TALLER PROGCOMP: TRACK GRAFOS

TOPOLOGICAL SORT

Gabriel Carmona Tabja

Universidad Técnica Federico Santa María,
Università di Pisa

October 27, 2024

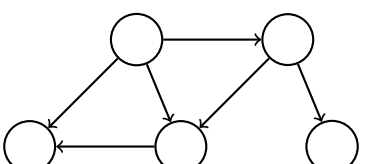
Part I

TOPOLOGICAL SORT

TOPOLOGICAL SORT

Definición

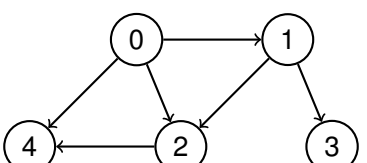
En un grafo digirido sin ciclos se quiere determinar un **orden para los vertices**.



TOPOLOGICAL SORT

Definición

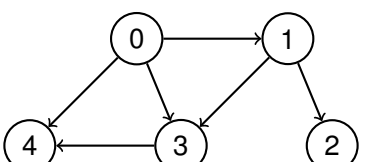
En un grafo digirido sin ciclos se quiere determinar un **orden para los vertices**.



TOPOLOGICAL SORT

Definición

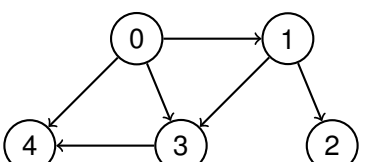
En un grafo digirido sin ciclos se quiere determinar un **orden para los vertices**.



TOPOLOGICAL SORT

Definición

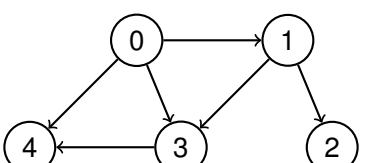
En un grafo digirido sin ciclos se quiere determinar un **orden para los vertices**.



¡Hagamos un algoritmo! (el grafo no debe tener ciclos)

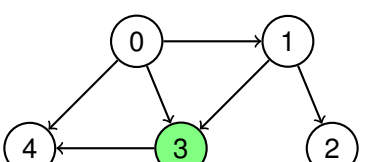
ALGORITMO

{}



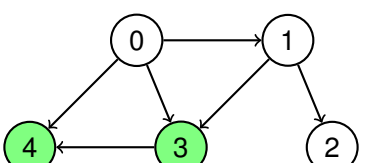
ALGORITMO

{}



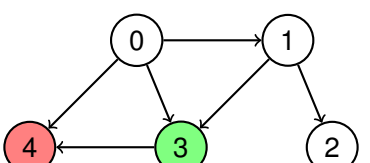
ALGORITMO

{}



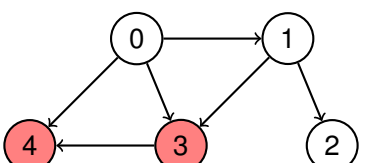
ALGORITMO

{4}



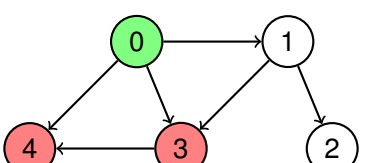
ALGORITMO

{4, 3}



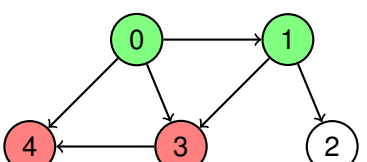
ALGORITMO

{4, 3}



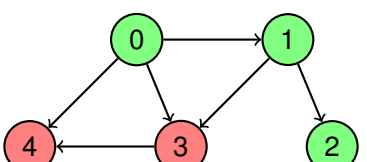
ALGORITMO

{4, 3}



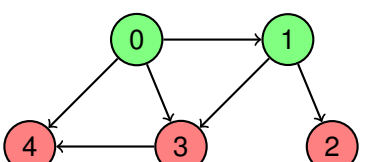
ALGORITMO

{4, 3}



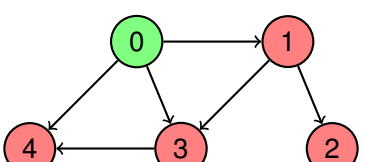
ALGORITMO

{4, 3, 2}



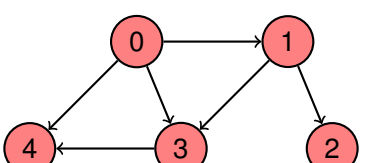
ALGORITMO

{4, 3, 2, 1}



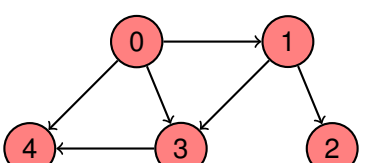
ALGORITMO

{4, 3, 2, 1, 0}



ALGORITMO

{0, 1, 2, 3, 4}



Código

```cpp
int n; // numero de vertices
vector< vector<int> > lista;
vector< bool > visitado;
vector< int > res;

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){
      dfs(v);
    }
  }
  res.push_back(u);
}

void top_sort() {
  for(int i = 0; i < n; i++) {
    if(visitado[i] == false) {
      dfs(i);
    }
  }
  reverse(res.begin(), res.end());
}
```

Utilidades y Complejidad

- Complejidad: $O(n + a)$

Utilidades y Complejidad

- Complejidad: $O(n + a)$
- Obtener dependencias

Utilidades y Complejidad

- Complejidad: $O(n + a)$
- Obtener dependencias
- Si el grafo es dirigido con pesos sin ciclos: calcular camino máximo/mínimo complejidad $O(n + a)$

Utilidades y Complejidad

- Complejidad: $O(n + a)$
- Obtener dependencias
- Si el grafo es dirigido con pesos sin ciclos: calcular camino máximo/mínimo complejidad $O(n + a)$
- ¡Algoritmo de Kosaraju! ¡Obtener componentes fuertemente conexa!

REFERENCES I