Jak wszyscy wiemy (a przynajmniej powinniśmy wiedzieć), w k-regularnym grafie dwudzielnym, dla k>0, zawsze istnieje skojarzenie doskonałe. Łatwo to udowodnić korzystając z twierdzenia Halla. Dowód tego twierdzenia jest konstruktywny i wynika z niego algorytm znajdujący najliczniejsze skojarzenie w czasie $O(m^2)$, gdzie m jest liczbą krawędzi. Okazuje się jednak, że można zrobić to dużo szybciej – istnieje algorytm znajdujący skojarzenie doskonałe w regularnym grafie dwudzielnym działający w czasie O(m log m). Dzisiaj zaimplementujemy ten algorytm w szczególnym przypadku, gdy stopień regularności jest potęgą dwójki; wtedy czas działania spada do O(m).

Przypuśćmy, że mamy $2^r$-regularny graf dwudzielny. Skojarzenie doskonałe można w tym przypadku znaleźć r-krotnie powtarzając operację „dzielenia grafu na pół”. Przez „dzielenie grafu na pół” rozumiemy usunięcie połowy krawędzi grafu w taki sposób, żeby stopień każdego wierzchołka zmniejszył się o połowę. Dobrze zaimplementowana operacja dzielenia na pół działa w czasie O(m)*, co gwarantuje całkowity czas działania O(m).

Zadanie polega na uzupełnieniu metod `perfectMatching` oraz `cyclePartition`. Metoda `perfectMatching` powinna znajdować skojarzenie doskonałe w zadanym $2^r$-regularnym grafie dwudzielnym w czasie O(m), gdzie m jest liczbą krawędzi. „Dzielenie grafu na pół” można zrealizować przez podział grafu na cykle, a następnie usunięcie co drugiej krawędzi z każdego cyklu. Podział na cykle realizuj w metodzie `cyclePartition`.

Można założyć, że dane będą zawsze poprawne, tzn.:

- Metoda `cyclePartition` będzie wywoływana dla grafu nieskierowanego w którym wszystkie wierzchołki mają parzysty stopień
- Metoda `perfectMatching` będzie wywoływana dla nieskierowanego grafu dwudzielnego w którym wszystkie wierzchołki mają stopień $2^r$ dla pewnego naturalnego r.

Punktacja:
3p - sprytna wersja metody `cyclePartition` działająca w czasie O(m)
1p - naiwna wersja metody `cyclePartition`, działająca w czasie większym niż O(m)
1p - metoda `perfectMatching`
Oczywiście punkty można dostać tylko za jedną wersję metody `cyclePartition` (część zajęciowa jest łącznie za 4p).
W części domowej obowiązuje wersja „sprytna”.

Wskazówki:
W wersji naiwnej metodę cyclePartition można zaimplementować korzystając z własnej metody findCycle zaimplementowanej na trzecich zajęciach. Zauważmy, że jeżeli każdy wierzchołek grafu ma parzysty stopień i graf zawiera przynajmniej jedną krawędź, to musi zawierać cykl, a po usunięciu krawędzi cyklu znów otrzymujemy graf o wszystkich stopniach parzystych.

Pełne rozwiązanie można uzyskać poprzez sprytną modyfikację przeszukiwania grafu (i nie da się tutaj wykorzystać bibliotecznych metod przeszukiwania grafu). Kiedy wykonujemy przeszukiwanie grafu w głąb i wykryjemy cykl, wystarczy usunąć wierzchołki tego cyklu ze stosu, usunąć krawędzie z grafu i kontynuować przeszukiwanie od tego stanu (a nie od początku, jak by się wydarzyło przy kolejnym wywołaniu metody findCycle).

* Zakładamy, że sąsiadów wierzchołka o stopniu d potrafimy wylistować w czasie O(d), oraz że sprawdzenie, czy graf zawiera daną krawędź, realizujemy w czasie O(1).