Marek Cygan, Jakub Radoszewski **Adam Iwanicki**
Treść zadania, Opracowanie Program

Biura

Firma Bajtel jest potentatem na bajtockim rynku telefonów komórkowych. Każdy jej pracownik otrzymał służbowy telefon, w którym ma zapisane numery telefonów niektórych swoich współpracowników (a wszyscy ci współpracownicy mają w swoich telefonach zapisany jego numer). W związku z dynamicznym rozwojem firmy zarząd postanowił przenieść siedzibę firmy do nowych biurowców. W celu polepszenia efektywności pracy zostało postanowione, że każda para pracowników, którzy będą pracować w różnych budynkach, powinna znać (nawzajem) swoje numery telefonów, tzn. powinni oni mieć już zapisane nawzajem swoje numery w służbowych telefonach komórkowych. Równocześnie, zarząd postanowił zająć jak największą liczbę biurowców, aby zapewnić pracownikom komfort pracy. Pomóż zarządowi firmy Bajtel zaplanować liczbę biur i ich wielkości tak, aby spełnić oba te wymagania.

Zadanie

Napisz program, który:

- *wyczyta ze standardowego wejścia opis, czyje numery telefonów mają zapisane w swoich telefonach komórkowych poszczególni pracownicy,*
- *wyznaczy maksymalną liczbę biurowców oraz ich wielkości, spełniające warunki postawione przez zarząd firmy Bajtel,*
- *wypisze wynik na standardowe wyjście.*

Wejście

Pierwszy wiersz wejścia zawiera dwie liczby całkowite: n oraz m ($2 \leqslant n \leqslant 100\,000$, $1 \leqslant m \leqslant 2\,000\,000$), oddzielone pojedynczym odstępem i oznaczające odpowiednio: liczbę pracowników firmy Bajtel i liczbę par współpracowników, którzy mają zapisane nawzajem swoje numery telefonów w swoich telefonach komórkowych. Pracownicy firmy są ponumerowani od 1 do n.

Każdy z kolejnych m wierszy zawiera po jednej parze liczb całkowitych a_i i b_i ($1 \leqslant a_i < b_i \leqslant n$ dla $1 \leqslant i \leqslant m$), oddzielonych pojedynczym odstępem i oznaczających, że pracownicy o numerach a_i i b_i mają zapisane nawzajem swoje numery telefonów w swoich telefonach komórkowych. Każda para liczb oznaczających pracowników pojawi się na wejściu co najwyżej raz.

Wyjście

Pierwszy wiersz wyjścia powinien zawierać jedną liczbę całkowitą: maksymalną liczbę biurowców, które powinna zająć firma Bajtel. Drugi wiersz wyjścia powinien zawierać niemalejący ciąg dodatnich liczb całkowitych, pooddzielanych pojedynczymi odstępami, *oznaczających wielkości biurowców (liczby rozlokowanych w nich pracowników). Jeżeli istnieje więcej niż jedno poprawne rozwiązanie, Twój program powinien wypisać dowolne z nich.*

Przykład

Dla danych wejściowych:

```
7 16
1 3
1 4
1 5
2 3
3 4
4 5
4 7
4 6
5 6
6 7
2 4
2 7
2 5
3 5
3 7
1 7
```

poprawnym wynikiem jest:

```
3
1 2 4
```

Przykładowy dobry przydział pracowników do biur wygląda następująco: do pierwszego biura pracownik o numerze 4, do drugiego pracownicy 5 i 7, a do trzeciego pracownicy 1, 2, 3 i 6.

Rozwiązanie

Analiza problemu

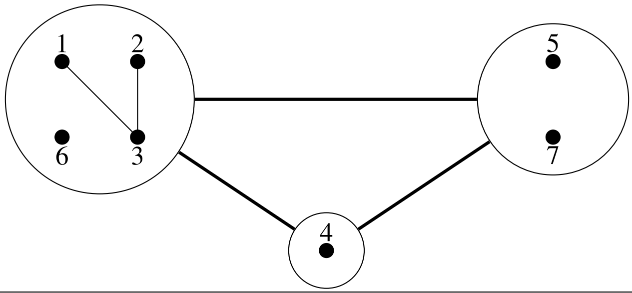
Problem z zadania można opisać za pomocą nieskierowanego grafu $G = (V,E)$. Przyjmijmy, że pracownicy Bajtelu to wierzchołki $V = \{1,2,\dots,n\}$. Zbiór krawędzi E to zbiór par (u,v) (możemy założyć, że $u < v$, ponieważ graf jest nieskierowany), gdzie u i v oznaczają pracowników posiadających nawzajem swoje numery telefonów.

Przydzielenie pracowników do biur to w interpretacji grafowej podział zbioru wierzchołków V na parami rozłączne, niepuste podzbiory $V_1 \cup \dots \cup V_k = V$, dla którego:

- dowolne dwa wierzchołki, należące do różnych podzbiorów są połączone krawędzią:

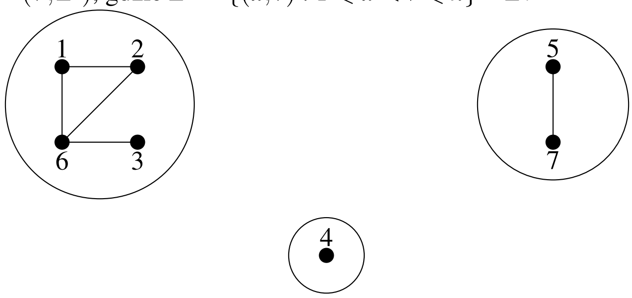
$$\forall_{1 \leqslant i < j \leqslant k} \forall_{u \in V_i} \forall_{v \in V_j} \ (u,v) \in E,$$

- liczba skonstruowanych podzbiorów $V_1, V_2, \dots, V_k$ jest możliwie największa.



Rys. 1: Struktura grafu G dla przykładu z treści zadania. Większe kółka oznaczają optymalny podział zbioru V (maksymalizujący k), a krawędzie między kółkami oznaczają, że każda para pracowników z połączonych podzbiorów jest w G połączona krawędzią.

Problem podziału grafu na zbiory wierzchołków spełniające podane wyżej kryteria nie brzmi naturalnie i znajomo. Okazuje się jednak, że modyfikując graf G, możemy sprowadzić zadanie do problemu dobrze znanego. Zmieńmy radykalnie zbiór krawędzi w grafie G — usuńmy wszystkie krawędzie ze zbioru E i dodajmy takie pary (u,v), które dotychczas nie były połączone krawędzią. Skonstruowany graf nazwiemy *dopełnieniem grafu G* i będziemy go oznaczać $G' = (V, E')$, gdzie $E' = \{(u,v) : 1 \leqslant u < v \leqslant n\} - E$.



Rys. 2: Graf G' — dopełnienie grafu dla przykładu z treści zadania.

Zauważmy, że jeżeli w zmodyfikowanym grafie G' dwa wierzchołki z V są połączone krawędzią, to muszą oznaczać pracowników pracujących w tym samym biurze, czyli każda spójna składowa grafu G' musi być w całości zawarta w jednym biurze. Z drugiej strony, pracownicy, którym odpowiadają wierzchołki z różnych składowych, mogą być umieszczeni w różnych biurach. Maksymalną liczbę biur k uzyskamy więc, lokując pracowników każdej składowej w odrębnym biurze.

Sprowadziliśmy zatem oryginalne zadanie do problemu znajdowania spójnych składowych w dopełnieniu zadanego grafu — problemu, który ma wiele klasycznych rozwiązań. Są wśród nich przeszukiwanie grafu wszerz (BFS) bądź w głąb (DFS), czy wykorzystanie struktury *Find-Union* (opisy tych metod można znaleźć np. w książce [19]). Jednak ich zastosowanie w naszym zadaniu może być trudne — przeszkodę stanowi fakt, że graf G' może mieć bardzo dużo krawędzi. Ich liczba może wynosić nawet $\frac{n \cdot (n-1)}{2}$, co w przypadku ograniczeń z zadania oznacza około 5 miliardów krawędzi. Poszukiwanie spójnych składowych w grafie G' o tak gigantycznym rozmiarze musimy przeprowadzić bez konstruowania grafu G'. Zanim pokażemy, jak to zrobić, zastanówmy się, jak najlepiej zaimplementować jeden z klasycznych algorytmów.

Rozwiązanie o złożoności czasowej $O(n^2)$

W plikach `bius0.pas` oraz `bius1.cpp` znajdują się rozwiązania zadania, w których spójne składowe grafu G' są wyznaczane algorytmem przeszukiwania grafu w głąb (DFS). W pliku `bius2.cpp` znajduje się rozwiązanie, wykorzystujące do tego celu algorytm *Find-Union*. Ponieważ graf G' może mieć prawie $\frac{n^2}{2}$ krawędzi, to złożoność czasowa zaimplementowanych rozwiązań wynosi $O(n^2)$ (w przypadku wykorzystania algorytmu BFS lub DFS) lub $O(n^2 \log^* n)$ (przy użyciu struktury *Find-Union*).

W załączonych rozwiązaniach nie konstruujemy *explicite* grafu G', gdyż wymagałoby to zbyt wiele pamięci. Dla każdego wierzchołka, na bieżąco, w miarę potrzeby wyznaczamy wychodzące z niego krawędzie. W tym celu wstępnie sortujemy listy sąsiadów wszystkich wierzchołków grafu G. Lista sąsiadów wierzchołka v w grafie G pozwala nam także przejrzeć w czasie $O(n)$ wszystkich sąsiadów wierzchołka v w grafie G' — są to wierzchołki, których brak na liście. Co więcej, taka reprezentacja grafu G' nie wymaga więcej pamięci niż reprezentacja grafu G.

Sortowanie list sąsiadów G można zrealizować w złożoności czasowej $O(m \log m)$ za pomocą jednego z klasycznych algorytmów sortowania, na przykład sortowania przez scalanie. Operację tę można wykonać także w czasie $O(n+m)$ za pomocą sortowania pozycyjnego. W tym celu krawędzie grafu G przedstawiamy jako pary liczb naturalnych z przedziału $[1,n]$. Opisy algorytmów sortowania przez scalanie (ang. *Merge Sort*) i sortowania pozycyjnego (ang. *Radix Sort*) znajdują się, na przykład, w książce [19].

Rozwiązanie wzorcowe

Poszukajmy efektywniejszej metody wyznaczania spójnych składowych grafu G' bez właściwej konstrukcji tego grafu. Zmodyfikujmy w tym celu algorytm *Find-Union* (patrz [19]), który polega na stopniowym grupowaniu wierzchołków w zbiory odpowiadające spójnym składowym. Początkowo tworzymy n zbiorów jednoelementowych — każdy wierzchołek będzie w swoim zbiorze. Następnie przetwarzamy wierzchołki grafu G w dowolnej kolejności. Dla każdego wierzchołka v chcielibyśmy zaktualizować istniejący podział na spójne składowe, uwzględniając wszystkie krawędzie w grafie G' incydentne z tym wierzchołkiem, tzn. łącząc zbiory, pomiędzy którymi przebiegają te krawędzie.

Załóżmy, że przed przetworzeniem wierzchołka v mamy w strukturze *Find-Union* zapisany aktualny podział grafu na składowe $S_1, S_2, \ldots, S_l$ i niech $v \in S_1$. Uwzględnienie krawędzi wychodzących z v może spowodować zmianę podziału G' na spójne składowe — pewne spośród składowych $S_2, \ldots, S_l$ mogą zostać połączone z S_1. Aby stwierdzić, czy krawędzie G' incydentne z v spowodują połączenie składowych S_1 oraz S_i, wystarczy porównać:

- liczbę krawędzi łączących wierzchołek v ze składową S_i w grafie G,
- z liczbą wierzchołków tej składowej $|S_i|$.

Jeśli pierwsza z tych liczb jest mniejsza, to w grafie G' musi istnieć krawędź łącząca wierzchołek v z wierzchołkiem ze zbioru S_i. W ten sposób, korzystając z reprezentacji grafu G, możemy konstruować składowe grafu G' — uwalniając się tym samym od konieczności myślenia o reprezentacji „niewygodnego” grafu G'.

Oznaczmy przez a_i liczbę krawędzi grafu G, łączących wierzchołek v ze spójną składową S_i. Czy potrafimy efektywnie wyznaczyć tę liczbę? Okazuje się, że tak!

- Na początku dla każdej spójnej składowej ustalamy a_i równe 0.
- Dla każdej krawędzi (v,u) wychodzącej z v w grafie G, gdzie $u \in S_j$ oraz $j \neq 1$, zwiększamy wartość a_j o jeden. Znalezienie zbioru S_j, do którego należy wierzchołek u, wykonywane jest za pomocą operacji *Find*.
- Każdą spójną składową S_i $(i \neq 1)$, dla której $a_i < |S_i|$, łączymy ze składową S_1. Łączenie zbiorów jest wykonywane za pomocą operacji *Union*.

Po rozważeniu wszystkich wierzchołków v ze zbioru V otrzymujemy podział G' na spójne składowe. Trzeba jeszcze zastanowić się, ile czasu zajmie nam osiągnięcie tego efektu. Niejasne wydaje się, dlaczego zaprezentowany algorytm miałby być szybki: wszak złożoność czasowa przetworzenia jednego wierzchołka v to pesymistycznie $O(n)$ (tyle może być łącznie składowych G'), a w algorytmie wykonywanych jest n kroków. Przyjrzyjmy się zatem dokładniej, ile razy poszczególne operacje zostają wykonane w trakcie działania algorytmu.

- Operacji zerowania zmiennych a_i (pierwsza grupa operacji) wydaje się być zdecydowanie za dużo. Zauważmy jednak, że jeśli zmienna a_i związana ze składową S_i na zakończenie fazy jest równa zero, to na pewno S_i zostanie połączona z S_1 i zniknie przed następną fazą. Operacji zerowania *takich zmiennych* może więc być w czasie trwania całego algorytmu najwyżej $O(n)$, bo tyle jest operacji łączenia składowych. Jeśli natomiast zmienna a_i jest w danej fazie zwiększana, to musiała istnieć krawędź w grafie G (łącząca S_1 z S_i), która to spowodowała. Niezależnie od tego, czy składowa S_i zostanie połączona z S_1 w tej fazie czy nie, takich operacji zerowania a_i może być w całym algorytmie najwyżej m. Razem operacje z tej grupy można więc wykonać w czasie $O(m+n)$.
- Operacje z drugiej grupy są w widoczny sposób związane z istnieniem krawędzi — dla każdej krawędzi grafu G wykonujemy jedno dodawanie i jedną operację *Find*. Razem wymaga to czasu $O(m \log^* n)$.
- Oszacujmy jeszcze koszt operacji z trzeciej grupy. Łączna liczba sprawdzeń warunku $a_i < |S_i|$ jest taka sama, jak łączna liczba zerowań zmiennych a_i, czyli $O(m+n)$. Z kolei liczba wykonań operacji *Union* jest oczywiście rzędu $O(n)$. Cała ta grupa operacji jest więc wykonywana w czasie $O(m + n \log^* n)$.

Ostatecznie pokazaliśmy, że złożoność czasowa zaprezentowanego rozwiązania wynosi $O((n+m) \cdot \log^* n)$. Jego implementacja znajduje się w plikach: `biu.cpp` oraz `biu1.pas`.

Rozwiązanie alternatywne

Spójne składowe G' można także znaleźć, stosując zupełnie odmienne podejście niż poprzednio. W zależności od własności grafu G' zastosujemy do niego klasyczny algorytm znajdowania składowych lub wykażemy, że graf ma postać, przy której efektywne będzie inne podejście.

Podstawą podziału grafów na grupy będzie liczba krawędzi i wielkość składowych. Wielkości te można powiązać następująco:

Obserwacja 1 Jeśli w grafie G' istnieje spójna składowa S zawierająca k wierzchołków, to w grafie G jest co najmniej $k \cdot (n-k)$ krawędzi.

Dowód Poza spójną składową S w grafie G (a zatem i w G') jest dokładnie $n-k$ wierzchołków. Żaden z tych wierzchołków nie może być połączony krawędzią w grafie G' z żadnym wierzchołkiem z S. To oznacza, że w G każdy spośród tych $n-k$ wierzchołków jest połączony z każdym wierzchołkiem S, czyli w grafie G jest co najmniej $k \cdot (n-k)$ krawędzi. ■

Widzimy więc, że jeżeli graf G' zawiera sporą składową, powiedzmy o rozmiarze zbliżonym do $\frac{n}{2}$, to graf G musi mieć prawie $\frac{n^2}{4}$ krawędzi. Stąd, jeżeli m jest małe w stosunku do n^2, to w G' nie może być tak dużych składowych. Zastanówmy się, jak wykorzystać tę zależność do rozwiązania zadania. Zdefiniujmy stałą, którą wykorzystamy przy klasyfikacji grafów,

$$k_0 = \max(k \leqslant \frac{n}{2} : k \cdot (n-k) \leqslant m)$$

(dla uproszczenia zapisu będziemy odtąd zakładać, że $2|n$).

Przypadek 1. Jeżeli $k_0 = \frac{n}{2}$, to $\frac{n^2}{4} = k_0 \cdot (n-k_0) \leqslant m$. To oznacza, że $2m \geqslant \frac{n^2}{2} \geqslant \frac{n(n-1)}{2}$. Ponieważ razem w grafie G oraz w grafie G' jest $\frac{n(n-1)}{2}$ krawędzi, to uzyskana zależność oznacza, że graf G' posiada co najwyżej $\frac{n(n-1)}{2} - m \leqslant 2m - m = m$ krawędzi. W takim przypadku możemy po prostu wyznaczyć graf G', a jego spójne składowe odnaleźć za pomocą jednego ze standardowych algorytmów opisanych na początku opracowania. Stosując przeglądanie DFS lub BFS, można uzyskać złożoność czasową $O(n+m)$.

Przypadek 2. Zastanówmy się, jak wygląda graf G', gdy $k_0 < \frac{n}{2}$. W tym wypadku mamy następującą własność.

Obserwacja 2 Jeżeli $k_0 < \frac{n}{2}$, to w grafie G' musi istnieć spójna składowa o liczności co najmniej $n - k_0$.

Dowód Pokażemy najpierw, że w tym przypadku w grafie G' musi istnieć spójna składowa rozmiaru większego niż $\frac{n}{2}$. Dowód przeprowadzimy przez sprzeczność.

Gdyby wszystkie składowe G' były nie większe niż $\frac{n}{2}$, to z każdego wierzchołka G wychodziłoby co najmniej $\frac{n}{2}$ krawędzi, łączących go z wierzchołkami z pozostałych składowych G'. To oznacza, że graf G miałby co najmniej $n \cdot \frac{n}{2} \cdot \frac{1}{2} = \frac{n^2}{4}$ krawędzi, czyli $\frac{n^2}{4} = \frac{n}{2}\left(n - \frac{n}{2}\right) \leqslant m$. To jest jednak sprzeczne z założeniem, że $k_0 < \frac{n}{2}$.

Wiemy już, że w grafie G' istnieje spójna składowa rozmiaru $l > \frac{n}{2}$. To pozwala nam stwierdzić, że graf G zawiera co najmniej $l \cdot (n-l)$ krawędzi, czyli $m \geqslant l \cdot (n-l)$.

Zauważmy, że wartości $i(n-i)$ wzrastają dla $i = 1, 2, \ldots, \frac{n}{2}$. Gdyby więc zachodziło $l < n - k_0$, to z prostego przekształcenia nierówności mielibyśmy $k_0 < n - l \leqslant \frac{n}{2}$ i dalej $k_0 \cdot (n-k_0) < (n-l) \cdot (n-(n-l)) = l(n-l) \leqslant m$. To jest jednak sprzeczne z definicją stałej k_0, więc nierówność $l < n - k_0$ nie może zachodzić. Ostatecznie $l \geqslant n - k_0$ i rzeczywiście w grafie G' istnieje spójna składowa rozmiaru co najmniej $n - k_0$. ■

Pokazaliśmy, że w rozważanym przypadku G' ma dużą składową. Pozostaje jeszcze ją znaleźć. Poniższe spostrzeżenie pozwala nam zidentyfikować sporą grupę wierzchołków tej składowej.

Obserwacja 3 Jeżeli graf G' zawiera spójną składową S o liczności nie mniejszej niż $n-k_0 > \frac{n}{2}$, to należy do niej każdy wierzchołek ze zbioru V, z którego w grafie G wychodzi mniej niż $n-k_0$ krawędzi.

Dowód Z dowolnego wierzchołka, który nie należy do S, wychodzi w G co najmniej $n-k_0$ krawędzi, prowadzących właśnie do wierzchołków składowej S. Skoro tak, to wszystkie wierzchołki G, z których w grafie G wychodzi mniej niż $n-k_0$ krawędzi, muszą należeć do S. ■

Zastanówmy się jeszcze, ile może być w grafie G wierzchołków o stopniu nie mniejszym niż $n-k_0$ (takie wierzchołki mogą, ale nie muszą należeć do S). Oznaczmy przez x liczbę tych wierzchołków. Łączna liczba incydentnych z nimi krawędzi w G to co najmniej $\frac{x(n-k_0)}{2}$, więc także $\frac{x(n-k_0)}{2} \leqslant m$. Gdyby $x \geqslant 2k_0+2$, to mielibyśmy zatem $(k_0+1)(n-k_0) \leqslant m$, ale to jest niemożliwe, gdyż z definicji stałej k_0 wynika, że $(k_0+1)(n-k_0) > (k_0+1)(n-(k_0+1)) > m$. Widzimy więc, że $x \leqslant 2k_0+1$.

Algorytm. Jesteśmy gotowi do zapisania szkicu algorytmu, opartego na wszystkich wykonanych oszacowaniach:

1. Wyznaczamy stałą $k_0 = \max(k \leqslant \frac{n}{2} : k \cdot (n-k) \leqslant m)$, na przykład przeglądając wszystkie możliwości w złożoności czasowej $O(n)$.
2. Jeżeli $k_0 = \frac{n}{2}$, to do rozwiązania zadania wykorzystujemy klasyczną metodę: konstruujemy graf G' i dzielimy go bezpośrednio na spójne składowe. Złożoność czasowa w tym przypadku to $O(n+m)$.
3. Jeżeli $k_0 < \frac{n}{2}$, to:
   (a) Wszystkie wierzchołki grafu G o stopniu mniejszym niż $n-k_0$ łączymy w jedną spójną składową S grafu G'. Wierzchołki te znajdujemy w czasie $O(n+m)$.
   (b) Dla każdego z pozostałych wierzchołków (jest ich co najwyżej $2k_0+1$) znajdujemy listy krawędzi incydentnych z nim w G' — można to zrobić w czasie $O(k_0 n)$ za pomocą algorytmu podobnego do metody znajdowania dopełnienia grafu, opisanej w pierwszym rozdziale.
   (c) Przekształcamy graf G', zastępując wszystkie wierzchołki ze składowej S jednym wierzchołkiem. W tak zmienionym grafie znajdujemy spójne składowe za pomocą klasycznego algorytmu. Złożoność czasowa tego kroku to $O(k_0 n)$, gdyż graf ma co najwyżej $2k_0+2$ wierzchołków i co najwyżej $(2k_0+2)n$ krawędzi. Znalezione spójne składowe tego grafu, to (po „rozwinięciu” składowej S) szukane spójne składowe grafu G'.

Jedynym aspektem powyższego algorytmu, jaki może nas martwić, jest występowanie w punktach 3b oraz 3c trudnej do oszacowania złożoności czasowej $O(k_0 n)$. Spróbujmy więc znaleźć górne ograniczenie na wartość wyrażenia $k_0 n$, pamiętając o tym, że $k_0 < \frac{n}{2} < n - k_0$:

$$k_0 n = k_0(k_0 + (n - k_0)) < k_0((n - k_0) + (n - k_0)) = 2 \cdot k_0(n - k_0) \leqslant 2m.$$

Wykonane szacowanie pozwala nam ostatecznie podsumować złożoność zaprezentowanego algorytmu — wynosi ona $O(n + m)$. Implementacje algorytmu znajdują się w plikach `biu2.pas` oraz `biu3.cpp`.

Testy

Rozwiązania zawodników były sprawdzane na zestawie 10 testów. Krawędzie w każdym teście są w losowym porządku.

Nazwa	n	m	Opis
biu1.in	30	306	test poprawnościowy: graf G' składa się z kliki, drzewa wzbogaconego o kilka krawędzi oraz małej liczby losowych krawędzi,
biu2.in	100	3 877	graf G' składa się z dwóch klik i drzewa,
biu3.in	500	96 063	graf G' składa się z klik o rozmiarach będących kolejnymi potęgami 2,
biu4.in	1 000	483 264	graf G' składa się z niewielkich klik oraz losowej części,
biu5.in	5 000	1 962 785	graf G' składa się z kilku klik i drzew oraz losowej części, będącej grafem stosunkowo rzadkim,
biu6.in	10 000	1 944 928	graf G' składa się z 5 klik i 5 drzew oraz losowej części, będącej grafem stosunkowo gęstym,
biu7.in	20 000	1 955 565	graf G' składa się z 5 klik uzupełnionych kilkoma krawędziami oraz losowej części, będącej grafem stosunkowo rzadkim,
biu8.in	40 000	1 832 878	duży test, zawierający jednego pracownika, posiadającego telefony do wszystkich pozostałych (czyli tworzącego jednoosobowe biuro) oraz grupę kilku innych pracowników, którzy mogą zajmować oddzielne biuro,
biu9.in	69 671	1 792 601	graf G' składa się z dwóch niedużych klik z kilkoma dodatkowymi krawędziami oraz losowej części, będącej grafem bardzo rzadkim,
biu10.in	99 328	993 211	graf G' składa się z 3 małych klik oraz losowej, pozostałej części.