COMPUTER SOCIETY OF INDIA (CSI)
COEP TECH STUDENT CHAPTER

PRESENTS

CODE QUEST 5.0

EDITIORIAL

SAM THE PRO GAMER

PROBLEM STATEMENT

Sam is a pro gamer and keeps trying new games and also recommends and rates the games based on his adventure experience about that particular game. He has recently purchased a new game "Zephyr's Labyrinth" which consists of completing challenges session wise. Each session consists of multiple levels which are to be completed in one go. Each level points to another succeeding level and levels which are not pointed to by any level are called starting levels.

Once a level is played, it is compulsory to immediately play its succeeding level and the same is true for all the succeeding levels. Any level other than the starting levels can be pointed to by one or more levels, but a level can be played only once (A level can belong to multiple sessions but can be played only in the session which Sam chooses to play before).

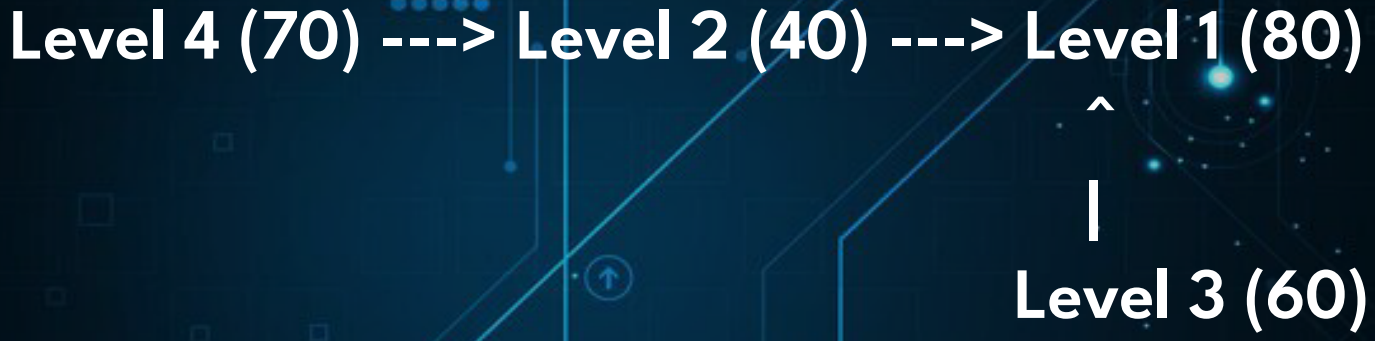
PROBLEM STATEMENT

The adventure associated with each level is given by an integer, and Sam's adventure for a session is defined as the maximum adventure of any level played in that session. For example, if Sam plays a session which has 3 levels of adventures 40,60,50 then Sam's adventure for that session is the maximum of (40,60,50) which is 60. Sam's adventure for the game is given by the sum of his adventures for all the sessions. Similarly if Sam plays a session with levels having adventures 20,50,90 and if the levels with adventures 50 and 90 are already played during some other session, then the adventure for this session is 20 (as 50 and 90 were played earlier and cannot be played again).

The task is to find the order of playing the sessions (starting levels) in a way that maximises Sam's adventure for the game.

PROBLEM STATEMENT

Example : If the game has 4 levels with adventures A1 = 80, A2 = 40, A3=60, A4 = 70. And Level 2 and 3 point to level 1, and level 4 points to level 2.



Here Level 3 and Level 4 are the starting levels (as they are not pointed to by any other level). Level 1 does not point to any other level.

Case 1) If Sam choses to play level 3 first then he has to play level 3 (60) and level 1 (80) in that session and the adventure for this session will be max(60,80) which is 80. In the next session Sam plays level 4 (70) and then level 2 (40) (Note

PROBLEM STATEMENT

that level 1 cannot be played again since it was already played in the first session where level 3 pointed to level 1). So the adventure for this session is max(70,40) which is 70. Hence the total adventure for this game is 80 + 70 = 150.

Case 2) If Sam choses to play level 4 first then he has to play level 4 (70), level 2 (40) and level 1 (80) in that session and the adventure for this session will be max(70,40,80) which is 80. In the next session Sam plays level 3 (60) only (Note that level 1 cannot be played again since it was already played in the first session where level 4 pointed to level 2 and level 2 pointed to level 1). So the adventure for this session is max(60) which is 60.
Hence the total adventure for this game is 80 + 60 = 140. Hence the maximum possible adventure for the game is 150 (Obtained in case 1).

INPUT FORMAT

The first line of the input gives the number of test cases, T. T test cases follow. Each test case starts with a line with a single integer N, the number of levels in the game.

The second line contains N integers A1,A2,...,AN where Ai is the adventure associated with the i-th level. The third line contains N integers P1,P2,...PN. If Pi=0, that means level i does not point to any other level. Otherwise, level i points to level Pi.

OUTPUT FORMAT & CONSTRAINTS

$1 <= A_i <= 10^9$

$1 <= N <= 10^5$

$1 <= T <= 100$

$0 <= P_i <= i-1$

For each test case, output a single integer representing the maximum adventure for the game which Sam can have by playing the starting levels in the game in the best possible order.

SAMPLE TESTCASES

(1)

SAMPLE INPUT 0 : 1

4

60 20 40 50

0 1 1 2

SAMPLE OUTPUT 0 : 110

(2)

SAMPLE INPUT 1 : 1

5

3 2 1 4 5

0 1 1 1 0

SAMPLE OUTPUT 1 : 14

SOLUTION

```cpp
#include <bits/stdc++.h>

int main() {
    using namespace std;
    ios_base::sync_with_stdio(false), cin.tie(nullptr);

    int T; cin >> T;
    for (int case_num = 1; case_num <= T; case_num ++) {

        int N; cin >> N;
        std::vector<int64_t> F(N);
        for (auto& f : F) cin >> f;
        std::vector<int> P(N);
        for (auto& p : P) { cin >> p; p--; }

        const int64_t INF = 1e18;
        int64_t ans = 0;
        std::vector<int64_t> ch_min(N, INF);
        for (int i = N-1; i >= 0; i--) {
            int64_t v = F[i];
            if (ch_min[i] != INF) {
                ans -= ch_min[i];
                v = std::max(v, ch_min[i]);
            }
            ans += v;
            if (P[i] != -1) {
                ch_min[P[i]] = std::min(ch_min[P[i]], v);
            }
        }

        cout <<  ans << '\n';
    }

    return 0;
}

```