LASTZERO

Ta có nhận xét số chữ số 0 tận cùng bên phải của một số bằng min(pw2, pw5) trong đó pw2 và pw5 là số lượng thừa số 2 và 5 trong dạng phân tích thành nhân tử của số đó. Vậy trước tiên, ta sẽ tính $pw2_i$ và $pw5_i$ của mỗi a_i.

Sub 1: Quay lui sinh 2^k dãy nhị phân rồi duyệt bình thường.

Sub2: Ta sẽ sử dụng thuật toán quy hoạch động như sau:

Hàm đệ quy có nhớ solve(i, j, l) trong đó i kiểm soát các phần tử của dãy a, j kiểm soát số lượng phần tử đã chọn và l kiểm soát pw5 của tích các phần tử đã chọn. Hàm sẽ trả về pw2 lớn nhất có thể của tích các phần tử được chọn.

Công thức:

solve(i, j, l) = max(solve(i, j, l), solve(i + 1, j, l)) (Không chọn a_i)

solve(i, j, l) = max(solve(i, j, l), solve(i + 1, j + 1, l + $pw5_i$) + $pw2_i$) (Chọn a_i)

Vì n ≤ 100 nên việc cài đặt thuật toán theo mô hình như trên sẽ bị tràn bộ nhớ mặc dù vẫn đủ đảm bảo về mặt thời gian. Ta sẽ khử đệ quy bằng cách dùng mảng f:

f[i + 1][j][l] = max(f[i + 1][j][l], f[i][j][l])

f[i + 1][j + 1][l + $pw5_i$] = max(f[i + 1][j + 1][l + $pw5_i$], f[i][j][l] + $pw2_i$)

Đáp án của bài toán: max(min(l, f[n + 1][k][l]))

Nhận thấy f[i + 1][…][…] chỉ cập nhật kết quả theo f[i][…][…] nên ta có thể khử 1 chiều của f bằng cách sử dụng 2 mảng 2 chiều tính toán song hành với nhau, vậy là giải quyết được vấn đề về bộ nhớ.

RACE

Trước tiên, ta sẽ dùng thuật toán Floyd tìm đường đi ngắn nhất giữa mọi cặp địa điểm đối với mỗi chiếc xe. Lưu lại kết quả vào mảng d, trong đó d[i][u][v] là độ dài đường đi ngắn nhất từ địa điểm u đến địa điểm v nếu sử dụng xe thứ i.

Sub 1: Vì m = 1 tức là ta chỉ có duy nhất 1 chiếc xe để sử dụng, khi đó ở vòng đua thứ i ta không cần quan tâm đến k_i nữa. Đáp án là d[1][s_i][f_i].

Sub 2: Tại vòng đua thứ i, vì $k_i = 0$ nên ta chỉ được sử dụng 1 chiếc xe, ta sẽ chọn chiếc tối ưu nhất. Đáp án là min(d[j][s_i][f_i]).

Sub 3: Gọi f[u][v][l] là độ dài đường đi ngắn nhất từ địa điểm u đến địa điểm v với số lần đổi xe tối đa là l. Từ sub 2 ta suy ra f[u][v][0] = min(d[i][u][v]). Dựa vào đó, ta dễ dàng tìm ra công thức QHĐ tính toán mảng f:

f[u][v][l] = min(f[u][x][l – 1] + f[x][v][0])

Tuy nhiên vấn đề ở đây $k_i \leq 10^5$ là rất lớn, nếu ta QHĐ đơn thuần theo công thức trên thì sẽ bị quá thời gian. Nhận xét là đường đi ngắn nhất ta cần tìm luôn là đường đi đơn, tức là mỗi địa điểm xuất hiện trên đường đi không quá 1 lần. Mà ta lại chỉ có thể đổi xe mỗi khi dừng lại tại địa điểm nào đó, vậy nên thực chất số lần tối đa mà ta cần đổi xe không bao giờ vượt quá n.

SEQUENCE

Sub 1: Vì n nhỏ nên ta có thể duyệt bằng đệ quy.

Sub 2: Ta sẽ sử dụng thuật toán QHĐ theo mô hình đệ quy có nhớ:

solve(pos, open) trong đó pos kiểm soát các phần tử của mảng a, open kiểm soát số lượng ngoặc mở chưa được ghép cặp với ngoặc đóng hiện tại. Hàm trả lại tổng số cách giải bài toán của Norman. Công thức QHĐ:

-Nếu a[pos] + open = h - 1: (lúc này ta bắt buộc phải tăng a[pos] lên 1 đơn vị)

+) solve(pos, open) += solve(pos + 1, open) (tạo 1 cặp ngoặc mở đóng ngay tại pos)

+) solve(pos, open) += solve(pos + 1, open + 1) (tạo thêm 1 ngoặc mở tại pos)

+) solve(pos, open) += open * solve(pos + 1, open) (tạo thêm 1 cặp ngoặc đóng mở tại pos, ngoặc đóng này có thể ghép cặp với open ngoặc mở trước đó)

-Nếu a[pos] + open = h:

+) solve(pos, open) += solve(pos + 1, open) (bỏ qua pos, về sau pos sẽ được đưa vào giữa cặp ngoặc mở đóng nào đó)

+) solve(pos, open) += open * solve(pos + 1, open - 1) (tạo 1 ngoặc đóng tại pos, ngoặc đóng này có thể ghép cặp với open ngoặc mở trước đó)

Neo đệ quy: nếu pos = n + 1, kiểm tra xem open có bằng 0 hay không. Nếu open = 0 thì trả về 1, ngược lại trả về 0.