SỞ GIÁO DỤC VÀ ĐÀO TẠO

HÀ NỘI

ĐÁP ÁN
ĐỀ CHÍNH THỨC

KÌ THI CHỌN ĐỘI TUYỂN HSG THÀNH PHỐ

LỚP 12 THPT NĂM HỌC 2021 - 2022

Môn thi: **TIN HỌC**
Ngày thi: *12 tháng 01 năm 2022*
Thời gian làm bài: *180 phút*

Bài 1: Chia hết 9
Một số chia hết cho 9 khi và chỉ khi tổng các chữ số của số đấy chia hết cho 9.
Ta lần lượt thực hiện:

- Ta lần lượt duyệt qua từng chữ số của số đã cho theo thứ tự từ trái sang phải. Thử thay chữ số đấy với chữ số bất kỳ nhỏ hơn nó. Nếu số mới chia hết cho 9, trả về kết quả là số mới tìm được.
- Xét xem nếu số đưa vào chia hết cho 9 thì trả về số đưa vào.
- Duyệt qua từng chữ số theo thứ tự từ phải sang trái. Thử thay chữ số đấy với chữ số bất kỳ lớn hơn nó. Nếu số mới chia hết cho 9, thì trả về số này.

Bài 2: Xếp phòng họp

- **Subtask 1, 2:** Duyệt từng người và kiểm tra từng người khác xem có bao nhiêu người có đoạn giao nhau.
- **Subtask 3:** Xét một đoạn $[a_i, b_i]$ ta sẽ cần đếm xem có bao nhiêu đoạn $[a_j, b_j]$ giao với đoạn này. Đầu tiên ta chuẩn bị 2 mảng prefix sum $A[p]$ là tổng số đoạn có điểm xuất phát $a_i \leq p$, và $B[p]$ là tổng số đoạn có điểm kết thúc $b_i \leq p$. Ta có hai trường hợp giao như sau:
    - $a_j < a_i$ và $b_j \geq a_i$: Số đoạn j giao với i là số đoạn có $a_j < a_i$ và loại đi những đoạn có $b_j < a_i$, do đó số lượng j bằng $A[a_i - 1] - B[a_i - 1]$.
    - $a_i \leq a_j$ và $a_j \leq b_i$: Số đoạn j giao thỏa mãn sẽ có điểm xuất phát nằm trong $[a_i, b_i]$ nên sẽ bằng $A[b_i] - A[a_i - 1]$.

  Tổng hợp hai trường hợp trên ta sẽ có được kết quả là số đoạn giao với đoạn i, chú ý rằng ta đã đếm cả chính đoạn i.
- **Subtask 4:** Sử dụng cách tính như subtask 3, tuy nhiên vì m lớn do đó ta xây dựng A và B là danh sách các điểm, từ đó sử dụng lower_bound để đếm như công thức trong subtask 3.

Bài 3: Liên hoan
Bài toán này ta sẽ sử dụng thuật toán tham lam như sau: Chừng nào còn chưa đủ k món khác nhau thì ta sẽ duyệt từng người có món ăn trùng với người khác và thay đổi bằng món mới có giá trị nhỏ nhất mà lớn hơn giá trị của người đó mà chưa có ai chọn. Người được chọn sẽ là người có chi phí cần sử dụng là nhỏ nhất.

- **Subtask 1:** Vì tất cả a_i bằng nhau, nên cách tối ưu nhất sẽ là chọn $k - 1$ món ăn có giá trị nhỏ nhất mà lớn hơn giá trị của món a_i để gán cho $k - 1$ người là ta sẽ có đủ k món ăn khác nhau. Do đó ta chỉ cần sort sau đó lấy lần lượt $k - 1$ món ăn có giá trị lớn hơn giá trị của món ăn a_i.
- **Subtask 2:** Với $k = n$, ta sort các món ăn theo giá trị rồi sử dụng two pointer để chọn món cần đổi cho mỗi người có món ăn trùng với người khác.
- **Subtask 3:** Với $n, m \leq 15$ Subtask này cho phép sử dụng quy hoạch động trạng thái khi không nhận ra thuật tham lam trên.
- **Subtask 4, 5:** Dựa vào thuật tham lam mô tả ban đầu, để đảm bảo được rằng chọn được các chênh lệch nhỏ nhất thì ta sẽ duyệt từ những món ăn có giá trị lớn nhất đến các món ăn có giá trị nhỏ nhất, với mỗi món ăn có nhiều người cùng lựa chọn ta sẽ tìm món ăn chưa có ai chọn gần nhất để đổi, việc tìm kiếm món chưa được chọn hiệu quả sẽ cần sử dụng stack hoặc dsu để tìm kiếm. Lưu ý với subtask 5 khi ta sắp xếp thì những món ăn có giá trị gần nhau có thể nằm gần nhau, do đó phải xử lý thêm chút đoạn tìm kiếm sao cho món ăn được chọn có giá trị lớn hơn.

Bài 4: Tìm kho báu

- **Subtask 1:** Với $k = 1$ đây là bài toán tìm đường đi dài nhất trên cây, ta dfs 2 lần sẽ tìm được kết quả
- **Subtask 2:** $n \leq 20$, duyệt thử toàn bộ, duyệt đỉnh xuất phát, sau đó thử đi đến từng đỉnh, mỗi lần cập nhật lại số lượng bánh tại mỗi đỉnh

- **Subtask 3:** $n \leq 300, k = 2$, ta sẽ cần duyệt vị trí xuất phát và coi đó là gốc của cây, sau đó sử dụng quy hoạch động trên cây để tìm kết quả tối ưu với đỉnh xuất phát đang xét:
    - gọi $dp[u][0]$ là đường đi dài nhất từ u đi xuống các đỉnh con của u và không quay lại đỉnh cha của u
    - gọi $dp[u][1]$ là đường đi dài nhất từ u xuống các đỉnh con của u sau đó quay ngược trở lại đỉnh cha của u
    - Vì $k = 2$ nên nếu từ u di chuyển rồi quay lại cha thì u chỉ có thể đi xuống một đỉnh con duy nhất do đó $dp[u][1] = \max(dp[v][1]) + 2$ với v là đỉnh con trực tiếp của u. Còn với trường hợp u không quay trở lại cha thì ta sẽ chọn một đỉnh v_1 để đi xuống rồi quay trở lại sau đó đi xuống v_2 và không quay trở lại nữa, do đó $dp[u][0] = \max(dp[v_1][1] + dp[v_2][0]) + 3$
    - Vì phải duyệt v_1, v_2 do đó độ phức tạp cho mỗi lần dp sẽ là $O(n^2)$ => độ phức tạp tổng thể sẽ là $O(n^3)$
- **Subtask 4:** $n \leq 1000$ tương tự như subtask 3 ta cũng quy hoạch động như trên, nhưng vì k có thể lớn hơn 2 do đó ta sẽ có
    - $dp[u][1] = \max(dp[v_1][1] + dp[v_2][1] + \cdots + dp[v_{k-1}][1]) + 2(k-1)$ ta sẽ cần chọn $v_1, v_2, \dots v_{k-1}$ sao cho biểu thức là lớn nhất, đây là bài toán cho x số f_icần chọn ra tối đa $k-1$ số sao cho tổng là lớn nhất, thì ta chỉ cần lấy $k-1$ số lớn nhất là xong.
    - Tương tự $dp[u][0] = \max(dp[v_1][1] + dp[v_2][1] + \cdots + dp[v_{k-1}][1] + dp[v_k][0]) + 2k - 1)$ để tính được biểu thức này, ta sẽ sắp xếp theo $dp[v][1]$ và dùng mảng cộng dồn để duyệt lần lượt v_k và tính được tổng lớn nhất của $k-1$ đỉnh còn lại.
    - Như vậy ta sẽ có độ phức tạp là $O(n^2 logn)$
- **Subtask 5:** $n \leq 10^5$, vẫn ý tưởng như subtask 4, tuy nhiên ta sẽ không quy hoạch động lại hoàn toàn với từng đỉnh gốc. Ban đầu ta coi đỉnh 1 là gốc, và quy hoạch động từ đỉnh 1 này, sau đó ta sẽ dfs lần thứ 2 để chuyển dần gốc thành các đỉnh khác. Giả sử đỉnh 2 là đỉnh con trực tiếp của đỉnh 1, khi coi 2 là gốc ta sẽ cần tính $dp[1][0]$ và $dp[1][1]$ khi coi 2 là gốc, để làm điều này, ta chỉ cần xây dựng $m[p][u][0]$ = $dp[u][0]$ khi coi p là cha của u , tương tự $m[p][u][1] = dp[u][1]$ khi coi p là cha của u, số lượng trạng thái cần lưu trữ chính là số cạnh của cây, do đó ta dùng map để lưu trữ các thông tin này, và dựa vào các thông tin này ta sẽ dễ dàng tính lại được dp khi dịch chuyển gốc. Độ phức tạp sẽ là $O(nlogn)$.

Bài 5: Dãy liên tiếp

Cho dãy số nguyên dương có n phần tử $a_1, a_2, \dots, a_n$. Xét truy vấn như sau:

- Xét dãy con $b = a_l, a_{l+1}, \dots, a_r$, ta xây dựng dãy c bằng cách xét tất cả các dãy con liên tiếp của dãy b, với mỗi dãy con ta thêm tất cả các phần tử vào c. Ví dụ dãy $b = \{1, 2, 1, 3\}$ ta sẽ có dãy $c = \{1,2,1,3,1,2,2,1,1,3,1,2,1,2,1,3,1,2,1,3\}$
- Sắp xếp dãy c không giảm và đưa ra phần tử nhỏ thứ k, ví dụ với dãy c trên ta sắp xếp thành {1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,3,3,3,3} và phần tử thứ 11 là 2.

Cho dãy a_i và Q truy vấn như trên, hãy đưa ra kết quả phần tử nhỏ thứ k của dãy c trong mỗi truy vấn.

- **Subtask 1:** $n, Q \leq 100$: mỗi truy vấn dựng lại dãy và sort O(n^2m)
- **Subtask 2:** $n, a_i \leq 100$: dựng ma trận $cnt[i][j][x]$ là số lượng x xuất hiện trong dãy số dựng từ i đến j, với số x ở vị trí k ta có số lượng xuất hiện trong dãy dựng từ i đến j là $(j - k) \times (k - i + 1)$, nên dựng mảng cnt mất n^3. Với mỗi truy vấn ra đi duyệt lần lượt từng màu và đếm xem số lượng đã vượt k hay chưa, độ phức tạp O(QX) với $X = max(a_i)$.
- **Subtask 3:** $n, a_i \leq 5000$ tìm kiếm nhị phân kết quả, với mỗi giá trị mid đi đếm số lượng các giá trị $\leq mid$ xuất hiện trong dãy dựng từ đoạn i đến j mất một vòng for từ i đến j. Thuật toán sẽ là O($QnlogX$)
- **Subtask 4:** $n, Q \leq 10^5, a_i \leq 50$: Sử dụng mảng cộng dồn cho mỗi giá trị a_i, ta có thể tính được mỗi giá trị x xuất hiện bao nhiêu lần trong dãy từ i đến j với O(1), thuật toán sẽ là O($(n+Q)X$).
- **Subtask 5:** $n, Q, a_i \leq 10^5$: sử dụng tìm kiếm nhị phân song song để làm offline tính kết quả cho các truy vấn, sử dụng cây segment tree hoặc fenwick tree để tính được số lượng số $\leq mid$ trên một đoạn. Độ phức tạo O($QlognlogX$)
- **Subtask 6:** $n, Q, a_i \leq 5 \cdot 10^5$: Dùng Persistent Tree độ phức tạp chỉ còn O($(n+Q)logX$).

- - - - - - - - - Hết - - - - - - - - -