسبه‌ی دوره‌ی بزرگ:** محاسبه‌ی دوره‌ی بزرگ (H).
- **محاسبه‌ی مهلت‌ها:** محاسبه‌ی لیست مهلت‌ها.
- **اجرای تست:** اجرای تست کاهش فواصل برای بررسی امکان زمان‌بندی وظایف.
- **چاپ خلاصه و نتایج تحلیلی:** چاپ خلاصه‌ی نتایج و تحلیل‌های دقیق در قالب جدول.

```python
def main():
    """
    Main function to read task data, initialize scheduler, and print results.
    """
    with open(FILE_PATH) as f:
        tasks_data = json.load(f)
    scheduler = TaskScheduler(tasks_data)

    utilization = scheduler.total_utilization()
    l_star = scheduler.compute_l_star()
    hyper_period = scheduler.compute_hyper_period()
    deadlines = scheduler.compute_deadlines()
    test_intervals = scheduler.run()

    print("Summary of Results")
    print("------------------")
    print(f"Total Utilization (U) = {utilization:.2f}")
    print(f"Critical Time (L*) = {l_star}")
    print(f"Hyper Period (H) = {hyper_period}")
    print(f"Deadlines (D) = {', '.join(map(str, deadlines))}\n")

    print("Detailed Analysis")
    print("-----------------")
    print(tabulate(test_intervals, headers=['L', 'g(0, L)', 'Result'], tablefmt="fancy_grid"))


if __name__ == '__main__':
    main()
```

لغت‌نامه

- **بهره‌وری کل (Total Utilization)**: میزان بهره‌وری کلی وظایف که از جمع بهره‌وری تک تک وظایف به دست می‌آید.
- **زمان بحرانی (Critical Time)**: زمانی که به عنوان حد بحرانی برای اجرای وظایف در نظر گرفته می‌شود.
- **دوره‌ی بزرگ (Hyper Period)**: کوچک‌ترین دوره‌ای که وظایف در آن تکرار می‌شوند.
- **تابع بار تقاضا (Demand Bound Function)**: تابعی که میزان بار تقاضا را در یک زمان مشخص محاسبه می‌کند.