Algorithm Design Laboratory with Applications

Prof. Stefano Leucci

Problem: *Sliding Token.*

Let $G = (V, E)$ be a directed acyclic graph where $n = |V|$, $m = |E|$, and $V = \{0, 1, \ldots, n-1\}$. A coin is initially placed on vertex 0 and two players, Alice and Bob, take turns sliding it along the edges of G. More precisely, a turn consists of removing the token from its current vertex u and placing it on another vertex v such that $(u, v) \in E$. The first player that can no longer move the coin loses (and the other player wins). Alice goes first.

Your task is to design an algorithm that, given G, determines whether it is possible for Alice to always win the game (regardless of Bob's moves).

Input. The input consists of a set of instances, or *test-cases*, of the previous problem. The first line of the input contains the number T of test-cases. The first line of each test-case is the integer n. The next n lines each describe a vertex and its outgoing edges in G. In particular, the $(i+2)$-th line of the test-case contains a list of integers $d, v_1, v_2, \ldots, v_d$ where d is the *outdegree* of the i-th vertex in G and $(i, v_j) \in E \ \forall j = 1, \ldots, d$.

Output. The output consists of T lines, each containing a single character. The i-th line is "`A`" if Alice can win the game in the i-th test-case, and "`B`" otherwise (i.e., if Bob can always win).

Assumptions. $1 \leq T \leq 10$; $\quad 1 \leq n \leq 2^{18}$.

Example.

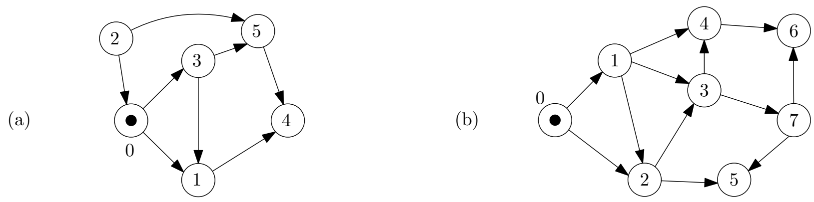

Figure 1: (a) A graph G in which Alice can always win. (b) A graph G in which Bob can always win.

Input (corresponding to the graphs in Figure 1):

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

Output:

```
A
B
```

Requirements. Your algorithm should require time $O(n+m)$ (with reasonable hidden constants).

Notes. A reasonable implementation should not require more than 3 seconds for each input file.