Báo Cáo CS112

Bài A – Fucs và gánh nặng mưu sinh

1. Phương pháp thiết kế thuật toán (40%)

Subtask 1 – Brute Force

Với ràng buộc $N \leq 20$, ta áp dụng phương pháp **Brute Force** để tìm nghiệm tối ưu tuyệt đối.

Ý tưởng chính:

- Duyệt tất cả các tập con tác vụ (mỗi tập con được biểu diễn bằng bitmask).
- Với mỗi tập con, xét mọi thứ tự thực hiện có thể.
- Mô phỏng quá trình chạy tuần tự các tác vụ theo thứ tự đã chọn.
- Một tác vụ chỉ được tính lợi nhuận nếu hoàn thành không muộn hơn deadline.
- Lấy giá trị tổng lợi nhuận lớn nhất trong tất cả các khả năng.

Do số lượng tác vụ nhỏ, phương pháp brute force đảm bảo duyệt hết không gian nghiệm và luôn tìm được lời giải tối ưu.

Subtask 2 – Heuristic

Với $N \leq 500$, việc áp dụng brute force là không khả thi. Vì vậy, ta sử dụng **phương pháp heuristic**, cụ thể là **Beam Search kết hợp Dynamic Programming theo deadline**.

Nguyên lý hoạt động:

- Sắp xếp các tác vụ theo deadline tăng dần.
- Xây dựng dần các trạng thái $(time, profit)$, trong đó:
  - $time$: tổng thời gian đã sử dụng
  - $profit$: tổng lợi nhuận đạt được
- Tại mỗi tác vụ, từ mỗi trạng thái sinh ra hai lựa chọn:
  - Không chọn tác vụ hiện tại
  - Chọn tác vụ hiện tại nếu vẫn kịp deadline
- Áp dụng **Pareto pruning** để loại bỏ các trạng thái kém hiệu quả.
- Giới hạn số trạng thái được giữ lại bằng **Beam Width**.

Thuật toán heuristic không đảm bảo tối ưu tuyệt đối nhưng cho nghiệm có chất lượng rất cao trong thời gian cho phép.

—

2. Tính phù hợp của phương pháp (40%)

Subtask 1

Với $N \leq 20$, tổng số khả năng là hữu hạn và không quá lớn. Phương pháp brute force phù hợp vì:

- Duyệt đầy đủ không gian nghiệm
- Không bỏ sót phương án tối ưu
- Dễ cài đặt và kiểm chứng tính đúng đắn

Do đó, brute force hoàn toàn đáp ứng yêu cầu tìm nghiệm tối ưu tuyệt đối của subtask 1.

Subtask 2

Subtask 2 yêu cầu tìm nghiệm gần đúng tốt nhất trong thời gian giới hạn. Việc sử dụng **heuristic Beam Search** là phù hợp vì:

- Không gian nghiệm rất lớn, không thể duyệt toàn bộ.
- Beam Search cho phép giữ lại các trạng thái tiềm năng nhất.
- Có thể điều chỉnh chất lượng nghiệm thông qua tham số Beam Width.

Cấu hình sử dụng:

- Beam Width = 15000
- Áp dụng Pareto pruning theo $(time, profit)$

Cấu hình này giúp cân bằng giữa thời gian chạy và chất lượng nghiệm, phù hợp với cơ chế chấm điểm của bài toán.

—

3. Phân tích độ phức tạp (20%)

Subtask 1

- Thời gian: $O(N! \cdot N)$ trong trường hợp duyệt mọi hoán vị
- Không gian: $O(2^N)$

Với $N \leq 20$, chi phí này là chấp nhận được.

Subtask 2

Gọi B là Beam Width.

- Mỗi tác vụ tạo ra tối đa $2B$ trạng thái
- Việc sắp xếp và pruning tốn $O(B \log B)$ mỗi bước

Tổng độ phức tạp:

$$O(N \cdot B \log B)$$

Bộ nhớ sử dụng:

$$O(N \cdot B)$$

Với $N = 500$, $B = 15000$, thuật toán chạy hiệu quả trong giới hạn thời gian.

—

Kết luận

Bài toán được giải bằng hai hướng tiếp cận:

- Subtask 1 sử dụng brute force để đảm bảo nghiệm tối ưu tuyệt đối.
- Subtask 2 sử dụng heuristic Beam Search để tìm nghiệm gần tối ưu với hiệu năng cao.

Cách tiếp cận này phù hợp với yêu cầu của từng subtask và cơ chế chấm điểm của bài toán.