**HƯỚNG DẪN GIẢI**

**ĐỀ THI CẤP CHỨNG CHỈ THUẬT TOÁN ỨNG DỤNG SAMSUNG NĂM 2025**

*(Hướng dẫn giải gồm có 03 trang)*

**BÀI A. MA TRẬN 3x3**

Gọi ma trận 3x3 như sau:

|  |  |  |
| --- | --- | --- |
| X11 | X12 | X13 |
| X21 | X22 | X23 |
| X31 | X32 | X33 |

**Subtask 1:**

* Dùng 6 vòng FOR để thử chọn 6 vị trí, 3 vị trí còn lại tìm được dựa theo tổng A, B, C hoặc D, E, F.
* Chẳng hạn, có thể chọn 6 vị trí ở 2 cột đầu là X11, X12, X21, X22, X31 và X32:

+) Tất nhiên, X11 phải < Tổng Hàng 1, nên ta for X11 từ 1 đến A – 1; X12 cũng từ 1 đến A – 1; X21 và X22 từ 1 đến B – 1; X31 và X32 từ 1 đến C - 1

+) Khi đó: X13 = A – X11 – X12; X23 = B – X21 – X22; X33 = C – X31 – X32.

* Lúc này, đương nhiên điều kiện về tổng mỗi hàng là thoả mãn. Bây giờ, kiểm tra xem:

+) X13, X23 và X33 có dương không

+) Tổng cột 1, 2, 3 có lần lượt bằng D, E, F hay không

* Có **tối đa** 96 = 531.441 trường hợp cần duyệt

**Subtask 2:**

* Dùng 4 vòng for cho X11, X12, X21, X22. 5 số còn lại tính theo A, B, D, E và C
* Chẳng hạn:

X13 = A – X11 – X12

X23 = B – X21 – X22

X31 = D – X11 – X21

X32 = E – X12 – X22

X33 = C – X31 – X32 = F – X13 – X23

* Lúc này, đương nhiên tổng hàng 1 là A, hàng 2 là B, cột 1 là D, cột 2 là E. Bây giờ, kiểm tra xem:

+) Các số X13, X23, X31, X32 và X33 có dương không

+) Số X33 được tính theo 2 cách có cho ra kết quả như nhau hay không

* Có **tối đa** 304 = 707.281 (Khoảng > 7.105) trường hợp cần duyệt

**BÀI B. XÂU KHÔNG CÂN BẰNG**

**Subtask 1:**

* Có thể làm bằng sinh kế tiếp hoặc quay lui. Với giới hạn không quá 20 thì đây là bài toán duyệt tất cả các xâu nhị phân cơ bản

**Subtask 2: Quy hoạch động trạng thái.**

* Khi đi từ trái sang phải, để biết việc thêm kí tự mới có tạo ra một palindrome độ dài K hay không, chỉ cần biết K - 1 kí tự cuối trước đó.
* Gọi trạng thái là bitmask **state** biểu diễn K−1 kí tự cuối (A = 0, B = 1).
* Gọi dp[i][state] là số lượng xâu thỏa mãn khi duyệt tới kí tự thứ i và có K - 1 kí tự cuối bằng state.
* Với mỗi vị trí i, từ dp[i - 1][state] cập nhật ra dp[i][new\_state] theo các lựa chọn (với S[i] là ‘A’, ‘B’ hay ‘?’).
* Khi thêm bit b mới (0 hoặc 1), trạng thái mới là new\_state = ((state<<1) & mask) | b.
* Nếu đã đặt ≥ K kí tự, kiểm tra cửa sổ K bit cuối window = (state<<1) | b có đối xứng không, nếu đối xứng thì loại.
* Độ phức tạp: O(N.2K) với K ≤ 10.

**BÀI C. DI CHUYỂN TRONG MÊ CUNG – 1**

* Đây là bài toán BFS cơ bản trên đồ thị vô hướng không có trọng số

**BÀI D. TRUY VẤN VỚI DÃY NGOẶC ĐÚNG**

**Subtask 1:**

* Đề bảo gì làm vậy
* Với mỗi truy vấn loại 2, kiểm tra dãy ngoặc đúng bằng stack:
* Gặp mở thì push vào
* Gặp đóng không có mở thì sai, đóng có mở thì pop
* Cuối cùng, stack không rỗng tức là còn dấu mở đang chờ thì sai
* Độ phức tạp: O(Q.N)

**Subtask 2:**

* Dùng cây IT lưu lại giá trị tích lũy của sum và min\_prefix.
* Nếu s[i] là (, sum += 1, min\_prefix += 1
* Nếu s[i] là ), sum -= 1, min\_prefix -= 1
* Điều kiện để xâu con [L … R] là dãy ngoặc đúng:
* sum[L … R] = 0
* min\_prefix[L … R] ≥ min\_prefix[L]
* Đây chính là hai bài toán kinh điển về cây IT, bài toán tính tổng và bài toán tìm giá trị nhỏ nhất trên một đoạn.
* Độ phức tạp: O((N + Q)logN)

**BÀI E. THAO TÁC XOÁ XÂU**

**Subtask 1:**

* Đề bảo gì làm thế, gọi f(t) là số bước ít nhất để xoá xâu t thành rỗng
* Dùng map <string, int> để memo
* Với mỗi i, tìm j xa nhất mà đoạn [i i + 1 … j] còn giống nhau. Đây là đoạn cần xoá
* Thực hiện xoá đoạn [i .. j] của xâu s, gọi là xâu t
* Đệ quy tới f(t). Cập nhật kết quả hiện tại so với 1 + f(t).
* Gán lại i mới = j + 1

**Subtask 2: Quy hoạch động**

* Gọi dp[l][r] là số thao tác ít nhất để xóa xâu con s[l...r].
* Trường hợp cơ sở:
* dp[i][i] = 1 (Xóa một ký tự tốn 1 thao tác).
* dp[i][j] = 0 nếu i > j (Xâu rỗng, không tốn thao tác).
* Công thức truy hồi:

1. Lựa chọn 1: Xóa s[l] một mình. Chi phí là 1 + dp[l + 1][r].

(Tốn 1 thao tác cho s[l], cộng với chi phí xóa phần còn lại s[l+1...r]).

1. Lựa chọn 2: Xóa s[l] cùng với một s[k] khác (trong đó s[l] == s[k] và l < k <= r).

Ý tưởng: s[l] và s[k] (cùng các ký tự s[l] ở giữa nếu có) được xóa trong cùng 1 lần

* Trước tiên, xóa phần ở giữa: s[l + 1 ... k – 1]. Chi phí: dp[l+1][k-1].
* Sau đó, s[l] và s[k] (giờ đã liền kề hoặc s[l] là phần đầu của khối bắt đầu bằng s[k]) được xóa chung. Chi phí xóa phần còn lại từ s[k] là dp[k][r]. Vì s[l] được xóa “ké” vào thao tác xóa s[k], ta không tốn thêm thao tác cho s[l] ở bước này.
* Tổng chi phí cho lựa chọn này với một k cụ thể: dp[l + 1][k – 1] + dp[k][r]. Ta chọn k tốt nhất.
  + Kết quả cuối cùng: dp[0][n – 1] (nếu xâu đánh chỉ số từ 0).

**BÀI F. DI CHUYỂN TRONG MÊ CUNG – 2**

* + Có 2 loại thao tác:
* Loại 1: Di chuyển tới 4 ô chung cạnh mất 1 bước
* Loại 2: Với mỗi ô (i, j), di chuyển tới 1 trong 24 ô bất kỳ của hình vuông 5x5 do (i, j) làm tâm mất 2 bước
  + Đây là bài toán di chuyển trên đồ thị có trọng số, vì vậy thuật toán chuẩn sẽ là quy về bài toán tìm đường đi ngắn nhất sử dụng Dijkstra.

**HẾT**