Problem O
It's All About the Miles
Problem ID: miles

There are cheap airfares at the moment and you would like to go on a bunch of flights for no other reason than to collect frequent flyer miles. (This is known among people who like to do this sort of thing as a "mileage run.")

You have decided to fly in a big loop, visiting (the airports of) a number of cities in sequence before returning home. For instance, you might have decided to fly from Chicago to Detroit, then from Detroit to Omaha, then Pittsburgh, then finally back to Chicago.

You will never leave the airports; you will just transfer between flights. You have two rules: there is a minimum connection time, t_{min}, because you have to make it from the arrival gate to the departure gate of the next flight; and there is a maximum connection time, t_{max}, because you hate sitting around in airports for too long.

To make your scheduling easier, you have decided to represent time as a real number of hours, with 0.0 being the time when your mileage run starts (the moment you arrive at the first airport to begin your journey). Time continues counting up in hours indefinitely; the time two days after your mileage run starts is, for instance, 48.0. The values t_{min} and t_{max} are also a real numbers of hours.

When preparing for a mileage run, you research the flight schedules for each airport and prepare a list of flights for each pair of departure and arrival cities. Each entry in this list is for a particular possible flight between these cities, and each flight is represented by a tuple (t_{dep}, t_{arr}): the departure time and the arrival time. Remember that these are not times of the day; they are times as defined earlier. So, $(3.0, 4.0)$ represents a flight that departs three hours after you started your mileage run, and arrives at the destination one hour later (four hours after you started your mileage run). Furthermore, you have recorded *all* the flights you would be willing to take for the entire period you would be willing to conduct your mileage run—for instance, if you would still be willing to fly during the third day, then your list of possible flights will include flights with departure times in the range $[48.0, 72.0]$.

For example, suppose we want to do a mileage run from Chicago to Detroit to Omaha and back to Chicago, and your research produces the following lists of flights:

```
chicago -> detroit: (1.5, 3.0) (2.75, 4.25) (5.0, 6.0)
detroit -> omaha:   (2.0, 3.0) (3.0, 4.0) (4.0, 5.0)
omaha -> chicago:   (1.5, 4.5) (6.0, 9.0) (6.5, 9.5)
```

Suppose $t_{min} = 1.0$ and $t_{max} = 2.0$. Ultimately, we want to create a list of *every* possible sequence of flights you could take. A flight from Chicago to Detroit leaves an hour and a half after your mileage run starts, and this is the only flight you could take. Flight `(2.75, 4.25)` would require waiting more than t_{max} hours; your first airport is not technically a "layover", but you still don't want to wait that long to start your mileage run! Similarly, you still need to make it through security, etc. and we assume the minimum time to do this is the same as t_{min}. So, if you take flight `(1.5, 3.0)`, you would arrive in Detroit at time `3.0`, and the only flight that meets the t_{min} and t_{max} constraints is flight `(4.0, 5.0)`.

However, when you get to Omaha at time `5.0`, there will actually be *two* flights to Chicago that meet the

t_{min} and t_{max} constraints: `(6.0, 9.0)` and `(6.5, 9.5)`. The results is two possible sequences of flights to complete the mileage run:

```
chicago -> detroit (1.5, 3.0)
detroit -> omaha (4.0, 5.0)
omaha -> chicago (6.0, 9.0)
```

And:

```
chicago -> detroit (1.5, 3.0)
detroit -> omaha (4.0, 5.0)
omaha -> chicago (6.5, 9.5)
```

Note that we have explained the example in terms of sequences of flights that return to their origin, but that is not always the case (and whether you do or do not return to the origin does not affect the algorithm to find the sequences).

Input

The input starts with a line containing five values, each separated by a single space: A F I t_{min} t_{max}

A is the number of airports, F is the number of flights, and I is the number of airports in the mileage run itinerary ($2 \leq A, I \leq 100$ and $2 \leq F \leq 1000$). t_{min} and t_{max} are as defined earlier ($0.1 \leq t_{min}, t_{max} \leq 24.0$).

The input is followed by A lines, each with the name of an airport. The name of an airport contains only lowercase letters (a–z), no whitespace, a minimum length of 1 character, and a maximum length of 50 characters.

Next, there are F lines, one for each flight. Each flight is specified by four values, each separated by a single space: the origin airport, the destination airport, the departure time of the flight, and the arrival time of the flight (t_{dep} and t_{arr} as specified above, where $0.0 \leq t_{dep}, t_{arr} \leq 1000.0$).

Finally, there are I lines, one for each airport in the desired itinerary for the mileage run. The only airport that can appear twice in the itinerary is the first airport (and, in this case, the second occurrence of that airport can only be at the end of itinerary). Other than that, each airport appears at most once in the itinerary.

Output

The output will be the sequences of flights that complete the mileage run. Each sequence is composed of $I - 1$ lines, each representing a flight in the same format as in the input. Each sequence of flights ends with a single line containing three hash symbols (`###`). The sequences can appear in any order. So, for the given sample input, your output doesn't have to match our output byte-by-byte, but the individual sequences of flights must be correct.

If no sequences of flights are found that complete the mileage run, you must print `NO RUNS`.

Sample Input 1

```
3 9 4 1.0 2.0
chicago
detroit
omaha
chicago detroit 1.5 3.0
chicago detroit 2.75 4.25
chicago detroit 5.0 6.0
detroit omaha 2.0 3.0
detroit omaha 3.0 4.0
detroit omaha 4.0 5.0
omaha chicago 1.5 4.5
omaha chicago 6.0 9.0
omaha chicago 6.5 9.5
chicago
detroit
omaha
chicago
```

Sample Output 1

```
chicago detroit 1.5 3.0
detroit omaha 4.0 5.0
omaha chicago 6.0 9.0
###
chicago detroit 1.5 3.0
detroit omaha 4.0 5.0
omaha chicago 6.5 9.5
###
```

Sample Input 2

```
3 9 4 1.0 2.0
chicago
detroit
omaha
chicago detroit 1.5 3.0
chicago detroit 2.75 4.25
chicago detroit 5.0 6.0
detroit omaha 2.0 3.0
detroit omaha 3.0 4.0
detroit omaha 4.0 5.0
omaha chicago 1.5 4.5
omaha chicago 10.0 11.0
omaha chicago 10.5 11.5
chicago
detroit
omaha
chicago
```

Sample Output 2

```
NO RUNS
```

Sample Input 3	Sample Output 3
4 26 5 1.0 2.0 chicago detroit omaha pittsburgh chicago pittsburgh 2.5 3.5 chicago omaha 1.0 2.0 chicago omaha 3.0 4.0 chicago detroit 1.5 3.0 chicago detroit 2.75 4.25 chicago detroit 5.0 6.0 detroit pittsburgh 2.5 3.5 detroit pittsburgh 5.0 6.0 detroit omaha 2.0 3.0 detroit omaha 3.0 4.0 detroit omaha 4.0 5.0 detroit chicago 1.0 2.0 detroit chicago 3.0 4.0 omaha pittsburgh 1.5 4.5 omaha pittsburgh 6.0 9.0 omaha pittsburgh 6.5 9.5 omaha detroit 5.5 7.5 omaha detroit 8.0 9.0 omaha chicago 1.0 2.0 omaha chicago 3.0 4.0 pittsburgh omaha 2.0 3.0 pittsburgh omaha 3.0 4.0 pittsburgh detroit 1.5 2.5 pittsburgh detroit 4.0 5.0 pittsburgh chicago 10.5 11.5 pittsburgh chicago 11.25 12.75 chicago detroit omaha pittsburgh chicago	chicago detroit 1.5 3.0 detroit omaha 4.0 5.0 omaha pittsburgh 6.0 9.0 pittsburgh chicago 10.5 11.5 ### chicago detroit 1.5 3.0 detroit omaha 4.0 5.0 omaha pittsburgh 6.5 9.5 pittsburgh chicago 10.5 11.5 ### chicago detroit 1.5 3.0 detroit omaha 4.0 5.0 omaha pittsburgh 6.5 9.5 pittsburgh chicago 11.25 12.75 ###