Informe
Matemática Discreta I
Number of Simple Paths
Problema:1454E

Abel Molina Sánchez
Grupo 2-11
Ciencias de la Computación
Universidad de La Habana

26 de diciembre de 2020

1. Problema

E. Number of Simple Paths

You are given an undirected graph consisting of n vertices and n edges. It is guaranteed that the given graph is connected (i. e. it is possible to reach any vertex from any other vertex) and there are no self-loops and multiple edges in the graph.
Your task is to calculate the number of simple paths of length at least 1 in the given graph. Note that paths that differ only by their direction are considered the same (i. e. you have to calculate the number of undirected paths). For example, paths $[1, 2, 3]$ and $[3, 2, 1]$ are considered the same.
You have to answer t independent test cases.
Recall that a path in the graph is a sequence of vertices $v_1, v_2, \ldots, v_k$ such that each pair of adjacent (consecutive) vertices in this sequence is connected by an edge. The length of the path is the number of edges in it. A simple path is such a path that all vertices in it are distinct.

Input:
The first line of the input contains one integer t ($1 \leq t \leq 2 * 10^4$) — the number of test cases. Then t test cases follow.
The first line of the test case contains one integer n ($3 \leq n \leq 2 * 10^5$) — the number of vertices (and the number of edges) in the graph.
The next n lines of the test case describe edges: edge i is given as a pair of vertices
u_i, v_i ($1 \leq u_i, v_i \leq n$, $u_i \neq v_i$), where u_i and v_i are vertices the $i-th$ edge connects. For each pair of vertices (u, v), there is at most one edge between u and v. There are no edges from the vertex to itself. So, there are no self-loops and multiple edges in the graph. The graph is undirected, i. e. all its edges are bidirectional. The graph is connected, i. e. it is possible to reach any vertex from any other vertex by moving along the edges of the graph.
It is guaranteed that the sum of n does not exceed $2 * 10^5$ ($\sum n \leq 2 * 10^5$).

Output:
For each test case, print one integer: the number of simple paths of length at least 1 in the given graph. Note that paths that differ only by their direction are considered the same (i. e. you have to calculate the number of undirected paths).

2. Interpretación e idea

En el problema se pide determinar el número de caminos simples en un grupo de grafos conexos no dirigidos que tienen igual cantidad de vértices y aristas.

La idea de la solución pasa por determinar que el grafo contiene un único ciclo simple, y en base a esto vamos a buscar la fórmula con la cual determinar el número de caminos. Para esto, tenemos en cuenta que los vértices que no pertenecen al ciclo inducirán árboles junto al vértice del ciclo con el cual comparten aristas. O sea, que de quitar las aristas del ciclo, lo que quedan son componentes conexas que constituyen árboles. Luego vamos a determinar la cantidad de caminos simples que existen en un árbol y luego buscaremos todos los caminos que incluyen las aristas del ciclo. Esto en el sentido teórico. Una vez demostrado, en la implementación realizaremos una búsqueda en profundiad en el grafo para determinar el ciclo que contiene. Luego recorreremos

cada una de las 'componentes' descritas anteriormente con otra búsqueda en profundidad para determinar su cantidad de vértices; y luego aplicaremos la fórmula para determinar la cantidad de caminos simples. La fórmula la hallaremos en el desarrollo de la demostración del problema a continuación.

3. Demostración

Lema1: *Un grafo G es conexo $\Leftrightarrow$ admite un árbol abarcador.*
Demostración:
($\Leftarrow$) Sea T un árbol abarcador de G. Sean u, v vértices de G. En T existe un camino p de u a v. Como T es subgrafo de G, p será también un camino en G de u a v. Por lo tanto G es conexo.
($\Rightarrow$) Por inducción en la cantidad de vértices de G. Si G es trivial G es árbol abarcador de si mismo. Supongamos que se cumple para todo grafo G conexo con n vértices. Sea G conexo con $n+1$ vértices. Seleccionemos un vértice v de G, extraemos v con todas sus aristas incidentes, si $G - \{v\}$ es conexo, entonces, tiene un árbol abarcador por hipótesis, lo seleccionamos, le agregamos v con una de sus aristas y tendremos un árbol abarcador para G. Ahora analicemos cuando v desconecta a G. Sean G_i las k componentes conexas que se forman al extraer v; v tiene aristas con al menos un v_i de cada componente, porque G era conexo. Por hipótesis, sean T_i los árboles abarcadores de cada uno de los G_i. Ahora construyamos T resultado de unir los T_i conectándolos a v a través de una de las aristas (v, v_i). No se forman ciclos porque G_i eran componentes conexas(no hay vértices entre ellas). Ahora $\sum |V[T_i]| = n$ porque están todos los vértices de G excepto v, luego al formar T se tiene $|V[T]| = n + 1$.
Ahora $\sum_i^k |E[T_i]| = \sum_i^k |V[T_i]| - 1 = n - k$, por lo cual luego de conectar los T_i a v a través de las aristas (v, v_i) se agregan k aristas por lo cual $|E[T]| = n - k + k = n$, luego T es un árbol abarcador de G. Con lo cual se completa la demostración por inducción matemática.

Teorema1: *Un grafo $G = (V, E)$ no dirigido conexo con $|V| = n$ y $|E| = n$ ($n \geq 3$) tiene un único ciclo.*
Demostración: Sea T un árbol abarcador de G(Lema1), T tiene n vértices y $n - 1$ aristas. Sea $e = (u, v)$ la arista de G que no está en T. Sea p el camino de u a v en T. $p \cup \{(u, v)\}$ forman un ciclo.
Supongamos que hay otro ciclo C en G. Si C no contiene a e, entonces C forma parte de T lo cual contradice que T es un árbol. Luego si C contiene a e, C está formado por e en unión con un camino entre u y v en T, sea q este camino. $q = p$ porque de lo contrario habría dos caminos, p y q entre u y v en T, lo cual contradice que T sea un árbol.

Sea ahora el grafo G inicial, sea $C = \{c_1, c_2, ..., c_k, c_1\}$ el ciclo de G, C es simple ya que es único en G. Se tiene que:
G tiene n vértices y n aristas.
C tiene k vértices y k aristas
$G - C$ tiene $n - k$ vértices y $n - k$ aristas.
Sean C_i las componentes conexas que se froman al eliminar las k aristas del ciclo($c_i \in V[C_i]$ ya que los vértices del ciclo se conectan entre ellos solo a través de aristas del ciclo, ya que de lo contrario,

existiría algún camino que conectara a un c_i con un c_j que unido a uno de los caminos del ciclo crearía otro ciclo lo cual es una contradicción).
Demostremos que C_i son árboles. No existen cilcos en C_i porque esto implicaría la existencia de un ciclo en G, lo cual contradice que C sea el único ciclo en G. Luego como C_i es conexo y no posee ciclos, C_i es un árbol.

Luego, vamos a buscar la fórmula para calcular la cantidad de caminos en G, llamemos S a esta cantidad. Tenemos que:
$S = S_1 + S_2$
donde S_1 será la cantidad de caminos que no incluyen aristas del ciclo C, y S_2 la cantidad que incluyen aristas del ciclo C.

Busquemos S_1 apoyados en el siguiente Lema:

Lema2: *La cantidad de caminos simples diferentes en un árbol T no dirigido con n vértices es $n(n-1)/2$.*
Demostración: Hagamos inducción en la cantidad de vértices. Si $|V[T]| = 1$, no hay caminos y se tiene que $0 = 1(0)/2$, por lo cual se cumple para $|V[T]| = 1$. Supongamos por hipótesis que se cumple para $n = k$, esto es en un árbol con $|V[T]| = k$, hay $k(k-1)/2$ caminos simples diferentes. Ahora demostremos que se cumple para $k+1$. Sea T un árbol de k vértices si agregamos un vértice v por una arista (v, t_i), $t_i \in V[T]$, $T' = T + \{v\}$ es un árbol con $k+1$ arsitas ya que: la ocurrencia de un ciclo en T' implica la ocurrencia de un ciclo en T porque v tiene una única arista incidente con lo cual no puede fomar parte de un ciclo. Luego T' es acíclico y al agregar un vértice y T' es conexo ya que como T es conexo hay un camino que une a cualquier vértice con t_i, ese camino unido a la arista (t_i, v_i) es un camino desde esos vértices hasta v. Ahora, al agregar v, tenemos k nuevos caminos en T', los que unen a v con cada uno de los k restantes vértices. Luego, tenemos:

$$\begin{aligned} \frac{k+1(k)}{2} &=^{?} \frac{k(k-1)}{2} + k \\ &=^{?} \frac{(k^2 - k + 2k)}{2} \\ &=^{?} \frac{(k^2 + k)}{2} \\ &= \frac{(k+1)k}{2} \end{aligned} \tag{1}$$

Luego $k \Rightarrow k+1$ hemos demostrado el lema por inducción matática.

Luego tenemos que la cantidad de caminos que no incluyen aristas del ciclo será:

$$S_1 = \sum_{1}^{k} \frac{|V[C_i]|(|V[C_i]|-1)}{2}$$

Ahora vamos a buscar la cantidad de caminos que incluyen aristas del ciclo:

Lema4: *Todos los caminos que unen a vértices de C_i con vértices de C_j pasan por c_i.*
Demostración: Sean v_i y v_j dos vértices conectados a través de caminos que no incluyen a c_i y c_j, luego como existen los caminos $v_i \rightsquigarrow c_i \rightsquigarrow \ c_j \rightsquigarrow v_j$ habrían dos caminos en el grafo que los conectan, lo cual implica la existencia de un ciclo y como v_i,$v_j \notin C$, sería un ciclo distinto de C lo cual contradice que C es el único ciclo del grafo.

Lema5: *Sean c_i y c_j dos vértices de C, existen dos caminos y solo dos caminos.*
Demostración: Demostremos que existen dos caminos y luego que son únicos. Que existen dos caminos es directo del hecho de que forman parte del ciclo C, elegimos c_i como vértice inicial y tendremos un camino de $c_i \rightsquigarrow c_j$ con un grupo de vértices de C y un camino de $c_j \rightsquigarrow c_i$ con los restantes. Ahora supongamos que existiera un tercer camino distinto de los anteriores, llamémosle q, si q es disjunto de p_1 y p_2(los dos caminos del ciclo), entonces, con q y uno de los dos se forma otro ciclo lo cual contradice la unicidad de C. Ahora supongamos que tuvieran intersección en algún vértice, sin pérdida de generalidad digamos que en $v \in p_1$, luego tendríamos un ciclo formando por el subcamino de p_1 de c_i a v y otro con el subacamino de q de c_i a v lo cual contradice la unicidad de C.

Teorema2: *Para cada vértice de $v_i \in C_i$ existen dos caminos que lo unen a los vértices de $G - \{C_i\}$*
Demostración: Por resultado directo de los lemas 4 y 5 tenemos que cada camino desde los v_i a un C_j pasa contiene a c_i y a c_j y que entre c_i y c_j hay exactamente 2 caminos. Sean estos caminos p_1 y p_2, tenemos que para los v_i:
los caminos $v_i \rightsquigarrow c_i \rightsquigarrow^{p_1} c_j \rightsquigarrow v_j$ y $v_i \rightsquigarrow c_i \rightsquigarrow^{p_2} c_j \rightsquigarrow v_j$ (posible $c_i = v_i$, $c_j = v_j$).

Luego vamos a determinar el valor de S_2 :
Para cada $v_i \in C_i$ tenemos 2 caminos que los conectan con los vértices de $G - \{C_i\}$, entones la cantidad de caminos que empiezan en v_i y que contienen aristas del ciclo serán $2(|V[G - \{C_i\}]|)$
Luego como esto se mantiene para v_i que pertenece a los C_i por fórmula general para todos los caminos que no contienen aristas de C se tendrá:
$\sum_{1}^{k} 2|V[C_i]|(|V[G - \{C_i\}]|)$,
pero como se tiene que el grafo es no dirigido y un camino de v_i a v_j es igual a un camino entre v_j a v_i con los mismos vértices, se estaría contando doble el número de caminos, por lo cual se llega a que:

$$\begin{aligned} S_2 &= \frac{\sum_{1}^{k} 2|V[C_i]|(|V[G - \{C_i\}]|)}{2} \\ &= \sum_{1}^{k} \frac{2|V[C_i]|(|V[G - \{C_i\}]|)}{2} \\ &= \sum_{1}^{k} |V[C_i]|(|V[G - \{C_i\}]|) \end{aligned} \tag{2}$$

Luego el total de caminos en el grafo G será:

$$
\begin{aligned}
S &= S_1 + S_2 \\
&= \sum_1^k \frac{|V[C_i]|(|V[C_i]|-1)}{2} + \sum_1^k |V[C_i]|(|V[G-\{C_i\}]|) \\
&= \sum_1^k \frac{|V[C_i]|(|V[C_i]|-1)}{2} + |V[C_i]|(|V[G-\{C_i\}]|)[1]
\end{aligned}
\tag{3}
$$

4. Implementación

```
solveTest(G):
    visited1 = [False foreach v in V[G]]
    visited2 = [False foreach v in V[G]]
    stack = Stack()
    C = {}
    detect_cycle_dfs(G,0,0)
    S = 0
    foreach ci in C:
        cant_Ci = 0
        count_dfs(G,ci)
        S += ( cant_Ci(cant_Ci-1)/2 + cant_Ci(|V[G]|-cant_Ci)
    return S
```

```
detect_cycle_dfs(G,v,pv):
    visited1[v] = True
    stack.push(v)
    foreach u in G[v]:
        if visited1[u] and u!=pv:
            while stack.peek()!= u:
                ci = stack.pop()
                visited2[ci] = True
                C = C + {ci}
            visited2[u] = True
            C = C + {u}
            return True
        else if not visited[u]:
            if detect_cycle_dfs(G,u,v)
                return True
    stack.pop()
    return False
```

```
|count_dfs(G,v,pv):
|    cant_Ci++
|    visited2[v] = True
|    foreach u in G[v]:
|        if not visited2[u]:
|            count_dfs(G,u,v)
```

Ahora demostremos que el algoritmo determina la cantidad de caminos simples en el grafo G. Para hello debemos demostrar las modificaciones agregadas al algoritmo dfs, primero para determinar el ciclo en G y luego para determinar la cantidad de vértices en cada C_i.

Primero analizaremos el correcto funcionamiento del método *detect-cycle-dfs*. Tenemos que demostrar que que luego de terminado, en C se encuentran los vértices del ciclo.
Para ello:

Lema6: *Encontrar una arista de retroceso en el recorrido dfs de un grafo G no dirigido implica la existencia de un ciclo.*
Demostración: Si se encuentra un arista de retroceso en el recorrido, digamos la arista de (u, v) durante el anális del vértice u, entonces se tendría a v como ancestro de u en el árbol dfs, con lo cual se tiene un camino de v a u en G, que unido a la arista (u, v) conforman el ciclo.

Luego, como hemos visto que el grafo G contendrá a C como único ciclo, al detectar la arista de retroceso, que se garantiza en la condición *if visited1[u] and u!=pv:* habremos detectado el ciclo del grafo.
Ahora necesitamos probar que dentro de la condición *if visited1[u] and u!=pv:* agregamos a C los vértices del ciclo. Esto lo vemos por el hecho de que en el momento en el que un vértice es descubierto (esto es, al momento en el que se realiza una llama recursiva con él como vértice), su padre en el recorrido está en el tope de la pila. Y esto se garantiza, ya que cada vértice se agrega a la pila antes de iterar por su lista de adyacencia, al comienzo del método, y luego de analizado, es extraído, con lo cual se mantiene el hecho que durante el ciclo for del vértice v_i este se ecuentra en el tope de la cola, con lo cual en el llamado a un vértice descendiente, que ocurre durante el ciclo esto se mantiene y por tanto cuando el vértice es descubierto su padre está en el tope de la pila. Luego, y como se mantiene la invariante de que el vértice está en el tope de la cola durante su ciclo for al momento de detectar la arista de retroceso durante el análisis del vértice v, en la cola estará v y sus ancestros de forma continua, en particular también estará u que es el que cierra el ciclo, con lo cual, luego de iterar la pila hasta el vértice u y agregar este, se habrán agregado los vértices del ciclo a C.

Ahora, una vez detectados los vértices del ciclo, se garantiza también que en el arreglo $visited2$ estos quedan marcados en $True$. Luego como cada $c_i \in C$ posee solo aristas con c_j, c_k otros dos vértices de C y con vértices de C_i, al marcar los vértices del ciclo como visitados se garantiza que cuando se inicializa el segundo recorrido en profunidad(*count-dfs*) con c_i, solo se visitarán los vértices de C_i(hemos visto anteriormente que todos los caminos entre $v_i \in C_i$ y $v_j \in C_j$ contienen a c_i y c_j). Luego se visitarán todos los vértices de C_i y por cada uno se aumenta la $cant_ci$, con lo cual luego de terminada la ejecución $cant - ci = |V[C_i]|$.

Luego se calcula S siguiendo [1].

Ahora analicemos la complejidad temporal del algoritmo de solución para cada caso:
Tenemos que $|V[G]| = |E[G]| = n$. Las inicializaciones son $O(n)$. *detect-cycle-dfs*, en el peor caso se recorre la lista de adyacencia de G completa antes de detectar el ciclo, esto es $O(|V| + |E|)$, que se tiene que es $O(n)$, al momento de hallar el ciclo, agregar los vértices a C es en el peor caso $O(n)$, con lo cual, el método es $O(n)$. Luego en el ciclo for se recorre cada C_i y se tiene que $G = \bigcup C_i$, con lo cual se termina recorriendo el grafo que es $O(n)$. El resto de operaciones se realizan en $O(1)$, con lo cual por regla de la suma se tiene que la solución es $O(n)$ para cada grafo. Luego como se analizan T grafos, la complejidad temporal del algoritmo es $O(T * n)$