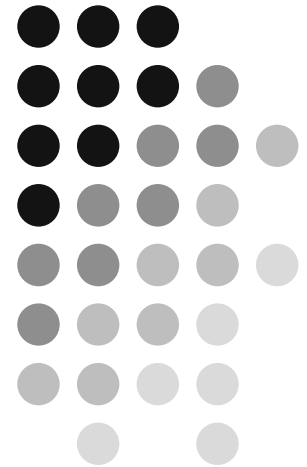


Grafos: ordenamiento topológico y conectividad

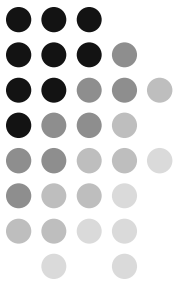


UNIVERSIDAD
DE LOS ANDES

Diseño y Análisis de Algoritmos
Cátedra de Programación
Carrera de Ingeniería de Sistemas
Prof. Isabel Besembel Carrera

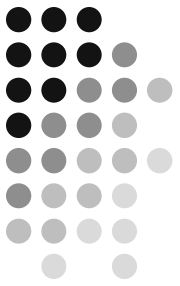


Orden parcial

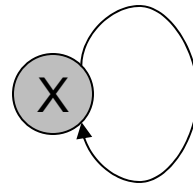


- Un orden parcial es una relación reflexiva, antisimétrica y transitiva.
- **Dominio:** es un conjunto de valores
Ejm: $D1 = \{\text{'rojo'}, \text{'verde'}, \text{'negro'}, \text{'azul'}\}$
 $D2 = \{\text{'ford'}, \text{'chevrolet'}, \text{'fiat'}, \text{'toyota'}, \text{'renault'}\}$
- **Relación:** es un subconjunto del producto cartesiano de una lista de dominios, no necesariamente disjuntos.
Ejm: $R1 = \{(\text{'rojo'}, \text{'ford'}), (\text{'verde'}, \text{'ford'}), (\text{'negro'}, \text{'chevrolet'}), (\text{'azul'}, \text{'toyota'})\}$
 $R2 = \{(\text{'fiat'}, \text{'verde'})\}$
 $R3 = \{ \}$

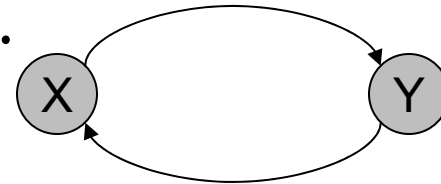
Propiedades de las relaciones



- Reflexividad: Una relación R es reflexiva si $X R X$ para todo X en S .



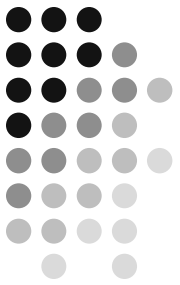
- Simetría: Una relación R es simétrica si $X R Y$ implica $Y R X$ para todo X y Y .



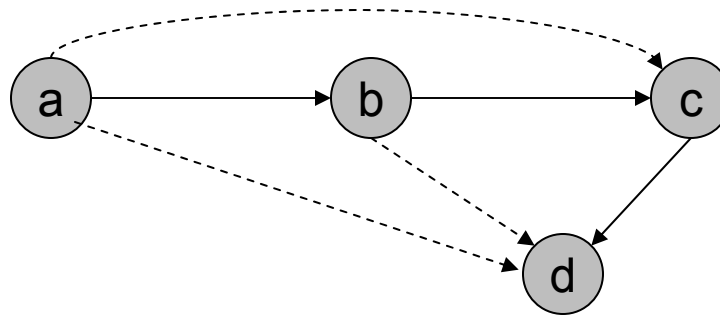
- Antisimetría: Una relación R es antisimétrica, si $X R Y$ y $Y R X$ implica $X = Y$, para todo X y Y .

La relación \leq es antisimétrica, $x \leq p$ y $p \leq x$ implica que $x=p$

Propiedades de las relaciones

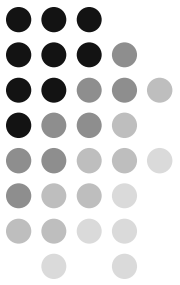


- Transitividad: Una relación R es transitiva si $X R Y$ y $Y R Z$ implica que $X R Z$, para todo X, Y, Z .

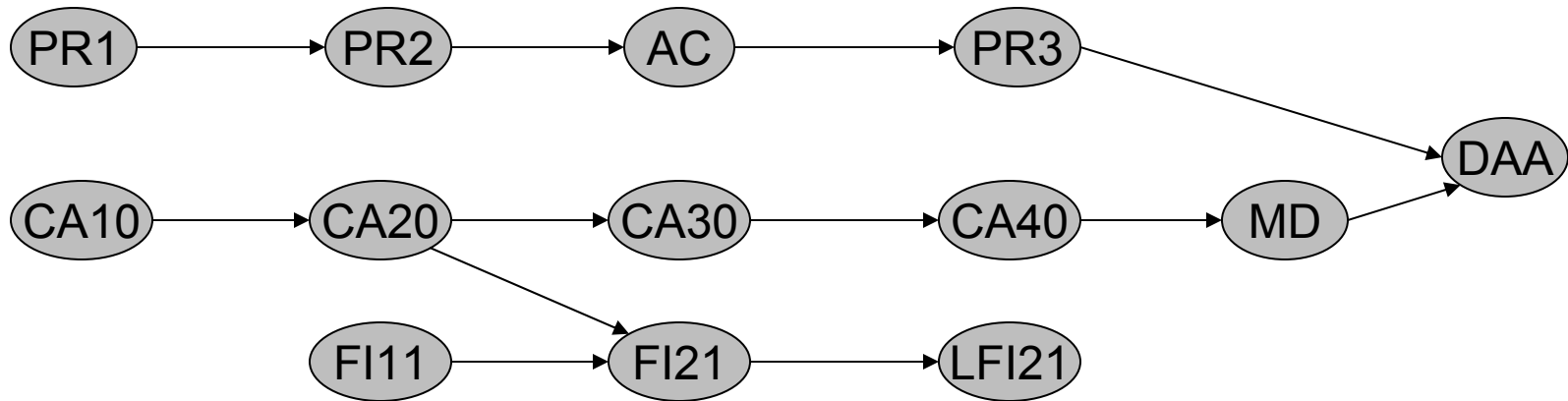
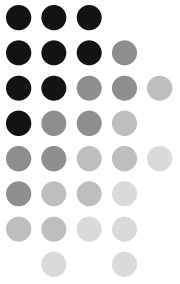


- La importancia de la teoría de las relaciones es que está relacionada con cualquier tipo de grafo.
- Un orden parcial (toda relación antisimétrica y transitiva) tiene un digrafo acíclico

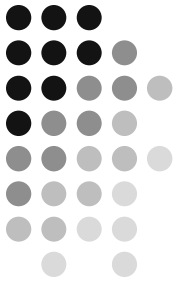
Ordenamiento topológico



- Para un digrafo acíclico (dag) $G = (\mathbb{N}, A)$ el orden lineal de todos los nodos tal que si G contiene el arco (u, v) , entonces u aparece antes de v en el orden dado.
- Si G tiene ciclos el orden lineal no es posible.
- Ordenamiento topológico de todos los nodos de un digrafo es una manera de visitar todos sus nodos, uno por uno, en una secuencia que satisface una restricción dada.



- Un orden topológico de los nodos de un digrafo $G = \{N, A\}$ es una secuencia (n_1, n_2, \dots, n_k) tal que $N = \{n_1, n_2, \dots, n_k\}$ y para todo (n_i, n_j) en N , n_i precede a n_j en la secuencia.

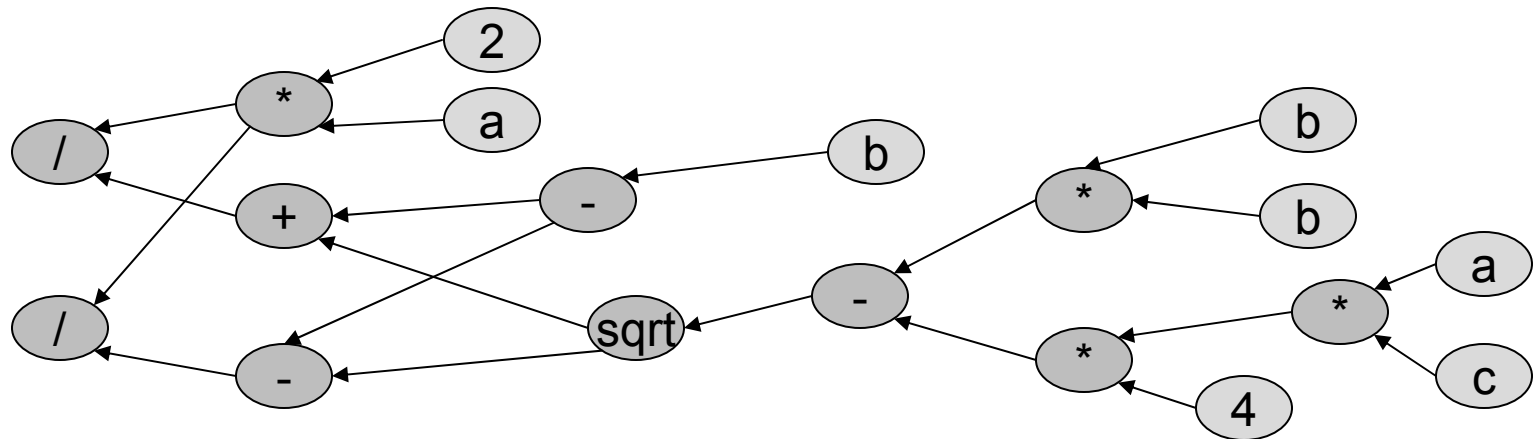


- Modelado de actividades para resolver problemas de planificación o programación de las mismas siguiendo un criterio de minimización o maximización.

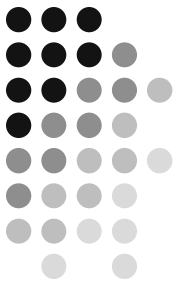
- Evaluación de expresiones aritméticas

$$(-b + \sqrt{b * b - 4 * a * c}) / 2 * a$$

$$(-b - \sqrt{b * b - 4 * a * c}) / 2 * a$$



Teorema (Szpilrajn, 1930)



- En cualquier grafo G con $n \geq 1$ nodos hay un nodo con grado de incidencia negativo (g_{in}) igual a cero

Demostración: por contradicción

Si todos los nodos del digrafo G tienen un g_{in} de al menos 1, entonces G contiene ciclos

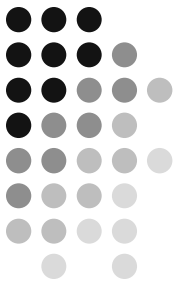
Asuma que todos los nodos tienen un g_{in} de al menos 1

Se comienza con cualquier nodo n_1 , se traza hacia atrás a lo largo de cualquier arista que incide en n_2 .

Desde n_2 hasta n_3 y así sucesivamente

Como todos los nodos tienen un g_{in} de al menos 1, este proceso nunca acaba, pero como G es finito, eventualmente un nodo debe ser alcanzado cuando ya se ha pasado por él, por lo que G tiene ciclo

Algoritmo genérico de ordenamiento topológico



Abril/05

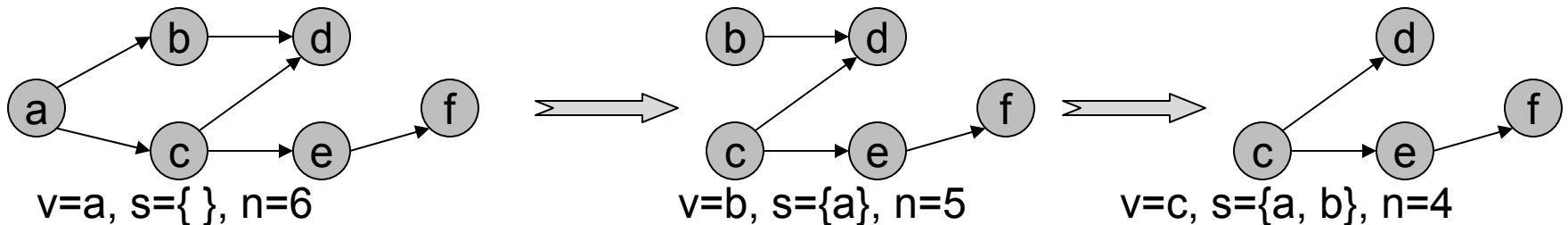
ordenTopologico(): Lista

{pre: $n > 0$ }

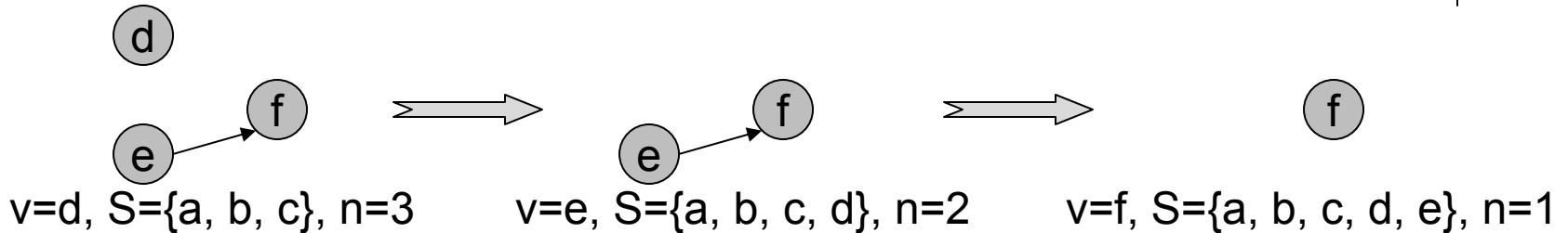
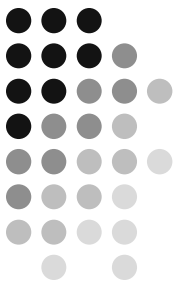
{pos: $n = 0$ }

1 ($n > 0$) [$v = \text{nodo con } \text{gin} = 0$
elimine v y todas las aristas que salen de v
 $s.\text{inserta}(v)$]
2 regrese s

- v : Entero. Nodo del digrafo.
- $\text{inserta}()$. Definida en Lista.
- s . Lista. Lista que contiene el orden topológico de los nodos del grafo.



Algoritmo genérico de ordenamiento topológico



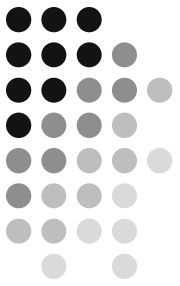
- Al finalizar $S = \{a, b, c, d, e, f\}$ con $n=0$
- Teorema de la invariante del lazo: Sea un digrafo $G = \{N, A\}$, entonces al comienzo de la k -ésima iteración del $\text{ordenTopologico}()$, $S = (n_1, n_2, \dots, n_{k-1})$ tiene la propiedad siguiente: para todo arco (n_i, n_j) en A , si n_j está en S , entonces n_i precede a n_j en S .

Demostración: por inducción sobre k

Etapa base: $k=1, S_1 = \{\}$, el teorema es cierto

Etapa inductiva: hipótesis: Sea $S_k = \{n_1, n_2, \dots, n_{k-1}\}$ el estado al comienzo de la k -ésima iteración y sea G_k el estado de G .

Algoritmo genérico de ordenamiento topológico



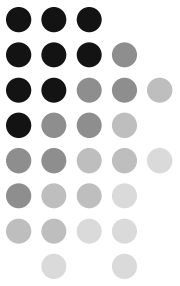
Por la hipótesis $S_{k+1} = \{n_1, n_2, \dots, n_k\}$ donde n_k es el nodo seleccionado por el algoritmo durante la k -ésima iteración.

Si n_k existe, el teorema de Szpilrajn puede ser aplicado al dag G_k , como n_k tiene $g_{in}=0$ en G_k , en el digrafo original G , todos los predecesores de n_k están en $\{n_1, n_2, \dots, n_{k-1}\}$, lo que indica que para todo (n_i, n_j) en A , n_i precede a n_k en S_{k+1} .

Por lo que el teorema es cierto para $S_{k+1} = \{n_1, n_2, \dots, n_k\}$

- La invariante del lazo implica que S_{n+1} es un orden topológico de los nodos de g y es correcto
- Todos los nodos de cualquier dag pueden ser ordenados topológicamente

Algoritmo de ordenamiento topológico



26/11/98

ordenTopologico(): ListaDe [Entero]

{pre: $n > 0$ }

{pos: $n > 0 \wedge G' = G \wedge \text{padre} \supset \text{bosque en profundidad}$ }

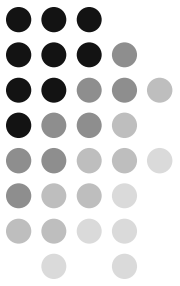
```

1 [ color(u), padre(u) = 'blanco', 0 ] u ∈ N
2 t = 0
3 [ Si ( color(u) = 'blanco' ) entonces
   visitaBusPro(u, t, color, padre, d, f, lot)
   fsi ] u ∈ N
4 regrese lot

```

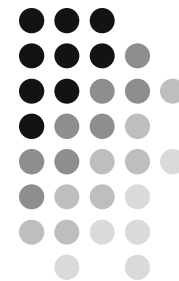
-u, t: Entero. Variable auxiliar y contador.
-color, padre, d, f: Arreglo[n] De [Entero].
 Contienen el color del nodo, el nodo predecesor inmediato, la etiqueta donde se empezó a procesar el nodo y donde se terminó de procesar cada nodo.
-visitaBusPro(). Rutina recursiva para hacer la búsqueda en profundidad.
-lot. ListaDe [Entero]. Lista que contiene el orden topológico de los nodos del grafo.

Digrafo acíclico



- Lema 12: Un grafo G es acíclico si y solo si la búsqueda en profundidad de G no arroja arcos hacia atrás (arcos que conectan un nodo u a uno de sus ancestros v)

Algoritmo de búsqueda en profundidad



26/11/98

visitaBusPro(Entero: u, t, Arreglo[n]De [Entero]: color, padre, d, f, ListaDe [Entero]: lot)

{pre: $n > 0 \wedge u \in N$ }

{pos: $n > 0 \wedge G' = G \wedge \text{padre} \supset \text{bosque en profundidad}$ }

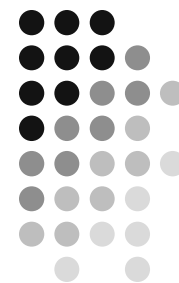
```

1  color(u), t, d(u) = 'gris', t + 1, t
2  [ Si ( color(v) = 'blanco' ) entonces
      padre(v) = u
      visitaBusPro(u, t, color, padre, d, f, lot)
    fsi ]  $\forall v \in u.\text{listaAdyacencia}$ 
3  color(u), t, f(u) = 'negro', t + 1, t
4  lot.insLista(u)
5  regrese
  
```

-u, v, t: Entero. Variables auxiliares y contador.
-color, padre, d, f: Arreglo[n] De [Entero].
 Contienen el color del nodo, el nodo predecesor inmediato, la etiqueta donde se empezó a procesar el nodo y donde se terminó de procesar cada nodo.
-insLista(). Función de la clase ListaDe [X].

$$T(n) = \Theta(N + A)$$

Componentes fuertemente conexas



- Un componente es fuertemente conexo en un digrafo $G = (\mathbf{N}, A)$ si el máximo conjunto de sus nodos $U \subseteq \mathbf{N}$ tal que para cada par de nodos u y v en U , se tiene un camino desde u hasta v y viceversa, esto es u y v son alcanzables de uno a otro.
- Cálculo de los componentes fuertemente conexos de G : se realizan dos búsquedas en profundidad, la primera sobre G y la segunda sobre GT
- `componentesFuertementeConexas()`
 1. `G.busProf()`
 2. `GT = G.transpuesto()`
 3. `GT.busProf()`, pero en el lazo principal de `busProf()` considerar los nodos en orden decreciente de su $f(u)$
 4. Desplegar los nodos de cada árbol en profundidad de GT como una componente conexa para G .
- $T(n) = \Theta(\mathbf{N} + A)$