Trại Đông Đà Lạt

Hướng dẫn giải bài tập

29-31/10/2023

Ngày 31 tháng 10 năm 2023

Bài 1. Bàn tiệc sang trọng – LUXTABLE

Bài 2. Xây dựng cây cầu – BRIDGEPOL

Bài 3. Đường đi đối xứng – STRPATH

Bàn tiệc sang trọng – LUXTABLE

Tâm đã nấu N món ăn và xếp chúng thành một hàng trên bàn tiệc, món thứ i có độ hấp dẫn là a_i và không có 2 món ăn liên tiếp nào có độ hấp dẫn giống nhau. Tâm phải nấu lại và thay thế một số món ăn. Mỗi lần nấu lại Tâm sẽ chọn một món ăn trong bàn tiệc và thay bằng món ăn mới có độ hấp dẫn khác, phải thỏa mãn 2 yêu cầu sau:

- Bàn tiệc không bao giờ có 2 món ăn liên tiếp có độ hấp dẫn giống nhau ở mọi thời điểm.
- Chỉ có chính xác 2 giá trị hấp dẫn khác nhau trong các món ăn trên bàn ăn cuối cùng.

Yêu cầu: Hãy giúp Tâm xác định phải thay thế tối thiểu bao nhiêu món ăn để thỏa mãn yêu cầu trên.

Subtask 1 (Tổng các giá trị N trong tất cả các test không vượt quá 500): $O(N^3)$

- Dễ thấy độ hấp dẫn của các món ăn trên bàn tiệc cuối cùng có dạng $[x, y, x, y, ...]$. Như vậy ta duyệt các cặp giá trị (x, y) $(1 \leq x \neq y \leq N)$ và mỗi cặp (x, y) như vậy ta đặt $cnt[x, 1]$ là số lượng số x ở vị trí lẻ, $cnt[y, 0]$ là số lượng số y ở vị trí chẵn.
- Mục đích ta duyệt cặp (x, y) để tạo mảng $[x, y, x, y, ...]$ với x ở vị trí lẻ, y ở vị trí chẵn. Nhìn qua ta nghĩ số lượng món ăn phải thay thế là: $S = N - cnt[x, 1] - cnt[y, 0]$.
- Tuy nhiên nếu trong mảng có các đoạn con $[y, x, y, x, ...]$ với y ở vị trí lẻ, x ở vị trí chẵn thì ta không thể thay thế trực tiếp y bằng x được vì sẽ có 2 món ăn liên tiếp có độ hấp dẫn giống nhau. Ta sẽ gọi đây là một đoạn con tệ. Mỗi khi xuất hiện một đoạn con tệ có độ dài L thì ta phải thay thế thêm $L/2$ món ăn nữa. Vậy số lượng món ăn phải thay thế: $S = N - cnt[x, 1] - cnt[y, 0] + (L_1/2 + L_2/2 + ... + L_k/2)$. Trong đó $L_1, L_2, ..., L_k$ là độ dài các đoạn con tệ xuất hiện trong mảng.

Subtask 2 (Tổng các giá trị N trong tất cả các test không vượt quá 4000): $O(N^2)$

- Gọi $f[x, y]$ là số món phải thay thế thêm khi xuất hiện những dãy con tệ, tức là $f[x, y] = (L_1/2 + L_2/2 + ... + L_k/2)$ ở trên.
- Ta sử dụng Two pointer để tìm kiếm các đoạn con tệ $a_i, a_{i+1}, a_{i+2}, ..., a_j$ có dạng $[x, y, x, y, \dots]$.
- Đặt $a_i = x, a_{i+1} = y$ ta có:
  - nếu i lẻ thì $f[y, x] = (j - i + 1)/2$,
  - nếu i chẵn thì $f[x, y] = (j - i + 1)/2$.

  Vậy số lượng món ăn phải thay thế với mỗi cặp (x, y) ta duyệt là: $S = N - cnt[x, 1] - cnt[y, 0] + f[x, y]$.

Subtask 3 (Tổng các giá trị N trong tất cả các test không vượt quá 200000.): $N * \log N$

Thay vì dùng mảng ta sử dụng map để tính $f[x, y]$. Sử dụng một vector để lưu các số y theo thứ tự $cnt[y, 1]$ giảm dần.
Đối với mỗi số x, ta sẽ duyệt các số y theo thứ tự trong vector và cập nhập kết quả:

- $Ans = \min(Ans, N - cnt[x, 1] - cnt[y, 0] + f[x, y])$.
- Nếu $f[x, y] = 0$ thì sau khi tính vào kết quả, ta break để thoát khỏi vòng duyệt y. Việc break giúp giảm thiểu thời gian, vì số cặp (x, y) ta xét đều có $f[x, y] > 0$ thì chỉ có N cặp (x, y) như vậy.

Bài 1. Bàn tiệc sang trọng – LUXTABLE

Bài 2. Xây dựng cây cầu – BRIDGEPOL

Bài 3. Đường đi đối xứng – STRPATH

Xây dựng cây cầu – BRIDGEPOL

Người ta đã đóng N cây cột trên dòng sông để xây dựng một cây cầu. Cột thứ i sẽ có chiều cao là h_i. Bắt buộc phải sử dụng cột thứ nhất và cột thứ N để xây dựng.
Chỉ cần chọn 1 tập các cột $h_{i_1}, h_{i_2}, \ldots, h_{i_k}$ trong đó $1 = i_1 < i_2 < \ldots < i_k = N$ để đưa vào xây cầu. Việc xây dựng cây cầu từ các cột đã chọn có chi phí là $\sum_{j=2}^{k}(h_{i_j} - h_{i_{j-1}})^2$. Đối với các cột không được sử dụng thì bắt buộc phải gỡ bỏ chúng. Cây cầu thứ i gỡ bỏ tốn chi phí là w_i. Tổng toàn bộ chi phí thực hiện tính bằng chi phí xây dựng cộng với chi phí gỡ bỏ.
Yêu cầu: Hãy tính phương án cho tổng chi phí tối thiểu có thể để thực hiện xong dự án xây cầu.

Subtask 1 ($N \leq 1000$): $O(N^2)$

Gọi $dp[i]$ là tổng chi phí tối thiểu khi xét đến cột thứ i và ta có sử dụng cột đó.

- $dp[1] = 0$.
- $dp[i] = \min\{dp[i], dp[j] + (h_j - h_i)^2 + S_{i-1} - S_j\}$ với $S_i = w_1 + ... + w_i$.

Ta có tính trước giá trị $(w_{j+1} + ... + w_{i-1})$.
Kết quả bài toán là $dp[N]$.

Subtask 2 (Giải pháp tối ưu có tối đa 2 cột được chọn thêm (không tính cột thứ nhất và cột thứ N) và $|w_i| \leq 20$): $O(N * \log N * 20)$

- Đầu tiên ta xét trường hợp chỉ có 1 cột được chọn thêm. Duyệt cột i $(1 < i < N)$ đóng vai trò là cột được chọn thêm, khi đó tổng chi phí là
  $S = (h_1 - h_i)^2 + (h_i - h_N)^2 + (w_2 + ... + w_{i-1} + w_{i+1} + ... + w_N)$.
- Với trường hợp có 2 cột được chọn thêm, Ta duyệt cột i là cột được thêm sau đó ta sẽ đi tìm cột j $(j < i)$ tốt nhất để tối ưu chi phí. Lúc này chi phí là: $S' = S - (h_1 - h_i)^2 + (h_1 - h_j)^2 + (h_j - h_i)^2 - w_j$.
- Giả sử chúng ta không quan tâm w_j, thì S' tối thiểu khi h_j gần với giá trị $((h_1 + h_i))/2$ nhất. Như vậy ta sẽ tìm 2 giá trị h_j nhỏ nhất lớn hơn hoặc bằng $((h_1 + h_i))/2$ và h_j lớn nhất nhỏ hơn hoặc bằng $((h_1 + h_i))/2$. Để làm được điều này, ta duy trì 1 set để lưu các giá trị $h_2, h_3, ..., h_{i-1}$.
- Đối với w_j, ta nhìn thấy trong subtask này, giá trị w_j khá nhỏ. Vậy ta có thể duyệt các giá trị w_j, rồi với mỗi giá trị w_j thì ta tìm h_j thỏa mãn ở trên.

Subtask 3 ($N \leq 10^5$): $O(N * \log N)$

Theo công thức quy hoạch động ở subtask 1:

$$dp[i] = \min\{dp[i], dp[j] + (h_j - h_i)^2 + S_{i-1} - S_j\}$$
$$= \min\{dp[i], -2 * h_j * h_i + dp[j] + h_j^2 - S_j + h_i^2 + S_{i-1}\}$$

Dựa vào công thức trên, ta thấy phải áp dụng quy hoạch động bao lồi.

Bài 1. Bàn tiệc sang trọng – LUXTABLE

Bài 2. Xây dựng cây cầu – BRIDGEPOL

Bài 3. Đường đi đối xứng – STRPATH

Đường đi đối xứng – STRPATH

Một bảng chữ gồm m hàng, n cột. Ô (i, j) có thể chứa một kí tự từ 'a' đến 'z' hoặc là ô cấm. Khi bắt đầu trò chơi, người chơi được cho một xâu S và nhiệm vụ của người chơi là xuất phát từ ô (1,1) cần di chuyển tới ô (m, n). Tại mỗi bước, người chơi chỉ được di chuyển sang ô bên phải hoặc ô nằm bên dưới ô hiện tại và không được phép di chuyển vào ô cấm. Người chơi được gọi là thắng cuộc nếu khi ghép lần lượt các kí tự trong các ô đi trên đường đi sẽ tạo thành một xâu đối xứng T chứa xâu S.

Yêu cầu: Cho bảng chữ và xâu S, hãy đếm số cách đi để dành chiến thắng.

Subtask 1 (20%): $m, n \leq 10$: $O(2^{m+n})$

Quay lui sinh hết đường đi.

Subtask 2 (20%): $m, n \leq 30$, ngoài các ô cấm thì các ô còn lại chứa kí tự giống nhau và xâu S chỉ có một kí tự giống với kí tự nằm ở ô (1,1) : $O(m.n)$

Bài toán chính là đếm số đường đi từ ô (1,1) đến ô (m, n), vì mọi đường đi đều tạo thành xâu đối xứng và đều chứa xâu S. Gọi $f[x, y]$ là số đường đi từ ô (1,1) đến ô (x, y), công thức xây dựng cũng chỉ liên hệ từ $f[x-1, y]$ và $f[x, y-1]$.

Subtask 3 (20%): $m, n \leq 30$ và xâu S chỉ có một kí tự nằm ở ô (1,1): $O(4.m^2.n^2)$

- Chia đường đi thành hai phần, phần một bắt đầu ở ô (1,1) và phần hai bắt đầu ở ô (m, n). Gọi $f[u, v, x, y]$ là số đường đi bắt đầu từ ô (1,1) đến ô (m, n) khi mà phần 1 đã tới được ô (u, v) và phần 2 đã tới được ô (x, y). Kết quả là tổng của mọi $f[x, y, u, v]$ mà $x < u, y < v$ và $|x - u| + |y - v| \leq 2$.
- Khi đó $(x, y) \rightarrow (x + 1, y)$ hoặc $(x, y + 1)$ ta gọi là (x', y').
- Còn $(u, v) \rightarrow (u - 1, v)$ hoặc $(u, v - 1)$ ta gọi là (u', v').
- Ta sẽ đảm bảo kí tự tại điểm (x', y') bằng kí tự tại (u', v'), và ta có công thức:

$$f[x', y', u', v'] += f[x, y, u, v]$$

Subtask 4 (20%): $m, n \leq 30$ và xâu S chỉ có một kí tự : $O(4.2.m^2.n^2)$

Tư tưởng không khác với subtask 3, nhưng phải quản lý thêm một biến để biết đã chứa xâu S hay chưa, do đó gọi $f[u, v, x, y, ok]$ là số đường đi bắt đầu từ ô (1,1) đến ô (m, n) khi mà phần 1 đã tới được ô (u, v) và phần 2 đã tới được ô (x, y) và $ok = 0/1$ để biết đã chứa xâu S hay chưa.
Tương tự như subtask 3 ta có công thức:

$$f[u', v', x', y', ok] || (\text{kí tự}(u', v') = S] += f[u', v', x', y', ok].$$

Subtask 5 (10%): $m, n \leq 30 : O(4.m^2.n.|S|^2)$

- Tư tưởng như subtask 4, nhưng thay vì quản lý biến $ok = 0/1$ thì ok có thể nhận giá trị từ $0 \rightarrow length(S)$, ok cho biết phần suffix độ dài ok của đường đi $(1, 1) \rightarrow (u, v)$ trùng với phần prefix độ dài ok của S.
- Tới đây thì phức tạp hơn vì khi đi thêm một kí tự vào đường đi thì thay đổi giá trị ok để ý nghĩa đó được đảm bảo. Để làm việc này ta cần xây dựng trước một mảng $NxtPrefix[u, ch]$ cho biết nếu như đang trùng với prefix độ dài u, mà tiếp theo đi vào kí tự ch thì nó sẽ reset về bao nhiêu, tất nhiên $NxtPrefix[length(S), ch] = length(S)$ với mọi ch.

Subtask 5 (10%): $m, n \leq 30$

- Gọi $f[u, v, x, y, pre, suf]$ với ý tưởng như subtask 4 nhưng ở đây *pre* và *suf* thay thế cho biến *ok*, *suf* vai trò giống với *pre* nhưng là của xâu S đảo ngược. Đến lúc phần một và phần hai giao nhau, giả sử ở trường hợp số kí tự là lẻ thì nó sẽ dừng lại khi $u = x$ và $v = y$, đến lúc đó, chúng ta có *pre* và *suf*, và chỉ cần kiểm tra xem xâu "$S[1 \ldots pre] + S[length(S) - suf + 1 \ldots length(S)]$" có chứa xâu S hay không là giải quyết được bài toán.
- Để kiểm tra nhanh thì ta có thể dễ dàng chuẩn bị được mảng $check[pre, suf] = 0/1$ với ý nghĩa là "$S[1 \ldots pre] + S[length(S) - suf + 1 \ldots length(S)]$" chứa xâu S hay không. Hiện tại thì không gian lưu trữ là $O(30^4.20^2) > O(3.10^8)$, việc lưu trữ là không khả thi!
- Nhưng xét thấy $u + v = x + y$, do đó chúng ta chỉ cần duy trì u, v, x còn biến y có thể dễ dàng suy ra được, như vậy không gian lưu trữ lúc này chỉ cỡ $O(30^3.20^2) < O(2.10^7)$, OK!
- Gọi kí tự ở vị trí (u', v') là *ch*, ta có:

  $f[u', v', x', NxtPrefix[pre, ch], NxtSuffix[suf, ch]] += f[u, v, x, pre, suf]$

Subtask 6 (10%): $m, n \leq 150$

- Do tính chất đối xứng của đường đi nên ta chỉ cần lưu *pre* hoặc *suf*, ở đây ta sẽ lưu giá trị lớn hơn vì rõ ràng giá trị lớn hơn quản lý được nhiều giá trị hơn.
- Giả sử $pre > suf$, thì trước đó ta lại xây dựng thêm một mảng $LCS[pre]$ là độ dài suffix dài nhất của xâu S mà trùng với xâu đảo ngược của suffix trên xâu $S[1 \ldots pre]$. Ví dụ $LCS[3]$ trên xâu "cbadab" là 2. Tức là khi có được giá trị lớn hơn ở một trong hai giá trị thì sẽ tính được giá trị còn lại.
- Mặc dù vậy ta cần biết được giá trị mình lưu là *pre* hay *suf*, để quản lý điều này thì ta thêm một biến $id = 0/1$ với 0 là giá trị *pre* đang được lưu trữ còn 1 là đang lưu trữ *suf*.
- Như vậy hàm đệ quy của chúng ta sẽ là $f[u, v, x, len, id]$ với $len = pre/suf$ và $id = 0/1$, không gian lưu trữ là $O(m^2.n.|S|.2) \leq O(150^3.20.2) < O(1,5.10^8)$, OK!

Subtask 6 (10%): $m, n \leq 150$

- Về công thức, với $id = 0$, giả sử ta đang lưu trữ pre, gọi kí tự ta muốn đi tới là ch, khi đó ta sẽ lấy $suf = LCS[pre]$, ta đặt $pre' = NxtPrefix[pre, ch]$ và $suf' = NxtSuffix[suf, ch]$ tương tự như subtask 5. Sau đó, công thức của chúng ta sẽ hình thành:

$$f[u', v', x', pre' > suf' \ ? \ pre' : suf', pre' > suf' \ ? \ 0 : 1]$$

$$+= f[u', v', x', len, id]$$

Trong đó: $a > b \ ? \ c : d$ sẽ trả về c khi $a > b$, ngược lại thì trả về d.