A Link to the Past Randomizer

Eines von Florians absoluten Lieblingsspielen ist *The Legend of Zelda: A Link to the Past*. Vor kurzem entdeckte er, dass einige geschickte Hacker einen Randomizer dafür programmiert haben. Das ist ein Programm, das alle Items im Spiel zufällig durchmischt, sodass man am Anfang beispielsweise nicht mit dem Schwert, sondern dem Bogen startet. Florian fühlt sich in eine Kindheit zurückversetzt. Endlich kann er die fantastische Welt von Hyrule neu erkunden und dieses Abenteuer erneut durchleben.

Im Nachhinein beginnt er darüber nachzudenken, wie denn das technisch so gemacht wird. Auf der Webseite wird behauptet, dass die generierten ROMs immer lösbar sind. Es kann also nicht passieren, dass man im Spiel an einer Stelle stecken bleibt, weil ein Item, das man hier notwendigerweise braucht, leider erst in einer Truhe später im Spiel zu finden ist. Mit dieser Problematik wollen wir uns in dieser Aufgabe beschäftigen:

In unserer vereinfachten Legend of Zelda Version gibt es n verschiedene Items, die man in n Truhen irgendwo im Spiel verteilt finden kann. Sie sind von 1 bis n durchnummeriert. Man startet mit keinem einzigen Item und hat gewonnen, sobald man alle gefunden hat. Für jedes Item weißt du, welche anderen Items man braucht um es freizuschalten. Es kann mehrere Wege geben, wie man an ein Item kommen kann. Z.B. könnte man die Bomben entweder mit dem Enterhaken oder gemeinsam mit dem Bumerang und den Pegasus-Stiefeln bekommen können. Diese Information ist die in Form von m Bedingungen gegeben. Jede davon sagt aus, dass man ein gewisses Item mit einer Menge anderer Items freischalten kann. Falls es für ein Item keine Bedingungen gibt, kann man es ohne Unterstützung anderer Items direkt am Anfang bekommen. Kannst du feststellen, ob man eine gegebene Konfiguration überhaupt gewinnen kann? Wenn ja, finde eine Reihenfolge die Items freizuschalten.

Eingabe

Die erste Zeile enthält n und m, die Anzahl an Items und Bedingungen. Die folgenden m Zeilen beschreiben die Bedingungen. Jede Zeile hat die Form $i\ k\ a_1 \dots a_k$, wobei i die Nummer des freizuschaltenden Items angibt und $a_1 \dots a_k$ die k Items angibt, die man dafür braucht.

Ausgabe

Gib -1 aus, falls man nicht gewinnen kann. Anderenfalls gib eine Permutation aus, in der man die Items freischalten kann. Falls es mehrere Lösungen gibt kannst du irgendeine ausgeben.

Beispiele

Eingabe	Ausgabe	Anmerkungen
2 2 1 1 2 2 1 1	-1	Zwei Items, die man aber jeweils nur mit dem anderen freischalten kann.

Eingabe	Ausgabe	Anmerkungen
3 3 1 1 2 1 1 3 2 1 1	3 1 2	Wie die vorherige Eingabe, allerdings mit einem dritten Item. Item 1 kann man hier alternativ auch durch Item 3 freigeschalten. Da es für Item 3 gibt es keine Bedingungen gibt, kann man es direkt Anfang aufsammeln.

Eingabe	Ausgabe	Anmerkungen
5 4 2 1 1 3 1 2 4 1 3 5 4 1 2 3 4	1 2 3 4 5	Item 1 ist von Anfang an verfügbar. Für Items 2 bis 4 braucht man jeweils das Item mit der um eins niedrigeren Nummer. Um Item 5 freizuschalten sind alle Items notwendig.

Subtasks

Allgemein gilt:

- $1 \leq n, m \leq 10^6$
- Für jede Bedingung gilt:
  - $1 \leq i \leq n$
  - $1 \leq k \leq n$
  - Alle a_j sind verschieden
- Die Summe der k aller Bedingungen überschreitet 10^6 nicht

Subtask 1 (25 Punkte): $n \leq 1000$ und für jedes Item gibt es maximal eine Möglichkeit es freizuschalten.

Subtask 2 (25 Punkte): Für jedes Item gibt es maximal eine Möglichkeit es freizuschalten

Subtask 3 (10 Punkte): $n \leq 10$

Subtask 4 (15 Punkte): $n \leq 1000$

Subtask 5 (25 Punkte): Keine Einschränkungen

Limits

Zeitlimit: 1 s **Speicherlimit:** 256 MB