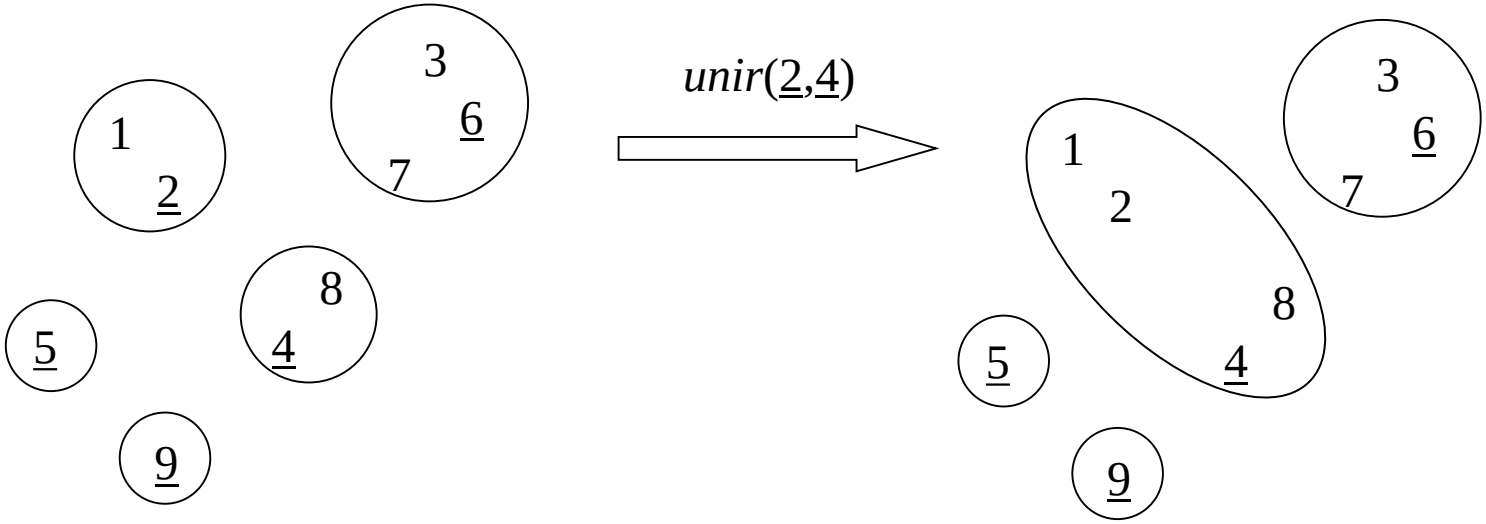
Estructuras de conjuntos disjuntos

- Conectividad en redes.
- Percolación (flujo de un líquido a través de un medio poroso).
- Procesamiento de imágenes.
- Antecesor común más próximo (en un árbol).
- Equivalencia de autómatas de estados finitos.
- Inferencia de tipos polimórficos o tipado implícito (algoritmo de Hinley-Milner).
- Árbol de recubrimiento mínimo (algoritmo de Kruskal).
- Juego (*Go*, *Hex*).
- Compilación de la instrucción EQUIVALENCE en Fortran.
- Mantenimiento de listas de copias duplicadas de páginas web.
- Diseño de VLSI.

Estructuras de conjuntos disjuntos

- Mantener una partición de S={1,2,…,n} con las operaciones de



encontrar(3)=6

Estructuras de conjuntos disjuntos

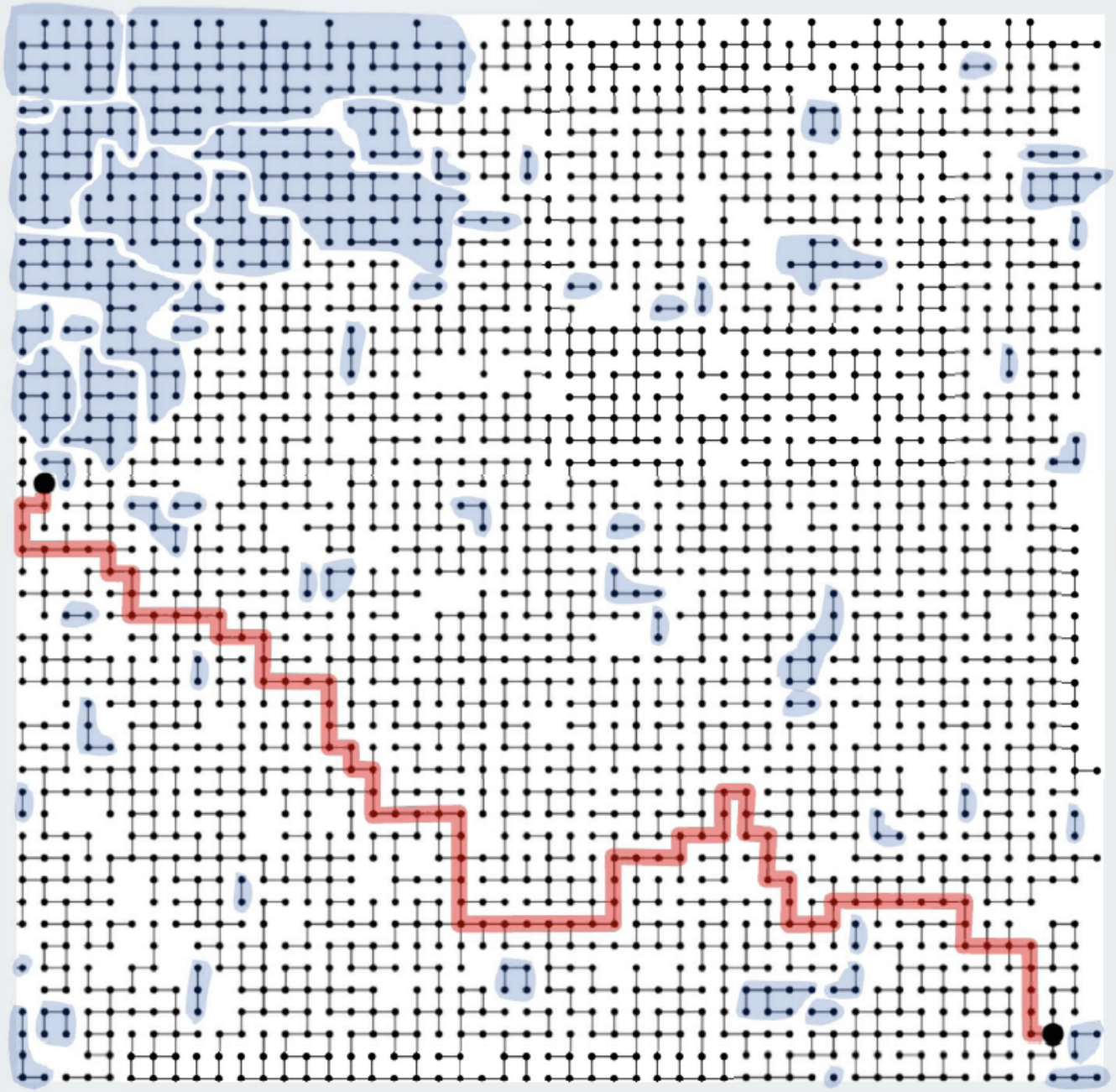
- Ejemplo sencillo de aplicación: calcular las componentes conexas de un grafo no dirigido

```
procedimiento componentes_conexas(g)
principio
  para todo v vértice de g hacer
    crear(v)
  fpara;
  para toda (u,v) arista de g hacer
    si encontrar(u)≠encontrar(v) entonces
      unir(u,v)
    fsi
  fpara
fin
```

¿encontrar(u) = encontrar(v)?

¿encontrar(u) = encontrar(v)?

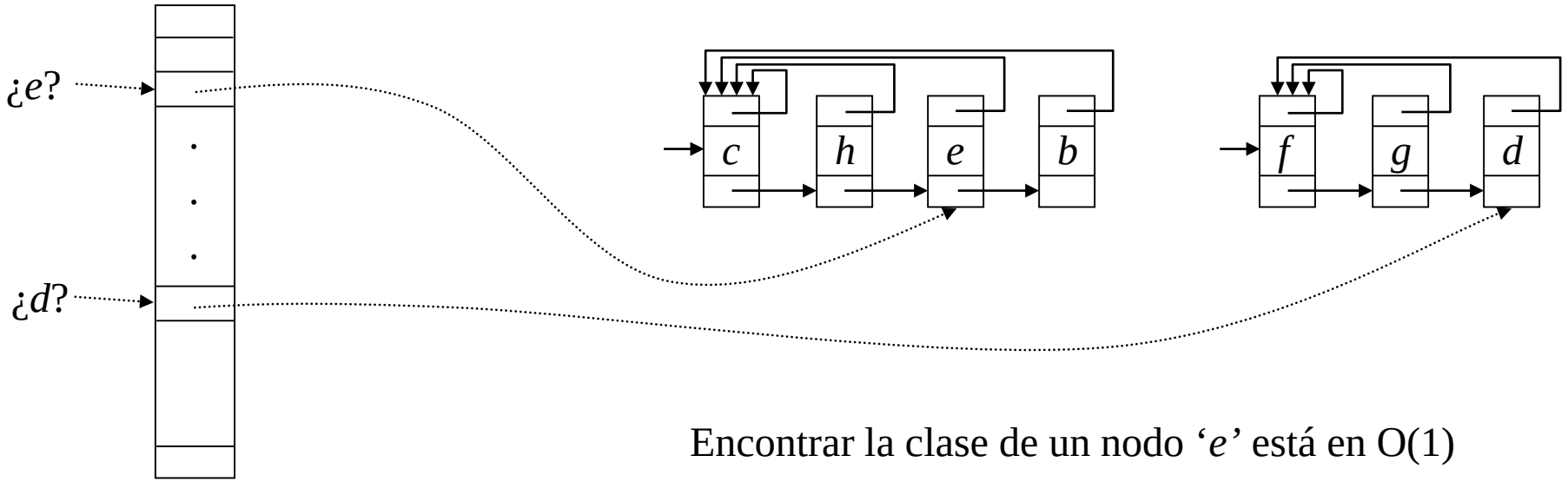
verdad



63 componentes

Estructuras de conjuntos disjuntos

- Implementación naif: listas encadenadas

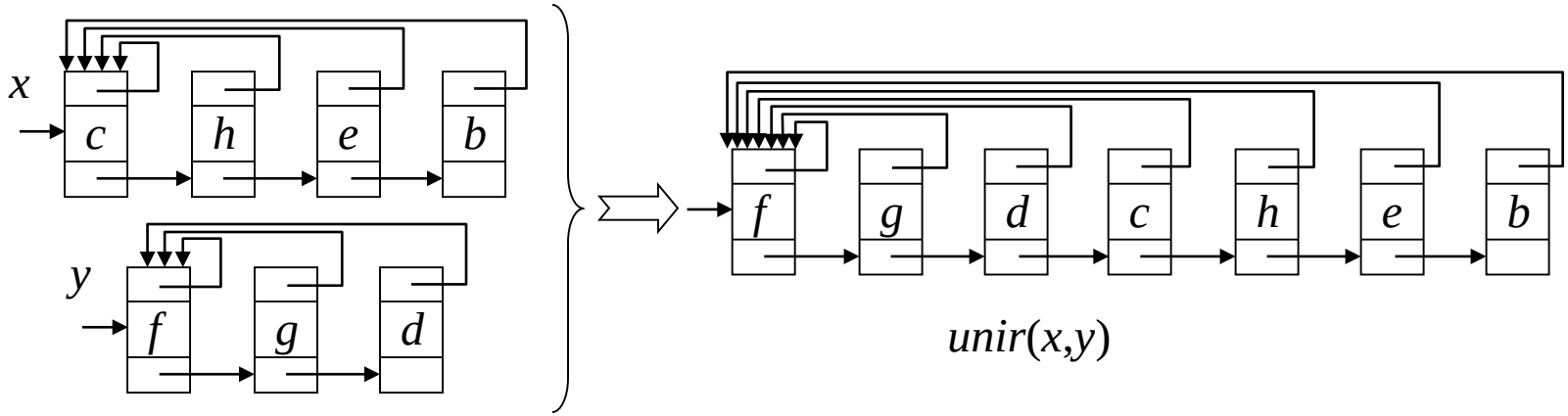


Encontrar la clase de un nodo ‘*e*’ está en O(1)

Tabla *hash* para
acceder
a cada nodo

Estructuras de conjuntos disjuntos

Implementación de *unir* (o fusionar clases):

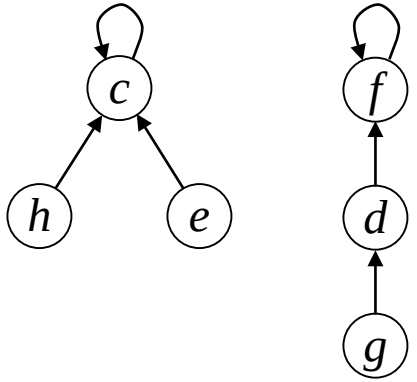


- Concatenar la primera lista tras la segunda y modificar los punteros al primero en la primera lista ➔ coste lineal en la longitud de esa lista
- Si cada lista almacena explícitamente su longitud, optar por añadir siempre la lista más corta al final de la más larga.
  - Con esta heurística sencilla el coste de una única operación sigue siendo el mismo, pero…

    Una secuencia de m operaciones con n elementos distintos cuesta $O(m + n\log n)$ en tiempo.

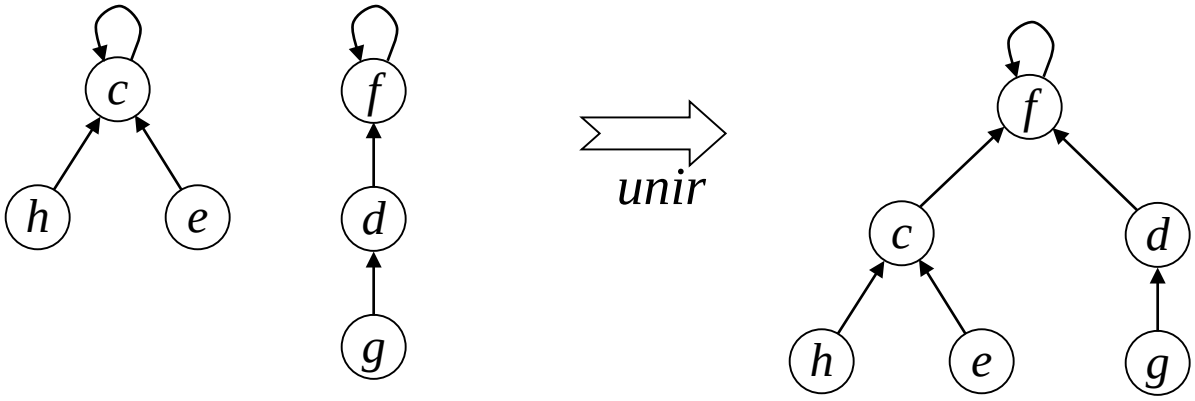
Estructuras de conjuntos disjuntos

- Buena implementación: bosque de conjuntos disjuntos
  - Conjunto de árboles, cada uno representando un conjunto disjunto de elementos.
  - Representación con puntero al padre.
  - La raíz de cada árbol es el representante y apunta a si misma.



Estructuras de conjuntos disjuntos

- La implementación “trivial” de las operaciones no mejora la eficiencia de la implementación con listas.
- Solución buena:
  - *Unir*: hacer que la raíz del árbol con menos “rango” apunte a la raíz del otro con mayor “rango” (unión por rango).



  - Al *crear* un árbol, su *rango* es 0.
  - Al *unir* dos árboles se coloca como raíz la del árbol de mayor *rango*, y éste no cambia; en caso de empate, se elige uno cualquiera y se incrementa su rango en una unidad.

Estructuras de conjuntos disjuntos

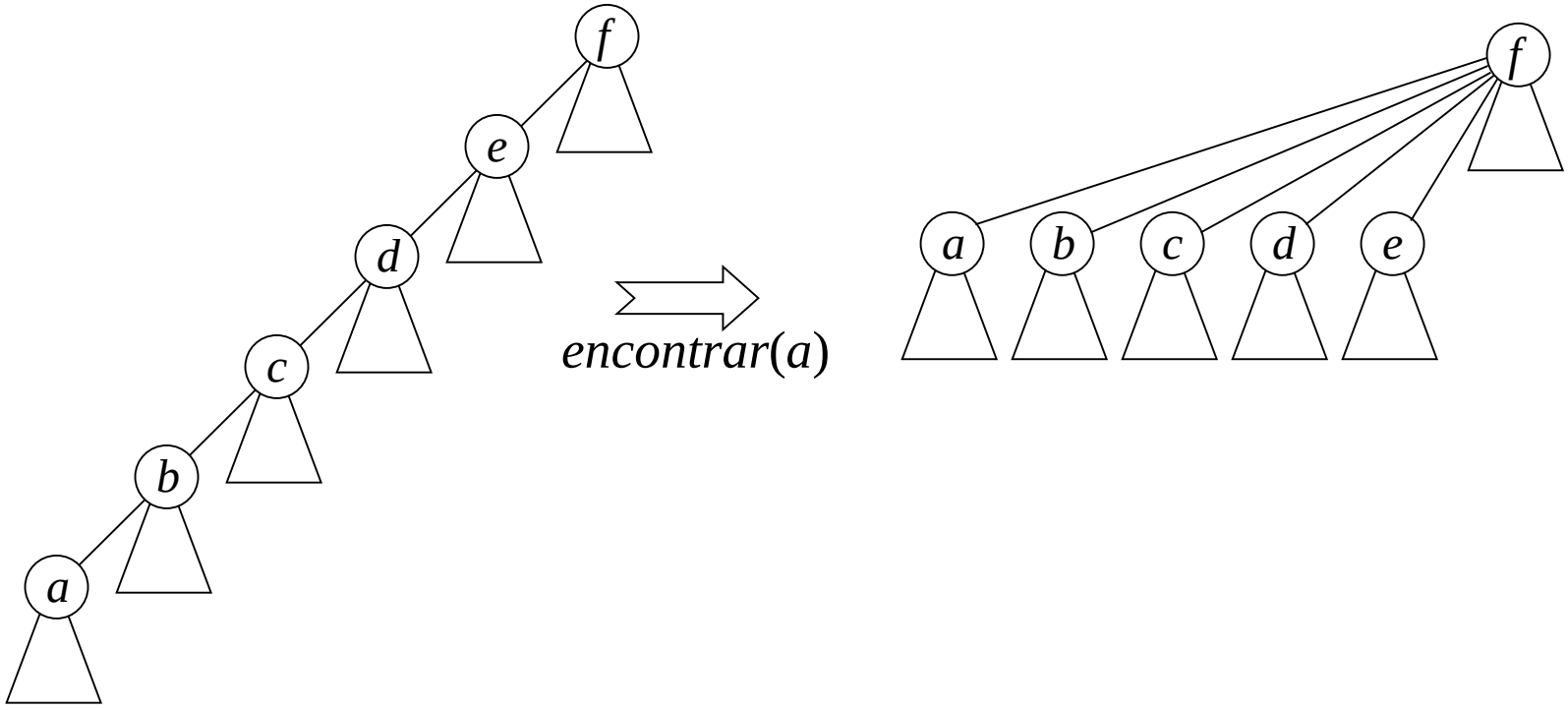
```
procedimiento crear(x)
principio
  nuevoArbol(x);
  x.padre:=x;
  x.rango:=0
fin
```

```
procedimiento unir(x,y)
principio
  enlazar(encontrar(x),encontrar(y))
fin
```

```
procedimiento enlazar(x,y)
principio
  si x.rango>y.rango ent
    y.padre:=x
  sino
    x.padre:=y;
    si x.rango=y.rango ent y.rango:=y.rango+1 fsi
  fsi
fin
```

Estructuras de conjuntos disjuntos

- *Encontrar*: heurística de “compresión de caminos”
  - los nodos recorridos en el *camino de búsqueda* pasen a apuntar directamente a la raíz
  - no se modifica la información sobre el rango



Estructuras de conjuntos disjuntos

```
función encontrar(x) devuelve puntero a nodo
principio
  si x≠x.padre ent x.padre:=encontrar(x.padre) fsi;
  devuelve x.padre
fin
```

Ojo! Modifica x

Estructuras de conjuntos disjuntos

- Coste:
  - Si se usan la unión por rango y la compresión de caminos, el coste en el caso peor para una secuencia de m operaciones con n elementos distintos es

$$O(m\ \alpha(m,n))$$

donde $\alpha(m,n)$ es una función (parecida a la inversa de la función de Ackerman) que crece **muy** despacio.

Tan despacio que en cualquier aplicación práctica que podamos imaginar se tiene que $\alpha(m,n) \leq 4$, por tanto puede interpretarse el coste como lineal en m, en la práctica.

Estructuras de conjuntos disjuntos

– Función de Ackerman, definida para enteros $i, j \geq 1$:

$A(1,j) = 2^j$, para $j \geq 1$

$A(i,1) = A(i-1,2)$, para $i \geq 2$

$A(i,j) = A(i-1,A(i,j-1))$, para $i, j \geq 2$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$
$i = 1$	2^1	2^2	2^3	2^4
$i = 2$	2^2	2^{2^2}	$2^{2^{2^2}}$	$2^{2^{2^{2^2}}}$ =…
$i = 3$	2^{2^2}	$2^{2^{\cdot^{\cdot^{\cdot^2}}}}\}16$	$2^{2^{\cdot^{\cdot^{\cdot^2}}}}\}2^{2^{\cdot^{\cdot^{\cdot^2}}}}\}16$	$2^{2^{\cdot^{\cdot^{\cdot^2}}}}\}2^{2^{\cdot^{\cdot^{\cdot^2}}}}\}2^{2^{\cdot^{\cdot^{\cdot^2}}}}\}16$

Estructuras de conjuntos disjuntos

- Función $\alpha(m,n)$:
  - Es “una especie de inversa” de la función de Ackerman (no en sentido matemático estricto, sino en cuanto a que crece tan despacio como deprisa lo hace la de Ackerman).
  - $\alpha(m,n) = \min\{i \geq 1 \mid A(i,\lfloor m/n \rfloor) > \log n\}$
  - ¿Por qué podemos suponer que siempre (en la práctica) $\alpha(m,n) \leq 4$?
    - Nótese que $\lfloor m/n \rfloor \geq 1$, porque $m \geq n$.
    - La función de Ackerman es estrictamente creciente con los dos argumentos, luego $\lfloor m/n \rfloor \geq 1 \Rightarrow A(i,\lfloor m/n \rfloor) \geq A(i,1)$, para $i \geq 1$.
    - En particular, $A(4,\lfloor m/n \rfloor) \geq A(4,1) = A(3,2) = 2^{2^{\cdot^{\cdot^{2}}}}\}16$
    - Luego sólo ocurre para n’s “enormes” que $A(4,1) \leq \log n$, y por tanto $\alpha(m,n) \leq 4$ para todos los valores “razonables” de m y n.