TALLER PROGCOMP: TRACK GRAFOS

LOWEST COMMON ANCESTOR

Gabriel Carmona Tabja

Universidad Técnica Federico Santa María,
Università di Pisa

November 3, 2024

Part I

Ordered Tree

Ordered Tree

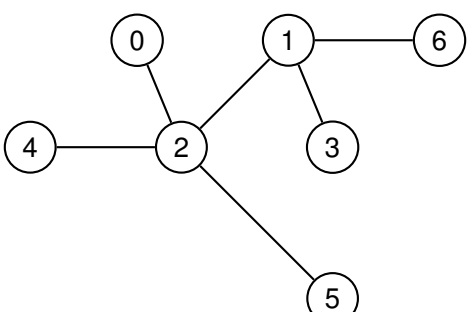
Definición

Es un árbol, pero que tiene un nodo designado como ráiz y existe un orden de los hijos de cada nodo.

Ordered Tree

Definición

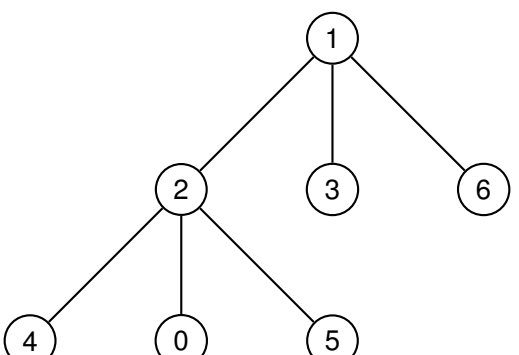
Es un árbol, pero que tiene un nodo designado como ráiz y existe un orden de los hijos de cada nodo.



Ordered Tree

Definición

Es un árbol, pero que tiene un nodo designado como ráiz y existe un orden de los hijos de cada nodo.



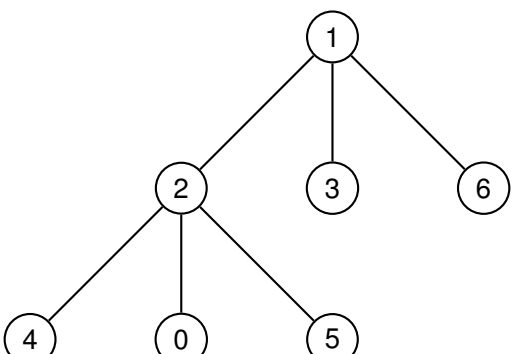
Part II

Lowest Common Ancestor

Lowest Common Ancestor

Definición

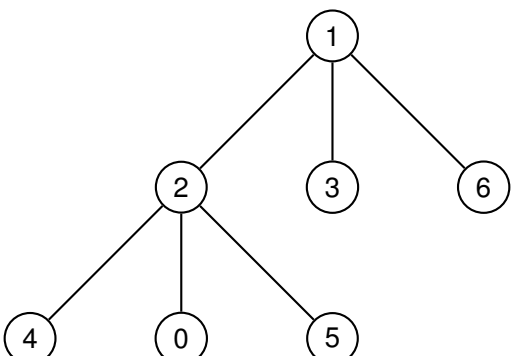
Dado dos vértices *a* y *b* de un *ordered tree*, se quiere determinar el menor ancestro en común.



Lowest Common Ancestor

Definición

Dado dos vértices *a* y *b* de un *ordered tree*, se quiere determinar el menor ancestro en común.

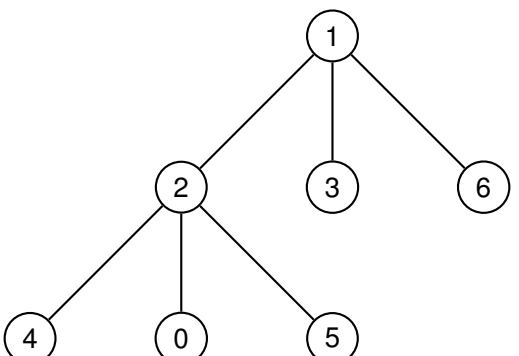


- $LCA(4, 5) = 2$

Lowest Common Ancestor

Definición

Dado dos vértices *a* y *b* de un *ordered tree*, se quiere determinar el menor ancestro en común.

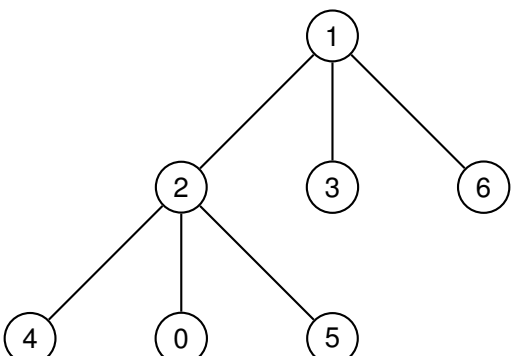


- $LCA(4,5) = 2$
- $LCA(0,3) = 1$

Lowest Common Ancestor

Definición

Dado dos vértices *a* y *b* de un *ordered tree*, se quiere determinar el menor ancestro en común.



- $LCA(4, 5) = 2$
- $LCA(0, 3) = 1$

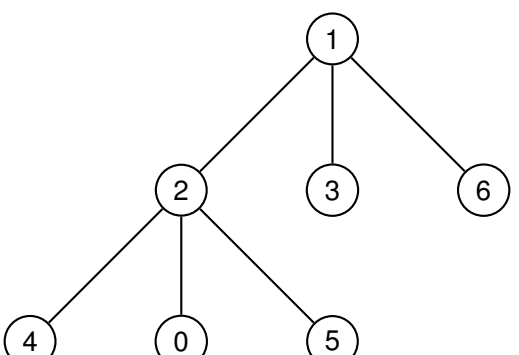
¿Cómo resolverlo eficientemente?

Euler Tour

1. Partiendo de la raiz, se realiza un DFS en el árbol, se agrega a un arreglo el nodo al momento de “entrar al nodo” en el DFS.

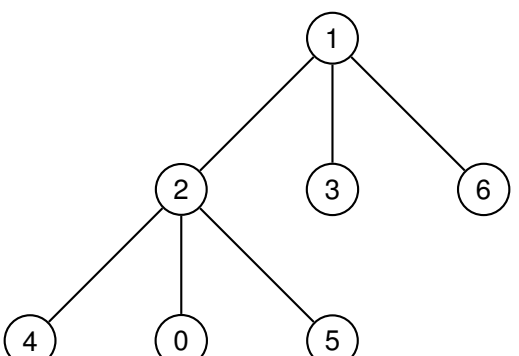
Euler Tour

1. Partiendo de la raiz, se realiza un DFS en el árbol, se agrega a un arreglo el nodo al momento de “entrar al nodo” en el DFS.
2. Se guarda la altura del nodo al momento de “entrar al nodo” en el DFS.



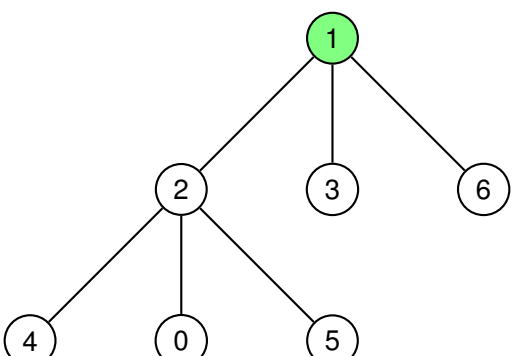
Euler Tour

Euler Tour	[]
Altura	[]



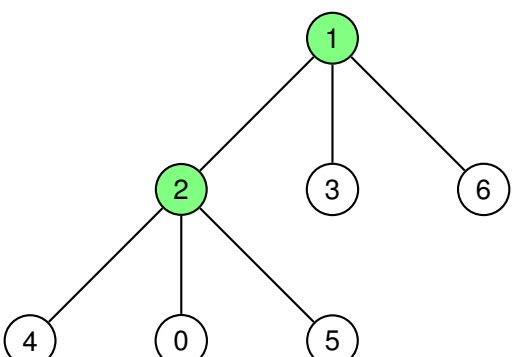
Euler Tour

Euler Tour	[1]
Altura	[0]



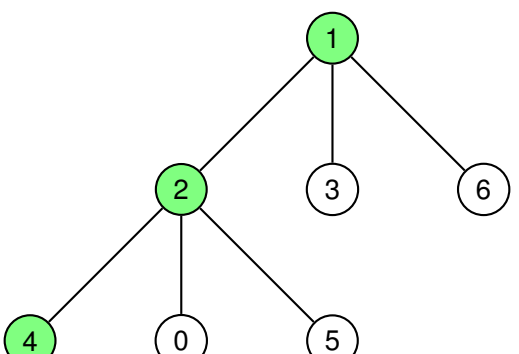
EULER TOUR

Euler Tour	[1, 2]
Altura	[0, 1]



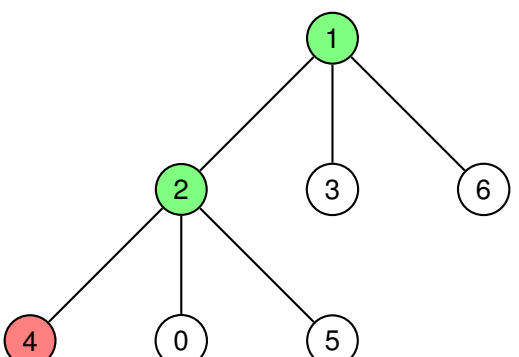
Euler Tour

Euler Tour	$[1, 2, 4]$
Altura	$[0, 1, 2]$



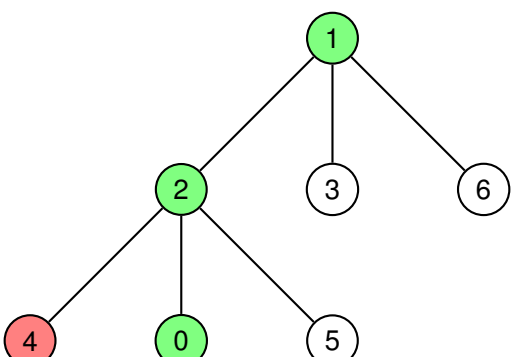
Euler Tour

Euler Tour	$[1, 2, 4, 2]$
Altura	$[0, 1, 2, 1]$



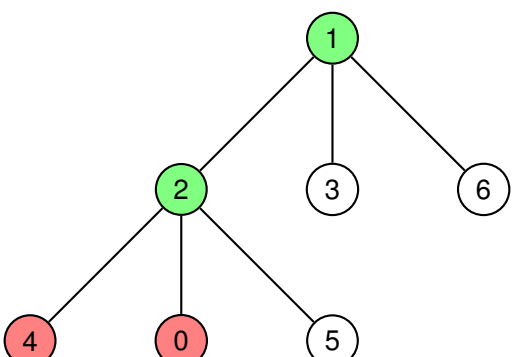
Euler Tour

Euler Tour	$[1, 2, 4, 2, 0]$
Altura	$[0, 1, 2, 1, 2]$



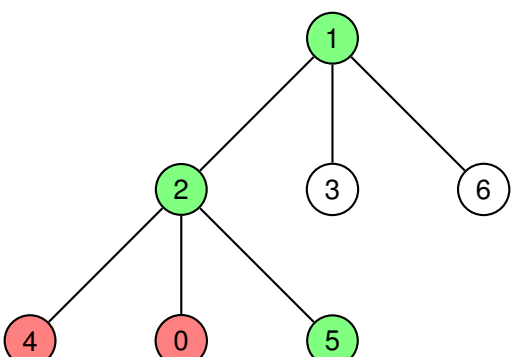
Euler Tour

Euler Tour	$[1, 2, 4, 2, 0, 2]$
Altura	$[0, 1, 2, 1, 2, 1]$



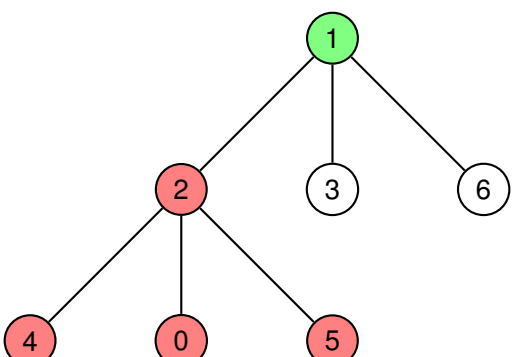
Euler Tour

Euler Tour	$[1, 2, 4, 2, 0, 2, 5]$
Altura	$[0, 1, 2, 1, 2, 1, 2]$



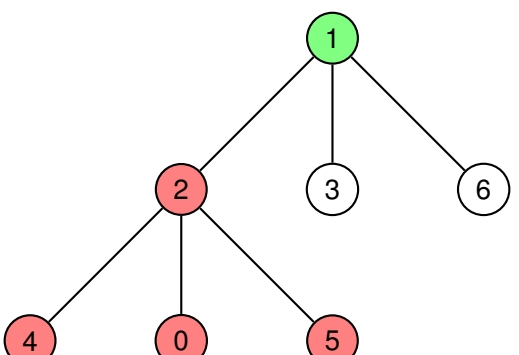
Euler Tour

Euler Tour	$[1, 2, 4, 2, 0, 2, 5, 2]$
Altura	$[0, 1, 2, 1, 2, 1, 2, 1]$



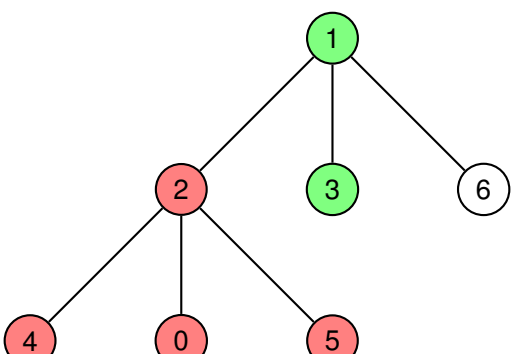
Euler Tour

Euler Tour	$[1, 2, 4, 2, 0, 2, 5, 2, 1]$
Altura	$[0, 1, 2, 1, 2, 1, 2, 1, 0]$



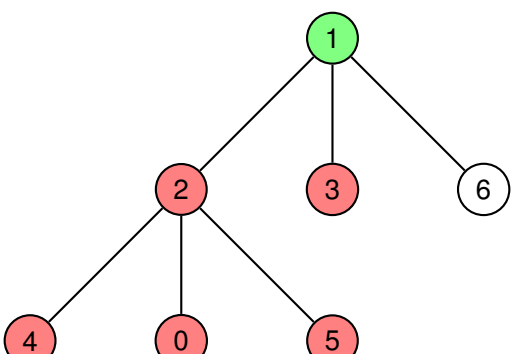
Euler Tour

Euler Tour	$[1, 2, 4, 2, 0, 2, 5, 2, 1, 3]$
Altura	$[0, 1, 2, 1, 2, 1, 2, 1, 0, 1]$



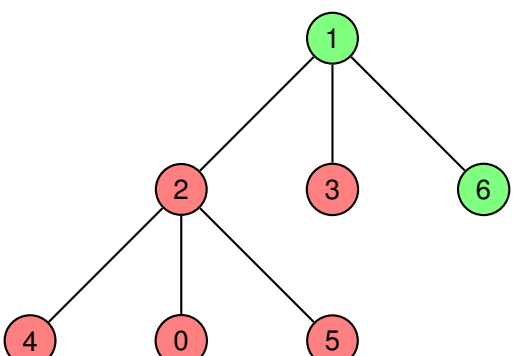
Euler Tour

Euler Tour	$[1, 2, 4, 2, 0, 2, 5, 2, 1, 3, 1]$
Altura	$[0, 1, 2, 1, 2, 1, 2, 1, 0, 1, 0]$



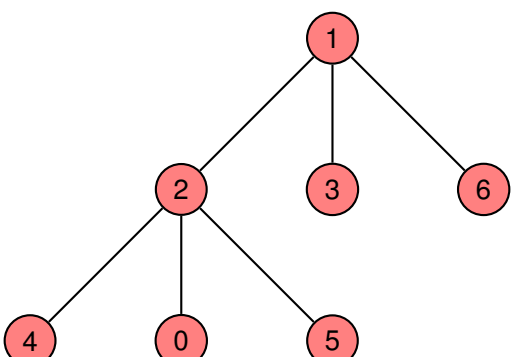
Euler Tour

Euler Tour	$[1, 2, 4, 2, 0, 2, 5, 2, 1, 3, 1, 6]$
Altura	$[0, 1, 2, 1, 2, 1, 2, 1, 0, 1, 0, 1]$



Euler Tour

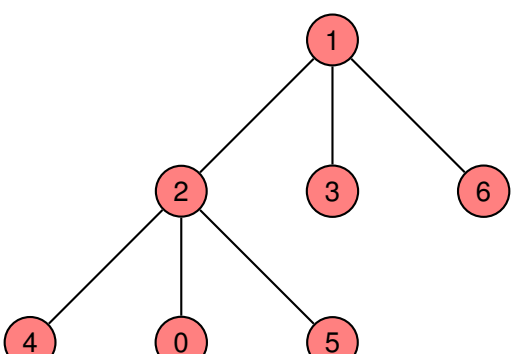
Euler Tour	$[1, 2, 4, 2, 0, 2, 5, 2, 1, 3, 1, 6, 1]$
Altura	$[0, 1, 2, 1, 2, 1, 2, 1, 0, 1, 0, 1, 0]$



Query sobre Euler Tour

¿$LCA(0, 3)$?

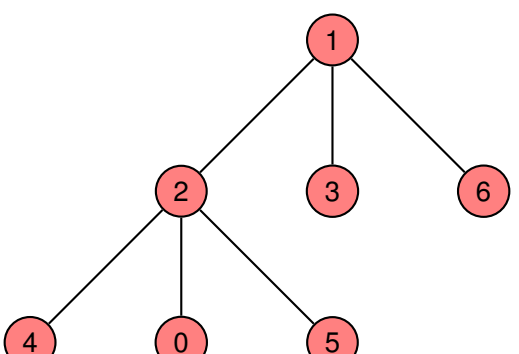
Euler Tour	$[1, 2, 4, 2, 0, 2, 5, 2, 1, 3, 1, 6, 1]$
Altura	$[0, 1, 2, 1, 2, 1, 2, 1, 0, 1, 0, 1, 0]$



Query sobre Euler Tour

¿$LCA(0, 3)$?

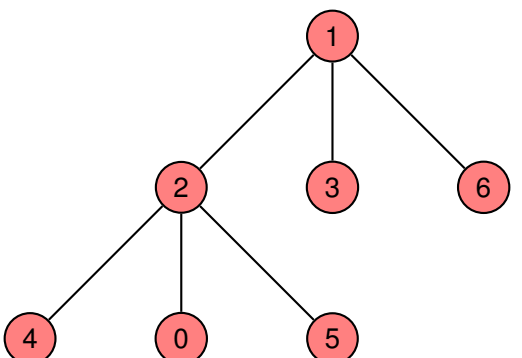
Euler Tour	$[1, 2, 4, 2, 0, 2, 5, 2, 1, 3, 1, 6, 1]$
Altura	$[0, 1, 2, 1, 2, 1, 2, 1, 0, 1, 0, 1, 0]$



Query sobre Euler Tour

¿$LCA(0, 3)$?

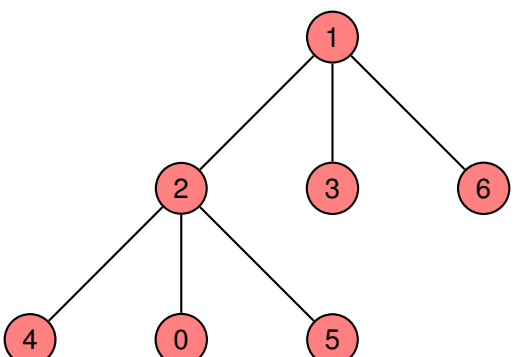
Euler Tour	$[1, 2, 4, 2, 0, 2, 5, 2, 1, 3, 1, 6, 1]$
Altura	$[0, 1, 2, 1, 2, 1, 2, 1, 0, 1, 0, 1, 0]$



Query sobre Euler Tour

¿$LCA(0, 3)$?

Euler Tour	$[1, 2, 4, 2, 0, 2, 5, 2, 1, 3, 1, 6, 1]$
Altura	$[0, 1, 2, 1, 2, 1, 2, 1, 0, 1, 0, 1, 0]$

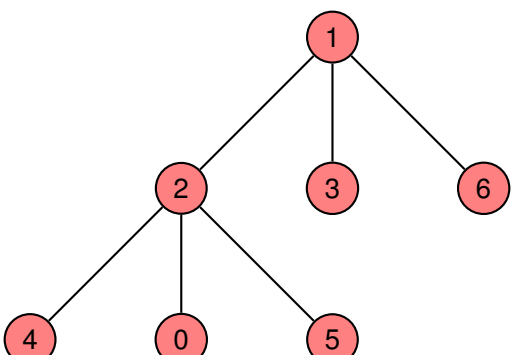


Complejidad: $O(n + a) + O(lca(a, b))$

QUERY SOBRE EULER TOUR

¿$LCA(0, 3)$?

Euler Tour	$[1, 2, 4, 2, 0, 2, 5, 2, 1, 3, 1, 6, 1]$
Altura	$[0, 1, 2, 1, 2, 1, 2, 1, 0, 1, 0, 1, 0]$



Complejidad: $O(n + a) + O(lca(a, b))$

- SQRT Decompositon: $O(\sqrt{n})$ con $O(n)$ en construcción
- Segment Tree/ Fenwick Tree: $O(\log n)$ con $O(n)$ en construcción
- Sparse Table: $O(1)$ con $O(n \log n)$ en construcción

Código

```cpp
int n, v;
vector< bool > visitado;
vector< vector < int > > lista;
// cuando aparece por primera vez el nodo i
vector< int > first;
// euler tour
vector< int > alts;
vector< int > euler;

void dfs(int u, int alt) {
  visitado[u] = true;
  alts.push_back(alt);
  euler.push_back(u);
  first[u] = euler.size() - 1;
  for(int i = 0; i < lista[u].size(); i++) {
    int v = lista[u][i];
    if(visitado[v] == false){
      dfs(v, alt + 1);
      alts.push_back(alt);
      euler.push_back(u);
    }
  }
}
```

Código usando Segment Tree

```cpp
// nodo de segment tree
// first -> cual nodo del arbol tiene minima altura en el rango
// second -> la altura de ese nodo del arbol
pair< int, int > merge(pair< int, int > a, pair< int, int > b) {
  if(a.second < b.second) return a;
  return b;
}

int main() {
  // se asume que generaron el arbol correctamente

  dfs(root); // root es el nodo raiz del arbol
  vector< pair< int, int > > euler_alts;
  for(int i = 0; i < euler.size(); i++) {
    euler_alt.push_back({euler[i], alts[i]});
  }
  segment_tree< pair< int, int >, merge > st(euler_alts);

  int a, b;
  cin >> a >> b;
  if(first[a] > first[b]) swap(a, b);
  cout << st.query(first[a], first[b]).first << "\n";
}
```

REFERENCES I