COT 4210 Program #2: DFA Minimization

The Problem

Given a DFA, output an equivalent DFA with a minimal number of states. For grading purposes, clear rules will be given for the number of the states in the output DFA such that only single output will be deemed correct. Make sure to properly follow these EXTRA rules specified below:

When minimizing a DFA, the output results in states where each of the states in the output represent a set of states from the input DFA that forms a partition of the original states. For example, if the input DFA has states 0, 1, 2, 3, 4, 5, 6, 7 and 8, then the output DFA might group together the following sets of states:

{0, 2}
{1, 3, 6}
{4, 5}
{7}

Order and renumber the subsets as follows:

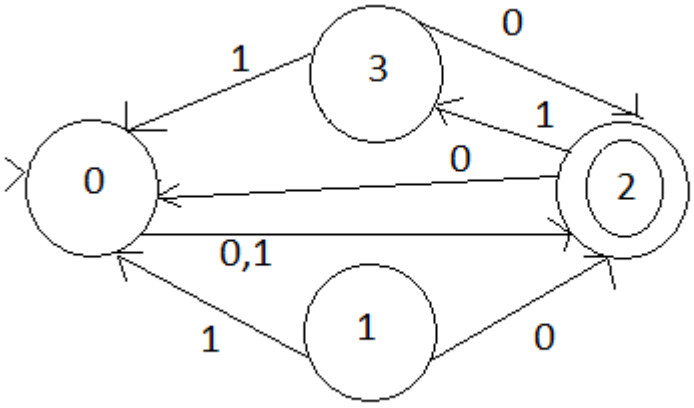
1) For each subset, chose its minimal member as the representative of the subset.
2) Sort the subsets in order of their representatives, from lowest to highest.
3) Renumber each subset in this sorted list in numerical order, starting at 0:

{0,2} – state 0
{1, 3, 6} – state 1
{4, 5} – state 2
{7} – state 3

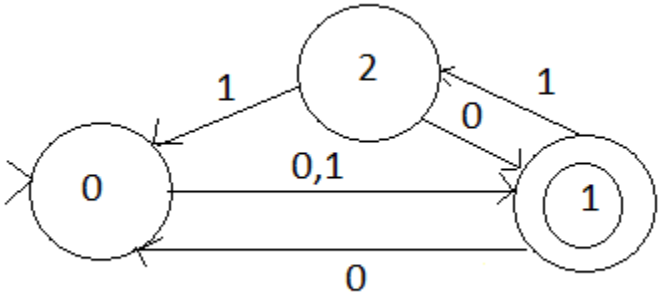
One other issue that arises is that some states in an arbitrary DFA might not be reachable from the start state. Clearly, there is no need to include any of these states in a minimized DFA. (In the algorithm shown from the Sudkamp text, this wasn't clarified.) Thus, please do the following in minimizing the input DFAs:

1) Mark all reachable states from the start state. (Any graph search algorithm is fine.)
2) Run the minimization algorithm shown in class from the Sudkamp text on ONLY the set of states that are reachable from the start state. All of the others should be ignored. For simplicity, please do NOT relabel any of the vertices at any point in time.

As an example of how your algorithm should work, consider the following DFA:



If we were to skip step #1 noted on the previous page, our algorithm would find that states 1 and 3 are equivalent, since both lead to states 2 and 0, respectively, on a 0 and 1. Yet, state 1 isn't reachable. Thus, if we skipped step #1, we would label our state to be 1. But, if you follow the designated rules in the order given, then we'd first cut out state 1, and then run our algorithm. In running the algorithm, we'd find that no pair of remaining states are equivalent, so our final DFA would simply have states 0, 2 and 3, which would then get relabeled to 0, 1 and 2, respectively. The correct minimized version of this DFA for the purposes of this program is as follows:



Input Format

The first line of the input will contain a single positive integer, *n (n < 100)*, representing the number of DFAs that are going to be described in the file.

For each DFA, the first line will have a two space separated positive integer, *s (s ≤ 1000)*, representing the number of states and *v (v ≤ 10)*, representing the size of the input alphabet. The states of the DFA will be 0 through *s*-1, and the input alphabet will be the first *v* lowercase letters. The second line will contain a positive integer, *a (a ≤ s)*, representing the number of accept states in the DFA. This will be followed by a space and *a* space separated integers in increasing order, representing the states of the DFA that are accept states. The next *s* lines will contain the transition function for the DFA with the j^{th} integer on the i^{th} line representing where to move in the DFA from state i when reading the j^{th} letter, where $0 \leq i < s$ and $0 \leq j < v$.

Output Format

For each input DFA, output a header as follows:

```
DFA #k:
```

where k represents the DFA number, starting at 1.

Follow this with the description of the minimized DFA, following exactly, the specification for each of the input DFAs. Namely, the first line of the description should have the number of states and number of characters in the input alphabet, separated by spaces. The second line should have an integer specifying the number of accept states, followed by these accept states in increasing order, separated by spaces. The next lines should specify the transition function for the minimized DFA in the same manner as described previously.

Put a blank line after the output for each DFA.

Sample Input

```
2
8 2
6 1 2 3 4 5 6
1 4
2 3
7 7
7 3
5 6
7 7
7 6
7 7
6 2
1 0
0 1
2 3
4 5
0 1
2 3
4 5
```

Sample Output

```
DFA #1:
5 2
3 1 2 3
1 1
2 3
4 4
4 3
4 4

DFA #2:
4 2
1 0
0 1
2 3
1 2
0 1
```

Implementation Restrictions

1) Write your program **in Java.**
2) Your program must read input from standard in and output to standard out.
3) Submit your source file, **dfamin.java**, via WebCourses.