HƯỚNG DẪN GIẢI ĐỀ THI CHẶNG 2
CODE TOUR VTF 2020 – O(N)

A. AUTO CHESS - HP

Solution:

C++	https://ideone.com/EkCZP9

Tóm tắt đề:

Cho kết quả các trận đấu của N round đấu, mỗi round có 8 trận đấu bao gồm thông tin về người chơi phòng thủ, người chơi chơi tấn công và sức mạnh còn lại của họ sau trận đấu, tương ứng với 8 người chơi. Ban đầu mỗi người có X điểm sinh lực, khi bắt đầu mỗi trận đấu sẽ có 100 điểm sức mạnh, người chơi phòng thủ thua sẽ bị trừ điểm sinh lực bằng số điểm sức mạnh còn lại của người chơi tấn công. Hãy đưa ra bảng xếp hạng của 8 người chơi sau round thứ K theo điểm sinh lực còn lại, nếu hai người cùng số điểm sinh lực thì người chơi có chỉ số nhỏ hơn sẽ xếp hạng cao hơn.

Input

- Dòng đầu tiên, 3 số nguyên dương:
  - X, sinh lực ban đầu của 8 người chơi $(1 \leq X \leq 100)$.
  - N, số lượng hiệp đấu $(1 \leq N \leq 60)$.
  - K, hiệp đấu cần xem bảng xếp hạng $(1 \leq K \leq N \leq 60)$.
- $N * 8$ dòng tiếp theo, kết quả mỗi trận đấu, 4 con số nguyên không âm:
  - D, Defender, đại diện cho người chơi phòng thủ $(1 \leq D \leq 8)$.
  - RD, Remain of Defender, số điểm còn lại của người chơi phòng thủ sau trận đấu $(0 \leq RD \leq 100)$.
  - I, Invader, đại diện cho người chơi tấn công $(1 \leq I \leq 8)$.
  - RI, Remain of Invader, số điểm còn lại của người chơi tấn công sau trận đấu $(0 \leq RI \leq 100)$.

Big-O & VNG

Lưu ý, thông tin về mỗi trận đấu đảm bảo sẽ có ít nhất một người thua (tức là $RD = 0$ hoặc $RI = 0$), và tới trận đấu thứ K, chưa có người chơi nào bại trận.

Output

8 dòng, mỗi dòng gồm 2 con số, cách nhau 1 khoảng trắng, gồm chỉ số đại diện cho người chơi và sinh lực của người chơi đó ở hiệp thứ K, sắp xếp giảm dần theo sinh lực.

Ví dụ

100 1 1 7 0 8 4 1 0 2 1 4 1 5 0 2 0 3 2 3 0 4 3 6 1 7 0 5 0 6 0 8 0 1 1	4 100 5 100 6 100 1 99 8 99 2 98 3 97 7 96

Big-O & VNG

Giải thích ví dụ

Thấy rằng người chơi số 4, 6 đã phòng thủ thắng lợi, và người chơi số 5 đã hòa, nên điểm của ba người chơi này không đổi. Với những người chơi còn lại (người chơi số 1, 2, 3, 7, 8), đã phòng thủ thất bại và bị trừ đi số điểm tương ứng.

Hướng dẫn giải:

Trong bài toán này, thực chất chúng ta không cần phải nhập hết toàn bộ N vòng đấu đấu. Có thể xử lý online. Ban đầu, tạo một mảng $lifePoint[i] = X, i\ [1..8]$ lưu lại số điểm sinh lực còn lại của người chơi thứ i.

Với mỗi trận đấu, ta nhập các giá trị D, RD, I, RI. Nếu $RD = 0$, ta giảm sinh lực của người chơi D đi một lượng RI $(lifePoint[D] -= RI)$

Khi đã nhập và xử lý đủ số liệu cho K vòng đấu, ta có mảng $lifePoint$ hiện đang lưu trữ số điểm sinh lực của từng người chơi. Lúc này ta chỉ cần sắp xếp lại theo yêu cầu đề bài (giảm dần theo điểm sinh lực, nếu hai người chơi có cùng điểm sinh lực thì xếp tăng dần theo chỉ số), in kết quả và thoát chương trình.

Độ phức tạp: $O(K)$với K là trận cuối cùng trước khi đưa ra kết quả.

Big-O & VNG

B. STOCKS ARE CRASHING

Solution:

C++	https://ideone.com/88btd9
Python	https://ideone.com/lcHfDr

Tóm tắt đề:

Bạn được giao quản lý một tài khoản gồm n cổ phiếu, trong đó mỗi cổ phiếu có một giá trị a_i. Cổ phiếu sẽ phải trải qua q ngày trên sàn giao dịch, trong đó tại ngày j thì 1 trong 3 sự kiện sau có thể xảy ra:

- Sự kiện 1: Toàn bộ cổ phiếu từ vị trí u đến v rớt giá xuống còn 0.
- Sự kiện 2: Ông chủ cần biết số tiền có được nếu bán số cổ phiếu từ u đến v.
- Sự kiện 3: Các cổ phiếu từ vị trí u đến v cùng tăng giá 1 giá trị d.

Input

- Dòng đầu tiên chứa hai số n ($1 \le n \le 10^5$) và q ($1 \le q \le 10^5$) - số cổ phiếu và số ngày.
- Dòng thứ hai chứa n số nguyên - a_i ($1 \le i \le n$, $0 \le a_i \le 10^9$) là giá trị ban đầu của cổ phiếu thứ i.
- Trong q dòng tiếp theo, dòng thứ j chứa thông tin vè sự kiện thứ j:
  - Nếu dòng có 3 số thì sự kiện này thuộc dạng 1 hoặc 2. Số đầu tiên là loại của sự kiện (1 hoặc 2). Số thứ hai là giá trị u. Số cuối cùng là giá trị v.
  - Nếu dòng có 4 số thì sự kiện này thuộc dạng 3. Số đầu tiên là số 3. Ba số còn lại lần lượt là u, v và d.
- Điều kiện: $1 \le u, v \le n$, $0 \le d \le 10^6$.

Output

Với mỗi thao tác thuộc loại 2 thì in ra tổng giá trị của số cổ phiếu từ u đến v.

6 6 1 2 3 4 5 6 2 3 5 2 1 6	12 21 9 0

Big-O & VNG

3 3 4 1 2 3 4 1 3 4 2 3 4	

Giải thích:

- Ban đầu dãy n cổ phiếu có giá trị như sau: 1 2 3 4 5 6
- Sau mỗi ngày, giá trị của cổ phiếu thay đổi như sau:
  - Ngày 1: Không cổ phiếu nào đổi giá. In ra tổng giá trị từ 3 đến 5: 3 + 4 + 5 = 12. - Ngày 2: Không cổ phiếu nào đổi giá. In ra 1 + 2 + 3 + 4 + 5 + 6 = 21.
  - Ngày 3: 1 2 4 5 5 6.
  - Ngày 4: Không cổ phiếu nào đổi giá. In ra 4 + 5 = 9.
  - Ngày 5: 1 2 0 0 5 6
  - Ngày 6: Không cổ phiếu nào đổi giá. In ra 0 + 0 = 0.

Hướng dẫn giải:

Cách đầu tiên và đơn giản nhất để giải quyết bài này là với mỗi truy vấn thì ta duyệt qua các phần từ u đến v để xử lí. Tuy nhiên, với cách làm này thì thời gian chạy của chương trình sẽ vượt quá giới hạn thời gian cho phép (độ phức tạp O(q * n)). Vì vậy, để giải quyết bài này, chúng ta sẽ cần sử dụng một cấu trúc dữ liệu là Segment Tree (kết hợp với Lazy Propagation) để giải quyết vấn đề này (độ phức tạp O(q * log(n))).

Hai loại truy vấn 2 và 3 là hai thao tác thường thấy ở Segment Tree, tuy nhiên truy vấn 1 sẽ ít gặp hơn. Với thao tác 1, ta có thể xử lí như sau:

- Tạo thêm 1 biến ở từng node trong Segment Tree để đánh dấu lại là đoạn trên mảng được quản lí bởi node đó đã từng bị xóa (trở về 0).
- Trong quá trình thực hiện Lazy Propagation (lưu ý các thao tác cập nhật Lazy Propagation sẽ diễn ra trong phần đầu của từng thao tác 1, 2, 3 để đảm bảo giá trị các thao tác nhận được là mới nhất):
  - Nếu như tại một node được đánh dấu đã bị xóa thì ta sẽ thực hiện việc đưa giá trị tổng và giá trị lazy của mỗi node con về 0 và đánh dấu 2 node này đã được xóa, đồng thời xóa đánh dấu của nó hiện tại (để không diễn ra trường hợp node đó bị xóa mãi mãi, không cập nhật lại được). - Sau khi xóa như bước trên thì ta mới kiểm tra đến giá trị lazy của node hiện tại, nếu node hiện tại có gía trị lazy khác 0 thì ta sẽ cập nhật giá trị lazy của hai node con.

Độ phức tạp: O(q * log(n)) với q là số lượng truy vấn, n là số lượng phần tử của mảng.

Big-O & VNG

C. ADVENTURE TIME

Solution:

C++	https://ideone.com/N2WTxe
Java	https://ideone.com/VDpj8X
Python	https://ideone.com/2rsUCI

Tóm tắt đề:

Bạn là trưởng nhóm khảo sát trên 1 dòng sông có N vị trí quan trọng trong đó K vị trí cần được khảo sát. Nhiệm vụ của bạn là tìm số nhân viên ít nhất để mỗi người bắt đầu đi từ nguồn đi tới các vị trí cần thiết mà mỗi người chỉ có thể đi từ nguồn tới 1 cuối, không thể quay lại các vị trí đã đi qua trước đó.

Input:

Dòng đầu gồm 1 số nguyên N ($1 \le N \le 5 \times 10^4$) tương ứng là số vị trí trên sông.

Dòng thứ hai gồm N số nguyên $p_1, p_2, ..., p_N$, trong đó đỉnh cha của đỉnh ***i*** là đỉnh $\boldsymbol{p_i}$. Nếu đỉnh i là đỉnh gốc, $p_i = 0$ cũng là nguồn của sông, là điểm xuất phát của các nhân viên. Dữ liệu vào đảm bảo đồ thị đã cho là một cây.

Dòng tiếp theo gồm 1 số nguyên k ($1 \le k \le 10^3$) tương ứng với số địa điểm cần khảo sát.

Dòng tiếp theo gồm k số nguyên liên tiếp là các vị trí cần khảo sát.

Output:

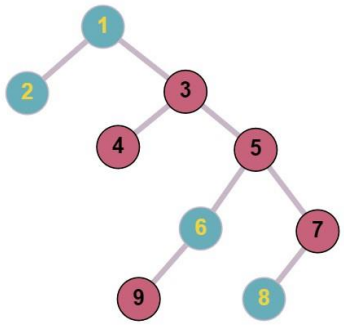
In ra một dòng gồm 1 số nguyên duy nhất là số nhân viên ít nhất để khảo sát tất cả các địa điểm.

Ví dụ:

9 0 1 1 3 3 5 5 7 6 5 4 3 5 9 7	3

Giải thích:

Big-O & VNG



Với ví dụ thứ nhất ta có thể phân bố như sau:

- Từ vị trí 1 - nguồn, với nhân viên thứ nhất có thể khảo sát tại vị trí 3 và 4.
- Với nhân viên thứ 2, có thể khảo sát tại vị trí 5 và 7.
- Với nhân viên thứ 3, có thể khảo sát tại vị trí 9

Vậy chỉ với 3 nhân viên ta có thể hoàn thành việc khảo sát.

Hướng dẫn giải:

Phân tích: Ta nhận thấy tại 1 node bất kì trên cây, số nhân viên ít nhất để thực hiện tại các vị trí dưới node đó là tổng số nhân viên ít nhất của từng nhánh cây con. Thậm chí nếu node hiện tại là 1 node cần thực hiện công việc thì chỉ cần 1 nhân viên thực hiện và di chuyển tiếp xuống 1 trong các nhánh cây con cần thực hiện công việc.

Vậy cuối cùng ta chỉ cần tìm tổng số nhân viên ít nhất của mỗi nhánh cây con và với các node cần khảo sát nhưng các nhánh con không cần nhân viên khảo sát thì trả về kết quả là 1 (là chỉ cần 1 nhân viên tới vị trí này). Từ nhận xét trên ta có cách giải như sau:

- Ta đọc dữ liệu đầu vào với dạng danh sách đỉnh kề và tạo 1 mảng là spot với kích thước mảng bằng N, với spot[i] = 1 là đỉnh cần khảo sát và spot[i] = 0 là đỉnh không cần khảo sát.
- Tại đỉnh nguồn là đỉnh **u** có cha là 0, ta bắt đầu duyệt DFS với kết quả trả về là số nhân viên ít nhất để tới các vị trí cần khảo sát trên cây có đỉnh là **u**.
    - Trong thuật toán DFS, ta khởi tạo **cnt** là số nhân viên cần thiết để đi tới các vị trí cần khảo sát ở các nhánh cây con.

Big-D & VNG

    - Ta duyệt qua từng cây con có đỉnh gốc là **v**, ta thực hiện phép tính: **cnt = cnt + DFS(v)**
    - Sau khi tính tổng các nhánh, ta so sánh nếu không cần nhân viên nào **cnt = 0**, nghĩa là không có đỉnh cần khảo sát ở các cây con thì ta sẽ xét đỉnh đang đứng:
    - Nếu đỉnh đang đứng là vị trí cần khảo sát thì trả kết quả đệ quy DFS này là 1, ngược lại thì là 0.
- Vậy kết quả bài toán sẽ là DFS(u) với u là đỉnh gốc của cây.

Độ phức tạp: **O(N)** với N là số đỉnh trên cây mà đề cho.

Big-O & VNG

D. AUTO CHESS - UPGRADE

Solution:

C++	https://ideone.com/7yoJqW
Java	https://ideone.com/zf1iok
Python	https://ideone.com/Zy6yWN

Tóm tắt đề:

Độ khó của mỗi password được miêu tả bằng một con số là số lượng các âm c, o, v, i, d xuất hiện trong password đó. Cho N password bất kỳ, đếm xem có tối đa bao nhiêu cặp được tạo thành có độ khó bằng nhau, biết rằng mỗi password chỉ được ghép cặp đúng một lần.

Input

- Dòng đầu tiên chứa số nguyên dương N ($N \leq 10^5$).
- N dòng tiếp theo, mỗi dòng chứa một password gồm các chữ cái Latin thường, độ dài của mỗi password không quá 10^4 ký tự.
- Dữ liệu đảm bảo tổng độ dài của N xâu không vượt quá 10^7.

Output

In ra số lượng cặp password tối đa được tạo thành có độ khó bằng nhau.

5 covy wuhan corona diraxa lalala	2

Giải thích:

Với test ví dụ này, ta có:

- "covy" có độ khó là 3.
- "wuhan" có độ khó là 0.
- "corona" có độ khó là 3.
- "diraxa" có độ khó là 2.
- "lalala" có độ khó là 0.

Big-O & VNG

Vậy ta có tối đa 2 cặp có độ khó bằng nhau được tạo thành đó là "covy" và "corona", "wuhan" và "lalala".

Hướng dẫn giải:

Xây dựng hàm countDifficulty tính độ khó của từng password. Mỗi password có bao nhiêu ký tự c, o, v, i, d thì sẽ có độ khó tương ứng.

Bên cạnh đó, sử dụng mảng difficulty để lưu lại số lượng password có cùng độ khó. Từ đó tính được số cặp, do mỗi password chỉ được ghép cặp đúng 1 lần.

Độ phức tạp: O(N * |S|) với N là số lượng password, |S| là độ dài của password.

Big-O & VNG