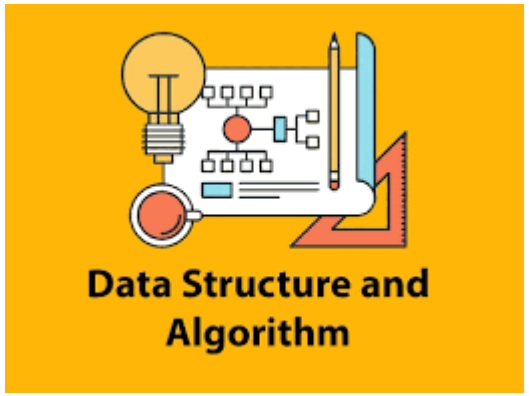
ĐỘI TUYỂN OLP Tin học PTIT 2025

CHỮA BÀI TUẦN 3 (Oct 13 – Oct 19)



Contest cá nhân – Thứ 7

- Bài 1: Dãy số (DSU – segment tree)
- Bài 2: Phân số nhỏ nhất (Bitmask)
- Bài 3: Bắt cá (Toán, xử lí sai số)
- Bài 4: Diện tích N hình chữ nhật (Sweepline)

Bài 1: Dãy số - HMĐức

Tóm tắt đề

- Cho dãy N số a[1], a[2], …, a[n]
- Cho Q thao tác thuộc một trong 3 loại:
  - Loại 1: **1 i x** – gán a[i] = x
  - Loại 2: **2 i** – truy vấn giá trị của a[i]
  - Loại 3: **3 l r** – Với mọi **i** sao cho a[i] thuộc đoạn [L, R], nếu a[i] <= (L+R)/2, gán a[i] = L – 1; ngược lại, gán a[i] = R + 1.
- Yêu cầu: In ra kết quả của các truy vấn loại 2.

Bài 1: Dãy số - HMĐức

Lời giải:

- Chia truy vấn loại 3 (L, R) thành 2 truy vấn dạng (L', R', W') là chuyển tất cả các số có giá trị trong đoạn [L', R'] thành W'.
- Xét TH không có truy vấn loại 1:
- Thấy rằng nếu sau truy vấn (L', R', W') là có (L'', R'', W'') gần nhất sao cho W' thuộc [L'', R''] thì các giá trị a[i] ban đầu thuộc đoạn [L', R'] bằng W''
  - => Nếu tiếp tục thực hiện thao tác này với (L'', R'', W'') đến khi không tìm được truy vấn đằng sau nữa thì có thể xác định được giá trị a[i] nếu nằm trong đoạn [L', R']
- Để tìm được truy vấn kết thúc nhanh chóng, có thể áp dụng DSU, khi xét đến truy vấn (L'', R'', W'') thì kết nối những truy vấn (L', R', W') thoả mãn điều kiện trên mà chưa kết nối đến truy vấn nào sau nó đến truy vấn (L'', R'', W'').

Bài 1: Dãy số - HMĐức

Lời giải:

- Nếu có truy vấn loại 1:
- Coi mỗi truy vấn loại 1 là một đỉnh truy vấn (-1, -1, x) và thực hiện như phần trên.
  - Để dễ dàng cài đặt, coi dãy số ban đầu là danh sách các truy vấn loại 1.
- Với mỗi truy vấn loại 2, tìm lần cuối cùng thực hiện truy vấn loại 1 trên đỉnh **i** và lấy kết quả từ đó .

Bài 1: Dãy số - ver 2

Tóm tắt đề

- Cho dãy N số a[1], a[2], …, a[n]
- Cho Q thao tác thuộc một trong 3 loại:
  - Loại 1: **1 i x** – gán a[i] = x
  - Loại 2: **2 i** – truy vấn giá trị của a[i]
  - Loại 3: **3 l r** – Với mọi **i** sao cho a[i] thuộc đoạn [L, R], nếu a[i] <= (L+R)/2, gán a[i] = L – 1; ngược lại, gán a[i] = R + 1.
- Yêu cầu: In ra kết quả của các truy vấn loại 2.
- Nhận xét: Sau mỗi truy vấn 3, giá trị khoảng [L, mid] cùng biến thành giá trị giống nhau (L-1), khoảng [mid R] thành R+1. Các giá trị này sẽ chuyển động cùng nhau trong các thao tác 3 tiếp theo.

Bài 2: Phân số nhỏ nhất

- Cho phân số P và Q. Cần xóa đi một tập chữ số ở tử số P để thu được A, xóa tập chữ số tương tự của mẫu số Q để thu được B sao cho P/Q = A/B và A nhỏ nhất có thể?

- **Subtask 1:** Có 9 chữ số, xóa đi K chữ số của P, xóa đi K chữ số của Q, kiểm tra xem 2 tập bị xóa này có giống nhau và kết quả phân số có bằng nhau hay không?

- **Subtask 2:** có 19 chữ số, liệt kê tất cả 2^19 lựa chọn ở tử số P (thu được A), rồi tìm B và kiểm tra có thể thu được B từ Q bằng cách xóa đi 1 số lượng chữ số giống P hay không?

Cần xử lí khéo vì giới hạn thời gian là 1s, nếu không bị TLE 1,2 test.

Bài 3: Bắt cá

- Phương trình con cá:

$p(t) = X + V * t$

- Con cá j nằm trước con cá i một đoạn đúng bằng A nếu như:

$X_j + V_j*t = X_i + V_i*t + A$

$\Rightarrow t = (X_j - X_i - A) / (V_i - V_j)$, nếu V_i khác V_j

- **Subtask 1:** Duyệt tập con K của N (N <= 20) con cá rồi kiểm tra xem có tồn tại thời điểm t để K con này nằm chung trong một đoạn bằng A hay không?

Bài 3: Bắt cá

- **Nhận xét:** đoạn thả lưới A cần chứa ít nhất 2 con cá
- Tổng quát, với con cá thứ i, vị trí tương đối của con cá thứ j là:

$$\Delta x_{ij}(t) = (x_j - x_i) + (v_j - v_i)\, t.$$

- Con cá j nằm trong phạm vi cửa sổ A của con cá thứ i nếu như:

$$0 \leq \Delta x_{ij}(t) \leq A.$$

- Mốc sự kiện khi con cá j vào "vùng" và thoát vùng của con cá i:

$$\Delta x_{ij}(t) = 0 \quad \text{hoặc} \quad \Delta x_{ij}(t) = A$$

Bài 3: Bắt cá

- **Nhận xét:** đoạn thả lưới A cần chứa ít nhất 2 con cá
- Tổng quát, với con cá thứ i, vị trí tương đối của con cá thứ j là:

$$\Delta x_{ij}(t) = (x_j - x_i) + (v_j - v_i)\,t.$$

- Giải phương trình trên, ta có:
- Nếu $v_j > v_i$
  - Thời gian vào (chạm biên trái): $t = \dfrac{x_i - x_j}{v_j - v_i}$
  - Thời gian ra (chạm biên phải): $t = \dfrac{x_i + A - x_j}{v_j - v_i}$
- Nếu $v_j < v_i$
  - Thời gian vào (chạm biên trái): $t = \dfrac{x_j - x_i}{v_i - v_j}$
  - Thời gian ra (chạm biên phải): $t = \dfrac{x_j - x_i - A}{v_i - v_j}$
- Nếu vận tốc bằng nhau: phụ thuộc vào khoảng cách ban đầu

Bài 3: Bắt cá

- **Subtask 2: O(N^3):** chốt 2 con cá i, j rồi kiểm tra con cá k có nằm trong đoạn thời gian của (i, j) hay không?
- **Subtask 3: O(N^2 log N):** khung cửa sổ
  - Với mỗi con cá i, duyệt các con cá j tìm ra mốc 2 sự kiện T_in, T_out
  - Sorting tất cả các sự kiện lại
  - Dùng nguyên lý khung cửa sổ để cập nhật giá trị các con cá có thể bắt được, answer = max(tổng giá trị cho mỗi lần trượt)

Bài 3: Bắt cá

- **Lưu ý: sai số do số thực**
- Test mạnh sẽ làm cho việc sort các sự kiện lưu dưới dạng số thực bị mismatch với nhau.
- Ví dụ như 2 sự kiện 2.9999 và 3.0001 bị coi là 2 sự kiện rời rạc
- Hướng giải quyết: xử lí với số nguyên, lưu các mốc thời gian dưới dạng phân số

```
int main() {
    float x = 0.3;
    double y = 0.3;
    long double z = 0.3;
    cout << fixed << setprecision(60);
    cout << x << endl;
    cout << y << endl;
    cout << z << endl;

}
/**
0.300000011920928955078125000000000000000000000000000000000000
0.299999999999999988897769753748434595763683319091796875000000
0.299999999999999988897769753748434595763683319091796875000000
```

Bài 4: Diện tích N hình chữ nhật

- Subtask 1: N <= 20

Dùng nguyên lý inclusive – exclusive cho từng tập K hình tam giác

- Subtask 2: N <= 100, các tọa độ <= 10^5

Sử dụng sweep line

2025 ICPC vòng miền Trung

A: Khôi

G: Vinh

K: thầy Kiên

Bài A: Ascending Garden

Tóm tắt đề

- Cho mảng n phần tử, được thực hiện m thao tác, mỗi thao tác chọn ra 1 đoạn [l, r] và tăng các phần tử a[i] thêm 1 đơn vị

(l <= i <= r).

- Giới hạn: n, m, a[i] <= 20

- Đếm số lượng cách khác nhau thực hiện m thao tác sao cho khi thực hiện xong thì mảng a sẽ tăng tuyệt đối. Kết quả lấy modulo 1e9 + 7.
- Ví dụ với m = 2 và mảng A ban đầu = [1, 1] thì:

[{1, 2}, {2, 2}] và [{2, 2}, {1, 2}] được coi là 2 cách khác nhau

Bài A: Ascending Garden

Lời giải

- Gọi F[i][j][k] là số cách thực hiện k thao tác khi xét đến phần tử thứ i và hiện tại có j thao tác đang kết thúc ở phần tử thứ i
- Vì hiện tại có j thao tác đang kết thúc ở phần tử thứ i nên ta giả sử gọi x là số lượng thao tác có dạng [i, i] trong j thao tác này, khi đó ta biết được ở i - 1 sẽ có j - x thao tác kéo dài từ i - 1 lên i, gọi h là số thao tác kết thúc ở i - 1 khi đó ta có công thức chuyển trạng thái là:

$$F[i][j][k] \mathrel{+}= F[i - 1][h][k - x] * C(j - x, h) * C(x, k)$$

- Vì ở i - 1 có h thao tác đang kết thúc ở đó mà ta cần chọn ra j - x thao tác để kéo lên i nên số cách sẽ nhân với C(j - x, h)
- Vì có tổng cộng k thao tác nên ta cần chọn ra x thao tác để ở đó sẽ thực hiện thao tác [i, i] còn những thao tác còn lại đã được tính và sắp xếp theo thứ tự trong F[i - 1][h][k - x] nên ta sẽ nhân với F[i - 1][h][k - x] và C(x, k)
- ĐPT: Ta cần for i, j, k, h, và x nên ĐPT sẽ là O(n ^ 5)

Bài G: Garden of Stone

Tóm tắt đề

- Cho bảng N hàng M cột chỉ gồm 0 và 1.

- Có Q truy vấn có dạng H W K :
- hỏi có bao nhiêu hình chữ nhật con cao H rộng W mà mỗi cột chứa không quá k số 1.

- $N * M <= 10^6$
- $Q <= 10^5$
- $K <= 5$

Bài G: Garden of Stone

Tóm tắt đề

- Cho bảng N hàng M cột chỉ gồm 0 và 1.

- Có Q truy vấn có dạng H W K :
- hỏi có bao nhiêu hình chữ nhật con cao H rộng W mà mỗi cột chứa không quá k số 1.

- $N * M <= 10^6$
- $Q <= 10^5$
- $K <= 5$

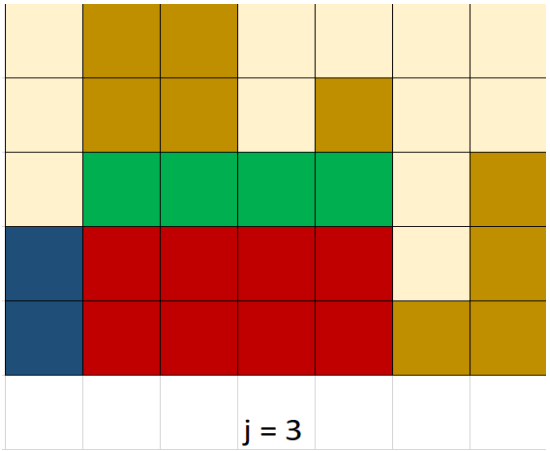
Bài G: Garden of Stone

Ý tưởng:

- Với mỗi K ta cần đếm các HCN u x v với mọi u, v. Khi đã đếm được mọi hình như vậy ta chỉ cần query trong O(1)

- Với mỗi hàng i ta tìm xét mỗi ô (i, j) có thể đi xuống thêm bao nhiêu ô nữa mà vẫn thỏa mãn có nhiều nhất K số 1.
- Gọi dãy vừa tìm là a[1], a[2], …, a[m].
- Dùng stack để tìm đoạn [L, R] rộng nhất sao cho a[j] là min
- Khi đó ta sẽ được một HCN có độ cao h = a[j], rộng là w = R – L + 1.
- Vậy ta cần tăng các hình con (x, y) của nó lên (h – x + 1) * (w – y + 1)
- Nhưng như vậy có thể bị thừa vì có thể hình con của hình mà ta đang xét cũng là hình con của một hình chữ nhật khác. Như hình sau:

Bài G: Garden of Stone



- Ví dụ như đang xét hình chữ nhật tại j = 3 ta thấy phần màu đỏ bị trùng với hình chữ nhật màu xanh biển.
- Vậy mỗi hình ta sẽ chỉ tăng kết quả cho các hình con có h từ max(a[L-1], a[R + 1]) + 1 đến a[j] để tránh trùng
- Ví dụ:

  Với j = 0, ta chỉ cộng thêm cho các hình con có h từ a[5] + 1 đến a[j], tức h từ 2 đến 2

  Với j = 3, ta chỉ cộng them cho các hình con có h từ max(a[0], a[5]) + 1 đến a[j], tức h từ 3 đến 3

Bài G: Garden of Stone

- Với mỗi h x w đang xét thì số HCN:
  - h x w sẽ tăng lên 1
  - h x (w – 1) sẽ tăng lên 2
  - h x (w – 2) sẽ tăng lên 3
  - …
  - h x 1 sẽ tăng lên w

  Do đó, số HCN h x w’ sẽ tăng lên w – w’ + 1

  Vậy để biết số lượng HCN con h x w’ ta cần biết tổng w của các HCN h x w với w >= w’ và số lượng các HCN h x w với w >= w’.

Bài G: Garden of Stone

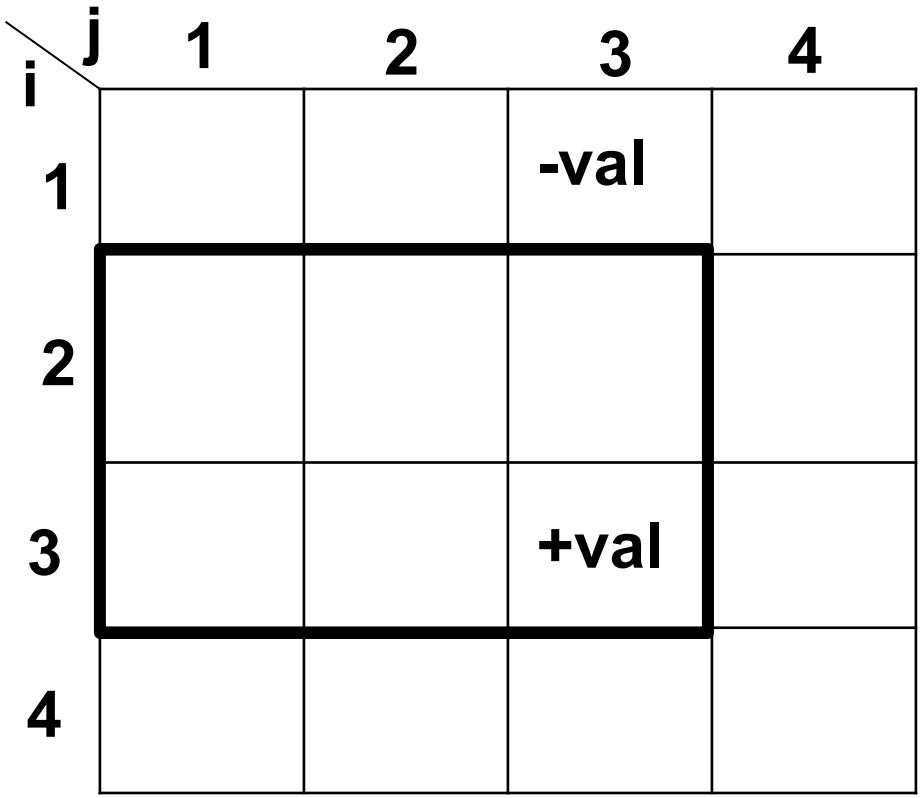
Ta sẽ tạo 2 mảng 2D để tính

Đặt lo = max(a[L – 1], a[R + 1]) + 1

- Với mỗi HCN h x w
  - Trên mảng thứ nhất, ta cộng w vào mọi ô (i, j) sao cho lo ≤ i ≤ h và 1 ≤ j ≤ w
  - Trên mảng thứ hai, ta cộng 1 vào mọi ô (i, j) sao cho lo ≤ I ≤ h và 1 ≤ j ≤ w
- Làm như vậy ta sẽ mất $O(n^2)$
- Thay vì vậy ta sẽ dùng mảng cộng dồn để tính

Bài G: Garden of Stone

- Từ cập nhật bảng con ta sẽ chuyển sang bài toán cập nhật ô rồi tổng dồn sau

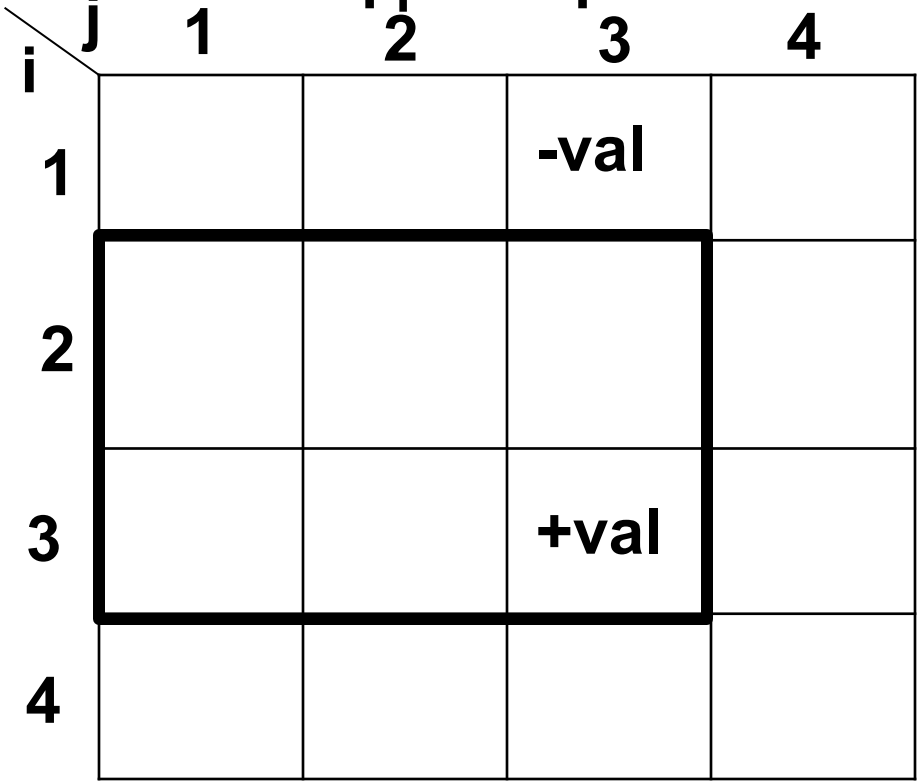


lo = 2
h = 3, w = 3

Thay vì update bảng con thì ta có thể update hai ô (h, w) và (lo – 1, w) rồi sau đó tổng dồn 2D

Bài G: Garden of Stone

- Từ cập nhật bảng con ta sẽ chuyển sang bài toán cập nhật ô rồi tổng dồn sau

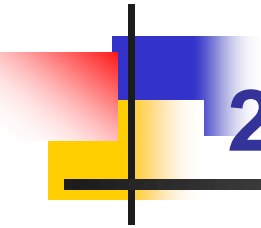


lo = 2
h = 3, w = 3

Thay vì update bảng con thì ta có thể update hai ô (h, w) và (lo – 1, w) rồi sau đó tổng dồn 2D

Bài G: Garden of Stone

- Tổng kết
  - Với mỗi k ta đếm số lượng mọi HCN trong O(n*m)
  - ĐPT: k * n * m



2025 ICPC Wuhan Invitation Contest

J: anh Dương

LCP table

Problem J: Dictionary

Tóm tắt đề

Cho một xâu S độ dài n. Ban đầu ta có một danh sách rỗng. Cho q truy vấn, mỗi truy vấn có dạng [l, r], cần thêm vào danh sách các xâu con có tiền tố là s[l, r]. Sau mỗi truy vấn, in ra số lượng xâu phân biệt trong danh sách ban đầu.

Problem J: Dictionary

Ví dụ:

abcabd

3

1 3

1 2

6 6

Sau truy vấn 1: [abc, abca, abcab, abcabd] -> 4

Sau truy vấn 2: [abc, abca, abcab, abcabd, ab, abd] -> 6

Sau truy vấn 3: [abc, abca, abcab, abcabd, ab, abd, d] -> 7

Problem J: Dictionary

Kiến thức cần có:

Suffix array: Mảng chứa tất cả các hậu tố của xâu ban đầu, sắp xếp theo thứ tự tăng dần về thứ tự từ điển (chỉ lưu lại vị trí của hậu tố)

- Ví dụ: S = abcabd
- Các hậu tố của S: [abcabd, bcabd, cabd, abd, bd, d]
- Sắp xếp lại các hậu tố theo thứ tự từ điển:
  [abcabd, abd, bd, bcabd, cabd, d]

-> Suffix array: [0, 3, 5, 1, 2, 6]

Problem J: Dictionary

Kiến thức cần có:

LCP (Longest common prefix): Mảng chứa độ dài tiền tố dài nhất giữa hai hậu tố kề nhau trong Suffix array, nói cách khác $lcp_i =$ độ $dài\ tiền\ tố\ dài\ nhất\ giữa\ suffix(i)\ và\ suffix\ (i+1)$

- Ví dụ:
- Suffix array: [abcabd, abd, bd, bcabd, cabd, d]
- LCP: [2, 0, 1, 0, 0]

Độ dài tiền tố lớn nhất giữa hai suffix i và j ($i \leq j$) bất kì là $\min(lcp_i, lcp_{i+1}, \dots, lcp_{j-1})$ -> Có thể xây Sparse table để tính toán trong O(1)

Problem J: Dictionary

Kiến thức cần có:

Xét bài toán: Cho xâu S độ dài n, đếm số lượng xâu con phân biệt của S.

Đáp án của bài toán:

Số xâu con phân biệt = (Tổng số xâu con) – (Số lượng xâu con bị trùng)

$$= \frac{n(n+1)}{2} - \sum_{0}^{n-2} lcp_i$$

Problem J: Dictionary

Lời giải:

- Xây mảng hậu tố **sa** và **lcp** dựa trên xâu S ban đầu, đồng thời xây sparse table để truy vấn $LCP(i, j)$ với hai suffix i và j bất kì
- Với mỗi truy vấn [l, r], tìm tất cả các suffix có cùng prefix là s[l, r]. Các suffix này sẽ nằm nằm trong một đoạn liên tiếp trong **sa**. Gọi L và R là đoạn mà suffix tại $sa_L, sa_{L+1}, \dots, sa_R$ cùng có prefix là s[l,r]. L, R có thể được tìm bằng binary search. Cụ thể:
  - L là vị trí i nhỏ nhất: $LCP(sa_i, sa_l) \geq r - l + 1$
  - R là vị trí j lớn nhất: $LCP(sa_l, sa_j) \geq r - l + 1$
- Khi đó số lượng xâu phân biệt ở mỗi truy vấn là:

$$\sum_{i=L}^{R}(n - sa_i) - (R - L + 1)(r - l) + \sum_{i=L}^{R-1} lcp_i - (R - L)(r - l)$$

Problem J: Dictionary

Lời giải:

- Khi đó số lượng xâu phân biệt ở mỗi truy vấn là:

$$\sum_{i=L}^{R}(n - sa_i) - (R - L + 1)(r - l) + \sum_{i=L}^{R-1} lcp_i - (R - L)(r - l)$$

Trong đó:

- Tổng số lượng xâu có prefix là s[l, r]:

$$\sum_{i=L}^{R}(n - sa_i) - (R - L + 1)(r - l)$$

- Số lượng xâu con bị trùng:

$$\sum_{i=L}^{R-1} lcp_i - (R - L)(r - l)$$

Problem J: Dictionary

Lời giải:

- Tạo 2 segment tree:
  - seg_sa: tính tổng xâu con "còn lại" trong đoạn
  - seg_lcp: tính tổng lcp "còn lại" trong đoạn
- Ban đầu, cập nhật seg_sa với $seg\_sa_i = n - sa_i$ và $seg\_lcp_i = lcp_i$. Khởi tạo res = 0 là số lượng xâu phân biệt hiện tại.
- Đáp án ở mỗi truy vấn có thể được viết lại:

$$add = seg\_sa.query(L, R) - (R - L + 1)(r - l) + seg\_lcp.query(L, R - 1) - (R - L)(r - l)$$

- Nếu add <= 0: không làm gì thêm
- Nếu add > 0: tăng res thêm add, đồng thời:
  - Cập nhật seg_sa trong đoạn [L, R] với cùng giá trị (r – l)
  - Cập nhật seg_lcp trong đoạn [L, R – 1] với cùng giá trị (r – l)
- Độ phức tạp: $O(nlogn + qlogn)$