TALLER PROGCOMP: TRACK GRAFOS

DIAMETRO DE UN ÁRBOL

Gabriel Carmona Tabja

Universidad Técnica Federico Santa María,
Università di Pisa

November 3, 2024

Part I

ÁRBOL

Árbol

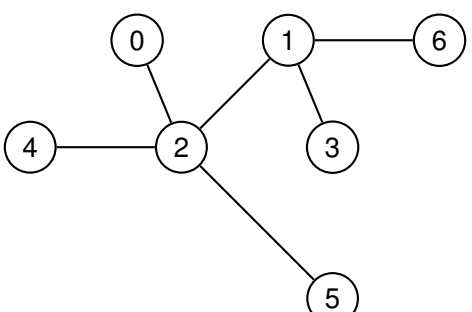
Definición

Grafo no dirigido conexo y sin ciclos.

ÁRBOL

Definición

Grafo no dirigido conexo y sin ciclos.



Part II

DIAMETRO

DIAMETRO

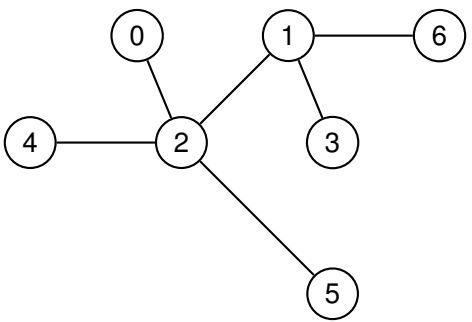
Definición

El largo más grande de un camino que existe en el árbol.

DIAMETRO

Definición

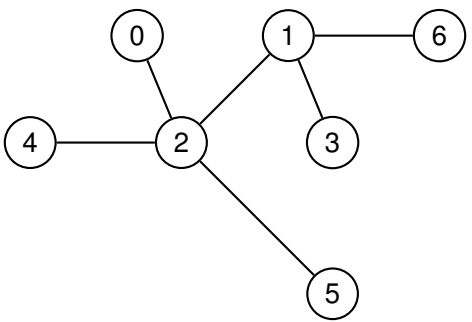
El largo más grande de un camino que existe en el árbol.



DIAMETRO

Definición

El largo más grande de un camino que existe en el árbol.

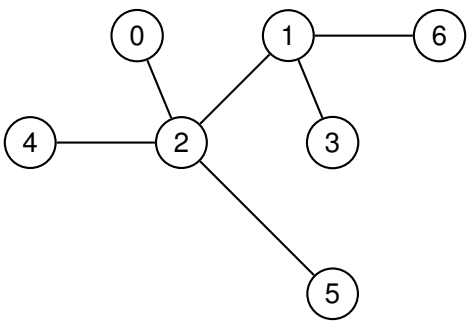


El diametro del árbol es 3.

DIAMETRO

Definición

El largo más grande de un camino que existe en el árbol.



El diametro del árbol es 3.

- Camino: $4 - 2 - 1 - 6$
- Camino: $4 - 2 - 1 - 3$
- Camino: $0 - 2 - 1 - 6$
- Camino: $0 - 2 - 1 - 3$
- Camino: $5 - 2 - 1 - 6$
- Camino: $5 - 2 - 1 - 3$

DIAMETRO

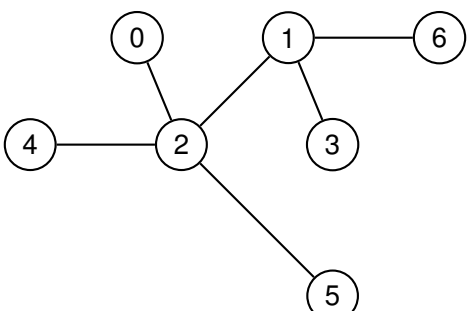
Para calcular, primero vamos a ejecutar un DFS desde algún nodo, digamos 2.

DIAMETRO

Para calcular, primero vamos a ejecutar un DFS desde algún nodo, digamos 2.
Como es un árbol, mantendremos la distancia de llegar del nodo 2 a cualquier nodo del árbol.

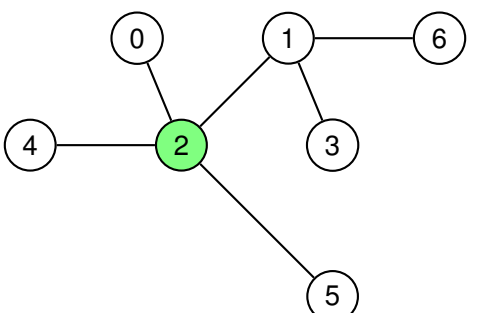
DIAMETRO

Para calcular, primero vamos a ejecutar un DFS desde algún nodo, digamos 2.
Como es un árbol, mantendremos la distancia de llegar del nodo 2 a cualquier nodo del árbol.



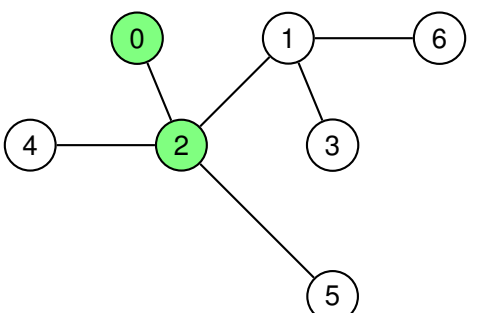
DIAMETRO

Nodo	0	1	2	3	4	5	6
Distancia	_	_	0	_	_	_	_



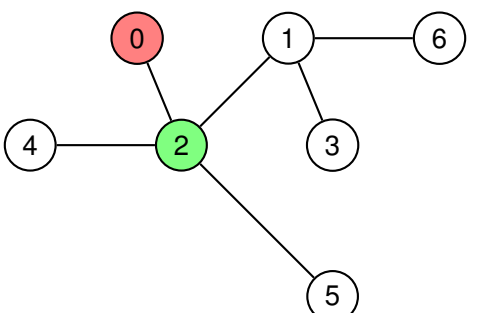
DIAMETRO

Nodo	0	1	2	3	4	5	6
Distancia	1	_	0	_	_	_	_



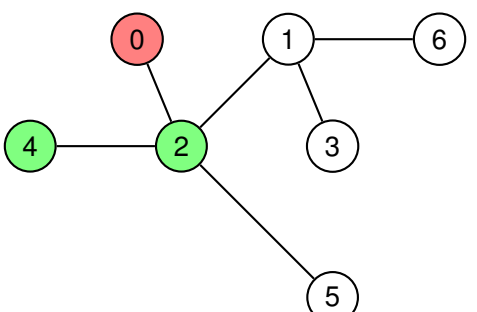
DIAMETRO

Nodo	0	1	2	3	4	5	6
Distancia	1	_	0	_	_	_	_



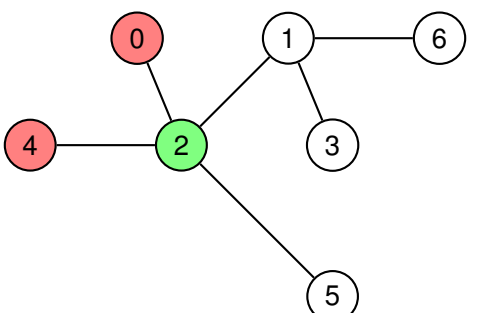
DIAMETRO

Nodo	0	1	2	3	4	5	6
Distancia	1	_	0	_	1	_	_



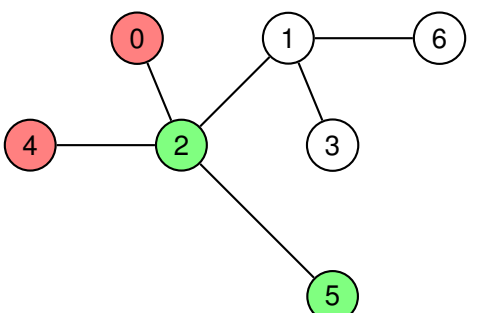
DIAMETRO

Nodo	0	1	2	3	4	5	6
Distancia	1	_	0	_	1	_	_



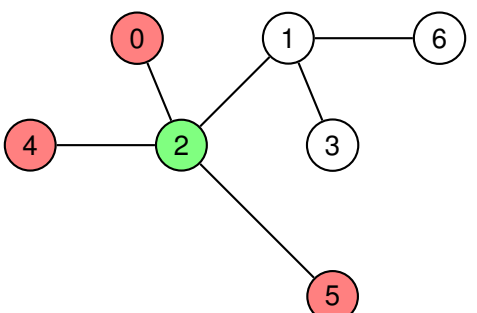
DIAMETRO

Nodo	0	1	2	3	4	5	6
Distancia	1	_	0	_	1	1	_



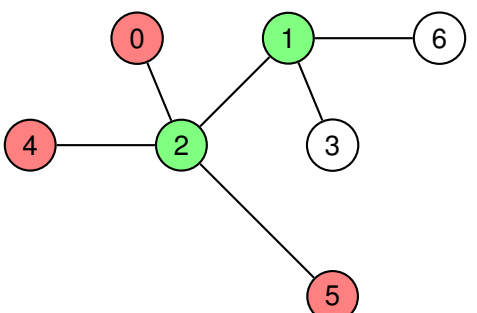
DIAMETRO

Nodo	0	1	2	3	4	5	6
Distancia	1	_	0	_	1	1	_



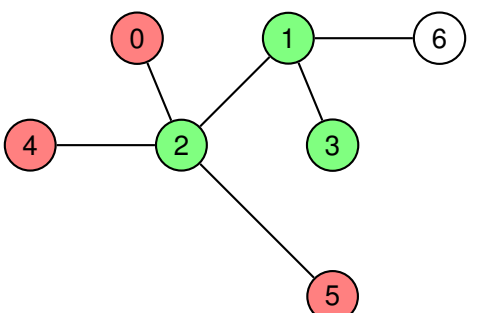
DIAMETRO

Nodo	0	1	2	3	4	5	6
Distancia	1	1	0	_	1	1	_



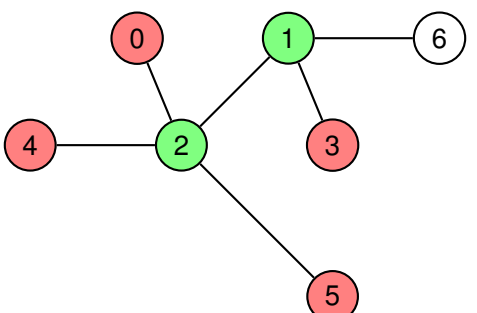
DIAMETRO

Nodo	0	1	2	3	4	5	6
Distancia	1	1	0	2	1	1	_



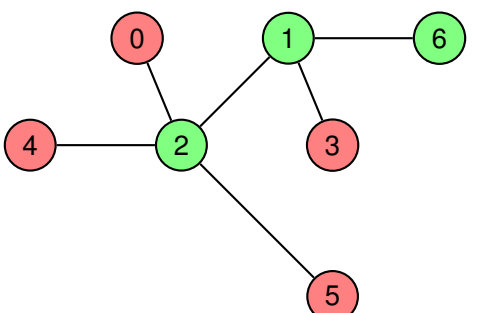
DIAMETRO

Nodo	0	1	2	3	4	5	6
Distancia	1	1	0	2	1	1	_



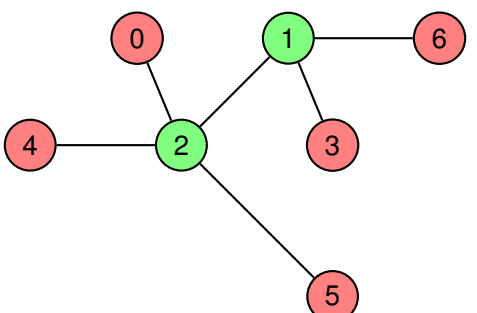
DIAMETRO

Nodo	0	1	2	3	4	5	6
Distancia	1	1	0	2	1	1	2



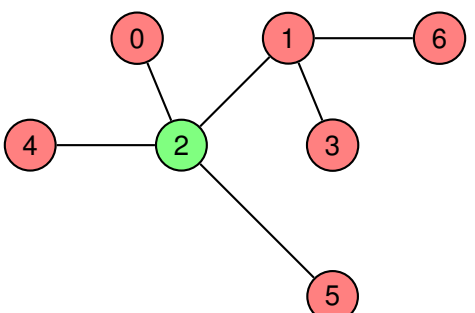
DIAMETRO

Nodo	0	1	2	3	4	5	6
Distancia	1	1	0	2	1	1	2



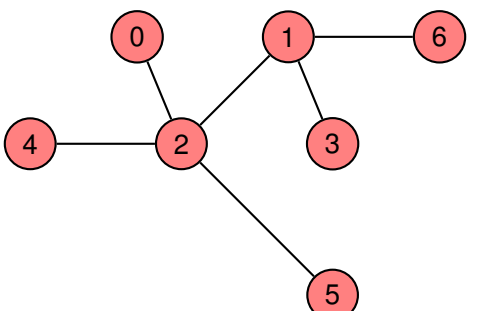
DIAMETRO

Nodo	0	1	2	3	4	5	6
Distancia	1	1	0	2	1	1	2



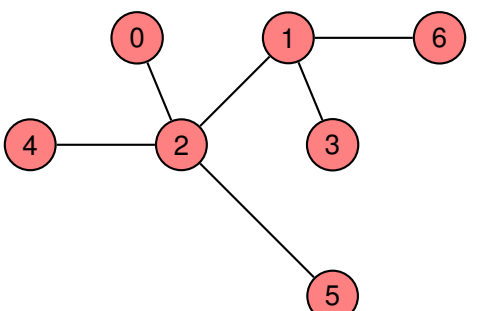
DIAMETRO

Nodo	0	1	2	3	4	5	6
Distancia	1	1	0	2	1	1	2



DIAMETRO

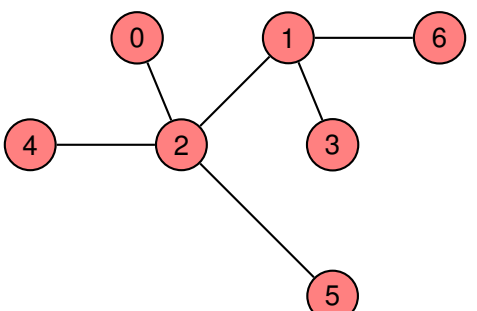
Nodo	0	1	2	3	4	5	6
Distancia	1	1	0	2	1	1	2



Elejimos el nodo que tenga mayor distancia del 2.

DIAMETRO

Nodo	0	1	2	3	4	5	6
Distancia	1	1	0	2	1	1	2

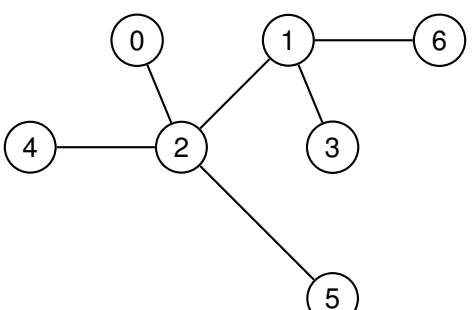


Elejimos el nodo que tenga mayor distancia del 2.
En este caso podemos elegir el 3 o el 6.

DIAMETRO

Ahora ejecutamos el mismo DFS, pero ahora partiendo del nodo elegido. Para el ejemplo digamos que elegimos 6

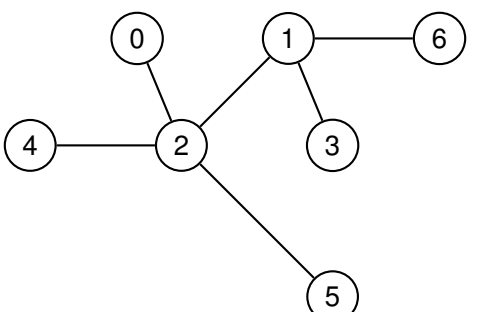
Nodo	0	1	2	3	4	5	6
Distancia	_	_	_	_	_	_	_



DIAMETRO

Ahora ejecutamos el mismo DFS, pero ahora partiendo del nodo elegido. Para el ejemplo digamos que elegimos 6

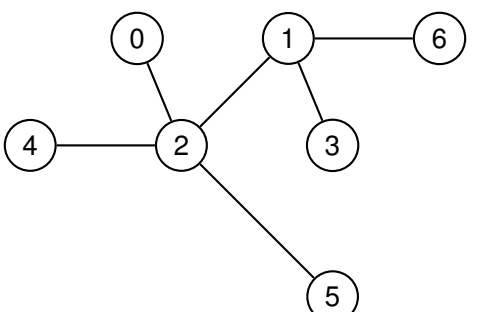
Nodo	0	1	2	3	4	5	6
Distancia	3	1	2	2	3	3	0



DIAMETRO

Ahora ejecutamos el mismo DFS, pero ahora partiendo del nodo elegido. Para el ejemplo digamos que elegimos 6

Nodo	0	1	2	3	4	5	6
Distancia	3	1	2	2	3	3	0

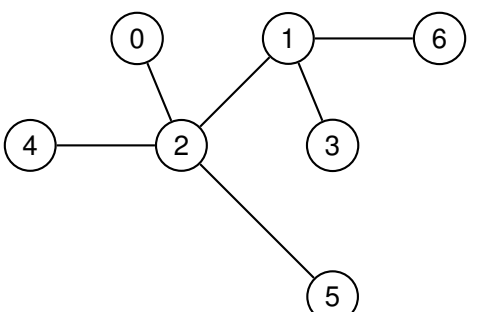


El camino más largo es 3 y va desde 6 hasta 0, 4 o 5.

DIAMETRO

Ahora ejecutamos el mismo DFS, pero ahora partiendo del nodo elegido. Para el ejemplo digamos que elegimos 6

Nodo	0	1	2	3	4	5	6
Distancia	3	1	2	2	3	3	0



El camino más largo es 3 y va desde 6 hasta 0, 4 o 5.
Complejidad: $O(n + a)$

Código

```cpp
// n cantidad de nodos
// a cantidad de aristas
int n, v;
// true es para visitado , false es para no visitado
vector< bool > visitado;
vector< vector < int > > lista;
vector< int > dists;

void dfs(int u) {
  visitado[u] = true;
  for(int i = 0; i < lista[u].size(); i++) {
    int v = lista[u][i];
    if(visitado[v] == false){
      dists[v] = dists[u] + 1;
      dfs(v);
    }
  }
}
```

Código

```cpp
int main() {
  cin >> n >> v;
  lista.resize(n);
  visitado.resize(n, false);
  dists.resize(n, 0);

  for(int i = 0; i < v; i++) {
    int a, b; cin >> a >> b;
    lista[a].push_back(b);
    lista[b].push_back(a);
  }

  dfs(0);
  int max_node = 0;
  for(int i = 0; i < n; i++) {
    if(dists[max_node] < dists[i]) max_node = i;
  }

  visitado.resize(n, false); dists.resize(n, 0);
  dfs(max_node);
  int max_dist = 0;
  for(int i = 0; i < n; i++) {
    if(max_dist < dists[i]) max_dist = dists[i];
  }

  // hacer lo que quieran con esa info
}
```

REFERENCES I