SOLUTION 20/12 - C

Bài 1: SWAP.

Sub1: Duyệt dãy số từ đầu tới cuối, nếu tìm thấy $a_{i+1} > a_i$ thì đổi chỗ 2 phần tử đó, đếm số lần đổi cho đến khi bằng *k* thì dừng.

Sub2: Xét dãy các phần tử từ phần tử thứ *i* tới phần tử thứ *i+k*, tìm phần tử lớn nhất trong dãy đó, tiến hành đổi chỗ để đưa a_{max} về về vị trí thứ *i*. Cứ làm như thế cho đến khi hết k lần đổi chỗ thì dừng.

Sub3: Tương tự *sub2*, nhưng thay vì đưa phần tử *max* trong đoạn từ *i* đến *i+k* về vị trí *i* thì ta đẩy phần tử *max* vào hàng đợi, xóa phần tử *max* ra khỏi dãy và cập nhật lại cây Segment Tree (IT).

Xây dựng cây Segment Tree, mỗi nút lưu hai thuộc tính (giá trị, vị trí của giá trị đó trong dãy ban đầu).

Bài 2: SEQPART.

*QHĐ.

Gọi F(g,i) là chi phí nhỏ nhất nếu ta phân hoạch i phần tử đầu tiên thành g nhóm, khi đó kết quả bài toán sẽ là F(G,L).

Để tìm công thức truy hồi cho hàm F(g,i), ta sẽ quan tâm đến nhóm cuối cùng. Coi phần tử 0 là phần tử cầm canh ở trước phần tử thứ nhất, thì người cuối cùng không thuộc nhóm cuối có chỉ số trong đoạn [0,i][0,i]. Giả sử đó là người với chỉ số k, thì chi phí của cách phân hoạch sẽ là F(g−1,k)+Cost(k+1,i) với Cost(i,j) là chi phí nếu phân j−i+1 người có chỉ số [i,j]vào một nhóm. Như vậy:

F(g,i)=min(F(g−1,k)+Cost(k+1,l))với 0<=k<=i.

Chú ý là công thức này chỉ được áp dụng với g>1, nếu g=1,F(1,i)=Cost(1,i)đây là trường hợp cơ sở.

Chú ý là ta sử dụng mảng sum[] tiền xử lí O(L) để có thể truy vấn tổng một đoạn (dùng ở hàm cost()) trong O(1). Như vậy độ phức tạp của thuật toán này là O(G∗L∗L).

➢ Thuật toán tối ưu hơn

Gọi P(g,i) là k nhỏ nhất để cực tiểu hóa F(g,i), nói cách khác P(g,i])là k nhỏ nhất mà F(g,i)=F(g−1,k)+Cost(k+1,i).

Tính chất quan trọng để có thể tối ưu thuật toán trên là dựa vào tính đơn điệu của P(g,i), cụ thể: $P(g,0) \le P(g,1) \le P(g,2) \le \cdots \le P(g,L-1) \le P(g,L)$.

✓ Chia để trị

Để ý rằng để tính F(g,i), ta chỉ cần quan tâm tới hàng trước F(g−1) của ma trận: F(g−1,0),F(g−1,1),...,F(g−1,L).

Như vậy, ta có thể tính hàng F(g) theo thứ tự bất kỳ.

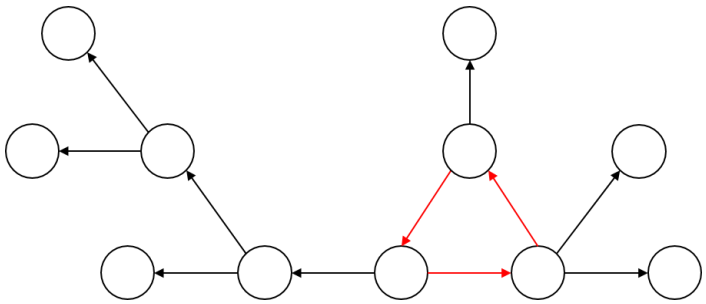
Ý tưởng là với hàng g, trước hết ta tính F(g,mid)F(g,mid) và P(g,mid) với mid=L/2, sau đó sử dụng tính chất nêu trên P(g,i)≤P(g,mid) với i<mid và P(g,i)≥P(g,mid) với i>mid để đi gọi đệ quy đi tính hai nửa còn lại.

Bài 3:

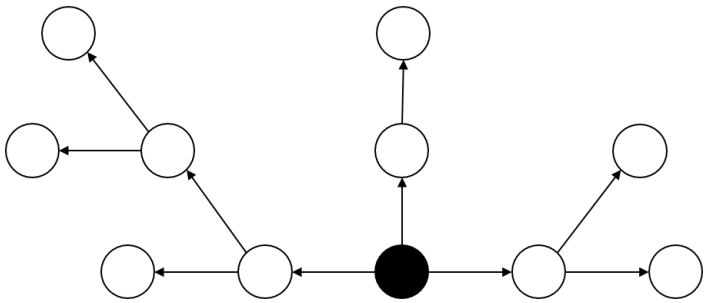
Trước hết, ta xét bài toán tìm cách mở rương tối ưu với đầy đủ K chìa khoá. Ta cần xây dựng một đồ thị N đỉnh, với đỉnh i đại diện cho chiếc rương thứ i. Với chìa khóa thứ i, ta thêm cạnh giữa đỉnh A_i và đỉnh B_i. Ta xem việc dùng chìa khóa thứ i mở rương A_i hay B_i tương đương với việc định chiều cạnh i. Khi đó, chiếc rương thứ i sẽ được mở nếu ít nhất một cạnh đi vào đỉnh i.

Khi đó, xét từng thành phần liên thông (TPLT):

- Nếu TPLT có tồn tại chu trình $p_1,p_2,\ldots,p_k$, thì ta có thể định hướng các cạnh trong chu trình theo chiều $p_1 \to p_2 \to p_3 \to \ldots \to p_k \to p_1$. Sau đó, ta dùng một thuật toán DFS, xuất phát từ các đỉnh trong chu trình, và khi ta thăm đỉnh v từ đỉnh u thì định hướng cạnh $u \to v$. Khi đó, các đỉnh trong TPLT đều có ít nhất một cạnh đi vào.



- Nếu TPLT hiện tại không có chu tình, ta xuất phát từ đỉnh đại diện cho kho báu có giá trị thấp nhất trong TPLT (gọi đỉnh này là r) và thực hiện thuật toán DFS tương tự như trên. Khi đó, các đỉnh trong TPLT trừ r đều có ít nhất một cạnh đi vào.



Do đó, đáp án sẽ là tổng giá trị các kho báu trừ đi tổng giá trị nhỏ nhất của các kho báu trong mỗi TPLT.

Ta trở lại với bài toán ban đầu. Thao tác xóa chìa khóa tương ứng với việc xóa cạnh trong đồ thị. Ta sẽ thực hiện các truy vấn theo thứ tự ngược lại: ban đầu, đồ thị không có cạnh nào, và mỗi truy vấn sẽ thêm một cạnh vào đồ thị.

Khi đó, ta có thể giải quyết bài toán bằng disjoint-set union (DSU). Với mỗi thành phần liên thông, ta sẽ lưu thêm thông tin về đỉnh có giá trị nhỏ nhất trong TPLT, và TPLT đó có chu trình hay không. Các bạn có thể xem bài giải mẫu để hiểu rõ hơn chi tiết cài đặt.

Độ phức tạp: $O(N + K \log K)$ hoặc $O(N + K\alpha(K))$ (với $\alpha(K)$ là nghịch đảo hàm Ackermann, có thể xem như hằng số), tùy vào cách cài đặt DSU.