Zadanie: Walki robotów

W I Liceum Ogólnokształcącym im. Stanisława Lema w Jatnem co roku odbywają się rozgrywki ligi walk robotów. Każda klasa wystawia drużynę, której celem jest zdobycie jak największej liczby punktów. Ponieważ walki trwają aż do unieruchomienia któregoś z robotów, nie ma mowy o remisie. Zwycięstwo jest nagradzane jednym punktem, a porażka nie zmienia dorobku punktowego drużyny.

Ponieważ w czasie trwania zawodów żyje nimi cała szkoła, gazetka szkolna stara się możliwie dokładnie opisywać przebieg rozgrywek. Najistotniejsza jest możliwość określenia, które zespoły w danym momencie sezonu mają jeszcze szanse na mistrzostwo (chociażby ex aequo). Okazuje się, że nie jest to zadanie trywialne. Rozważmy tabelę:

Drużyna	Punkty	Pozostałe mecze
0	5	6
1	4	5
2	3	3

przy kolejnych meczach:

vs	0	1	2
0		4	2
1	4		1
2	2	1	

Może się zdawać, że drużyna nr 2 ma jeszcze szansę na mistrzostwo - do lidera brakuje jej dwóch punktów, a pozostały trzy spotkania zatem również trzy punkty do zdobycia. Problemem jest fakt, że drużyny nr 0 i 1 będą grały ze sobą jeszcze czterokrotnie, tak więc któraś z nich z pewnością osiągnie wynik przynajmniej siedmiu punktów i zapewni sobie mistrzostwo.

Etap I

Redaktor naczelny gazetki szkolnej poprosił Cię o napisanie programu, który wyznaczy, które drużyny wciąż mają szanse na mistrzostwo w danym punkcie sezonu. Parametrami wejściowymi metody FindPossibleWinners są:

- int[] points - tablica dorobków punktowych wszystkich drużyn w tabeli, np. points[0] oznacza ile w danym momencie punktów ma na koncie drużyna nr 0
- int[,] vs - tablica pozostałych spotkań: vs[2, 3] oznacza ile jeszcze razy drużyna nr 2 będzie walczyła z drużyną nr 3 (vs jest oczywiście symetryczne względem przekątnej)

natomiast wyjściem są:

- List<int> - lista drużyn, które wciąż mają szansę na mistrzostwo (chociażby ex aequo)
- out int[][] finalResult - tablica, która na i-tym miejscu ma zawierać przykładowy stan tabeli na koniec sezonu, przy założeniu że i-ta drużyna wygrała wszystkie swoje pozostałe spotkania. Jeśli i-ta drużyna nie ma szans na mistrzostwo, finalResult[i] powinno przyjąć wartość null.

Etap II

Okazuje się, że fani niektórych zespółów są przyjaźnie nastawieni do niektórych innych drużyn. Twoim zadaniem jest wyznaczenie maksymalnego poziomu satysfakcji kibiców ze spotkań innych zespołów przy założeniu, że ich ulubiony zespół zdobędzie mistrzostwo wygrywając wszystkie swoje spotkania. Przykładowo, jeśli fani pewnej drużyny darzą drużynę nr 2 sympatią o wartości 10, natomiast drużynę 3 sympatią o wartości 30 to wygrana drużyny 3 będzie oznaczała wzrost satysfakcji kibiców o 20(=30-10). W przeciwnym przypadku ich satysfakcja się nie zmieni (czyli jej zmiana wyniesie 0).

- int[] points, int[,] vs - jak w etapie I
- int[,] fondness - oznacza, jaką sympatią kibice darzą inne drużyny, dokładniej fondness[2, 3] oznacza jaką sympatią kibice drużyny nr 2 darzą drużynę nr 3

Wyjście:

- int[] - satysfakcja kibiców o ile dana drużyna może wygrać ligę, -1 w przeciwnym przypadku

Punktacja

- Etap I, wyznaczenie możliwych zwycięzców – 1,5 pkt.
- Etap I, wyznaczenie możliwych tabel końcowych – 0,5 pkt.
- Etap II – 0,5 pkt.

Złożoność

- Etap I powinien działać w czasie n*MF
- Etap II powinien działać w czasie n*MC,

gdzie n to liczba drużyn, a MF(C) to złożoność wyznaczania maksymalnego przepływu (*o minimalnym koszcie*), przy czym zarówno liczba wierzchołków jak i krawędzi wejściowych grafów wykorzystywanych przez algorytmy przepływowe powinna być rzędu $O(n^2)$.

Uwagi

- Limity czasu **w trakcie laboratorium** mają charakter orientacyjny i ich przekroczenie nie wpływa na punktację
- Przy optymalizacji czasów w części domowej może opłacać się przetestować różne metody wyznaczania maksymalnego przepływu dostępne w bibliotece