Jakub Radoszewski
Treść zadania, Opracowanie

Michał Adamaszek
Program

OI, Etap I

Profesor Szu

W mieście Bajtion ma swoją siedzibę bajtocki uniwersytet. Oprócz głównego gmachu, uniwersytet ma do dyspozycji n domków dla pracowników naukowych. Domki połączone są jednokierunkowymi drogami, może jednak być wiele dróg łączących dwa domki, istnieją także drogi łączące gmach uniwersytetu z domkami (droga może łączyć także pewien obiekt z nim samym). Bajtion został tak skonstruowany, żeby żadne drogi nie przecinały się w miejscach innych niż domki lub gmach (ale mogą przebiegać mostami i tunelami); ponadto każda droga zaczyna się w pewnym domku lub w gmachu i kończy się w domku lub w gmachu. Wiadomo ponadto, że istnieje co najmniej jedna droga łącząca pewien domek z gmachem uniwersytetu.

Pewnego razu uniwersytet zapragnął zatrudnić u siebie znanego specjalistę informatyki teoretycznej — profesora Szu. Jak wielu wielkich naukowców, profesor Szu ma dziwny zwyczaj; otóż każdego dnia lubi dojeżdżać do gmachu uniwersytetu inną trasą (będącą drogą bądź układem dróg, z których każda następna zaczyna się w domku, w którym kończy się poprzednia; trasa może przechodzić przez ten sam domek bądź główny gmach uniwersytetu wielokrotnie). Profesor dwie trasy uważa za różne, jeżeli różnią się chociaż jedną wykorzystaną drogą (przy czym kolejność dróg jest ważna, a dwie różne drogi łączące te same domki uważa on za różne).

Znając schemat połączeń między domkami Bajtionu, pomóż uniwersytetowi znaleźć domek, z którego istnieje najwięcej różnych tras do gmachu uniwersytetu (zamieszkawszy w tym domku, profesor Szu będzie chciał najdłużej pracować na uczelni) — jeżeli takich domków jest więcej niż jeden, to podaj wszystkie z nich. Jeżeli przy tym z jakiegoś domku istnieje więcej niż 36 500 tras do gmachu, to zakładamy, że profesor może tam zamieszkać na zawsze (jako że nie może żyć nieskończenie długo, a 100 lat to dość bezpieczna granica).

Zadanie

Napisz program, który:

- *wczyta ze standardowego wejścia schemat połączeń między domkami Bajtionu,*
- *wyznaczy domki, w których profesor Szu mógłby mieszkać najdłużej, oraz najdłuższy czas jego zamieszkiwania,*
- *wypisze wynik na standardowe wyjście.*

Wejście

Pierwszy wiersz wejścia zawiera dwie liczby całkowite n oraz m ($1 \leqslant n,m \leqslant 1\,000\,000$) oddzielone pojedynczym odstępem i oznaczające odpowiednio liczbę domków i liczbę dróg w Bajtionie (domki są ponumerowane liczbami od 1 do n, a umownie nadajemy gmachowi uniwersytetu numer $n+1$). W wierszach o numerach od 2 do $m+1$ znajdują się pary liczb całkowitych a_i, b_i ($1 \leqslant a_i, b_i \leqslant n+1$ dla $1 \leqslant i \leqslant m$) oddzielone pojedynczymi odstępami i oznaczające odpowiednio numer domku, w którym zaczyna się, i numer domku, w którym kończy się i–ta droga.

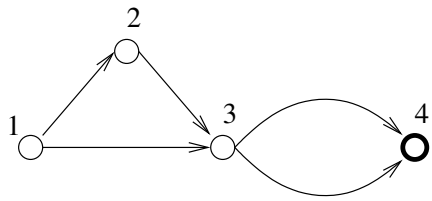
Wyjście

Pierwszy wiersz wyjścia powinien zawierać maksymalną liczbę dni jaką profesor Szu może mieszkać w Bajtionie lub jedno słowo „`zawsze`*”, jeżeli ta liczba przekracza 36 500 dni. W drugim wierszu powinna znajdować się liczba domków, zamieszkanie w których zapewnia profesorowi okres pobytu podany w pierwszym wierszu wyjścia. W trzecim wierszu powinny znaleźć się numery wszystkich takich domków, oddzielone pojedynczymi odstępami i podane w kolejności rosnącej. Wszystkie domki, w których profesor może zamieszkać na zawsze uważamy przy tym za jednakowo dobre.*

Przykład

Dla danych wejściowych:

```
3 5
1 2
1 3
2 3
3 4
3 4
```

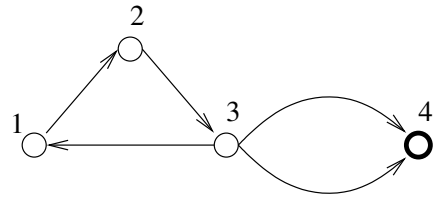


poprawnym wynikiem jest:

```
4
1
1
```

Natomiast dla danych:

```
3 5
1 2
2 3
3 1
3 4
3 4
```



poprawnym wynikiem jest:

```
zawsze
3
1 2 3
```

Rozwiązanie

Analiza problemu

Na wstępie zauważmy, że sytuację opisaną w treści zadania możemy przedstawić jako graf skierowany, którego wierzchołkami są domki i gmach uniwersytetu, a krawędziami — jednokierunkowe drogi. Graf ten może być multigrafem, to znaczy mogą w nim występować krawędzie wielokrotne i pętle.

Na pytanie postawione w treści zadania będziemy mogli łatwo odpowiedzieć, jeżeli uda nam się każdy wierzchołek przyporządkować do jednej z następujących grup:

- *wierzchołków zerowych* — czyli takich, z których nie da się dojść do gmachu uniwersytetu,
- *wierzchołków skończonych* — są to wierzchołki, z których istnieje skończenie wiele tras do gmachu uniwersytetu,
- *wierzchołków nieskończonych* — z których do gmachu uniwersytetu prowadzi nieskończenie wiele tras; wierzchołek jest nieskończony tylko wtedy, gdy można z niego osiągnąć cykl, z którego da się dojść do gmachu uniwersytetu (można wtedy skonstruować nieskończenie wiele tras, które przechodzą przez ten cykl $0, 1, 2, \ldots$ razy) — gdyby taki cykl nie istniał, to każda trasa do gmachu mogłaby zawierać co najwyżej n różnych wierzchołków, czyli tras byłoby skończenie wiele.

Dla każdego wierzchołka skończonego trzeba także policzyć liczbę różnych tras, prowadzących z niego do gmachu. Na jej podstawie dzielimy dalej wierzchołki skończone na *skończone małe*, czyli takie, dla których liczba tras jest niewiększa niż 36 500, oraz *skończone duże* — czyli pozostałe.

Po dokonaniu takiej klasyfikacji wierzchołków wynik jest prawie gotowy. Jeżeli istnieje jakikolwiek wierzchołek nieskończony lub wierzchołek skończony duży, to należy wypisać odpowiedź „`zawsze`" i podać wszystkie takie wierzchołki. W pozostałych przypadkach znajdujemy maksimum z wartości przypisanych wierzchołkom skończonym małym i wypisujemy wszystkie wierzchołki, dla których liczba różnych tras do gmachu uniwersytetu jest równa temu maksimum. Ten krok jesteśmy w stanie wykonać w czasie $O(n)$. Ponieważ w zadaniu jest założenie, że istnieje przynajmniej jeden domek, z którego można dojść do gmachu uniwersytetu, więc opisane przypadki wyczerpują wszystkie możliwości. Pozostaje więc tylko sklasyfikować wierzchołki — na tym skupimy się w dalszej części rozwiązania.

Klasyfikacja wierzchołków

Wierzchołki zerowe

Najłatwiej będzie nam zidentyfikować wierzchołki zerowe. Z definicji są to takie wierzchołki, z których nie jest osiągalny gmach uniwersytetu; bezpośrednie sprawdzenie tej własności (na przykład poprzez przeszukiwanie grafu wszerz czy w głąb startując kolejno z każdego z tych wierzchołków) nie jest jednak wystarczająco szybkie — ma złożoność czasową aż $O(n \cdot (n+m))$.

Szybszy algorytm uzyskamy, rozważając osiągalność w nieco zmienionym grafie. Jeżeli z pewnego wierzchołka v nie da się w grafie G osiągnąć gmachu g, to w grafie odwróconym G^T (utworzonym z G przez zmianę zwrotu wszystkich krawędzi na przeciwny) z gmachu g nie da się przejść do wierzchołka v. Wynika to stąd, że każda trasa z g do v w grafie G^T odpowiada trasie z v do g w wyjściowym grafie. Wystarczy więc „odwrócić" graf G w czasie $O(n+m)$, a następnie, wykonując tylko jedno przeszukiwanie grafu G^T z wierzchołka g w czasie $O(n+m)$, wyznaczyć wierzchołki nieosiągalne z gmachu, czyli wierzchołki zerowe.

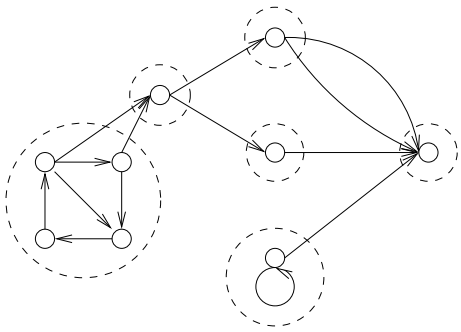
W dalszym opisie algorytmu będziemy więc pomijać wierzchołki zerowe, ponieważ nie mogą one stanowić rozwiązania ani wchodzić w skład interesujących nas dróg prowadzących do gmachu.

Graf silnie spójnych składowych

Przedstawimy dwa sposoby klasyfikacji wierzchołków na skończone i nieskończone. W pierwszym korzystamy z dość zaawansowanych pojęć, co sprawia, że algorytm jest niełatwy w implementacji. Jest jednak elegancki w zapisie i można go łatwo przystosować do rozwiązywania innych, podobnego typu zadań. W drugim algorytmie wykorzystujemy tylko elementarne pojęcia, jest więc stosunkowo prosty w opisie i implementacji, jednakże jego poprawność jest mniej oczywista i nie uogólnia się on łatwo na inne podobne problemy.

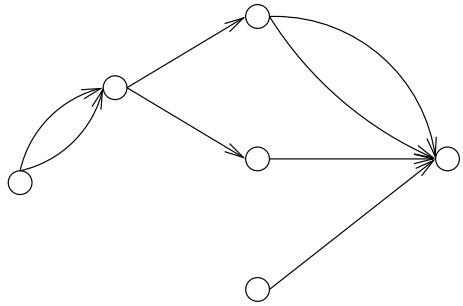
Rozpocznijmy od przypomnienia pojęcia silnie spójnej składowej w grafie.

Definicja 1 Powiemy, że dwa wierzchołki w_1 i w_2 należą do tej samej *silnie spójnej składowej* grafu, jeżeli w grafie istnieje trasa z wierzchołka w_1 do w_2 oraz z wierzchołka w_2 istnieje trasa do w_1.



Rys. 1: Przykładowy graf z zaznaczonymi silnie spójnymi składowymi

W książce [18] można znaleźć algorytm podziału grafu na silnie spójne składowe, którego złożoność czasowa wynosi $O(n+m)$. Po wyznaczeniu takich składowych, możemy graf G „skompresować" tworząc graf G', którego wierzchołkami są obliczone składowe; ponadto w grafie G' istnieje krawędź z wierzchołka v_1 do v_2 (dla $v_1 \neq v_2$), jeżeli w grafie G istniały połączone krawędzią wierzchołki w_1 i w_2, należące do silnie spójnych składowych odpowiadających v_1 i v_2. Konstrukcja G' ściśle zgodna z powyższą definicją nie jest prosta do wykonania w czasie $O(n+m)$ — trudno wykryć i wyeliminować krawędzie wielokrotne. Jednakże w naszym przypadku usuwanie ich nie jest konieczne, a wręcz mogłoby prowadzić do błędnych wyliczeń liczby ścieżek. Dlatego możemy skorzystać z szybkiej wersji algorytmu konstrukcji grafu skompresowanego, w wyniku której otrzymamy graf G', w którym z wierzchołka v_1 do v_2 prowadzi tyle krawędzi, ile jest w grafie G różnych krawędzi (w_1, w_2) takich, że w_1 należy do składowej odpowiadającej v_1, a w_2 — do składowej v_2.



Rys. 2: Graf silnie spójnych składowych, odpowiadający grafowi z poprzedniego rysunku

Zauważmy, że G' jest grafem acyklicznym (grafy acykliczne skierowane często oznaczamy skrótem DAG). Gdyby bowiem zawierał cykl $v_1 \to v_2 \to \ldots \to v_k \to v_1$, to z każdego wierzchołka należącego do składowej v_1 dałoby się dojść do każdego wierzchołka ze składowej v_2 i odwrotnie. A to oznaczałoby, że v_1 i v_2 tworzyłyby jedną silnie spójną składową.

Odnotujmy także jeszcze jedno istotne spostrzeżenie. Jeżeli silnie spójna składowa grafu G zawiera co najmniej jedną krawędź łączącą jej wierzchołki (w szczególności może być to pętla łącząca wierzchołek z samym sobą), to każdy wierzchołek należący do tej składowej należy do pewnego cyklu w grafie G. Z drugiej strony, wierzchołki grafu G nienależące do żadnego cyklu tworzą jednoelementowe silnie spójne składowe, niezawierające żadnych krawędzi grafu G.

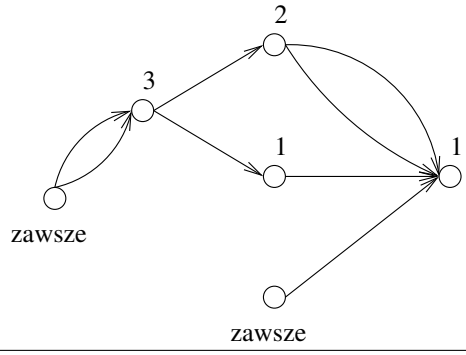
Wierzchołki skończone i nieskończone

Załóżmy, że mamy już zbudowany graf G'. Korzystając z poczynionych spostrzeżeń możemy stwierdzić, że każdy niezerowy wierzchołek grafu G należący do silnie spójnej składowej zawierającej przynajmniej jedną krawędź, należy do jakiegoś cyklu, z którego można dojść do gmachu, a tym samym jest wierzchołkiem nieskończonym.

Aby wyznaczyć liczbę ścieżek prowadzących do gmachu dla wszystkich pozostałych wierzchołków będziemy przeglądać wierzchołki G' w kolejności odwrotnej do topologicznej. Oznacza to, że przed przejrzeniem wierzchołka v grafu G' musimy odwiedzić wszystkie wierzchołki, do których prowadzą z v krawędzie — jest to możliwe, gdyż G' jest acykliczny. Pierwsza w tym porządku jest składowa zawierająca gmach uniwersytetu. Jeżeli zawiera ona krawędź, to z gmachu (oraz ze wszystkich wierzchołków tworzących z nim silnie spójną składową) do gmachu możemy przejść na nieskończenie wiele sposobów. W przeciwnym razie istnieje dokładnie jedna ścieżka (długości zero) prowadząca z gmachu do gmachu i składowa gmachu składa się z tego jedynego wierzchołka.

Obliczmy liczbę ścieżek dla pozostałych wierzchołków G' w wyznaczonym porządku. Jeśli kolejny wierzchołek jest składową zawierającą krawędź, to znamy odpowiedź — wszystkie wierzchołki grafu G tworzące tę składową są wierzchołkami nieskończonymi. W przeciwnym razie składowa zawiera dokładnie jeden wierzchołek grafu G i liczba tras łączących go z gmachem jest równa sumie wyznaczonych wyników dla wszystkich wierzchołków G', do których prowadzą krawędzie z tego wierzchołka (zauważmy, że w rozumowaniu tym istotne jest pozostawienie wielokrotnych krawędzi łączących różne silnie spójne składowe). Jeżeli wynikiem dla któregokolwiek z tych wierzchołków jest

nieskończoność, to i cała suma jest nieskończona. Jeżeli suma jest skończona, ale przekracza 36 500, to możemy przyjąć, że wynik także wynosi nieskończoności (mamy bowiem do czynienia z wierzchołkiem skończonym dużym, którego nie musimy odróżniać od nieskończonego).



Rys. 3: Poprzedni graf z wynikami wyznaczonymi opisaną metodą

Pierwszy algorytm

Jesteśmy już gotowi do zapisania pseudokodu pierwszego rozwiązania zadania:

```
1:  function Profesor(g,G)
2:  { Wyznacza wynik dla każdej silnie spójnej składowej grafu G. }
3:  begin
4:    { Wyznaczanie wierzchołków zerowych. }
5:    Wyznacz graf odwrócony G^T;
6:    visited := BFS(g,G^T); { Przeszukaj graf G^T, zaczynając od gmachu. }
7:    { Właściwy algorytm, w którym będziemy eliminować wierzchołki zerowe. }
8:    Wyznacz graf silnie spójnych składowych G';
9:    foreach v in G' w kolejności odwrotnej do topologicznej do
10:   begin
11:     wynik[v] := 0;
12:     if (∃w ∈ v : not visited[w]) then
13:       continue; { Wierzchołek zerowy w zawarty w składowej v. }
14:     if (silnie spójna składowa v zawiera krawędź) then
15:       wynik[v] := ∞;
16:     else
17:     begin
18:       foreach e = (v,w) ∈ G' do
19:         wynik[v] := wynik[v] + wynik[w]; { a + ∞ = ∞ + a = ∞ }
20:       if (wynik[v] > 36500) then
21:         wynik[v] := ∞;
22:     end
23:   end
24:   return wynik;
25: end
```

Przeanalizujmy złożoność czasową przedstawionego rozwiązania. Jak już pisaliśmy, kroki algorytmu w wierszach 4-8 można wykonać w czasie $O(n+m)$. W głównej pętli algorytmu (instrukcje 9-23) dla każdej silnie spójnej składowej rozważamy wszystkie krawędzie wychodzące z niej w grafie G' (każdą tylko raz) — liczba przeanalizowanych krawędzi będzie więc co najwyżej taka, jak liczba krawędzi w grafie G (może być mniejsza, gdyż pomijamy krawędzie wychodzące z wierzchołków zerowych). Ostatecznie pętla ma więc złożoność $O(n+m)$ i taka jest też złożoność czasowa całego algorytmu.

Drugie rozwiązanie

Pierwsze rozwiązanie prezentuje ogólny schemat postępowania stosowany w przypadku rozwiązywania wielu problemów grafowych — z grafu konstruujemy graf silnie spójnych składowych, a następnie wykorzystujemy jego acykliczność przeglądając wierzchołki „grupami". Niestety konstrukcja tego grafu jest stosunkowo skomplikowana, zatem poniżej przedstawimy inne rozwiązanie — ideologicznie podobne, acz nie odwołujące się do pojęcia silnie spójnych składowych.

Przypomnijmy, że potrafimy łatwo wyznaczyć wierzchołki zerowe grafu, więc w dalszych rozważaniach zakładamy, że ich w grafie nie ma (przypomnimy sobie o nich dopiero w pseudokodzie rozwiązania). Zastanówmy się teraz, jak wygląda część grafu zawierająca wyłącznie wierzchołki skończone. Musi to być oczywiście graf acykliczny — z żadnego wierzchołka skończonego nie można dojść do żadnego cyklu. Aby go zlokalizować, możemy rozpocząć od wierzchołków o zerowym stopniu wyjściowym (czyli takich, z których nie wychodzi żadna krawędź). Oczywiście należą one do poszukiwanego grafu — usuwamy je więc z oryginalnego grafu wraz z prowadzącymi do nich krawędziami. Jako kolejne usuwamy (wraz z krawędziami wchodzącymi) wierzchołki, z których krawędzie prowadziły jedynie do już wybranych, czyli wierzchołki mające w tej chwili zerowy stopień wyjściowy. Iterujemy opisany proces do momentu, gdy w grafie nie będzie już żadnego wierzchołka o stopniu wyjściowym równym 0. W ten sposób w grafie pozostaną jedynie wierzchołki nieskończone.

Do uzasadnienia poprawności powyższej procedury klasyfikacji wierzchołków wystarczą następujące spostrzeżenia.

- Po pierwsze w trakcie procesu nie usuniemy z grafu wierzchołka w, jeśli jest on nieskończony. Z każdego wierzchołka nieskończonego istnieje bowiem ścieżka do jakiegoś cyklu, a my naszą procedurą nie jesteśmy w stanie usunąć z grafu żadnego cyklu — który wierzchołek miałby zostać usunięty jako pierwszy? Nie możemy także usunąć żadnego wierzchołka należącego do ścieżki prowadzącej od w do cyklu, bo każdy wierzchołek na tej ścieżce ma cały czas krawędź wychodzącą.

- Po drugie pokażmy, że algorytm usuwa z grafu wszystkie wierzchołki skończone. W tym celu zauważmy, że każdy nieusunięty wierzchołek w ma co najmniej jedną krawędź wychodzącą. Wobec tego startując z wierzchołka w i przechodząc po dowolnych krawędziach w końcu dojdziemy do wierzchołka, w którym już w trakcie tego „spaceru" byliśmy — wynika to z zasady szufladkowej Dirichlet'a. To dowodzi, że z wierzchołka w doszliśmy do pewnego cyklu, czyli że wierzchołek ten jest nieskończony.

Warto zauważyć, że powyższy proces rozpoznawania wierzchołków skończonych jest analogiczny do sortowania topologicznego stosowanego również w pierwszym algorytmie. Jest on jednak przeprowadzany bardziej „elementarnymi" środkami, stąd tak istotny jest powyższy dowód pozwalający przekonać się o jego poprawności.

Drugi algorytm

Możemy teraz dokładnie zapisać algorytm. Będziemy w nim sukcesywnie obliczać wartości tablicy *wynik*, zawierającej wykrytą dla danego wierzchołka liczbę ścieżek prowadzących z niego do gmachu. Na początku przypiszmy gmachowi uniwersytetu wynik równy 1 (być może powinien on być równy nieskończoność, ale do tego jeszcze dojdziemy — na razie znamy tylko jedna ścieżkę), a wszystkim pozostałym wierzchołkom przyporządkujmy tymczasowo 0. Znajdowanie wszystkich wierzchołków skończonych wykonamy za pomocą kolejki — najpierw umieścimy w niej wszystkie wierzchołki, z których początkowo nie wychodzi żadna krawędź. Potem wyjmując wierzchołek z kolejki będziemy usuwać go z grafu i jednocześnie sprawdzać, czy wskutek tego żaden nowy wierzchołek nie powinien do kolejki trafić. Ponadto będziemy zwiększać wartości *wynik* dla wszystkich wierzchołków, z których prowadzi do niego jakakolwiek krawędź, o liczbę dróg prowadzących przez usuwany wierzchołek. Ponieważ w czasie usuwania wierzchołków i aktualizacji liczby ścieżek przeglądamy graf w porządku odwrotnym do „topologicznego" (pamiętając, że nasz graf może zawierać cykle, a więc porządek topologiczny nie jest w nim właściwie zdefiniowany), więc cały czas utrzymujemy graf odwrócony G^T. Po obliczeniu wyniku dla wierzchołków skończonych w grafie pozostają wierzchołki zerowe — nieosiągalne z gmachu uniwersytetu, oraz wierzchołki nieskończone — nierozważane w trakcie znajdowania wierzchołków skończonych.

Po tak dokładnym opisie możemy przedstawić implementację rozwiązania:

```
1:  function Profesor2(g,G)
2:  { Wyznacza wynik dla każdego wierzchołka grafu G. }
3:  begin
4:    { Wyznaczanie wierzchołków zerowych. }
5:    Wyznacz graf odwrócony G^T;
6:    visited := BFS(g,G^T); { Przeszukaj graf G^T, zaczynając od gmachu. }
7:    { Inicjacja zmiennych. }
8:    foreach v in G do
9:      wynik[v] := 0;
10:   wynik[g] := 1;
11:   rozwazone := ∅;
12:   kol := ∅ { Prosta kolejka FIFO. }
13:   foreach v in G do
14:     if outdeg[v] = 0 then
15:       Insert(v,kol);
16:   { Główna pętla algorytmu }
17:   while (kol ≠ ∅) do
18:   begin
19:     v := Pop(kol);
20:     Insert(v,rozwazone);
```

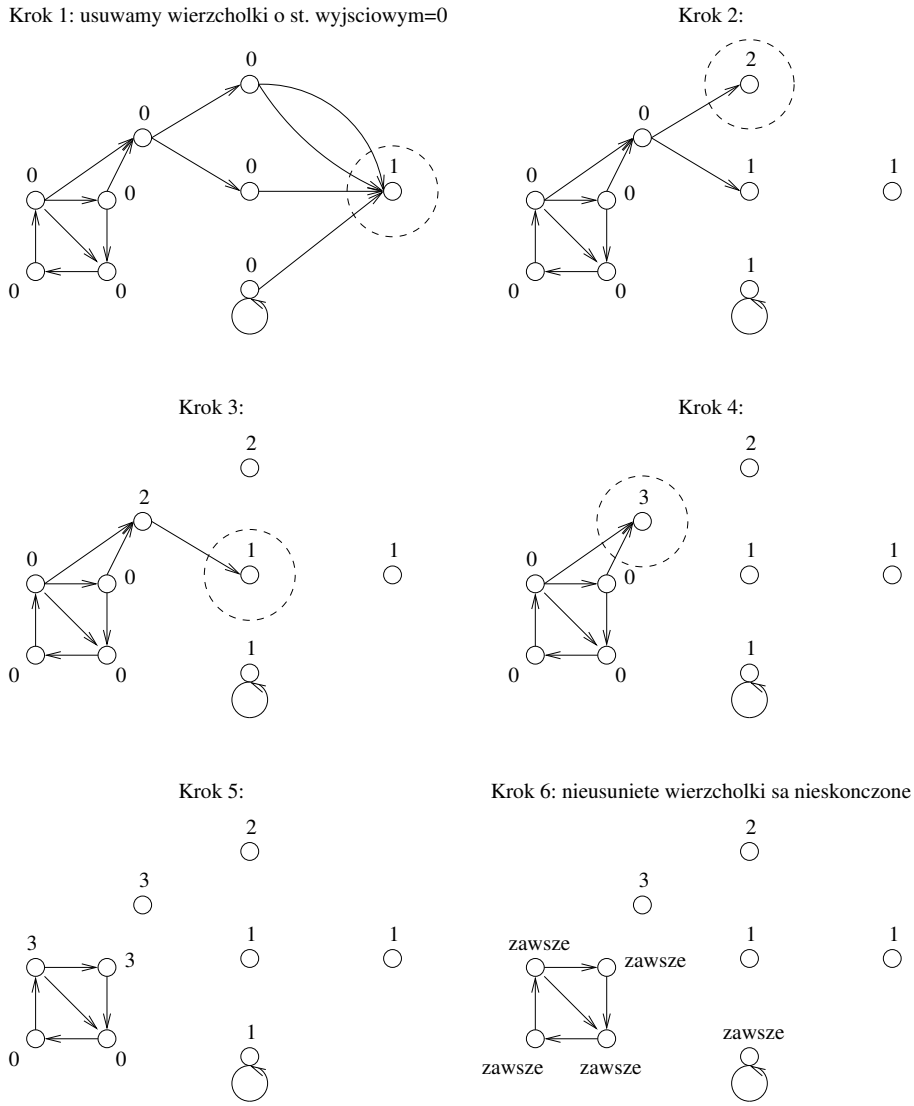
```
21:       if (not visited[v]) then
22:         continue; { Nie rozważamy wierzchołków zerowych. }
23:       foreach e = (u,v) ∈ G do
24:       begin
25:         wynik[u] := wynik[u] + wynik[v];
26:         if (wynik[u] > 36500) then
27:           wynik[u] := ∞;
28:         Usuń krawędź e z grafu G;
29:         if (outdeg[u] = 0) then
30:           Insert(u,kol);
31:       end
32:     end
33:     { Nierozważone wierzchołki są nieskończone. }
34:     foreach v in G do
35:       if (v ∉ rozwazone) then
36:         { Wszystkie wierzchołki zerowe znajdują się w zbiorze rozwazone. }
37:         wynik[v] := ∞;
38:     return wynik;
39: end
```

Pozostało nam jeszcze oszacować czas działania zaprezentowanego rozwiązania. Łatwo zauważyć, że wszystkie kroki poza główną pętlą (wiersze 16-32) mają złożoność mieszczącą się w ograniczeniu $O(n+m)$. W głównej pętli także rozważamy każdy wierzchołek i każdą krawędź co najwyżej raz, cały algorytm ma więc rzeczywiście złożoność czasową $O(n+m)$.

Powyższe rozwiązanie było rozwiązaniem wzorcowym; jego implementacja znajduje się na dysku dostarczonym do książeczki, a przykład jego działania można zobaczyć na rysunku 4.

Gorsze i błędne rozwiązania

Nieoptymalne wykonanie któregokolwiek kroku algorytmu zazwyczaj prowadziło do rozwiązania o złożoności $O(n \cdot (n+m))$. Nie był to jednak główny powód utraty punktów przez zawodników. Podstawową trudnością była poprawna implementacja rozwiązania z uwzględnieniem wszystkich przypadków szczególnych (na przykład istnienie cyklu zawierającego gmach czy jednoczesne występowanie wierzchołków nieskończonych i skończonych dużych). Dodatkowym utrudnieniem był limit pamięciowy równy 32 MB, który wymuszał oszczędne gospodarowanie pamięcią.



Rys. 4: Działanie rozwiązania drugiego na przykładowym grafie

Testy

Zadanie testowane było na zestawie 10 danych testowych. W poniższej tabeli zamieszczone są przybliżone wartości parametrów dla testów (liczby wierzchołków i krawędzi) oraz krótkie charakterystyki zawartych w nich grafów. Tylko dla trzech punktowanych testów (o numerach 1, 6 i 10) odpowiedzią nie było „`zawsze`”.

Nazwa	n	m	Opis
pro1.in	10	13	prosty graf bez cykli
pro2.in	30	53	dwa rozłączne cykle, wierzchołek skończony i acykliczny podgraf z więcej niż 36 500 trasami do gmachu
pro3.in	22	42	cykl z zerowych wierzchołków, niezerowe wierzchołki tworzące graf acykliczny, niezerowy cykl
pro4.in	100	601	pętla w gmachu, połowa wierzchołków zerowa
pro5.in	13 000	48 500	mały cykl i jeden wierzchołek z wynikiem 36 501
pro6.in	80 000	96 000	graf acykliczny z długimi trasami i kilkoma dodatkowymi krawędziami
pro7.in	100 000	1 000 000	dosyć gęsty graf acykliczny oraz cykle: z wierzchołków zerowych i niezerowych
pro8.in	10 000	1 000 000	losowy, bardzo gęsty graf acykliczny i cykl
pro9.in	1 000 000	1 000 000	prawie losowy test
pro10.in	1 000 000	1 000 000	bardzo duży graf acykliczny i cykle z wierzchołków zerowych