Câu 1. BĂNG RÔN OLYMPIC

Có tất cả 20 test, mỗi test được 0.3 điểm với phân bố như ràng buộc ở đề bài.

Subtask 1: Có 25% số test ứng với 25% số điểm của bài thỏa mãn: $3 \le n \le 10^2$.

Duyệt vét cạn hai lần theo cặp chỉ số (ℓ, r) sao cho $1 \le \ell < r \le n$.

Ứng với mỗi cặp chỉ số, sử dụng thêm một vòng lặp nữa để đếm các ký tự O, L, P trong chuỗi con (ℓ, r) của băng rôn đã cho xem chuỗi con có thỏa mãn điều kiện "đẹp" hay không. Mỗi lần được thêm một chuỗi con thỏa mãn thì tăng biến đếm kết quả lên 1 đơn vị.

Độ phức tạp: $O(n^3)$.

Subtask 2: Có 25% số test ứng với 25% số điểm của bài thỏa mãn: $10^2 \le n \le 10^3$.

Để cải tiến hơn so với cách trên, ta tạo trước ba mảng cộng dồn x_i, y_i, z_i đếm số lượng chữ cái O, L, P trong đoạn từ 1 đến i. Để đếm số chữ cái O trong chuỗi con (ℓ, r), ta chỉ cần tính giá trị $x_r - x_{\ell-1}$ là được; tương tự với các chữ cái L, P sẽ lần lượt là $y_r - y_{\ell-1}$ và $z_r - z_{\ell-1}$.

Như vậy, ở mỗi lần thực hiện hai vòng lặp, ta chỉ tốn chi phí $O(1)$ để kiểm tra xem chuỗi con tương ứng có thỏa mãn tính chất đẹp hay không.

Độ phức tạp: $O(n^2)$.

Subtask 3: Có 50% số test ứng với 50% số điểm của bài thỏa mãn: $10^3 < n \le 10^5$.

Ta có nhận xét rằng một chuỗi con có độ dài ít nhất là 7 thì theo nguyên lý Dirichlet, sẽ có một ký tự xuất hiện ít nhất $\left\lceil \frac{7}{3} \right\rceil = 3$ lần trở lên nên thỏa mãn ràng buộc về chuỗi đẹp. Như vậy, ứng với mỗi vị trí $\ell = i$, ta sẽ kiểm tra trực tiếp xem có bao nhiêu chuỗi con ứng với $r + 2 \le r \le i + 5$ là thỏa mãn, còn với $r = i + 6$ trở đi thì chắc chắn là đẹp nên sẽ lấy hết các chuỗi con này.

Độ phức tạp: $O(n)$.

Cách khác: Ngoài ra, ta cũng có thể khai thác tiếp mảng cộng dồn ở subtask 2 bằng việc thực hiện tìm kiếm nhị phân xem ứng với mỗi $\ell = i$ thì chỉ số r đầu tiên nào thỏa mãn một trong các ràng buộc $x_r - x_{\ell-1} \ge 3$, $y_r - y_{\ell-1} \ge 3$, $z_r - z_{\ell-1} \ge 3$ là được.

Độ phức tạp: $O(n \log n)$.

Câu 2. ĐƯỜNG ỐNG THOÁT NƯỚC

Có tất cả 28 test, mỗi test được 0.25 điểm với phân bố như ràng buộc ở đề bài.

Subtask 1: Có 25% số test ứng với 25% số điểm của bài thỏa mãn: $k = 1$.

Ta coi hệ thống đường ống thoát nước này là một graph đơn vô hướng $G = (V, E)$ có tất cả n đỉnh và m cạnh.

Yêu cầu của bài toán là: chọn ra không quá k đỉnh có cùng bậc sao cho số lượng đỉnh của G có đường đi đến đỉnh nào đó của k đỉnh được chọn là nhiều nhất, cho biết số lượng nhiều nhất này.

Trước hết, ta thực hiện duyệt DFS qua các đỉnh để chia graph thành các thành phần liên thông trong độ phức tạp $O(V + E)$, đánh dấu mỗi đỉnh với chỉ số của thành phần liên thông tương ứng.

Hơn nữa, ta cũng có thể cài đặt thích hợp để kiểm tra được mỗi thành phần liên thông sẽ bao gồm các đỉnh nào và bậc của chúng cụ thể là bao nhiêu. Nhận xét rằng nếu chọn từ một thành phần ra một đỉnh nào đó thì ta có thể lấy được hết số lượng đỉnh của thành phần liên thông đó cộng vào kết quả cần tìm.

Ở đây chỉ cần lấy ra $k = 1$ đỉnh nên đơn giản là chọn trong các thành phần liên thông ở trên ra thành phần có kích thước lớn nhất là được.

Độ phức tạp: $O(V + E)$.

Ghi chú: có thể dùng thuật toán DSU để tìm số thành phần liên thông của graph G đã cho với độ phức tạp $O(V \log V + E)$.

Subtask 2: Có 25% số test ứng với 25% số điểm của bài thỏa mãn: $1 \le n \le 100$.

Với giá trị k tùy ý, vấn đề là làm sao chọn được các đỉnh có cùng bậc và tổng kích thước các thành phần liên thông chứa chúng là lớn nhất. Rõ ràng các bậc của các đỉnh có thể dao động từ 0 đến $n-1$. Ta thực hiện duyệt qua các bậc d đó, ứng với mỗi bậc thì duyệt qua tiếp các thành phần liên thông xem có những thành phần nào chứa đỉnh có bậc d thì lưu thông tin kích thước của nó lại thành một danh sách $size[d]$.

Ta sắp xếp danh sách $size[d]$ này lại và chọn ra k giá trị lớn nhất trong đó rồi lấy tổng của chúng, đây là kết quả lớn nhất ứng với việc chọn ra các đỉnh bậc d. Đáp số của bài toán sẽ là giá trị lớn nhất trong các kết quả trên khi kiểm tra hết các giá trị d.

Độ phức tạp: $O(V^2)$.

Subtask 3: Có 50% số test ứng với 50% số điểm của bài thỏa mãn các ràng buộc ban đầu của đề.

Ta thực hiện tương tự trên nhưng có cải tiến bước duyệt kết hợp với cách lưu trữ cẩn thận hơn. Với mỗi đỉnh, ta sẽ cài đặt thích hợp để lưu chỉ số của thành phần liên thông chứa nó. Duyệt qua các đỉnh $v \in \{1, 2, \ldots, n\}$, ta tính được bậc của v, giả sử là $d = \deg[v]$. Trước đó, ta tạo một set để lưu lại chỉ số của các thành phần liên thông mà trong đó có chứa một đỉnh có bậc bằng d, đặt là $size\_component[d]$ và có thể xây dựng được các set này thông qua mỗi lần duyệt trên (ta dùng set để tránh việc hai đỉnh cùng một thành phần bị tính lặp lại).

Ứng với thông tin trong mỗi $size\_component[d]$, ta sẽ đổi chỉ số của các thành phần liên thông thành kích thước của nó và lưu thành một mảng, sắp xếp mảng đó lại để tính tổng của k số lớn nhất là được, đây là kết quả lớn nhất ứng với việc chọn ra các đỉnh bậc d. Đáp số của bài toán sẽ là giá trị lớn nhất trong các kết quả trên khi kiểm tra hết các giá trị d.

Độ phức tạp: $O(V \log V)$.

Câu 3. HỌC MÁY

Có tất cả 50 test, mỗi test được 0.14 điểm với phân bố như ràng buộc ở đề bài.

Subtask 1: Có 20% số test ứng với 20% số điểm của bài thỏa mãn T tập tin đều có kích thước không vượt quá 2^8.

Với kích thước tập tin n nhỏ, ta có thể dùng quy hoạch động. Gọi $dp[i]$ là tổng nhỏ nhất cần tìm ứng với số i thì $dp[1]=2$. Với mỗi số $i \in \{1,2,\ldots,n\}$, ta kiểm tra xem có thể biểu diễn nó ở dạng $i=j^b$ hay không, và nếu được thì số mũ b nhỏ nhất là bao nhiêu.

Ứng với mỗi i, ta duyệt $j=2,3,\ldots,\sqrt{i}$ và tiếp tục duyệt $b=2,3,\ldots,8$ để xem có đẳng thức $i=j^b$ hay không (chú ý rằng khi $i \le 2^8$ thì $b \le 8$) nên tốn chi phí $O(8n\sqrt{n})$.

Từ đó, ta sẽ có mảng $f[i]$ với $1 \le i \le n$, trong đó

- $f[i]=-1$ nếu i không thể biểu diễn thành dạng lũy thừa như yêu cầu.
- $f[i]=b$ với b là số mũ nhỏ nhất để có biểu diễn $i=j^b$.

Với mỗi $i=2,3,\ldots,n$, ta sẽ duyệt tất cả số $j=1,2,\ldots,i-1$ để cập nhật mảng $dp[i]$ như sau :

$$dp[i]=\min\{dp[i], dp[i-j]+f[j]\} \text{ với } f[j]>0.$$

Sau khi tính được xong mảng $dp[i]$, ta trả lời được đồng thời T truy vấn dễ dàng. Kết quả cần tìm cho tập tin kích thước n sẽ là $dp[n]$.

Độ phức tạp $O(n^2)$.

Subtask 2: Có 40% số test ứng với 40% số điểm của bài thỏa mãn: $2^8 < N \le 2^{16}$ với N là tổng kích thước các tập tin.

Ta cải tiến bước tính $f[i]$ ở trên như sau: nhận xét rằng khi $i \le 2^{16}$ thì số mũ b trong biểu diễn $i=j^b$ sẽ thỏa mãn $2 \le b \le 16$ nên ta kiểm tra xem i có phải là lũy thừa đúng bậc b hay không. Điều này có thể thực hiện được nhờ thư viện pow có sẵn, chi phí tính toán là $O(n \log n)$. Ta lưu hết các giá trị i ứng với số mũ $f[i] \ge 2$ này lại thành một mảng $nice\_pow$ để duyệt trên đó, có thể kiểm tra được kích thước mảng này không quá lớn, nếu đặt nó là k thì $k<350$.

Để cập nhật mảng $dp[i]$, vẫn dùng công thức trên nhưng thay vì duyệt tất cả số $j=1,2,\ldots,i-1$ thì ta chỉ cần duyệt trên mảng $nice\_pow$ với các số không vượt quá i là được.

Độ phức tạp $O(kn)$.

Subtask 3: Có 40% số test ứng với 40% số điểm của bài thỏa mãn: $2^{16} < N \le 10^9$ với N là tổng kích thước các tập tin.

Với giá trị n lớn, ta không thể duyệt vét cạn $i=2,3,\ldots,n$ để cập nhật mảng $dp[i]$ mà cần khảo sát sâu hơn tính chất của các số $b_1,b_2,\ldots,b_k$ trong tổng

$$n=a_1^{b_1}+a_2^{b_2}+\cdots+a_k^{b_k}.$$

Ý tưởng chính ở đây là dựa theo tính chất đã được gợi ý "ngầm" trong phần mô tả của đề bài. Ta thấy rằng chắc chắn số n sẽ luôn viết được thành tổng của không quá 4 bình phương nên đáp số của bài toán sẽ bị chặn trên bởi 8, đó là một nhận xét quan trọng để có thể chia trường hợp.

Chú ý thêm rằng ta không cần xét lũy thừa bậc chẵn lớn hơn 2 vì luôn có thể quy về bậc 2 bởi công thức $a^{2m} = (a^m)^2$ nên điểm khó chính là ở bậc lẻ. Ta kiểm tra lần lượt tính chất của n theo thứ tự sau (nếu thỏa mãn điều kiện ở bước nào thì dừng ngay ở bước đó):

(1) Nếu n là số chính phương, kết quả là 2. Độ phức tạp: $O(1)$.

(2) Nếu n là lập phương đúng, kết quả là 3. Độ phức tạp: $O(1)$.

(3) Nếu n có dạng $a^2 + b^2$, kết quả là 4. Để kiểm tra, ta vét cạn các số $a^2 \le n$ rồi kiểm tra $n - a^2$ có chính phương hay không. Độ phức tạp: $O(\sqrt{n})$.

(4) Nếu n có dạng $a^2 + b^3$, ta cũng vét cạn tương tự trên, kết quả là 5. Ngoài ra, nếu n có dạng c^5 thì cũng có kết quả bằng 5. Độ phức tạp chung cho hai trường hợp là $O(\sqrt[3]{n})$.

(5) Nếu n có dạng $a^3 + b^3$ thì kết quả là 6, ta vét cạn với độ phức tạp $O(\sqrt[3]{n})$. Thêm vào đó, nếu n có dạng $a^2 + b^2 + c^2$ thì cũng có kết quả là 6, vấn đề là làm sao kiểm tra được? Đây là bước khó vì về lý thuyết, nếu thực hiện hai vòng lặp for lồng nhau thì độ phức tạp sẽ là $O(n)$.

Ta xử lý khéo léo như sau: lúc này, n đã không là bình phương hoặc tổng 2 bình phương nữa, khi đó nó có thể biểu diễn thành tổng của 3 hoặc 4 bình phương. Ta dùng tính chất đã nêu ở đầu bài để kiểm tra xem n có phải ở dạng $4^k(8m+7)$ hay không trong chi phí $O(\log n)$.

Nếu nó không có dạng này thì nó sẽ biểu diễn được thành tổng ba bình phương như mong muốn. Độ phức tạp chung cho hai trường hợp là $O(\sqrt[3]{n})$.

(6) Nếu n có các dạng $a^2 + b^2 + c^3$, hoặc $a^2 + b^5$ hoặc c^7 thì đều có kết quả là 7. Độ phức tạp lớn nhất là $O(\sqrt[3]{n} \cdot \sqrt{n}) = O(n^{5/6})$ ứng với hai vòng lặp duyệt cho căn bậc hai và căn bậc ba lồng nhau, vẫn đủ để chạy trong thời gian cho phép với $n \le 10^9$ (chú ý rằng trên thực tế thì sẽ không cần đến $(10^9)^{5/6} \approx 3 \cdot 10^7$ vì vòng lặp thứ hai sẽ có có chi phí giảm dần theo vòng lặp đầu).

(7) Nếu n không có các dạng trên thì đáp số là 8.

Độ phức tạp: $O(n^{5/6})$.

--------- HẾT ---------