- Sắp xếp công việc:

- Cho n công việc phải hoàn thành , mõi công việc thực hiện trong một đơn vị thời gian. Việc i đem lại Wi tiền thưởng nếu hoàn thành đúng hạn di.

- Tìm cách thực hiện các công việc để có lợi nhuận cao nhất mà thời gian thực hiện là ít nhất.

8

di wi

4 20

2 40

4 50

6 40

3 90

7 60

9 10

2 30

- Có tất cả n bộ { i, di , Wi ) mình phải tìm ra lịch thực hiện công việc sao cho T là lớn nhất ( T là tổng số tiền được thưởng ).

- Ý tưởng tham lam : Sắp xếp lại lịch theo chiều giảm dần của phần thưởng Wi . Thay đổi i và di tương ứng.

Dòng mã bạn đã đề cập là một phần của hàm **maxProfit**, nơi chúng ta đang thực hiện việc tìm một thời điểm trống để thực hiện công việc và tính tổng số tiền thưởng.

Dòng **for (int i = 0; i < jobs.size(); i++)** duyệt qua từng công việc trong danh sách **jobs**.

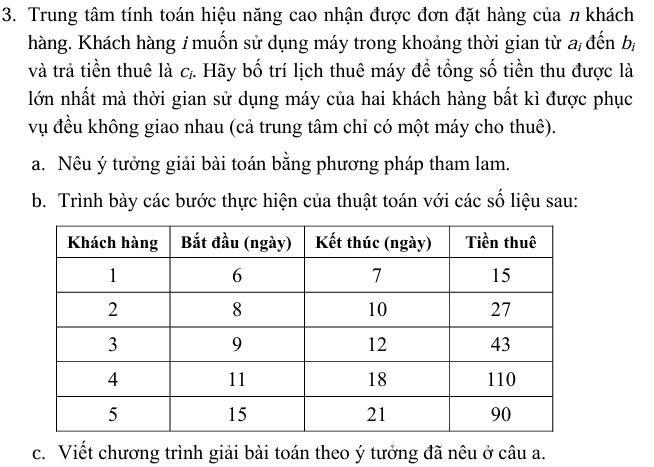
Trong vòng lặp này, chúng ta sẽ tìm một thời điểm trống để thực hiện công việc. Việc này được thực hiện bằng cách duyệt qua các thời điểm từ thời điểm gần nhất đến thời điểm cuối cùng mà công việc đó phải hoàn thành, hoặc đến cuối danh sách nếu thời điểm cuối cùng lớn hơn kích thước của danh sách công việc.

Dòng **for (int j = min(jobs[i].deadline, (int)jobs.size()) - 1; j >= 0; j--)** tạo một vòng lặp để duyệt qua các thời điểm từ thời điểm gần nhất đến thời điểm cuối cùng mà công việc đó phải hoàn thành hoặc đến cuối danh sách, tùy thuộc vào điều kiện nào nhỏ hơn. Trong đó:

* **jobs[i].deadline** là thời điểm cuối cùng mà công việc thứ **i** phải hoàn thành.
* **(int)jobs.size()** là kích thước của danh sách công việc, đảm bảo rằng ta không vượt qua phạm vi của danh sách.
* **min(jobs[i].deadline, (int)jobs.size()) - 1** đảm bảo rằng ta chỉ duyệt qua các thời điểm hợp lệ, không vượt quá kích thước của danh sách.

Trong vòng lặp con này, chúng ta kiểm tra xem thời điểm **j** có trống không bằng cách sử dụng điều kiện **if (slot[j] == false)**. Nếu thời điểm đó trống (**slot[j]** có giá trị **false**), chúng ta thực hiện công việc tại thời điểm đó bằng cách đặt **slot[j] = true**, đồng thời cộng thêm phần thưởng của công việc vào tổng số tiền thưởng **totalProfit**. Sau đó, chúng ta thoát khỏi vòng lặp bằng lệnh **break**, vì đã tìm thấy thời điểm trống để thực hiện công việc.

Khi một công việc không thể thực hiện đúng hạn do không có thời điểm trống, chúng ta sẽ không thực hiện gì cả và tiếp tục với công việc tiếp theo trong danh sách.



Trung tâm tính toán hiệu năng cao nhận được đơn đặt hàng của n khách hàng

- Khách hàng i muốn sử dụng máy trong khoảng thời gian từ ai đến bi và trả tiền thuê là ci . Hãy bố trí lịch thuê máy để tổng số tiền thu được là lớn nhất mà thời giân sử dụng máy của hai khách hàng bất kì được phục vụ đều không giao nhau.

- Số liệu :

Khách hàng 1 : bắt đầu : 6 -> kết thúc 7 : tiền thuê 15

Khách hàng 2 : bắt đầu : 8 -> kết thúc 10 : tiền thuê 27

Khách hàng 3 : bắt đầu : 9 -> kết thúc 12 : tiền thuê 43

Khách hàng 4 : bắt đầu : 11 -> kết thúc 18 : tiền thuê 110

Khách hàng 5 : bắt đầu : 15 -> kết thúc 21 : tiền thuê 90