Завдання для лабораторної роботи:

1. **Обчислити час відгуку і час обробки для трьох завдань довжиною 200 з використанням алгоритмів SJF та FIFO.**

Завдання:

- Маємо три завдання: Job 0, Job 1 та Job 2.
- Довжина кожного завдання = 200.

1. Алгоритм FIFO (First In, First Out)

Визначення:
FIFO — це алгоритм планування, який виконує завдання в порядку їх надходження (першим виконується перше за чергою завдання).
Кроки для обчислення:

- Час відгуку (Response Time) для кожного завдання в FIFO буде рівний моменту часу, коли завдання починає виконуватись. Оскільки завдання виконуються по черзі, час відгуку для кожного наступного завдання буде рівним часу, коли попереднє завдання завершується.
- Час обробки (Turnaround Time) — це час, що проходить від моменту надходження завдання до моменту його завершення. Оскільки всі завдання мають однакову довжину, це буде просто час завершення кожного завдання.

Обчислення:

1. Job 0:
   - Response Time = 0 (воно починається одразу).
   - Turnaround Time = 200 (завершується через 200 одиниць часу).
2. Job 1:
   - Response Time = 200 (воно починається після завершення Job 0).
   - Turnaround Time = 400 (завершується через 200 одиниць часу після початку).
3. Job 2:
   - Response Time = 400 (воно починається після завершення Job 1).
   - Turnaround Time = 600 (завершується через 200 одиниць часу після початку).

Загальний підсумок для FIFO:

- Середній час відгуку = (0 + 200 + 400) / 3 = 200.
- Середній час обробки = (200 + 400 + 600) / 3 = 400.

```
sophiamikhaylova@MacBook-Pro-Sophia Downloads % python3 scheduler.py -c SJF -l 200,200,200

ARG policy FIFO
ARG jlist 200,200,200

Here is the job list, with the run time of each job:
  Job 0 ( length = 200.0 )
  Job 1 ( length = 200.0 )
  Job 2 ( length = 200.0 )


** Solutions **

Execution trace:
  [ time   0 ] Run job 0 for 200.00 secs ( DONE at 200.00 )
  [ time 200 ] Run job 1 for 200.00 secs ( DONE at 400.00 )
  [ time 400 ] Run job 2 for 200.00 secs ( DONE at 600.00 )

Final statistics:
  Job   0 -- Response: 0.00  Turnaround 200.00  Wait 0.00
  Job   1 -- Response: 200.00  Turnaround 400.00  Wait 200.00
  Job   2 -- Response: 400.00  Turnaround 600.00  Wait 400.00

  Average -- Response: 200.00  Turnaround 400.00  Wait 200.00
```

Відповідь збігається з результатами виконання sheduler.py із заданими параметрами для FIFO довжиною 200

2. Алгоритм SJF (Shortest Job First)

Визначення:
SJF — це алгоритм планування, який виконує завдання в порядку їх довжини, тобто спочатку виконуються завдання з найменшою тривалістю. У нашому випадку всі завдання мають однакову довжину, тому алгоритм SJFбуде працювати так само, як FIFO, і обробить завдання в порядку їх надходження.
Обчислення:

- Job 0:
  - Response Time = 0 (воно починається одразу).
  - Turnaround Time = 200 (завершується через 200 одиниць часу).
- Job 1:
  - Response Time = 200 (починається після завершення Job 0).
  - Turnaround Time = 400 (завершується через 200 одиниць часу після початку).
- Job 2:
  - Response Time = 400 (починається після завершення Job 1).
  - Turnaround Time = 600 (завершується через 200 одиниць часу після початку).

Загальний підсумок для SJF:

- Середній час відгуку = (0 + 200 + 400) / 3 = 200.
- Середній час обробки = (200 + 400 + 600) / 3 = 400.

Висновок:

- Оскільки всі завдання мають однакову тривалість, алгоритм SJF працює точно так само, як FIFO, і результати будуть однаковими:
  - Середній час відгуку = 200.
  - Середній час обробки = 400

```
sophiamikhaylova@MacBook-Pro-Sophia Downloads % python3 scheduler.py -c SJF -l 200,200,200

ARG policy FIFO
ARG jlist 200,200,200

Here is the job list, with the run time of each job:
  Job 0 ( length = 200.0 )
  Job 1 ( length = 200.0 )
  Job 2 ( length = 200.0 )


** Solutions **

Execution trace:
  [ time   0 ] Run job 0 for 200.00 secs ( DONE at 200.00 )
  [ time 200 ] Run job 1 for 200.00 secs ( DONE at 400.00 )
  [ time 400 ] Run job 2 for 200.00 secs ( DONE at 600.00 )

Final statistics:
  Job   0 -- Response: 0.00  Turnaround 200.00  Wait 0.00
  Job   1 -- Response: 200.00  Turnaround 400.00  Wait 200.00
  Job   2 -- Response: 400.00  Turnaround 600.00  Wait 400.00

  Average -- Response: 200.00  Turnaround 400.00  Wait 200.00
```

Відповідь збігається з результатами виконання sheduler.py із заданими параметрами для FIFO довжиною 200

2. Те саме, але з різними довжинами завдань: 100, 200 та 300.

Завдання:

- Job 0: довжина 100.
- Job 1: довжина 200.
- Job 2: довжина 300.

1. Алгоритм FIFO (First In, First Out)

Визначення:

У FIFO завдання виконуються в порядку їх надходження. Для наших завдань:

- Job 0 буде виконуватись першим.
- Потім виконується Job 1.
- І на завершення виконується Job 2.

Обчислення:

- Job 0:
  - Response Time = 0 (воно починається одразу).
  - Turnaround Time = 100 (завершується через 100 одиниць часу).
- Job 1:
  - Response Time = 100 (воно починається після завершення Job 0).
  - Turnaround Time = 300 (завершується через 200 одиниць часу).
- Job 2:
  - Response Time = 300 (воно починається після завершення Job 1).
  - Turnaround Time = 600 (завершується через 300 одиниць часу).

Загальний підсумок для FIFO:

- Середній час відгуку = (0 + 100 + 300) / 3 = 133.33
- Середній час обробки = (100 + 300 + 600) / 3 = 333.33

```
sophiamikhaylova@MacBook-Pro-Sophia Downloads % python3 scheduler.py -c  FIFO -l 100,200,300

ARG policy FIFO
ARG jlist 100,200,300

Here is the job list, with the run time of each job:
  Job 0 ( length = 100.0 )
  Job 1 ( length = 200.0 )
  Job 2 ( length = 300.0 )


** Solutions **

Execution trace:
  [ time   0 ] Run job 0 for 100.00 secs ( DONE at 100.00 )
  [ time 100 ] Run job 1 for 200.00 secs ( DONE at 300.00 )
  [ time 300 ] Run job 2 for 300.00 secs ( DONE at 600.00 )

Final statistics:
  Job   0 -- Response: 0.00  Turnaround 100.00  Wait 0.00
  Job   1 -- Response: 100.00  Turnaround 300.00  Wait 100.00
  Job   2 -- Response: 300.00  Turnaround 600.00  Wait 300.00

  Average -- Response: 133.33  Turnaround 333.33  Wait 133.33
```

Відповідь збігається з результатами виконання sheduler.py із заданими параметрами для FIFO довжиною 100, 200, 300

2. Алгоритм SJF (Shortest Job First)

Визначення:

У SJF завдання виконуються в порядку їх тривалості, тобто спочатку буде виконуватись завдання з найменшою тривалістю. У нашому випадку, це означає, що завдання будуть виконуватись в такому порядку:

1. Job 0 (довжина 100),
2. Job 1 (довжина 200),
3. Job 2 (довжина 300).

Обчислення:

- Job 0:
  - Response Time = 0 (воно починається одразу).
  - Turnaround Time = 100 (завершується через 100 одиниць часу).
- Job 1:
  - Response Time = 100 (воно починається після завершення Job 0).
  - Turnaround Time = 300 (завершується через 200 одиниць часу).
- Job 2:
  - Response Time = 300 (воно починається після завершення Job 1).
  - Turnaround Time = 600 (завершується через 300 одиниць часу).

Загальний підсумок для SJF:

- Середній час відгуку = (0 + 100 + 300) / 3 = 133.33
- Середній час обробки = (100 + 300 + 600) / 3 = 333.33

```
sophiamikhaylova@MacBook-Pro-Sophia Downloads % python3 scheduler.py -c SJF -l 100,200,300

ARG policy FIFO
ARG jlist 100,200,300

Here is the job list, with the run time of each job:
  Job 0 ( length = 100.0 )
  Job 1 ( length = 200.0 )
  Job 2 ( length = 300.0 )


** Solutions **

Execution trace:
  [ time   0 ] Run job 0 for 100.00 secs ( DONE at 100.00 )
  [ time 100 ] Run job 1 for 200.00 secs ( DONE at 300.00 )
  [ time 300 ] Run job 2 for 300.00 secs ( DONE at 600.00 )

Final statistics:
  Job   0 -- Response: 0.00  Turnaround 100.00  Wait 0.00
  Job   1 -- Response: 100.00  Turnaround 300.00  Wait 100.00
  Job   2 -- Response: 300.00  Turnaround 600.00  Wait 300.00

  Average -- Response: 133.33  Turnaround 333.33  Wait 133.33
```

Відповідь збігається з результатами виконання sheduler.py із заданими параметрами для FIFO довжиною 100, 200, 300

Висновок:

Для FIFO та SJF:

- Обидва алгоритми працюють однаково, оскільки завдання мають різну довжину (100, 200, 300), і порядок виконання завдань у FIFO і SJF в цьому випадку збігається.
- Результати для FIFO та SJF будуть однаковими, а саме:
  - Середній час відгуку = 133.33.
  - Середній час обробки = 333.33.

4. **Те саме, але додатково з використанням алгоритму Round Robin (RR) з тайм-слайсом 1.**

1. Кожне завдання виконується по черзі з тайм-слайсом 1.
2. Якщо завдання не завершене після 1 одиниці часу, воно ставиться в кінець черги, і виконання переходить до наступного завдання.
3. Response Time — це час з моменту надходження завдання до моменту його першого запуску.
4. Turnaround Time — це час з моменту надходження завдання до його завершення.
5. Wait Time — це час, який завдання очікує в черзі до свого запуску.

Завдання:

- Job 0: довжина 100
- Job 1: довжина 200
- Job 2: довжина 300

Визначення Response Time, Turnaround Time та Wait Time:

1. Job 0 (довжина 100):
   - Response Time: 0 — це перше завдання, воно починається одразу.
   - Turnaround Time: 100 — воно завершується через 100 одиниць часу.
   - Wait Time: 0 — немає часу очікування.
2. Job 1 (довжина 200):
   - Response Time: 1 — воно починається після того, як Job 0 використає перший тайм-слайс.
   - Turnaround Time: 299 — після виконання на кожному циклі по 1 одиниці часу, воно завершується на 299-му циклі.
   - Wait Time: 198 — воно чекає на свій хід, поки Job 0 завершить свою роботу, потім Job 2.
3. Job 2 (довжина 300):
   - Response Time: 2 — воно починається після завершення Job 0 та Job 1.
   - Turnaround Time: 599 — завершиться після 300 одиниць часу.
   - Wait Time: 299 — воно чекає, поки Job 0 і Job 1 завершаться, а потім починає виконання.

Розрахунок:

Загальний підсумок для Round Robin (RR) з тайм-слайсом 1:

- Середній час відгуку: (0 + 1 + 2) / 3 = 1.00
- Середній час обробки (Turnaround Time): (298 + 499 + 600) / 3 = 465.67

```
sophiamikhaylova@MacBook-Pro-Sophia Downloads % python3 scheduler.py -p RR -l 100,200,300 -q 1 -c

ARG policy RR
ARG jlist 100,200,300

Here is the job list, with the run time of each job:
  Job 0 ( length = 100.0 )
  Job 1 ( length = 200.0 )
  Job 2 ( length = 300.0 )
```

```
Final statistics:
  Job   0 -- Response: 0.00  Turnaround 298.00  Wait 198.00
  Job   1 -- Response: 1.00  Turnaround 499.00  Wait 299.00
  Job   2 -- Response: 2.00  Turnaround 600.00  Wait 300.00

  Average -- Response: 1.00  Turnaround 465.67  Wait 265.67

sophiamikhaylova@MacBook-Pro-Sophia Downloads %
```

- Середній час очікування (Wait Time): (198 + 299 + 300) / 3 = 265.67

Відповідь збігається з результатами виконання sheduler.py із заданими параметрами для RR довжиною 100, 200, 300

4. Для яких типів навантажень SJF дає такий самий час обробки, як FIFO?

Алгоритм Shortest Job First (SJF) дає такий самий час обробки (turnaround time), як FIFO в тих випадках, коли всі завдання мають однакову довжину. Ось чому:

Чому SJF і FIFO дають однакові результати при однакових довжинах завдань:

1. FIFO планує завдання в порядку їх надходження, тобто перше завдання виконується першим, друге — після нього, і так далі. Час обробки для кожного завдання в FIFO буде залежати від його початкового моменту (коли воно почало виконуватись) і тривалості попередніх завдань.
2. SJF (Shortest Job First) планує завдання в порядку їх тривалості. Завдання з найменшою тривалістю виконується першим. Якщо всі завдання мають однакову тривалість, то порядок виконання в SJF не змінюється порівняно з FIFO, оскільки всі завдання є однаковими за довжиною і можуть бути оброблені в тому ж порядку, що й у FIFO.

Як виглядає ситуація:

- Якщо є три завдання з однаковими тривалостями, наприклад:
  - Job 1 — довжина 200,
  - Job 2 — довжина 200,
  - Job 3 — довжина 200.
- У FIFO вони виконуватимуться в порядку надходження:
  - Job 1 почнеться на часі 0, завершиться на 200.
  - Job 2 почнеться на 200, завершиться на 400.
  - Job 3 почнеться на 400, завершиться на 600.
- У SJF також не буде змін у порядку, оскільки всі завдання мають однакову довжину:
  - Job 1 почнеться на 0, завершиться на 200.
  - Job 2 почнеться на 200, завершиться на 400.
  - Job 3 почнеться на 400, завершиться на 600.

Отже, коли завдання мають однакову довжину, SJF працює точно так само, як FIFO, і обидва алгоритми дають однаковий середній час обробки.

Висновок:
SJF дає такий самий час обробки, як FIFO, тільки в тих випадках, коли всі завдання мають однакову тривалість. В інших випадках SJF може дати кращі результати (швидше завершення коротших завдань), ніж FIFO.

5. Для яких типів навантажень та довжин тайм-слайсів SJF дає такий самий час відгуку, як RR?

Shortest Job First (SJF) та Round Robin (RR) можуть давати однаковий час відгуку, якщо кілька умов виконуються:

1. Завдання мають однакову довжину або майже однакову (для SJF), і у кожного завдання є тільки одна можливість отримати процесор перед тим, як інші завдання будуть виконані.
2. Тайм-слайс у RR має значення, яке не обмежує виконання завдань перед тим, як вони не завершать свою роботу. Тобто, кожне завдання повинно мати або достатньо великий тайм-слайс для виконання без розривів, або ці завдання повинні бути короткими й не потребувати додаткових циклів.

Як це працює:

- Якщо у RR використовується достатньо великий тайм-слайс, який дозволяє кожному завданню виконатись до завершення, то кожне завдання буде виконуватись без перерв, так само як у SJF, де коротші завдання виконуються першими.
- Якщо SJF та RR виконують завдання за таким самим порядком (наприклад, всі завдання мають однакову довжину), то обидва алгоритми мають однаковий час відгуку.

Запускаємо програму для однакових завдань та порівнюємо результати.

```
[sophiamikhaylova@MacBook-Pro-Sophia Downloads % python3 scheduler.py -p RR -l 100,100,100 -q 100 -c

ARG policy RR
ARG jlist 100,100,100

Here is the job list, with the run time of each job:
  Job 0 ( length = 100.0 )
  Job 1 ( length = 100.0 )
  Job 2 ( length = 100.0 )


** Solutions **

Execution trace:
  [ time   0 ] Run job   0 for 100.00 secs ( DONE at 100.00 )
  [ time 100 ] Run job   1 for 100.00 secs ( DONE at 200.00 )
  [ time 200 ] Run job   2 for 100.00 secs ( DONE at 300.00 )

Final statistics:
  Job   0 -- Response: 0.00  Turnaround 100.00  Wait 0.00
  Job   1 -- Response: 100.00  Turnaround 200.00  Wait 100.00
  Job   2 -- Response: 200.00  Turnaround 300.00  Wait 200.00

  Average -- Response: 100.00  Turnaround 200.00  Wait 100.00
```

```
[sophiamikhaylova@MacBook-Pro-Sophia Downloads % python3 scheduler.py -c SJF -l 100,100,100

ARG policy FIFO
ARG jlist 100,100,100

Here is the job list, with the run time of each job:
  Job 0 ( length = 100.0 )
  Job 1 ( length = 100.0 )
  Job 2 ( length = 100.0 )


** Solutions **

Execution trace:
  [ time   0 ] Run job 0 for 100.00 secs ( DONE at 100.00 )
  [ time 100 ] Run job 1 for 100.00 secs ( DONE at 200.00 )
  [ time 200 ] Run job 2 for 100.00 secs ( DONE at 300.00 )

Final statistics:
  Job   0 -- Response: 0.00  Turnaround 100.00  Wait 0.00
  Job   1 -- Response: 100.00  Turnaround 200.00  Wait 100.00
  Job   2 -- Response: 200.00  Turnaround 300.00  Wait 200.00

  Average -- Response: 100.00  Turnaround 200.00  Wait 100.00
```

- SJF: Оскільки всі завдання мають однакову довжину, SJF обробляє їх по черзі, і порядок виконання не змінюється.
- RR з тайм-слайсом 100: Оскільки тайм-слайс достатньо великий, щоб кожне завдання виконувалось до завершення за один цикл, алгоритм Round Robin працює подібно до FIFO і обробляє завдання в порядку їх надходження.

6. Що відбувається з часом відгуку при збільшенні довжини завдань у SJF? Чи можна це продемонструвати за допомогою симулятора?

Shortest Job First (SJF) — це алгоритм, який виконує завдання з найменшою довжиною першим. Основна ідея SJF полягає в тому, що коротші завдання обробляються першими, що мінімізує середній час обробки завдань.

Як збільшення довжини завдань впливає на час відгуку (Response Time)?

- Час відгуку — це час, який проходить від моменту надходження завдання до того, як воно починає виконуватись. Для SJF час відгуку залежить від того, чи є завдання короткими або довгими.

7. Короткі завдання: Якщо в системі є короткі завдання, вони будуть виконуватися першими, що може привести до короткого часу відгуку для цих завдань.
8. Довгі завдання: Якщо довгі завдання з'являються на початку, вони будуть виконуватися пізніше (через принцип SJF), що збільшує час відгуку для інших завдань.

Які зміни відбудуться при збільшенні довжини завдань?

- Коли довжина завдань збільшується, час відгуку для коротших завдань буде збільшуватися, оскільки ці завдання повинні будуть чекати довші завдання, перш ніж вони отримають процесор. Це може збільшити середній час відгуку для всіх завдань.
- Якщо довгі завдання з'являються першими, час відгуку для коротких завдань може збільшитись навіть більше, оскільки SJF завжди буде виконувати найкоротші завдання перше, а довгі будуть виконуватися останніми.

Демонстрація за допомогою scheduler.py

Тут ми маємо три завдання з різними довжинами — 50, 100 та 150. Це дозволить побачити, як SJF обробляє їх у порядку зростання довжини.

```
[sophiamikhaylova@MacBook-Pro-Sophia Downloads % python3 scheduler.py -p SJF -l 50,100,150 -c

ARG policy SJF
ARG jlist 50,100,150

Here is the job list, with the run time of each job:
  Job 0 ( length = 50.0 )
  Job 1 ( length = 100.0 )
  Job 2 ( length = 150.0 )


** Solutions **

Execution trace:
  [ time   0 ] Run job 0 for 50.00 secs ( DONE at 50.00 )
  [ time  50 ] Run job 1 for 100.00 secs ( DONE at 150.00 )
  [ time 150 ] Run job 2 for 150.00 secs ( DONE at 300.00 )

Final statistics:
  Job   0 -- Response: 0.00  Turnaround 50.00  Wait 0.00
  Job   1 -- Response: 50.00  Turnaround 150.00  Wait 50.00
  Job   2 -- Response: 150.00  Turnaround 300.00  Wait 150.00

  Average -- Response: 66.67  Turnaround 166.67  Wait 66.67
```

Завдання з більшими довжинами

```
sophiamikhaylova@MacBook-Pro-Sophia Downloads % python3 scheduler.py -p SJF -l 100,200,300 -c

ARG policy SJF
ARG jlist 100,200,300

Here is the job list, with the run time of each job:
  Job 0 ( length = 100.0 )
  Job 1 ( length = 200.0 )
  Job 2 ( length = 300.0 )


** Solutions **

Execution trace:
  [ time   0 ] Run job 0 for 100.00 secs ( DONE at 100.00 )
  [ time 100 ] Run job 1 for 200.00 secs ( DONE at 300.00 )
  [ time 300 ] Run job 2 for 300.00 secs ( DONE at 600.00 )

Final statistics:
  Job   0 -- Response: 0.00  Turnaround 100.00  Wait 0.00
  Job   1 -- Response: 100.00  Turnaround 300.00  Wait 100.00
  Job   2 -- Response: 300.00  Turnaround 600.00  Wait 300.00

  Average -- Response: 133.33  Turnaround 333.33  Wait 133.33
```

Тепер ми можемо побачити, як збільшення довжини завдань (від 50 до 300) впливає на час відгуку. Завдання з більшими довжинами будуть запускатися пізніше, що збільшить час відгуку для коротших завдань.

Пояснення результатів:

- Коли довжина завдань збільшується, час відгуку для коротших завдань стає більшим, оскільки довші завдання потребують більше часу для виконання. В результаті коротші завдання чекають довше, щоб отримати процесор.
- Зі збільшенням довжини завдань, середній час відгуку також збільшується, тому що довші завдання займають більше часу, і SJF обробляє їх пізніше, коли коротші завдання чекають на свою чергу.

Висновок:

SJF дає однаковий час відгуку для всіх завдань, коли їх довжина однакова. Проте, при збільшенні довжини завдань час відгуку для коротших завдань збільшується, оскільки вони повинні чекати довші завдання. За допомогою симулятора scheduler.py можна побачити це явище на прикладі різних наборів завдань.

7. Що відбувається з часом відгуку при збільшенні довжини тайм-слайса в RR? Чи можна написати рівняння для найгіршого часу відгуку, якщо є N завдань

Round Robin — це алгоритм, який надає процесору певний тайм-слайс для кожного завдання в черзі. Тривалість цього тайм-слайса визначає, скільки часу кожне завдання може виконуватись до того, як буде повернуте в кінець черги, і наступне завдання почне своє виконання.

Вплив збільшення довжини тайм-слайса:

1. Малий тайм-слайс:
   - Якщо тайм-слайс дуже малий, кожне завдання буде виконуватись лише протягом короткого часу і часто перемикатиметься між завданнями. Це може призвести до того, що час відгуку для завдань зросте, оскільки кожне завдання буде часто змінюватись і виконуватись за кілька інтервалів, а не за один.
   - Завдання, які мають більший час виконання, будуть часто відкладатись на чергу.
2. Великий тайм-слайс:
   - Якщо тайм-слайс значно збільшується (наприклад, більший за довжину завдання), алгоритм RRпочинає працювати більше як FIFO, де кожне завдання буде виконуватись до кінця в рамках одного циклу.
   - Це дозволяє зменшити час відгуку, оскільки завдання можуть бути виконані без перерв, однак для довших завдань це може збільшити чистий час виконання (turnaround time).

Вплив на час відгуку:

Час відгуку в RR залежить від кількості завдань у черзі і довжини тайм-слайса. Якщо тайм-слайс дуже малий, час відгуку для кожного завдання може зрости, оскільки завдання виконуються по частинах і мають багато перемикань. Коли тайм-слайс збільшується, кожне завдання отримує більше часу на виконання за один цикл, що може зменшити час відгуку.

Як змінюється час відгуку при збільшенні довжини тайм-слайса?

- При малих тайм-слайсах (наприклад, 1) кожне завдання виконується тільки частково, і довші завдання чекають на своє чергове виконання.
- При великих тайм-слайсах (наприклад, більше за довжину завдання) завдання отримують доступ до процесора на довший період, що дозволяє зменшити час відгуку.

Найгірший час відгуку для Round Robin з N завданнями:

1. Визначення найгіршого часу відгуку: У найгіршому випадку, коли кожне завдання виконується по черзі, час відгуку для кожного завдання збільшується. Якщо N завдань мають різні довжини, то найбільший час відгуку буде спостерігатись для самого останнього завдання, яке почне виконуватись тільки після того, як всі попередні завдання будуть виконані.
2. Формула для найгіршого часу відгуку: Якщо N завдань мають рівні довжини, то найгірший час відгуку буде для останнього завдання, і цей час буде пропорційний кількості завдань в черзі та довжині тайм-слайса. Тому для N завдань з однаковими довжинами, найгірший час відгуку можна обчислити за такою формулою:
3. $\text{Response Time}_{max} = (N-1) \times Q$, де:
    - N — кількість завдань,
    - Q — довжина тайм-слайса.

Пояснення:

- У найгіршому випадку завдання, яке потрапляє на кінець черги, мусить чекати, поки всі попередні $N-1$ завдань не отримають свій тайм-слайс.
- Кожне завдання виконується по черзі, і воно може отримувати доступ до процесора тільки після того, як всі попередні завдання отримають свою частину часу, тому для останнього завдання максимальний час відгуку буде $(N-1) \times Q$.

Демонстрація на scheduler.py:

Запускаємо програму для RR з різними значеннями тайм-слайса:

1.

```
[sophiamikhaylova@MacBook-Pro-Sophia Downloads % python3 scheduler.py -p RR -l 100,100,100 -q 1 -c

ARG policy RR
ARG jlist 100,100,100

Here is the job list, with the run time of each job:
  Job 0 ( length = 100.0 )
  Job 1 ( length = 100.0 )
  Job 2 ( length = 100.0 )
```

```
Final statistics:
  Job   0 -- Response: 0.00  Turnaround 298.00  Wait 198.00
  Job   1 -- Response: 1.00  Turnaround 299.00  Wait 199.00
  Job   2 -- Response: 2.00  Turnaround 300.00  Wait 200.00

  Average -- Response: 1.00  Turnaround 299.00  Wait 199.00
```

2.

```
sophiamikhaylova@MacBook-Pro-Sophia Downloads % python3 scheduler.py -p RR -l 100,100,100 -q 50 -c

ARG policy RR
ARG jlist 100,100,100

Here is the job list, with the run time of each job:
  Job 0 ( length = 100.0 )
  Job 1 ( length = 100.0 )
  Job 2 ( length = 100.0 )
```

```
Final statistics:
  Job   0 -- Response: 0.00  Turnaround 200.00  Wait 100.00
  Job   1 -- Response: 50.00  Turnaround 250.00  Wait 150.00
  Job   2 -- Response: 100.00  Turnaround 300.00  Wait 200.00

  Average -- Response: 50.00  Turnaround 250.00  Wait 150.00
```

3.

```
sophiamikhaylova@MacBook-Pro-Sophia Downloads % python3 scheduler.py -p RR -l 100,100,100 -q 100 -c

ARG policy RR
ARG jlist 100,100,100

Here is the job list, with the run time of each job:
  Job 0 ( length = 100.0 )
  Job 1 ( length = 100.0 )
  Job 2 ( length = 100.0 )


** Solutions **

Execution trace:
  [ time   0 ] Run job   0 for 100.00 secs ( DONE at 100.00 )
  [ time 100 ] Run job   1 for 100.00 secs ( DONE at 200.00 )
  [ time 200 ] Run job   2 for 100.00 secs ( DONE at 300.00 )

Final statistics:
  Job   0 -- Response: 0.00  Turnaround 100.00  Wait 0.00
  Job   1 -- Response: 100.00  Turnaround 200.00  Wait 100.00
  Job   2 -- Response: 200.00  Turnaround 300.00  Wait 200.00

  Average -- Response: 100.00  Turnaround 200.00  Wait 100.00
```

Висновок:

Збільшення тайм-слайса зменшує час відгуку в Round Robin, оскільки кожне завдання має більший час на виконання перед тим, як наступне завдання буде виконане. У найгіршому випадку для N завдань час відгуку останнього завдання буде пропорційний кількості завдань та довжині тайм-слайса.