B. Preparing for the Contest

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Soon there will be held the world's largest programming contest, but the testing system still has m bugs. The contest organizer, a well-known university, has no choice but to attract university students to fix all the bugs. The university has n students able to perform such work. The students realize that they are the only hope of the organizers, so they don't want to work for free: the i-th student wants to get c_i 'passes' in his subjects (regardless of the volume of his work).

Bugs, like students, are not the same: every bug is characterized by complexity a_j, and every student has the level of his abilities b_i. Student i can fix a bug j only if the level of his abilities is not less than the complexity of the bug: $b_i \geq a_j$, and he does it in one day. Otherwise, the bug will have to be fixed by another student. Of course, no student can work on a few bugs in one day. All bugs are not dependent on each other, so they can be corrected in any order, and different students can work simultaneously.

The university wants to fix all the bugs as quickly as possible, but giving the students the total of not more than s passes. Determine which students to use for that and come up with the schedule of work saying which student should fix which bug.

Input

The first line contains three space-separated integers: n, m and s ($1 \leq n, m \leq 10^5$, $0 \leq s \leq 10^9$) — the number of students, the number of bugs in the system and the maximum number of passes the university is ready to give the students.

The next line contains m space-separated integers a_1, a_2, ..., a_m ($1 \leq a_i \leq 10^9$) — the bugs' complexities.

The next line contains n space-separated integers b_1, b_2, ..., b_n ($1 \leq b_i \leq 10^9$) — the levels of the students' abilities.

The next line contains n space-separated integers c_1, c_2, ..., c_n ($0 \leq c_i \leq 10^9$) — the numbers of the passes the students want to get for their help.

Output

If the university can't correct all bugs print "`NO`".

Otherwise, on the first line print "`YES`", and on the next line print m space-separated integers: the i-th of these numbers should equal the number of the student who corrects the i-th bug in the optimal answer. The bugs should be corrected as quickly as possible (you must spend the minimum number of days), and the total given passes mustn't exceed s. If there are multiple optimal answers, you can output any of them.

Examples

input

```
3 4 9
1 3 1 2
2 1 3
4 3 6
```

output

```
YES
2 3 2 3
```

input

```
3 4 10
2 3 1 2
```

```
2 1 3
4 3 6
```

output

```
YES
1 3 1 3
```

input

```
3 4 9
2 3 1 2
2 1 3
4 3 6
```

output

```
YES
3 3 2 3
```

input

```
3 4 5
1 3 1 2
2 1 3
5 3 6
```

output

```
NO
```

Note

Consider the first sample.

The third student (with level 3) must fix the 2nd and 4th bugs (complexities 3 and 2 correspondingly) and the second student (with level 1) must fix the 1st and 3rd bugs (their complexity also equals 1). Fixing each bug takes one day for each student, so it takes 2 days to fix all bugs (the students can work in parallel).

The second student wants 3 passes for his assistance, the third student wants 6 passes. It meets the university's capabilities as it is ready to give at most 9 passes.