Thi thử TST2022 - Ngày 2

Hướng dẫn giải

27-03-2022

Bài 1. SUMDIS

Bài 2. BALANCE

Bài 3. towns

Đường đi trên bảng — SUMDIS

Tính tổng tất cả các độ dài đường đi ngắn nhất của các cặp ô trắng trong bảng.

Subtask 1: $N \leq 30$, $H, W \leq 40$

Với mỗi ô trắng ta sẽ sử dụng thuật toán BFS. Độ phức tạp $O(H^4)$.

Subtask 2: $1 \leq N \leq 2$, $H, W \leq 10^6$

Xét các cặp ô bị tăng độ dài đường đi ngắn nhất.

Subtask 3: $N \leq 30$, $H, W \leq 10^6$

- Nhận xét: bảng vuông trong bài rất thưa!
- Xét hai hàng liên tiếp cùng không chứa ô đen nào, (hàng i và hàng $i+1$). Gọi hàng trên là UP, hàng dưới là DOWN.
- Nhận xét: các đường đi ngắn nhất mà cắt qua cạnh giữa hai hàng này khi và chỉ khi một ô thuộc phần UP và một ô thuộc phần DOWN. Do vậy nếu ta thay đổi trọng số của cạnh nối các ô hàng i và $i+1$ thì giá trị cuối cùng sẽ giảm đi (số ô trắng thuộc UP) x (số ô trắng thuộc DOWN).
- Từ đây chúng ta có thể nén các hàng toàn ô trắng cạnh nhau lại với nhau. Từ đó ta được một bảng vuông có số ô $O(N * N)$.
- Lưu ý chúng ta cần chú ý đến trọng số của các ô. Ví dụ khi merge hai hàng lại với nhau thì mỗi ô ở hàng mới sẽ có trọng số là tổng 2 ô tương ứng của 2 hàng được merge.
- Và khi xét 2 ô (A,B) trong bảng mới được nén, khi đó ta sẽ cần cộng vào kết quả độ dài đường đi ngắn nhất (A, B) * weights[A] * weights[B].

Bài 1. SUMDIS

Bài 2. BALANCE

Bài 3. towns

Dãy cân bằng — BALANCE

Một dãy các số nguyên dương $a_1, a_2, \ldots, a_K$ được gọi là dãy cân bằng nếu thỏa mãn:

$$a_1 + a_2 + \ldots + a_K = a_1 * a_2 * \ldots * a_K$$

Gọi $f(K)$ là số lượng dãy cân bằng có độ dài là $K (K > 1)$.
Yêu cầu: Cho một số nguyên dương N, hãy tính các giá trị $f(2), f(3), \ldots, f(N)$. Vì các giá trị rất lớn nên hãy lấy dư khi chia cho P với (P là số nguyên tố).

Subtask 1: $N \leq 3$

Giải bằng tay.

Subtask 2: $N \leq 15$

- Ta có thể chứng minh được với $a_1 + a_2 + \ldots + a_N = a_1 * a_2 * \ldots * a_N$ thì $a_i \leq N$.
- Do đó $a_1 + a_2 + \ldots + a_N \leq N^2$ dẫn đến $a_1 * a_2 * \ldots * a_N \leq N^2$.
- Gọi $dp[i, sum, product]$ là số lượng cách tạo ra bộ $(a_1, a_2, \ldots, a_N)$ mà tổng của chúng là sum còn tích của chúng là $product$.
- ĐPT: $O(N^6)$.

Subtask 3: $N \leq 50$

- Ta sẽ chia a_i thành hai loại giá trị: $a_i = 1$ và $a_i > 1$.
- Nhận xét: số lượng giá trị $a_i > 1$ không vượt quá $2 * \log_2 N$ do tích của các giá trị này đúng bằng $a_1 + a_2 + \ldots + a_N \leq N^2$.
- Từ đó suy ra $a_1 + a_2 + \ldots + a_N < 2 * N * \log_2 N + N$ vì có không quá $2 * \log_2 N$ giá trị lớn hơn 1, còn lại đều bằng 1. Vậy ta có thể xem tổng và tích xấp xỉ $N * \log N$. Nên ta sẽ quy hoạch động trên các giá trị $a_i > 1$ và sau đó chèn các số 1 vào sau cùng.
- Gọi $g[i, sum, product]$ là số lượng dãy số gồm i giá trị (các giá trị lớn hơn 1) có tổng là sum và tích là $product$.
- Với một giá trị $g[i, sum, product]$ ta nhận thấy cần phải chèn $product - sum$ giá trị 1, khi đó ta sẽ dùng tổ hợp để đưa vào $f[product - sum]$.
- ĐPT: $O((N * \log_2 N)^3)$.

Subtask 4: $N \leq 400$

- Tiếp tục tối ưu subtask 3, gọi *sum* và *product* lần lượt là tổng và tích của các giá trị lớn hơn 1. Khi đó $product - sum$ là một giá trị không âm và không giảm khi thêm một giá trị lớn hơn 1 vào.
- Chứng minh được: $product - sum \leq product * x - (sum + x)$ với $x \leq 2$.
- Mặt khác, giá trị $product - sum$ chính là số lượng giá trị 1 chèn vào sau cùng do đó $product - sum \leq N$. Như vậy, ta có thể gọi $p[i, product, delta]$ hoặc $p[i, sum, delta]$ thay cho $g[i, sum, product]$ như subtask 3 để giảm thiểu bộ nhớ và thời gian ($delta = product - sum$).
- Chọn $p[i, product, delta]$ bởi vì khi ta chọn giá trị x trong chuyển trạng thái thì ta chỉ cần chọn *product* là tích của giá trị x (tương tự như sàng nguyên tố) thì ta sẽ chỉ mất thêm lnN chuyển trạng thái.
- ĐPT: $O(N^2 * \log_2 N * lnN)$.

Subtask 5: $N \leq 2 * 10^5$

- Đặt $W = a_1 * a_2 \ldots a_k - (a_1 + a_2 + \ldots + a_k)$.
- Ta sẽ chỉ cần duyệt hết tất cả các bộ $(a_1, a_2, \ldots, a_k)$ với giá trị $W \leq N$ và $a_i \leq N$. Ta xây dựng với các giá trị $a_i > 1$, các giá trị $a_i = 1$ ta sẽ chèn sau cùng.
- Gọi hàm đệ quy $call(sum, product, size, ways, iter)$ nghĩa là duyệt đến giá trị $iter$, có tổng và tích hiện tại là sum và $product$, độ dài lúc này là $size$ và có $ways$ dãy như thế.
- Bắt đầu bằng cách gọi hàm $call(0, 1, 0, 1, 2)$.
- Nếu giá trị $iter$ không được chọn, chuyển sang $call(sum, product, size, ways, iter + 1)$.
- Tiếp tục làm khi $iter \leq N$ và $size + product - sum \leq N$ và $product * iter - (sum + iter) \leq N$.

Subtask 5: $N \leq 2 * 10^5$

- Nếu chọn T giá trị *iter* chèn vào thì sẽ gọi đến hàm: $call(sum + T * iter, product * iter^T, size + T, ways * \binom{size+cnt}{cnt}, iter + 1)$.
- Đặt cận $size + T + product * iter^T - (sum + T * iter) \leq N$.
- Cuối cùng thêm $ways * \binom{size+W}{size}$ vào $f[size + W]$.
- Kể cả với $N \leq 10^6$ bạn cũng chỉ mất 2s để sinh ra toàn bộ các trường hợp theo hàm đệ quy này.
- Lưu ý tràn stack trong quá trình đệ quy!

Bài 1. SUMDIS

Bài 2. BALANCE

Bài 3. towns

towns

Đây là một bài toán tương tác trong đề IOI 2015.

Hướng dẫn giải

- Đầu tiên tìm (u, v) là đường kính của cây hay đường đi dài nhất: sử dụng $2n - 3$ câu hỏi dạng $(0, u)$ để tìm u xa 0 nhất và (u, v) để tìm v xa u nhất. Vậy cần $\lceil 3n/2 \rceil + 3$ câu để xác định hub và xem có cân bằng hay không.
- Nhận xét 1: có duy nhất 1 hoặc 2 hub nằm trên (u, v) là tâm của cây và cũng nằm trên đường $(0 - u)$.
- Nhận xét 2: cần xác định tất cả các thành phố lớn trên đường $0 - u$. Với mỗi thị trấn nhỏ x, gọi x' là thành phố lớn mà đường $(0 - u)$ cắt đường $(x - u)$: như vậy khoảng cách $0 - x'$ là $(d(0, u) + d(0, x) - d(u, x))/2$.
  Do đó ta tính được các hub.
- Nhận xét 3: mỗi nút trên đường $0 - u$, ta tính được số lượng nút nối với $0 - u$ bởi nút đó. Điều này giúp ta một phần trả lời câu hỏi 1 hub có là cân bằng không: nếu có nhiều hơn $n/2$ lá nằm về 1 bên của đường $0 - u$ thì hub ko cân bằng.

Hướng dẫn giải

Sau các nhận xét trên ta tìm ra được duy nhất 1 hub đặt tên là h còn nghi vấn cân bằng hay không.

- Xây dựng tập các lá nối với $0 - u$ tại h. Loại h đi thì những lá này nằm vào 1 hoặc nhiều hơn 1 thành phần.
- Bây giờ cần xác định sự cân bằng $n/2$ ở các thành phần: lấy bất kỳ 2 lá, hỏi khoảng cách để xác định xem chúng có nằm cùng 1 thành phố hay ko sau khi loại h.
- Bài toán đưa về: cho 1 mảng n phần tử, hãy xác định mảng đó chứa phần tử trội hay ko: nghĩa là tồn tại 1 phần tử có xuất hiện quá $n/2$ lần hay ko: thuật toán sử dụng $\lceil 3n/2 \rceil - 2$ phép so sánh sử dụng phương pháp major voting algorithm. `https://gregable.com/2013/10/majority-vote-algorithm-find-majority.html`

Hướng dẫn giải

Các bài interactive sử dụng template sau để test:

```
int getInput() {

#ifdef LOCAL
  return
      generateLocalInputOrAnythingYouWant();
#else
  int x;
  cin>>x;
  return x;
#endif
}

x = getInput();
```