Jakub Wojtaszczyk — Treść zadania, Opracowanie
Alan Kutniewski — Program

Dostępna pamięć: 64 MB. *OI, Etap III, dzień pierwszy, 06.04.2011*

Impreza

Bajtazar chciałby zorganizować imprezę. Zależy mu na tym, żeby impreza była udana, i sądzi, że najlepszym sposobem na to jest, żeby wszyscy zaproszeni goście się znali. Teraz Bajtazar musi ustalić listę swoich znajomych, których ma zaprosić.

Bajtazar ma n znajomych, przy czym n jest podzielne przez 3. Szczęśliwie, większość znajomych Bajtazara zna się nawzajem. Więcej — Bajtazar pamięta, że był na imprezie, na której było $\frac{2}{3}n$ jego znajomych i na której wszyscy znali się nawzajem. Niestety, poza tym jednym faktem Bajtazar nieszczególnie wiele z tej imprezy pamięta... W szczególności nie ma pojęcia, którzy znajomi byli wtedy obecni.

Bajtazar nie musi organizować wielkiej imprezy, ale chciałby zaprosić przynajmniej $\frac{n}{3}$ spośród swoich znajomych. Nie ma za bardzo pomysłu, jak ich dobrać, więc poprosił Cię o pomoc.

Wejście

W pierwszym wierszu standardowego wejścia podane są dwie liczby całkowite n i m ($3 \leqslant n \leqslant 3\,000$, $\frac{\frac{2}{3}n(\frac{2}{3}n-1)}{2} \leqslant m \leqslant \frac{n(n-1)}{2}$), oddzielone pojedynczym odstępem, oznaczające odpowiednio liczbę znajomych Bajtazara oraz liczbę par znajomych Bajtazara, którzy znają się nawzajem. Znajomi Bajtazara są ponumerowani od 1 do n. W kolejnych m wierszach znajdują się po dwie liczby całkowite oddzielone pojedynczym odstępem. W wierszu $i+1$ (dla $i = 1, 2, \ldots, m$) znajdują się liczby a_i i b_i ($1 \leqslant a_i < b_i \leqslant n$), oddzielone pojedynczym odstępem, które oznaczają, że osoby a_i oraz b_i znają się. Każda para liczb pojawia się na wejściu co najwyżej raz.

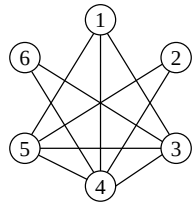
Wyjście

Twój program powinien wypisać w pierwszym i jedynym wierszu standardowego wyjścia $\frac{n}{3}$ liczb, w kolejności rosnącej, pooddzielanych pojedynczymi odstępami, oznaczających numery znajomych, których Bajtazar ma zaprosić na imprezę. Jako że istnieje wiele rozwiązań, należy wypisać dowolne z nich.

Przykład

Dla danych wejściowych:

```
6 10
2 5
1 4
1 5
2 4
1 3
4 5
```



```
4 6
3 5
3 4
3 6
```

poprawnym wynikiem jest:

```
2 4
```

Wyjaśnienie do przykładu: *Znajomi nr 1, 3, 4, 5 znają się nawzajem. Prawidłowym wynikiem jest dowolna dwójka znajomych, którzy się znają. Nie muszą oni nawet być częścią jakiejkolwiek czwórki znającej się wzajemnie.*

Rozwiązanie

Wprowadzenie

Postawiony przed nami problem łatwo wyrazić w terminologii grafowej. Mamy dany graf nieskierowany $G = (V, E)$, którego wierzchołki reprezentują znajomych Bajtazara, zaś krawędź łącząca dwa wierzchołki oznacza, że odpowiednie dwie osoby się znają. Warunki zadania gwarantują nam, że w grafie istnieje zbiór przynajmniej $2|V|/3$ wierzchołków, wśród których każda para jest połączona krawędzią; taki zbiór wierzchołków w teorii grafów nazywa się *kliką*. Naszym zadaniem jest znaleźć w tym grafie dowolną klikę rozmiaru $|V|/3$. Przypomnijmy też założenie, że $3 \mid |V|$.

Rozwiązanie wzorcowe

Niech $n = |V|$ oznacza liczbę wierzchołków w wejściowym grafie. Rozwiązanie wzorcowe działa następująco:

1. Jeżeli w grafie występują dwa wierzchołki, które nie są połączone krawędzią, to usuwamy obydwa te wierzchołki z grafu i kontynuujemy.
2. Jeżeli w grafie (po usunięciu pewnych wierzchołków zgodnie z krokiem 1) każde dwa wierzchołki są połączone krawędzią, to wybieramy dowolne $n/3$ wierzchołków i zwracamy je jako rozwiązanie.

Zbiór wierzchołków, który zwrócimy w drugim kroku, oczywiście będzie kliką. Jedyne, co trzeba sprawdzić, by zweryfikować poprawność powyższego rozwiązania, to to, czy w momencie dojścia do drugiego kroku w grafie faktycznie pozostanie nam co najmniej $n/3$ wierzchołków.

Aby się o tym upewnić, rozważmy dowolny zbiór wierzchołków W będący kliką rozmiaru $2n/3$ — nie umiemy takiego zbioru znaleźć, ale z warunków zadania wynika, że taki zbiór istnieje. Niech $U = V \setminus W$ będzie zbiorem pozostałych wierzchołków. Początkowo $|W| = 2n/3$, $|U| = n/3$. Zauważmy, że wśród dowolnych dwóch wierzchołków usuniętych w pierwszym kroku algorytmu zawsze co najmniej jeden jest z U — w przeciwnym razie w zbiorze W znajdowałyby się dwa wierzchołki niepołączone krawędzią, co przeczy temu, że W jest kliką. Wobec tego w każdym kroku algorytmu rozmiar zbioru U maleje co najmniej o jeden, a zatem algorytm wykona nie więcej

niż $n/3$ kroków. Jednocześnie w każdym kroku usuwamy dwa wierzchołki — a zatem na końcu pozostanie nam co najmniej $n/3$ wierzchołków, wobec czego algorytm faktycznie jest poprawny.

Implementacja

Naiwna implementacja algorytmu wzorcowego w każdym kroku przegląda wszystkie pary wierzchołków w poszukiwaniu jakiejś, którą można by usunąć — jako że par wierzchołków jest $O(n^2)$, a kroków — $O(n)$, to łączna złożoność takiego algorytmu to $O(n^3)$. Nietrudno jednak zobaczyć, jak tę złożoność poprawić: każdą parę wierzchołków warto przeglądać tylko raz. Faktycznie, jeżeli przy pierwszym przejrzeniu danej pary wierzchołków nie usuwamy z grafu, to znaczy, że są one połączone krawędzią, więc już nigdy nie będziemy mogli tej pary usunąć. Wobec tego możemy utrzymywać dla każdego wierzchołka informację, czy wciąż znajduje się on w grafie, czy też nie, a następnie przeglądać po kolei wszystkie pary wierzchołków. Jeśli dwa wierzchołki nie są połączone krawędzią oraz żadnego z nich jeszcze nie usunęliśmy z grafu, to usuwamy obydwa. To rozwiązanie ma złożoność czasową $O(n^2)$, a zatem liniową względem rozmiaru grafu, tzn. liczby krawędzi. Można się spodziewać, że dla $n \leqslant 3\,000$ takie rozwiązanie będzie działało dostatecznie szybko. Implementacje tego rozwiązania można znaleźć w plikach `imp.cpp`, `imp1.pas` i `imp2.cpp`.

Uwagi

Czytelnika, który trochę interesuje się teorią grafów, niniejsze zadanie może nieco zaskoczyć. Dość znanym faktem jest to, że problem rozstrzygnięcia, czy w danym grafie istnieje klika danego rozmiaru, albo też znalezienia kliki o danym rozmiarze, nawet jeśli wiadomo, że ona istnieje, jest *NP-trudny* — co oznacza, że nie jest znany algorytm o złożoności wielomianowej, który rozwiązywałby ten problem, i byłoby sporym zaskoczeniem, gdyby taki algorytm istniał.

Co więcej, NP-trudny jest również problem aproksymacji kliki, tj. znalezienia w grafie kliki, która byłaby co najwyżej C razy mniejsza od największej istniejącej w tym grafie. Co może być zaskakujące, problem ten jest trudny nie tylko dla ustalonej liczby C, jak na przykład 2, ale też dla C zależnego od liczby n wierzchołków w grafie, np. dla $C(n) = \sqrt{n}$. Przykładowo, bardzo trudno jest znaleźć w grafie klikę rozmiaru $n^{1/4}$, nawet jeśli wiemy, że jest w nim klika rozmiaru $n^{3/4}$. W tym kontekście treść zadania może wydawać się bardzo zaskakująca, jako że przybliżamy w nim — co prawda w bardzo specyficznym przypadku — maksymalną klikę z dokładnością co do połowy jej rozmiaru.

Testy

Programy zgłoszone przez zawodników były sprawdzane na jedenastu zestawach testowych. Każdy zestaw składał się z pięciu testów.

W każdym z zestawów graf zawiera klikę rozmiaru $2n/3$, zaś pozostałe wierzchołki są połączone ze sobą w sposób pseudolosowy i zależny od oznaczenia testu:

- Testy oznaczone literą *a* zawierają kilka wierzchołków o bardzo wysokich stopniach, które znają dużą część (ale nie wszystkie) wierzchołki kliki, choć same do kliki nie należą. Pozostałe wierzchołki są połączone losowo.
- W testach oznaczonych literami *b*, *c*, *d* oraz *e* większość wierzchołków spoza głównej kliki jest również połączona w klikę, krawędzie pomiędzy klikami są mniej więcej losowe, a do tego jest dodanych kilka wierzchołków izolowanych.

Nazwa	n	m
imp1a.in	30	297
imp1b.in	48	784
imp1c.in	99	3 465
imp1d.in	99	3 465
imp1e.in	99	3 429
imp2a.in	198	13 357
imp2b.in	498	96 287
imp2c.in	498	96 287
imp2d.in	498	96 287
imp2e.in	498	95 864
imp3a.in	399	54 459
imp3b.in	999	387 154
imp3c.in	999	387 154
imp3d.in	999	387 154
imp3e.in	999	385 937
imp4a.in	600	123 293
imp4b.in	1 500	870 939
imp4c.in	1 500	870 939
imp4d.in	1 500	870 939
imp4e.in	1 500	873 750
imp5a.in	900	277 948
imp5b.in	1 800	1 255 786
imp5c.in	1 800	1 255 786
imp5d.in	1 800	1 255 786
imp5e.in	1 800	1 257 899
imp6a.in	1 500	781 719
imp6b.in	1 998	1 547 054
imp6c.in	1 998	1 547 054
imp6d.in	1 998	1 547 054
imp6e.in	1 998	1 547 436
imp7a.in	2 001	1 383 666
imp7b.in	2 298	2 049 890
imp7c.in	2 298	2 049 890
imp7d.in	2 298	2 049 890
imp7e.in	2 298	2 051 731
imp8a.in	2 298	1 808 780
imp8b.in	2 499	2 421 079
imp8c.in	2 499	2 421 079
imp8d.in	2 499	2 421 079
imp8e.in	2 499	2 421 510
imp9a.in	2 499	2 141 401
imp9b.in	2 700	2 829 686
imp9c.in	2 700	2 829 686
imp9d.in	2 700	2 829 686
imp9e.in	2 700	2 824 605
imp10a.in	2 700	2 504 388
imp10b.in	2 898	3 257 199
imp10c.in	2 898	3 257 199
imp10d.in	2 898	3 257 199
imp10e.in	2 898	3 255 867
imp11a.in	3 000	3 087 131
imp11b.in	3 000	3 494 958
imp11c.in	3 000	3 494 958
imp11d.in	3 000	3 494 958
imp11e.in	3 000	3 493 490