Baseball Elimination

Given the standings in a sports division at some point during the season, determine which teams have been mathematically eliminated from winning their division.

The baseball elimination problem. In the baseball elimination problem, there is a division consisting of n teams. At some point during the season, team i has `w[i]` wins, `l[i]` losses, `r[i]` remaining games, and `g[i][j]` games left to play against team j. A team is mathematically eliminated if it cannot possibly finish the season in (or tied for) first place. The goal is to determine exactly which teams are mathematically eliminated. For simplicity, we assume that no games end in a tie (as is the case in Major League Baseball) and that there are no rainouts (i.e., every scheduled game is played).

The problem is not as easy as many sports writers would have you believe, in part because the answer depends not only on the number of games won and left to play, but also on the schedule of remaining games. To see the complication, consider the following scenario:

```
                w[i] l[i] r[i]        g[i][j]
i  team         wins loss left   Atl Phi NY  Mon
-----------------------------------------------
0  Atlanta       83   71    8     -   1   6   1
1  Philadelphia  80   79    3     1   -   0   2
2  New York      78   78    6     6   0   -   0
3  Montreal      77   82    3     1   2   0   -
```

Montreal is mathematically eliminated since it can finish with at most 80 wins and Atlanta already has 83 wins. This is the simplest reason for elimination. However, there can be more complicated reasons. For example, Philadelphia is also mathematically eliminated. It can finish the season with as many as 83 wins, which appears to be enough to tie Atlanta. But this would require Atlanta to lose all of its remaining games, including the 6 against New York, in which case New York would finish with 84 wins. We note that New York is not yet mathematically eliminated despite the fact that it has fewer wins than Philadelphia.

It is sometimes not so easy for a sports writer to explain why a particular team is mathematically eliminated. Consider the following scenario from the American League East on August 30, 1996:

```
                w[i] l[i] r[i]          g[i][j]
i  team         wins loss left   NY Bal Bos Tor Det
-------------------------------------------------
0  New York      75   59   28     -   3   8   7   3
1  Baltimore     71   63   28     3   -   2   7   7
2  Boston        69   66   27     8   2   -   0   3
3  Toronto       63   72   27     7   7   0   -   3
4  Detroit       49   86   27     3   7   3   3   -
```

It might appear that Detroit has a remote chance of catching New York and winning the division because Detroit can finish with as many as 76 wins if they go on a 27-game winning steak, which is one more than New York would have if they go on a 28-game losing streak. Try to convince yourself that Detroit is already mathematically eliminated.

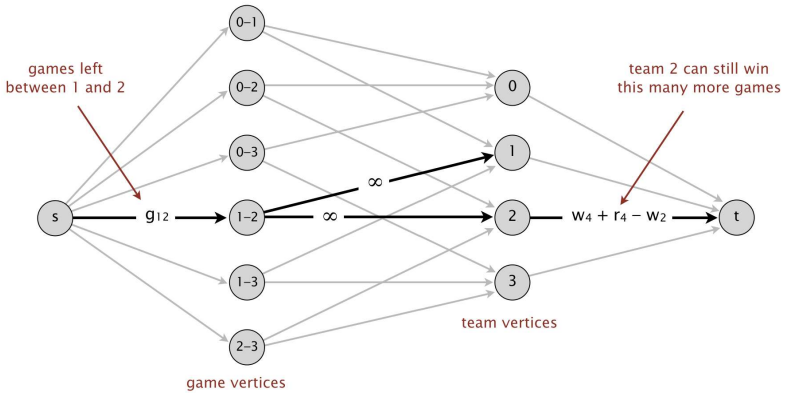
A maxflow formulation. We now solve the baseball elimination problem by reducing it to the maxflow problem. To check whether team x is eliminated, we consider two cases.

- *Trivial elimination.* If the maximum number of games team x can win is less than the number of wins of some other team i, then team x is trivially eliminated (as is Montreal in the example above). That is, if `w[x] + r[x] < w[i]`, then team x is mathematically eliminated.
- *Nontrivial elimination.* Otherwise, we create a flow network and solve a maxflow problem in it. In the network, feasible integral flows correspond to outcomes of the remaining schedule. There are vertices corresponding to teams (other than team x) and to remaining divisional games (not involving team x). Intuitively, each unit of flow in the network corresponds to a remaining game. As it flows through the network from s to t, it passes from a game vertex, say between teams i and j, then through one of the team vertices i or j, classifying this game as being won by that team.

  More precisely, the flow network includes the following edges and capacities.

  - We connect an artificial source vertex s to each game vertex i-j and set its capacity to `g[i][j]`. If a flow uses all `g[i][j]` units of capacity on this edge, then we interpret this as playing all of these games, with the wins distributed between the team vertices i and j.
  - We connect each game vertex i-j with the two opposing team vertices to ensure that one of the two teams earns a win. We do not need to restrict the amount of flow on such edges.
  - Finally, we connect each team vertex to an artificial sink vertex t. We want to know if there is some way of completing all the games so that team x ends up winning at least as many games as team i. Since team x can win as many as `w[x] + r[x]` games, we prevent team i from winning more than that many games in total, by including an edge from team vertex i to the sink vertex with capacity `w[x] + r[x] - w[i]`.

  If all edges in the maxflow that are pointing from s are full, then this corresponds to assigning winners to all of the remaining games in such a way that no team wins more games than x. If some edges pointing from s are not full, then there is no scenario in which team x can win the division. In the flow network below Detroit is team $x = 4$.



What the min cut tells us. By solving a maxflow problem, we can determine whether a given team is mathematically eliminated. We would also like to *explain* the reason for the team's elimination to a friend in nontechnical terms (using only grade-school arithmetic). Here's such an explanation for Detroit's elimination in the American League East example above. With the best possible luck, Detroit finishes the season with 49 + 27 = 76 wins. Consider the subset of teams R = { New York, Baltimore, Boston, Toronto }. Collectively, they already have 75 + 71 + 69 + 63 = 278 wins; there are also 3 + 8 + 7 + 2 + 7 = 27 remaining games among them, so these four teams must win at least an additional 27 games. Thus, on average, the teams in R win at least 305 / 4 = 76.25 games. Regardless of the outcome, one team in R will win at least 77 games, thereby eliminating Detroit.

In fact, when a team is mathematically eliminated there always exists such a convincing *certificate of elimination*, where R is some subset of the other teams in the division. Moreover, you can always find such a subset R by choosing the team vertices on the source side of a min s-t cut in the baseball elimination network. Note that although we solved a maxflow/mincut problem to find the subset R, once we have it, the argument for a team's elimination involves only grade-school algebra.

Your assignment. Write an immutable data type `BaseballElimination` that represents a sports division and determines which teams are mathematically eliminated by implementing the following API:

```
public BaseballElimination(String filename)                  // create a baseball division from given filename in format specified below
public              int numberOfTeams()                      // number of teams
public Iterable<String> teams()                              // all teams
public              int wins(String team)                    // number of wins for given team
public              int losses(String team)                  // number of losses for given team
public              int remaining(String team)               // number of remaining games for given team
public              int against(String team1, String team2)  // number of remaining games between team1 and team2
public          boolean isEliminated(String team)            // is given team eliminated?
public Iterable<String> certificateOfElimination(String team)  // subset R of teams that eliminates given team; null if not eliminated
```

The last six methods should throw an `IllegalArgumentException` if one (or both) of the input arguments are invalid teams.

Input format. The input format is the number of teams in the division n followed by one line for each team. Each line contains the team name (with no internal whitespace characters), the number of wins, the number of losses, the number of remaining games, and the number of remaining games against each team in the division. For example, the input files teams4.txt and teams5.txt correspond to the two examples discussed above.

```
% cat teams4.txt
4
Atlanta       83 71  8  0 1 6 1
Philadelphia  80 79  3  1 0 0 2
New_York      78 78  6  6 0 0 0
Montreal      77 82  3  1 2 0 0

% cat teams5.txt
5
New_York    75 59 28   0 3 8 7 3
Baltimore   71 63 28   3 0 2 7 7
Boston      69 66 27   8 2 0 0 3
Toronto     63 72 27   7 7 0 0 3
Detroit     49 86 27   3 7 3 3 0
```

You may assume that $n \geq 1$ and that the input files are in the specified format and internally consistent. Note that a team's number of remaining games does not necessarily equal the sum of the remaining games against teams in its division because a team may play opponents outside its division.

Output format. Use the following `main()` function, which reads in a sports division from an input file and prints whether each team is mathematically eliminated and a certificate of elimination for each team that is eliminated:

```
public static void main(String[] args) {
    BaseballElimination division = new BaseballElimination(args[0]);
    for (String team : division.teams()) {
        if (division.isEliminated(team)) {
            StdOut.print(team + " is eliminated by the subset R = { ");
            for (String t : division.certificateOfElimination(team)) {
                StdOut.print(t + " ");
            }
            StdOut.println("}");
        }
        else {
            StdOut.println(team + " is not eliminated");
        }
    }
}
```

Below is the desired output:

```
% java-algs4 BaseballElimination teams4.txt
Atlanta is not eliminated
Philadelphia is eliminated by the subset R = { Atlanta New_York }
New_York is not eliminated
Montreal is eliminated by the subset R = { Atlanta }

% java-algs4 BaseballElimination teams5.txt
New_York is not eliminated
Baltimore is not eliminated
Boston is not eliminated
Toronto is not eliminated
Detroit is eliminated by the subset R = { New_York Baltimore Boston Toronto }
```

Analysis (optional and ungraded). Analyze the worst-case memory usage and running time of your algorithm.

- What is the order of growth of the amount of memory (in the worst case) that your program uses to determine whether *one* team is eliminated? In particular, how many vertices and edges are in the flow network as a function of the number of teams n?
- What is the order of growth of the running time (in the worst case) of your program to determine whether *one* team is eliminated as a function of the number of teams n? In your calculation, assume that the order of growth of the running time (in the worst case) to compute a maxflow in a network with V vertices and E edges is $V E^2$.

Also, use the output of your program to answer the following question:

- Consider the sports division defined in teams12.txt. Explain in nontechnical terms (using the results of certificate of elimination and grade-school arithmetic) why Japan is mathematically eliminated.

Web submission. Submit a .zip file containing `BaseballElimination.java` and any other supporting files (excluding those in `algs4.jar`). Your may not call any library functions other than those in `java.lang`, `java.util`, and `algs4.jar`.

This assignment was developed by Kevin Wayne.