A table of progress with numbers and letters

AI-generated content may be incorrect.

**Dựa trên thông tin từ đề bài và yêu cầu, tôi sẽ giải thích chi tiết các bước và kết quả để giải bài toán của bạn.**

**Phân tích đề bài:**

* **Sử dụng thuật toán Preemptive Priority Scheduling:**
  + **Tiến trình với ưu tiên cao nhất sẽ chạy trước.**
  + **Nếu có tiến trình cùng ưu tiên đến, thuật toán sử dụng round-robin với thời gian quantum = 10 đơn vị.**
  + **Nếu tiến trình có ưu tiên cao hơn đến trong khi tiến trình khác đang chạy, tiến trình hiện tại sẽ bị preempted (tạm dừng) và tiến trình có ưu tiên cao hơn sẽ được thực thi.**

**Sắp xếp tiến trình theo mức ưu tiên và thời gian đến:**

1. **P2: Ưu tiên = 3, thời gian đến = 0, thời gian burst = 20.**
2. **P1: Ưu tiên = 8, thời gian đến = 0, thời gian burst = 15.**
3. **P3: Ưu tiên = 4, thời gian đến = 20, thời gian burst = 20.**
4. **P4: Ưu tiên = 4, thời gian đến = 25, thời gian burst = 20.**
5. **P5: Ưu tiên = 5, thời gian đến = 45, thời gian burst = 5.**
6. **P6: Ưu tiên = 5, thời gian đến = 55, thời gian burst = 15.**

**Sơ đồ Gantt:**

1. **P2 chạy từ 0 đến 10 (10 đơn vị).**
2. **P1 chạy từ 10 đến 15 (5 đơn vị).**
3. **P3 chạy từ 20 đến 30 (10 đơn vị).**
4. **P4 chạy từ 30 đến 40 (10 đơn vị).**
5. **P3 tiếp tục chạy từ 40 đến 50 (10 đơn vị).**
6. **P4 tiếp tục chạy từ 50 đến 60 (10 đơn vị).**
7. **P5 chạy từ 60 đến 65 (5 đơn vị).**
8. **P6 chạy từ 65 đến 75 (10 đơn vị).**
9. **P6 tiếp tục chạy từ 75 đến 80 (5 đơn vị).**
10. **P1 tiếp tục chạy từ 80 đến 95 (15 đơn vị).**

**Thời gian hoàn thành (Turnaround Time):**

* **Công thức: Turnaround Time = Completion Time - Arrival Time.**

**Tính thời gian hoàn thành cho từng tiến trình:**

1. **P1: Completion Time = 95, Arrival Time = 0 → Turnaround Time = 95 - 0 = 95.**
2. **P2: Completion Time = 20, Arrival Time = 0 → Turnaround Time = 20 - 0 = 20.**
3. **P3: Completion Time = 50, Arrival Time = 20 → Turnaround Time = 50 - 20 = 30.**
4. **P4: Completion Time = 60, Arrival Time = 25 → Turnaround Time = 60 - 25 = 35.**
5. **P5: Completion Time = 65, Arrival Time = 45 → Turnaround Time = 65 - 45 = 20.**
6. **P6: Completion Time = 80, Arrival Time = 55 → Turnaround Time = 80 - 55 = 25.**

**Thời gian chờ (Waiting Time):**

* **Công thức: Waiting Time = Turnaround Time - Burst Time.**

**Tính thời gian chờ cho từng tiến trình:**

1. **P1: Turnaround Time = 95, Burst = 15 → Waiting Time = 95 - 15 = 80.**
2. **P2: Turnaround Time = 20, Burst = 20 → Waiting Time = 20 - 20 = 0.**
3. **P3: Turnaround Time = 30, Burst = 20 → Waiting Time = 30 - 20 = 10.**
4. **P4: Turnaround Time = 35, Burst = 20 → Waiting Time = 35 - 20 = 15.**
5. **P5: Turnaround Time = 20, Burst = 5 → Waiting Time = 20 - 5 = 15.**
6. **P6: Turnaround Time = 25, Burst = 15 → Waiting Time = 25 - 15 = 10.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Tóm tắt:**

* **Sơ đồ Gantt** cho thấy thứ tự các tiến trình theo thuật toán **preemptive, priority-based round-robin**.
* **Thời gian quay vòng** và **thời gian chờ** được tính toán cho từng tiến trình dựa trên kết quả sơ đồ Gantt.
* **Priority**: Số nhỏ hơn có ưu tiên cao hơn (3 > 4 > 5 > 8).
* **Time Quantum**: 10 đơn vị (cho Round-Robin khi cùng ưu tiên).
* **Quy tắc**: Ưu tiên cao nhất chạy trước; cùng ưu tiên thì dùng Round-Robin; nếu có tiến trình ưu tiên cao hơn đến, tiến trình hiện tại bị tạm dừng.

**Bước 1: Mô phỏng thuật toán**

* **0-20**: P2 (ưu tiên 3) chạy 20ms (0-10, 10-20). Hoàn thành tại 20.
* **20-30**: P3 (ưu tiên 4) chạy 10ms. Còn 10ms.
* **30-40**: P4 (ưu tiên 4) chạy 10ms (Round-Robin). Còn 10ms.
* **40-50**: P3 chạy tiếp 10ms. Hoàn thành tại 50.
* **50-60**: P4 chạy tiếp 10ms. Hoàn thành tại 60.
* **60-65**: P5 (ưu tiên 5) chạy 5ms. Hoàn thành tại 65.
* **65-80**: P6 (ưu tiên 5) chạy 15ms (65-75, 75-80). Hoàn thành tại 80.
* **80-95**: P1 (ưu tiên 8) chạy 15ms. Hoàn thành tại 95.

0-10: P2 | 10-20: P2 | 20-30: P3 | 30-40: P4 | 40-50: P3 | 50-60: P4 | 60-65: P5 | 65-75: P6 | 75-80: P6 | 80-95: P1

**Bước 2: Tính toán**

* **Thời gian hoàn thành**:
  + P1: 95
  + P2: 20
  + P3: 50
  + P4: 60
  + P5: 65
  + P6: 80
* **Thời gian quay vòng (TAT = Completion - Arrival)**:
  + P1: 95 - 0 = 95
  + P2: 20 - 0 = 20
  + P3: 50 - 20 = 30
  + P4: 60 - 25 = 35
  + P5: 65 - 45 = 20
  + P6: 80 - 55 = 25
* **Thời gian chờ (WT = TAT - Burst)**:
  + P1: 95 - 15 = 80
  + P2: 20 - 20 = 0
  + P3: 30 - 20 = 10
  + P4: 35 - 20 = 15
  + P5: 20 - 5 = 15
  + P6: 25 - 15 = 10

A screenshot of a puzzle

AI-generated content may be incorrect.

A screenshot of a puzzle

AI-generated content may be incorrect.

**Bài toán yêu cầu:**

* **a.** Một luồng để kiểm tra xem mỗi cột trong bảng Sudoku có chứa tất cả các chữ số từ 1 đến 9 không.
* **b.** Một luồng để kiểm tra xem mỗi hàng trong bảng Sudoku có chứa tất cả các chữ số từ 1 đến 9 không.
* **c.** Chín luồng để kiểm tra từng ô con 3x3 có chứa tất cả các chữ số từ 1 đến 9 không.
* **d.** Viết một phương thức main để kiểm tra ứng dụng đa luồng của bạn với bảng Sudoku mẫu trong hình ảnh.

**Cách triển khai chi tiết:**

**1. Cấu trúc dữ liệu bảng Sudoku**

Đầu tiên, tôi sẽ sử dụng một ma trận 2D để đại diện cho bảng Sudoku, nơi mỗi phần tử trong ma trận là một ô trong bảng.

**2. Các luồng cần thiết**

* **Luồng cho cột**: Một luồng kiểm tra tất cả các cột trong bảng Sudoku.
* **Luồng cho hàng**: Một luồng kiểm tra tất cả các hàng.
* **Luồng cho các ô con 3x3**: Chín luồng, mỗi luồng sẽ kiểm tra một ô con 3x3.

A paper with text on it

AI-generated content may be incorrect.

Bài toán này mô phỏng một hệ thống với ba **người tiêu dùng** (consumers) và một **đại lý** (agent). Mỗi người tiêu dùng cần một nguyên liệu khác nhau, và đại lý sẽ cung cấp nguyên liệu cho mỗi người tiêu dùng khi họ yêu cầu. Hệ thống này có thể được mô phỏng bằng **đa luồng**, với mỗi người tiêu dùng và đại lý là các **luồng (threads)** riêng biệt.

**Giải thích chi tiết bài toán:**

* **Người tiêu dùng (C1, C2, C3)**: Mỗi người tiêu dùng yêu cầu một nguyên liệu khác nhau (C1 yêu cầu A, C2 yêu cầu B, C3 yêu cầu C). Các tiêu dùng này sẽ **chờ đợi thông báo** từ đại lý khi nguyên liệu của mình đã sẵn sàng.
* **Đại lý (Agent)**: Đại lý có một nguồn cung vô hạn các nguyên liệu A, B và C. Đại lý sẽ **cung cấp ngẫu nhiên một nguyên liệu** cho người tiêu dùng khi họ yêu cầu và **thông báo** cho người tiêu dùng.
* **Chia sẻ tài nguyên**: Đại lý và các người tiêu dùng chia sẻ một **bàn (table)**, nơi đại lý sẽ đặt nguyên liệu và người tiêu dùng lấy nguyên liệu đó từ bàn.

Bài toán này liên quan đến việc sử dụng **synchronization** trong đa luồng để **đảm bảo việc chia sẻ tài nguyên** một cách đồng bộ mà không gặp phải tình trạng **race condition**.

**Bài toán này tương tự với vấn đề trong các bài tập về Producer-Consumer, thường được giải quyết bằng semaphores hoặc mutexes để đồng bộ các luồng và tránh xung đột khi truy cập tài nguyên chung.**

**Các yêu cầu trong đề bài:**

* **a.** Các phương thức được gọi bởi người tiêu dùng (Consumers).
* **b.** Các phương thức được gọi bởi đại lý (Agent).
* **c.** Viết một phương thức main để kiểm tra chương trình.

**Mô phỏng hệ thống theo yêu cầu đề bài:**

1. **Các phương thức của người tiêu dùng**:
   * **wait()**: Người tiêu dùng sẽ chờ đợi thông báo từ đại lý.
   * **consume()**: Sau khi nhận được thông báo, người tiêu dùng sẽ lấy nguyên liệu từ bàn và tiêu thụ trong một khoảng thời gian.
2. **Các phương thức của đại lý**:
   * **generateIngredient()**: Đại lý sẽ ngẫu nhiên tạo ra một nguyên liệu và đặt vào bàn.
   * **notifyConsumer()**: Sau khi nguyên liệu được đặt lên bàn, đại lý sẽ thông báo cho người tiêu dùng.
3. **Các phương thức trong main**:
   * **Khởi tạo các luồng**: Tạo ba luồng cho ba người tiêu dùng và một luồng cho đại lý.
   * **Khởi chạy và đồng bộ**: Các luồng sẽ được đồng bộ hóa và chạy song song

**Giải thích chi tiết mã:**

1. **Table (Bàn chia sẻ)**:
   * Lớp Table đại diện cho một bàn mà đại lý và người tiêu dùng chia sẻ. Các nguyên liệu sẽ được đặt lên bàn, và người tiêu dùng sẽ lấy chúng đi khi được thông báo.
2. **Consumer (Người tiêu dùng)**:
   * Mỗi người tiêu dùng (C1, C2, C3) sẽ cần một nguyên liệu cụ thể (A, B, C).
   * Người tiêu dùng sẽ **chờ thông báo** từ đại lý và lấy nguyên liệu khi nó sẵn sàng.
   * Sau khi tiêu thụ nguyên liệu, người tiêu dùng sẽ **thông báo cho đại lý** rằng bàn đã sẵn sàng để nhận nguyên liệu mới.
3. **Agent (Đại lý)**:
   * Đại lý sẽ **ngẫu nhiên tạo ra một nguyên liệu** và đặt nó lên bàn.
   * Sau khi đặt nguyên liệu, đại lý sẽ **thông báo tất cả các người tiêu dùng** để họ có thể lấy nguyên liệu.
4. **Main method**:
   * Tạo ba luồng cho ba người tiêu dùng và một luồng cho đại lý.
   * Các luồng sẽ chạy song song, với người tiêu dùng chờ đợi và lấy nguyên liệu, còn đại lý cung cấp nguyên liệu ngẫu nhiên.

**Kết luận:**

Bài toán này mô phỏng hệ thống có ba người tiêu dùng và một đại lý sử dụng **đa luồng (multithreading)** với việc đồng bộ hóa thông qua **synchronization**. Mỗi người tiêu dùng và đại lý là các luồng riêng biệt, và chúng tương tác với nhau thông qua việc chia sẻ một tài nguyên (bàn) trong hệ thống. ​