B. 烘培大師 (ovenmasters)

Time limit: 2 seconds
Memory limit: 1024 MiB

你是「義大利頂級披薩烘焙大師」比賽的記者，這場活動剛結束，有 N 位義大利最厲害的披薩師傅在這裡一決高下。 每位師傅做了一份披薩，評審為這些披薩排了名，分別從 0 (最好) 到 $N-1$ (最差)；每位師傅的排名與他們的披薩排名相同。

比賽結束後，現在是披薩慶祝餐會時間。所有師傅都會參加，並帶著自己的披薩來參加餐會。 師傅們會陸續抵達（順序不一定是依照排名）；會場有 $M \le N$ 張桌子，編號從 0 到 $M-1$。 前 M 位抵達的師傅會按抵達順序將披薩放在 0 到 $M-1$ 號桌上； 剩下 $N-M$ 位師傅進場後，想吃比自己披薩好、但又不會「太好」的披薩（這樣才不會感到自卑）。 每次有師傅進場，他們會挑選目前桌上「比自己披薩好」且「排名最差」(即數字最大) 的披薩來吃。 他們會坐在那張桌子吃掉選中的披薩，然後把自己帶來的披薩留在桌上，給下一位師傅吃。 如果沒有適合的披薩 (因為桌上的披薩排名都比自己的差)，師傅就會失望地離開，並帶走自己的披薩。

以下範例展示了一個有 $M = 2$ 張桌子的餐會，師傅抵達的排名順序為：$1, 0, 3, 5, 4, 2$， 這對應到第一個範例的輸入與輸出。

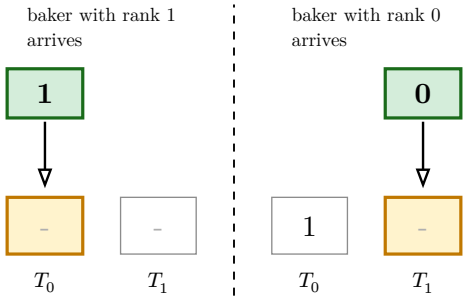

Figure 1: 前 $M = 2$ 位師傅依抵達順序將披薩放到空桌 (T_0, T_1) 上。

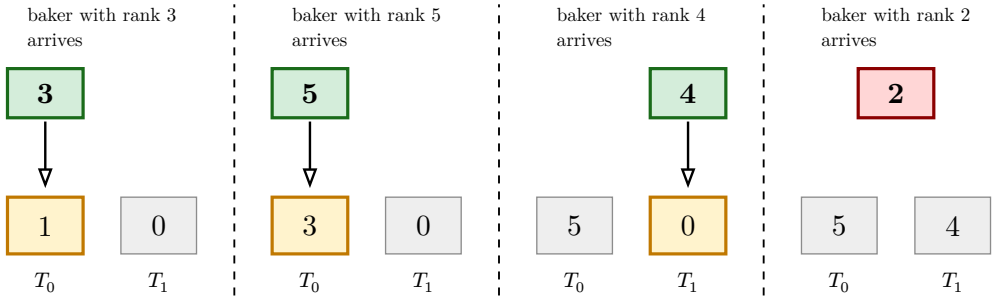

Figure 2: 當所有桌子都坐滿後，每位進場的師傅會走向桌上有「比自己好」且排名最差的披薩（箭頭所示）的桌子，吃掉那份披薩，並留下自己的披薩。如果沒有適合的披薩，該師傅就會失望地離開（無箭頭）。

你想要在報導中還原師傅抵達披薩慶祝餐會的順序；但不幸的是，你當時被美味的披薩分散了注意力，忘了記錄師傅抵達順序。幸運的是，在每張桌子上，你都可以看到該桌按順序供應的披薩托盤堆疊起來。

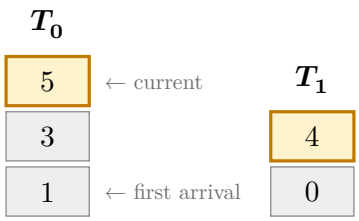

Figure 3: 對應第一個範例的托盤堆疊狀況：每個堆疊顯示了該桌的師傅抵達順序，從底部（最先抵達）到頂部（最後抵達）；每個堆疊頂部以塗色和粗框強調的托盤是餐會結束後留在該桌的披薩。

你想要利用這些資訊來還原師傅抵達的順序。你知道可能有多種可能的順序，因此，為了獲得滿分，你必須輸出「字典序最小」的合法順序。[1]

Input

第一行包含兩個整數 N 和 M，分別為師傅數量和桌子數量。

接下來有 M 行，每行描述一張桌子上的托盤堆疊。第 i 行以一個整數 T_i 開頭，代表桌子 i 上的托盤數量，隨後跟著 T_i 個整數 $b_{\{i,j\}}$，表示桌子 i 上供應的第 j 個披薩的排名。

Output

如果沒有符合條件的順序，輸出 `NO`。如果有可能的順序，則輸出 `YES`。在這種情況下，輸出第二行，包含 N 個整數 $a_0, a_1, ..., a_{N-1}$，即師傅以抵達順序的排名。如果存在多種這樣的排列，你必須輸出字典序最小的那一個。注意，部分正確的答案仍可獲得部分分數，詳見「Scoring」部分。

Constraints

- $1 \le M \le N \le 300\,000$.
- $0 \le b_{i,j} \le N - 1$.
- 所有 $b_{\{i,j\}}$ 皆相異.
- $1 \le T_i \le N$.

Scoring

你的程式會以幾組測試資料來進行測試，這些測資會被分成幾個子任務。要獲得某個子任務的分數，你必須正確解出該子任務中的所有測資。

⇨ 若只輸出了正確的第一行（`YES` 或 `NO`），可得 20% 分數。若輸出了正確的第一行且給出了一個**合法的**順序（不一定是字典序最小的），可額外獲得 20% 分數。要獲得剩下的 60% 分數，則必須在答案為 `YES` 時輸出字典序最小的合法順序。

- **Subtask 0 [0 points]**: 範例。
- **Subtask 1 [20 points]**: $M = 1$。
- **Subtask 2 [10 points]**: $M = 2$, $N \le 200$, 且所有 T_i 的總和為 N (換句話說，沒有師傅失望地離開)。
- **Subtask 3 [20 points]**: $M \le N \le 200$, 且所有 T_i 的總和為 N (換句話說，沒有師傅失望地離開)。
- **Subtask 4 [20 points]**: $M \le 10$。
- **Subtask 5 [30 points]**: 無額外限制。

[1] 序列 $a_0, a_1, \ldots, a_{n-1}$ 在字典序上小於序列 $b_0, b_1, \ldots, b_{n-1}$ 的條件是：存在一個索引 $0 \le t < n$，使得對於所有 $i < t$，都有 $a_i = b_i$，且 $a_t < b_t$。

Examples

stdin	stdout
6 2 3 1 3 5 2 0 4	YES 1 0 3 5 4 2
6 2 3 1 3 4 2 0 2	NO
4 2 2 0 3 2 1 2	NO
3 1 2 0 2	YES 0 2 1
8 1 8 7 6 5 4 3 2 1 0	NO
12 4 3 2 3 4 1 5 1 6 5 7 8 9 10 11	YES 2 5 6 7 0 1 3 4 8 9 10 11

Explanation

第一個範例的輸入與輸出對應到題目描述中的圖示。 具體來說，Figure. 2 中師傅抵達餐會的順序 $1, 0, 3, 5, 4, 2$ 正是字典序最小的合法抵達順序。

在第二個範例中，托盤堆疊順序不一致，因為不存在任何到達順序能讓排名第五的烘焙師感到失望而離開。因此，答案是「否」。

在第三個和第五個範例中，托盤堆疊順序也不一致（不存在任何到達順序能產生這種結果），所以答案也是「否」。

在第四個範例中（$N = 3$, $M = 1$），只有一種可能的抵達順序，即 $0, 2, 1$。

在第六個範例中（$N = 12$, $M = 4$），請注意數字 0 和 1 沒有出現在 $b_{\{i,j\}}$ 的數值中。這意味著在餐會期間，師傅 0 和 1 都曾失望地離開。範例輸出顯示了字典序最小的合法抵達順序。其他合法的抵達順序也存在，例如 $2, 5, 6, 7, 8, 1, 3, 4, 9, 10, 11, 0$。若輸出 YES 後接上像這樣的替代合法順序（而非字典序最小的順序），將被視為部分正確，可獲得 40% 的分數。