

# Lời giải chi tiết bài tập nhóm 6

## Phân tích và thiết kế thuật toán

Nguyễn Duy Khang - 24520755

Hà Bùi Trọng Nghĩa - 24520020

## A. Fucs và gánh nặng mưu sinh

### Tóm tắt bài toán

Server của Fucs nhận được một hàng đợi gồm  $N$  tác vụ cần xử lý. Mỗi tác vụ  $i$  được mô tả bởi ba tham số:

- $r_i$ : thời gian chạy của tác vụ,
- $t_i$ : hạn hoàn thành (deadline),
- $p_i$ : lợi nhuận thu được nếu tác vụ hoàn thành không trễ.

CPU chỉ có thể xử lý tuần tự từng tác vụ, và một khi đã bắt đầu thực hiện một tác vụ thì phải chạy liên tục cho đến khi hoàn thành (nguyên tắc *atomic*). Ta có quyền chọn hoặc không chọn mỗi tác vụ. Một tác vụ chỉ mang lại lợi nhuận nếu nó hoàn thành trước hoặc đúng thời điểm  $t_i$ .

**Mục tiêu:** chọn và sắp xếp thứ tự thực hiện các tác vụ sao cho tổng lợi nhuận thu được là lớn nhất.

---

### 1. Phương pháp thiết kế thuật toán

#### Subtask 1 – Brute Force (Exact Algorithm)

Đối với subtask 1, giới hạn  $N \leq 20$  cho phép duyệt toàn bộ không gian nghiệm.

**Ý tưởng** Mỗi tác vụ có hai khả năng: được chọn hoặc không được chọn. Do đó, ta có thể duyệt qua tất cả các tập con của  $N$  tác vụ.

## Thuật toán

1. Sinh tất cả các tập con của tập tác vụ ban đầu.
2. Với mỗi tập con:
  - Sắp xếp các tác vụ theo deadline tăng dần.
  - Mô phỏng quá trình thực hiện tuần tự, cộng dồn thời gian chạy.
  - Nếu tại bất kỳ thời điểm nào tổng thời gian vượt quá deadline của tác vụ hiện tại thì tập con này không hợp lệ.
  - Nếu hợp lệ, tính tổng lợi nhuận và cập nhật kết quả tốt nhất.

**Nhận xét** Phương pháp này đảm bảo tìm ra nghiệm tối ưu tuyệt đối do duyệt đầy đủ mọi khả năng.

---

## Subtask 2 – Heuristic (Beam Search)

Đối với subtask 2,  $N$  có thể lên tới 500 và các giá trị  $r_i, t_i$  rất lớn (lên tới  $10^{12}$ ), do đó các thuật toán chính xác như brute force hoặc quy hoạch động truyền thống là không khả thi.

**Ý tưởng** Ta sử dụng **Beam Search**, một dạng heuristic có kiểm soát, để tìm nghiệm gần tối ưu trong thời gian cho phép.

## Nguyên lý

- Các tác vụ được sắp xếp theo deadline tăng dần.
- Mỗi trạng thái biểu diễn một phương án thực hiện các tác vụ đã xét, bao gồm:
  - tổng thời gian đã sử dụng,
  - tổng lợi nhuận đạt được.
- Tại mỗi bước, ta mở rộng các trạng thái hiện có bằng cách:
  - không chọn tác vụ hiện tại,
  - hoặc chọn tác vụ nếu không vi phạm deadline.

**Cắt tỉa trạng thái** Để kiểm soát số lượng trạng thái:

- Áp dụng **Pareto dominance pruning**: nếu tồn tại hai trạng thái mà một trạng thái vừa tồn nhiều thời gian hơn vừa có lợi nhuận không cao hơn, thì trạng thái đó bị loại bỏ.
- Giữ lại tối đa một số lượng trạng thái tốt nhất (beam width cố định).

### Lý do chọn Beam Search

- Phù hợp với bài toán có không gian nghiệm rất lớn.
- Dễ kết hợp với các chiến lược cắt tỉa.
- Cho nghiệm chất lượng cao và ổn định, phù hợp với hình thức chấm điểm gần đúng (scoring).

---

## 2. Tính phù hợp của phương pháp

### Subtask 1

- Giới hạn nhỏ ( $N \leq 20$ ).
- Brute force đảm bảo tính đúng tuyệt đối.
- Không yêu cầu heuristic hay xấp xỉ.

Do đó, phương pháp brute force là hoàn toàn phù hợp cho subtask 1.

---

### Subtask 2

- Không yêu cầu nghiệm tối ưu tuyệt đối.
- Dữ liệu lớn khiến các thuật toán chính xác trở nên không khả thi.
- Beam Search cho phép cân bằng giữa thời gian chạy và chất lượng nghiệm.

Việc sử dụng heuristic là phù hợp và hiệu quả trong bối cảnh bài toán được chấm điểm theo mức độ gần đúng so với lời giải chuẩn.

---

### 3. Phân tích độ phức tạp

#### Subtask 1

- Thời gian:

$$O(2^N \cdot N \log N)$$

- Không gian:

$$O(N)$$

—

#### Subtask 2

Gọi  $K$  là độ rộng beam.

- Thời gian:

$$O(N \cdot K)$$

- Không gian:

$$O(K)$$

Trong thực tế, bước cắt tỉa dominance giúp giảm đáng kể số trạng thái cần xử lý.

—

### Kết luận

Bài toán là một bài toán lập lịch tác vụ có deadline và lợi nhuận, thuộc lớp bài toán NP-hard. Subtask 1 được giải bằng thuật toán chính xác để đảm bảo tính đúng tuyệt đối. Subtask 2 sử dụng heuristic nhằm tìm nghiệm chất lượng cao trong giới hạn thời gian, phù hợp với hình thức chấm điểm gần đúng của bài toán.