ĐẠI HỌC QUỐC GIA
THÀNH PHỐ HỒ CHÍ MINH

ĐÁP ÁN CHÍNH THỨC

KỲ THI TUYỂN SINH LỚP 10 NĂM 2025
TRƯỜNG PHỔ THÔNG NĂNG KHIẾU
Môn thi: TIN HỌC
Thời gian làm bài: 150 phút (không kể thời gian phát đề)

Bài 1: CHUỖI ONLINE BẠN BÈ

Ý tưởng cơ bản:

- Bài toán yêu cầu tìm độ dài chuỗi "ONLINE" liên tục dài nhất trong T phút theo dõi
- Đây là bài toán duyệt tuyến tính đơn giản với việc đếm chuỗi con liên tục

1. Duyệt qua mảng trạng thái từ đầu đến cuối
2. Duy trì hai biến: currentStreak (chuỗi hiện tại) và maxStreak (chuỗi dài nhất)
3. Khi gặp "ONLINE": tăng currentStreak, cập nhật maxStreak
4. Khi gặp trạng thái khác: reset currentStreak = 0

Trường hợp đặc biệt:

- Nếu không có phút nào online: kết quả = 0
- Nếu tất cả phút đều online: kết quả = T

Độ phức tạp: O(T)

Bài 2: HÀNH TRÌNH XE ĐIỆN

Ý tưởng:

Bài toán áp dụng thuật toán tham lam.

1. Sắp xếp các trạm sạc theo thứ tự tăng dần.
2. Từ vị trí hiện tại với pin hiện có, luôn đi đến trạm sạc xa nhất có thể.
3. Nếu không thể đến trạm nào nữa: kiểm tra xem có thể đến đích trực tiếp không. Nếu không, kết quả là -1.
4. Nếu đến được một trạm: di chuyển đến trạm đó, sạc đầy pin, tăng số lần sạc, và lặp lại từ bước 2.
5. Nếu có thể đến đích trực tiếp từ vị trí hiện tại: dừng thuật toán.

Trường hợp đặc biệt:

- Pin ban đầy đủ di thẳng đến đích: số lần sạc = 0
- Không thể đến trạm sạc nào: trả về -1

Độ phức tạp: O(N log N)

Bài 3: TRÒ CHƠI XẾP CHỮ

- Bài toán tìm số ký tự ít nhất cần xóa để tạo palindrome.
- Equivalent: Tìm chuỗi con palindrome dài nhất (Longest Palindromic Subsequence - LPS).
- Gọi f[i][j] là độ dài LPS của chuỗi con từ vị trí i đến j.
- Công thức quy hoạch động:
  - Nếu s[i] = s[j]: f[i][j] = f[i+1][j-1] + 2
  - Nếu s[i] ≠ s[j]: f[i][j] = max(f[i+1][j], f[i][j-1])
- Nghiệm ban đầu: f[i][i] = 1
- Kết quả: n - f[0][n-1] (độ dài chuỗi trừ độ dài LPS)

Độ phức tạp: $O(n^2)$

Bài 4: TỐI ƯU GIAO DỊCH BLOCKCHAIN

Ý tưởng:

- Bài toán được cho là dạng bài toán Knapsack có thêm các điều kiện ràng buộc phụ thuộc. Trong yêu cầu đề bài, có nêu lên các điều kiện phụ thuộc có dạng: "Nếu chọn giao dịch A thì phải chọn giao dịch B."
- **Dữ liệu đầu vào:** gồm có N giao dịch, mỗi giao dịch chuỗi khối có chi phí và kích thước cho trước, cùng với **D** ràng buộc.
- **Yêu cầu đặt ra:** tổng kích thước các giao dịch được chọn phải nhỏ hơn hoặc bằng S_{max}

Phương pháp giải quyết:

- Bài toán được đưa ra là bài toán Knapsack có điều kiện ràng buộc.
- Bài toán Knapsack là bài toán có độ phức tạp NP-Hard, nhưng nhờ các điều kiện phụ thuộc được cung cấp đầu vào, ta có thể tạo ra một đồ thị có hướng không chu trình giữa các giao dịch được chọn.

- Chính vì các điều kiện phụ thuộc này, dù bài toán có độ phức tạp cao ban đầu, ta có thể chuyển đổi thành một bài toán Knapsack 0-1 với các nhóm ràng buộc, nên vẫn có thể giải được bằng quy hoạch động (DP) khi S_{max} không quá lớn.
- Đây là hướng tiếp cận phổ biến cho việc giải quyết các bài toán tối ưu. Nhờ các điều kiện ràng buộc đầu vào, không gian tìm kiếm lời giải sẽ được thu hẹp đi rất nhiều. Từ đó, chúng ta có thể giải tiếp bài toán trên không gian tìm kiếm đã được thu gọn.
- Liên quan đến bài toán, có hai hướng tiếp cận như sau:

Cách 1: Bitmask (cho N ≤ 25)

1. Sử dụng Bitmask để duyệt tất cả tập con có thể (2^N tổ hợp)
2. Với mỗi tập con, kiểm tra:
   - Tổng kích thước ≤ S_{max}
   - Thỏa mãn tất cả ràng buộc phụ thuộc
3. Trả về tập con có tổng chi phí lớn nhất

Cách 2: Sử dụng quy hoạch động (Dynamic Programming - DP) với xử lý ràng buộc (cho N ≤ 50)

1. Xây dựng đồ thị phụ thuộc từ các ràng buộc
2. Tạo các "super transactions": Với mỗi giao dịch i, tính tổng kích thước và chi phí của giao dịch i cùng tất cả ràng buộc bắt buộc liên quan.
3. Áp dụng phương pháp quy hoạch động cho bài toán Knapsack trên các "super transactions" vừa tạo ra.
4. Loại bỏ trùng lặp khi một giao dịch xuất hiện trong nhiều "super transaction" để xuất ra kết quả.

Tối ưu hóa cho test lớn:

- Sử dụng quy hoạch động với nén trạng thái (state compression).
- Sắp xếp Topo (Topological sort) để xử lý ràng buộc hiệu quả hơn.

Trường hợp đặc biệt:

- Không có giao dịch nào có kích thước ≤ S_{max}: kết quả = 0.
- Không có ràng buộc (D = 0): bài toán Knapsack chuẩn.
- Có chu trình trong các ràng buộc: không xảy ra theo đề bài.

Độ phức tạp:

- **Cách 1 (Bitmask):** $O(2^N \times N)$ với $N \leq 25$.
- **Cách 2 (DP):** $O(N \times S_{max} + N \times D)$ với $N \leq 50$, S_max ≤ 1000.

---------HẾT----------