Đầu tiên, tui có biến tick\_time

vì trong hàm update có tick\_time++. ++ tới bao giờ? nhỡ nó tràn thì sao? thế nên tui sẽ tìm cách để giảm **tick\_time** vào 1 thời điểm nào đó. tui quyết định mỗi khi có 1 hay nhiều task, tại 1 thời điểm, được schedule, thì trừ cái **tick\_time** xuống 0, sử dụng **min\_delay.**

trong task array, ta sẽ luôn có những task cần thực thi.

**Add\_task**: Giả sử, đầu tiên ông thêm vào task 400ms. min\_delay = 400

**Check\_Ready\_Task:** tick\_time sẽ tăng dần. nếu tick\_time == min\_delay, nó sẽ:

Dò hết mảng task, đi trừ delay của từng task với min\_delay.

Nếu delay mới = 0, cho ready++.

Nếu task đó có period, thì gán period = delay

sau khi quét hết mảng, đưa tick\_time về 0 để nó đếm lên lại tới 400ms.

**Add\_task**: Giả sử, đầu tiên ông có task A 400ms với min\_delay là 400ms. ở thời điểm tick\_time = 100ms, bỗng có 1 task B thêm vào thời delay là 10ms. rõ là ông sẽ muốn task B chạy trước. Nên mình gán delay của task B khi thêm vào array là 110 + 10 = 110ms. Đồng thời vì 110ms < min\_delay hiện tại, min\_delay mới thành 110ms. Và khi tick\_time = 110, nó sẽ schedule task B.

**Check\_Ready\_Task:**

Quét hết mảng task, vì khi này tick\_time = min\_delay = 110. ta có taskB delay = 110 - 110 = 0. taskA delay = 400 – 110 = 290. taskB.ready++, còn taskA thì không.

Trong lúc quét mảng task, ta sẽ đồng thời tìm kiếm min\_delay mới trong mảng task (không tính các delay = 0). Giả sử taskB chỉ chạy 1 lần nên bỏ qua, nên lúc này ta sẽ chọn min\_delay mới là 290 của taskA.

tick\_time reset lại về 0. nó sẽ đếm tiếp 290ms nữa để schedule task A

**Trong trường hợp tồi tệ**

Trong hàm main của tui, tui gọi Check\_Ready\_task trước khi dispact. nghĩa là Check\_Ready\_Task sẽ kiểm tra xem task nào được thực thi, và đồng thời tính 1 cái min\_delay mới. Nhưng, giả sử khi vào hàm dispact, 1 task chạy quá lâu đến nỗi tick\_time vượt quá min\_delay:

Ông có các task: 100ms, 110ms, 120ms. tick\_time chạy tới 100ms, nó bằng min\_delay (100ms) nên schedule cho task 0 chạy và delay mới sẽ là 0, 10, 20, min\_delay mới là 10, tick\_time reset về 0. Giả sử task 0 chạy tới tận 100ms mới xong, khi đó tick\_time đã là 100 rồi, nó không = min\_delay nữa, và có tận 2 task cần được schedule

để tiếp tục schedule 2 task phía sau, trong hàm Check\_Ready\_Task, mình vẫn lấy delay – tick\_time. Nếu delay mới < 0, nghĩa là nó đáng lẽ đã được schedule khi mà task 0 đang chạy, nên mình cho ready++. Nhưng nhỡ task 1 có period = 10ms? nghĩa là nó đáng lẽ phải được schedule lúc 10ms, 20ms, 30ms, 40ms….100ms. vậy thì để đảm bảo schedule đủ, nếu 1 task có period, mà delay hiện tại là âm, mình cứ liên tục lấy delay + period đồng thời tăng ready, cho tới khi nào delay dương thì thôi.